Optimal Approximate Distance Oracle for Planar Graphs

Hung Le
University of Massachusetts at Amherst
Amherst, USA
hungle@cs.umass.edu

Christian Wulff-Nilsen
University of Copenhagen (DIKU)
Copenhagen, Denmark
koolooz@di.ku.dk

***Abstract*—A $(1+\epsilon)$-approximate distance oracle of an edge-weighted graph is a data structure that returns an approximate shortest path distance between any two query vertices up to a $(1+\epsilon)$ factor. Thorup (FOCS 2001, JACM 2004) and Klein (SODA 2002) independently constructed a $(1+\epsilon)$-approximate distance oracle with $O(n \log n)$ space, measured in number of words, and $O(1)$ query time when G is an undirected planar graph with n vertices and ϵ is a fixed constant. Many follow-up works gave $(1+\epsilon)$-approximate distance oracles with various trade-offs between space and query time. However, improving $O(n \log n)$ space bound without sacrificing query time remains an open problem for almost two decades. In this work, we resolve this problem affirmatively by constructing a $(1+\epsilon)$-approximate distance oracle with optimal $O(n)$ space and $O(1)$ query time for undirected planar graphs and fixed ϵ.**

We also make substantial progress for planar digraphs with non-negative edge weights. For fixed $\epsilon > 0$, we give a $(1+\epsilon)$-approximate distance oracle with space $o(n \log(Nn))$ and $O(\log \log(Nn)$ query time; here N is the ratio between the largest and smallest positive edge weight. This improves Thorup's (FOCS 2001, JACM 2004) $O(n \log(Nn) \log n)$ space bound by more than a logarithmic factor while matching the query time of his structure. This is the first improvement for planar digraphs in two decades, both in the weighted and unweighted setting.

***Keywords*-Distance Oracles, Planar Graphs, Data Structures**

I. Introduction

Computing shortest path distances in edge-weighted planar graphs[1] is a fundamental problem with lots of practical applications, such as planning, logistics, and traffic simulation [51]. The celebrated algorithm of Henzinger et al. [33] can compute the shortest path distance in a planar graph between any given pair of vertices in linear time. However, when we are given a large number of distance queries and the network is huge, a linear time algorithm for answering *each* distance query becomes unsatisfactory. This motivates the development of *(exact or approximate) distance oracles*: a data structure that can quickly answer any (exact or approximate) distance query.

Trees are simplest planar graphs that admit exact distance oracles with linear space and constant query time; the construction is by a simple reduction to constructing a lowest common ancestor (LCA) data structure. This basic fact leads to the following fundamental question that has been driving the field:

Question 1. *Is it possible to construct a distance oracle, exact or approximate, for edge-weighted planar graphs, directed or undirected, with guarantees matching those for trees: linear space and constant query time?*

Despite significant research efforts spanning more than two decades [38], [52], [47], [54], [36], [28], [16], [37], [1], [46], [27], [19], [17], [43], Question 1 remains largely open in all four basic settings: exact/approximate oracles for directed/undirected planar graphs.

Cohen-Addad, Dahlgaard and Wulff-Nilsen [19] constructed the first exact distance oracle with truly subquadratic space and $O(\log(n))^2$ query time for *directed* planar graphs; previous exact oracles with truly subquadratic space had polynomial query time. Several follow-up works [27], [17], [43] improved the oracle of Cohen-Addad, Dahlgaard and Wulff-Nilsen [19] in several ways, getting either space $\tilde{O}(n)$ and $n^{o(1)}$ query time or space $n^{1+o(1)}$ and $\tilde{O}(1)$ query time [43]. ($\tilde{O}$ notation hides a polylogarithmic factor of n.) However, no exact distance oracle with *constant query time* and *truly subquadratic* space is known, even for undirected weighted or unweighted directed planar graphs. For a special case of *unweighted*, undirected planar graphs, Fredslund-Hansen, Mozes and Wulff-Nilsen [25] recently constructed an exact distance oracle with $O(n^{\frac{5}{3}+\epsilon})$ space and $O(\log \frac{1}{\epsilon})$ query time for any choice of parameter $\epsilon > 0$. If we only want to query exact distances of value at most a *constant* k in an unweighted, directed planar graph, Kowalik and Kurowski [41] showed that an oracle with linear space and constant query time exists. For a more thorough review of exact distance oracles in planar graphs, see Section I-B.

Given the remote prospect of answering Question 1 for exact distance oracles, we focus on $(1+\epsilon)$-*approximate* distance oracles where the query output is never smaller than the true shortest path distance and not larger by more than a factor of $(1+\epsilon)$, for any given $\epsilon > 0$. Note that Question 1 is only relevant when ϵ is a fixed constant, and this is the most basic regime that we are interested in.

Over the past 20 years, significant progress has been made

[1]Graphs considered in this paper are edge-weighted unless specified otherwise.

[2]In this paper, n is the number of vertices of the graph.

in constructing $(1+\epsilon)$-approximate distance oracle for *undirected* planar graphs. In their seminal papers, Thorup [52] and Klein [38] indpendently constructed distance oracles with $O(n(\log n)\epsilon^{-1})$ space[3] and $O(\epsilon^{-1})$ query time. When $\epsilon = \Theta(1)$, their oracles have $O(n \log n)$ space and $O(1)$ query time. (Henceforth, we do not spell out the dependency on ϵ unless it is important to do so.) Kawarabayashi, Klein and Sommer [36] reduced the space to $O(n)$ at the cost of increasing the query time to $O(\log^2 n)$. Kawarabayashi, Sommer and Thorup [37] designed a distance oracle with $\overline{O}(n)$ space and $\overline{O}(1)$ query time when *every edge has weight at most* $\log^{O(1)}(n)$ (and at least 1); $\overline{O}(.)$ notation hides a $\log\log n$ factor. Wullf-Nilsen [54] was the first to break the $\Omega(n \log n)$ bound on the trade-off between space and query time by giving an oracle with $O(n(\log\log n)^2)$ space and $O((\log\log n)^3)$ query time[4]. While the oracle of Wulff-Nilsen implies that the $O(n \log n)$ space-time tradeoff is not the best possible, it is suboptimal in both space and query time. In this paper, for the first time, we provide an affirmative answer to 1 in one of the four basic settings: approximate distance oracles for undirected planar graphs.

Theorem 1. *Given an edge-weighted n-vertex planar undirected graph G and a fixed parameter $\epsilon < 1$, there is a $(1+\epsilon)$-approximate distance oracle with $O(n)$ space and $O(1)$ query time. Furthermore, the oracle can be constructed in worst-case time $O(n\text{polylog}(n))$ time.*

The precise dependencies of the space, query time, and construction time on ϵ are $O(n\epsilon^{-2})$, $O(\epsilon^{-2})$ and $O(n\epsilon^{-3}\log^6(n))$, respectively. For the simplicity of the presentation, we do not try to optimize the dependency on ϵ as well as the logarithmic factor in the construction time.

Our result is optimal in the sense that any $(1+\epsilon)$-approximate distance oracle for n-vertex weighted undirected planar graphs must use $\Omega(n)$ space. This lower bound holds regardless of query time and holds even for n-vertex simple paths with unique integer edge weights in $\{2^0, 2^1, \ldots, 2^{n-1}\}$ and with $\epsilon < 1$: the number of such paths is $n!$ and since $1+\epsilon < 2$, a $(1+\epsilon)$-approximate distance oracle for any such path P can be queried to derive the weight of each edge of P. Hence, any $(1+\epsilon)$-approximate oracle needs $\Omega(n \lg n)$ bits of space. Although an integer edge weight like 2^{n-1} cannot be stored in a $\Theta(\lg n)$ bit word using the standard binary representation of an integer, it can be stored in 1 (or possibly $O(1)$) words when represented as a floating point since the exponent can be represented using $O(\lg n)$ bits. Also note that for unweighted undirected planar graphs, a $(1+\epsilon)$-approximate oracle with $o(n)$ space would not allow the user to freely assign unique labels to the vertices since there are $n!$ such assignments.

[3]Unless otherwise stated, each space bound in this paper is in the number of *words* that the oracle uses.

