A WSPD, Separator and Small Tree Cover for c-Packed Graphs

Lindsey Deryckere
The University of Sydney, Australia

Joachim Gudmundsson
The University of Sydney, Australia

André van Renssen
The University of Sydney, Australia

Yuan Sha
The University of Sydney, Australia

Sampson Wong
The University of Copenhagen, Denmark

Abstract

The c-packedness property, proposed in 2010, is a geometric property that captures the spatial distribution of a set of edges. Despite the recent interest in c-packedness, its utility has so far been limited to Fréchet distance problems. An open problem is whether a wider variety of algorithmic and data structure problems can be solved efficiently under the c-packedness assumption, and more specifically, on c-packed graphs.

In this paper, we prove two fundamental properties of c-packed graphs: that there exists a linear-size well-separated pair decomposition under the graph metric, and there exists a constant size balanced separator. We then apply these fundamental properties to obtain a small tree cover for the metric space and distance oracles under the shortest path metric. In particular, we obtain a tree cover of constant size, an exact distance oracle of near-linear size and an approximate distance oracle of linear size.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Well-separated pair decomposition, separator, tree cover, distance oracles, realistic graphs

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.21

Related Version *Full Version*: `https://arxiv.org/abs/2505.06884`

Funding This research was partially funded by the Australian Government through the Australian Research Council (project number DP240101353).
Joachim Gudmundsson: This research was partially funded by the Australian Government through the Australian Research Council (project number DP240101353).
André van Renssen: This research was partially funded by the Australian Government through the Australian Research Council (project number DP240101353).
Sampson Wong: This research was partially funded by the European Union through the Marie Skłodowska-Curie Actions Postdoctoral Fellowship (project number 101146276).

1 Introduction

The study of graphs and their properties is a cornerstone of theoretical computer science. A wide variety of graph properties have been proposed in the literature, such as planarity [34], treewidth [4] and doubling dimension [23]. By assuming these graph properties, one can often obtain better algorithmic or data structure solutions to graph theoretic problems.

The c-packedness property [14], proposed in 2010, is a geometric property that captures the spatial distribution of the edges in a graph. A graph is c-packed if, for any positive real r and any ball of radius r, the length of the edges contained in the ball is at most $c \cdot r$. Driemel, Har-Peled and Wenk [14] introduced the c-packedness property for polygonal curves, and showed that one can compute the Fréchet distance between a pair of c-packed curves in near-linear time. In 2013, Gudmundsson and Smid [21] adapted the c-packedness definition to graphs, and proposed a Fréchet distance data structure on c-packed trees with long edges that decides if there is a path in the tree with small Fréchet distance to a query curve. In 2023, Gudmundsson, Seybold and Wong [20] generalised the result of [21] by proposing a Fréchet distance data structure for all c-packed graphs.

Despite the recent interest in c-packedness, its utility so far has been limited to Fréchet distance problems. Moreover, the fundamental properties of c-packed graphs are not well studied, thereby limiting the number of problems that can be solved efficiently on c-packed graphs. An open problem is whether c-packed graphs have applications beyond Fréchet distance problems.

1.1 Our Results

We show that c-packed graphs lie in the intersection of two important graph classes, that is, doubling metrics and bounded treewidth graphs. Using the properties of doubling metrics and bounded treewidth graphs, one can obtain, for c-packed graphs, a linear-size well-separated pair decomposition, a linear-size exact distance oracle, a linear-size approximate distance oracle, and a constant-size tree cover. However, these constructions have processing times that are either (i) randomised, (ii) depend on the spread of the c-packed metric, or (iii) are exponential in the treewidth. See Table 1, Column 3[1].

We provide the first deterministic constructions that are independent of the spread of the c-packed metric, for the aforementioned structures. Moreover, our preprocessing times and data structure sizes are polynomial in both c and ε, whereas previous constructions are not.

We summarise the main results of our paper. First, we show that any c-packed graph in $\mathbb{R}^d$ admits a well-separated pair decomposition (WSPD) of size $O((c^3/\varepsilon) \cdot n)$, where $1/\varepsilon$ is the separation constant of the WSPD. Note that we avoid the $(1/\varepsilon)^d$ factor that appears in the sizes of many other WSPD constructions [7, 25]. Then, we show that any c-packed graph in $\mathbb{R}^d$ admits an $O(c)$-size separator. We use this separator to show that c-packed graphs have $O(c)$ treewidth, and admits an exact distance oracle (EDO) of size $O(cn \log n)$. Finally, we combine our WSPD and the EDO to construct a $(1 + \varepsilon)$ distortion tree cover with $O(c^{2d+2}/\varepsilon^d)$ trees. Our tree cover implies an approximate distance oracle (ADO) of size $O((c^{2d+2}/\varepsilon^d) \cdot n)$ and $O(c^{2d+2}/\varepsilon^d)$ query time. We summarise our results in Table 1.

1.2 Related work

The c-packedness property is a popular model for realistic curves. A wide range of Fréchet distance problems have been studied on c-packed curves, including map matching [11], the mean curve [27], the shortcut Fréchet distance [13], subtrajectory clustering [5, 18] and the approximate nearest neighbour data structure [12]. However, c-packed graphs are less well understood [20, 21], despite often being used as a vital stepping stone towards a related property called λ-low density, which was also introduced by Driemel and Har-Peled [13].

[1] A reviewer pointed us to the tree-like properties of the graph and the possibility of adapting the work by Chazelle [10]. However, such adaptations would be likely to incur an exponential dependency on c.

Table 1 In the table, c = the c-packedness value, ε = either an ε^{-1} separation constant or a $(1+\varepsilon)$ approximation ratio, Rand. = randomised algorithm, Δ = spread of the doubling metric, d = dimension of the Euclidean space, *dd* = previous results using doubling dimension, *tw* = previous results using treewidth.

