Approximate Distance Oracles for Planar Graphs with Subpolynomial Error Dependency

Hung Le[1]

[1]University of Massachusetts at Amherst

Abstract

Thorup [FOCS'01, JACM'04] and Klein [SODA'01] independently showed that there exists a $(1+\epsilon)$-approximate distance oracle for planar graphs with $O(n(\log n)\epsilon^{-1})$ space and $O(\epsilon^{-1})$ query time. While the dependency on n is nearly linear, the space-query product of their oracles depend *quadratically* on $1/\epsilon$. Many follow-up results either improved the space *or* the query time of the oracles while having the same, sometimes worst, dependency on $1/\epsilon$. Kawarabayashi, Sommer, and Thorup [SODA'13] were the first to improve the dependency on $1/\epsilon$ from quadratic to *nearly linear* (at the cost of $\log^*(n)$ factors). It is plausible to conjecture that the linear dependency on $1/\epsilon$ is optimal: for many known distance-related problems in planar graphs, it was proved that the dependency on $1/\epsilon$ is at least linear.

In this work, we disprove this conjecture by reducing the dependency of the space-query product on $1/\epsilon$ from linear all the way down to *subpolynomial* $(1/\epsilon)^{o(1)}$. More precisely, we construct an oracle with $O(n\log(n)(\epsilon^{-o(1)}+\log^* n))$ space and $\log^{2+o(1)}(1/\epsilon)$ query time. Our construction is the culmination of several different ideas developed over the past two decades.

Contents

1 Introduction

Computing distances is one of the most fundamental primitives in graph algorithms. Approximate distance oracle is a data structure invented specifically for this purpose. A *t-approximate distance oracle* of an edge-weighted and undirected[1] graph $G = (V, E, w)$ is a data structure that given any two vertices u and v, return an approximate distance $\hat{d}(u, v)$ such that $d_G(u, v) \leq \hat{d}(u, v) \leq t \cdot d_G(u, v)$. The breakthrough result of Thorup and Zwick [TZ05] gives a $(2k-1)$-approximate distance oracle for undirected n-vertex graphs with $O(kn^{1+1/k})$ space and $O(k)$ query time for any $k \geq 1$. However, reducing the distance error to smaller than a factor of 3 requires $\Omega(n^2)$ [TZ05] space for dense graphs. In many practical applications, it is desirable to have the distance error as close to 1 as possible. Constructing an approximate distance oracle with such an error guarantee requires exploiting specific structures of input graphs. Planarity is a natural structure that has been extensively studied for decades; it is used to model road networks on which querying distances is a central problem.

Thorup [Tho04] and Klein [Kle02] independently constructed a $(1+\epsilon)$-approximate distance oracle with $O(n \log(n)\epsilon^{-1})$ space and $O(\epsilon^{-1})$ query time for any $\epsilon \in (0, 1)$; the construction time of their oracle is $O(n \log^3(n)/\epsilon^2)$ (see Theorem 3.19 in [Tho04]). Their results triggered many follow-up papers over the past two decades; we can generally divide them into two directions. One direction assumes that ϵ is a fixed constant and aims to improve the dependency on n [WN16, LWN21], which culminated in a distance oracle with $O(n)$ space and $O(1)$ query time by Le and Wulff-Nilsen [LWN21]. However, the dependency on $1/\epsilon$ of their oracle's space-query product is $O(\epsilon^{-4})$.

Another direction focuses on reducing the quadratic dependency on $1/\epsilon$ of the space-query product. In practical applications such as logistics and planning, a reduction of 1% distance error could lead to a huge economic saving. In such scenarios, ϵ is very small and potentially depends on n. In an extreme regime, such as $\epsilon \in [1/\sqrt{n}, 1/n]$, the quadratic dependency on $1/\epsilon$ implies a quadratic dependency on n, making the oracle trivial. Even in a moderately small regime, for example, $\epsilon = 1/\text{poly}(\log n)$, the dependency on $1/\epsilon$ remains a dominating factor. Therefore, it is of both theoretical and practical interest to reduce the quadratic dependency on $1/\epsilon$.

Kawarabayashi, Klein, and Sommer [KKS11] argued that "a very low space requirement is essential", and gave a $(1+\epsilon)$-approximate oracle with truly linear space $O(n)$ and query time $O(\epsilon^{-2}\log(n))$. While the space is information-theoretically optimal, the query time is blown up by a factor $O(\epsilon^{-1}\log n)$, making the space-query product worst than that of Thorup [Tho04] and Klein [Kle02]. The first real improvement was achieved by Kawarabayashi, Sommer, and Thorup [KST13] who constructed a $(1+\epsilon)$-approximate oracle with $O(n \log n \log\log(1/\epsilon) \log^* n)$ space and $O(\epsilon^{-1}\log^2(1/\epsilon)\log\log(1/\epsilon)\log^* n)$ query time. Ignoring $\log^*(n)$ and $\text{polylog}(1/\epsilon)$ factors, the space bound is $O(n \log n)$ while the query time is $O(1/\epsilon)$, giving a quadratic improvement in $1/\epsilon$ dependency in the space-query product. And yet a decade has passed, and the improvement of Kawarabayashi, Sommer, and Thorup [KST13] remains state-of-the-art.

Recent works instead focus on improving the dependency on $1/\epsilon$ of the *query time* at the cost of a larger space bound. Gu and Xu [GX19] constructed an oracle with $O(1)$ query time and $O(n\log(n)(\log(n/\epsilon) + 2^{O(1/\epsilon)}))$ space; their space bound is exponential in $1/\epsilon$. Chan and Skrepetos [CS19], using the Voronoi diam technique of Cabello [Cab18], improved the space of the oracle of Gu and Xu [GX19] to a (large) polynomial in $1/\epsilon$ at the cost of a slightly worst query time $O(\log(1/\epsilon))$. However, the construction of Chan and Skrepetos [CS19] is randomized. Li and

[1] All oracles in this paper are for undirected and edge-weighted graphs unless mentioned otherwise.

Space	Query time	Reference
$O(n\log(n)/\epsilon)$	$O(1/\epsilon)$	[Tho04, Kle02]
$O(n)$	$O(\log(n)/\epsilon^2)$	[KKS11]
$O(n\log n\log\log(1/\epsilon)\log^* n)$	$O((1/\epsilon)\log^2(1/\epsilon)\log\log(1/\epsilon)\log^* n)$	[KST13]
$O(n\log(n)(\log(n/\epsilon)+2^{O(1/\epsilon)}))$	$O(1)$	[GX19]
$O(n\log^2(n)\epsilon^{-1}+n\log(n)\epsilon^{-(4+\delta)})$ any fixed $\delta>0$	$O(\log(1/\epsilon))$	[CS19]
$O(n\log^2(n)(1/\epsilon)^c)^{++}$ for some constant $c\geq 6$	$O(1)^{++}$	[LP19]
$O(n\log(n)((1/\epsilon)^{o(1)}+\log^* n))$	$\log^{2+o(1)}(1/\epsilon)$	Theorem 1
$O(n\log(n)(\log^{2+o(1)}(1/\epsilon)+\log^* n))$	$(1/\epsilon)^{o(1)}$	Theorem 1

Table 1: Space and query time of known $(1+\epsilon)$-approximate distance oracles for *undirected* planar graphs. The bounds marked $^{++}$ are our own estimation following the description in [LP19]; these bounds are not explicitly computed in [LP19]. The last two rows are our results.

Parter [LP19] devised the VC-dimension technique to reduce the space of the oracle by Gu and Xu [GX19]; the space of their oracle is not explicitly computed but remains polynomial in $1/\epsilon$, and the query time, while is not explicitly mentioned, is $O(1)$. See Table 1 for a summary.

On the other hand, it is plausible to conjecture that the linear dependency on $1/\epsilon$ of the space-query product is optimal. For some distance-based problems in planar graphs where one seeks to have structures that preserve distances approximately with an error parameter ϵ, such as light $(1+\epsilon)$-spanners [ADD+93] or treewidth embedding with low (additive) distortion [FKS19, CFKL20, LF22] or approximate planar emulators [CKT22], it was proved that the dependency on $1/\epsilon$ of the output's quality (such as lightness or treewidth or the number of edges) is $\Omega(1/\epsilon)$. Another related problem is $(1+\epsilon)$-approximate distance labeling scheme where the best-known scheme [Tho04] has labels of size $O(\log(n)/\epsilon)$; again, the dependency on $1/\epsilon$ is linear. For all problems seeking some kinds of $(1+\epsilon)$ approximation in planar graphs that we are aware of, none of them has a sublinear the dependency on $1/\epsilon$ (though there exist such problems for trees [GKK+01, FGNW17], which form a very restricted subclass of planar graphs). Furthermore, the constructions of all known approximate oracles rely on the same fundamental building block using shortest path separators pioneered by Thorup [Tho04] and Klein [Kle02]. More precisely, for each shortest path in the separator, one marks $1/\epsilon$ vertices on the shortest path to serve as *portals* for computing approximate distances. This $1/\epsilon$ factor creeps into the space and/or the query time, which makes the linear dependency seem unavoidable.

In this work, we break the long-standing linear dependency on $1/\epsilon$ in space-query product of approximate distance oracles for the first time. Indeed, we improve the dependency of the space-query product on $1/\epsilon$ from linear all the way down to *subpolynomial* $(1/\epsilon)^{o(1)}$.

Theorem 1. *Let $\epsilon\in(0,1)$ be positive parameter and $G=(V,E,w)$ be an undirected, edge-weighted planar graphs with n vertices. We can construct in $O(n\text{poly}(\log(n),1/\epsilon))$ time a $(1+\epsilon)$-approximate distance oracle that has:*

(1) $O(n\log(n)((1/\epsilon)^{o(1)}+\log^* n))$ space and $\log^{2+o(1)}(1/\epsilon)$ query time or

(2) $O(n\log(n)(\log^{2+o(1)}(1/\epsilon)+\log^* n))$ space and $(1/\epsilon)^{o(1)}$ query time.

Given the aforementioned lower bounds for related problems, we find the result in Theorem 1 rather surprising. It opens the real possibility that the dependency on $1/\epsilon$ could be sublinear or even subpolynomial for distance-related problems where there is no currently-known linear lower bound, such as computing $(1+\epsilon)$-approximate diameter [WY16, CS19] or $(1+\epsilon)$-approximate distance labelings [Tho04]. For approximate distance labelings, by a reduction from exact distance labeling [GPPR04, Tho04], the lower bound dependency on $1/\epsilon$ one can show is $\Omega((1/\epsilon)^{1/3})$. Furthermore, our technique described below might also be used to progress on these problems.

We remark that we do not attempt to minimize the $\text{poly}(1/\epsilon, \log(n))$ factor in the construction time in Theorem 1. In the following section, we review previous techniques and give an overview of our construction.

1.1 Previous and Our Techniques

A conceptual contribution of our work is to view approximate distance oracle constructions through the lens of *local portalization* vs *global portalization*. This view explains why constructions developed over the past two decades fail to break the linear, sometimes quadratic, dependency on $1/\epsilon$ in the space-query product. Through this view, we identify key strengths and weaknesses of each construction, and the challenges in overcoming the linear $1/\epsilon$ barrier. We then design a framework that could exploit the strengths of all of them through which we obtain a sublinear bound $(1/\epsilon)^{o(1)}$ in the space-query product. First, we give a more detailed exposition of existing constructions.

All approximate distance oracles, including ours, use *balanced shortest path separators*. The influential results of Lipton and Tarjan [LT79, LT80] showed that a triangulated planar graph of n vertices has a separator C, called a balanced shortest path separator, which is a Jordan curve consisting of two shortest paths and one edge connecting the two endpoints of the paths, such that there are at most $\frac{2n}{3}$ vertices in the interior and exterior of C, denoted by $\mathsf{Int}(C)$ and $\mathsf{Ext}(C)$, respectively.

A key idea in the construction of Thorup [Tho04] and Klein [Kle02] is to use *portals* along each shortest path of a shortest path separator. They showed that for every vertex v in one side of C, say the interior, one can find a set P_v of $1/\epsilon$ vertices, called portals, for v such that for every vertex $u \in \mathsf{Ext}(C)$, there exists a portal $p \in P_v$ where $d_G(v,p) + d_G(u,p) \leq (1+\epsilon) d_G(u,v)$. This means that for every vertex $v \in V$, we only need to store $O(1/\epsilon)$ distances to vertices in P_v such that the distance between any two vertices u, v in two different sides of C can be approximated within $(1+\epsilon)$ factor by computing $\min_{p \in P_v, q \in P_u} \{d_G(v,p) + d_G(u,q)\}$ in time $O(|P_v| + |P_u|) = O(1/\epsilon)$. The distances between vertices in the same side of C can be handled recursively, at the cost of a $\log(n)$ factor in the space bound since the depth of the recursion is $O(\log(n))$.

We view the portalization of Thorup [Tho04] and Klein [Kle02] as a *local portalization* scheme in the sense that each vertex needs its own set of portals. Evading $O(1/\epsilon^2)$ factor in the space-query product requires breaking the locality. The follow-up construction of Kawarabayashi, Klein, and Sommer used r-division [Fed87] on top of the constructions by Thorup [Tho04] and Klein [Kle02]; their goal is to have an oracle with $O(n)$ space and $O(\epsilon^{-2} \log^2 n)$ query time. Their construction also followed local portalization and, hence, the space-query product remained the same.

Kawarabayashi, Sommer, and Thorup [KST13] (KST) improved the quadratic bound $O(1/\epsilon^2)$ in the space-query product by breaking the locality of portals entirely. Specifically, up to a factor of $\text{polylog}(1/\epsilon, \log^* n)$, they improved the space to $n \log(n)$ while keeping the same query time $1/\epsilon$. Their approach reduced constructing oracles with multiplicative stretch $(1+\epsilon)$ to constructing oracles with *additive stretch* $+\epsilon D$ where D is the diameter of the graph. (We say that an oracle has

an additive stretch $+\Delta$ if for every two vertices u and v, the distance returned by the oracle $\hat{d}(u,v)$ satisfies: $d_G(u,v) \le \hat{d}(u,v) \le d_G(u,v) + \Delta$.) The reduction introduces a small loss of an $O(\log n)$ factor in the space and an $O(1)$ factor in query time. A key advantage of additive stretch over multiplicative stretch is that the portals become global: for every shortest path (of length at most D) in a shortest path separator, we can place a set P of $O(1/\epsilon)$ portals (independent of the vertices) such that for any two vertices u, v in different sides of C, $\min_{p\in P}\{d_G(u,p)+d_G(v,p)\} \le d_G(u,v)+\epsilon D$. Thus, vertices in the graph now share the same set of portals, which is the source of the space improvement. To answer a query, they need to iterate over P, which takes $O(|P| = O(1/\epsilon))$ time.

Subsequent constructions [CS19, GX19, LP19] improved the query time of the KST oracle to either $O(1)$ [GX19, LP19] or $O(\log(1/\epsilon))$ [CS19] at the cost of a large dependency on $1/\epsilon$ in the space. Gu and Xu [GX19] employed the distance encoding argument of Weimann and Yuster [WY16] that has a factor $2^{O(1/\epsilon)}$ in the space. Li and Parter [LP19] reduced the factor $2^{O(1/\epsilon)}$ to $O(\text{poly}(1/\epsilon))$ using their VC-dimension technique. Chan and Skrepetos [CS19] employed the Voronoi diagram technique of Cabello [Cab18]; their construction broke away from the global portalization however: each vertex in the graph must store its own Voronoi diagram defined by its distances to the portals. As a result of this locality, the space of their oracle has a factor $(1/\epsilon)^4$. Thus, the Voronoi diagram technique, though a cornerstone of *exact* distance oracle constructions (see Section 1.2), does not seem to be the right approach for breaking the linear factor $1/\epsilon$ in the space-query product achieved by KST oracle.

Viewing KST oracle through our global lens of portalization is particularly illuminating for breaking the linear $1/\epsilon$ bound. Specifically, we show that, for breaking the $O(1/\epsilon)$ bound, it suffices to store $\text{poly}(1/\epsilon)$ machine words *globally* per shortest path separator. More precisely, each time we apply the shortest path separator to separate a graph, we could store a global data structure of up to $\text{poly}(1/\epsilon)$ words of space. Every vertex in the graph has a pointer to (a portion of) the data structure; the pointer only costs $O(\log(1/\epsilon))$ bits of space. In retrospect, the KST construction can be seen as storing only $O(1/\epsilon)$ words of space globally, one word for each portal of the paths in the separator. The key difference of our construction over KST's is that, in KST, each vertex needs to compute distances to the (shared) portals during the query stage, then computing distances between two vertices requires looping over the portals that takes $O(1/\epsilon)$ time. Our key idea to remove the $1/\epsilon$ factor completely in the query time is the following. We precompute a small set of approximate distances in the global data structure. Then, given two vertices and their pointers to the global data structure, we can look up their approximate distance in $O(1)$ time.

In our construction, each vertex holds a pointer to an *approximate distance pattern* stored in the global data structure in the graph. Our approximate distance pattern is an approximate version of the *exact distance pattern* introduced by Fredslund-Hansen, Mozes, and Wulff-Nilsen [FHMWN21] in their construction of *exact distance oracles* for *unweighted* planar graphs. (Other than using distance patterns, their construction is different from ours and other approximate distance oracle constructions, and it only works for unweighted graphs.) An approximate distance pattern encodes the (approximate) distances from a vertex to the portals on a shortest path of the shortest path separator. We pre-compute all approximate distance patterns and the distance between any two approximate distance patterns, and store this information in the global data structure. The idea is that, given access to two approximate distance pointers of two vertices u and v, we can access their pre-computed distance in $O(1)$ time in the global data structure. Our global data structure has only $\text{poly}(1/\epsilon)$ space and, hence, the number of patterns must also be $\text{poly}(1/\epsilon)$; for this, we employ the VC-dimension argument of Li and Parter [LP19].