[4]The dependency on ϵ of the oracles in [36], [54], [37] is polynomial in $\frac{1}{\epsilon}$.

For planar digraphs[5], Thorup [52] designed the first $(1+\epsilon)$-approximate distance oracle with $O(n \log n \log(nN))$ space and $O(\log\log(nN))$ query time where edge weights are non-negative and N is the ratio between the largest and smallest positive edge weight.[6] (The precise dependency on ϵ of Thorup's oracle is $O(n \log n \log(nN)/\epsilon)$ for space and $O(\log\log(nN)+1/\epsilon)$ for query time.) Since its introduction two decades ago, Thorup's oracle has not been improved. An added challenge in the directed setting is that *portals* – an important concept in approximate distance oracles – in digraphs are less well-behaved than their undirected counterparts. Specifically, for a given vertex v and a shortest path P, the number of vertices on P, called portals, through which we need to re-route the shortest paths from v to all vertices on P (with $1+\epsilon$ multiplicative error) is $O(\frac{1}{\epsilon})$ in the undirected case, while the number of such portals could be up to $\Theta(|P|)$ in the directed case. Even for the special case of unweighted digraphs, the bounds of Thorup's oracle (by taking $N = 1$) has remained state-of-the-art: $O(n \log^2 n)$ space and $O(\log\log n)$ query time for constant ϵ.

In this paper, we give the first improvement over the space bound Thorup's oracle by more than a logarithmic factor, while keeping the same query time for a constant ϵ. For $x > 0$, define $\log^{(1)} x = \log x$ and for integer $k > 1$, $\log^{(k)} x = \log(\log^{(k-1)} x)$. We show the following.

Theorem 2. *Given a planar n-vertex digraph G with non-negative edge weights and given $\epsilon > 0$ and any integer $k = \Theta(1)$, there is a $(1+\epsilon)$-approximate distance oracle for G with space $O(n \log(Nn) \log^{(k)}(n)/(\epsilon \log\log\log(Nn)))$ and query time $O(\log\log(Nn) + 1/\epsilon^{5.01})$ where N is the ratio between the largest and smallest positive edge weight.*

Note that for $k \geq 4$, the space bound is $o(n \log(Nn)/\epsilon)$ in Theorem 2.

By setting $N = 1$, we obtain the first approximate distance oracle for unweighted planar digraphs with $o(n \log^2 n)$ space and $O(\log\log n)$ query time (Corollary 1 below). In fact, we get a space bound of only $o(n \log n)$. Our result might suggest that it is possible to construct an oracle with $O(n)$ space and $O(1)$ query time for unweighted planar digraphs. We believe that constructing such an oracle is an important step towards resolving Question 1 for approximate distance oracles in edge-weighted planar digraphs.

Corollary 1. *Given an unweighted planar n-vertex digraph G and given $\epsilon > 0$ and any integer $k = \Theta(1)$, there is a $(1+\epsilon)$-approximate distance oracle for G with $O(n \log(n) \log^{(k)}(n)/(\epsilon \log\log\log n))$ space and $O(\log\log n + 1/\epsilon^{5.01})$ query time.*

[5]Digraphs is a shorthand for directed graphs.

[6]Thorup in fact assumes that edge weights are integers in $\{0, 1, \ldots, N\}$ but this is only needed to get a small preprocessing time since it allows the use of fast priority queues. Since we do not focus on preprocessing for planar digraphs, we avoid the integer weight assumption.

For unweighted planar digraphs, there is a space lower bound of $\Omega(n \lg n)$ bits, i.e., $\Omega(n)$ words with word size $\Theta(\lg n)$, and this lower bound holds even for a data structure that answers reachability queries regardless of query time [34]. Thus the space bound in Corollary 1 is only a factor of $o(\log(n)/\epsilon)$ away from optimal.

We have not focused on preprocessing time of our oracle for planar digraphs but it is easy to see from the description of this oracle that preprocessing time is bounded by a polynomial in n times $\log N$.

Model of computation: The model of computation considered in our paper is the standard WORD RAM with word size $\overline{\omega} = \Omega(\log n)$. In this model, arithmetic operations $(+, -, *, /, \%)$, comparisons $(<, >, =, \leq, \geq)$, and bitwise operations (AND, OR, XOR, SHIFT) on words take constant time each.

A. Techniques

Approximate distance oracles for undirected planar graphs: Our technique for undirected planar graphs is inspired by that of Kawarabayashi, Sommer, and Thorup [37] who constructed a distance oracle with $\overline{O}(n \log n)$ space and $\overline{O}(1/\epsilon)$ query time. Their idea is to construct an oracle with *multiplicative* stretch $(1+\epsilon)$ from (a collection of) distance oracles with *additive stretch* via a clever use of *sparse covers*[7]. Specifically, sparse covers are used to construct a set of subgraphs of G, called *clusters*, and then a distance oracle with additive stretch ϵD is constructed for each cluster; here D is the diameter of the cluster. However, the space bound of the oracle by Kawarabayashi, Sommer, and Thorup is superlinear for two reasons: (1) the additive distance oracle has space that is superlinear in the number of vertices of each cluster and (2) the total size (the number of vertices) of all clusters is $\Omega(n \log n)$. Nevertheless, the technique of Kawarabayashi, Sommer, and Thorup [37] suggests an interesting connection to geometry since sparse covers have a very natural geometric interpretation[8].

In doubling metrics, it was shown how to construct a distance oracle with $O(n)$ space and $O(1)$ query time [30], [31], [8] for constant values of ϵ and constant dimensions. Thus, it is natural to ask: Can we exploit techniques developed for doubling metrics to construct a $(1+\epsilon)$-approximate distance oracle for planar graphs with $O(n)$ space and $O(1)$ query time? To be able to answer this question positively, there are several technical barriers that we need to overcome. The most fundamental one is that space bounds of all known oracles for doubling spaces have an *exponential dependency* on the dimension [9], while very simple planar graphs, such as star graphs, have doubling dimension $\Omega(\log n)$. Thus, it is somewhat counter-intuitive that we are able to exploit the geometric techniques in the construction of our apporixmate oracle for planar graphs.

We overcome all the technical barriers, as detailed below, to resolve two obstacles in the construction of Kawarabayashi, Sommer, and Thorup [37] by capitalizing on the techniques developed in the geometric context. Like the previous distance oracle constructions in doubling metrics [31], [30], [8], we start with a *net tree* in which each level i of the net tree is a 2^i-net[10] of the shortest path metric of $G(V, E)$. (There are $O(\log \Delta)$ levels where Δ is the spread[11] of the metric.) In doubling metrics, for each level-i of the net tree, we could store all distances from a net point p to other net points in the same level within radius $O(\frac{2^i}{\epsilon})$ from p; there are only $O(\epsilon^{-d}) = O(1)$ such points when the dimension is a constant. However, in planar graphs, for each net point[12] p, there could be $\Omega(n)$ net points at the same level i within radius $O(\frac{2^i}{\epsilon})$ from p. We cannot afford to store all such distances. An important observation that we rely on in our construction is that we only need to (approximately) preserve distances from p to net points at level i within in radius $\Theta(\frac{2^i}{\epsilon})$ (rather than $O(\frac{2^i}{\epsilon})$) from p. Note that there could still be $\Omega(n)$ such net points.

Our first contribution is a technique to construct a (d, α, S)-*restricted* distance oracle with $O(|S|)$ space and $O(1)$ query time for every parameter d and a *constant* α. The oracle guarantees that, for every pair $(u, v) \in S \times S$ such that $d_G(u, v) \in [d, \alpha d]$, the returned distance is within $[d_G(u, v), (1 + \epsilon) d_G(u, v)]$. An important property of a (d, α, S)-restricted distance oracle, beside having $O(|S|)$ space, is that its space does not depend on n, the number of vertices of the graph. Essentially, we overcome the first obstacle in the construction of Kawarabayashi, Sommer, and Thorup [37]. We use (d, α, S)-restricted oracles in the following way: for each level i of the net tree, we construct a (d, α, S)-restricted distance oracle with $d = \frac{2^i}{\epsilon}$, $\alpha = O(1)$ and $S = N_i$ where N_i is the set of net points at level i. This oracle will guarantee multiplicative stretch $(1 + \epsilon)$ for every pair of net points at level i whose distance from each other is $\Theta(\frac{2^i}{\epsilon})$ – for other pairs, the distance error could be arbitrarily large. The idea behind this construction is that the space incurred *per net point on average* is $O(1)$. We remark that the construction of restricted oracles heavily relies on planarity.

