Cutting Planarians: Planar Emulators for String Graphs*

Hsien-Chih Chang
Dartmouth College
Hanover, USA
hsien-chih.chang@dartmouth.edu

Jonathan Conroy
Dartmouth College
Hanover, USA
jonathan.conroy.gr@dartmouth.edu

Zihan Tan
University of Minnesota
Minneapolis, USA
ztan@umn.edu

Da Wei Zheng
Institute of Science and Technology Austria
Klosterneuburg, Austria
dzheng@ista.ac.at

Abstract

In this paper we construct distance sketches for intersection graphs of *arbitrary* path-connected regions in the plane (known as the *string graphs*) in the constant and $1+\varepsilon$ distortion regimes. Furthermore, the distance sketches themselves are *planar graphs*. First, we show that every unweighted string graph G has an $O(1)$-distortion *planar emulator*: that is, there exists an edge-weighted planar graph H containing every vertex in G, such that every pair of vertices (u,v) satisfies $\delta_G(u,v) \le \delta_H(u,v) \le O(1)\cdot\delta_G(u,v)$. Furthermore, we show that for any constant $\varepsilon > 0$, there is an edge-weighted planar graph H' such that every pair of vertices (u,v) satisfies $\delta_G(u,v) \le \delta_{H'}(u,v) \le (1+\varepsilon)\cdot\delta_G(u,v) + O(\varepsilon^{-4}\operatorname{poly}\log n)$. No previous constructions of sparse distance sketches were known even for intersection graphs of simple shapes like axis-parallel rectangles or fat convex polygons.

As applications, we construct the first $(1+\varepsilon, +O(1))$ mixed-distortion tree cover and distance oracle for arbitrary string graphs, as well as the first additive $+(\varepsilon\Delta + O(1))$-distortion embedding of string graphs G with diameter Δ into graphs of constant treewidth $O(\varepsilon^{-4})$.

CCS Concepts

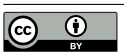
• **Theory of computation → Computational geometry**; **Shortest paths**; • **Mathematics of computing → Graphs and surfaces**.

Keywords

Planar graphs, Geometric intersection graphs, Emulators, Metric embedding

ACM Reference Format:
Hsien-Chih Chang, Jonathan Conroy, Zihan Tan, and Da Wei Zheng. 2026. Cutting Planarians: Planar Emulators for String Graphs. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC '26), June 22–26, 2026, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3798129.3800917

*Full version of this paper can be found at https://arxiv.org/abs/2510.21700



STOC '26, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-2536-4/2026/06
https://doi.org/10.1145/3798129.3800917

1 Introduction

Given a set of geometric objects $\mathcal{S}$, the *intersection graph* of $\mathcal{S}$, denoted $G_{\mathcal{S}}$, is a graph with vertex set $\mathcal{S}$, such that an edge exists between two objects if and only if they intersect. The most general form of geometric intersection graph in the plane is the class of *string graphs*, where vertices are arbitrary path-connected regions. One can assume without loss of generality that the regions are connected arcs, thus the name "string" graphs. One may also study intersection graphs of more restricted class of objects, such as disks, fat objects, or axis-aligned rectangles.

Our goal is to study *distances* between objects on a geometric intersection graph G. The *distance* between two vertices u and v in G, denoted $\delta_G(u,v)$, is the number of edges in the shortest path between u and v. (Note that we assume edges are unweighted; if we allowed arbitrary edge weights then all metrics are realizable as string graph metrics, as the complete graph is a string graph.[1]) Distance problems on geometric intersection graphs have attracted a great deal of attention, including diameter computation [7, 10, 12, 13, 17], clustering [4, 26], spanners [6, 9, 11, 21, 43], low-diameter decomposition and its applications to LP rounding [32–34, 37], and much more. However, few results have been obtained for the most general string graphs beyond restricted classes of shapes, such as unit-disk graphs (UDGs) and axis-parallel unit-square graphs.