A problem with using a global data structure of size $\text{poly}(1/\epsilon)$ for each subgraph of G arising in the construction is the space bound. Specifically, if we recursively decompose G using balanced shortest path separators until each subgraph has $O(1)$ size, we end up with a separation tree, denoted by $\mathcal{T}$, with $O(n)$ nodes. As each node of $\mathcal{T}$ is associated with a global data structure of size $\text{poly}(1/\epsilon)$ (for the subgraph of that node), the total size of the data structure is $O(n\text{poly}(1/\epsilon))$. A possible solution to this problem is the following simple idea used by many oracle constructions [KKS11, KST13, CS19]: stop separating a subgraph once it has size $O(1/\epsilon^c)$ for some sufficiently big c. (For each subgraph of size $O(1/\epsilon^c)$, the standard approach is to use an exact distance oracle [KKS11, KST13]; we will come back to this issue later.) The number of nodes of the separation tree is $O(n/\epsilon^c)$, and hence the total size of all data structures associated with its nodes is $O((n/\epsilon^c)\text{poly}(1/\epsilon)) = O(n)$ for an appropriate choice of c. Then for each vertex v, we store a pointer to its approximate pattern w.r.t. the boundary portals of the leaf subgraph in $\mathcal{T}$ containing v.

Yet a new problem arises: for each vertex v in a leaf node α of $\mathcal{T}$, we need to compute the approximate pattern of v w.r.t. the portals of an ancestor node, say β. Following Fredslund-Hansen, Mozes, and Wulff-Nilsen [FHMWN21], we can compute a pattern *induced by* the approximate pattern of v for portals of α (see Definition 6 for a precise definition). The issue is that, since distances are approximate, the induced pattern might not be in the set of approximate patterns stored at β; therefore, we can no longer look up the distance stored at the global data structure of β. (In the exact distance setting of [FHMWN21], this issue does not happen since the induced pattern of an exact pattern is another exact pattern.) Our key idea to overcome this problem is the following. We show that the induced pattern is close (in ℓ^∞ norm) to an approximate distance pattern stored at β. Then, for each induced pattern $\mathbf{p}$ between α and β, we store a *pointer* to the approximate pattern in β close to $\mathbf{p}$. Therefore, once the induced pattern is computed, we follow the pointer to the closest approximate pattern stored at β.

We now go back to the subgraphs of size $O(1/\epsilon^c)$ associated with leaves of $\mathcal{T}$. The standard approach is to use an exact distance oracle for each subgraph [KKS11, KST13]. For breaking the linear bound $1/\epsilon$ in the space-query product, it suffices to use the oracle of Charalampopoulos et al. [CGMW19]. Here we use the recent exact distance oracle by Long and Pettie [LP21] instead to obtain a better dependency on $1/\epsilon$. Specifically, we apply the Long-Pettie oracle (for n-vertex graphs) in two different regimes: (a) $n^{1+o(1)}$ space and $\log(n)^{2+o(1)}$ query time and (b) $n\log^{2+o(1)}(n)$ space and $n^{o(1)}$ query time. Two regimes lead to two different oracles with additive stretch $+\epsilon D$: regime (a) gives an oracle with $O(n(1/\epsilon)^{o(1)}\log(n))$ space and $O(\log^{2+o(1)}(1/\epsilon))$ query time and regime (b) gives an oracle with $O(n\log^{2+o(1)}(1/\epsilon)\log(n))$ space and $O((1/\epsilon)^{o(1)})$ query time.

Finally, with some additional ideas on top of our framework, we remove the $\log(n)$ factor in the space of the additive oracle by recursion at the cost of an additive $\log^*(n)$ term in the space. We note that, unlike KST, our oracle does not have the factor $\log^*(n)$ in the query time.

Theorem 2. *Let $\epsilon > 0, D > 0$ be a positive parameter and $G = (V, E, w)$ be an undirected n-vertex planar graph of diameter D. There is an approximate distance oracle of additive stretch $+\epsilon D$ with construction time $O(n\text{poly}(\log n, \epsilon))$ and:*

(1) $O(n((1/\epsilon)^{o(1)} + \log^*(n)))$ space and $\log^{2+o(1)}(1/\epsilon)$ query time or

(2) $O(n(\log^{2+o(1)}(1/\epsilon) + \log^*(n)))$ space and $(1/\epsilon)^{o(1)}$ query time.

1.2 Related Work

A closely related data structure in planar graphs is *exact* distance oracles. (All exact oracles mentioned in the following work for planar *digraphs*; thus, they are naturally applicable to planar undirected graphs.) There is a very long line of work on constructing exact distance oracles, starting from the seminal papers of Lipton and Tarjan [LT79, LT80] who constructed an exact oracle with $O(n^{3/2})$ space and $O(\sqrt{n})$ query time. Subsequent results [ACC+96, Dji96, CX00, FR01, Cab10, WN10, MS12] improved the result of Lipton and Tarjan in two ways: designing new space-query time trade-offs [ACC+96, Dji96, CX00, WN10] or obtaining a truly subquadratic space-query product [Dji96, CX00, Cab10, FR01, MS12]. However, none of these oracles has a *truly subquadratic* space and *polylogarithmic query time* until the work of Cohen-Addad, Dahlgaard, and WulffNilsen [CADWN17]. Specifically, they constructed an oracle with $O(n^{5/3})$ space and $O(\log n)$ query time. The result of Cohen-Addad, Dahlgaard, and Wulff-Nilsen is the major turning point for exact distance oracles: they were the first to use the Voronoi diagram technique of Cabello [Cab18]. Follow-up results [GMWWN18, CGMW19, LP21], all based on the Voronoi diagram technique, significantly improved the space-query time trade-off of Cohen-Addad, Dahlgaard, and Wulff-Nilsen [CADWN17], culminating in the oracles by Long and Pettie [LP21] that have (i) $O(n^{1+o(1)})$ space and $O(\log^{2+o(1)} n)$ query time or (ii) $O(n \log^{2+o(1)})$ space and $O(n^{o(1)})$ query time. (For various other trade-offs, see Table 1 in [LP21] for details.) The $n^{o(1)}$ factor, while sublinear, is $2^{\Omega(\sqrt{\log n})}$. Therefore, though we have witnessed tremendous progress on exact distance oracles, the dependency on n of spaces/query time of exact oracles remains far from that of approximate oracles.

The exact oracle construction of Fredslund-Hansen, Mozes, and Wulff-Nilsen [FHMWN21] is fundamentally different from the constructions mentioned above: the main tool is the VC-dimension technique of Li and Parter [LP19]. Their main goal is to get an oracle with a constant query time; the space bound is $O(n^{5/3+\delta})$ for any fixed constant $\delta > 0$. The space-query product of their oracle is not competitive with the Voronoi-diagram-based oracles [GMWWN18, CGMW19, LP21]. Furthermore, their construction only works for undirected and unweighted planar graphs.

2 Preliminaries

Given a graph G, we denoted by $V(G)$ the vertex set of G and $E(G)$ the edge set of G. We denote an edge-weighted graph G with a vertex set V, edge set E, and non-negative edge-weight function $w : E \to \mathbb{R}^+$ by $G = (V, E, w)$. For any two vertices $u, v \in V$, we denote by $d_G(u, v)$ the distance between u and v in G. We denote by $SP(u, v, G)$ a shortest path from u to v in G. For a given path P containing two vertices x and y, we denote by $P[x, y]$ the x-to-y subpath of P.

Let $G = (V, E, w)$ be an edge-weighted planar graph equipped with a planar embedding, called a *plane graph*. A region R of $G = (V, E, w)$ is a subgraph of G. A hole of R is a face of R that is not a face of G. The boundary of R, denoted by ∂R, is the set of vertices of R that are on the boundaries of the holes of R. Vertices in $V(R) \setminus \partial R$ are called *interior vertices*. A vertex $u \in V(G) \setminus V(R)$ is inside a hole h if u is embedded inside the face h of R.

Next we define the notion of *crossing* between two simple paths, say P and Q, drawn on the plane. We say that a path X is a proper subpath of P if X does not contain any endpoint of P. Assume that there exists a maximal subpath $X \subseteq P \cap Q$ that is proper. We orient P and Q such that their orientations agree on X. Let B_ϵ be a topological disk containing all points on the plane

of distance at most infinitesimal $\epsilon > 0$ from points on X. P partitions B_ϵ in two regions called the left side and the right side of P. We say that Q crosses P if the edge entering X and the edge leaving X of Q contain points in different sides of P (see Figure 1(a)). This crossing definition generalizes naturally to the case where Q is a cycle instead of a path; in this case, any subpath of Q is proper.

Distance preserving minors. Let P be a set of terminals in a graph $G = (V, E, w)$. Let K be the graph obtained by adding all pairwise shortest paths in G between terminals in P, and contracting degree-2 vertices that are not in P; we assume that the shortest paths are chosen in such a way that the intersection of any two shortest paths is either empty or connected. The weight of an edge in K is the shortest distance between its endpoints in G. K is a minor of G, and is called a *distance preserving minor* for P [KNZ14]. If G is planar, then K is also planar.

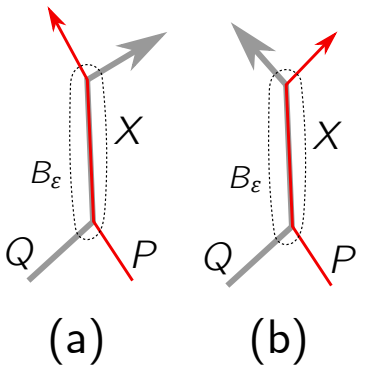


Figure 1: (a) P and Q cross. (b) P and Q are non-crossing.

Lemma 1 (Theorem 2.1 [KNZ14]). *Let P be a set of k terminal in a graph $G = (V, E, w)$, then its distance preserving minor K has size $O(k^4)$.*

Our construction uses Lemma 1 for the case where G is planar and P is on the outer face of G.

Exact distance oracles. Long and Pettie [LP21] constructed an exact distance oracle for planar digraphs. In our paper, we use their results for undirected graphs.

Theorem 3 (Theorem 1.1 [LP21]). *Let $G = (V, E, w)$ be any given planar digraph with n vertices. We can construct an exact distance oracle in time $n^{3/2+o(1)}$ that has:*

(1) $n^{1+o(1)}$ space and $\log^{2+o(1)} n$ query time or

(2) $n \log^{2+o(1)}$ space and $n^{o(1)}$ query time.

VC dimension. Let U be a ground set, and $\mathcal{F}$ be a family of subsets of U. We say that $\mathcal{F}$ shatters a set $X \subseteq U$ if for every subset $Y \subseteq X$, there exists a set $Z \in \mathcal{F}$ such that $Z \cap X = Y$. We say that $\mathcal{F}$ has VC-dimension k if the largest set shattered by $\mathcal{F}$ has size k. The famous Sauer–Shelah lemma [Sau72, She72] bounds the size of $\mathcal{F}$ when its VC-dimension is at most k.

Lemma 2 (Sauer–Shelah Lemma). *Let $\mathcal{F}$ be a family of subsets of a ground set with n elements. If VC-dimension of $\mathcal{F}$ is at most k, then $|\mathcal{F}| = O(n^k)$.*

We use $[k]$ to denote the set $\{1, 2 \ldots, k\}$. If $\mathbf{x}$ is a k-dimensional vector, we denote by $\mathbf{x}[i : j]$ for given $i \le j \le k$ the $(j - i + 1)$-dimensional vector such that s-th entry of $\mathbf{x}[i : j]$ is $\mathbf{x}[i + s - 1]$ for any $s \in [j - i + 1]$. We call vector $\mathbf{x}[i : j]$ the (i, j)-*restriction* of $\mathbf{x}$.

Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$ be two k-dimensional vectors. If $\|\mathbf{x}, \mathbf{y}\|_\infty \le \delta$, we write $\mathbf{x} \approx_\delta \mathbf{y}$. (The same notation applies to scalars since we can view them as 1-dimensional vectors.) Observe by the triangle inequality that:

Observation 1. *If $\mathbf{x} \approx_{\delta_1} \mathbf{y}$ and $\mathbf{y} \approx_{\delta_2} \mathbf{z}$, then $\mathbf{x} \approx_{\delta_1+\delta_2} \mathbf{z}$.*

We can show directly from the definition that:

Claim 1. *If* $\mathbf{x}_1 \approx_{\delta_1} \mathbf{x}_2$ *and* $\mathbf{y}_1 \approx_{\delta_2} \mathbf{y}_2$*, then:*

$$\min_{i\in[k]}\{\mathbf{x}_1[i]+\mathbf{y}_1[i]\} \approx_{\delta_1+\delta_2} \min_{i\in[k]}\{\mathbf{x}_2[i]+\mathbf{y}_2[i]\} \ .$$

where k *is the dimension of these vectors.*

Proof. Observe by definition that $|\mathbf{x}_1[i]-\mathbf{x}_2[i]| \le \delta_1$ and $|\mathbf{y}_1[i]-\mathbf{y}_2[i]| \le \delta_2$ for every $i \in [k]$. It follows from the triangle inequality that $\mathbf{x}_1[i]+\mathbf{y}_1[i] \approx_{\delta_1+\delta_2} \mathbf{x}_2[i]+\mathbf{y}_2[i]$ for every $i \in [k]$. Let $i^* = \arg\min_{i\in[k]}\{\mathbf{x}_2[i]+\mathbf{y}_2[i]\}$. Then:

$$\begin{aligned}\min_{i\in[k]}\{\mathbf{x}_1[i]+\mathbf{y}_1[i]\} &\le \mathbf{x}_1[i^*]+\mathbf{y}_1[i^*] \\ &\le \mathbf{x}_2[i^*]+\mathbf{y}_2[i^*]+(\delta_1+\delta_2) \\ &= \min_{i\in[k]}\{\mathbf{x}_2[i]+\mathbf{y}_2[i]\}+(\delta_1+\delta_2) \ .\end{aligned}$$

By the same argument, $\min_{i\in[k]}\{\mathbf{x}_2[i]+\mathbf{y}_2[i]\} \le \min_{i\in[k]}\{\mathbf{x}_1[i]+\mathbf{y}_1[i]\}+(\delta_1+\delta_2)$; this implies the claim. □

3 Distance Oracles with Additive Stretch

In this section, we construct an oracle with additive stretch for planar graphs as claimed in Theorem 2. For a simpler presentation of the ideas, we first prove a weaker version of Theorem 2, which is Theorem 4 below, where we allow a $\log^{o(1)}(n)$ factor in the space and query time. We then present a full proof of Theorem 2 in Section 3.5.

Theorem 4. *Let* $\epsilon > 0, D > 0$ *be a positive parameters and* $G = (V, E, w)$ *be an undirected* n*-vertex planar graph of diameter* D*. There is an approximate distance oracle with additive stretch* $+\epsilon D$ *that has* $O(n\mathrm{poly}(\log n, \epsilon))$ *construction time and:*

(1) $n(1/\epsilon)^{o(1)}\log^{o(1)}(n)$ *space and* $\log^2(1/\epsilon)+(\log\log(n))^{2+o(1)}$ *query time or*

(2) $n(\log^{2+o(1)}(\log n)+\log^{2+o(1)}(1/\epsilon))$ *space and* $\log^{o(1)}(n)(1/\epsilon)^{o(1)}$ *query time.*

This section is organized as follows. In Section 3.1 we introduce the notion of approximate patterns. In Section 3.2, we show how to compute an approximate distance of two vertices given their approximate patterns to portals on a shortest path separator. In Section 3.3, we study the composition of two approximate distance patterns, and show that the composition is close to another approximate distance pattern in ℓ^∞ norm. In Section 3.4 we prove Theorem 4 and in Section 3.5, we extend the proof of Theorem 4 to Theorem 2.

3.1 Approximate Patterns

Let $G = (V, E, w)$ be a planar graph and σ be a sequence of k vertices of G; the i-th vertex of σ is denoted by σ_i. For a real number $\Delta \in \mathbb{R}$, which could be negative, positive or zero, and an index $i \in [k-1]$, we define:

$$A_i^\Delta = \{v \in V(G) | d_G(v, \sigma_{i+1}) - d_G(v, \sigma_i) \le \Delta\} \ . \tag{1}$$

We call the pair (i, Δ) a *distance index* and A_i^Δ a vertex set associated with the distance index (i, Δ). See Figure 2. The following theorem is proved by Li and Parter [LP19].

Theorem 5 (Theorem 3.7, Li-Parter [LP19]). *Let $G = (V, E, w)$ be a planar graph and σ be a sequence of k vertices in clockwise order on a face f of G. Let M be a set of real numbers. For each vertex $v \in G$, let $X_v = \{(i, \Delta) : i \in [k-1], \Delta \in M, v \in A_i^{\Delta}\}$ be the set of distance indices whose associated vertex sets contain v. Let $\mathcal{F} = \{X_v\}_{v \in V(G)}$ be a family of sets of distance indices. Then $\mathcal{F}$ has VC-dimension at most 3.*

Remark 1. *An intuitive interpretation of Theorem 5 is the following. The (typically finite) set M tells us the difference between the distances from a vertex v to two consecutive vertices in the sequence σ: if $d_G(v, \sigma_{i+1}) - d_G(v, \sigma_i) \le \Delta$ for some $\Delta \in M$, then $(i, \Delta) \in X_v$. For each i, let $\Delta_i^* \in M$ be the smallest such that $(i, \Delta_i^*) \in X_v$. Then, given $d_G(v, \sigma_1)$ and X_v, we can inductively recover an* upper bound *on $d_G(v, \sigma_{i+1})$ for any given $i \in [1, k-1]$ by unrolling the recursion $d_G(v, \sigma_{i+1}) \le d_G(v, \sigma_i) + \Delta_i^*$. That is, we get $d_G(v, \sigma_{i+1}) \le d_G(v, \sigma_1) + \sum_{j=1}^{i} \Delta_j^*$. Depending on the choice of M, this upper bound could be an exact or approximate estimation of the distance $d_G(v, \sigma_{i+1})$. Thus, X_v and M encode the distance information from v to vertices in σ; the notion of approximate distance encoding below formalizes this intuition. From this point of view, the family $\mathcal{F}$ contains the approximate distance encodings of all vertices of G to vertices in σ. By Lemma 2, Theorem 5 implies that there are only a polynomial number (in k and $|M|$) of approximate distance encodings; the number of encodings does not depend on n!*

We note that Theorem 3.7 in [LP19] is only stated for $\Delta \in \{-1, 0\}$; however, as noted by Li and Parter [LP19] in the proof of Theorem 3.7, it holds for any set of real numbers. They used the general version to approximate weighted diameters of planar graphs.