However, the net tree has its own problem: the number of vertices of T could be $\Omega(n \log \Delta)$ where Δ could be exponential in n. A simple idea to reduce the number of vertices of T is to *compress* degree-2 vertices. But the compression of degree-2 vertices introduces two new problems.

[7] See Section II for a formal definition of a sparse cover.

[8] A sparse cover of a Euclidean space is simply a tiling of the space by overlapping hypercubes.

[9] Interestingly, the query time can be made independent of ϵ and d [8]

[10] An r-net of a metric (V, d_G) is a subset of points N such that $d_G(x, y) > r$ for every $x \neq y \in N$ and for every $z \notin N$, there exists $x \in N$ such that $d_G(x, z) \leq r$.

[11] Spread of a metric is the ratio of the maximum pairwise distance to the minimum pairwise distance.

[12] We use points and vertices interchangeably.

First, each point still "participates" in the construction of the restricted distance oracles of up to $O(\log \Delta)$ different levels, and hence, the compression of degree-2 vertices does not really help. Second, compressing the net-tree makes it harder to *navigate*. That is, given a leaf point p, we want to find an ancestor of p at a given level i in $O(1)$ time. In doubling metrics, to efficiently navigate the net-tree, previous constructions heavily rely on the fact that each vertex of the net tree has $O(1)$ children, which is not the case in our setting. Resolving both problems can be seen as overcoming the second obstacle in the construction of Kawarabayashi, Sommer, and Thorup [37].

We resolve the first problem by distinguishing two types of degree-2 vertces in T: those that are required in the construction of restricted oracles and those that are not. For the later type, it is safe to compress. For the former type, we are able to show that there are only $O(n \log(\frac{1}{\epsilon}))$ such degree-2 vertices; this linear bound is crucial to our construction. We resolve the second problem – navigating the net tree – by designing a new *weighted level ancestor* (WLA) data structure. The WLA problem, introduced by Farach and Muthukrishnan [22], is a generalization of the predecessor problem [40] where various *super-constant* lower bounds in query time when the space usage is restricted to $O(n)$ have been established [3], [44], [45], [9], [50], [49]. Here we need a data structure with linear space and constant query time. Our key insight is that for trees with *polylogarithmic depth*, we can design such a data structure[13]. Observe that when $\log(\Delta) = \text{polylog}(n)$, the net tree has a polylogarithmic depth. Hence, for planar graphs with *quasi-polynomial spread*, we can design a distance oracle with $O(n)$ space and $O(1)$ query time with all the ideas we have discussed. This turns out to be the hardest case: we adapt the contraction trick of Kawarabayashi, Sommer, and Thorup [37] and devise the bit packing technique to reduce the general problem to the case where the spread is quasi-polynomial.

The final tool we need is a data structure that allows us to quickly determine the level of the ancestors of u and v in the net tree for each query pair (u, v). Once the ancestors and their level are found, we can perform a distance query from the restricted oracle at that level. We observe that the level of the ancestors can be approximated within a constant additive error if we can determine the distance between u and v within *any constant factor*. To that end, we design an oracle with $O(1)$ multiplicative stretch, linear space, and constant query time. Our technique is simple and based on r-division, a basic tool to design algorithms for planar graphs, and our construction can be implemented in nearly linear time. Furthermore, our results apply to any graph in a hereditary class with sublinear separators. A similar distance oracle for minor-free graphs can be obtained from the tree cover with $O(1)$ trees and $O(1)$ distortion by Bartal, Fandina, and Neiman [7]. However, it is unclear that the tree cover can be constructed in nearly linear time.

To obtain a nearly linear time preprocessing time, the major obstacle in our construction is to compute the net tree. Indeed, to the best of our knowledge, it is not known how to construct an r-net of the shortest path metric of planar graphs in *sub-quadratic time*[14]. Instead, we show that a weaker version of r-nets (see Section II for a precise definition) can be computed in $O(n)$ time, and that we can use weak r-nets in place of r-nets in the net tree. A corollary of our weak net construction is a linear time algorithm to find a sparse cover, which improves upon the $O(n \log n)$ time algorithm of Busch, LaFortune, and Tirthapura [11].

Approximate distance oracles for planar digraphs: The techniques for our $(1+\epsilon)$-approximate distance oracle for planar digraphs build to some extent on those of Thorup [52]. Recall that N is the ratio between the largest and smallest edge weight. After normalizing, the shortest path distances can thus be partitioned into $O(\log(Nn))$ distance scales of the form $[\alpha, 2\alpha)$ where α is a power of 2. For each distance scale, Thorup uses an oracle with an additive error of at most $\epsilon\alpha$. A query is answered by applying binary search on the distance scales, querying one of the oracles in each step, and the final oracle queried then gives the desired multiplicative $(1+\epsilon)$-approximation. Thus, $O(\log\log(Nn))$ queries to oracles are needed. Thorup shows how each oracle can answer a query in constant time (for fixed ϵ).

Our first idea for improving space is to not have every oracle answer a query in constant time. In fact, it suffices for every $\Theta(\log\log(Nn))$ distance scale to have an oracle with $O(1)$ query time since binary search on these oracles brings us down to only $\Theta(\log\log(Nn))$ distance scales; since the total query time should be $O(\log\log(Nn))$, we can thus afford slower but more space-efficient oracles for the remaining $O(\log\log\log(Nn))$ steps of the binary search. Hence, only a $1/\Theta(\log\log(Nn))$ fraction of our oracles have $O(1)$ query time.

To improve space further, we make use of a recursive decomposition of the input digraph G into more and more refined r-divisions. Now, instead of storing portals for each vertex of each oracle (as in Thorup's paper), we instead store portals only for boundary vertices of pieces of r-divisions at each level of the recursive decomposition. Furthermore, each such boundary vertex stores only local portals belonging to the same parent piece, allowing us to use labels of length much smaller than $\lg n$ for each such local portal.

A query for a vertex pair (u, v) is then answered by starting at the lowest level of the recursive decomposition

[13] Alstrup and Holm[5] mentioned the construction of a data structure than can handle WLA in trees of polylogarithmic weights, which could be applicable in our work. However, the details were not given. On the other hand, our data structure instead exploits the polylogarithmic depth.

[14] In doubling metrics, it is possible to construct a net tree in nearly linear time [30], [31]. The construction relies heavily on the fact that the metric has a constant doubling dimension.

and obtaining local portals for u and v. For each combination of a local portal $p(u)$ of u and local portal $p(v)$ of v, a query for an approximate distance from $p(u)$ to $p(v)$ is then answered recursively by going one level up in the recursive decomposition. To avoid an exponential explosion in the number of local portals during the recursion, the recursive decomposition is set to have only $k = \Theta(1)$ height where k is the parameter in Theorem 2.

Since our final space bound is $o(n \log(Nn))$, we are only allowed space sublinear in n on each distance scale. This creates some additional obstacles not addressed by the above techniques. A main obstacle is to answer $\mathcal{LCA}$-queries in $O(1)$ time using $o(n)$ space. There are static tree data structures that can do this using only $O(n/\lg n)$ space (i.e., $O(n)$ bits of space). Unfortunately, these structures require labels of query vertices to be of a special form; for instance, the data structure in [35] that we rely on requires them to be preorder numbers in the tree. We give a new data structure that can convert in $O(1)$ time vertex labels in the input graph G to preorder numbers in such a tree using $o(n)$ space plus $O(n)$ additional space independent of the current distance scale. Here, we again make use of our recursive decomposition and show how it allows for a compact representation of preorder numbers for each tree.

In this extended abstract, we sketch the proof of Theorem 1. See the full version of our paper https://arxiv.org/abs/2111.03560 for a full proof and other results.