In partial explanation of the success enjoyed on unit-disk graphs, several results on UDGs can be traced back to the existence of an $O(1)$-distortion *planar spanner*: every unit-disk graph G has a subgraph H which is planar, such that for every pair of vertices (u,v), we have $\delta_G(u,v) \le \delta_H(u,v) \le O(1)\cdot\delta_G(u,v)$. In other words, the techniques that enables the results on UDGs relies on the similarity between unit-disk metrics and planar metrics. Planar metrics are well studied, and tools from planar graphs (in particular, shortest-path separators) can be combined with the planar spanner to design algorithms for UDGs. The $O(1)$-distortion planar spanner for UDGs is helpful for distance problems even in the $1+\varepsilon$ regime; for example, it is a crucial ingredient for a near-linear time $(1+\varepsilon)$-approximate diameter algorithm and a compact $(1+\varepsilon)$-approximate distance oracle [13], and in the design of $(1+\varepsilon)$-approximate coresets for clustering [4]. (Note that one can also obtain shortest-path separators on UDG directly, not using a planar spanner [28, 43]; this is

[1] For some classes of geometric intersection graphs, one can instead impose a natural weight on the edges—for example, a unit-disk graph may have edge weights set to be the Euclidean distance between the disk centers. We are primarily concerned with unweighted string graphs.

another way in which the metric structure of UDGs is similar to that of planar metrics.)

The use of planar techniques for geometric intersection graphs so far has seemed somewhat limited to UDGs and closely related classes: for example, the only classes of intersection graphs with planar spanners are Euclidean weighted UDGs [35], unweighted UDGs [6], and Euclidean weighted unit-square graphs [4]. General string graphs could have much more complex interactions than UDGs. Unlike disks and squares, objects like segments and rectangles may be *piercing* and thus permit non-local interactions; moreover, an n-vertex string graph may even require a representation (i.e. a drawing on the plane) where the number of crossings is exponential in n [31].[2]

Main Results. It might come as a surprise then, that in this paper we demonstrate how to construct a planar distance sketching structure for *general* string graphs, even when the objects are arbitrarily complex and piercing. Our first main result vastly generalizes the previous spanner constructions on UDGs: every string graph admits an $O(1)$-distortion *planar emulator*. No similar construction was known before even for intersection graphs of very restricted shapes, like axis-parallel rectangles or fat convex polygons. Philosophically speaking, our result suggests that metrics on string graphs are not so different from planar metrics in the $O(1)$-distortion regime.

Theorem 1.1. *For an absolute constant c, the following is true: For any unweighted string graph G, there is a weighted planar graph H with $V(G) \subseteq V(H)$, such that for every pair of vertices (u, v) in G,*

$$\delta_G(u,v) \le \delta_H(u,v) \le c \cdot \delta_G(u,v).$$

Moreover, if one is given a representation of G, the graph H can be computed in time polynomial in the number of crossings in the representation.[3]

Note that Theorem 1.1 allows for the number of vertices in H to be much larger than the number of vertices in G. Indeed, the number of vertices in H scales linearly in the complexity of the representation of G, which is perhaps unsatisfying because some string graphs require exponential-size representations [31]. However, this is simple to fix: we can reduce the number of vertices in H by applying $O(1)$-distortion Steiner point removal [16] on the set of terminals $V(G) \subseteq V(H)$. This produces an edge-weighted planar graph H' with $V(H') = V(G)$, which is an $O(1)$-distortion planar emulator of G. Note that our algorithm to *construct* such an H' still requires time polynomial in the size of the representation of G.

Our second main result might come as an even bigger surprise. Not only do $O(1)$-distortion planar emulator exists for arbitrary n-vertex string graphs, by changing the weight of the edges of the emulator, it is possible to preserve distances with a $(1+\varepsilon)$ multiplicative factor, with only an extra $\operatorname{poly}\log n$ additive distortion. Equivalently, this is a $(1+\varepsilon)$-distortion planar emulator that preserves distances between pairs of vertices that are at least polylogarithmically away. We say that arbitrary string graph admits an $(\alpha, +\beta)$-*mixed-distortion* planar emulator for $\alpha = 1+\varepsilon$ and $\beta = \operatorname{poly}\log n$. We focus on string graphs, but a similar result actually holds more generally for every graph class that admits $O(1)$-distortion planar emulators.

