Approximate Shortest Paths and Distance Oracles in Weighted Unit-Disk Graphs*

Timothy M. Chan[†] *and Dimitrios Skrepetos*[‡]

Abstract. We present the first near-linear-time algorithm that computes a $(1+\varepsilon)$-approximation of the *diameter* of a weighted unit-disk graph of n vertices. Our algorithm requires $O\left(n\log^2 n\right)$ time for any constant $\varepsilon > 0$, so we considerably improve upon the near-$O\left(n^{3/2}\right)$-time algorithm of Gao and Zhang [17]. Using similar ideas we develop $(1+\varepsilon)$-approximate *distance oracles* of $O(1)$ query time with a likewise improvement in the preprocessing time, specifically from near $O\left(n^{3/2}\right)$ to $O\left(n\log^3 n\right)$. We also obtain similar new results for a number of related problems in the weighted unit-disk graph metric such as the radius and the bichromatic closest pair.

As a further application we employ our distance oracle, along with additional ideas, to solve the $(1+\varepsilon)$-approximate *all-pairs bounded-leg shortest paths* (apBLSP) problem for a set of n planar points. Our data structure requires $O\left(n^2\log n\right)$ space, $O(\log\log n)$ query time, and nearly $O\left(n^{2.579}\right)$ preprocessing time for any constant $\varepsilon > 0$, and is the first that breaks the near-cubic preprocessing time bound given by Roditty and Segal [31].

1 Introduction

In this paper we study distance-related problems in *weighted unit-disk graphs.* Such a graph G is defined as the intersection graph of a set of unit-diameter disks in the plane. That is, vertices correspond to a set S of points (the disk centers), and there is an edge between $u, v \in S$ if and only if u and v are at Euclidean distance at most one. The *weight* or *length* of uv is the Euclidean distance between u and v. We assume that the graph G is not explicitly constructed, but is instead implicitly represented by the point set S. Unit-disk graphs have been widely studied (see e.g. [6, 9, 17, 21, 22, 28, 35]), as they can model ad-hoc communication networks.

We are interested in various fundamental shortest-path problems in such a weighted unit-disk graph G, notably:

- designing *approximate distance oracles*, i.e., data structures so that given any $s, t \in V$, we can quickly compute a $(1+\varepsilon)$-approximation of the shortest-path distance $dist_G[s,t]$ between s and t in G, and

*Work of the first author was in part supported by NSF grant CCF-1814026.

[†] *Department of Computer Science, University of Illinois at Urbana-Champaign*, tmc@illinois.edu

[‡] *Cheriton School of Computer Science, University of Waterloo*, dskrepet@uwaterloo.ca

The conference version of this paper appeared in SoCG 2018 [11].

- designing algorithms for computing a $(1+\varepsilon)$-approximation of various parameters such as the *diameter* $(\max_{s,t\in V} dist_G[s,t])$, the *radius* $(\min_{s\in V}\max_{t\in V} dist_G[s,t])$, the *bichromatic closest pair distance* between two sets $A, B \subset S$ $(\min_{a\in A,\, b\in B} dist_G[s,t])$, and so on.

Besides practical motivation from wireless networks, this collection of problems is interesting from a theoretical perspective as well because it connects computational geometry with graph data structures—indeed, our new algorithms will draw on ideas from both areas.

We now discuss the technical challenges that arise when trying to address $(1+\varepsilon)$-approximate shortest-path problems in weighted unit-disk graphs. For the rest of Section 1 we assume that $\varepsilon > 0$ is a constant.

Planar graph techniques There has already been an extensive body of work on shortest-path problems and distance oracles in planar graphs. For example, Thorup [32] gave $(1+\varepsilon)$-approximate distance oracles in weighted, undirected planar graphs with $O(n \operatorname{polylog} n)$ preprocessing time and space and $O(1)$ query time. Subsequent work [26, 24, 20, 10] improved the logarithmic factors and examined the dependency of the hidden factors on ε. Weimann and Yuster [34] presented a $(1+\varepsilon)$-approximation algorithm for the diameter in weighted, undirected planar graphs that requires $O\left(n\log^4 n\right)$ time. Chan and Skrepetos [10] improved the running time to $O\left(n\log^2 n\right)$ and also reduced the ε-dependency from exponential to polynomial. There has also been an exciting recent breakthrough on *exact* algorithms for diameter and distance oracles in planar graphs by Cabello [5] and subsequent researchers [18, 12, 19].

All the above approximation results for planar graphs rely heavily on *shortest-path separators*: a set of shortest paths with a common root, such that the removal of their vertices decomposes the graph into at least two disjoint subgraphs. Unfortunately, such separators do not seem directly applicable to a unit-disk graph G, and not only because G may be dense. Indeed, by grid rounding we can construct a sparse weighted graph $\widehat{G}$, such that it (i) approximately preserves distances in the original unit-disk graph G (e.g., see the proof of Lemma 2), and (ii) is "nearly planar", in the sense that each edge intersects at most a constant number of other edges. However, even for such a graph it is not clear how to define a shortest-path separator that divides it cleanly into an inside and an outside because edges may "cross" over the separator. At least one prior paper [35] worked on extending shortest-path separators to unit-disk graphs, but the construction was complicated and achieved only constant approximation factors.

Gao and Zhang's WSPD-based technique Gao and Zhang [17], in a seminal paper, obtained the first nontrivial set of results on shortest-path problems in weighted unit-disk graphs. To do so, they adapted a familiar technique in computational geometry, namely the *well-separated pair decomposition* (WSPD), introduced by Callahan and Kosaraju [7] for solving proximity problems in the Euclidean (or L_p) metric. Gao and Zhang proposed a new variant of WSPDs for the weighted unit-disk graph metric and proved that any set of n planar points has a WSPD of $O(n\log n)$ size under the new definition.

Consequently, Gao and Zhang obtained a $(1+\varepsilon)$-approximate distance oracle with $O(n \log n)$ size and $O(1)$ query time. Unfortunately, the preprocessing time, $O\left(n^{3/2}\sqrt{\log n}\right)$, is quite high, and becomes the bottleneck when the technique is applied to offline problems, such as computing the diameter. However, the issue is not constructing the WSPD itself, which can be done in near-linear time, but computing the shortest-path distances of a near-linear number of vertex pairs in the "nearly planar" graph $\widehat{G}$ mentioned above. Gao and Zhang computed these distances in almost $n^{3/2}$ time by showing that $\widehat{G}$ has a balanced separator [29, 16] and adapting a known exact distance oracle for planar graphs [1]. Cabello [4] has given an improved algorithm for computing multiple distances in planar graphs, and if it could be adapted here, the running time would be reduced to around $n^{4/3}$. However, near-linear time still seems out of reach with current techniques.

Gao and Zhang [17] observed that the preprocessing time can be made near-linear when the approximation factor is a certain constant (about 2.42). However, their solution cannot be applied to obtain a $1+\varepsilon$ approximation factor and also has no new implication to the diameter problem (for which a near-2-approximation is easy by running a single-source shortest paths algorithm).