B. Related Work

Optimizing the dependency on ϵ for planar undirected graphs: A closely related and somewhat orthogonal line of work initiated by Kawarabayashi, Sommer and Thorup [37] is to treat ϵ as a part of the input and optimize for the dependency on ϵ in the trade-off between space and query time. They showed that it is possible to achieve a nearly linear dependency on ϵ in the space and query time trade-off. Specifically, they constructed an oracle of $\overline{O}(n \log n)$ space and $\overline{O}(1/\epsilon)$ query time where $\overline{O}(.)$ notation hides $\mathrm{poly}(\log\log(n))$ and $\mathrm{polylog}(\frac{1}{\epsilon})$ factors. The dependency of space and query time product on ϵ in previous work [38], [52], [36] is at least quadratic. Other recent developments [28], [16] focus on improving the dependency on ϵ in the query time: Gu and Xu [28] constructed a distance oracle with $O(1)$ query time and $O(n \log n(\log n/\epsilon + 2^{O(\frac{1}{\epsilon})}))$ space; Chan and Skerpetos constructed an oracle with $O(\log \frac{1}{\epsilon})$ query time and $O(n\mathrm{poly}(\frac{1}{\epsilon})\mathrm{polylog}(n))$ space.

Exact distance oracles for directed planar graphs: Arikati et al. [6] constructed the first distance oracle for directed planar graphs with $O(S)$ space and $O(\frac{n^2}{S})$ query time for $S \in [n^{3/2}, n^2]$. Independently from the work of Arikati et al. [6], Djidiev [20] constructed two oracles with different space-query time trade-offs: (1) an oracle with $O(S)$ space and $O(\frac{n^2}{S})$ query time for $S \in [n, n^2]$ and (2) an oracle with $O(S)$ space and $\tilde{O}(\frac{n}{\sqrt{S}})$ query time for $S \in [n^{4/3}, n^{3/2}]$. ($\tilde{O}$ notation hides a $\mathrm{polylog}(n)$ factor.) Subsequent works aimed to widen the range of S in the space-query time tradeoff in the second oracle of Djidiev [20]. Specifically, Chen and Xu [18] pushed the range of S to $[n^{4/3}, n^2]$; Fakcharoenphol and Rao [21] constructed an oracle with $S = n \log n$; Mozes and Sommer [46] widened the range of S to $[n \log\log n, n^2]$.

The work of Cabello [13] focused on improve the preprocessing time; specifically, Cabello [13] constructed an oracle with $O(S)$ space, $\tilde{O}(\frac{n}{\sqrt{S}})$ query time and $O(S)$ construction time for any $S \in [n^{4/3}\log^{1/3}(n), n^2]$. Wulff-Nilsen [53] designed an oracle with constant query time and $o(n^2)$ space; this space bound has not been improved for oracles with constant query time. The first *linear space* oracle with $O(n^{\frac{1}{2}+\epsilon})$ query time for any constant $\epsilon > 0$ was obtained independently by Mozes and Sommer [46] and Nussbaum [48]; the result remains the state-of-the-art if we insist on having an oracle with linear space.

All exact distance oracles mentioned so far were constructed based on (variants) of *r-division*, a technique introduced by Frederickson [23]. None of them achieves *truly subquaratic* space and polylogarithmic query time. In a breakthrough work, Cohen-Addad, Dahlgaard, and Wulff-Nilsen [19] broke this barrier by constructing an exact distance oracle with $O(n^{\frac{5}{3}})$ space and $O(\log n)$ query time. Indeed, they obtained a more general trade-off: $O(S)$ space and $\tilde{O}(\frac{n^{5/3}}{S^{3/2}})$ query time. Their construction is based on planar Voronoi diagrams introduced by Cabello [14]. Gawrychowski et al. [27] improved the result Cohen-Addad, Dahlgaard, and Wulff-Nilsen [19] to obtain an oracle with $O(S)$ space and $\tilde{O}(\max\{1, \frac{n^{3/2}}{S}\})$ query time. Recently, Charalampopoulos et al. [17] obtained three exact distance oracles with *almost optimal* space-query time trade-offs (ignoring low order terms): (1) $\tilde{O}(n^{1+\epsilon})$ space and $\tilde{O}(1)$ query time for any constant $\epsilon > 0$, (2) $\tilde{O}(n)$ space and $O(n^{\epsilon})$ query time, and (3) $n^{1+o(1)}$ space and $n^{o(1)}$ query time. Their trade-offs were furthered improved by Long and Pettie [43] in two regimes: $n^{1+o(1)}$ space with $\tilde{O}(1)$ query time and $\tilde{O}(n)$ space with $n^{o(1)}$ query time.

Distance oracles for low dimensional metrics: The idea of using net-tree in the construction of $(1+\epsilon)$-approximate distance oracles, which we also use in this paper, was introduced by Har-Peled and Mendel [31] in metrics of constant doubling dimension. Their oracle has $O(n \log n)$ space, $O(1)$ query time, and $O(n \log n)$ construction time with ϵ and d being fixed constants. Their result improved an earlier result by Gudmundsson et al. [29] who constructed a $(1+\epsilon)$-approximate distance oracle for $t-$spanners of point sets in $\mathbb{R}^d$. In the same paper, Har-Peled and Mendel [31] presented a $(1+\epsilon)$-approximate distance oracle for doubling metrics of dimension d with $\epsilon^{-O(d)}n$ space, $O(d)$ query time and $\mathrm{poly}(n)$ construction time. That is, the query time depends linearly, instead of exponentially, on the dimension. Bartal et al. [8] improved the result of Hard-Peled and Mendel by

designing an oracle with $(\epsilon^{-O(d)} + 2^{O(d \log d)})n$ space, $O(1)$ query time and nearly linear *expected* construction time.

II. Preliminaries

Let G be a graph. We denote by $V(G)$ the vertex set of G and by $E(G)$ the edge set of G. We sometimes write $G(V, E)$ to explicitly indicate that $V(G) = V$ and $E(G) = E$, and write $G(V, E, w)$ to indicate that w is the weight function on the edges of G. We denote by $\mathsf{SP}_G(x, y)$ a shortest path between two vertices $x, y \in V$.

In this paper, we sometimes view G as a metric (V, d_G) with the shortest path distance. The *spread*, denoted by Δ, of G is defined to be the spread of (V, d_G), which is:

$$\Delta = \frac{\max_{u,v} d_G(u, v)}{\min_{u \neq v} d_G(u, v)} \tag{1}$$

Let C be a simple closed curve on the plane $\mathbb{R}^2$. Removing C from $\mathbb{R}^2$ divides the plane into two parts, called the *interior* and *exterior* of C, denoted by $\mathsf{Int}(C)$ and $\mathsf{Ext}(C)$, respectively.

Let T be a tree and x, y be two vertices of T. We denote by $T[x, y]$ the (unique) path between x and y in T.

Shortest path separators of planar graphs: Let G be given as a planar embedded graph and G_Δ be a triangulation of G. We call edges in $E(G_\Delta) \setminus E(G)$ *pseudo-edges*. Let T be a shortest path tree of G; T is also a spanning tree of G_Δ. A path P of T is *monotone* if one endpoint of P is an ancestor of all vertices in P; this endpoint is called the *root* of P.

A *shortest path separator* C of G is a fundamental cycle of G_Δ w.r.t T. Since edges of G_Δ may not be in G, C consists of two monotone paths P_1, P_2 of G rooted at the same endpoint and a (possibly imaginary) edge (u, v) between two other endpoints of P_1 and P_2. Thorup [52] and Klein [38] observed that the following lemma is implicit the proof of the planar separator theorem by Lipton and Tarjan [42].