Our construction relies on the notion of approximate patterns. For a given positive real number δ, we define $[a]_\delta$ to be the closest integer multiple of δ that is at least a. Specifically,

$$[a]_\delta = \lceil \frac{a}{\delta} \rceil \cdot \delta \qquad \forall a \in \mathbb{R} \,. \tag{2}$$

Next, we define approximate pattern and approximate distance decoding. Our approximate pattern is the approximate version of (exact) patterns introduced by Fredslund-Hansen, Mozes, and Wulff-Nilsen [FHMWN21].

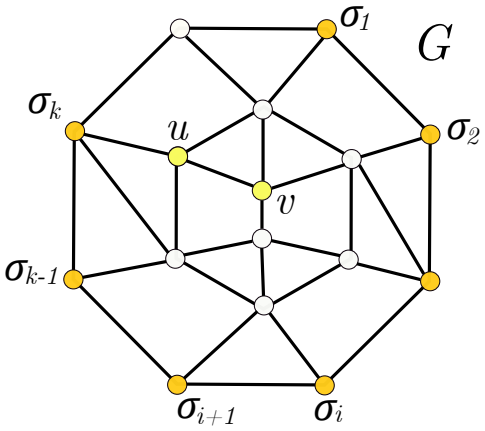


Figure 2: G is unweighted and σ is on the outer face. Both u and v are in A_i^0.

Definition 1 (Approximate Pattern and Distance Encoding). *Let σ be a sequence of vertices in a graph G. Let u be a vertex in G. A* δ-approximate pattern *of u w.r.t. σ in G for some parameter $\delta > 0$ is a $(k-1)$-dimensional vector $\mathbf{p}$ such that $\mathbf{p}[i] = [d_G(u, \sigma_{i+1}) - d_G(u, \sigma_i)]_\delta$ for every $i \in [k-1]$.*
A δ-approximate distance encoding *of u w.r.t. σ in G is a k-dimensional vector $\mathbf{d}$ such that $\mathbf{d}[1] = [d_G(u, \sigma_1)]_\delta$ and $\mathbf{d}[2:k] = \mathbf{p}$. That is, $\mathbf{d}[i] = \mathbf{p}[i-1] = [d_G(u, \sigma_i) - d_G(u, \sigma_{i-1})]_\delta$ for all $2 \le i \le k$.*

Given the distance encoding $\mathbf{d}$ of a vertex u, we can *decode* $\mathbf{d}$ to get a k-dimensional distance vector $\mathbf{a}$ from u to vertices in σ by computing $\mathbf{a}[i] = \sum_{j=1}^{i} \mathbf{d}[j]$ for every $i \in [k]$. (In Lemma 3

below, we show that $\mathbf{a}[i]$ is close to $d_G(u, \sigma_i)$.) We can generalize the decoding procedure for any k-dimensional vector $\mathbf{x}$ even if it does not correspond to a distance encoding of any vertex in the graph. We will use this generalization in our distance oracle construction.

Definition 2 (Distance Decoding)**.** *Let $\mathbf{x}$ be a k-dimensional vector. The distance decoding of $\mathbf{x}$ is a k-dimensional vector, denoted by $\mathbf{x}^{\leq}$, such that $\mathbf{x}^{\leq}[i] = \sum_{j=1}^{i} \mathbf{x}[j]$.*

Next, we show that the distance decoding of a δ-approximate distance encoding of a vertex u in the graph gives approximate distances from u to vertices in σ.

Lemma 3. *Let $\mathbf{d}$ be a δ-approximate distance encoding of u w.r.t. a sequence σ of k vertices in graph G. For any $i \in [k]$, we define $\tilde{d}_G(u, \sigma_i) = \mathbf{d}^{\leq}[i]$. Then, it holds that*

$$d_G(u, \sigma_i) \leq \tilde{d}_G(u, \sigma_i) \leq d_G(u, \sigma_i) + i \cdot \delta. \tag{3}$$

Proof. Let $\mathbf{p}$ be the δ-approximate pattern of u w.r.t. σ. Observe by the definition of $[\cdot]_\delta$ and Definition 1 that for any $j \in [k-1]$:

$$d_G(u, \sigma_{j+1}) - d_G(u, \sigma_j) \leq \mathbf{p}[j] \leq d_G(u, \sigma_{j+1}) - d_G(u, \sigma_j) + \delta \tag{4}$$

By summing both sides of Equation (4) when $j = 1, \ldots, i-1$, it follows that:

$$d_G(u, \sigma_i) - d_G(u, \sigma_1) \leq \sum_{j=1}^{i-1} \mathbf{p}[j] \leq d_G(u, \sigma_i) - d_G(u, \sigma_1) + (i-1)\delta \ .$$

Thus, we have:

$$d_G(u, \sigma_i) \leq d_G(u, \sigma_1) + \sum_{j=1}^{i-1} \mathbf{p}[j] \leq d_G(u, \sigma_i) + (i-1)\delta \ . \tag{5}$$

By definition of $[\cdot]_\delta$, $d_G(u, \sigma_1) \leq [d_G(u, \sigma_1)]_\delta \leq d_G(u, \sigma_1) + \delta$. Thus, Equation (5) implies:

$$d_G(u, \sigma_i) \leq [d_G(u, \sigma_1)]_\delta + \sum_{j=1}^{i-1} \mathbf{p}[j] \leq d_G(u, \sigma_i) + i\delta \ . \tag{6}$$

By Definition 1, $[d_G(u, \sigma_1)]_\delta + \sum_{j=1}^{i-1} \mathbf{p}[j] = \sum_{j=1}^{i} \mathbf{d}[j] = \mathbf{d}^{\leq}[i]$. The lemma now follows from Equation (6). □

We note that by the definition of $\tilde{d}_G$ in Lemma 3,

Remark 2. $\tilde{d}_G(u, \sigma_1) = [d_G(u, \sigma_1)]_\delta$.

The following lemma bounds the number of patterns when $|d_G(u, \sigma_{i+1}) - d(u, \sigma_i)|$ is not much larger than δ.

Lemma 4. *Let $G = (V, E, w)$ be a planar graph and σ be a sequence of k vertices in clockwise order on a face f of G. Let g be a non-negative integer such that:*

$$-g\delta \leq d_G(u, \sigma_{i+1}) - d(u, \sigma_i) \leq g\delta \qquad \forall i \in [k-1] \tag{7}$$

For every vertex $u \in V$, let $\mathbf{p}_u$ be the δ-approximate pattern of u w.r.t. σ. Let $\mathbf{P} = \{\mathbf{p}_u : u \in V\}$ be the set of all δ-approximate patterns w.r.t. σ. Then $|\mathbf{P}| = O((kg)^3)$.

Proof. Let $M = \{-g\delta, (-g+1)\delta, \ldots, -\delta, 0, \delta, \ldots, g\delta\}$ be a set of $(2g+1)$ real numbers. Recall that $X_u = \{(i, \Delta)\}$ is the set of distance indices associated with a vertex u, where $\Delta \in M$. By Theorem 5, $\mathcal{F} = \{X_v\}_{v \in V(G)}$ has VC-dimension at most 3. Since the ground set $\{(i, \Delta)\}_{i \in [k-1], \Delta \in M}$ has size at most $k(2g+1)$, by Lemma 2, $|\mathcal{F}| = O((k(2g+1))^3) = O((kg)^3)$.

We show below that the map φ that maps $\mathbf{p}_u$ to X_u is a bijection from $\mathbf{P}$ to $\mathcal{F}$, which would imply the claimed bound on $|\mathbf{P}|$. By definition, φ is surjective. Next, we show that φ is injective.

Let $u \neq v$ be two vertices such that $\mathbf{p}_u \neq \mathbf{p}_v$. Then there exists some $i \in [k-1]$ such that $[d_G(u, \sigma_{i+1}) - d_G(u, \sigma_i)]_\delta \neq [d_G(v, \sigma_{i+1}) - d_G(v, \sigma_i)]_\delta$. Let $\Delta_u = [d_G(u, \sigma_{i+1}) - d_G(u, \sigma_i)]_\delta$ and $\Delta_v = [d_G(v, \sigma_{i+1}) - d_G(v, \sigma_i)]_\delta$. Since $\Delta_u \neq \Delta_v$, w.l.o.g, we assume that $\Delta_u < \Delta_v$. By the definition of $[\cdot]_\delta$, $d_G(u, \sigma_{i+1}) - d_G(u, \sigma_i) \leq \Delta_u$. Thus, $u \in A_i^{\Delta_u}$, and that $(i, \Delta_u) \in X_u$.

Similarly, $v \in A_i^{\Delta_v}$. Furthermore, by the definition of $[\cdot]_\delta$ that Δ_v is the least multiple of δ that is at least $d_G(v, \sigma_{i+1}) - d_G(v, \sigma_i)$. Thus, there is no other $\Delta' \in M$ such that $\Delta' < \Delta_v$ and $v \in A_i^{\Delta'}$. Since $\Delta_u < \Delta_v$, $v \notin A_i^{\Delta_u}$. That is, $(i, \Delta_u) \notin X_v$. It follows that $X_u \neq X_v$. Therefore, φ is injective. □

We conclude this section by defining the distance between an approximate pattern and a vertex. Recall that $\tilde{d}_G(u, \sigma_i)$ is the approximate distance from u to vertex σ_i computed from the δ-approximate distance encoding of u w.r.t. σ (see Lemma 3).

Definition 3 (Pattern-Vertex Distance)**.** *Let v be a vertex and $\mathbf{p}$ be an approximate pattern (of some vertex) w.r.t. a sequence σ in $G = (V, E, w)$. Then, the distance between $\mathbf{p}$ and v is defined as $\tilde{d}_G(v, \mathbf{p}) = \min_{1 \leq i \leq k} \{\tilde{d}_G(v, \sigma_i) + \sum_{j=1}^{i-1} \mathbf{p}[j]\}$.*

3.2 Computing Distances from Approximate Distance Encodings

A basic building block in our distance oracle is to query the distance between two vertices separated by a cycle C given their approximate patterns to a sequence of vertices on the cycle. In the work of Fredslund-Hansen, Mozes, and Wulff-Nilsen [FHMWN21], (exact) patterns are defined w.r.t. *all vertices* on C. Using this fact, they can show that, to query the distance between a vertex u outside a cycle C to a vertex v, it suffices to compute the distance from u to a fixed vertex $\sigma_1 \in C$, and compute the distance from v to the pattern of u w.r.t. C. Cycle C does not have to have any special structure for the distance query to work, and this follows from the fact that their exact patterns encode exact distances. However, this property no longer holds in our setting, as pattern are approximate and defined w.r.t. a *subset* of vertices on C.

We introduce a property, call the *single-crossing property*, and we show that if the separating cycle C is single-crossing, one can retrieve the approximate distance between u and v in different sides of C based on their approximate patterns to the boundary.

Definition 4 (Single-Crossing Property)**.** *Let C be a simple cycle in a plane graph $G = (V, E, w)$. We say that C is single-crossing if for any two vertices u, v such that $u \in \mathsf{Int}(C)$ and $v \in \mathsf{Ext}(C)$, then there exists a shortest path from u to v crossing C at most once.*

Let σ be a sequence of vertices ordered clockwise on C. We say that σ is a *τ-cover* of C for some $\tau \geq 0$ if for every $u \in C$, $d_C(\sigma_i, u) \leq \tau$ where $i \in [k]$ is the index such that $u \in C[\sigma_i, \sigma_{i+1}]$; here $\sigma_{k+1} = \sigma_1$. In this section, we show the following.

Lemma 5. *Let $\delta > 0, \tau \geq 0$ be parameters. Let C be single-crossing simple cycle of a plane graph $G = (V, E, w)$, and σ be a sequence of k vertices ordered clockwise on C that τ-covers C. Let G^{in}*

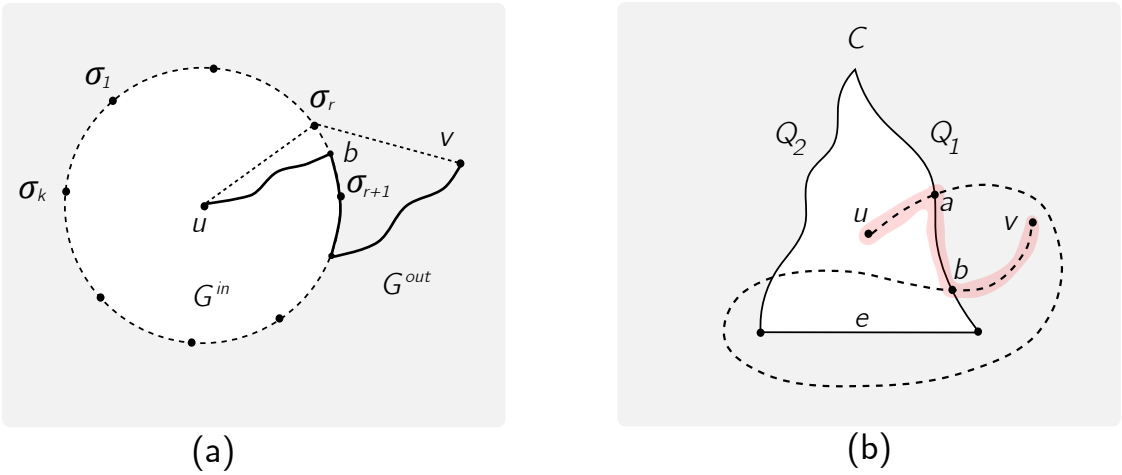


Figure 3: (a) The solid path from u to v is $SP(u, v, G)$; this path crosses the shortest path separator (dashed cycle) once. (b) The dashed path is a shortest path from u to v that crosses the path Q_1 in a shortest path separator at least twice. The red-highlighted path is a new shortest path from u to v that crosses Q_1 only once.

(G^{out}) be the subgraph of G induced by vertices inside (outside) or on C. Let $u \in V(G^{in})$ and $v \in V(G^{out})$ be any two vertices. Let $\mathbf{p}_u$ and $\mathbf{d}_u$ ($\mathbf{p}_v$ and $\mathbf{d}_v$) be the δ-approximate pattern and δ-approximate distance encoding, respectively, of u (v) w.r.t. σ in G^{in} (G^{out}). Let

$$\tilde{d}_G(u, v) \stackrel{def.}{=} \min_{1 \le i \le k} \left\{ \tilde{d}_{G^{in}}(u, \sigma_i) + \tilde{d}_{G^{out}}(v, \sigma_i) \right\} \tag{8}$$

where $\tilde{d}_{G^{in}}(u, \sigma_i) = \mathbf{d}_u^{\le}[i]$ and $\tilde{d}_{G^{out}}(v, \sigma_i) = \mathbf{d}_v^{\le}[i]$. Then, the followings hold:

(1) $d_G(u, v) \le \tilde{d}_G(u, v) \le d_G(u, v) + 2k\delta + 2\tau$.

(2) $\tilde{d}_G(u, v) = \tilde{d}_{G^{out}}(\mathbf{p}_u, v) + \tilde{d}_{G^{in}}(u, \sigma_1)$.

Proof. See Figure 3(a) for an illustration. Note that G is undirected. By Lemma 3, we have that $\tilde{d}_{G^{in}}(u, \sigma_i) \ge d_{G^{in}}(u, \sigma_i)$ and that $\tilde{d}_{G^{out}}(v, \sigma_i) \ge d_{G^{out}}(v, \sigma_i)$. By the triangle inequality, we have that $\tilde{d}_{G^{in}}(u, \sigma_i) + \tilde{d}_{G^{out}}(v, \sigma_i) \ge d_G(u, v)$ for any $i \in [k]$. It follows that

$$\tilde{d}_G(u, v) = \min_{i \in [k]} \{ d_{G^{in}}(u, \sigma_i) + d_{G^{out}}(v, \sigma_i) \} \ge d_G(u, v) \ .$$

This implies the left inequality in Item (1).

To show the right inequality in Item (1), we observe that, since C is simple-crossing, there is a shortest path $SP(u, v, G)$ from u to v cross C at most once. Thus, there exists a vertex $b \in C$ such that $SP(u, v, G) = SP(u, b, G^{in}) \circ SP(b, v, G^{out})$, where $SP(u, b, G^{in})$ ($SP(b, v, G^{out})$) is the shortest u-to-b (b-to-v) path in G^{in} (G^{out}). Let $r \in [k]$ be such that $d_C[\sigma_r, b] \le \tau$; r exists since σ is a τ-cover of C.By the triangle inequality we have:

$$d_{G^{in}}(u, \sigma_r) + d_{G^{out}}(\sigma_r, v) \le d_G(u, v) + 2d_C[\sigma_r, b] \le d_G(u, v) + 2\tau \tag{9}$$

Furthermore, by Lemma 3, the RHS of Equation (8) is at most

$$\begin{aligned}\min_{i\in[k]}\{d_{G^{in}}(u,\sigma_i)+d_{G^{out}}(v,\sigma_i)+2i\delta\} &\le d_{G^{in}}(u,\sigma_r)+d_{G^{out}}(v,\sigma_r)+2k\delta\\ &\le d_G(u,v)+2k\delta+2\tau\end{aligned}$$

by Equation (9).