		Previous			New		
		Preprocessing	Size	Source	Preprocessing	Size	Source
WSPD	*dd*	$(1/\varepsilon)^{O(\log c)} \cdot n \log(\Delta n)$	$(1/\varepsilon)^{O(\log c)} \cdot n$	[38]	$(c^3/\varepsilon) \cdot n \log n$	$(c^3/\varepsilon) \cdot n$	Thm 3
		Rand. $(1/\varepsilon)^{O(\log c)} \cdot n \log n$		[25]			
EDO	*tw*	$2^{O(c^3)} n$	$c^2 n$	[9]+ [15]	$c^2 n \log^2 n$	$cn \log n$	Thm 18
Tree cover	*dd*	$(1/\varepsilon)^{O(\log c)} \cdot n \log^2(\Delta n)$	$(1/\varepsilon)^{O(\log c)}$	[3] + [38]	$(c^{2d+6}/\varepsilon^{d+1}) \cdot n \log n$	c^{2d+2}/ε^d	Thm 5
ADO	*dd*	$(1/\varepsilon)^{O(\log c)} \cdot n \log^2(\Delta n)$	$(1/\varepsilon)^{O(\log c)} \cdot n$	[3] + [38]	$(c^{2d+6}/\varepsilon^{d+1}) \cdot n \log n$	$(c^{2d+2}/\varepsilon^d) \cdot n$	Thm 6

Low density graphs have been studied in map matching [6, 11]. The well-separated pair decomposition of [22] has size polynomial in λ and ε, but its disadvantage over those stated in Table 1 is that it has size $O(n \log n)$.

Well-Separated Pair Decompositions (WSPD) are used for compact representation of the quadratic distances between pairs of points in the metric. For metrics that allow for a sub-quadtratic size WSPD, they have therefore been used as fundamental tools to approximate solutions to a range of proximity problems that require looking at the distances between all pairs of points, such as nearest neighbour, diameter, stretch and minimum spanning tree. Not all metrics allow for a WSPD of subquadratic size, an example of which is the metric induced by a star tree with unit weight on all edges. For this metric, any WSPD requires a quadratic number of pairs. For a point set in $\mathbb{R}^d$, where d is considered a constant, Callahan and Kosaraju [7] showed that there exists a WSPD with separation factor σ of size $O(\sigma^d n)$ that can be computed in $O(n \log n + \sigma^d n)$ time. In contrast to this, we show that for c-packed graphs, the size of the WSPD is not exponential in d, while maintaining that the size is linear. For graphs with bounded doubling dimension (dim), Har-Peled and Mendel [25] designed a $O(2^{O(dim)} n \log n + n\varepsilon^{-O(dim)})$ expected time randomised algorithm to construct a WSPD of linear size with logarithmic query time. They also designed a deterministic construction which incurs a logarithmic dependency on the aspect ratio of the metric space. In the full version of this paper we show that a c-packed graph has doubling dimension $O(\log c)$. However, in contrast to previous results, our deterministic construction of the WSPD is, to the best of our knowledge, the first that does not depend on the aspect ratio of the metric space and does not incur any factors exponential in the dimension, assuming constant dimension.

Distance oracles are shortest path data structures for graphs. For planar graphs, Lipton and Tarjan [34, 35] proved the planar separator theorem, which states that any planar graph with n vertices has a balanced separator of size $\sqrt{n}$. Using the separator to build a separator hierarchy, they constructed an exact distance oracle of size $O(n\sqrt{n})$. Thorup [39] and Klein [30] independently presented $(1+\varepsilon)$-approximate distance oracles for planar graphs of size $O(n \log n)$, for any constant $\varepsilon > 0$. Subsequent works [28, 29, 33, 41] improved the preprocessing time, space, and query time. Dvořák and Norin [15] showed that if a graph admits a small size balanced separator, it also has small treewidth. Combined with our separator results this implies that c-packed graphs have treewidth $O(c)$. Chaudhuri and Zaroliagis [9] designed an exact distance oracle whose preprocessing time is single exponential in the treewidth of the graph. In contrast to these results, our algorithms do not incur any terms exponential in c.

Approximate distance oracles have been studied on non-planar graphs. Thorup and Zwick [40] constructed a $(2k-1)$-approximate distance oracle of $O(kn^{1+1/k})$ size on any general graph, and showed that the size bound is optimal under the Erdös girth conjecture. Gudmundsson, Levcopoulos, Narasimhan and Smid [19] provided a $(1+\varepsilon)$-approximate distance oracle of size $O(n \log n)$ on any t-spanner, assuming $\varepsilon, t > 0$ are constants. Gao and Zhang [16] constructed a $(1+\varepsilon)$-approximate distance oracle of size $O(n \log n)$ for any unit-disk graph, assuming $\varepsilon > 0$ is a constant.

Balanced separators of sublinear size have been found for planar graphs [34], graphs of bounded genus [17], graphs that exclude a fixed minor [1], and graphs that are the 1-skeletons of simplicial complexes in 3-dimensions [36]. They have been used as a fundamental tool in devising efficient algorithms for graphs [1, 17, 35] and in numerical analysis [32, 36]. Randomized balanced separators can be found in expected linear time for c-packed graphs [26].

Metric embeddings approximate harder metric spaces by simpler, well-structured metric spaces, such as tree metrics. A tree cover for a finite metric space is a small number of trees such that the distance between any two points in the metric is preserved with low distortion in at least one of the trees. Tree covers with $(1+\epsilon)$-distortion and a constant number of trees have been found for Euclidean metrics [2], doubling metrics [3] and planar metrics [8].

2 Preliminaries

Let $G(V, E)$ be a geometric graph in $\mathbb{R}^d$ consisting of the point set V and edge set E. In this paper, we consider graphs in a fixed d-dimensional space that satisfy c-packedness. We denote the graph distance between two nodes $u, v \in V$ as $dist_G(u, v)$, and their Euclidean distance as $dist(u, v)$. We first define a Well-Separated Pair Decomposition [7] for Euclidean point sets, followed by its counterpart for geometric graphs.

▶ **Definition 1** (Geometric Well-Separated Pair). *Let σ be a real positive number, and let A and B be two finite sets of points in $\mathbb{R}^d$. We say that A and B are* well-separated with respect to σ *if the distance between the bounding boxes C_A and C_B of A and B, respectively, is at least $\sigma \cdot \max(rad(C_A), rad(C_B))$.*

Here $rad(C_A)$ refers to the radius of C_A, which is half its diameter.

▶ **Definition 2** (Geometric Well-Separated Pair Decomposition (WSPD)). *Let S be a set of n points in $\mathbb{R}^d$, and let σ be a real positive number. A* well-separated pair decomposition *(WSPD) for S, with respect to σ, is a sequence $\{A_1, B_1\}, \dots \{A_m, B_m\}$ of pairs of nonempty subsets of S, for some integer m, such that (i) for each i with $1 \le i \le m$, A_i and B_i are well separated w.r.t. σ, and (ii) for any two distinct points p and q of S, there is exactly one index i with $1 \le i \le m$, such that $p \in A_i$ and $q \in B_i$, or vice versa.*

The notion of well-separated pairs and WSPD can easily be extended to graphs. For the graph version we will simply replace all the geometric distances with the distances in the graph. We will refer to a WSPD in a graph G as WSPD$_G$.