Lemma 1 (Lipton and Tarjan [42])**.** *Let T be a shortest path tree rooted at a vertex r of an edge-weighted planar graph G. Let $\omega : V \to \mathbb{R}^+$ be a weight function on vertices of G, and $W = \sum_{u \in V(G)} \omega(u)$. There is a shortest path separator C of G such that $\max(\omega(V(G) \cap \mathsf{Int}(C)), \omega(V(G) \cap \mathsf{Ext}(C))) \leq \frac{2W}{3}$. Furthermore, C can be found in $O(n)$ time.*

r-division: Given an integer $r \geq 1$ and a planar graph G, an *r-division* of G with n vertices is a partition of the edge set of G into subsets inducing subgraphs $\{R_1, R_2, \ldots, R_k\}$ of G, called *pieces*, such that:

1) $k = O(\frac{n}{r})$ and $|V(R_i)| \leq r$ for all $i \in [1, k]$,
2) $|\partial R_i| = O(\sqrt{r})$ where ∂R_i is the set of vertices in R_i, called *boundary vertices*, such that each has at least one neighbor outside R_i.

Frederickson [23] introduced the notion of *r-division* and devised an algorithm to compute an r-division for any given r in time $O(n \log r + \frac{n}{\sqrt{r}} \log n)$. Klein, Mozes, and Sommer [39] recently improved the running time of finding an r-division to linear.

Approximate labeling schemes: Thorup [52] and Klein [38] independently came up with similar constructions of a $(1 + \epsilon)$-approximate distance oracle for an edge-weighted, undirected planar graph $G(V, E)$ of n vertices with $O(n \log n \epsilon^{-1})$ space and $O(\epsilon^{-1})$ query time. Thorup [52] showed that the oracle can be constructed in nearly linear time. Furthermore, Thorup [52] observed that the distance oracle could be distributed as a *labeling scheme*: each vertex u is assigned a label $\ell(u)$ of $O(\epsilon^{-1} \log n)$ words and there is a decoding function $\mathcal{D}$ that, given two vertices u and v, returns a $(1 + \epsilon)$-approximate distance between u and v in $O(\epsilon^{-1})$ time by looking at their labels only.

Theorem 3 (Theorem 3.19 [52], Lemma 4.1 [38])**.** *Given an edge-weighted undirected planar graph G, we can construct in $O(n\epsilon^{-2} \log^3 n)$ time a labeling scheme for $(1 + \epsilon)$-approximate distances with maximum label size $O(\log n \epsilon^{-1})$ and decoding time $O(\epsilon^{-1})$.*

Sparse cover: A (β, s, Δ)-sparse cover of a graph $G(V, E)$, denoted by $\mathcal{C} = \{C_1, \ldots, C_k\}$, is a collection of subgraphs, called *clusters*, of G such that: noitemsep

(1) The diameter of C_i is at most Δ for any $i \in [1, k]$.
(2) For every $v \in G$, $B_G(v, \Delta/\beta) \subseteq C_i$ for some $i \in [1, k]$.
(3) Every vertex $v \in V$ is contained in at most s clusters.

We say that G admits a *(β, s)-sparse covering scheme* if there exists a (β, s, Δ)-sparse cover of G for any given $\Delta \in \mathbb{R}^+$. Δ is called the diameter of the sparse cover.

Busch, LaFortune, and Tirthapura [11] constructed an $(O(1), O(1))$-sparse covering scheme for planar graphs; they slightly improved the constants in the journal version [12]. Abraham, Gavoille, Malkhi, and Wieder [2] constructed an $(O(r^2), 2^r(r + 1)!)$-sparse covering scheme for K_r-minor-free graphs.

Theorem 4 (Theorem 5.2 [11])**.** *Edge-weighted planar graphs admits an $(O(1), O(1))$-sparse covering scheme.*

Weak nets: Given an edge-weighted graph $G(V, E)$ and a set of terminals $K \subseteq V$. A *weak (r, γ)-net* of K, for $\gamma \geq 1$, is a subset of vertices $N \subseteq K$ such that: (a) $d_G(p, q) \geq r$ for every $p \neq q \in N$ and (b) for every $x \in K$, there exists a $p \in N$ such that $d_G(p, x) \leq \gamma r$. An assignment $\mathcal{A}$ associated with a weak (r, γ)-net N is a family of subsets of K such that for each $x \in N$, there exists a set, denoted by $\mathcal{A}[x] \subseteq K$, in $\mathcal{A}$ that contains x and satisfies $d_G(x, y) \leq \gamma r$ for every $y \in \mathcal{A}[x]$. We say that an assignment $\mathcal{A}$ *covers* K if $\cup_{A \in \mathcal{A}} = K$.

III. An Approximate Distance Oracle with Constant Stretch

In this section, we prove the following theorem:

Theorem 5. *Given an edge-weighted n-vertex planar graph G, there is an 8-approximate distance oracle with space $O(n)$ and query time $O(1)$ that can be constructed in worst-case time $O(n \log^3 n)$.*

Thorup [52] and Klein [38] independently gave a $(1+\epsilon)$-approximate distance oracle for undirected planar graphs with $O(\frac{1}{\epsilon} n \log n)$ space and $O(\frac{1}{\epsilon})$ query time, and Thorup showed how to obtain worst-case construction time $O(n \log^3 n/\epsilon^2)$. We will make use of the special case where $\epsilon = 1$ and we exploit that the distance oracle of Thorup is a labeling scheme, meaning that a query for vertex pair (u, v) can be answered using only the $O(\log n)$ words associated with u and v, respectively (Theorem 3):

Lemma 1 (Thourup [52])**.** *Given an undirected planar graph G with n vertices and given a subset S of the vertices of G, there is a 2-approximate distance oracle for G with $O(|S| \log n)$ space and $O(1)$ query time which can answer queries for any vertex pair in $S \times S$. The oracle can be constructed in $O(n \log^3 n)$ worst-case time.*

In the following, we shall refer to the data structure of Lemma 1 as an *S-restricted oracle* of G. We now present our approximate distance oracle for planar graph G, ignoring its efficient construction as well as some space-saving tricks for later.

The oracle keeps a 3-level recursive decomposition of G. This decomposition has an associated tree $\mathcal{T}$ where at level 0, the root is G having $r_0 = n$ vertices. Letting $r_1 = (\log r_0)^2 = (\log n)^2$, the children of G in $\mathcal{T}$ are the pieces of an r_1-division of G and these are the level 1-nodes of $\mathcal{T}$. Finally, each piece P at level 1 of $\mathcal{T}$ has as children the pieces of an r_2-division of P where $r_2 = (\log r_1)^2 = O((\log \log n)^2)$.

Each vertex $u \in V$ is associated with a leaf piece $P_2(u)$ containing u as well as the two ancestor pieces $P_0(u)$ and $P_1(u)$ of $P_2(u)$ at levels 0 and 1, respectively.

For each non-leaf piece P of $\mathcal{T}$, let $i \in \{0, 1\}$ be its level. Associated with P is a B_P-restricted oracle of P where B_P is the set of boundary vertices in the r_{i+1}-division of P.

For $i = 1, 2$, each vertex u is associated with a nearest boundary vertex $b_i(u)$ of $P_i(u)$. We also associate the distance $d_{P_i(u)}(u, b_i(u))$ with u.

For each leaf piece P of $\mathcal{T}$, we essentially store a lookup table containing 2-approximations of distances $d_P(u, v)$ for each pair of vertices u and v in P. However, we need some space-saving tricks to ensure that these tables require only linear space in total; for now, we delay the details on how to do this and just assume black box lookup tables with constant query time.

Answering a query: A query for a vertex pair (u, v) is answered as follows. For $i = 0, 1$, the query algorithm first computes

$$d_i = \begin{cases} d_{P_{i+1}(u)}(u, b_{i+1}(u)) + \tilde{d}_{P_i(u)}(b_{i+1}(u), b_{i+1}(v)) + \\ d_{P_{i+1}(v)}(v, b_{i+1}(v)) & \text{if } P_i(u) = P_i(v) \\ \infty & \text{otherwise,} \end{cases}$$

where $\tilde{d}_{P_i(u)}(b_{i+1}(u), b_{i+1}(v))$ is the output of the oracle for $P_i(u) = P_i(v)$ when queried with the pair $(b_{i+1}(u), b_{i+1}(v))$ of vertices from $B_{P_i(u)} = B_{P_i(v)}$.