Theorem 1.2. *Let G be a n-vertex string graph. For any small constant $\varepsilon > 0$, there is a weighted planar graph H with $O(n)$ vertices and $V(G) \subseteq V(H)$, such that for every pair of vertices (u, v) in G,*

$$\delta_G(u,v) \le \delta_H(u,v) \le (1+\varepsilon) \cdot \delta_G(u,v) + O(\varepsilon^{-4} \cdot \log^{18} n).$$

Moreover, if one is given a representation of G, the graph H can be computed in time polynomial in the number of crossings in the representation.

This result provides the first compelling evidence towards a conjecture of Agelos Georgakopoulos (as described by Nguyen, Scott, and Seymour [38]) in coarse graph theory: Every (unweighted) graph that can be embedded into a planar graph with $(\alpha, +\beta)$-distortion can also be embedded into a planar graph with pure additive distortion $+\beta'$, for some (possibly bigger) constant β'. (The paper [38] actually gives a more general conjecture for arbitrary class of graphs instead of just planar graphs, and in terms of *quasi-isometry*.) The conjecture may seem bold, but there are no known counter-examples. It is known to hold in other very restricted classes of graphs, such as trees [19, 30] and bounded-pathwidth graphs [38]. While we remain neutral on the validity of the original conjecture, we leave as an open question whether a $(1+\varepsilon, +O(1))$ or a $(1, +\operatorname{poly}\log n)$-distortion planar emulator exists.

Applications. In addition to $(1+\varepsilon, +\operatorname{poly}\log n)$-mixed-distortion planar emulators, we have two other applications on constructing distance-sketching structures for any graph class $\mathcal{G}$ (closed under induced subgraphs) that admits $O(1)$-distortion planar emulators. First, we can construct $(1+\varepsilon, +O(1))$-tree covers of size $O(\varepsilon^{-3}\log(\varepsilon^{-1}))$ for every graph in $\mathcal{G}$. A tree T is a *dominating tree* of a graph G if $V(G) \subseteq V(T)$, and $\delta_G(u,v) \le \delta_T(u,v)$ for every pair of vertices (u, v) in G. An (α, β)-*tree cover* for a graph G is a family of dominating trees $\mathcal{T}$, such that for every pair of vertices (u, v) in G, there exists some tree $T \in \mathcal{T}$ with

$$\delta_T(u,v) \le \alpha \cdot \delta_G(u,v) + \beta.$$

The *size* of the tree cover is the cardinality $|\mathcal{T}|$. Tree covers are widely studied for both general metrics and restricted families such as Euclidean/doubling metrics, planar metrics, and minor-free metrics [2, 3, 5, 15, 16, 27], as well as on unit-disk graphs [42]. In particular, if G is planar, then for any constant $\varepsilon > 0$, there is a $(1+\varepsilon, +0)$-tree cover of G with constant size $O(\varepsilon^{-3}\log(\varepsilon^{-1}))$ [15]. We can use our $O(1)$-distortion planar emulator for string graphs to recover a similar result. (Again we focus on string graphs, but the proofs do apply more generally to any suitable class of graphs $\mathcal{G}$ that admits $O(1)$-distortion planar emulators.) Notice that the additive distortion needed for the tree cover result in an absolute constant $O(1)$, instead of $\operatorname{poly}\log n$ as in the result of $(1+\varepsilon)$-mixed-distortion planar emulator.

[2] We note a subtlety about the representation: although the number of crossings in the drawing may be exponential, there exists a compressed representation of the drawing via straight-line programs that has polynomial size [39–41]. In this paper, we do not work with the compressed representation.

[3] A careful reading of the proofs shows that the constant c is at most $2 \cdot (170+1)(24 \cdot 156)^2 = 4793997312$, around 4.8 billion. We have not attempted to optimize this constant.

Theorem 1.3. *Let $\mathcal{G}$ be a class of unweighted graphs that is closed under induced subgraphs, such that every $G \in \mathcal{G}$ has an $O(1)$-distortion planar emulator. Then for any $\varepsilon > 0$, G has a $(1+\varepsilon, +O(1))$-tree cover of size $O(\varepsilon^{-3}\log(\varepsilon^{-1}))$.*

One key application of tree covers is an easy construction of *compact distance oracle*: here one aims to construct a data structure that can quickly return an (approximate) distance when queried on a pair of vertices. If a graph has small tree cover, constructing a distance oracle reduces to lowest-common-ancestor data structures in trees [15, Theorem 1.6].