3 Technical Overview

In Section 3.1, we describe our deterministic WSPD$_G$ construction. In Section 3.2, we discuss our separator theorem. Finally, in Section 3.3 we show how to use the WSPD$_G$ and the exact distance oracle to construct a $(1+\varepsilon)$-distortion tree cover of size $O(c^{2d+2}(\frac{1}{\varepsilon})^d)$ for c-packed metrics. Due to space limitation, all omitted proofs are available in the full version of the paper on ArXiv.

3.1 A Well-Separated Pair Decomposition for c-packed Graphs

Split trees [7] or quadtrees [37] are commonly used in the construction of a geometric WSPD of a point set. An essential property of these is that the maximum Euclidean distance between two points contained in the same cell is always bounded by a function of the diameter of the cell. We construct a tree that fulfills a similar purpose to the trees above but for graph distances between points. We call this new type of tree a *δ-connected tree* (δ-CT). Each cell, corresponding to a cube s, of the δ-connected tree is a δ-connected set, meaning that points contained in the cell are within a graph distance of at most $\delta \cdot diam(s)$ from one another. To construct the δ-CT, we use a bottom up approach. The leaves of the compressed quadtree are already δ-connected sets. At higher levels of the compressed quadtree, we consider the δ-connected sets of its children, and merge together pairs of previously δ-connected sets that are also a δ-connected set in the higher level. To obtain an efficient running time, we make two observations. First, when computing the δ-connected sets for a higher level, it suffices to maintain a vertex representative for each δ-connected set of the lower level. Second, to check if a pair of sets are δ-connected, it suffices to check whether their representatives are path-connected in the cube centred at the cell but with double its radius.

To upper bound the graph diameter of the δ-connected set in each cell of the δ-CT we compute the length of intersection of edges with the cell and the 3^d surrounding cells in a canonical grid. To do this efficiently, we note that not all edges intersecting with a cell can contribute to a δ-connected component inside the cell. We therefore construct a data structure that can be queried for the total length of all edges that can contribute to a δ-connected component contained in a cell.

Combining these ideas obtains the following theorem. For details refer to Section 4.2.

▶ **Theorem 3.** *Given a c-packed graph G in $\mathbb{R}^d$, for fixed d, one can construct a WSPD$_G$ with separation factor σ of size $O(c^3\sigma n)$ in $O(cn\log n + c^3\sigma n)$ time, using $O(cn)$ space.*

3.2 A Separator Theorem for c-packed Graphs

We prove that every c-packed graph admits a balanced vertex separator of size $O(c)$. We start with the ring separator of Har-Peled and Mendel [25], which states that for a point set in $\mathbb{R}^d$, one can efficiently compute a pair of balls so that $n/2\lambda^3$ of the points are inside the inner ball, and $n/2\lambda^3$ of the points are outside the outer ball, where λ is the doubling constant of $\mathbb{R}^d$. Using the ring separator, we construct a max-flow instance in a similar fashion to Gudmundsson et al. [20] to locate a cut of size $O(c)$. This cut $(1 - 1/2\lambda^3)$-separates the graph, in that it separates the graph into two components each with at most $n \cdot (1 - 1/2\lambda^3)$ points. We obtain the following theorem, for details refer to Section 5.

▶ **Theorem 4.** *Given a c-packed graph G in $\mathbb{R}^d$, where d is fixed, with n vertices, one can find a separator of size $O(c)$ that $(1 - \frac{1}{2\lambda^3})$-separates G, in $O(c^2 n)$ time.*

3.3 A Small Tree Cover for c-packed Graphs

Our approach follows that of the celebrated "Dumbbell Theorem" [2] for Euclidean metrics. For c-packed graphs, we construct a linear number of dumbbells from the WSPD$_G$, which is constructed using the graph distances. The dumbbells are partitioned into groups, each of which satisfies the *length-group property* and the *empty-region property* with respect to the graph distance. The c-packedness property enables us to partition the dumbbells into a small number (depending only on the packedness value c and the separation ratio σ of WSPD$_G$) of groups, each of which satisfies the empty-region property. The main difficulty is proving the packing lemmas required for establishing the empty-region property. A dumbbell tree, which connects the dumbbells in a group hierarchically, is built for each group of dumbbells. The c-packedness property and the c-CT also enable us to do range searching and efficiently build the dumbbell trees. The dumbbell trees together constitute a tree cover for the c-packed metric. We obtain the following theorem. For details refer to the full version of the paper.

▶ **Theorem 5.** *Given a c-packed graph G in $\mathbb{R}^d$ of fixed d and any $0 < \varepsilon < 1$. In $O(n \log n(c^{2d+6}(\frac{1}{\varepsilon})^{d+1} + c \log n))$ time, one can construct $O(c^{2d+2}(\frac{1}{\varepsilon})^d)$ dumbbell trees, each of size $O(n)$, such that the dumbbell trees constitute a $(1+\varepsilon)$-distortion tree cover for the graph metric induced by G.*

The tree cover of Theorem 5 immediately implies a $(1+\varepsilon)$-approximate distance oracle for the c-packed metric.

▶ **Corollary 6.** *Given a c-packed graph G in $\mathbb{R}^d$ of fixed d, and any $0 < \varepsilon < 1$, one can preprocess it in $O(n \log n(c^{2d+6}(\frac{1}{\varepsilon})^{d+1} + c \log n))$ time, using $O(c^{2d+2}(\frac{1}{\varepsilon})^d n)$ space, to answer a $(1+\varepsilon)$-approximate distance query between any two vertices in G in $O(c^{2d+2}(\frac{1}{\varepsilon})^d)$ time.*

4 A Well-Separated Pair Decomposition for c-packed Graphs

Given a c-packed graph $G(V,E)$ in d dimensions, where d is fixed, and a positive constant σ, we show how to deterministically construct a linear-size WSPD$_G$ with separation factor σ (Section 4.2). Section 4.1 introduces some important terminology. In Section 4.2 we show that one can construct such a well-separated pair decomposition in $O(n \log n + c^3 \sigma n)$ time.

4.1 Notation and Preliminaries

Given a geometric graph $G(V,E)$ and cube s in $\mathbb{R}^d$, we define the diameter of s to be the length of a diagonal and denote it as $diam(s)$. We define the radius of s to be half the diameter and denote it as $rad(s)$. We define $V(s)$ to be the subset of vertices of V within s. We will need the following definitions.