Our technique In Section 2, we give the first near-linear-time algorithm that computes a $(1+\varepsilon)$-approximation of the diameter of a weighted unit-disk graph of n vertices. Specifically our algorithm runs in $O\left(n \log^2 n\right)$ time, and its dependencies on the hidden factors on ε are polynomial. A similar result holds for $(1+\varepsilon)$-approximate distance oracles: we obtain $O\left(n \log^3 n\right)$ preprocessing time, $O(n \log n)$ space, and $O(1)$ query time. We have thus answered one of the main questions left open in Gao and Zhang's paper. Applications to other related problems follow.

Our approach is conceptually simple: we just go back to known shortest-path separator techniques for planar graphs [32, 24]!

But how do we get around the issue that unit-disk graphs do not have nice path separators? The idea is to first find a *spanner* subgraph H which is planar and has constant approximation/stretch factor—such spanners are already known to exist for unit-disk graphs [28], fortunately (and they were also used by Gao and Zhang [17]). We then apply divide-and-conquer over the shortest-path separator decomposition tree for H instead of G.

Although the above plan may sound obvious in hindsight, the details are tricky to get right. For example, how could the use of an $O(1)$-spanner eventually lead to a $1+\varepsilon$ approximation factor? The known divide-and-conquer approaches for planar graphs select a small number of vertices, called *portals*, along each separator and compute distances from each with a single-source shortest paths (SSSP) algorithm. That works well because a shortest path in a planar graph crosses a separator only at vertices. In our case, however, we need to use the original (non-planar, unit-disk) graph G when computing distances from portals, but therein a shortest path could "cross" the separator over an edge. We show that we can nevertheless re-route such a path to pass through a separator vertex without increasing the length by much, by using the fact that H is an $O(1)$-spanner.

Application to all-pairs bounded-leg shortest paths In the last part of the paper, we discuss a problem called *all-pairs bounded-leg shortest paths* (apBSLP). Given a set S of n planar points, we define $G_{\leq L}$ to be the subgraph of the complete Euclidean graph of S that contains only edges of weight at most L. Then, we want to preprocess S, such that given two points $s, t \in S$ and any positive number L, we can quickly compute a $(1+\varepsilon)$-approximation of the length of the s-to-t shortest path in $G_{\leq L}$ (i.e., the shortest path under the restriction that each leg of the trip has length bounded by L). To see the connection of apBLSP with the earlier problems, note that $G_{\leq L}$ for each fixed L is a weighted unit-disk graph, after rescaling the radii. One important difference however is that L is not fixed in apBLSP, and we want to answer queries for any of the $\binom{n}{2}$ combinatorially different L's.

Bose et al. [2] introduced the problem and described a method with $O\left(n^5\right)$ preprocessing time, $O\left(n^2 \log n\right)$ space, and $O(\log n)$ query time. Roditty and Segal [31] improved the preprocessing time to roughly $O\left(n^3\right)$ and the query time to $O(\log\log n)$. They also gave a data structure for the variation of the problem in general weighted, directed graphs with $\widetilde{O}\left(n^{2.5}\right)$ space and $\widetilde{O}\left(n^4\right)$ preprocessing time. Duan and Pettie [14] improved the space and the preprocessing time of Roditty and Segal's result in general graphs to $\widetilde{O}\left(n^2\right)$ and $\widetilde{O}\left(n^3\right)$ respectively. In a recent independent work that appeared after the conference version of this paper, Duan and Ren [15] presented the first data structure for the problem in general graphs with subcubic preprocessing time, namely $\widetilde{O}\left(n^{2.6865}\right)$ (the space remains near quadratic).

We apply our $(1+\varepsilon)$-approximate distance oracle for weighted unit-disk graphs, along with additional new ideas, in Section 3 to obtain the first data structure for $(1+\varepsilon)$-approximate apBLSP in the Euclidean metric that breaks the cubic preprocessing barrier given by Roditty and Segal: namely, we obtain nearly $\widetilde{O}\left(n^{2.667}\right)$ preprocessing time, while the space and query time remain $\widetilde{O}\left(n^2\right)$ and $O(\log\log n)$ respectively as in [31]. With fast matrix multiplication, we can further reduce the preprocessing time to $O\left(n^{2.579}\right)$, assuming a polynomial bound on the *spread*, i.e., the ratio of the maximum to the minimum Euclidean distance over all pairs of points in V.

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

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

2 Approximate diameter and distance oracles

Let S be a set of planar points whose weighted unit-disk graph G has diameter Δ. A key subproblem for both (i) computing a $(1+\varepsilon)$-approximation of the diameter of G and (ii) building a $(1+\varepsilon)$-approximate distance oracle for G is the construction of a distance oracle with *additive stretch* $O(\varepsilon\Delta)$: a data structure, such that given any $s, t \in S$, we can quickly compute a value $\widetilde{d}$ with $dist_G[s,t] \leq \widetilde{d} \leq dist_G[s,t] + O(\varepsilon\Delta)$. We describe our solution for that subproblem in Section 2.2, after giving two preliminary ingredients in Section 2.1. Then, we show in Section 2.3 how to employ that oracle, along with existing techniques, to address the two original problems.

2.1 Preliminaries

The first ingredient we need is the existence of a planar spanner with constant stretch factor in any weighted unit-disk graph. More formally, a subgraph H is a *c-spanner* of a graph $G = (V, E)$ if for every $s, t \in V$, $dist_G[s,t] \leq dist_H[s,t] \leq c\,dist_G[s,t]$.

Lemma 1. (Planar spanner) *Given a set S of n planar points, we can find a subgraph H of its weighted unit-disk graph G in $O(n \log n)$ time, such that H is (i) a planar graph and (ii) a c-*spanner *of G, for some absolute constant $c > 0$.*

Li, Calinescu, and Wan [28] proved the above lemma with $c = 2.42$ by simply building the Delaunay triangulation of the given points and discarding edges of weight more than one. Li et al.'s analysis builds upon existing work on the stretch factor of the Delaunay triangulation for complete Euclidean graphs [13, 25].

The second ingredient is an efficient algorithm for the single-source shortest paths (SSSP) problem in weighted unit-disk graphs. The currently best exact result is due to Cabello and Jejčič [6], requires $O\left(n \log^{12+o(1)} n\right)$ time and employs complicated dynamic data structures for additively weighted Voronoi diagrams [8, 23]. For our purposes though, it suffices to consider the $(1+O(\varepsilon))$-approximate version of the problem instead, i.e., given a set of points S and a source $s \in S$, compute a path of length $\widetilde{d}[s,t]$ for each $t \in S$, such that $dist_G[s,t] \leq \widetilde{d}[s,t] \in (1+O(\varepsilon))dist_G[s,t]$. Our algorithm first finds a sparse graph $\widehat{G}$ that $(1+O(\varepsilon))$-approximately preserves distances in G. Then, it runs Dijkstra's algorithm in $\widehat{G}$. Sparsification in weighted unit-disk graphs has been used before (e.g., see [17, Section 4.2]).