Next, we prove Item (2). Note by Remark 2 that $\tilde{d}_{G^{in}}(u,\sigma_1)=[d_{G^{in}}(u,\sigma_1)]_\delta$. By Definition 1 and Definition 2, we have:

$$\tilde{d}_{G^{in}}(u,\sigma_i)=\mathbf{d}_u^{\le}[i]=[d_{G^{in}}(u,\sigma_1)]_\delta+\sum_{j=1}^{i-1}\mathbf{p}_u[j] \tag{10}$$

By Definition 3, we have:

$$\begin{aligned}\tilde{d}_{G^{out}}(\mathbf{p}_u,v)+\tilde{d}_{G^{in}}(u,\sigma_1) &= \min_{1\le i\le k}\{\tilde{d}_{G^{out}}(v,\sigma_i)+\sum_{j=1}^{i-1}\mathbf{p}_u[j]\}+\tilde{d}_{G^{in}}(u,\sigma_1)\\ &= \min_{1\le i\le k}\{\tilde{d}_{G^{out}}(v,\sigma_i)+\sum_{j=1}^{i-1}\mathbf{p}_u[j]+[d_{G^{in}}(u,\sigma_1)]_\delta\}\\ &= \min_{1\le i\le k}\{\tilde{d}_{G^{out}}(v,\sigma_i)+\tilde{d}_{G^{in}}(u,\sigma_i)\} \qquad \text{(by Equation (10))}\\ &= \tilde{d}_G(u,v)\ ,\end{aligned} \tag{11}$$

as desired. □

By Lemma 5, to obtain an approximate distance between u and v, it suffices to know the δ-approximate distance encodings of u and v w.r.t. σ in G^{in} and G^{out}, respectively. We show later that, by choosing δ appropriately, the size of the set of all approximate distance encodings $\{\mathbf{d}_u\}_{u\in V(G^{in})}$ of G^{in} is polynomial in $1/\epsilon$. Thus, we can store all these distance encodings in a table, say T_1. Then for each vertex $u\in G^{in}$, we only store a *pointer* to the corresponding entry in the table, which costs only $O(1)$ machine words (instead of $O(1/\epsilon)$ machine words to store the actual distance encoding of u). The same holds for v in graph G^{out}. Furthermore, we precompute distances $\tilde{d}$ from every pair of distance encodings, one from G^{in} and the other from G^{out}, and store the result in a lookup table, say T_2, which costs only $O(\text{poly}(1/\epsilon))$ space. To retrieve the distance from u to v, we simply follow the pointers to access their approximate distance encodings in T_1 and then access the $\tilde{d}_G(u,v)$ in table T_2 in $O(1)$ time. The total space is $O(n+\text{poly}(1/\epsilon))$. Note that this is only for querying distances from a vertex in G^{in} to a vertex in G^{out}; we need to recurse to construct a data structure for all pairs of vertices. Furthermore, we want the space bound to be $\tilde{O}(n(1/\epsilon)^{o(1)})$ rather than $O(n+\text{poly}(1/\epsilon))$, and for this, we need additional ideas. Nevertheless, the fact that the space dependency on $1/\epsilon$ is additive instead of multiplicative is the key to our construction later.

In our oracle construction, sometimes for a given vertex $u\in G^{in}$ and $v\in G^{out}$, we can only obtain approximate distance encodings that are sufficiently close to the true approximate distance encodings of u and v. We show below that we can still recover the approximate distance between u and v. To formally state our result, we need some notation. Given two approximate distance encodings $\mathbf{d}_1$ and $\mathbf{d}_2$ of dimension k, we say that:

$$\mathbf{d}_1\approx_{\delta_1,\delta_2}\mathbf{d}_2 \qquad \text{if } \mathbf{d}_1[1]\approx_{\delta_1}\mathbf{d}_2[1] \text{ and } \mathbf{d}_1[2:k]\approx_{\delta_2}\mathbf{d}[2:k] \tag{12}$$

Definition 5 (Approximate Distance from Approximate Distance Encodings)**.** *Let* $\mathbf{d}_1$ *and* $\mathbf{d}_1$ *be two approximate distance encodings of dimension* k*. We define their distance, denoted by,* $\|\mathbf{d}_1, \mathbf{d}_2\|$*, as follows:*

$$\|\mathbf{d}_1, \mathbf{d}_2\| = \min_{1 \le i \le k} \{\mathbf{d}_1^{\le}[i] + \mathbf{d}_2^{\le}[i]\} \tag{13}$$

If $\mathbf{d}_u$ and $\mathbf{d}_v$ are two δ-approximate distance encodings as in Lemma 5, then $\|\mathbf{d}_u, \mathbf{d}_v\| = \tilde{d}_G(u, v)$ by definition (Equation (8)). We show below that we can recover the approximate distance from u to v from the approximations of their approximate distance encodings.

Lemma 6. *Let* $\mathbf{d}_1$ *and* $\mathbf{d}_2$ *be two approximate distance encodings of dimension* k*, and* $u \in G^{in}$ *and* $v \in G^{out}$ *be two vertices of* G *where* G^{in} *and* G^{out} *are as defined in Lemma 5. Let* $\mathbf{d}_u$ *(*$\mathbf{d}_v$*) be the* δ*-approximate distance encoding of* u *(*v*) w.r.t.* σ *in* G^{in} *(*G^{out}*). Suppose that:*

$$\mathbf{d}_1 \approx_{\delta_1, \delta_2} \mathbf{d}_u \qquad \text{and} \qquad \mathbf{d}_2 \approx_{\delta_1, \delta_2} \mathbf{d}_v$$

Then, $\|\mathbf{d}_1, \mathbf{d}_2\| \approx_{2\delta_1 + 2(k-1)\delta_2} \tilde{d}_G(u, v)$*.*

Proof. By Observation 1, $\sum_{j=2}^{i} \mathbf{d}_1[j] \approx_{(i-1)\delta_2} \sum_{j=2}^{i} \mathbf{d}_u[j]$ for any $i \in [k]$. Thus, by Definition 2 and Observation 1, $\mathbf{d}_1^{\le}[i] \approx_{\delta_1 + (i-1)\delta_2} \mathbf{d}_u^{\le}[i] = \tilde{d}_{G^{in}}(u, \sigma_i)$. It follows that $\mathbf{d}_1^{\le} \approx_{\delta_1 + (k-1)\delta_2} \tilde{d}_{G^{in}}(u, \cdot)$; here $\tilde{d}_{G^{in}}(u, \cdot)$ is the k-dimensional vector whose i-th component is $\tilde{d}_{G^{in}}(u, \sigma_i)$. By the same argument, we have that $\mathbf{d}_2^{\le} \approx_{\delta_1 + (k-1)\delta_2} \tilde{d}_{G^{out}}(v, \cdot)$. The lemma then follows from Claim 1 and Equation (8). □

We close this section by showing that shortest path separators in planar graphs are single-crossing cycles. We rely on the well-known property that each shortest path separator cycle consists of two shortest paths and a single edge.

Lemma 7. *If a simple cycle* C *of a plane graph* $G(V, E, w)$ *is composed of two shortest paths and a single edge, then* C *is single-crossing.*

Proof. Suppose that $C = Q_1 \circ e \circ Q_2$ where Q_1, Q_2 are shortest paths of G, and e is a single edge. Let $u \in G^{in}$ and $v \in G^{out}$ be two vertices separated by C, where G^{in} (G^{out}) be the subgraph of G induced by vertices inside (outside) or on C. Let $SP(u, v, G)$ be a shortest path from u to v in G that crosses C a minimum number of times. (If u and v are both on C, we regard u as inside C and v as outside C.) By Jordan curve theorem, $SP(u, v, G)$ must cross C by an odd number of times. If $SP(u, v, G)$ crosses C at least 3 times, then $SP(u, v, G)$ must cross, say Q_1, at least twice. See Figure 3(b) for an illustration. Let a and b be the first and the last crossing points on the path $SP(u, v, G)$ oriented from u to v. By replacing the subpath $SP(u, v, G)[a, b]$ by $Q_1[a, b]$, we obtain another path from u to v while the number of crossing is reduced by at least one, contradicting that $SP(u, v, G)$ has a minimum number of crossings. Thus, C is single-crossing. □

3.3 Approximate Pattern Composition

We define patterns induced by approximate patterns. This definition is similar to the patterns induced by patterns of [FHMWN21]. Recall the distance between a pattern and a vertex is defined in Definition 3.

Definition 6 (Pattern Induced by an Approximate Distance)**.** *Let* σ *be a sequence of* k *vertices on the boundary of a face* f *of a plane graph* G*. Let* $\mathbf{p}$ *be an approximate pattern (w.r.t. some sequence of vertices that may be different from* σ*). The pattern induced by* $\mathbf{p}$ *w.r.t.* σ *is a* $(k-1)$*-dimensional vector* $\widehat{\mathbf{p}}$ *where* $\widehat{\mathbf{p}}[i] = \tilde{d}_G(\mathbf{p}, \sigma_{i+1}) - \tilde{d}_G(\mathbf{p}, \sigma_i)$ *for every* $i \in [k-1]$*.*

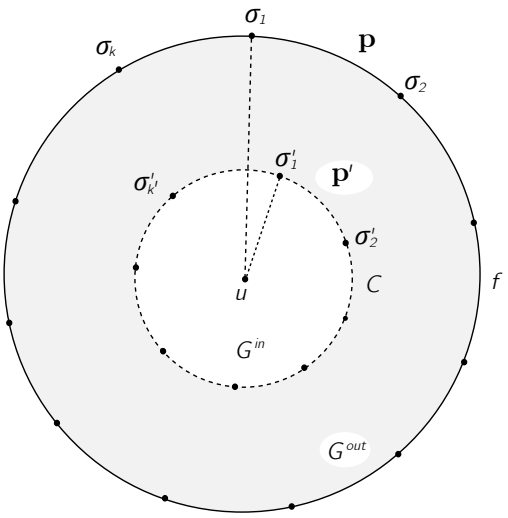


Figure 4: Two patterns $\mathbf{p}'$ and $\mathbf{p}$ of u w.r.t. two sequences σ' and σ in G^{in} and G, respectively. Lemma 8 implies that the pattern induced by $\mathbf{p}'$ in G^{out} is a $k\delta$-approximation of $\mathbf{p}$.

In the following lemma, we show that, given a face f of a plane graph, and a simple-separating cycle C such that f lies outside C, for any vertex u inside C, we can construct a new pattern from u w.r.t. some vertex sequence σ in f by *composing* two patterns: a pattern from u w.r.t. some vertex sequence σ' on C, and the pattern w.r.t. σ induced by a pattern w.r.t. σ'. The new pattern is may not be the same as the approximate pattern $\mathbf{p}$ of u w.r.t. σ in G as defined in Definition 1, but it is close to $\mathbf{p}$. See Figure 4 for an illustration.

Lemma 8 (Pattern Composition)**.** *Let σ be a sequence of k vertices on the boundary of a face f of a plane graph G, and σ' be a sequence of k' vertices on a single-crossing simple cycle C such that $f \subseteq \mathsf{Ext}(C)$. Let G^{in} (G^{out}) be the subgraph of G induced by vertices inside (outside) or on C. Let u be a vertex in G^{in}, $\mathbf{p}'$ be a pattern of u in G^{in} w.r.t. σ', and $\mathbf{p}$ be a pattern of u in G w.r.t. σ. Let $\widehat{\mathbf{p}}'$ be the pattern induced by $\mathbf{p}'$ in G^{out}. Then, it holds that $\widehat{\mathbf{p}}' \approx_{k\delta} \mathbf{p}$.*

Proof. By Definition 6, for any $i \in [k-1]$, we have:

$$\begin{aligned}
\widehat{\mathbf{p}}'[i] &= \tilde{d}_{G^{out}}(\mathbf{p}', \sigma_{i+1}) - \tilde{d}_{G^{out}}(\mathbf{p}', \sigma_i) \\
&= (\tilde{d}_G(u, \sigma_{i+1}) - \tilde{d}_{G^{in}}(u, \sigma'_1)) - (\tilde{d}_G(u, \sigma_i) - \tilde{d}_{G^{in}}(u, \sigma'_1)) \qquad \text{(by Item (2) in Lemma 5)} \\
&= \tilde{d}_G(u, \sigma_{i+1}) - \tilde{d}_G(u, \sigma_i).
\end{aligned}$$

By Equation (3), we have:

$$\begin{aligned}
\tilde{d}_G(u, \sigma_{i+1}) - \tilde{d}_G(u, \sigma_i) &\le [d_G(u, \sigma_{i+1}) - d_G(u, \sigma_i)]_\delta + (i+1)\delta \\
\tilde{d}_G(u, \sigma_{i+1}) - \tilde{d}_G(u, \sigma_i) &\ge [d_G(u, \sigma_{i+1}) - d_G(u, \sigma_i)]_\delta - (i+1)\delta \ .
\end{aligned} \tag{14}$$

Thus, $\widehat{\mathbf{p}}'[i] \approx_{(i+1)\delta} \mathbf{p}[i]$, which implies the lemma, since $i \le k-1$. □

3.4 A Weaker Oracle: Proof of Theorem 4

In this section, we construct an oracle as claimed in Theorem 4. The construction is described in Section 3.4.1, ignoring implementation issues. The analysis of query time and space is presented in Section 3.4.2 and Section 3.4.3. Finally, in Section 3.4.4, we discuss the implementation.

3.4.1 The Construction

First, we review the standard recursive decomposition using shortest path separators. Let T be a shortest path tree of G rooted at a chosen vertex r. We assume w.l.o.g that G is triangulated. A shortest path separator is a fundamental cycle C of T that comprises of two shortest paths rooted at r and an edge e connecting two other endpoints of the two paths. By Lemma 7, C is single-crossing. It was known that, for any given non-negative weight function $w_V : V \to \mathbb{R}^+$, there is a shortest path separator such that the total weight of vertices strictly inside or outside C is at most $\frac{2w_V(G)}{3}$ where $w_V(G) = \sum_{u \in V} w_V(u)$.

We use the shortest path separators to recursively separate G into *regions* as follows. Initially every vertex of G is marked. Starting from G, we construct a shortest path separator C such that the total number of vertices strictly inside/outside C is at most $\frac{2n}{3}$, obtaining two regions sharing the same boundary C. We then distribute marked vertices on C evenly to the two regions, so that each region gets at most $2n/3$ marked vertices. Some vertices on C in one region get unmarked because they are marked vertices in the other region. Next, pick any region R that has at least $\lambda > 0$ marked vertices for some parameter λ (defined in the algorithm below), we separate R using the shortest path separator C_R into two smaller regions R_1, R_2. The separator C_R is chosen to balance the number of holes or the number of marked vertices inside child regions. In particular, if R has exactly 5 holes, we choose C_R such that child regions R_1, R_2 each has at most $\lfloor \frac{2 \cdot 5}{3} \rfloor + 1 = 4$ holes; this can be done by assigning weights to vertices on the hole appropriately. Otherwise, we choose C_R such that the number of marked vertices of each child region is at most $2/3$ the number of marked vertices of R; again, we distribute marked vertices on C_R evenly to both sides. It follows from the construction that each region has at most 5 holes.

Let $\mathcal{T}$ be the recursion tree induced by the recursive decomposition of G. Each node of $\mathcal{T}$ is associated with a region resulting from the decomposition. It is well known that:

Lemma 9. *$\mathcal{T}$ has depth $O(\log(n))$ and $O(\frac{n}{\lambda})$ nodes that can be constructed in $O(n \log n)$ time.*

By construction, the internal vertices of a region are all marked. Unmarked vertices are on the boundaries of the holes. We say that a region A is an ancestor of a region R if A is associated with an ancestor node of R's node in $\mathcal{T}$. Clearly, $R \subseteq A$. Each region R, except G, has a special hole h whose boundary is the separating cycle C_P of its parent region P. We call h the *parental hole* of R (see Figure 6(a)). (The parental hole of R could be the infinite face of R in the planar embedding inherited from G.) The boundary of h is called the *parental boundary* of R. In the following, we construct our distance oracle in four steps. Figure 5 and Figure 6 illustrates each step of the construction.