If $P_2(u) = P_2(v)$, let d_2 be the 2-approximation of $d_{P_2(u)}(u, v)$ that is output using the lookup table associated with $P_2(u) = P_2(v)$. Otherwise, let $d_2 = \infty$. The query algorithm computes d_2 and then outputs $\min\{d_0, d_1, d_2\}$.

IV. An Approximate Oracle with $(1+\epsilon)$ Stretch for Undirected Planar Graphs

In this section, we construct a $(1+\epsilon)$-approximate distance oracle with linear space and constant query time for edge-weighted planar graphs; graphs in this section are *undirected*. The construction is divided into four major steps:

1) In Section IV-A, we construct a distance oracle restricted to any given subset of vertices S with a small *additive stretch*. The oracle has space $O(|S|)$ and constant query time.
2) In Section IV-B, we use the additive oracle to construct an oracle restricted to any given subset of vertices S with multiplicative $(1+\epsilon)$ stretch. The oracle has space $O(|S|)$ and constant query time. The caveat is that the stretch guarantee only applies to pairs of vertices whose distances are in $[d, \alpha d]$ for a given parameter d and a constant α.
3) In Section IV-C, we show that planar graphs with quasi-polynomial edge length have $(1+\epsilon)$-approximate distance oracle with linear space and constant query time. The construction combines three different tools: a net-tree, a weighted level ancestor data structure, and the restricted oracles in the second step.
4) Finally, in Section IV-D, we remove the assumption on the edge length of the graph.

We use $\mathsf{Space}(\mathcal{X})$ to denote the total space (in words) of a data structure $\mathcal{X}$.

A. Additive Restricted Distance Oracles

An *S-restricted distance oracle* $\mathcal{D}$ for a planar graph G with *additive stretch* t is a data structure that given any two vertices $u, v \in S$, the estimated distance returned by the oracle, denoted by $d_{\mathcal{D}}(u, v)$, satisfies:

$$d_G(u, v) \leq d_{\mathcal{D}}(u, v) \leq d_G(u, v) + t \tag{2}$$

This section is devoted to proving the following theorem.

Theorem 6. *Given an edge-weighted n-vertex planar graph $G(V, E)$ with diameter D, an error parameter $\epsilon < 1$, and*

a subset of vertices S, there is an S-restricted distance oracle $\mathcal{D}$ with $O(|S|\epsilon^{-2})$ space, $O(\epsilon^{-2})$ query time, and additive stretch ϵD. Furthermore, $\mathcal{D}$ can be constructed in $O(\epsilon^{-3}n\log^3 n)$ time.

Proof sketch: We base our construction on the idea of Kawarabayashi, Sommer, and Thorup [37], called *KST construction*, that relies on recursive decompositions of G using shortest path separators. However, our construction is different from KTS in two respects. First, we restrict the distance query to be between vertices in a given subset of vertices S, and the space bound must be linear in S, which could be much smaller than n; the space bound in KTS oracle is $\Omega(n)$. As a result, our recursive decompositions must be tailored specifically to S, and there are several properties that the decomposition must satisfy altogether. Second, our construction only has three levels, instead of $\log^*(n)$ levels as in the KTS construction. Specifically, the top level is the vertex set S and a recursive decomposition for S; the second level contains subsets of S corresponding to leaves of the recursive decomposition for S; the third level contains subsets of those in the second level. Subsets of S in the third level are small enough that we can afford to have a table lookup that contains *encodings* of approximate distances (instead of these distances themselves). With these ideas, we are able to achieve space and query time bounds independent of n, while KTS has a query time bound of $O(\log^* n)$ and a space bound of $\Omega(n)$ when ϵ is a constant. ■

B. Multiplicative Restricted Distance Oracles

In this subsection, we prove Theorem 7 that we restate below. The main tool we use in this section is sparse covers.

Theorem 7. *Given parameters $d, \epsilon > 0, \alpha > 1$ and a subset of vertices S of an edge-weighted n-vertex planar graph G, there exists an approximate distance oracle $\mathcal{D}$ with $O(|S|\alpha^2\epsilon^{-2})$ space and $O(\alpha^2\epsilon^{-2})$ query time such that the distance returned by $\mathcal{D}$ for a given query pair $(u,v) \in S\times S$, denoted by $d_{\mathcal{D}}(u,v)$, is always at least $d_G(u,v)$, and:*

$$d_{\mathcal{D}}(u,v) \leq (1+\epsilon)d_G(u,v) \text{ if } d_G(u,v) \in [d, \alpha d].$$

Furthermore, $\mathcal{D}$ can be constructed in time $O(\epsilon^{-3}\alpha^3 n\log^3 n)$.

Proof sketch: Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a $(\beta, s, \beta\alpha d)$-sparse cover of G with $\beta, s = O(1)$. We remove from $\mathcal{C}$ every cluster C_i such that $C_i \cap S = \emptyset$. Let $\mathcal{C}_S$ be the resulting set of clusters.

Let $D = \beta\alpha d = O(\alpha d)$. For each cluster $C \in \mathcal{C}_S$, let $S_C = S \cap C$. We apply Theorem 6 to construct a distance oracle $\mathcal{D}_C$ for S_C in (planar) graph C with additive stretch $\epsilon_0 D$ with $\epsilon_0 = \frac{\epsilon}{\beta\alpha} = O(\frac{\epsilon}{\alpha})$. Our data structure $\mathcal{D}$ consists of all oracles $\{\mathcal{D}_C\}_{C\in\mathcal{C}_S}$, and additionally, for each vertex $v \in S$, we will store (the id of) $\mathcal{D}_C$ such that C contains $B_G(v, \alpha d)$; C exists by property (2) of sparse covers.

Given a query pair $(u,v) \in S \times S$, we first identify in $O(1)$ time the oracle $\mathcal{D}_C$ such that C contains $B_G(v, \alpha d)$. If $d_G(u,v) < \alpha d$, u may not belong to C, and if this is the case, the oracle returns $+\infty$. Otherwise, $u \in B_G(v, \alpha d)$ and hence, $u \in C$. We then query $\mathcal{D}_C$ in $O(\epsilon_0^{-2}) = O(\alpha^2\epsilon^{-2})$ time to get the approximate distance $d_{\mathcal{D}_C}(u,v)$. We return this distance as an approximate distance by $\mathcal{D}$. ■

C. Oracles for Planar Graphs with Quasi-polynomial Spread

In this section, we construct a $(1+\epsilon)$-approximate distance oracle for planar graphs with quasi-polynomial edge weights. The oracle has linear space and constant query time.

Theorem 8. *Given an n-vertex planar graph $G(V,E,w)$ with spread $\Delta = 2^{O(\log^c n)}$ for some constant $c \geq 1$, there is a $(1+\epsilon)$-approximate distance oracle $\mathcal{D}$ with $O(n\epsilon^{-2}\log\frac{1}{\epsilon})$ space and query time $O(\epsilon^{-2})$. Furthermore, $\mathcal{D}$ can be constructed in time $O(\epsilon^{-3}n\log^{c+3} n\log\frac{1}{\epsilon})$.*

Before proving Theorem 8, we introduce the toolbox used in this section.

Weak net tree: Let $\eta \geq 1$ be a constant. We view $G(V,E,w)$ as a metric space (V, d_G) with shortest path distances. (We use points and vertices interchangeably.) The spread of (V, d_G) is Δ. Let $\mathcal{H}$ be a hierarchy of nets $V = N_0 \supseteq N_1 \supseteq \ldots \supseteq N_{\lceil\log\Delta\rceil}$ where N_{i+1} is a *weak $(2^{i+1}, \eta)$-net* of N_i for each $i \in [1, \lceil\log\Delta\rceil]$.

The hierarchy of nets naturally induces a *η-weak net tree* T where i-th level of T is N_i, and the children of each point $p \in N_{i+1}$ are points in $\mathcal{A}_{i+1}(p) \subseteq N_i$.

Since a point p can appear in many levels of a net tree T, to avoid confusion, we sometimes use (p,i) to refer to the copy of p at level i.

One operation that will be very useful in our construction is querying an ancestor of a given leaf at a given level. Such queries can be done in $O(1)$ time using a level ancestor data structure with space $O(|V(T)|)$. However, it could be that the number of nodes in T is superlinear in n; note that T only has n leaves. Thus, it is more space-efficient to work with a *compressed* version of T that compresses nodes of degree 2 in T. For a technical reason that will be apparent later, we will not compress all degree-2 nodes but a subset of them.