Lemma 2. (Approximate SSSP) *Given a set S of n planar points, we can solve the $(1+\varepsilon)$-approximate SSSP problem in its weighted unit-disk graph G in $O\left((1/\varepsilon)^2 n \log n\right)$ time.*

Proof. First, we build a uniform grid of side length ε and construct a sparse weighted graph $\widehat{G}$ by placing a vertex at each non-empty grid cell and an edge between every two such cells c and c' if and only if there exist points $p \in c$ and $p' \in c'$ with $\|pp'\| \leq 1$. The weight of that edge is equal to the distance between the center of c and the center of c'. Each grid cell has at most $O\left((1/\varepsilon)^2\right)$ neighbors, so $\widehat{G}$ has at most $O\left((1/\varepsilon)^2 n\right)$ edges and can be constructed in $O\left((1/\varepsilon)^2 n \log n\right)$ time by a Euclidean bichromatic closest pair algorithm [30] over $O\left((1/\varepsilon)^2 n\right)$ pairs of grid cells, with a total size of $O\left((1/\varepsilon)^2 n\right)$ points.

We now prove that for any points s and t of S with $\|st\| > 1$, $dist_{\widehat{G}}[c_s, c_t] \leq (1 + O(\varepsilon))dist_G[s,t]$, where c_s and c_t are the grid cells containing s and t respectively. Let $p_0 p_1 \cdots p_\ell$ be an s-to-t shortest path in G, with $p_0 = s$ and $p_\ell = t$. Two consecutive edges therein have lengths whose sum is at least one because otherwise we could take a short-cut and obtain a shorter path. Thus, $dist_G[s,t] \geq \lfloor \ell/2 \rfloor$. Consider the path $c_0 c_1 \cdots c_\ell$ in $\widehat{G}$, where $c_0 = c_s$, $c_\ell = c_t$, and each other c_i is the cell that contains p_i. Since for each i, the weight of the edge $c_i c_{i+1}$ in $\widehat{G}$ is at most $\|p_i p_{i+1}\| + O(\varepsilon)$, it follows that $dist_{\widehat{G}}[c_s, c_t] \leq dist_G[s,t] + O(\varepsilon\ell) \leq (1 + O(\varepsilon))dist_G[s,t]$.

Conversely, we prove that $dist_G[s,t] \leq (1 + O(\varepsilon))dist_{\widehat{G}}[c_s, c_t]$. Let $c_0 c_1 \cdots c_\ell$ be a c_s-to-c_t shortest path in $\widehat{G}$, with $c_0 = c_s$ and $c_\ell = c_t$. Two consecutive edges therein have weights in $\widehat{G}$ whose sum is at least $1 - c\varepsilon$ (for a sufficiently large constant c) because otherwise we could take a short-cut and obtain a shorter path. Thus, $dist_{\widehat{G}}[c_s, c_t] \geq \Omega(\ell)$. Let (p_i, q_i) is the bichromatic closest pair of c_i and c_{i+1} (which has already been computed during preprocessing). Note that there is an edge from q_i to p_{i+1} in G of distance $O(\varepsilon)$, since q_i and p_{i+1} lie in a common grid cell. Consider the path $sp_0q_0p_1q_1 \cdots p_\ell q_\ell t$ in G. Since for each i, $\|p_i q_i\|$ is at most the weight of the edge $c_i c_{i+1}$ in $\widehat{G}$ plus $O(\varepsilon)$, it follows that $dist_G[s,t] \leq dist_{\widehat{G}}[c_s, c_t] + O(\varepsilon\ell) \leq (1 + O(\varepsilon))dist_{\widehat{G}}[c_s, c_t]$.

Given a source $s \in S$, we can invoke Dijkstra's algorithm in $\widehat{G}$ to compute the shortest path tree from c_s and return $\widetilde{d}[s,t] = (1 + c'\varepsilon)dist_{\widehat{G}}[c_s, c_t]$ for each $t \in S$, where c_s and c_t are the grid cells that contain s and t, respectively, and c' is a sufficiently large constant. From the previous paragraph, we can find a corresponding path of the desired length. Note that if $\|st\| \leq 1$, the shortest distance is trivially $\|st\|$. □

2.2 Distance oracles with additive stretch

We now describe a distance oracle with additive stretch $O(\varepsilon\Delta)$ for an arbitrary weighted, undirected graph $G = (V, E)$ of n vertices and of diameter Δ that has the following properties, which are the only ones needed from weighted unit-disk graphs.

(I) There exists a planar c-spanner H of G, for some constant $c > 0$.

(II) For any induced subgraph of G with n' vertices, the $(1+\varepsilon)$-approximate SSSP problem can be solved in $T(n')$ time, for some function $T(\cdot)$ such that $T(n_1) + T(n_2) \leq T(n_1 + n_2)$.

(III) Every edge weight in G is at most $\varepsilon\Delta$.

If G is a weighted unit-disk graph, Lemmas 1 and 2 imply (I) and (II), respectively, where $c = 2.42$ and $T(n') = O\left((1/\varepsilon)^2 n' \log n'\right)$, and (III) holds as long as $\Delta \geq 1/\varepsilon$.

Shortest-path separators in H. Although G may not have nice shortest-path separators, we know by planarity that H does. Thus we apply a known shortest-path separator decomposition for H, namely the version of Kawarabayashi, Sommer, and Thorup [24, Section 3.1], to compute in $O(n \log n)$ time a *decomposition tree* $\mathcal{T}$ with the following properties.

- $\mathcal{T}$ has $O(\log n)$ height.
- Each node μ of $\mathcal{T}$ is associated with a subset $V^{(\mu)} \subseteq V$. The subsets $V^{(\nu)}$ over all children ν of μ are disjoint and contained in $V^{(\mu)}$. If μ is the root, $V^{(\mu)} = V$; if μ is a leaf, $V^{(\mu)}$ has $O(1)$ size.
- Each non-leaf node μ of $\mathcal{T}$ is associated with a set of $O(1)$ paths, called *separator paths*, which are classified as "internal" and "external". The internal separator paths cover precisely all vertices of $V^{(\mu)} - \bigcup_{\text{child } \nu \text{ of } \mu} V^{(\nu)}$, while the external separator paths are outside of $V^{(\mu)}$.
- For each child ν of a non-leaf node μ, every neighbor of the vertices of $V^{(\nu)}$ in H is either in $V^{(\nu)}$ or in one of the (internal or external) separator paths at μ.
- Each separator path is a shortest path in H and, in particular, has length at most the diameter $\Delta(H)$ of H (which is at most $c\Delta$).

Our data structure. To construct an additive oracle with $O(\varepsilon\Delta)$ stretch for G, we construct the above decomposition tree $\mathcal{T}$ and augment it with extra information, as follows. Let μ be an internal node of $\mathcal{T}$, and let σ be one of its internal separator paths. Since σ has length at most $\Delta(H) \leq c\Delta$, we can select with a linear walk a set of $O(1/\varepsilon)$ vertices thereon, called *portals*, such that each consecutive pair of them is at distance at most $\Theta(\varepsilon\Delta)$ on σ, unless they are adjacent in σ.