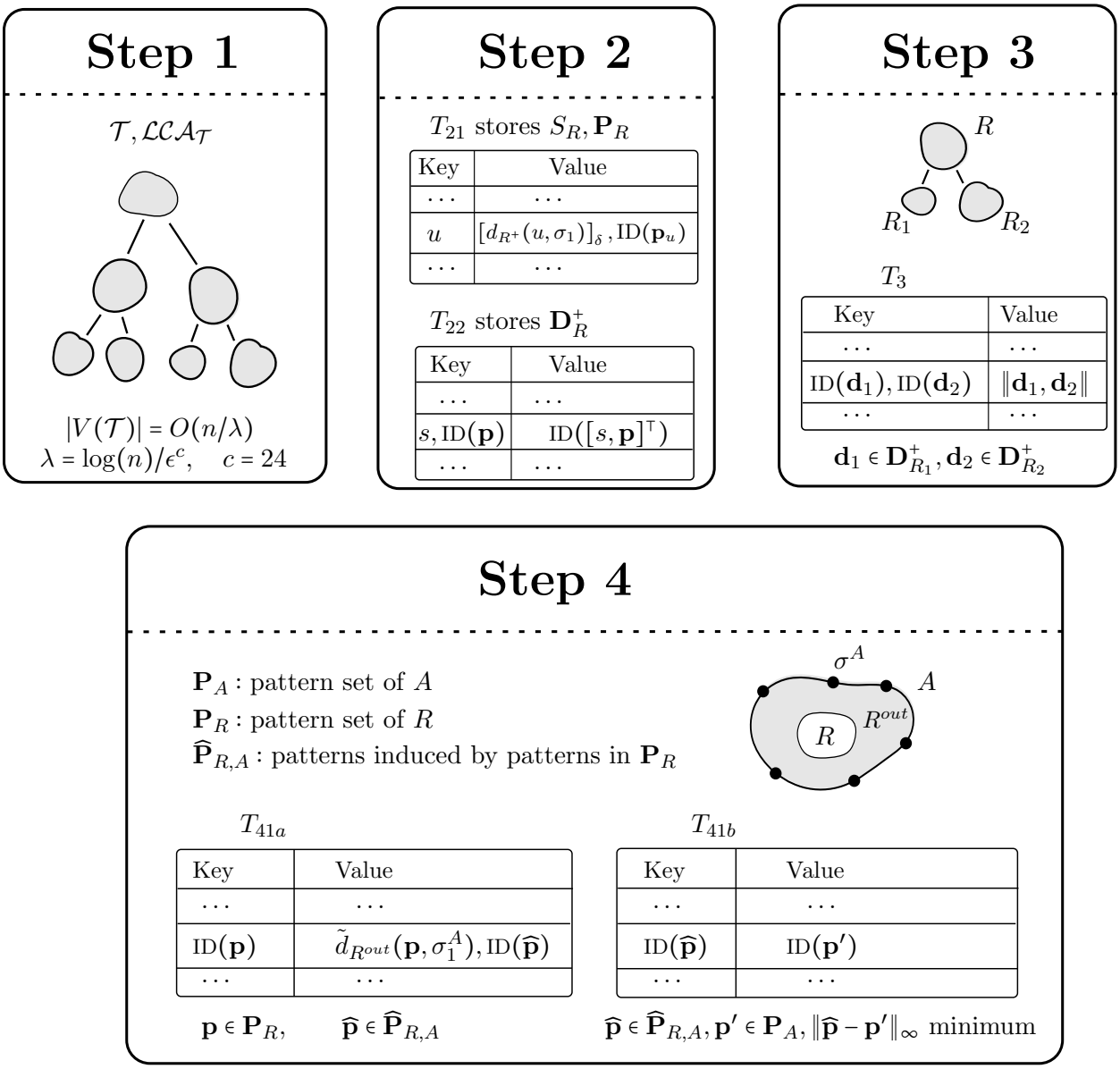


Figure 5: Tables and data structures used in each step of the oracle construction.

Distance Oracle Construction:

Step 1. Construct a recursive decomposition $\mathcal{T}$ of G with $\lambda = \frac{\log n}{\epsilon^c}$ for $c = 24$, and a lowest common ancestor data structure $\mathcal{LCA}_{\mathcal{T}}$ for $\mathcal{T}$. For each shortest path separator C in $\mathcal{T}$, we designate $O(1/\epsilon)$ portals such that the distance between any consecutive portals is ϵD; if there is an edge of length more than ϵD, we will subdivide the edge using portals. For each region R associated with a node of $\mathcal{T}$, let $M(R)$ be the set of marked vertices of R.

Step 2. Let $\delta = \epsilon^3 D$. For each region R associated with a non-root node of $\mathcal{T}$, let R^+ be obtained from R by *filling all the holes of R, except the parental hole of R.* That is, for each non-parental hole h, we add the edges and vertices of G inside h to R. We form a sequence σ of the portals on the parental boundary of R by ordering the portals clockwise order. (See Figure 6(b).) For each vertex $u \in M(R)$, let $\mathbf{p}_u$ be the δ-approximate pattern of u w.r.t. σ *in graph* R^+. Let $\mathbf{P}_R = \{\mathbf{p}_u : u \in M(R)\}$ and $S_R = \{[d_{R^+}(u, \sigma_1) : u \in M(R)]_\delta\}$. Let $\mathbf{D}^+_R = S_R \times \mathbf{P}_R$ be a set k-dimensional vectors whose first coordinates are scalars in S_R; here $k = |\sigma|$. We assign unique IDs to vectors in $\mathbf{P}_R$ and $\mathbf{D}^+_R$. At node R, we store S and $\mathbf{P}_R$ in a table T_{21} and $\mathbf{D}^+_R$ in a table T_{22}. Given u, we can query $[d_{R^+}(u, \sigma_1)]_\delta$ and the ID of $\mathbf{p}_u$ in T_{21} in $O(1)$ time. Given a scalar $s \in S_R$ and an ID of a vector $\mathbf{p} \in \mathbf{P}_R$, we can query the ID of the corresponding vector $[s, \mathbf{p}]^\top \in \mathbf{D}^+_R$ in T_{22} in $O(1)$ time.

Step 3. For each non-leaf node R with two child regions R_1 and R_2. Observe that $R_1^+ \cup R_2^+ = G$ by definition, and that $R_1^+ \cap R_2^+ = C_R$ where C_R is the shared parental boundary of R_1 and R_2. (See Figure 6(a).) Let σ be the sequence of portal vertices on C_R. For each pair of distance encodings $\mathbf{d}_1 \in \mathbf{D}^+_{R_1}$ and $\mathbf{d}_2 \in \mathbf{D}^+_{R_2}$, we compute $\|\mathbf{d}_1, \mathbf{d}_2\|$ using Definition 5, and store it in a lookup table T_3, indexed by the IDs of $\mathbf{d}_1$ and $\mathbf{d}_2$. Thus, given the IDs of $\mathbf{d}_1$ and $\mathbf{d}_2$, we can look up the distance $\|\mathbf{d}_1, \mathbf{d}_2\|$ in T_3 in $O(1)$ time.

Step 4. For each *leaf region* R, we store the following information at the node of R.

(4.1) For each ancestor region A of R, let R^{out} be the graph induced by the edge set $(E(A^+) \setminus E(R^+)) \cup \partial R^+$. Let σ^A be the sequence of portals of the parental boundary of A; see Figure 6(c). Let $\widehat{\mathbf{P}}_{R,A}$ be the set of patterns w.r.t. σ^A induced by δ-approximate patterns in $\mathbf{P}_R$ in graph R^{out}.

- (a) We store in a table T_{41a} for each pattern $\mathbf{p} \in \mathbf{P}_R$ the distance $\tilde{d}_{R^{out}}(\mathbf{p}, \sigma_1^A)$, and the ID of the pattern $\widehat{\mathbf{p}} \in \widehat{\mathbf{P}}_{R,A}$ induced by $\mathbf{p}$. Given the ID of $\mathbf{p}$, we can access the corresponding entry of T_{41a} in $O(1)$ time.
- (b) Recall that $\mathbf{P}_A$ is the set of δ-approximate patterns of A computed in Step 2. For each induced pattern $\widehat{\mathbf{p}} \in \widehat{\mathbf{P}}_{R,A}$, we store in a table T_{41b} the ID of the pattern $\mathbf{p}' \in \mathbf{P}_A$ closest to $\widehat{\mathbf{p}}$. That is, $\mathbf{p}'$ minimize $\|\widehat{\mathbf{p}} - \mathbf{p}'\|_\infty$ over all patterns in $\mathbf{P}_A$. We can access entries of T_{41b} in $O(1)$ time given the ID of $\widehat{\mathbf{p}}$.

(4.2) We construct a graph R', called a *contracted filled graph*, from R as follows. (See Figure 6(d).) For each hole h of R, let P_h be the set of portals on the boundary of h constructed in Step 1. Let G^h be the graph induced by edges of G inside and on the boundary of h. Let K^h be the distance preserving minor of P_h in G^h. Next, let $\overline{R}$ be obtained from R by contracting each unmarked vertices on the boundaries of the holes of R to the nearest portal. The weight of new edges between two portals on the boundary of $\overline{R}$ is their distance in R. For every set of parallel edges whose one endpoint is non-portal, we keep the minimum-weight edge. Let:

$$R' = \overline{R} \bigcup (\cup_h \text{ is a hole of } {}_R K^h). \tag{15}$$

Observe that R' is planar. We apply Theorem 3 to construct an exact distance oracle $\mathcal{E}_R$ for R' with:

- (a) $|V(R')|^{1+o(1)}$ space and $\log^{2+o(1)}(|V(R')|)$ query time or
- (b) $|V(R')| \log^{2+o(1)}(|V(R')|)$ space and $|V(R')|^{o(1)}$ query time.

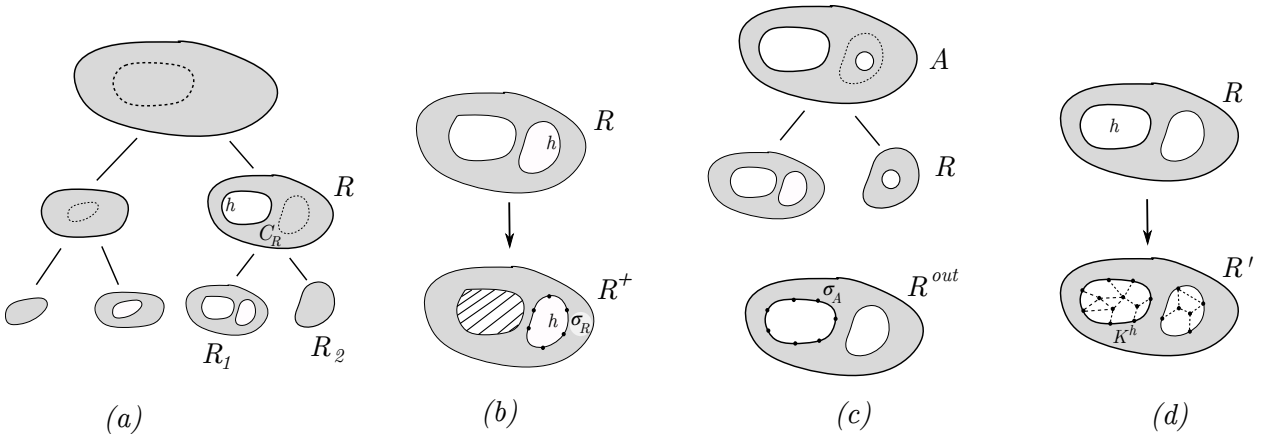
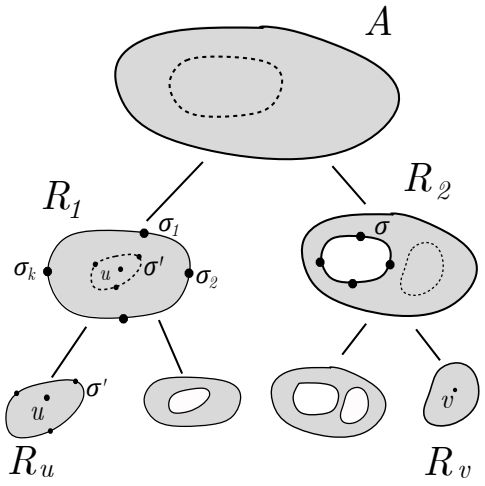


Figure 6: (a) A recursive decomposition tree $\mathcal{T}$; h is the parental hole of R and the boundary of h is the parental boundary of R; separator C_R separates R into two children R_1 and R_2. (b) R has two holes, where h is the parental hole; R^+ is obtained from R by filling up the non-parental hole; σ^R is the sequence of vertices on the boundary of h. (c) A region R and its parent A (top) and the subgraph R^{out} induced by the edge set $(E(A^+) \setminus E(R^+)) \cup \partial R^+$ (bottom). (d) The contracted filled graph R' of R is obtained by first contracting some edges on the boundary of R to obtain $\overline{R}$ and then filling each hole h in $\overline{R}$ by a graph K^h.

Here we briefly describe the idea of each step in the construction. Step 1 constructs the recursive decomposition $\mathcal{T}$ and a basic data structure to navigate $\mathcal{T}$. Step 2 stores the set of approximate patterns of each region $\mathbf{P}_R$ and an *extended* set of approximate distance decodings $\mathbf{D}_R^+$. The set of approximate distance decodings of R is $\mathbf{D}_R = \{\mathbf{d}_u : u \in M(R)\}$, which is a subset of $\mathbf{D}_R^+$. The reason we store the extended set is that we could not query the approximate distance decoding $\mathbf{d}_u$ of u from the information stored at the leaf region containing u due to the pattern composition step. The pattern composition only gives an approximate distance decoding $\tilde{d}_u$ that is close to $\mathbf{d}_u$ (in ℓ_∞ norm) and $\tilde{d}_u \in \mathbf{D}_R^+$. Step 3 precomputes the distance of every pair of approximate distance decodings in a table so that we can look up the distance in $O(1)$ time during the query stage. Step 4(1) implements the pattern composition discussed in Section 3.3, and Step 4(2) constructs an exact distance oracle for each leaf region.

Answering a query. Let u and v be two given vertices in the query. First, we find two leaf regions R_u and R_v containing u and v such that $u \in M(R_u)$ and $v \in M(R_v)$. If $R_u = R_v = R$, we query the distance between u and v in $\mathcal{E}_R$, denoted by $\mathcal{E}_R(u, v)$. We then return $\mathcal{E}_R(u, v) + 2\epsilon D$ as the approximate distance. (We add $+2\epsilon D$ to $\mathcal{E}_R(u, v)$ because of the contraction in the construction of $\overline{R}$.)

We now assume that $R_u \neq R_v$. Let $A = \mathcal{LCA}_\mathcal{T}(R_u, R_v)$ be the lowest common ancestor region of R_u and R_v. Let R_1 and R_2 be two child regions of A where R_1 (R_2) is an ancestor of R_u (R_v). Let σ be the sequence of vertices on the parental boundary of R_1 and R_2. By construction, R_1 and R_2 share the same parental boundary. Next, we



use the following lemma, whose proof is deferred to the end of this section.

Lemma 10. *Given $u \in R_u$, we can query the ID of a vector $\tilde{\mathbf{d}}_u \in \mathbf{D}^+_{R_1}$ in $O(1)$ time such that $\tilde{\mathbf{d}}_u \approx_{0,2k\delta} \mathbf{d}_u$ where $\mathbf{d}_u$ is the approximate distance encoding of u w.r.t. σ in R^+_1.*

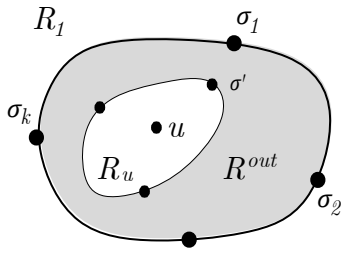
We apply Lemma 10 to query the IDs of approximate distance encodings $\tilde{\mathbf{d}}_u \in \mathbf{D}^+_{R_1}$ and $\tilde{\mathbf{d}}_v \in \mathbf{D}^+_{R_2}$ of u and v, respectively. Given the two approximate distance encoding IDs, we query table T_3 of A in $O(1)$ time to obtain $\|\tilde{\mathbf{d}}_u, \tilde{\mathbf{d}}_v\|$. Finally, we return:

$$\|\tilde{\mathbf{d}}_u, \tilde{\mathbf{d}}_v\| + c_0 \epsilon D \qquad \text{for some constant } c_0 \text{ chosen later} \tag{16}$$

as an approximate distance between u and v of our oracle.

Proof of Lemma 10. Let R^{out}_u be the graph induced by the edge set $(E(R^+_1) \setminus E(R^+_u)) \cup \partial R^+_u$. ($R^{out}_u$ is exactly the graph in Step 4(1) obtained from the ancestor region R_1 and the leaf region R_u.)

Recall each vector $\tilde{\mathbf{d}}_u \in \mathbf{D}^+_{R_1}$ can be written as $\tilde{\mathbf{d}}_u = \left[\left[d_{R^+_1}(u, \sigma_1)\right]_\delta, \mathbf{p}\right]^\dagger$ for some approximate pattern $\mathbf{p} \in \mathbf{P}_{R_1}$. (Pattern $\mathbf{p}$ might be different from $\mathbf{p}_u$, the approximate pattern of u in $\mathbf{P}_{R_1}$.) Furthermore, we can query the ID of $\tilde{\mathbf{d}}_u$ from table T_2 stored at R_1 constructed in Step 2. To do so, we need to know the value of $\left[d_{R^+_1}(u, \sigma_1)\right]_\delta$ and the ID of $\mathbf{p}$.



Recall by Remark 2, $\left[d_{R^+_1}(u, \sigma_1)\right]_\delta = \tilde{d}_{R^+_1}(u, \sigma_1)$. We query $\tilde{d}_{R^+_1}(u, \sigma_1)$ as follows. Let σ' be the sequence of portals in the parental hole of R_u. We use u to query table T_2 of R_u (constructed in Step 2) to get $\left[d_{R^+_u}(u, \sigma'_1)\right]_\delta$ and the approximate pattern of u in graph R^+_u, denoted by $\mathbf{p}'_u$. (We reserve $\mathbf{p}_u$ notation for the approximate pattern of u in R^+_1.) Next, we use the ID of $\mathbf{p}'_u$ to query $\tilde{d}_{R^{out}}(\mathbf{p}'_u, \sigma_1)$ in table T_{41a} (stored at R_u) constructed in Step 4(1a). The total query time is $O(1)$. Observe that:

$$\begin{aligned}
&\left[d_{R^+_u}(u, \sigma'_1)\right]_\delta + \tilde{d}_{R^{out}}(\mathbf{p}'_u, \sigma_1) \\
&= \tilde{d}_{R^+_u}(u, \sigma'_1) + \tilde{d}_{R^{out}}(\mathbf{p}'_u, \sigma_1) \qquad \text{(by Remark 2)} \\
&= \tilde{d}_{R^+_1}(u, \sigma_1) \qquad \text{(by Item (2) of Lemma 5)}
\end{aligned}$$

which will give us the value of $\left[d_{R^+_1}(u, \sigma_1)\right]_\delta$.