Corollary 1.4. *Let $\mathcal{G}$ be class of unweighted graphs that is closed under induced subgraphs, such that every $G \in \mathcal{G}$ has an $O(1)$-distortion planar emulator. Let G be an n-vertex graph in $\mathcal{G}$. For any $\varepsilon > 0$, there is a data structure with size $O_\varepsilon(n)$ which, when queried on a pair of vertices (u, v) in G, returns an estimate $\tilde{\delta}(u, v)$ such that $\delta_G(u, v) \le \tilde{\delta}(u, v) \le (1+\varepsilon)\cdot\delta_G(u, v) + O(1)$.*

- *In the word RAM model with word size $\Omega(\log n)$, queries can be answered in $O_\varepsilon(1)$ time.*
- *In the pointer machine model, queries can be answered in $O_\varepsilon(\log\log n)$ time.*

For our next application, an active line of work attempts to embed planar (or minor-free) graphs G into graphs of small *treewidth*, while incurring a small additive distortion of $+\varepsilon\cdot\mathrm{diam}(G)$, for any constant ε [15, 16, 20, 24, 25]. In our language, we seek a $(1, +\varepsilon\,\mathrm{diam}(G))$-distortion planar emulator H such that the treewidth tw of H is minimized. In planar graphs, one can take $\mathrm{tw} = O(\varepsilon^{-4})$ [15]. The proof of [15] uses the small-size tree cover; we show that their construction can be extended to string graphs, with the help of the $O(1)$-distortion planar emulator.

Theorem 1.5. *Let G be an unweighted graph with diameter Δ, such that G has a $O(1)$-distortion planar emulator. For any $\varepsilon > 0$, there is a $(1, +\varepsilon\Delta + O(1))$-distortion emulator G' of G with treewidth $O(\varepsilon^{-4})$.*

Parallel Work. An equivalent result to our first main theorem (Theorem 1.1) has recently been announced independently by James Davies [22]: every unweighted string graph is *quasi-isometric* to an unweighted planar graph. A quasi-isometry implies the existence of an $O(1)$-distortion planar emulator, and one can readily check that our emulator indeed gives a quasi-isometry.[4]

The proof of Davies was announced publicly on arXiv in October 2025, and our proof of Theorem 1.1 was posted on arXiv two days later. Subsequently, we learned through personal communication with Davies that he had obtained his proof earlier, though the result was not published until 2025. There was no communication between the authors and Davies before both works appeared on arXiv.

Our core technique for proving Theorem 1.1 is an adaptation of the *shortcut partition* construction using gridtrees developed in [15], whereas the work by Davies builds the quasi-isometry from first principles. To the best of our knowledge, our second main result (Theorem 1.2) and the applications on tree covers and embedding into bounded-treewidth graphs in the $1+\varepsilon$ regime are completely new.

[4] Specifically, [22] constructs an unweighted planar graph H with $V(H) = V(G)$ such that $\delta_G(u, v)/O(1) \le \delta_H(u, v) \le O(1)\cdot\delta_G(u, v)$. To obtain this result from our Theorem 1.1, apply Steiner point removal to our emulator to produce a weighted graph on $V(G)$, and then ignore the weights; as in Corollary 4.1, this yields an unweighted graph with a similar distortion guarantee to [22].