Let $P^{(\mu)}$ denote the set of all portals over all internal separator paths at a non-leaf node μ of $\mathcal{T}$. For each such node and for each $p \in P^{(\mu)}$ and $v \in V^{(\mu)}$, we invoke $O(1/\varepsilon)$ times the SSSP algorithm from Property (II) to compute a $(1+\varepsilon)$-approximation, $\widetilde{d}_{\mu}[p,v]$, of the shortest-path distance from p to v in the subgraph of G induced by $V^{(\mu)}$. Then, for each leaf μ, we just find and store all pairwise distances in the subgraph of G that is induced by $V^{(\mu)}$. Overall, our oracle requires $O((1/\varepsilon)T(n) \cdot \log n)$ preprocessing time and $O((1/\varepsilon)n \cdot \log n)$ space.

Query algorithm. Given two vertices $s,t \in V$, we first identify all $O(\log n)$ nodes μ in $\mathcal{T}$, such that $s \in V^{(\mu)}$ and $t \in V^{(\mu)}$. To do so, we trivially start from the root and go down the tree along a path. For each such node μ, if μ is not a leaf, we compute a value $\widetilde{\delta}_{\mu}[s,t] = \min_{p\in P^{(\mu)}} \{\widetilde{d}_{\mu}[s,p] + \widetilde{d}_{\mu}[p,t]\}$ in $O(1/\varepsilon)$ time. If μ is a leaf, $\widetilde{d}_{\mu}[s,t]$ is the exact shortest-path distance in the subgraph of G induced by $V^{(\mu)}$, which we have already computed. Finally we return the minimum, $\widetilde{\delta}[s,t]$, over all $\widetilde{\delta}_{\mu}[s,t]$. The total query time is $O((1/\varepsilon)\log n)$.

Stretch analysis. We want to prove that for any $s,t \in V$, the value $\widetilde{d}[s,t]$ that our oracle returns satisfies that $dist_G[s,t] \leq \widetilde{d}[s,t] \leq dist_G[s,t] + O(\varepsilon\Delta)$. The left side of the inequality clearly holds because $\widetilde{d}[s,t]$ corresponds to the length of an s-to-t path in a subgraph of G. To prove the right side, let π be the shortest s-to-t path in G, and let μ be the lowest node in $\mathcal{T}$, such that all vertices of π lie in $V^{(\mu)}$. We assume that μ is a non-leaf node because otherwise we have already computed $dist_G[s,t]$ exactly.

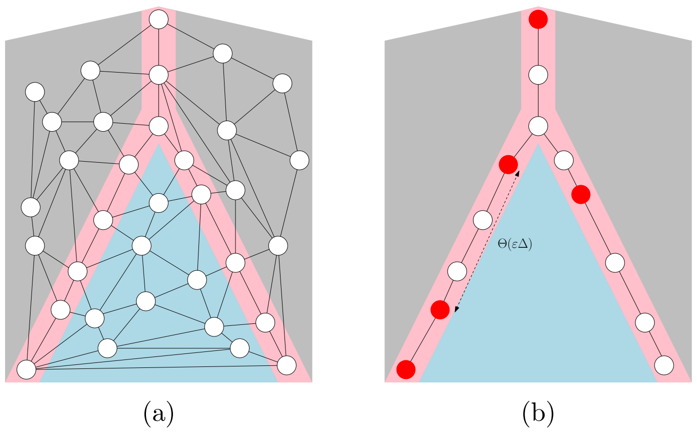

Figure 1: (a): Separator paths that divide a planar graph H into two components (corresponding to two children of the root of the decomposition tree $\mathcal{T}$). (b): Portals along the separator.

Although π is a path in the (not necessarily planar) graph G, we show that it is possible to re-route it to pass through a vertex on a separator path of μ without increasing its length by much.

Claim 1. (Detour through a separator vertex) *There exists an s-to-t path π' in G that (i) passes through some vertex w on a separator path of μ, (ii) uses only vertices of $V^{(\mu)}$ (except maybe for w itself) and (iii) has length at most $dist_G[s,t] + O(\varepsilon\Delta)$.*

Proof. We assume that none of the vertices on π lie on a separator path of μ because otherwise we can just set $\pi' = \pi$. Let ν be the child of μ with $s \in V^{(\nu)}$, let u be the last vertex on π that lies in $V^{(\nu)}$ (note that $u \neq t$, by definition of μ), and let v be the next vertex after u thereon. By Property (I) of G, there is a path $\pi_{u,v}$ from u to v in H of length at most $c \cdot$ (the weight of uv), which is at most $c\varepsilon\Delta$ by Property (III). Let w be the first vertex on $\pi_{u,v}$ that lies outside of $V^{(\nu)}$, which exists since v is outside of $V^{(\nu)}$. Then, from the fourth property of $\mathcal{T}$, we know that w must be on an (internal or external) separator path σ of μ. Thus, we set π' to be the path that goes from s to u along π, then from u to w along $\pi_{u,v}$ (which uses only vertices in $V^{(\nu)}$ as intermediates), then back from w to u along $\pi_{u,v}$, and finally from u to t along π. See Figure 2(a) (where σ is internal) and 2(b) (where σ is external). □

Next, we note how to further re-route π to pass through a portal.

Claim 2. (Detour through a portal) *There exists an s-to-t path π'' in G that (i) passes through a portal p on a separator path σ' of μ', where μ' is some ancestor of μ, (ii) uses only vertices of $V^{(\mu')}$, and (iii) has length at most $dist_G[s,t] + O(\varepsilon\Delta)$.*

Proof. Let w be as in Claim 1, and let μ' be the lowest ancestor of μ with $w \in V^{(\mu')}$. Notice that if $w \in V^{(\mu)}$, then $\sigma = \sigma'$. Otherwise, since μ' is the lowest ancestor, we have $w \notin \bigcup_{\text{child } \nu' \text{ of } \mu'} V^{(\nu')}$. Therefore, by the third property of $\mathcal{T}$, w must be on an internal

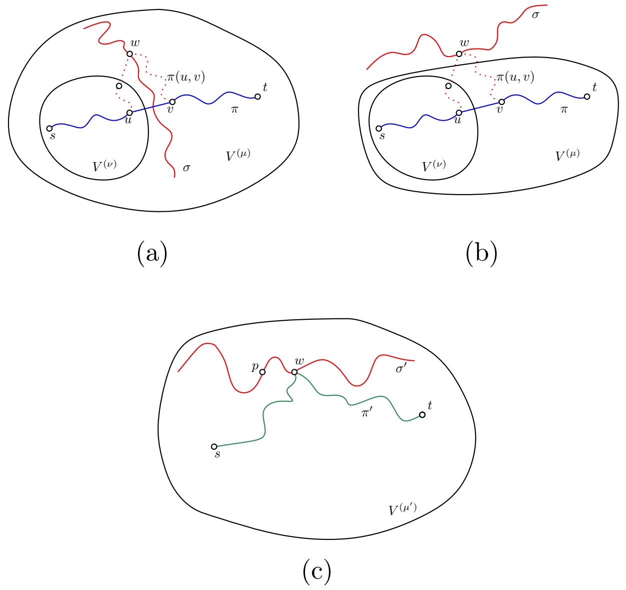

Figure 2: Detour through a vertex of a separator path σ in Claim 1, where σ may be internal, as in (a), or external, as in (b). Detour through a portal in Claim 2 in (c).

separator path σ' in μ'. Let p be the portal on σ' that is closest to w. Thus, the p-to-w distance on σ' is at most $O(\varepsilon\Delta)$. We set π'' to be the path the goes from s to w along π', then from w to p along σ' and back from p to w, and, finally, from w to u along π'. See Figure 2(c). $\square$

Let μ' be as in Claim 2. It follows that $\widetilde{\delta}[s,t] \leq \widetilde{\delta}_{\mu'}[s,t] \leq \widetilde{d}_{\mu'}[s,p] + \widetilde{d}_{\mu'}[p,t] \leq dist_G[s,t] + O(\varepsilon\Delta)$.