Next, we query the ID of $\mathbf{p}$ as follows. First, we query the ID of $\mathbf{p}'_u$ stored in table T_2 of R_u as above. Then, we use the ID of $\mathbf{p}'_u$ to query the ID of an approximate pattern $\mathbf{p}$ in table T_{41b} for ancestor R_1 of R_u constructed in Step 4(1b) in $O(1)$ time.

Now using $\left[d_{R^+_1}(u, \sigma_1)\right]_\delta$ and the ID of $\mathbf{p}$, we query the ID of $\tilde{\mathbf{d}}_u \in \mathbf{D}^+_{R_1}$ from table T_2 stored at R_1 constructed in Step 2. Note that $\tilde{\mathbf{d}}_u[1] = \left[d_{R^+_1}(u, \sigma_1)\right]_\delta = \tilde{d}_{R^+_1}(u, \sigma_1)$ and $\tilde{\mathbf{d}}_u[2:k] = \mathbf{p}$.

To complete the proof of the lemma, we show that $\tilde{\mathbf{d}}_u \approx_{0,2k\delta} \mathbf{d}_u$. Observe that $\tilde{\mathbf{d}}_u[1] = \left[d_{R^+_1}(u, \sigma_1)\right]_\delta = \mathbf{d}_u[1]$. Thus, by Equation (12), it remains to show that $\tilde{\mathbf{d}}_u[2:k] \approx_{2k\delta} \mathbf{d}_u[2:k]$ which is equivalent to showing that $\mathbf{p} \approx_{2k\delta} \mathbf{p}_u$.

By construction in Step 4(1b), $\mathbf{p}$ is the closest pattern of the pattern, say $\widehat{\mathbf{p}}'_u \in \widehat{\mathbf{P}}_{R_u,R_1}$, induced by $\mathbf{p}'_u$. By Lemma 8, $\widehat{\mathbf{p}}'_u \approx_{k\delta} \mathbf{p}_u$; that is, $\|\widehat{\mathbf{p}}'_u, \mathbf{p}_u\|_\infty \le k\delta$. Since $\mathbf{p}$ is closest to $\widehat{\mathbf{p}}'_u$ in ℓ_∞ norm, we have:

$$\|\mathbf{p}, \widehat{\mathbf{p}}'_u\|_\infty \le \|\widehat{\mathbf{p}}'_u, \mathbf{p}_u\|_\infty \le k\delta \tag{17}$$

Since $\mathbf{p} \approx_{k\delta} \widehat{\mathbf{p}}'_u$ and $\widehat{\mathbf{p}}'_u \approx_{k\delta} \mathbf{p}_u$, it follows from Observation 1 that $\mathbf{p} \approx_{2k\delta} \mathbf{p}_u$, as desired. □

3.4.2 Query time and stretch analysis

Query time analysis. Let u and v be two querying vertices. If $R_u = R_v = R$, the algorithm queries $\mathcal{E}_R$ in time:

$$\begin{cases} \log^{2+o(1)}(|V(R_u)|) = \log^{2+o(1)}(1/\epsilon) + \log^{2+o(1)}\log(n) & \text{in Regime 4(2a)} \\ |V(R_u)|^{o(1)} = \log^{o(1)}(n)(1/\epsilon)^{o(1)} & \text{in Regime 4(2b)} \end{cases} \tag{18}$$

We now consider the case $R_u \neq R_v$. The time to compute $A = \mathcal{LCA}_{\mathcal{T}}(R_u, R_v)$ is $O(1)$. Let R_1 and R_2 be two child regions of A where R_1 (R_2) is an ancestor of R_u (R_v). By Lemma 10, the time to query the IDs of approximate distance encodings $\tilde{\mathbf{d}}_u \in \mathbf{D}^+_{R_1}$ and $\tilde{\mathbf{d}}_v \in \mathbf{D}^+_{R_2}$ of u and v, respectively, is $O(1)$. Next, we query table T_3 of A in $O(1)$ time to obtain $\|\tilde{\mathbf{d}}_u, \tilde{\mathbf{d}}_v\|$. Finally, we return the approximate distance in Equation (16) in $O(1)$ time.

In summary, the total query time is dominated by the query time in Equation (18), which implies the query time claimed in Theorem 4.

Stretch analysis and choosing c_0 in Equation (16). If u and v are in the same leaf region R, then the exact distance query in $\mathcal{E}_R$ will return the exact distance between u to v in R'. We show in the following that this distance approximates the true distance in G.

Lemma 11. *$d_G(u,v) - 2\epsilon D \le d_{R'}(u,v) \le d_G(u,v) + 10\epsilon D$.*

Proof. We reuse the notation in Step 4(2) here. As the distance of each unmarked vertex to the nearest portal is at most ϵD, by the triangle inequality, for any two marked vertices u and v, it holds that:

$$d_R(u,v) - 2\epsilon D \le d_{\overline{R}}(u,v) \le d_R(u,v) \tag{19}$$

Let $\overline{G}$ be obtained from $\bar{R}$ by filling each hole exactly. That is:

$$\overline{G} = \overline{R} \bigcup (\cup_h \text{ is a hole of } {}_R G^h).$$

It follows from Equation (19) that:

$$d_G(u,v) - 2\epsilon D \le d_{\overline{G}}(u,v) \le d_G(u,v) \tag{20}$$

Since R' is obtained from $\overline{R}$ by approximately filling 5 holes through portals, and since the distance between any two consecutive portals is ϵD, by the triangle inequality, $d_{\overline{G}}(u,v) \le d_{R'}(u,v) \le d_{\overline{G}}(u,v) + 10\epsilon D$. Combined with Equation (20), we have $d_G(u,v) - 2\epsilon D \le d_{R'}(u,v) \le d_G(u,v) + 10\epsilon D$ as claimed. □

The distance that the oracle returns is $\mathcal{E}_R(u,v)+2\epsilon D = d_{R'}(u,v)+2\epsilon D$, which is at least $d_G(u,v)$ and at most $d_G(u,v)+12\epsilon D$ by Lemma 11.

It remains to consider the case $R_u \neq R_v$. Observe that for every region R in $\mathcal{T}$, the sequence σ of vertices forming from the portals of the parental boundary of R is a ϵD-cover of the boundary. Thus, $\tau = \epsilon D$. Since every shortest path has length at most D, there are at most $k = O(D/\tau) = O(1/\epsilon)$ vertices in σ.

In Equation (16), the oracle computes and returns $\|\tilde{\mathbf{d}}_u, \tilde{\mathbf{d}}_v\| + c_0\epsilon D$. The following lemma, whose proof is deferred to the end of this section, will help us bound the stretch.

Lemma 12. *$\|\tilde{\mathbf{d}}_u, \tilde{\mathbf{d}}_v\| \approx_{(4k^2-2k)\delta+2\tau} d_G(u,v)$. In particular, when $k = O(1/\epsilon), \delta = \epsilon^3 D$ and $\tau = \epsilon D$, then $\|\tilde{\mathbf{d}}_u, \tilde{\mathbf{d}}_v\| \approx_{c_0\epsilon D} d_G(u,v)$ for some constant c_0.*

We choose c_0 in Equation (16) to be the value in Lemma 12. It follows that:

$$d_G(u,v) \leq \|\tilde{\mathbf{d}}_u, \tilde{\mathbf{d}}_v\| + c_0\epsilon D \leq d_G(u,v) + 2c_0\epsilon D \ .$$

That is, the additive stretch of our oracle is $+O(\epsilon D)$. We can get back additive stretch $+\epsilon D$ by scaling ϵ.

Proof of Lemma 12. By construction, $R_1^+ \cup R_2^+ = G$. We denote $G^{in} = R_1^+$ and $G^{out} = R_2^+$. The cycle separating G^{in} and G^{out} is single-crossing by Lemma 7. Recall that σ is the sequence of at most k portals on the (shared) parental boundary of R_1 and R_2. Let $\mathbf{p}_u$ ($\mathbf{d}_u$) and $\mathbf{p}_v$ ($\mathbf{d}_v$) be the δ-approximate patterns (distance encodings) of u and v w.r.t. σ in G^{in} and G^{out}, respectively. By Lemma 10, we have:

$$\tilde{\mathbf{d}}_u \approx_{0,2k\delta} \mathbf{d}_u \qquad \text{and} \qquad \tilde{\mathbf{d}}_v \approx_{0,2k\delta} = \mathbf{d}_v$$

By applying Lemma 6 with $\delta_1 = 0$ and $\delta_2 = 2k\delta$, $\|\tilde{\mathbf{d}}_u, \tilde{\mathbf{d}}_v\| \approx_{4(k-1)k\delta} \tilde{d}_G(u,v)$. By Item (1) in Lemma 5, $\tilde{d}_G(u,v) \approx_{2k\delta+2\tau} d_G(u,v)$. It follows from Observation 1 that $\|\tilde{\mathbf{d}}_u, \tilde{\mathbf{d}}_v\| \approx_{(4k^2-2k)\delta+2\tau} d_G(u,v)$ as desired. □

3.4.3 Space analysis

First, we bound the number of δ-approximate patterns w.r.t. sequence σ_H of at most $k = O(1/\epsilon)$ vertices in a graph H arising during the construction of the oracle; H could be R^+ in Step 2, or R^{out} in Step 4(1). Recall that we set $\delta = \epsilon^3 D$ in Step 2. Since the distance between σ_{i+1} and σ_i is at most ϵD by construction for every $i \in [k-1]$, it follows from the triangle inequality that $-\epsilon D \leq d_G(u,\sigma_{i+1}) - d_G(u,\sigma_i) \leq \epsilon D$. That is, the constant g in Lemma 4 satisfies $g \leq \frac{\epsilon D}{\delta} \leq 1/\epsilon^2$. Let $\mathbf{P}_H$ be the set of all approximate patterns of H. By Lemma 4, we have:

$$|\mathbf{P}_H| = O((kg)^3) = O(1/\epsilon^9) \ . \tag{21}$$

In the next four lemmas, we bound the space of each step of the oracle construction. (See Figure 5 for an illustration.) Recall that $c = 24$ specified in Step 1.

Lemma 13. *The total space of Step 1 is $O(\frac{n\epsilon^{c-1}}{\log n})$.*

Proof. Since $\lambda = \frac{\log(n)}{\epsilon^c}$, $\mathcal{T}$ has $O(\frac{n\epsilon^c}{\log n})$ nodes by Lemma 9. Thus, the total space to store all portals in Step 1 is $O(\frac{n\epsilon^c}{\log n} \cdot (1/\epsilon)) = O(\frac{n\epsilon^{c-1}}{\log n})$. The LCA data structure $\mathcal{LCA}_{\mathcal{T}}$ (see, e.g., [BFC00]) has space $O(|V(\mathcal{T})|) = O(\frac{n\epsilon^c}{\log n})$, which implies the lemma. □

Lemma 14. *The total space of Step 2 is* $O(\frac{n\epsilon^{c-13}}{\log n})$.

Proof. Observe that there are $\frac{2D}{\delta} = O(1/\epsilon^3)$ different values of $\left[d_{R_1^+}(u,\sigma_1)\right]_\delta$ considered in Step 2. This is because the subgraph associated every node of $\mathcal{T}$ has a diameter at most $2D$ due to the special structure of regions separated by shortest path separators: the shortest path trees of these subgraphs are subtrees of T, the shortest path tree of G rooted at r. Since the total number of patterns for each region is $O(1/\epsilon^9)$ by Equation (21), the total number of approximate distance encodings is $O(1/\epsilon^{12})$. Since each distance encoding has size $O(1/\epsilon)$, the space of table T_{12} and T_{22} in Step 2 is $O(1/\epsilon^{13})$. It follows that the total space in Step 2 is $O(\frac{n\epsilon^c}{\log n}(1/\epsilon^{13})) = O(\frac{n\epsilon^{c-13}}{\log n})$. □

Lemma 15. *The total space of Step 3 is* $O(\frac{n\epsilon^{c-24}}{\log n})$.

Proof. The analysis of Step 2 in Lemma 14 shows that the number of approximate distance encodings of each region is $O(1/\epsilon^{12})$. Thus, the space of table T_3 is $O((1/\epsilon^{12})^2) = O(1/\epsilon^{24})$. It follows that the total space of Step 3 is $O(\frac{n\epsilon^c}{\log n}(1/\epsilon^{24})) = O(\frac{n\epsilon^{c-24}}{\log n})$. □

In Step 4, we have two different regimes, namely Regime 4(2a) and Regime 4(2b), for the choice of the exact distance oracle $\mathcal{E}_R$ in Step 4(2). Our analysis considers each regime separately.

Lemma 16. *The total space of Step 4 is* $n(\log(n)1/\epsilon)^{o(1)}$ *for Regime 4(2a) and* $n(\log^{2+o(1)}(\log n) + \log^{2+o(1)}(1/\epsilon))$ *for Regime 4(2b).*

Proof. We consider each substep separately. We assume that $1/\epsilon = n^{O(1)}$; otherwise, applying the exact oracle in Theorem 3 gives the desired bounds in Theorem 1. Thus, a value of size $\text{poly}(1/\epsilon)$ costs $O(1)$ words of space to store.

Step 4(1) We observe by Equation (21) that $|\mathbf{P}_R| = O(1/\epsilon^9)$ and $|\widehat{\mathbf{P}}_{R,A}| = O(1/\epsilon^9)$. Thus the ID of each pattern in $\mathbf{P}_R$ and $\widehat{\mathbf{P}}_{R,A}$ only costs $O(1)$ words. Storing $\widehat{\mathbf{P}}_{R,A}$ requires $O(1/\epsilon^{10})$ words of space since each pattern has size $O(1/\epsilon)$. In Step 4(1a), the total space of T_{41a} is $O(|\mathbf{P}_R|) = O(1/\epsilon^9)$. In Step 4(1b), the total space of T_{41b} is also $O(|\widehat{\mathbf{P}}_{R,A}|) = O(1/\epsilon^9)$. Since each leaf region R has $O(\log n)$ ancestors by Lemma 9, the total space of Step 4(1) is $O(\frac{n\epsilon^c}{\log n}(1/\epsilon^9)\log(n)) = O(n\epsilon^{c-9}) = O(n)$.

Step 4(2) Observe that $|M(R)| = \Theta(\lambda) = \Theta(\log(n)/\epsilon^c)$. Since R has at most 5 holes and each hole we add $O(1/\epsilon^4)$ in the construction of R', it follows that:

$$|V(R')| = O(\log(n)/\epsilon^c + 1/\epsilon^4) = O(\log(n)/\epsilon^c)\ . \tag{22}$$

There are two different regimes for the choice of $\mathcal{E}_R$:

(a) the space of $\mathcal{E}_R$ is $\log^{1+o(1)}(n)/\epsilon^{c+o(1)}$. Then the total space of Step 4(2) is:

$$O(\frac{n\epsilon^c}{\log(n)}) \cdot \frac{\log^{1+o(1)}(n)}{\epsilon^{c+o(1)}} = n(\log(n)1/\epsilon)^{o(1)}. \tag{23}$$

(b) the space of $\mathcal{E}_R$ is $\frac{\log(n)}{\epsilon^c}(\log^{2+o(1)}(\log(n)) + \log^{2+o(1)}(1/\epsilon))$. Then the total space of Step 4(2) is:

$$\begin{aligned} &O(\frac{n\epsilon^c}{\log(n)}) \cdot \frac{\log(n)}{\epsilon^c}\left(\log^{2+o(1)}(\log(n)) + \log^{2+o(1)}(1/\epsilon)\right) \\ &= n(\log^{2+o(1)}(\log n) + \log^{2+o(1)}(1/\epsilon)) \end{aligned} \tag{24}$$

□

Proving the space bound in Theorem 4. Since $c = 24$, by Lemmas 13 to 15, the total space of the oracle in Steps 1-3 is :

$$O(\frac{n\epsilon^{c-1}}{\log n} + \frac{n\epsilon^{c-13}}{\log n} + \frac{n\epsilon^{c-24}}{\log n}) = O(n) \tag{25}$$

which is dominated by the total space of Step 4. Thus, the space bound of the oracle as claimed in Theorem 4 follows from Lemma 16. □

3.4.4 Preprocessing

By Lemma 9, $\mathcal{T}$ can be constructed in $O(n \log n)$ time. The most time-consuming step is to compute R^+ for each region R associated with a node of $\mathcal{T}$ defined in Step 2. Recall that R^+ is obtained from R by filling in the non-parental holes. Note that $\mathcal{T}$ only has $O(n/\log(n)\epsilon^c) = O(n/\log(n))$ nodes. However, R^+ could have $\Omega(n)$ vertices, resulting in the total size of $\Omega(n^2/\log(n))$. A standard technique [Tho04, KST13, WY16, CS19] to reduce the size of R^+ is to *approximately* fill the holes of R such that $|V(R^+) \setminus V(R)| = \text{poly}(\log(n), 1/\epsilon)$ and for every $u, v \in M(R)$, $d_R(u,v) \le d_{R^+}(u,v) \le d_R(u,v) + \epsilon D$. We will show that the total size of all regions $\{R^+\}_{R \in \mathcal{T}}$ is $O(n\text{poly}(\log(n)))$. (Indeed, we can reduce the total size of all regions to $O(n \log n \text{poly}(1/\epsilon))$ using a more complicated adaptive filling technique of Chan and Skrepetos [CS19].) Next, we describe the filling procedure in detail using *dense portals*, following Weimann and Yuster [WY16].