2 Technical Overview

2.1 O(1)-Distortion Planar Emulator

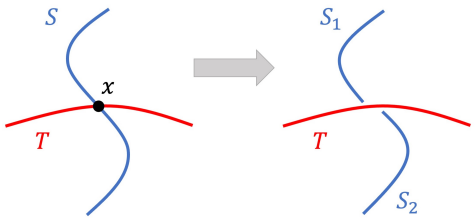


Figure 1: Shattering S at x

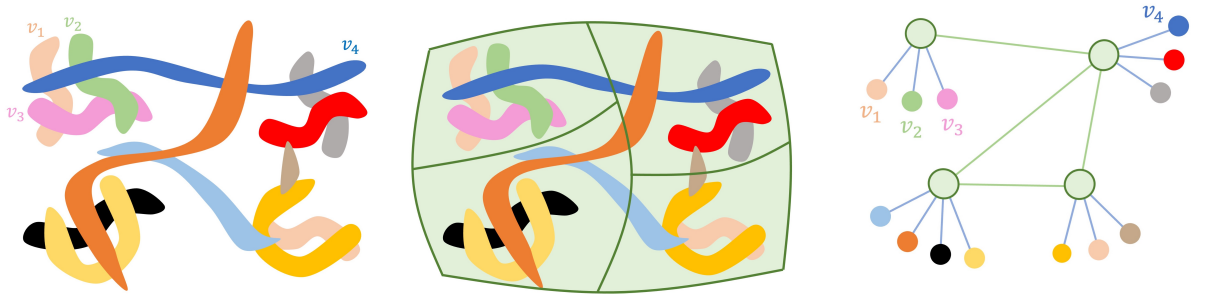
We begin by describing a natural, but *flawed* approach at creating a planar emulator for a string graph $G_{\mathcal{S}}$ on a set of strings $\mathcal{S}$. Let S and T be two strings that cross at some point $x \in \mathbb{R}^2$. The operation of *shattering* of S at x can be thought of as cutting apart S at x to produce two new strings S_1 and S_2, both of which *touch* T but neither of which *crosses* T at x. Given a set of strings $\mathcal{S}$, we could transform their intersection graph $G_{\mathcal{S}}$ into a planar graph by the following procedure: for every pair of strings that cross, shatter one of them to remove the crossing.[5] Each shattering procedure removes a crossing at the cost of changing distances by an O(1) amount — the single string S is turned into two strings S_1 and S_2 which are within distance 2 of each other in the new intersection graph (as witnessed by the path $[S_1, T, S_2]$). After all crossings are removed by the shattering procedure, it is not difficult to see that the resulting intersection graph on the shattered strings $\mathcal{S}'$ is planar. However, distances on the shattered graph $G_{\mathcal{S}'}$ could look very different than on the original $G_{\mathcal{S}}$. When the intersection graph G is a clique (for example), some string S might cross every other string, so S could be shattered into $\Theta(n)$ pieces, and these pieces could be at distance $\Theta(n)$ from each other. In this case, shattering does not seem helpful in producing a planar emulator.

On the other hand, it is easy to find an $O(1)$-planar emulator H when G is a clique: just take H to be a star graph. This gives us the following hope: we could try to remove some crossings using the shattering procedure (only shattering each string $O(1)$ times), and hope that the remaining intersections happen in local "clusters" that can be replaced with stars.

> **Goal.** Given strings $\mathcal{S}$, find a partition of $\mathbb{R}^2$ into connected clusters and shatter all strings in $\mathcal{S}$ by the boundary of the clusters such that (1) each string is only shattered by the boundary of $O(1)$ clusters, and (2) for each cluster C, the (shattered) strings in C are all within $O(1)$ distance of each other.

Given such a partition, it is fairly straightforward to find a planar emulator, by replacing each cluster with a star; see the proof of Theorem 1.1 below for the details, and Figure 2 for an example. Now,

[5] When a string S is shattered into S_1 and S_2, one of the strings S_1 or S_2 is chosen arbitrarily to represent S, and the other string is treated as a Steiner vertex.



(a) A region intersection graph G.

(b) A partition of $\mathbb{R}^2$ into clusters.

(c) A planar emulator for G by contracting all clusters and adding stars.

Figure 2: An illustration of planar emulator construction by contracting clusters.

a key insight is that our stated goal is reminiscent of the notion of *scattering partition* introduced by Filtser for general graphs [23].

> **Scattering partition.** A graph G has an $O(1)$-scattering partition if, for any distance $\Delta > 0$, there is a partition of $V(G)$ into connected clusters such that: (1) any *path* of length Δ intersects only $O(1)$ clusters, and (2) each cluster has diameter $O(\Delta)$.

Our intuition is that a scattering partition with $\Delta = O(1)$ is similar to the partition in our goal. Unfortunately, scattering partitions are only known for some very restricted families of graphs, such as trees, cactus graphs [23], and series-parallel graphs [29]. Recently, Chang *et al.* [15] constructed a very similar (albeit slightly weaker) object—the *shortcut partition*—for *planar graphs* (and later for minor-free graphs [16]). Our main technical contribution is to delicately adapt the planar construction of [15] to (almost) achieve our stated goal. Below, we formally state the partition we aim to construct (Lemma 2.1) and describe how to construct a planar emulator from this partition. We then sketch some intuition for the proof of Lemma 2.1.