Theorem 1. (General distance oracles of additive stretch) *Given a weighted graph G of n vertices and of diameter Δ that satisfies Properties (I)–(III), together with the subgraph H from Property (I), we can construct a distance oracle of $O(\varepsilon\Delta)$ additive stretch, $O((1/\varepsilon)T(n)\log n)$ preprocessing time, $O((1/\varepsilon)n\log n)$ space, and $O((1/\varepsilon)\log n)$ query time.*

As we explained earlier, weighted unit-disk graphs satisfy Properties (I)-(III), so we have the following theorem.

Corollary 1. (Distance oracles of additive stretch in unit-disk graphs) *Given a set S of n planar points, such that the weighted unit-disk graph of S has diameter $\Delta \geq 1/\varepsilon$, we can construct for that graph a distance oracle of $O(\varepsilon\Delta)$ additive stretch, $O\left((1/\varepsilon)^3 n\log^2 n\right)$ preprocessing time, $O((1/\varepsilon)n\log n)$ space, and $O((1/\varepsilon)\log n)$ query time.*

2.3 Applications

We now describe how to employ Corollary 1 to compute a $(1+\varepsilon)$-approximation of the diameter of a unit-disk graph and how to build a $(1+\varepsilon)$-approximate distance oracle for it.

Let S be a set of planar points, let G be the weighted unit-disk graph defined by S, and let H be a planar $O(1)$-spanner of G.

Approximate diameter. The WSPD-based technique of Gao and Zhang's [17] implies the following lemma.

Lemma 3. (A consequence of WSPDs in unit-disk graphs) *Given a set S of n planar points, we can find a set of $O\left((1/\varepsilon)^4 n \log n\right)$ pairs of them in $O\left((1/\varepsilon)^4 n \log n\right)$ time, such that the shortest-path distance between any two vertices in the weighted unit-disk graph of S can be $(1+\varepsilon)$-approximated by the shortest-path distance between one of these pairs, which can be found in $O(1)$ time.*

First, we compute in $O(n \log n)$ time [30] the Euclidean diameter Δ_0 of S (i.e., the Euclidean distance of the farthest pair). If $\Delta_0 \geq 1/\varepsilon$, then $\Delta \geq 1/\varepsilon$, and, to compute a $(1+\varepsilon)$-approximation of Δ, we can query the oracle of Corollary 1 of $O(\varepsilon\Delta)$ additive stretch with all $O\left((1/\varepsilon)^4 n \log n\right)$ pairs of Lemma 3 and return the maximum. Thus the approximation factor is $1 + O(\varepsilon)$. The total time required for this case is $O\left(((1/\varepsilon)^4 n \log n \cdot (1/\varepsilon) \log n\right)$.

If $1 < \Delta_0 < 1/\varepsilon$, the problem is more straightforward, because we can construct the sparsified graph $\widehat{G}$ from the proof of Lemma 2, which preserves distances approximately, and then run a standard all-pairs shortest paths (APSP) algorithm therein. Since $\widehat{G}$ has $\widehat{n} = O\left((\Delta_0/\varepsilon)^2\right) = O\left((1/\varepsilon)^4\right)$ vertices and $\widehat{m} = O\left((1/\varepsilon)^2 \widehat{n}\right) = O\left((1/\varepsilon)^6\right)$ edges, we need $O\left(\widehat{n}^2 \log \widehat{n} + \widehat{m}\widehat{n}\right) = O\left((1/\varepsilon)^{10}\right)$ time for this case. Finally, if $\Delta_0 < 1$, the unit-disk graph is a complete Euclidean graph, so we just return Δ_0.

Theorem 2. (Approximate diameter) *Given a set S of n planar points, we can compute in $O\left((1/\varepsilon)^5 n \log^2 n + (1/\varepsilon)^{10}\right)$ time a $(1+\varepsilon)$-approximation of the diameter of its weighted unit-disk graph.*

Remark: There might be an alternative solution that does not use Gao and Zhang's WSPD construction, but instead combines our techniques with those of Weimann and Yuster for planar graphs [34]. However, such an alternative method seems inferior, since it would increase the ε dependency to $2^{O(1/\varepsilon)}$.

Approximate distance oracles. To build $(1+\varepsilon)$-approximate distance oracles in weighted unit-disk graphs, we employ the oracle of Corollary 1 of $O(\varepsilon\Delta)$ additive stretch as a building block using a known technique, called *sparse neighborhood covers*, similarly to the results of Kawarabayashi et al. [24] and Gu and Xu [20] for the planar graphs case. The following lemma is by Busch et al. [3] and Kawarabayashi et al. [24].

Lemma 4. (Sparse neighborhood cover) *Given a weighted planar graph H of n vertices and a value r, for each r we can construct, in $O(n \log n)$ time, a collection of subsets $V_1, V_2, \ldots$ of V, such that (i) the diameter of the subgraph of H induced by each V_i is $O(r)$, (ii) every vertex resides in $O(1)$ subsets, and (iii) for every vertex v, the set of all vertices at distance at most r from v in H is contained in at least one of the V_i's.*

Every (weighted) shortest-path distance in G is upper bounded by n, so we first apply the above lemma to H for each value of $r \in \{2^0, 2^1, \ldots, 2^{\log n}\}$. Thus, we obtain collections of subsets $V_1^{(r)}, V_2^{(r)}, \ldots$ and then build the distance oracle of Corollary 1 for the weighted unit-disk graph of each $V_i^{(r)}$. The total preprocessing time and space over all $O(\log n)$ choices of r is $O\left(\log n \cdot (1/\varepsilon)^3 n \log^2 n\right)$ and $O(\log n \cdot (1/\varepsilon) n \log n)$ respectively. Given $s, t \in S$, we consider each r and each subset $V_i^{(r)}$ that contains both s and t, query the oracle for $V_i^{(r)}$, and return the minimum. The total query time over all $O(\log n)$ choices of r and $O(1)$ choices of $V_i^{(r)}$ (Lemma 4(ii)) is $O(\log n \cdot (1/\varepsilon) \log n)$.