X-compressed net tree: Given a weak net tree T and a subset of degree 2 nodes X in T, an *X-compressed net tree*, denoted by $T_{\mathsf{cpr}(X)}$, is an edge-weighted tree obtained by sequentially contracting, in an arbitrary order, each vertex of X to one of its neighbors. That is, we replace any monotone maximal path of T, whose internal vertices are in X only, with an edge between its endpoints. The weight of each new edge (x,y) of $T_{\mathsf{cpr}(X)}$ is the distance between its endpoints in T. Observe that the weight of every edge in $T_{\mathsf{cpr}(X)}$ is at most $\lceil\log\Delta\rceil$. Note that we still label each vertex $p \in T_{\mathsf{cpr}(X)}$ with its level in T.

Since the compressed net tree has weights on its edges, we need a data structure to query the weighted level ancestor (WLA). In general, querying level ancestors in a weighted tree is a generalization of the predecessor search problem [40] that we cannot hope to have a data structure with linear space and constant query time. On the other hand, we show that it is possible to construct such a WLA data structure if the *hop depth* (defined below) of the tree is polylogarithmic.

Weighted level ancestor data structures: Let T be an *edge-weighted* tree with n vertices rooted at r where every edge $e \in T$ is assigned an integral weight $\omega(e)$. The *depth* of a node $u \in T$ is the distance $d_T(u, r)$. The depth of T is the maximum depth over all vertices of T. The *hop depth* of u is the number of edges on the path from u to r, and the hop depth of T is the maximum hop depth over all vertices in T. A WLA data structure is a data structure that, given a query of the form (u, d) where $u \in V(T)$ and $d \in \mathbb{Z}^+$, returns the *lowest ancestor* of u at depth at most d. (It could be possible that there is no ancestor of u at depth exactly d.)

Lemma 2. *Given a rooted, edge-weighted tree T with n vertices and hop depth* $\mathrm{polylog}(n)$*, there is an algorithm that runs in $O(n)$ time and constructs a level ancestor data structure with $O(n)$ space and $O(1)$ query time.*

We now have all necessary tools to prove Theorem 8. Let (V, d_G) be the shortest path metric of the input planar graph G. Let T be a η-weak net tree of (V, d_G) with $\eta = O(1)$. We define a parameter τ as follows:

$$\tau \ = \ (\frac{8}{\epsilon} + 12)\eta \tag{3}$$

For technical convenience, we extend the net tree T to include *negative levels*:

$$N_{-\lfloor \log(\tau) \rfloor - 1} = N_{-\lfloor \log(\tau) \rfloor} = \ldots = N_{-1} = N_0 = V \tag{4}$$

where we can still interpret each N_i as a $(2^i, \eta)$-net of N_{i-1} when $-\lfloor \log(\tau) \rfloor \leq i \leq 0$. Note that the minimum pairwise distance is 1.

Let N_i be the weak $(2^i, \eta)$-net associated with i-the level of T for some $i \in [-\lfloor \log(\tau) \rfloor - 1, \lceil \log \Delta \rceil]$. Let:

$$N_i^\tau = \{v | v \in N_i \wedge (\exists u \neq v \in N_i, d_G(u, v) \leq \tau 2^i)\} \tag{5}$$

That is, N_i^τ is the set of net points in N_i that have at least one other net point within distance $\tau 2^i$. While the set N_i^τ has several interesting properties that can be exploited to construct our distance oracle, it is unclear how to compute N_i^τ efficiently without considering distances between all pairs of points in N_i, which could costs $\Omega(n^2)$ time. We instead consider a bigger set $N_i^{\tau,+}$ defined below that can be computed in $O(n)$ time. (Using $N_i^{\tau,+}$ instead of N_i^τ makes the argument for space bound somewhat more complicated, but the bound we get remains the same.) When $i = -\lfloor \log(\tau) \rfloor - 1$, $N_i^\tau = \emptyset$.

Construct $N_i^{\tau,+}$: Let $\mathcal{C}_i$ be a $(\beta, s, \beta\tau 2^i)$-sparse cover of G with $\beta = s = O(1)$. For each set $C \in \mathcal{C}_i$, if $|C \cap N_i| \leq 1$, we remove C from $\mathcal{C}_i$. Let $\mathcal{C}_i^-$ be the resulting cover. We then define $N_i^{\tau,+}$ to be the set of all points $v \in N_i$ such that there exists $C \in \mathcal{C}_i^-$ containing v.

Distance oracle construction: Let T_2 be the set of all degree-2 vertices in T. The oracle $\mathcal{D}$ consists of:

noitemsep

1) Oracles $\{\mathcal{D}_i\}_{i=-\lfloor \log(\tau) \rfloor - 1}^{\lceil \log \Delta \rceil}$ where $\mathcal{D}_i$ is the distance oracle for $N_i^{\tau,+}$ constructed by applying Theorem 7 with $d = (\tau/2 - 2\eta)2^i$ and $\alpha = \frac{2\tau}{\tau - 4\eta}$.
2) A constant stretch oracle $\mathcal{D}_c$ for $G(V, E, w)$ constructed by applying Theorem 5; $\mathsf{Space}(\mathcal{D}_c) = O(n)$.
3) An X-compressed net tree $T_{\mathsf{cpr}(X)}$ with
$$X = T_2 \bigcap (V(T) \setminus (\cup_{i=-\lfloor \log(\tau) \rfloor - 1}^{\lceil \log \Delta \rceil} N_i^{\tau,+}) \tag{6}$$
We store at each node of $T_{\mathsf{cpr}(X)}$ its depth in the tree.
4) A weighted level ancestor data structure $\mathcal{W}$ for $T_{\mathsf{cpr}(X)}$ by Lemma 2.

Oracle query: Given a query pair (u, v), $\mathcal{D}_c$ returns $d_{\mathcal{D}_c}(u, v)$ such that $d_G(u, v) \leq d_{\mathcal{D}_c}(u, v) \leq 5 \cdot d_G(u, v)$ by Theorem 5. We define:

$$\bar{i} = \lceil \log_2 \frac{2(1+\epsilon) d_{\mathcal{D}_c}(u, v)}{\tau - 4\eta} \rceil \tag{7}$$

Since $\log(\Delta) = \mathrm{polylog}(n)$, the hop depth of $T_{\mathsf{cpr}(X)}$ is $\mathrm{polylog}(n)$. Thus, each level ancestor query in $\mathcal{W}$ can be answered in $O(1)$ time by Lemma 2 if we assume that $\bar{i}$ can be obtained from $d_{\mathcal{D}_c}(u, v)$ in $O(1)$ time. This assumption requires justification since we are not assuming that the logarithm nor the ceiling function can be computed in $O(1)$ time. However, we omit the details here since we will focus on essentially the same problem in Section IV-D.

For each $j \in [\bar{i} - 5, \bar{i}]$, in $O(1)$ time, we query the ancestors $p_j(u)$ and $p_j(v)$ in $T_{\mathsf{cpr}(X)}$ at level j, or equivalently, at depth $\lceil \log \Delta \rceil + \lfloor \log(\tau) \rfloor + 1 - j$, of u and v, respectively, using $\mathcal{W}$. We then query the distance between $p_j(u)$ and $p_j(v)$ using oracle $\mathcal{D}_j$ in $O(\alpha^2 \epsilon^{-2}) = O(\epsilon^{-2})$ time. Here we use the fact that:

$$\alpha \ = \ \frac{2\tau}{\tau - 4\eta} \ \overset{\text{Eq. 3}}{=} \ \frac{2(8/\epsilon + 12)}{8/\epsilon + 8} \ = \ \frac{3\epsilon + 2}{\epsilon + 1} \ \leq \ 3. \tag{8}$$

Finally, we return:

$$d_{\mathcal{D}}(u, v) \overset{\text{def}}{=} \min_{j \in [\bar{i} - 5, \bar{i}]} (d_{\mathcal{D}_j}(p_j(u), p_j(v)) + \eta 2^{j+2}) \tag{9}$$

Using a lookup table to precompute powers of 2, we then get a total query time of $O(\epsilon^{-2})$. We note that there could be possible that the ancestors returned by $\mathcal{W}$ are not

$p_j(u)$ and/or $p_j(v)$ because they may be compressed in $(T_{\mathsf{cpr}(X)}, \varphi_{\mathsf{cpr}(X)})$; we can easily check if this is the case by comparing the depth of the returned ancestors and the desired depth in T, which is $\lceil \log \Delta \rceil + \lfloor \log(\tau) \rfloor + 1 - j$. In this case, we will exclude j in computing the approximate distance in Equation (9).