▶ **Definition 7** (δ-Connected Set)**.** *Given a geometric graph $G(V,E)$ and a cube s in $\mathbb{R}^d$. Let s^+ be the concentric cube with twice the diameter. Two vertices $u, v \in V(s)$ are* δ-connected *if there is a path between u and v that lies within s^+ and has length at most $\delta \cdot diam(s)$. We say that a set of vertices $C \subseteq V(s)$ is a δ-connected set with respect to s if all pairs of vertices in C are δ-connected, and no vertex in $V(s) \setminus C$ is δ-connected to a vertex in C.*

▶ **Definition 8** (Partition into δ-Connected Sets)**.** *Let s be a cube in $\mathbb{R}^d$. A partition $\Psi_\delta(V(s)) = \{C_1, ...C_k\}$ of $V(s)$ into δ-connected sets is a set of k disjoint subsets of $V(s)$ that satisfies the following properties:*

1. $\bigcup_{1 \le i \le k} C_i = V(s)$.
2. *For all i s.t. $1 \le i \le k$, C_i is a δ-connected set with respect to s.*

4.2 Constructing a WSPD for c-packed graphs

Split trees [7] or quadtrees [37] are commonly used in the construction of a geometric WSPD of a point set. An essential property of these is that the Euclidean distance between two points contained in the same cell is always bounded by the diameter of the cell. In order to construct a WSPD$_G$ we construct a new type of tree, which we will refer to as a δ-connected tree (δ-CT). This tree will satisfy the similar property, but relative to *graph distances* between the points in the graph, rather than their Euclidean distances. The main difference is that cells in a δ-connected tree may overlap. We formally define this tree below.

▶ **Definition 9** (δ-Connected Tree (δ-CT)). *Given a connected geometric graph $G(V, E)$ in $\mathbb{R}^d$ and a quadtree Q of V, a δ-connected tree T of G is a rooted tree, where every node u stores a cube s, corresponding to a cell of Q, and a representative point of a δ-connected set with respect to s. The root of T stores the cube s corresponding to the cell stored in the root of Q, and every leaf contains a single point in V. In particular, the leaves contained in the subtree rooted at u contain the points in V that are represented by the point stored at u.*

Once we have a δ-CT, it remains to upper bound the graph diameter of the δ-connected set represented in each cell. With this upper bound, one can then follow a standard approach to compute a WSPD$_G$ with separation factor σ from the δ-CT.

In Subsection 4.2.1 we show how to construct a c-CT, T, of a c-packed graph G. In Subsection 4.2.2 we then show how to upper bound the diameter of the c-connected set represented in each cell of T. In Subsection 4.2.4 we analyze the complexity of the construction.

4.2.1 Constructing a c-Connected Tree

The algorithm to compute a c-CT takes as input a c-packed graph G, as well as its corresponding compressed quadtree Q. Without loss of generality we assume that the cell representing the root of Q is a unit cube. The *level value* of a node u in Q is said to be i if the cell/cube $s(u)$ stored at u has side length 2^{-i}. The root of Q has level value 0.

We are now ready to construct a c-CT, T. Initially, every leaf in Q becomes a leaf in T, and are then deleted from Q. Assume w.l.o.g. that the number of distinct level values in the remaining compressed quadtree Q' is ℓ and let $I = \langle i_1, i_2, \ldots, i_\ell \rangle$ be the level values sorted in decreasing order. Let U_{i_j} be the set of nodes in Q' having level value i_j, $1 \le j \le \ell$.

Next, we iteratively process the values in I in decreasing order. For each value i_j in I we will build a graph H_{i_j} from $H_{i_{j-1}}$. Initially we set $H_{i_0}(V_{i_0}, E_{i_0}) = G$.

Set $H_{i_1} = H_{i_0}$. For each node $u \in U_{i_1}$, let $s^+(u)$ be a concentric cube of twice the diameter of $s(u)$. The algorithm picks an arbitrary point p in $V_{i_1}(s(u))$ and merges all nodes in $V_{i_1}(s(u))$ that are path-connected to p in H_{i_0} within the cube $s^+(u)$. For all edges (u, v), if both u and v have been merged, we remove the edge. If only one endpoint has been merged, we move this endpoint to p and redefine the length of the edge to be the Euclidean distance between this point and p. Finally, we remove any parallel edges. Note that all changes are made to H_{i_1}, while H_{i_0} stays unaffected. The surviving node p is the representative point in H_{i_1} of the merged set of nodes in H_{i_0}. A node is added to T storing p and its children are the nodes in T storing the points that were merged into p. The algorithm repeats this process until all the points in $V_{i_1}(s(u))$ have been merged into representative points and the process has been complete for each $u \in U_{i_1}$.

In the next iteration, we let $H_{i_2} = H_{i_1}$. The nodes in U_{i_2} are now processed using H_{i_1} as input to merge the nodes in H_{i_2}. This continues until all the level values in I have been processed. The process for iteration j of the algorithm is visualised in Figure 1.

The proof of Lemma 10 can be found in the full version of the paper.

▶ **Lemma 10.** *Given a c-packed graph $G(V, E)$ the above algorithm produces a valid c-connected tree T of $G(V, E)$.*

4.2.2 Bounding the Graph Diameter of a Cell in the c-CT

In order to construct the well-separated pairs, it is necessary to upper bound the graph diameter of the c-connected set represented in each cell of the c-CT, T, constructed above. Let u be a node in Q. Denote the corresponding canonical cube as $s(u)$, and a concentric cube

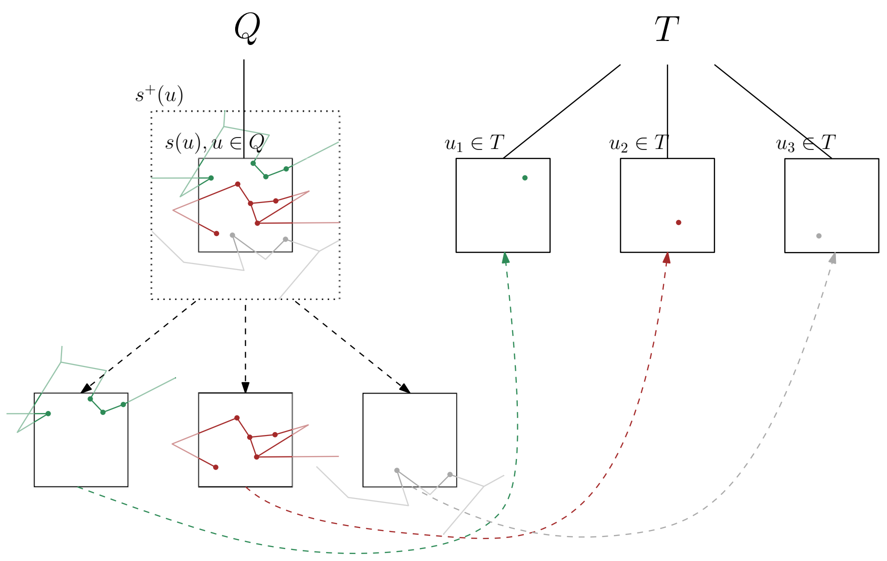

Figure 1 An illustration of iteration j of the algorithm constructing the c-connected tree.