If $dist_G[s,t] \geq 1/\varepsilon$, let $r \geq c/\varepsilon$ be such that $dist_G[s,t] \in (r/2c, r/c]$. Then, each vertex on the shortest path from s to t in G is at distance at most $c\,dist_G[s,t] \leq r$ from s in H, so it is contained in a common subset $V_i^{(r)}$. Hence, we approximate $dist_G[s,t]$ with an additive error of $O(\varepsilon r) = O(\varepsilon\, dist_G[s,t])$, obtaining thus $1 + O(\varepsilon)$ approximation factor.

If $1 < dist_G[s,t] < 1/\varepsilon$, we simply build the sparsified graph $\widehat{G}$ from the proof of Lemma 2, which preserves distances approximately. Then, from every vertex, we precompute the distances to all grid cells at Euclidean distance at most $1/\varepsilon$ by running Dijkstra's algorithm on a subgraph of $\widehat{G}$ with $n' = O\left((1/\varepsilon)^4\right)$ vertices and $O\left((1/\varepsilon)^2 n'\right) = O\left((1/\varepsilon)^6\right)$ edges in $O\left((1/\varepsilon)^6 \log(1/\varepsilon)\right)$ time. The preprocessing time and space over all sources is $O\left((1/\varepsilon)^6 n \log(1/\varepsilon)\right)$ and $O\left((1/\varepsilon)^4 n\right)$ respectively. Finally, if $dist_G[s,t] \leq 1$, the shortest-path distance of s and t is their Euclidean distance. We do not know a priori which of the cases we are in, so we try all of them and return the minimum distance found.

Theorem 3. (Approximate distance oracles) *Given a set S of n planar points, we can build a $(1+\varepsilon)$-approximate distance oracle for its weighted unit-disk graph with $O\left((1/\varepsilon)\log^2 n\right)$ query time, $O\left((1/\varepsilon)^3 n \log^3 n + (1/\varepsilon)^6 n \log(1/\varepsilon)\right)$ preprocessing time, and $O\left((1/\varepsilon) n \log^2 n + (1/\varepsilon)^4 n\right)$ space.*

To reduce the query time, we can combine the above method with Gao and Zhang's WSPD-based oracle [17, Section 5.1], which requires $O(1)$ query time and $O((1/\varepsilon) n \log n)$ space. The bottleneck in the construction time lies in finding $(1+\varepsilon)$-approximate shortest-path distances for $O\left((1/\varepsilon)^4 n \log n\right)$ pairs, but we can compute these distances by querying our oracle of Theorem 3 in $O\left((1/\varepsilon)^4 n \log n \cdot (1/\varepsilon)\log^2 n\right)$ total time.

Corollary 2. (Approximate distance oracles with $O(1)$ query time) *Given a set S of n planar points, we can construct a $(1+\varepsilon)$-approximate distance oracle for its weighted unit-disk graph with $O(1)$ query time, $O\left((1/\varepsilon)^5 n \log^3 n + (1/\varepsilon)^6 n \log(1/\varepsilon)\right)$ preprocessing time, and $O\left((1/\varepsilon)^4 n \log n\right)$ space.*

Similarly, we can use the distance oracle of Theorem 3 to improve Gao and Zhang's results for other distance-related problems on weighted unit-disk graphs (see [17, Section 5]):

Corollary 3. (Approximate radius and bichromatic closest pair) *Let S be a set n points in the plane, and let G be the corresponding weighted unit-disk graph. We can compute a $(1+\varepsilon)$-approximation of the radius of G or of the bichromatic closest pair distance of two given sets $A, B \subseteq S$ in G in $O\left((1/\varepsilon)^5 n \log^3 n + (1/\varepsilon)^6 n \log(1/\varepsilon)\right)$ time.*

Remarks:

- We did not seriously attempt to optimize the $\text{poly}(1/\varepsilon, \log n)$ factors, but some small improvements could be possible with more effort.
- Our distance oracle in Theorem 3 can be easily modified to report an approximate shortest path, not just its distance, in additional time proportional to the number of edges in the path. To do so, every time we find approximate shortest distances in a subgraph from a portal, we also store its approximate shortest-path tree.
- The same approach gives $(1 + O(\varepsilon))$-approximation results for *unweighted* unit-disk graphs, assuming that the diameter and the distances of the query vertices exceed $\Omega(1/\varepsilon)$. Specifically a WSPD-based technique can be employed for the unweighted case as well, but the error now has an extra additive term of $4+O(\varepsilon)$ [17, Lemma 6.2], which can be ignored under our assumption. Also, we need to replace the SSSP algorithm of Lemma 2 with the $O(n \log n)$-time exact SSSP algorithm by Cabello and Jejčič [6] or by Chan and Skrepetos [9].

3 Approximate apBLSP

In this section, we study the $(1+\varepsilon)$-approximate apBSLP problem. Given a set S of n planar points, let G be its complete weighted Euclidean graph, let $w_1, w_2, \ldots, w_N$, where $N = \binom{n}{2}$, be the weights of the edges of G in non-decreasing order, and let G^i be the subgraph of G that contains only the edges of weight at most w_i (note that $G^N = G$). We can assume that $w_1 \geq 1$. Otherwise, we can impose that assumption by simply rescaling S in linear time. We want to preprocess S into a data structure, such that we can quickly answer $(1+\varepsilon)$-approximate *bounded-leg distance* queries, i.e., given $s, t \in S$ and a positive number L, compute a $(1+\varepsilon)$-approximation of $dist_{G^i}[s,t]$, where i is the largest integer with $w_i \leq L$. First, we briefly review the previous methods of Bose et al. [2] and of Roditty and Segal [31], in Section 3.1, and then describe our own approach, in Section 3.2.

3.1 Previous methods

Let $s, t \in S$, and let $c(s,t)$ be the minimum index, such that s and t are connected in $G^{c(s,t)}$. Since each G^i is a subgraph of G^{i+1}, we have $dist_G[s,t] \leq dist_{G^{N-1}}[s,t] \leq \cdots \leq dist_{G^{c(s,t)}}[s,t]$. Moreover, the s-to-t shortest path in any G^i with $i > c(s,t)$ must have an edge of weight at least $w_{c(s,t)}$, so $dist_G[s,t] \geq w_{c(s,t)}$. Any shortest path has at most $n-1$ edges, thus $dist_{G^{c(s,t)}}[s,t] \leq (n-1)\, dist_G[s,t]$. Therefore, as Roditty and Segal [31, Section 2] noticed, we can compute and store a $(1+\varepsilon)$-approximation of the s-to-t shortest-path distance for each $s, t \in S$ in only $O\left(\log_{1+\varepsilon} n\right)$ graphs, such that a bounded-leg distance query can be answered with a binary search in $O\left(\log \log_{1+\varepsilon} n\right)$ time.