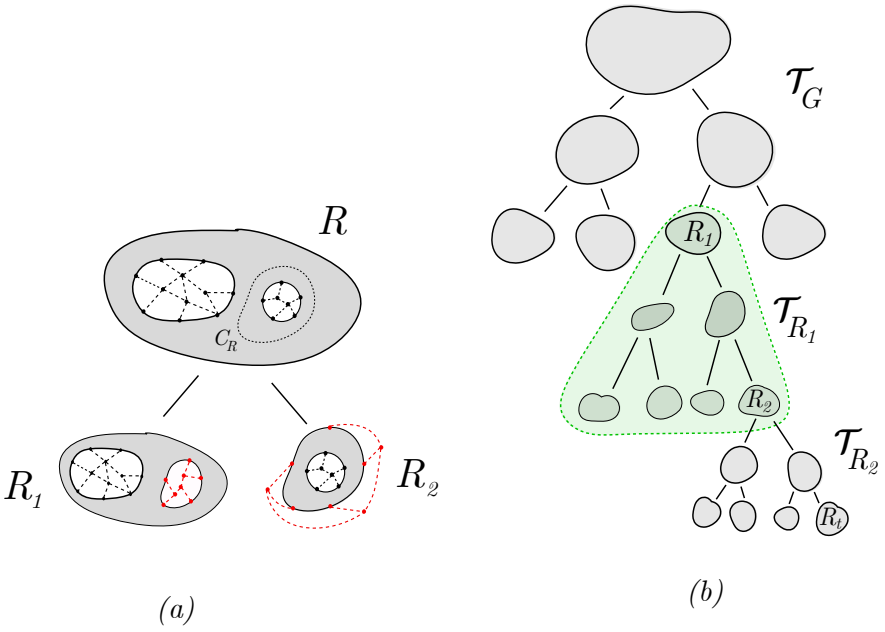


Figure 7: (a) A region R with two holes; two child regions R_1 and R_2; distance preserving minors H_1 (associated with R_1) and H_2 (associated with R_2) compose of red vertices and edges. (b) An illustration for the proof of Theorem 2; the highlighted subtree is the tree resulting from the recursive decomposition of a leaf region R_1 of the tree $\mathcal{T}_G$; the highlighted subtree corresponds to a node at the second level in the hierarchy of oracles $\mathcal{H}$.

The filling process is top-down. The root of $\mathcal{T}$ is associated with G. Let R be a region associated with a node of $\mathcal{T}$, and R_1, R_2 be its two child regions. Let G_R be the approximately filled graph of R where every hole was approximately filled. (Think of G_R as an approximation of G where the distances between marked vertices of R are approximately preserved.) G_R is given by induction; at the root, $G_R = G$.

Let C_R be the shortest path separator that separates R into R_1, R_2. We place $O(\epsilon/\log(n))$ portals, called *dense portals*, on C_R such that the distance between any two nearby dense portals is $c_1\epsilon D/\log(n)$ for some constant $c_1 \geq 1$. Next, we construct two distance preserving minors H_1, H_2 for the dense portals of C_R as in Step 4(2) of the oracle construction, one for the subgraph of G_R inside C_R (and include C_R), and the other for the subgraph of G_R outside C_R (and include C_R). See Figure 7(a) for an illustration. Assume that R_1 is outside C_R and R_2 is inside C_R. Then H_1 and the subgraphs of G_R outside C_R form the approximately filled region of R_1, and similarly, H_2 and the subgraphs of G_R outside C_R form the approximately filled region of R_2. Observe that $|V(H_1)| = O(\log^4(n)/\epsilon^4)$ and $|V(H_2)| = O(\log^4(n)/\epsilon^4)$ by Lemma 1. This means the total size of G_{R_1} is $M(R_1) + O(\log^4(n)/\epsilon^4)$ (as R_1 has only 5 holes). The same holds for R_2. We called the vertices in $(V(G_{R_1}) \cup V(G_{R_2})) \setminus (V(R_1) \cup V(R_2))$ *Steiner vertices*.

Since $|V(\mathcal{T})| = O(n\epsilon^c/\log(n))$ and $c = 24$, the total number of Steiner vertices is $O(n\epsilon^c/\log(n)) \cdot O(\log^4(n)/\epsilon^4) = O(n\log^3(n))$. Thus the total size of all approximately filled regions is $O(n\log^3 n)$. For each graph G_R, computing H_1 and H_2 takes $O(\log^2(n)/\epsilon^2|V(G_R)|)$ time, as it boils down to computing all-pairs shortest paths between dense portals of a hole that can be done $O(|V(G_R)|)$ time per shortest path [HKRS97]. (A more efficient way to compute H_1 and H_2 in $O(n\log(n) + n\log^4(n)\epsilon^{c-4})$ is to use the multiple-source shortest path algorithm of Klein [Kle05]; see Theorem 6 in [CS19].) The total time to compute all approximately filled regions is:

$$O((\sum_R |V(G_R)|)\log^2(n)/\epsilon^2) = O(n\log^3 n\epsilon^c \log^2(n)/\epsilon^2) = O(n\log^5(n)) \ ,$$

since $c = 24$.

For each region R, and its approximately filled graph G_R, we can extract the approximate graph R^+ by removing all the Steiner vertices in the parental hole of R.

Next we compute the patterns of R^+ w.r.t. the sequence of portals σ on the parental boundary of R. This can be done in $O(V(R^+)1/\epsilon)$ time by computing a shortest tree from each vertex in σ; there are only $O(1/\epsilon)$ vertices in the sequence σ. Given the patterns, all tables T_2, T_{41a}, T_{41b} can be computed in $O(\text{poly}(1/\epsilon))$ time per node. Thus, the total running time is $O(n\text{poly}(1/\epsilon))$ since the number of nodes of $\mathcal{T}$ is $O(n\epsilon^c/\log(n)) = O(n)$.

Finally, by Theorem 3, the exact distance oracle for each leaf region $\mathcal{E}_R$ can be computed in time $|V(R')|^{3/2+o(1)} = \text{poly}(\log n, 1/\epsilon)$ since $|V(R')| = O(\log(n)/\epsilon^c)$. Thus, the total running time to compute all the exact distance oracles is $O(n\text{poly}(\log n, 1/\epsilon))$

In summary, the total construction time of the oracle in Theorem 4 is $O(n\text{poly}(\log n, 1/\epsilon))$.

We now show that the additive distortion due to approximate filling is ϵD. Let u, v be any two marked vertices in R_1 (a child region of R). Since the distance between any two nearby dense portals is at most $c_1\epsilon D/\log(n)$, by the triangle inequality, $d_{G_R}(u,v) \leq d_{G_{R_1}}(u,v) \leq d_{G_R}(u,v) + 2c_1\epsilon D/\log(n)$. Since the depth of $\mathcal{T}$ is $O(\log n)$, by induction, it holds that $d_{G_{R_1}}(u,v) \leq d_G(u,v) + O(\log n)2c_1\epsilon D/\log(n) = d_G(u,v) + \epsilon D$ for an appropriate choice of constant c_1.

Since the approximate filling incurs an additive distortion ϵD, the distance returned by the oracle is at most:

$$(d_G(u,v) + \epsilon D) + \epsilon D = d_G(u,v) + 2\epsilon D$$

By scaling, we can get back additive distortion $+\epsilon D$.

3.5 Reducing Space and Query Time: Proof of Theorem 2

We employ the bootstrapping idea of Kawarabayashi, Sommer, and Thorup to replace $\log^{o(1)}(n)$ factor in the space and query time of the oracle in Theorem 4 with a $\log^*(n)$ factor. With more careful analysis and using an LCA data structure to navigate the hierarchy in the construction below, we save a $\log^*(n)$ factor in the query time and make the $\log^*(n)$ factor in the space *additive* instead of multiplicative. We restate Theorem 2 below for convenience.

Theorem 2. *Let $\epsilon > 0, D > 0$ be a positive parameter and $G = (V, E, w)$ be an undirected n-vertex planar graph of diameter D. There is an approximate distance oracle of additive stretch $+\epsilon D$ with construction time $O(n\text{poly}(\log n, \epsilon))$ and:*

(1) $O(n((1/\epsilon)^{o(1)} + \log^*(n)))$ space and $\log^{2+o(1)}(1/\epsilon)$ query time or

(2) $O(n(\log^{2+o(1)}(1/\epsilon) + \log^*(n)))$ space and $(1/\epsilon)^{o(1)}$ query time.

Proof. Starting from G, we apply all steps from 1-4 of the construction in Section 3.4 to G, creating a recursive decomposition tree $\mathcal{T}_G$, except that in Step 4(2), we do not construct an exact distance oracle $\mathcal{E}_{R_1}$ for each leaf region R_1. Instead, we recurse on R_1, creating a second-level recursive decomposition tree $\mathcal{T}_{R_1}$. That is, we apply all steps from 1-4 to R_1, except Step 4(2). See Figure 7(b) for an illustration of the construction. Note that $|M(R_1)| = \Theta(\log(n)/\epsilon^c)$ for $c = 24$. Let $\mathcal{T}_{R_1}$ be the recursion tree induced by the recursive decomposition of R_1; leaves of $\mathcal{T}_{R_1}$ correspond to regions, say R_2, of R_1 that have $|M(R_2)| = \Theta(\log\log(n)/\epsilon^c)$. We then continue to recurse on R_2. Generally, at step j of the recursion, the number of marked vertices in the region associated with each leaf R_j of the recursive decomposition tree, denoted by $\mathcal{T}_{R_{j-1}}$, is $O(\frac{\log^{(j)} n}{\epsilon^c})$ where $\log^{(j)}(n) = \log(\log(\cdots\log(n)\cdots))$; the logarithm is applied j times. We stop the recursion when a leaf region, say R_t, at some step t of the recursion, has $|M(R_t)| = \Theta(1/\epsilon^c)$. We apply Theorem 3 to construct an exact distance oracle $\mathcal{E}_{R_t}$ for the contracted filled graph R'_t of R_t with:

- **Regime 4(2a).** $|V(R'_t)|^{1+o(1)} = |V(R'_t)|(1/\epsilon)^{o(1)}$ space and $O(\log^{2+o(1)}(|V(R'_t)|)) = \log^{2+o(1)}(1/\epsilon)$ query time or
- **Regime 4(2b).** $|V(R'_t)|\log^{2+o(1)}(|V(R'_t)|) = |V(R'_t)|\log^{2+o(1)}(1/\epsilon)$ space and $|V(R'_t)|^{o(1)} = (1/\epsilon)^{o(1)}$ query time.

Note that $|V(R'_t)| = \text{poly}(1/\epsilon)$ since $|M(R_t)| = \Theta(1/\epsilon^c)$.

This construction gives us a *hierarchy* $\mathcal{H}$ of oracles: each recursion step corresponds to a level in $\mathcal{H}$. Each internal node τ of the hierarchy at level j corresponds to the oracle, say $\mathcal{O}_{R_{j-1}}$, for a leaf region R_{j-1} at level $j-1$ (when $j = 1$, we denote $R_0 = G$). The oracle $\mathcal{O}_{R_{j-1}}$ allows us to query distances between marked vertices that are not in the same leaf of the recursive decomposition tree $\mathcal{T}_{R_{j-1}}$ in $O(1)$ time, following the analysis in Section 3.4.2. Each leaf of $\mathcal{H}$ corresponds to an exact distance oracle for a region R_t.

Clearly, the recursion depth, which is also the depth of the hierarchy $\mathcal{H}$, is $t = O(\log^* n)$ since each time we recurse, the size of the region is reduced from $O(\frac{\log^{(j-1)}(n)}{\epsilon^c})$ to $O(\frac{\log^{(j)} n}{\epsilon^c})$ for some $j \in \{2, 3, \ldots, t\}$; in the first level ($j = 1$), the size is reduced from n to $O(\log(n)/\epsilon^c)$.

By the analysis in Section 3.4.3, in particular Equation (25), the total space of each *non-leaf level* of $\mathcal{H}$ is $O(n)$. Thus, the total space of $\mathcal{H}$ associated with non-leaf nodes is $O(n \log^* n)$. On the other hand, by the same analysis in Equations (23) and (24), the total space of the oracles at leaves of $\mathcal{H}$ is:

- $\sum_{R_t \text{ is a leaf of } \mathcal{H}} |V(R'_t)|(1/\epsilon)^{o(1)} = n\epsilon^{o(1)}$ space in Regime 4(2a) or
- $\sum_{R_t \text{ is a leaf of } \mathcal{H}} |V(R'_t)| \log^{2+o(1)}(1/\epsilon) = n \log^{2+o(1)}(1/\epsilon)$ in Regime 4(2b).

Thus, the total space of the oracle is $O(n(\epsilon^{o(1)} + \log^* n))$ in Regime 4(2a) and is $O(n(\log^{2+o(1)}(1/\epsilon) + \log^* n))$ in Regime 4(2b), as claimed.

To answer a query quickly, we augment $\mathcal{H}$ with the following: for each vertex $v \in G$, we store a pointer to a leaf node of $\mathcal{H}$ whose corresponding region R_t contains v as a marked vertex. Furthermore, we construct an LCA data structure for $\mathcal{H}$. Note that $\mathcal{H}$ has $O(n)$ nodes as it is a binary tree with at most n leaves, the total space augmented to $\mathcal{H}$ is $O(n)$.

Now given two vertices u and v, let $R_t(u)$ and $R_t(v)$ be two leaf regions of $\mathcal{H}$ containing u and v. If $R_t(u) \neq R_t(v)$, we query the lowest common ancestor, denoted by R_{uv}, of $R_t(u)$ and $R_t(v)$ in $O(1)$ time. Then, the approximate distance query can be done in $O(1)$ by querying $\mathcal{O}_{R_{uv}}$. If $R_t(u) = R_t(v) = R$, we query the exact distance oracle $\mathcal{E}_R$ to obtain an approximate distance between u and v in time $\log^{2+o(1)}(1/\epsilon)$ in Regime 4(2a) and in time $(1/\epsilon)^{o(1)}$ in Regime 4(2b). Following the stretch analysis in Section 3.4.2, the additive stretch is $+O(\epsilon)D$.

For the construction time, by Theorem 4, each level of $\mathcal{H}$ can be constructed in time $n\text{poly}(\log n, \epsilon)$ time. Since the depth of $\mathcal{H}$ is $O(\log^* n)$, the running time to construct $\mathcal{H}$ is $n\text{poly}(\log n, \epsilon) \cdot \log^* n = n\text{poly}(\log n, \epsilon)$, as desired. □

4 Distance Oracles with Multiplicative Stretch: Proof of Theorem 1

The construction relies on *sparse covers* as defined below. For a graph G, we denote by $\mathsf{diam}(G)$ the diameter of G. For a vertex $v \in V(G)$ and a parameter $r > 0$, we denote by $B_G(v, r) = \{u \in V(G) : d_G(u, v) \leq r\}$ the ball of radius r centered at v.

Definition 7 (Sparse Cover)**.** *A (β, s, Δ)-sparse cover of an edge-weighted graph $G = (V, E, w)$ is a collection of* induced subgraphs $\mathcal{C} = \{C_1, \ldots, C_k\}$*, called* clusters *such that:*

(1) $\mathsf{diam}(C_i) \leq \Delta$ *for every* $i \in [k]$.
(2) For every $v \in V$*, there exists* $i \in [k]$ *such that* $B_G(v, \Delta/\beta) \subseteq V(C_i)$.
(3) Every vertex v *is contained in at most* s *clusters in* $\mathcal{C}$.

If for any given $\Delta > 0$, G has a (β, s, Δ)-sparse cover, we say that G admits a (β, s)-*sparse covering scheme*.

The notion of sparse covers was introduced by Awerbuch and Peleg [AP90]. Busch, LaFortune, and Tirthapura [BLT07] showed that planar graphs admit an $(O(1), O(1))$-sparse covering scheme. Abraham, Gavoille, Malkhi, and Wieder [AGMW10] extended the result of Busch, LaFortune, and Tirthapura [BLT07] to minor-free graphs. Le and Wulff-Nilsen [LWN21] showed that a sparse cover of planar graphs can be constructed in linear time.

Lemma 17 (Lemma 1 in the full version of [LWN21])**.** *Given a planar graph $G = (V, E, w)$ with n vertices and any parameter $\Delta > 0$, then one can construct an $(O(1), O(1), \Delta)$-sparse cover of G in $O(n)$ time.*

Lemma 18. *Let $\epsilon \in (0,1), r > 0$ be positive parameters and $G = (V, E, w)$ be an n-vertex planar graph. We can construct in $O(n\text{poly}(\log(n), 1/\epsilon))$ time an oracle $\mathcal{O}_G$ such that:*

$$\begin{aligned} &d_G(u,v) \le \mathcal{O}_G(u,v) && \text{for all } u,v \in V \\ &\mathcal{O}_G(u,v) \le (1+\epsilon) d_G(u,v) && \text{for } u,v \in V \text{ s.t. } d_G(u,v) \in [r, 2r] \end{aligned} \tag{26}$$

Here $\mathcal{O}_G(u,v)$ is the distance returned by $\mathcal{O}_G$. Furthermore, $\mathcal{O}_G$ has

(1) $O(n((1/\epsilon)^{o(1)} + \log^*(n)))$ space and $\log^{2+o(1)}(1/\epsilon)$ query time or

(2) $O(n(\log^{2+o(1)}(1/\epsilon) + \log^*(n)))$ space and $(1/\epsilon)^{o(1)}$ query time.

Proof. We construct a $(\beta, s, \beta \cdot (2r))$-sparse cover $\mathcal{C}$ for G with $\beta = O(1)$ and $s = O(1)$ using Lemma 17. Since $s = O(1)$, by property (3) of Definition 7, we have:

$$\sum_{C \in \mathcal{C}} |V(C)| \le s|V| = O(n) \tag{27}$$

For each cluster $C \in \mathcal{C}$, we apply Theorem 2 to construct an additive distance oracle $\mathcal{O}_C$ with additive stretch $+\epsilon\mathsf{diam}(C)$. The oracle for G consists of the oracles for all clusters in $\mathcal{C}$. In addition, for each vertex u, we will store $O(1)$ pointers to each cluster $C \in \mathcal{C}$ that contains u. If we let $\mathcal{C}_{uv} \subseteq \mathcal{C}$ be the set of $O(1)$ clusters containing both u and v, then the approximate distance between u and v is:

$$\mathcal{O}_G(u,v) = \min_{C \in \mathcal{C}_{uv}} \mathcal{O}_C(u,v) \tag{28}$$

where $\mathcal{O}_C(u,v)$ is the distance returned by the oracle $\mathcal{O}_C$. (If $\mathcal{C}_{u,v} = \varnothing$, we simply set $\mathcal{O}_G(u,v) = +\infty$.)