2.1.1 Constructing an Emulator from a Clustering. Rather than working directly with string graphs drawn in $\mathbb{R}^2$, we work with region intersection graph on planar graphs, as defined by Lee [34]. Let G be an arbitrary planar graph, which we call the *base graph*. A *region* is a connected subset of vertices in G; these are the analog of strings. Let $\mathcal{R}$ be a set of regions on G. The *region intersection graph*, denoted $\mathrm{RIG}(G, \mathcal{R})$, is the graph with vertex set $\mathcal{R}$, and an edge is added between every two regions R_1 and R_2 that share at least one vertex in G. Clearly every string graph is a region intersection graph on a planar graph, and vice versa. A *cluster* of a graph G is a set of vertices $C \subseteq V(G)$ such that $G[C]$ is connected.

Lemma 2.1. *There are absolute constants $\alpha_{\mathrm{diam}} = 170$ and $\alpha_{\mathrm{hop}} = 3744 \cdot 3744$ such that the following holds. For any planar (base) graph G and any set of regions $\mathcal{R}$ on G, there is a partition of $V(G)$ into disjoint clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ such that*

(1) [Diameter.] *For every cluster $C_i \in \mathcal{C}$, for every pair of regions $R_1, R_2 \in \mathcal{R}$ that intersects C, we have $\delta_{\mathrm{RIG}(G,\mathcal{R})}(R_1, R_2) \le \alpha_{\mathrm{diam}}$.*

(2) [Scattering.] *For every region $R \in \mathcal{R}$ and every two vertices $v_1, v_2 \in R$, there is a path in G between v_1 and v_2 that intersects at most α_{hop} clusters.*

Notice the minor but crucial change in the [scattering] condition, compared to our goal stated earlier in the technical overview: instead of asking each string/region R to intersect only $O(1)$ many clusters, we instead guarantee the existence of *some path* between every pair of vertices in R that intersects $O(1)$ many clusters. (This change is similar to the difference between shortcut partition and scattering partition.) Equipped with Lemma 2.1, we can prove Theorem 1.1 by constructing an emulator H as follows.

Let $\mathcal{C}$ be the set of disjoint clusters from Lemma 2.1. Let $\hat{H}$ be the *cluster graph* obtained from G by contracting every (connected) cluster in $\mathcal{C}$ into a single vertex. More precisely, $\hat{H}$ is the graph with vertex set $\mathcal{C}$, where there is an edge between two clusters $C_1, C_2 \in \mathcal{C}$ if and only if some vertex in C_1 is adjacent to some vertex in C_2 in G. Initialize $H \leftarrow \hat{H}$. For every region $R \in \mathcal{R}$, create a new vertex in H representing R, and add an edge between this vertex and an arbitrary cluster $C \in V(\hat{H})$ that intersects R; this cluster C is called the *representative cluster* of R. See Figure 2. We will consider H to be a weighted graph where each edge has weight $\alpha_{\mathrm{diam}} + 1$. Clearly H is planar: the contracted graph $\hat{H}$ is planar as it is obtained by contractions of the planar base graph G, and attaching new vertices by a single edge to $\hat{H}$ maintains planarity. We claim H is an $O(1)$-distortion planar emulator for $\mathrm{RIG}(G, \mathcal{R})$; we defer the proof to Section 4.1.

Claim 2.2. *H is a $(2 \cdot (\alpha_{\mathrm{diam}} + 1) \cdot \alpha_{\mathrm{hop}})$-distortion planar emulator of $\mathrm{RIG}(G, \mathcal{R})$.*

2.1.2 Constructing the Clustering: A Comparison with [15] and [8]. How do we prove Lemma 2.1? As discussed above, we delicately adapt (and at some point completely deviate from) the shortcut partition construction of [15] in planar graphs.