Specifically, for every $s, t \in S$ and $j \in \{0, 1, \ldots, \lceil \log_{1+\varepsilon} n \rceil\}$, let $I^j(s,t)$ be the set of indices of the graphs G^i, such that $(1+\varepsilon)^j\, dist_G[s,t] \leq dist_{G^i}[s,t] \leq (1+\varepsilon)^{j+1} dist_G[s,t]$. If $I^j(s,t) \neq \emptyset$, we create two values $m^j(s,t)$ and $\ell^j(s,t)$, where the former is any index therein and the latter is equal to $w_{m^j(s,t)}$. Else, $m^j(s,t)$ and $\ell^j(s,t)$ are undefined.

The total space required over all pairs of S is $O\left(n^2 \log_{1+\varepsilon} n\right)$. Then, given a positive number L, we can find the largest i among the $m^j(s,t)$'s such that $w_i \le L$ with a binary search over the $\ell^j(s,t)$'s in $O\left(\log\log_{1+\varepsilon} n\right)$ time and return a $(1+\varepsilon)$-approximation of the s-to-t shortest-path distance in G^i.

To compute a possible index for $m^j(s,t)$ for each $s,t \in S$ and $j \in \{0,1,\ldots,\lceil\log_{1+\varepsilon} n\rceil\}$, Roditty and Segal performed $O\left(n^2 \log_{1+\varepsilon} n\right)$ independent binary searches, each making $O(\log n)$ $(1+\varepsilon)$-approximate bounded-leg distance queries (i.e., a query to find a $(1+\varepsilon)$-approximation of the s-to-t shortest-path distance in some graph G^i). Instead, we group the queries for all s,t,j into $O\left(\log n \cdot \log_{1+\varepsilon} n\right)$ rounds of n^2 offline queries each, where "offline" means that the queries in every round are given in advance.

Lemma 5. (Framework for approximate apBLSP) *Given a set S of n planar points, we can construct a data structure for the $(1+\varepsilon)$-approximate apBLSP problem of $O\left((1/\varepsilon)n^2\log n\right)$ space, $O(\log\log n + \log(1/\varepsilon))$ query time, $O\left(T_{\text{off}}(n,n^2,1+\varepsilon)\cdot(1/\varepsilon)\log^2 n\right)$ preprocessing time, where $T_{\text{off}}(n',q',1+\varepsilon')$ denotes the total time for answering q' offline $(1+\varepsilon')$-approximate bounded-leg distance queries for an n'-point set.*

Naively we could construct in near linear time a sparse $(1+\varepsilon)$-spanner of every graph G^i and then run Dijkstra's algorithm therein to answer each query (a similar idea was used by Roditty and Segal). Thus a near-cubic bound would be obtained for $T_{\text{off}}(n,n^2,1+\varepsilon)$. Instead, we show that by employing our $(1+\varepsilon)$-approximate distance oracle of Corollary 2 for weighted unit-disk graphs as a subroutine, we can obtain a truly subcubic bound on $T_{\text{off}}(n,n^2,1+\varepsilon)$, as we next describe.

3.2 Improved method

To obtain our improved method, we view the problem of answering n^2 approximate offline bounded-leg distance queries for each $s,t \in S$ and $j \in \{0,1,\ldots,\lceil\log_{1+\varepsilon} n\rceil\}$ as the problem of constructing and querying the following offline *semi-dynamic* (actually insertion-only) distance oracle.

Subproblem 1. (Semi-dynamic approximate distance oracles) *Given an arbitrary graph of n vertices with edge weights in $[1,\infty)$, we want to perform an offline sequence of q operations, each of which is either an edge insertion, or a query to compute a $(1+\varepsilon)$-approximation of the shortest-path distance between two vertices.*

Let $T_{\text{dyn}}(n,q,1+\varepsilon)$ be the complexity of Subproblem 1.

We could reduce our problem to Subproblem 1 by naively inserting the $O\left(n^2\right)$ edges of G in increasing order of weight to an initially empty graph and mix that sequence of insertions with the given sequence of bounded-leg distance queries. Hence we would have $T_{\text{off}}(n,n^2,1+\varepsilon) = O\left(T_{\text{dyn}}(n,n^2,1+\varepsilon)\right)$.

We propose a better reduction that employs a simple periodic rebuilding trick. First, we divide the sequence of the q edge insertions and queries into $O(q/r)$ *phases* of at most r operations each, where r is a parameter to be set later. At the beginning of each phase, the current graph is a weighted unit-disk graph (after rescaling), so we can build the $(1+\varepsilon)$-approximate distance oracle of Corollary 2 in $O\left((1/\varepsilon)^5 n \log^3 n\right)$ time. Then we query that oracle in $O\left(r^2\right)$ total time to approximate the shortest-path distances between all pairs of vertices that are involved in the upcoming r operations (i.e., are endpoints of the edges to be inserted, or belong to the pairs to be queried). We build the complete graph over these at most $2r$ vertices with the approximate shortest-path distances as edge weights. Each phase can then be handled by r edge insertions/queries on this smaller graph in $O\left(T_{\text{dyn}}(2r, r, 1+\varepsilon)\right)$ time. The resulting approximation factor is at most $(1+\varepsilon)^2 = 1+\Theta(\varepsilon)$. Thus, for $q = n^2$ we get the following bound:

$$T_{\text{off}}(n, n^2, 1+\Theta(\varepsilon)) \;=\; O\left(\frac{n^2}{r}\left((1/\varepsilon)^5 n \log^3 n + r^2 + T_{\text{dyn}}(2r, r, 1+\varepsilon)\right)\right). \tag{1}$$

To solve Subproblem 1, we could do nothing during insertions and in each query re-run Dijkstra's algorithm from scratch, thus obtaining $T_{\text{dyn}}(2r, r, 1+\varepsilon) = O(r^3)$. Then by setting $r = (1/\varepsilon)^{5/3} n^{1/3} \log n$, we can obtain a still better bound $T_{\text{off}}(n, n^2, 1+\Theta(\varepsilon)) = O\left((1/\varepsilon)^{10/3} n^{8/3} \log^2 n\right)$, which is truly subcubic.

Actually, by using fast matrix multiplication and additional techniques, we can establish a better bound on $T_{\text{off}}(n, n^2, 1+\Theta(\varepsilon))$. Our idea is to recursively divide phases into subphases, as in the proof of the following lemma. (Note that this lemma actually holds for general graphs. Although (semi-)dynamic shortest paths have been extensively studied in the literature, we are unable to find this particular result.)

Lemma 6. (A semi-dynamic approximate distance oracle) *We can solve Subproblem 1 in* $T_{\text{dyn}}(2r, r, 1+\Theta(\varepsilon)) = O\left((1/\varepsilon) r^\omega \log r \log \overline{W}\right)$ *total time, where* ω *is the matrix-multiplication exponent and* $\overline{W}$ *is an upper bound on the maximum (finite) shortest-path distances.*

Proof. Let H be the input graph of $2r$ vertices, and let H' be the graph that results from performing on H all edge insertions of the first $r/2$ operations. We run the $O\left((1/\varepsilon) r^\omega \log \overline{W}\right)$-time $(1+\varepsilon)$-approximate APSP algorithm of Zwick [36] on H and H' and answer all distance queries therein. Then, we construct two graphs H_1 and H_2 of r vertices each, where H_1 (respectively H_2) is the complete graph over all vertices that are involved in the first (respectively last) $r/2$ operations. We set each edge weight in H_1 (respectively H_2) to be a $(1+\varepsilon)$-approximation of the shortest-path distance of its endpoints in H (respectively H'), which we have already computed. Thus the error is increased by a $1+\varepsilon$ factor. Finally, we apply recursion in H_1 and H_2.