of twice its diameter as $s^+(u)$. We call a connected subgraph of G a connected component of a cell u if it is fully contained within $s^+(u)$, with at least one vertex inside $s(u)$. To compute an upper bound on the graph diameter of the c-connected set we instead give an upper bound on *any* connected component of $u \in Q$. This upper bound can then be copied across during the construction of T. We show how to compute this upper bound below.

Observe that in order for an edge to contribute to a connected component of u, its length can be at most twice the diameter of $s(u)$. Let us call the set of edges of length at most twice the diameter, which overlap with $s(u)$, *relevant* edges w.r.t. $s(u)$ (see Figure 2).

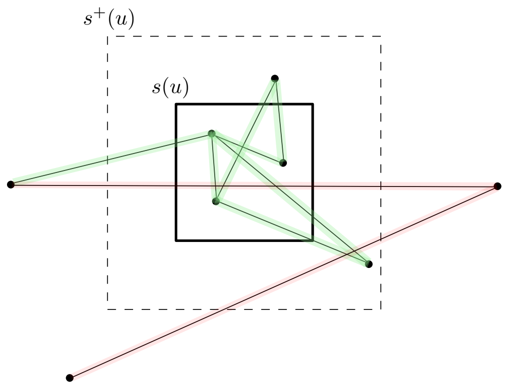

Figure 2 Set of *relevant* edges w.r.t. $s(u)$ (green edges) and *irrelevant* edges (red edges).

Let $N(s(u))$ be the set of $3^d - 1$ neighboring canonical cubes on a grid (note that not all members of $N(s(u))$ may correspond to cells in Q). Observe that all edges in any connected component associated with u must be a subset of the union of the relevant edges w.r.t. $s(u)$, and all the relevant edges w.r.t. cubes in $N(s(u))$. We refer to the total overlapping length of $s(u)$ with relevant edges w.r.t. $s(u)$ as $\mathrm{REL}(s(u))$. The above observation allows us to upper bound the graph diameter of any connected component associated with $u \in Q$ by $\mathrm{REL}(s(u)) + \sum_{s_n \in N(s(u))} \mathrm{REL}(s_n)$.

It thus remains to compute, for all $u \in Q$, $\mathrm{REL}(s(u))$ and $\mathrm{REL}(s_n)$ for all $s_n \in N(s(u))$. Let $\ell(e)$ denote the length of an edge $e \in E$. For all $e \in E$ we compute the canonical cubes of size 2^{-i} such that $2^{-i} = \lceil \ell(e)/2 \rceil$. Denote this set of canonical cubes as S. For each cube $s \in S$, store the length of the overlap of the edge with the cube. Note that this correctly gives us the value $\mathrm{REL}(s)$ for all $s \in S$. One can now use Lemma 2.11 from [24] to compute a compressed quadtree Q_S from S. For any cell v in Q_S whose corresponding cube is not in S, initialize its value $\mathrm{REL}(v) = 0$. Next, one can use a bottom-up approach to compute, for every cell in Q_S its relative edge length. For each parent, simply add the relative edge lengths stored in the children to its current value. In order to perform efficient searching on Q_S we preprocess it into a finger tree, T_f, using similar techniques to Theorem 2.14 in [24]. We are now ready to compute, for each cell $u \in Q$, an upper bound on the graph diameter of any connected component. For each $u \in Q$, search for the largest cell in T_f whose corresponding cube is fully contained within $s(u)$, and return the relative edge length stored at the cell. If such a cell does not exist, return zero. Repeat the search for each $s \in N(s(u))$ and add up the results. The resulting value is our upper bound on the graph diameter of any connected component of the cell u. We call this value $\mathrm{dub}_G(u)$.

When constructing the c-CT from Q using the algorithm in Section 4.2.1, we upper bound the graph diameter of any connected component of $u \in Q$ by $\mathrm{dub}_G(u)$.

The proof of Lemma 11 can be found in the full version of the paper.

▶ **Lemma 11.** *The value $dub_G(v)$ for any cell in $v \in T$ upper bounds the graph diameter of the connected subgraph represented in the cell v.*

We are now ready to describe the construction of a WSPD_G for c-packed graphs.

4.2.3 Constructing the WSPD$_G$

Construct a compressed quadtree Q and compute, for each cell $u \in Q$, $\mathrm{dub}_G(u)$ using the techniques described in Section 4.2.2. Next, construct a c-CT, T, from Q using the algorithm described in Section 4.2.1 and store, for each cell $v \in T$, created from $u \in Q$, $\mathrm{dub}_G(u)$ as an upper bound on the graph diameter of the connected component represented in v. Apply the algorithm of [24] (Section 3.1.1) to compute a WSPD_G from T. To determine whether a pair of nodes is well-separated we use the upper bound on the graph diameter and the Euclidean distance between their representatives.

To bound the size of the resulting WSPD_G we first bound the size of T by observing that each cell $u \in Q$ contains at most $O(c)$ c-connected sets w.r.t. u.

▶ **Lemma 12.** *For all $u \in U_{i_j}$, $1 \leq j \leq \ell$, each vertex in H_{i_j} within $s(u)$ represents a c-connected component in $H_{i_{j-1}}$ with respect to $s(u)$.*

Proof. We argue the algorithm maintains this property by induction on the number of iterations. The algorithm initially uses as input the c-packed graph $H_{i_1} = H_{i_0} = G$. After the first iteration of the algorithm, for each node $u \in U_{i_1}$, each surviving point x in V_{i_1} represents a set of points in V_{i_0} that were connected to x within $s^+(u)$. Since $H_{i_0} = G$ is c-packed, we can conclude that the distance between x and any point merged into x is at most c times the radius of $s^+(u)$, and therefore c times the diameter of $s(u)$.