Shortcut Partition of [15]. We begin by summarizing the partition of [15], which itself builds on the construction of [8] for sparse covers. Let $\Delta > 0$, and let G be a plane graph (i.e. a planar graph G given along with an embedding of G in the plane). We say that the pair (u, v) in G with $\delta_G(u, v) \le \Delta$ is *τ-scattered* by a set of clusters $\mathcal{C}$ if there is a path of length $O(\Delta)$ between u and v in G that intersects at most τ clusters. A *shortcut partition* is (roughly

speaking[6]) a partition of $V(G)$ into clusters $\mathcal{C}$ each of diameter $O(\Delta)$, such that any pair of vertices (u, v) in G with $d_G(u, v) \leq \Delta$ is $O(1)$-scattered by $\mathcal{C}$.

If G consists of a shortest path π (called the *spine*) together with some vertices within Δ distance of π, we call G a *spined supernode* (or just *supernode*, for short). Observe that it is easy to construct a scattering partition in the special case where G is a spined supernode. Indeed, one can first pick a maximal set of vertices $\mathcal{P}$ on π that are $\Theta(\Delta)$ apart, and then partition G into clusters according to a Voronoi partition with respect to $\mathcal{P}$ where each vertex is assigned to the cluster of its closest point in $\mathcal{P}$. It is clear that every cluster has diameter $O(\Delta)$, and (because π is a shortest path) any pair of vertices at distance $O(\Delta)$ in G are $O(1)$-scattered. If G is a general plane graph, [15] first partitions G into subgraphs, each of which is a spined supernode, such that every pair of vertices at distance $O(\Delta)$ is $O(1)$-scattered by the set of spined supernodes. A shortcut partition then follows from partitioning each spined supernode into clusters as described above.

The set of supernodes is constructed recursively: in each iteration [15] assign some of the vertices of G to spined supernodes, then recurse on the subgraph induced by the unassigned vertices. They guarantee that, in each iteration, every pair of vertices at distance $O(\Delta)$ is $O(1)$-scattered *by the supernodes created at the current iteration*. Moreover, they guarantee that in each iteration, all vertices within distance $O(\Delta)$ are assigned to some supernode; using this fact, they prove that any path of length Δ only intersects supernodes created in two iterations, and thus any pair of vertices at distance $O(\Delta)$ is $O(1)$-scattered by the final set of supernodes.

We now describe the [15] construction of supernodes in each iteration. Intuitively, they sweep across the plane graph G from left to right while maintaining a *separating supernode* S; the spine of S has endpoints on the outer face of G, and it separates the left side of G (which has already been processed) from the right side (which has yet to be processed). In a bit more detail: the supernode S is initialized to be the $O(\Delta)$ neighborhood around an arbitrary vertex on the outer face. Then the following process is repeated: add S to the set of spined clusters $\mathcal{C}$; then delete all vertices in S from G; then move S one step forward on the outer face (roughly, let v_1' and v_2' be two vertices adjacent to S on the outer face, and update S to be the supernode comprising the shortest path from v_1' to v_2' in $G - S$ plus an $O(\Delta)$-neighborhood around the path). In this way, S is a "thick path" that acts as a separator as it sweeps across all vertices in the outer face. The separation property of S is used to show that any pair of vertices at distance $O(\Delta)$ is $O(1)$-scattered by the supernodes. Observe that the construction guarantees all vertices on the outer face are assigned to a supernode. After sweeping across the entire graph G, [15] perform an *expansion* step: every vertex within distance $O(\Delta)$ of the outer face is assigned to its closest supernode (that is, each supernode grows, in a Voronoi manner, by distance $O(\Delta)$.

To summarize, the construction of [15] involves four steps[7]: (1) sweep along the outer face to select spined supernodes; (2) expand the supernodes by $O(\Delta)$ distance; (3) recurse on the subgraph induced by unassigned vertices; (4) chop each spined supernode into $O(\Delta)$-diameter clusters.

Adaptation to String Graphs. Our approach to proving Lemma 2.1, in the setting of region intersection graphs, also involves the four major steps of [15]. We would like to sweep across the intersection graph $G_{\mathcal{R}}$ (computing shortest paths in $G_{\mathcal{R}}$, and maintaining a separating supernode S which is a subset of regions $\mathcal{R}$), then expand the supernodes and partition them into clusters, and then recurse on the regions not assigned to any clusters. However, there are significant technical obstacles to this plan, and we must deviate from [15]. Here we highlight three differences, with increasing complexity.