Clearly, $\mathcal{O}_G(u,v) \ge d_G(u,v)$ for any $u, v \in V$ since $d_C(u,v) \le d_G(u,v)$ for any subgraph C of G.

Next, we consider the case where $d_G(u,v) \in [r, 2r]$. By property (2) in Definition 7, there is a cluster $X \in \mathcal{C}$ such that $B_G(v, (\beta 2r)/\beta) = B_G(v, 2r) \subseteq V(X)$. Since X is an induced subgraph of G, it holds that $d_X(u,v) = d_G(u,v)$. Furthermore, since the additive stretch of $\mathcal{O}_X$ is :

$$\epsilon\mathsf{diam}(X) \le \epsilon(2\beta r) = O(\epsilon) r = O(\epsilon) d_G(u,v) \ ,$$

it follows that:

$$\mathcal{O}_X(u,v) \le d_X(u,v) + O(\epsilon) d_G(u,v) = (1 + O(\epsilon)) d_G(u,v) \ .$$

This and Equation (28) imply that $\mathcal{O}(u,v) \le (1 + O(\epsilon)) d_G(u,v)$. By scaling ϵ, we get that $\mathcal{O}(u,v) \le (1+\epsilon) d_G(u,v)$, as claimed.

We now analyze the space and query time of $\mathcal{O}$.

If we use Regime (1) in Theorem 2 to construct $\mathcal{O}_C$, then the query time is $\log^{2+o(1)}(1/\epsilon)$ and the total space is:

$$\sum_{C \in \mathcal{C}} O(|V(C)|)((1/\epsilon)^{o(1)} + \log^*(|V(C)|)) = O(n((1/\epsilon)^{o(1)} + \log^*(n)))$$

by Equation (27).

If we use Regime (2) in Theorem 2 to construct $\mathcal{O}_C$, then the query time is $(1/\epsilon)^{o(1)}$ and the total space is:

$$\sum_{C\in\mathcal{C}} O(|V(C)|)(\log^{2+o(1)}(1/\epsilon)+\log^*(|V(C)|)) = O(n(\log^{2+o(1)}(1/\epsilon)+\log^*(n)))$$

by Equation (27).

For the construction time, by Lemma 17, $\mathcal{C}$ can be constructed in $O(n)$ time. By Theorem 2, $\mathcal{O}_C$ can be constructed in $|V(C)|\text{poly}(\log(|V(C)|),\epsilon) = |V(C)|\text{poly}(\log(n),\epsilon)$ time. Thus, by Equation (27), the total running time to construct all oracles $\mathcal{O}_C$ is $O(n\text{poly}(\log(n),\epsilon))$ as claimed. □

We are now ready to prove Theorem 1 that we restate below for convenience.

Theorem 1. *Let $\epsilon\in(0,1)$ be positive parameter and $G=(V,E,w)$ be an undirected, edge-weighted planar graphs with n vertices. We can construct in $O(n\text{poly}(\log(n),1/\epsilon))$ time a $(1+\epsilon)$-approximate distance oracle that has:*

(1) $O(n\log(n)((1/\epsilon)^{o(1)}+\log^* n))$ space and $\log^{2+o(1)}(1/\epsilon)$ query time or

(2) $O(n\log(n)(\log^{2+o(1)}(1/\epsilon)+\log^* n))$ space and $(1/\epsilon)^{o(1)}$ query time.

Proof. By scaling edge weights, we assume that the minimum distance is 1. For each $i=0,1,2,\ldots$, we denote $r_i=2^i$. Let G_i be obtained from G by contracting every edge of weight at most $(r_i\epsilon)/n$. Observe that, for every pair (u,v) such that $d_G(u,v)\in[r_i,2r_i]$, we have:

$$d_G(u,v)-\epsilon r_i \le d_{G_i}(u,v)\le d_G(u,v). \tag{29}$$

This is because we contract at most n edges of weight at most $r_i/(\epsilon n)$ each and, hence, the distance loss due to the contraction is at most $n\cdot(r_i\epsilon)/n\le\epsilon r_i$. Furthermore, by construction, each edge $e\in G$ belongs to at most $O(\log n)$ graphs G_i; it follows that:

$$\sum_{i\ge 0}|E(G_i)| = O(n\log n) \tag{30}$$

For each subgraph G_i, we apply Lemma 18 to construct a distance oracle $\mathcal{O}_{G_i}$.

Next, we construct a 2-approximate distance oracle $\mathcal{O}_2$ with $O(n\log n)$ space and $O(1)$ query time; such an oracle can be constructed in $O(n\log^3(n))$ time by applying the construction of Thorup [Tho04] and Klein [Kle02] with $\epsilon=1$.

Our final oracle, denoted by $\mathcal{O}$, consists of $\mathcal{O}_2$ and all oracles $\{\mathcal{O}_{G_i}\}_{i\ge0}$.

To query $\mathcal{O}$ given two vertices u and v, first we query $\mathcal{O}_2$ to get a 2-approximation of $d_G(u,v)$, denoted by $\mathcal{O}(u,v)$. Then we compute a set of 3 indices $I_{uv}=\{i_0-2,i_0-1,i_0\}$ with $i_0=\lfloor\log_2(\mathcal{O}_2(u,v))\rfloor$. Finally, for each index $j\in I_{uv}$, we query the oracle $\mathcal{O}_{G_j}$ and return:

$$\mathcal{O}(u,v)=\min_{j\in I_{uv}}\{\mathcal{O}_{G_j}(u,v)+\epsilon r_j\} \tag{31}$$

We now bound the stretch of $\mathcal{O}$. Let i_{uv} be such that $d_G(u,v)\in[r_{i_{uv}},2r_{i_{uv}})$. This means that if we query the oracle $\mathcal{O}_{G_{i_{uv}}}$, by Lemma 18, the returned distance $\mathcal{O}_{G_{i_{uv}}}(u,v)$ satisfies:

$$d_{G_{i_{uv}}}(u,v)\le\mathcal{O}_{G_{i_{uv}}}(u,v)\le(1+\epsilon)d_{G_{i_{uv}}}(u,v)$$

Thus, from Equation (29) and the fact that $d_G(u,v) \geq r_{i_{uv}}$, we have:

$$d_G(u,v) \leq d_{G_{i_{uv}}}(u,v) + \epsilon r_{uv} \leq \mathcal{O}_{G_{i_{uv}}}(u,v) + \epsilon r_{uv}$$
$$\mathcal{O}_{G_{i_{uv}}}(u,v) + \epsilon r_{uv} \leq (1+\epsilon) d_{G_{i_{uv}}}(u,v) + \epsilon r_{uv} \leq (1+2\epsilon) d_G(u,v),$$

implying that the (multiplicative) stretch of $\mathcal{O}$ is $(1+2\epsilon)$; by scaling ϵ, we get back stretch $(1+\epsilon)$.

We now bound the space and query time of $\mathcal{O}$; we consider two regimes in Lemma 18 that we use to construct $\mathcal{O}_i$.

1. **Regime (1) .** Since the query time of $\mathcal{O}_2$ is $O(1)$, I_{uv} can be computed in $O(1)$ time. Since $|I_{uv}| = 3$ and the query time of each $\mathcal{O}_{G_i}$ is $\log^{2+o(1)}(1/\epsilon)$, the total query time is $\log^{2+o(1)}(1/\epsilon)$. The space of $\mathcal{O}_2 = O(n \log n)$ and the total space of all $\{\mathcal{O}_{G_i}\}_{i \geq 0}$, by Lemma 18, is:

$$\sum_{i \geq 0} O(|V(G_i)|((1/\epsilon)^{o(1)} + \log^*(|V(G_i)|))) = O(n \log(n)((1/\epsilon)^{o(1)} + \log^* n))$$

   by Equation (30). This implies the claimed space bound.

2. **Regime (2) .** In this regime, the query time of each $\mathcal{O}_{G_i}$ is $(1/\epsilon)^{o(1)}$ which is also the total query time. The total space of all $\{\mathcal{O}_{G_i}\}_{i \geq 0}$, by Lemma 18 is:

$$\sum_{i \geq 0} O(|V(G_i)|(\log^{2+o(1)}(1/\epsilon) + \log^*(|V(G_i)|))) = O(n \log(n)(\log^{2+o(1)}(1/\epsilon) + \log^* n))$$

   by Equation (30), as desired.

For the construction time, recall that the construction time of $\mathcal{O}_2$ is $O(n \log^3(n))$. The construction time of each $\mathcal{O}_{G_i}$ is $|V(G_i)|\text{poly}(\log(|V(G_i)|), 1/\epsilon) = |V(G_i)|\text{poly}(\log(n), 1/\epsilon)$. By Equation (30), the total construction time of $\mathcal{O}$ is $n\text{poly}(\log(n), 1/\epsilon)$. □

Acknowledgement. This work is supported by the National Science Foundation under Grant No. CCF-2121952. We thank Christian Wulff-Nilsen for many helpful conversations.

References

[ACC+96] 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 *European Symposium on Algorithms, ESA'96*, pages 514–528, 1996, doi:10.1007/3-540-61680-2_79. 6

[ADD+93] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Computational Geometry*, 9(1):81–100, 1993, doi:10.1007/BF02189308. 2

[AGMW10] I. Abraham, C. Gavoille, D. Malkhi, and U. Wieder. Strong-diameter decompositions of minor free graphs. *Theory of Computing Systems*, 47(4):837–855, 2010, doi:10.1007/s00224-010-9283-6. 27

[AP90] B. Awerbuch and D. Peleg. Sparse partitions. In *Proceedings the 31st Annual Symposium on Foundations of Computer Science*, FOCS '90, 1990, doi:10.1109/fscs.1990.89571. 27

[BFC00] M. A. Bender and M. Farach-Colton. The lca problem revisited. In *Latin American Symposium on Theoretical Informatics (LATIN '00)*, pages 88–94, 2000, doi:10.1007/10719839_9. 22

[BLT07] C. Busch, R. LaFortune, and S. Tirthapura. Improved sparse covers for graphs excluding a fixed minor. In *Proceedings of the 26th annual ACM symposium on Principles of Distributed Computing*, PODC '07, 2007, doi:10.1145/1281100.1281112. 27

[Cab10] S. Cabello. Many distances in planar graphs. *Algorithmica*, 62(1-2):361–381, 2010. Announced at SODA '06, doi:10.1007/s00453-010-9459-0. 6

[Cab18] S. Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Transactions on Algorithms*, 15(2), 2018. Announced at SODA'17, doi:10.1145/3218821. 1, 4, 6

[CADWN17] V. Cohen-Addad, S. Dahlgaard, and C. Wulff-Nilsen. Fast and compact exact distance oracle for planar graphs. In *IEEE 58th Annual Symposium on Foundations of Computer Science*, FOCS '17, pages 962–973, 2017, doi:10.1109/FOCS.2017.93. 6

[CFKL20] V. Cohen-Addad, A. Filtser, P. N. Klein, and H. Le. On light spanners, low-treewidth embeddings and efficient traversing in minor-free graphs. In *61th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '21, pages 589–600, 2020. See: conference version, arXiv version,. 2

[CGMW19] 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 '19, pages 138–151, 2019, doi:10.1145/3313276.3316316. 5, 6

[CKT22] H. Chang, R. Krauthgamer, and Z. Tan. Almost-linear ϵ-emulators for planar graphs. In *The 54th Annual ACM Symposium on Theory of Computing*, STOC '22, pages 1311–1324, 2022. 2

[CS19] T. M. Chan and D. Skrepetos. Faster approximate diameter and distance oracles in planar graphs. *Algorithmica*, 81(8):3075–3098, 2019. Announced at ESA '17, doi:10.1007/s00453-019-00570-z. 1, 2, 3, 4, 5, 24, 25

[CX00] D. Z. Chen and J. Xu. Shortest path queries in planar graphs. In *Proceedings of the 32nd annual ACM symposium on Theory of computing*, STOC '00, pages 469—-478, 2000, doi:10.1145/335305.335359. 6

[Dji96] H. N. Djidjev. Efficient algorithms for shortest path queries in planar digraphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science, WG'96*, pages 151–165, 1996, doi:10.1007/3-540-62559-3_14. 6

[Fed87] G. N. Federickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987, doi:10.1137/0216064. 3

[FGNW17] O. Freedman, P. Gawrychowski, P. K. Nicholson, and O. Weimann. Optimal distance labeling schemes for trees. In *Proceedings of the 36th ACM Symposium on Principles of Distributed Computing*, PODC '17, 2017, doi:10.1145/3087801.3087804. 2

[FHMWN21] V. Fredslund-Hansen, S. Mozes, and C. Wulff-Nilsen. Truly Subquadratic Exact Distance Oracles with Constant Query Time for Planar Graphs. In *32nd International Symposium on Algorithms and Computation*, ISAAC '21, pages 25:1–25:12, 2021. https://arxiv.org/abs/2009.14716, doi:10.4230/LIPIcs.ISAAC.2021.25. 4, 5, 6, 9, 11, 14

[FKS19] E. Fox-Epstein, P. N. Klein, and A. Schild. Embedding planar graphs into low-treewidth graphs with applications to efficient approximation schemes for metric problems. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, page 1069–1088, 2019, doi:10.1137/1.9781611975482.66. 2

[FR01] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, 2001, doi:10.1109/sfcs.2001.959897. 6

[GKK+01] C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg. Approximate distance labeling schemes. In *Proceedings of the 9th Annual European Symposium on Algorithms*, pages 476–487. 2001, doi:10.1007/3-540-44676-1_40. 2

[GMWWN18] P. Gawrychowski, S. Mozes, O. Weimann, and C. Wulff-Nilsen. Better tradeoffs for exact distance oracles in planar graphs. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 515–529, 2018, doi:10.1137/1.9781611975031.34. 6

[GPPR04] C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004, doi:10.1016/j.jalgor.2004.05.002. 3

[GX19] Q. Gu and G. Xu. Constant query time $(1+\epsilon)$-approximate distance oracle for planar graphs. *Theoretical Computer Science*, 761:78–88, 2019. Annouced at ISAAC '15, doi:10.1016/j.tcs.2018.08.024. 1, 2, 4

[HKRS97] M. R. Henzinger, P. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997. 25

[KKS11] K. Kawarabayashi, P. N. Klein, and C. Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *The 38th International Colloquium on Automata, Languages and Programming*, ICALP '11, pages 135–146, 2011, doi:10.1007/978-3-642-22006-7_12. 1, 2, 5

[Kle02] P. Klein. Peprocessing an undirected planar network to enable fast approximate distance queries. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 820—-827, 2002, doi:10.5555/545381.545488. 1, 2, 3, 29

[Kle05] P. N. Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, page 146–155, 2005. 25

[KNZ14] R. Krauthgamer, H. L. Nguyen, and T. Zondiner. Preserving terminal distances using minors. *SIAM J. Discrete Math.*, 28(1):127–141, 2014, doi:10.1137/120888843. 7

[KST13] K. Kawarabayashi, C. Sommer, and M. Thorup. More compact oracles for approximate distances in undirected planar graphs. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, 2013, doi:10.1137/1.9781611973105.40. 1, 2, 3, 5, 24

[LF22] H. Le and A. Filtser. Low treewidth embeddings of planar and minor-free metrics. In *to appear in 63rd Annual IEEE Symposium on Foundations of Computer Science*, FOCS '22, 2022. https://arxiv.org/pdf/2203.15627.pdf. 2

[LP19] 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, page 152–163, 2019, doi:10.1145/3313276.3316358. 2, 4, 6, 8, 9

[LP21] Y. Long and S. Pettie. Planar distance oracles with better time-space tradeoffs. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA'21*, pages 2517–2537, 2021. 5, 6, 7

[LT79] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. 3, 6

[LT80] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980, doi:10.1137/0209046. 3, 6

[LWN21] H. Le and C. Wulff-Nilsen. Optimal approximate distance oracle for planar graphs. In *Proceedings the 62nd Annual Symposium on Foundations of Computer Science*, FOCS '21, pages 363–374, 2021. https://arxiv.org/abs/2111.03560, doi:10.1109/focs52979.2021.00044. 1, 27, 28

[MS12] S. Mozes and C. Sommer. Exact distance oracles for planar graphs. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'12, pages 209–222, 2012, doi:10.1137/1.9781611973099.19. 6

[Sau72] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972, doi:10.1016/0097-3165(72)90019-2. 7

[She72] S. Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1):247 – 261, 1972, doi:pjm/1102968432. 7

[Tho04] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004. Announced at FOCS' 01, doi:10.1145/1039488.1039493. 1, 2, 3, 24, 29

[TZ05] M. Thorup and U. Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, 2005, doi:10.1145/1044731.1044732. 1

[WN10] C. Wulff-Nilsen. *Algorithms for planar graphs and graphs in metric spaces*. PhD thesis, University of Copenhagen, 2010. 6

[WN16] C. Wulff-Nilsen. Approximate distance oracles for planar graphs with improved query time-space tradeoff. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '16, page 351–362, 2016, doi:10.1137/1.9781611974331.ch26. 1

[WY16] O. Weimann and R. Yuster. Approximating the diameter of planar graphs in near linear time. *ACM Transactions on Algorithms*, 12(1):1–13, 2016, doi:10.1145/2764910. 3, 4, 24