D. Removing the Spread Assumption

In this subsection, we remove the assumption on the spread using the contraction technique of Kawarabayashi, Sommer, and Thorup [37]. The same technique was used in previous results [28], [16]. The idea is to have for each scale $r \in \{2^0, 2^1, \ldots, 2^{\lceil \log \Delta \rceil}\}$, a graph G_r obtained from G by removing every edge of weight more than r and contracting every edge of weight less than $\frac{r}{n^2}$. Then, a distance oracle is constructed for each G_r and the total space bound (typically of $\Omega(n \log n)$) follows from the observation that each edge $e \in G$ belongs to at most $O(\log n)$ different graphs G_r (for different values of r).

To show a linear space bound, we need the scale r to be bigger, so that each edge $e \in G$ belongs to at most $O(1)$ graphs G_r; we naturally choose the scale to be $\{n^0, n^4, \ldots, n^{4i}, \ldots, n^{\lceil \log_{n^4} \Delta \rceil}\}$. To construct each G_r, we apply the same idea: delete every edge of weight more than $n^4 r$ and contract every edge of weight at most $\frac{r}{n^2}$. It follows directly from the construction that each edge e belongs to at most 2 graphs G_r. The issue now is that, while the spread of G_r is polynomial in n, it could be exponential in the *number of vertices of* G_r. In this case, we use the bit-packing technique that we formalize in the following lemma.

Lemma 3. *Let $G(V, E, w)$ be an undirected and edge-weighted planar graph with n vertices. If the machine word size is $\omega = \Omega(\log n^3)$, then in $O(\epsilon^{-2} n \log^3 n)$ time, we can construct a $(1+\epsilon)$-approximate distance oracle for $G(V, E, w)$ with $O(n\epsilon^{-1})$ space and $O(\epsilon^{-1})$ query time.*

ACKNOWLEDGMENT

Hung Le is supported by the National Science Foundation under Grant No. CCF-2121952. We thank Arnold Filtser for informing us about [7].

REFERENCES

[1] I. Abraham, S. Chechik, and C. Gavoille. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 1199–1218, 2012.

[2] 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.

[3] M. Ajtai. A lower bound for finding predecessors in yao's cell probe model. *Combinatorica*, 8(3):235–247, 1988.

[4] N. Alon and B. Schieber. Optimal preprocessing for answering on-line productqueries. *Technical Report, Department of Computer Science, School of Mathematical Sciences, Tel Aviv University*, 1987.

[5] S. Alstrup and J. Holm. Improved algorithms for finding level ancestors in dynamic trees. In *Automata, Languages and Programming*, ICALP '00, pages 73–84. Springer Berlin Heidelberg, 2000.

[6] 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.

[7] Y. Bartal, N. Fandina, and O. Neiman. Covering Metric Spaces by Few Trees. In *The 46th International Colloquium on Automata, Languages, and Programming*, ICALP '19, pages 20:1–20:16, 2019.

[8] Y. Bartal, L. Gottlieb, T. Kopelowitz, M. Lewenstein, and L. Roditty. Fast, precise and dynamic distance queries. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 840–853, 2011.

[9] P. Beame and F. E. Fich. Optimal bounds for the predecessor problem and related problems. *Journal of Computer and System Sciences*, 65(1):38–72, 2002.

[10] M. A. Bender and M. Farach-Colton. The lca problem revisited. In *Latin American Symposium on Theoretical Informatics (LATIN '00)*, pages 88–94, 2000.

[11] 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.

[12] C. Busch, R. LaFortune, and S. Tirthapura. Sparse covers for planar graphs and graphs that exclude a fixed minor. *Algorithmica*, 69(3):658–684, 2013.

[13] S. Cabello. Many distances in planar graphs. *Algorithmica*, 62(1-2):361–381, 2010. Announced at SODA '06.

[14] 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.

[15] T.-H. Hubert Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics. *ACM Trans. Algorithms*, 12(4):55:1–55:22, 2016. Preliminary version appeared in SODA 2005.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72(5):868–889, 2006. Annouced at FOCS'01.

[22] M. Farach and S. Muthukrishnan. Perfect hashing for strings: Formalization and algorithms. In *In Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching*, CMP '96, pages 130–140, 1996.

[23] G. Frederickson. Fast algorithms for shortest paths in planar graphs with applications. *SIAM Journal on Computing*, 16:1004–1022, 1987.

[24] M. L. Fredman and D. E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47(3):424–436, 1993.

[25] V. Fredslund-Hansen, S. Mozes, and C. Wulff-Nilsen. Truly subquadratic exact distance oracles with constant query time for planar graphs. *arXiv preprint arXiv:2009.14716*, 2020. https://arxiv.org/abs/2009.14716.

[26] P. Gawrychowski, M. Lewenstein, and P. K. Nicholson. In *Proceedings of the 22nd Annual European Symposium on Algorithms*, ESA '14, pages 455–466, 2014.

[27] 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*, number SODA '18, pages 515–529, 2018.

[28] 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.

[29] J. Gudmundsson, C. Levcopoulos, G. Narasimhan, and M. H. M. Smid. Approximate distance oracles for geometric spanners. *ACM Transactions on Algorithms*, 4(1), 2008.

[30] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. In *Proceedings of the 21st annual symposium on Computational geometry*, SoCG'05, 150–158, 2005.

[31] S. Har-Peled and M. Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

[32] D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, 1984.

[33] 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.

[34] Jacob Holm, Eva Rotenberg, and Mikkel Thorup. Planar reachability in linear space and constant time. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 370–389. IEEE Computer Society, 2015.

[35] Jesper Jansson, Kunihiko Sadakane, and Wing-Kin Sung. Ultra-succinct representation of ordered trees. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 575–584. SIAM, 2007.

[36] 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.

[37] 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.

[38] 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.

[39] P. N. Klein, S. Mozes, and C. Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, STOC '13, pages 505—-514, 2013.

[40] T. Kopelowitz and M. Lewenstein. Dynamic weighted ancestors. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '07, page 565–574, 2007.

[41] L. Kowalik and M. Kurowski. Short path queries in planar graphs in constant time. In *Proceedings of the 35th ACM symposium on Theory of computing, STOC'03*, page 143–148, 2003.

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

[43] 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.

[44] P. B. Miltersen. Lower bounds for union-split-find related problems on random access machines. In *Proceedings of the 26th annual ACM symposium on Theory of computing*, STOC '94, pages 625—-634, 1994.

[45] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. *Journal of Computer and System Sciences*, 57(1):37–49, 1998.

[46] 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.

[47] L. F. Muller and M. Zachariasen. Fast and compact oracles for approximate distances in planar graphs. In *Algorithms–ESA 2007*, ESA '07, pages 657–668, 2007.

[48] Y. Nussbaum. Improved distance queries in planar graphs. In *Proceedings of the 12th International Conference on Algorithms and Data Structures, WADS'11*, pages 642–653, 2011.

[49] M. Pătraşcu and M. Thorup. Time-space trade-offs for predecessor search. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, STOC '06, pages 232—240, 2006. Full version at https://arxiv.org/pdf/cs/0603043.pdf.

[50] P. Sen and S. Venkatesh. Lower bounds for predecessor searching in the cell probe model. *Journal of Computer and System Sciences*, 74(3):364–385, 2006.

[51] C. Sommer. Shortest-path queries in static networks. *ACM Computing Surveys*, 46(4), 2014.

[52] 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.

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

[54] 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.