Assume that the algorithm maintains the invariant for iterations 1 to $k < \ell$. Note that the graph H_{i_k}, used as input for iteration $k+1$, is no longer guaranteed to satisfy c-packedness. However, observe that after each merge step, the representative nodes in $V_{i_{k+1}}$ always correspond to original nodes present in G. Since the process of merging each connected set into a single node can only reduce the length of the paths between the remaining representatives, the graph distances between the remaining nodes in H_{i_k} can only have decreased in comparison to their original graph distance in G. We therefore maintain the property that, for every node $u \in U_{i_{k+1}}$, for each representative point in $V_{i_{k+1}}$ within u, the points in V_{i_k} that it represents can be at most c times the diameter of u apart in H_{i_k}. ◀

With the help of Lemma 12 one can bound the size of T.

▶ **Lemma 13.** *For every node $u \in Q$, the number of nodes added to T is at most $O(c)$.*

Proof. From Lemma 12 we know that each node in $V_{i_l}(s(u))$ must represent a valid c-connected set in $H_{i_{l-1}}(s(u))$. We now observe that one can upper bound the number of c-connected sets in $H_{i_{l-1}}(s(u))$ by the number of c-connected sets in $G(s(u))$. This observation follows from the fact that for any $1 \le j \le \ell$, H_{i_j} was constructed by merging c-connected sets w.r.t. $s(u)$ in $H_{i_{j-1}}$, for every $u \in U_{i_j}$. Note that the argument used in the proof of Lemma 12 shows that this process, for every j, can only cause the length of the paths between surviving nodes to become smaller. Since the position of surviving nodes does not change from their original position in G, it follows that the number of c-connected components in H_{i_j} within $s(u)$ can only be smaller than that in G within $s(u)$. It thus remains to argue that the number of c-connected components within each $s(u)$, for all $u \in V$ is at most $O(c)$. To this end, note that in order for any pair of points in $V(s)$ to be disconnected within s^+, every path between them must contain a subpath which intersects the boundaries of s and s^+ and therefore has length at least $\frac{r}{\sqrt{d}}$, as shown in Figure 3. From the c-packedness of G we conclude there cannot be more than $\frac{2cr\sqrt{d}}{r} = 2c\sqrt{d}$ such paths in s^+. We conclude there can be at most $O(c)$ c-connected sets found by the algorithm. ◀

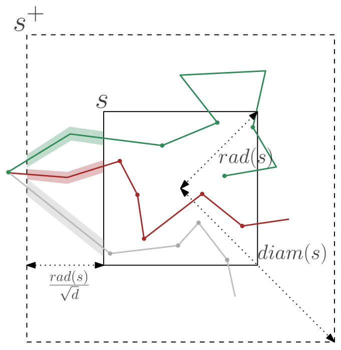

Figure 3 Two c-connected components in s must be separated by a path of length at least $\frac{rad(s)}{\sqrt{d}}$.

With the help of the above lemma we can analyse the size of the resulting WSPD$_G$ using a simple charging argument inspired by the one used in [24].

▶ **Lemma 14.** *The resulting WSPD$_G$, W, is σ-well separated and is of size $O(c^3\sigma n)$.*

Proof. The fact that W is σ-well separated follows from our use of the Euclidean distance as a lower bound on the distance between two representatives, and our use of dub$_G$ (Section 4.2.2) as upper bound on the graph diameter, to determine whether a pair is σ-well separated.

To prove the size of W we use a simple charging argument. For a pair $(u, v) \in W$, let $\pi(v)$ denote the parent of v in T. Assume the last call to the algorithm in [24] (Section 3.1.1) was via $(\pi(v), u)$. We charge the pair to u and argue any node can be charged at most $O(c^2\sigma)$ times. Since the pair $(\pi(v), u)$ was considered not well-separated by the algorithm, it follows that $d(\pi(v), u) \leq d_G(\pi(v), u) < \text{dub}_G(\pi(v))\sigma$. Consider a geometric ball $B(u, r)$ of radius $r = (1+\sigma)\text{dub}_G(\pi(v))$ centered at u. Note that the connected set in G represented by $\pi(v)$ must be fully contained inside B. By c-packedness, there can be at most $c\sigma + 1$ candidates for $\pi(v)$. Since $\pi(v)$ can have at most $2^d \cdot O(c)$ children, it follows that there are $O(c^2\sigma)$ candidates for v. We conclude a node u can be charged at most $O(c^2\sigma)$ in this way. By Lemma 13 there are $O(cn)$ nodes in T. This concludes the lemma. ◀

We now argue the runtime and space complexity of our construction.

4.2.4 Complexity analysis

In Lemma 15 the runtime of the construction of the c-CT is analysed, as well as the size of the resulting tree. We then analyse the runtime and space complexity for computing the upper bound on the graph diameter of the subgraph represented in each cell of the c-CT in Lemma 16. Finally, with the help of these lemmas we analyse the construction time and space complexity of our WSPD$_G$ in Theorem 3. The proofs of Lemma 15 and Lemma 16 can be found in the full version of the paper.

▶ **Lemma 15.** *Given a c-packed graph G, one can construct a c-CT of size $O(cn)$ in $O(n \log n + c^2 n)$ time.*

▶ **Lemma 16.** *Let G be a c-packed graph and Q its corresponding compressed quadtree. One can compute an upper bound on the graph diameter of any c-connected component contained in u, for all $u \in Q$, in $O(cn \log n)$ time, using at most $O(cn)$ space.*

With the help of the above lemmas we obtain the following theorem.

▶ **Theorem 3.** *Given a c-packed graph G in $\mathbb{R}^d$, for fixed d, one can construct a WSPD$_G$ with separation factor σ of size $O(c^3\sigma n)$ in $O(cn \log n + c^3\sigma n)$ time, using $O(cn)$ space.*

Proof. Constructing the compressed quadtree Q requires $O(n \log n)$ time and $O(n)$ space. Upper bounding the diameter using the techniques introduced in Section 4.2.2 can be done in $O(cn \log n)$ time, using $O(cn)$ space (Lemma 16). We conclude from Lemma 15 that we can construct a c-CT, T, from Q in $O(c^2 n)$ time, using $O(cn)$ space. Finally, using the construction from [24] we can compute the WSPD$_G$ from T in $O(c^3\sigma n)$ time using $O(cn)$ space. This concludes the proof. ◀

5 A Separator Theorem for c-packed Graphs

A subset of vertices C of a graph with n vertices is called a (balanced) separator if the remaining vertices can be partitioned into two sets A and B such that there are no edges between A and B, with $|A| < \alpha \cdot n$ and $|B| < \alpha \cdot n$ for some constant $1/2 \leq \alpha < 1$. The subset C is said to *α-separate* the graph.