The running time of our approach is $T_{\text{dyn}}(2r, r, (1+\varepsilon)^i) \le 2T_{\text{dyn}}(r, r/2, (1+\varepsilon)^{i+1}) + O\left((1/\varepsilon) r^\omega \log \overline{W}\right)$, where initially $i = 1$. Thus, $T_{\text{dyn}}(2r, r, (1+\varepsilon)) = O\left((1/\varepsilon) r^\omega \log r \log \overline{W}\right)$. The approximation factor is $(1+\varepsilon)^{\log r} = 1 + \Theta(\varepsilon \log r)$, which can be refined to $1 + \Theta(\varepsilon)$, by resetting $\varepsilon \leftarrow \varepsilon / \log r$. □

Combining (1) with the above lemma gives

$$T_{\text{off}}(n, n^2, 1+\Theta(\varepsilon)) \;=\; O\left(\frac{n^2}{r}\left((1/\varepsilon)^5 n \log^3 n + (1/\varepsilon) r^\omega \log r \log \overline{W}\right)\right).$$

Setting $r = n^{1/\omega}$ yields $T_{\text{off}}(n, n^2, 1+\Theta(\varepsilon)) = O\left((1/\varepsilon)^5 n^{3-1/\omega} \log^3(n\overline{W})\right)$, where $\overline{W} \leq nW$, and W is the spread of S, i.e., the ratio of the maximum-to-minimum pairwise Euclidean distance.

Theorem 4. (Approximate all-pairs bounded leg shortest paths) *Given a set S of n planar points of spread W, we can construct a data structure for the $(1+\varepsilon)$-approximate apBLSP problem with $O(\log\log n + \log(1/\varepsilon))$ query and $O\left((1/\varepsilon)^6 n^{3-1/\omega} \log^5(nW)\right)$ preprocessing time and $O\left((1/\varepsilon)n^2 \log n\right)$ space.*

The current best bound on the matrix-multiplication exponent [33, 27] is $\omega < 2.373$, which gives a preprocessing time of $O\left((1/\varepsilon)^6 n^{2.579} \log^5(nW)\right)$.

Remark: For the sake of simplicity we did not optimize the poly$(1/\varepsilon, \log(nW))$ factors. It might be possible that known techniques like balanced quadtrees could be used to eliminate the dependency on W.

References

[1] Srinivasa Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel H. M. Smid, and Christos D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In *Proceedings of the 4th European Symposium on Algorithms (ESA)*, pages 514–528, 1996.

[2] Prosenjit Bose, Anil Maheshwari, Giri Narasimhan, Michiel Smid, and Norbert Zeh. Approximating geometric bottleneck shortest paths. *Computational Geometry*, 29(3):233–249, 2004.

[3] Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. Improved sparse covers for graphs excluding a fixed minor. In *Proceedings of the 26th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 61–70, 2007.

[4] Sergio Cabello. Many distances in planar graphs. *Algorithmica*, 62(1-2):361–381, 2012.

[5] Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2143–2152, 2017.

[6] Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry*, 48(4):360–367, 2015.

[7] Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

[8] Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *Journal of the ACM*, 57(3):16:1–16:15, 2010.

[9] Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC)*, pages 24:1–24:13, 2016.

[10] Timothy M. Chan and Dimitrios Skrepetos. Faster approximate diameter and distance oracles in planar graphs. In *Proceedings of the 25th European Symposium on Algorithms (ESA)*, pages 25:1–25:13, 2017.

[11] Timothy M. Chan and Dimitrios Skrepetos. Approximate shortest paths and distance oracles in weighted unit-disk graphs. In *Proceedings of the 34th ACM Symposium on Computational Geometry (SoCG)*, 2018.

[12] Vincent Cohen-Addad, Søren Dahlgaard, and Christian Wulff-Nilsen. Fast and compact exact distance oracle for planar graphs. In *Proceedings of the 58th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 962–973, 2017.

[13] David P. Dobkin, Steven J. Friedman, and Kenneth J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete & Computational Geometry*, 5(4):399–407, 1990.

[14] Ran Duan and Seth Pettie. Bounded-leg distance and reachability oracles. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 436–445, 2008.

[15] Ran Duan and Hanlin Ren. Approximating all-pair bounded-leg shortest path and apsp-af in truly-subcubic time. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2018.

[16] David Eppstein, Gary L. Miller, and Shang-Hua Teng. A deterministic linear time algorithm for geometric separators and its applications. *Fundamenta Informaticae*, 22(4):309–329, 1995.

[17] Jie Gao and Li Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM Journal on Computing*, 35(1):151–169, 2005.

[18] Pawel Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\widetilde{O}\left(n^{5/3}\right)$ time. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 495–514, 2018.

[19] Pawel Gawrychowski, Shay Mozes, Oren Weimann, and Christian Wulff-Nilsen. Better tradeoffs for exact distance oracles in planar graphs. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 515–529, 2018.

[20] Qian-Ping Gu and Gengchun Xu. Constant query time $(1+\varepsilon)$-approximate distance oracle for planar graphs. In *Proceedings of the 26th International Symposium on Algorithms and Computation (ISAAC)*, pages 625–636, 2015.

[21] Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Spanners and reachability oracles for directed transmission graphs. In *Proceedings of the 31st International Symposium on Computational Geometry (SoCG)*, volume 34, 2015.

[22] Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs. *Algorithmica*, 80(3):830–848, 2018.

[23] Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2495–2504, 2017.

[24] Ken-ichi Kawarabayashi, Christian Sommer, and Mikkel Thorup. More compact oracles for approximate distances in undirected planar graphs. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 550–563, 2013.

[25] J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete & Computational Geometry*, 7(1):13–28, 1992.

[26] Philip Klein. Preprocessing an undirected planar network to enable fast approximate distance queries. In *Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 820–827, 2002.

[27] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014.

[28] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *Proceedings of the 21st Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1268–1277, 2002.

[29] Gary L. Miller, S.-H. Teng, and Stephen A. Vavasis. A unified geometric approach to graph separators. In *Proceedins of the 32nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 538–547, 1991.

[30] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[31] Liam Roditty and Michael Segal. On bounded leg shortest paths problems. *Algorithmica*, 59(4):583–600, 2011.

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

[33] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith–Winograd. In *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, pages 887–898, 2012.

[34] Oren Weimann and Raphael Yuster. Approximating the diameter of planar graphs in near linear time. *ACM Transactions on Algorithms*, 12(1):1–13, 2016.

[35] Chenyu Yan, Yang Xiang, and Feodor F. Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *Computational Geometry*, 45(7):305–325, 2012.

[36] Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM*, 49(3):289–317, 2002.