In this section we show how to compute a separator of size $O(c)$ for any c-packed graph that $(1 - \frac{1}{2\lambda^3})$-separates the graph. The constant λ is the doubling constant of $\mathbb{R}^d$.

We have the following separating lemma.

▶ **Lemma 17.** *Given a set P of n points in $\mathbb{R}^d$, where d is fixed, one can compute a ball $b(p, r)$, which is centered at p and has radius r, such that $\mathbf{b}(p, r)$ contains at least $n/(2\lambda^3)$ points of P, where λ is the doubling constant of $\mathbb{R}^d$, and $\mathbf{b}(p, 2r)$ of twice the radius contains at most $n/2$ points of P. The running time of the algorithm is $O(\lambda^{3d} n)$.*

Proof. Let $r_{opt}(P,k)$ be the radius of the smallest ball enclosing k points of P. Har-Peled and Mazumdar [24] gave an algorithm for computing a k-enclosing ball with radius at most $2r_{opt}(P,k)$ in $O(n(n/k)^d)$ time. As a result, one can compute a $(n/(2\lambda^3))$-enclosing ball $\mathbf{b}(p,r)$ in $O(\lambda^{3d} \cdot n)$ time.

Now one can prove that $\mathbf{b}(p,2r)$ contains at most $n/2$ points. Since $r \le 2r_{opt}(P,n/(2\lambda^3))$, by the doubling property, $\mathbf{b}(p,2r)$ can be covered by at most λ^3 balls of radius $r_{opt}(P,n/(2\lambda^3))/2$. Note that any ball of radius $r_{opt}(P,n/(2\lambda^3))/2$ contains strictly less than $n/(2\lambda^3)$ points. Thus $\mathbf{b}(p,2r)$ contains at most $n/2$ points. This concludes the proof of the lemma. ◀

Based on Lemma 17, we can show that one can construct separators of size $O(c)$ that $(1-\frac{1}{2\lambda^3})$-separate c-packed graphs, as shown in Theorem 4.

▶ **Theorem 4.** *Given a c-packed graph G in $\mathbb{R}^d$, where d is fixed, with n vertices, one can find a separator of size $O(c)$ that $(1-\frac{1}{2\lambda^3})$-separates G, in $O(c^2n)$ time.*

Proof. First we construct a separator C. Then we prove that C satisfies the required properties. Finally we analyze the running time of the algorithm.

Let P be the vertices of c-packed graph G and run the algorithm described in Lemma 17. Let $\mathbf{b}(p,r)$ be the computed ball that contains at least $n/(2\lambda^3)$ vertices of G. Let $\hat{A}$ be the vertices of G contained in $\mathbf{b}(p,r)$ and let $\hat{B}$ be the vertices of G lying outside $\mathbf{b}(p,2r)$. See Figure 4 for an illustration.

Now set up a max-flow instance. A similar construction was used in Lemma 11 of [20]. Set the capacity of each edge of G to 1. Let the vertices in $\hat{A}$ be the sources, and let the vertices in $\hat{B}$ be the sinks. Run the Ford-Fulkerson algorithm ([31], Chapter 7) on the instance. According to the max-flow min-cut theorem ([31], Chapter 7), the max-flow of the instance is equal to its min-cut. Let (S,T) be a min-cut and let $\{e_1,\dots,e_l\}$ be the set of edges from S to T, where $\hat{A} \subseteq S$ and let $\hat{B} \subseteq T$. Choose one endvertex for each edge in $\{e_1,\dots,e_l\}$ and add it to C (C is initially empty). Let $A = S \setminus C$ and let $B = T \setminus C$. This finishes the construction of C.

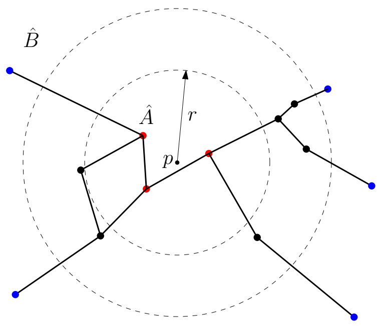

Figure 4 Illustration of the proof of Theorem 4 where p is the center of ball $\mathbf{b}(p,r)$. Vertices in $\hat{A}$ are drawn in red. Vertices in $\hat{B}$ are drawn in blue. The value of the min-cut in the figure is 4.

Next we show that C (i) has size $O(c)$, and (ii) $(1-\frac{1}{2\lambda^3})$-separates G. Property (ii) follows from (S,T) being a min-cut. Since $\hat{A}$ contains at least $n/(2\lambda^3)$ vertices of G and $\hat{A} \subset S$, A contains at least $n/(2\lambda^3)$ vertices of G. Since $\hat{B}$ contains at least $n/2$ vertices of G and $\hat{B} \subset T$, B contains at least $n/2$ vertices of G. Property (i) follows from c-packedness. In the max-flow instance, the value of the max-flow is l. Since all the edges have capacity 1, there are l edge-disjoint paths from $\hat{A}$ to $\hat{B}$. Each such path pierces both the inner and the outer boundaries of the d-dimensional spherical shell $\mathbf{b}(p, 2r) \setminus \mathbf{b}(p, r)$. Due to c-packedness, the length of edges inside the spherical shell is at most $2cr$. Since the width of the spherical shell is r, there are at most $2c$ edge-disjoint paths from $\hat{A}$ to $\hat{B}$. Therefore $|C| = l \leq 2c$.

Finally we analyze the running time of the algorithm. Running the algorithm in Lemma 17 on the vertices of G takes $O(\lambda^{3d} n)$ time. We use the Ford-Fulkerson algorithm to solve the max-flow instance. The running time of the algorithm is proportional to the number of edges in G times the value of the max-flow. Since $l \leq 2c$ and there are $O(cn)$ edges in G, solving the max-flow instance takes $O(c^2 n)$ time. Since $\lambda = O(d)$, the overall running time of the algorithm is $O(c^2 n)$. ◀

Putting the above results together gives us an exact distance oracle. We refer the reader to the full version of the paper for further details.

▶ **Theorem 18.** *Given any c-packed graph G with n vertices, using $O(c^2 n \log n + cn \log^2 n)$ preprocessing time and $O(cn \log n)$ space, a distance query between any two vertices in G can be answered in $O(c \log n)$ time.*

This enables us to obtain Theorem 5 and Corollary 6 (see the full version of the paper for details).

References

1 N. Alon, P. D. Seymour, and R. Thomas. A separator theorem for graphs with an excluded minor and its applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 293–299, 1990.

2 S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. H. M. Smid. Euclidean spanners: short, thin, and lanky. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC)*, pages 489–498, 1995.

3 Y. Bartal, O. N. Fandina, and O. Neiman. Covering metric spaces by few trees. *J. Comput. Syst. Sci.*, 130:26–42, 2022. `doi:10.1016/J.JCSS.2022.06.001`.

4 U. Bertele and F. Brioschi. *Nonserial dynamic programming*. Academic Press, Inc., 1972.

5 F. Brüning, J. Conradi, and A. Driemel. Faster approximate covering of subcurves under the Fréchet distance. In *Proceedings of the 30th Annual European Symposium on Algorithms, (ESA)*, volume 244 of *LIPIcs*, pages 28:1–28:16, 2022. `doi:10.4230/LIPICS.ESA.2022.28`.

6 K. Buchin, M. Buchin, J. Gudmundsson, A. Popov, and S. Wong. Map matching queries under Fréchet distance on low-density spanners. In *Proceedings of the 40th Annual Symposium on Computational Geometry (SoCG)*, 2024.

7 P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, 1995. `doi:10.1145/200836.200853`.

8 H. Chang, J. Conroy, H. Le, L. Milenkovic, and C. Than. Covering planar metrics (and beyond): O(1) trees suffice. In *Proceedings of the 64th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2231–2261, 2023.

9 S. Chaudhuri and C. D. Zaroliagis. Shortest paths in digraphs of small treewidth. part I: sequential algorithms. *Algorithmica*, 27(3):212–226, 2000. `doi:10.1007/S004530010016`.

10 B. Chazelle. Computing on a free tree via complexity-preserving mappings. *Algorithmica*, 2:337–361, 1987. `doi:10.1007/BF01840366`.

11 D. Chen, A. Driemel, L. J. Guibas, A. Nguyen, and C. Wenk. Approximate map matching with respect to the Fréchet distance. In *Proceedings of the 13th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 75–83, 2011.

12 J. Conradi, A. Driemel, and B. Kolbe. (1+ϵ)-ANN data structure for curves via subspaces of bounded doubling dimension. *Comput. Geom. Topol.*, 3(2):6:1–6:22, 2024. URL: `https://www.cgt-journal.org/index.php/cgt/article/view/45`.

13 A. Driemel and S. Har-Peled. Jaywalking your dog: Computing the Fréchet distance with shortcuts. *SIAM Journal on Computing*, 42(5):1830–1866, 2013. doi:10.1137/120865112.

14 A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discret. Comput. Geom.*, 48(1):94–127, 2012. doi:10.1007/S00454-012-9402-Z.

15 Z. Dvorák and S. Norin. Treewidth of graphs with balanced separations. *J. Comb. Theory B*, 137:137–144, 2019. doi:10.1016/J.JCTB.2018.12.007.

16 J. Gao and L. Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM J. Comput.*, 35(1):151–169, 2005. doi:10.1137/S0097539703436357.

17 J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5(3):391–407, 1984. doi:10.1016/0196-6774(84)90019-1.

18 J. Gudmundsson, Z. Huang, A. van Renssen, and S. Wong. Computing a subtrajectory cluster from c-packed trajectories. In *Proceedings of the 34th International Symposium on Algorithms and Computation (ISAAC)*, volume 283 of *LIPIcs*, pages 34:1–34:15, 2023. doi:10.4230/LIPICS.ISAAC.2023.34.

19 J. Gudmundsson, C. Levcopoulos, G. Narasimhan, and M. H. M. Smid. Approximate distance oracles for geometric spanners. *ACM Trans. Algorithms*, 4(1):10:1–10:34, 2008. doi:10.1145/1328911.1328921.

20 J. Gudmundsson, M. P. Seybold, and S. Wong. Map matching queries on realistic input graphs under the Fréchet distance. In *Proceedings of the 34th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1464–1492, 2023.

21 J. Gudmundsson and M. H. M. Smid. Fast algorithms for approximate Fréchet matching queries in geometric trees. *Comput. Geom.*, 48(6):479–494, 2015. doi:10.1016/J.COMGEO.2015.02.003.

22 J. Gudmundsson and S. Wong. A well-separated pair decomposition for low density graphs. *CoRR*, abs/2411.08204, 2024. doi:10.48550/arXiv.2411.08204.

23 A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the 44th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 534–543, 2003.

24 S. Har-Peled. *Geometric Approximation Algorithms.* American Mathematical Society, 2011.

25 S. Har-Peled and M. Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006. doi:10.1137/S0097539704446281.

26 S. Har-Peled and K. Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. *SIAM J. Comput.*, 46(6):1712–1744, 2017. doi:10.1137/16M1079336.

27 S. Har-Peled and Benjamin Raichel. The Fréchet distance revisited and extended. *ACM Trans. Algorithms*, 10(1):3:1–3:22, 2014. doi:10.1145/2532646.

28 K. Kawarabayashi, P. N. Klein, and C. Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 6755 of *Lecture Notes in Computer Science*, pages 135–146, 2011. doi:10.1007/978-3-642-22006-7_12.

29 K. Kawarabayashi, C. Sommer, and M. Thorup. More compact oracles for approximate distances in undirected planar graphs. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 550–563, 2013.

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

31 Jon M. Kleinberg and Éva Tardos. *Algorithm design.* Addison-Wesley, 2006.

32 R. Kyng, R. Peng, R. Schwieterman, and P. Zhang. Incomplete nested dissection. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 404–417, 2018.

33 H. Le and C. Wulff-Nilsen. Optimal approximate distance oracle for planar graphs. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 363–374, 2021.

34 R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math*, 36(2):177–189, 1979.

35 R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980. doi:10.1137/0209046.

36 G. L. Miller and W. P. Thurston. Separators in two and three dimensions. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 300–309, 1990.

37 G. Narasimhan and M. H. M. Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

38 K. Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 281–290, 2004.

39 M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004. doi:10.1145/1039488.1039493.

40 M. Thorup and U. Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005. doi:10.1145/1044731.1044732.

41 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 Algorithms (SODA)*, pages 351–362, 2016.