(1) When [15] sweep across the graph and maintain a separating supernode S, they essentially rely on the *Jordan curve theorem* to prove that S separates the already-processed part of the graph from the yet-to-be-processed part: roughly, if S contains a cycle in G, then no vertex inside the cycle is adjacent to a vertex outside the cycle. In region intersection graphs, the Jordan curve theorem does not immediately give a separator: even if S is set of regions in $G_{\mathcal{R}}$ that contains a cycle in G, there may be a region that both contains a vertex of G inside the cycle and outside the cycle. However, any such region must contain at least one vertex of G on the cycle. Thus the *1-neighborhood* of S is a separator for the intersection graph $G_{\mathcal{R}}$. We use this fact to prove that we can maintain a separating supernode while sweeping across $G_{\mathcal{R}}$.

(2) When sweeping across G, [15] delete the vertices in S and then continue sweeping across the remaining part of the graph. This guarantees that the supernodes are disjoint, and so (later on) each one can be independently partitioned into clusters. However, when we sweep across $G_{\mathcal{R}}$, it is *not* sufficient to just delete the regions in S from $\mathcal{R}$ and continue. This is because we eventually want a partition of $V(G)$ into clusters, not a partition of $\mathcal{R}$. Thus, if S_1 and S_2 are supernodes, it is not enough to guarantee that $S_1 \cap S_2 = \emptyset$, that is, there is no region of $\mathcal{R}$ contained in both S_1 and S_2; rather, we need to guarantee that S_1 and S_2 are *vertex disjoint*, no vertex of $V(G)$ is contained in both a region of S_1 and a region of S_2. To achieve this stronger requirement, we *shatter*[8] all regions in $\mathcal{R}$ along the boundary of supernode S, before deleting S and continuing to sweep. We show that (roughly speaking) each region only gets shattered $O(1)$ times. We introduce the *region contact graph* as a variant of the region intersection graph which behaves more nicely under the shatter operation, and use this to bound the distortion incurred by shattering.

(3) Finally, the last issue arises during the expansion step of [15]. This procedures requires computing a *Voronoi partition*: we assign each unassigned vertex to its closest supernode, producing a set of larger, vertex-disjoint, supernodes. The same Voronoi procedure is also used when partitioning each supernode into clusters. Unfortunately, in region intersection graphs, we have no comparable way to produce

[6] Actually, we are stating the closely-related definition of *approximate scattering partition* [16], which is slightly weaker than shortcut partition; this definition suffices for our exposition.

[7] A historical note: The first and fourth steps appeared in [8] in the context of sparse cover, and the second and third were new to [15].

[8] as described at the beginning of the technical overview; see Section 4 for a precise definition

vertex-disjoint clusters: performing a Voronoi partition on the regions $\mathcal{R}$ in $G_{\mathcal{R}}$ produces clusters that do not share regions, but again these clusters may not be vertex-disjoint. To get around this, we need to combine the step of supernode expansion with the step of partitioning each supernode. The combined expand-and-cluster procedure carves out clusters in a delicate order to simultaneously guarantee the bounded diameter and expansion properties.

2.2 $(1+\varepsilon, +\operatorname{poly}\log n)$-Distortion Planar Emulator

A Toy Example. Next we describe our plan to turn an $O(1)$-distortion planar emulator into another planar emulator with $(1+\varepsilon, +\operatorname{polylog} n)$-mixed-distortion guarantee. Like in the previous section, we begin with a toy example to see why such a reduction might even be possible. Let $C = O(1)$ be the constant so that every string graph has a C-distortion planar emulator. Let us imagine a string graph[9] G where every vertex of the graph is in the $O(C)$-neighborhood of a single shortest path π; that is, G is a *spined supernode*. In this case, the *path-straightening* approach of Nguyen, Scott, and Seymour [38] shows that there is a planar emulator for G with purely additive $+O(1)$ distortion.[10] Indeed, let H be a C-distortion planar emulator of G on the same set of vertices. Consider the image of π in H: every edge in π is mapped to a path of length at most C in H. The resulting walk π' may be longer than $\operatorname{len}(\pi)$ by a factor C, but [38] show that one can "straighten" π' into a shortest path with length exactly equal to $\operatorname{len}(\pi)$ by assigning edge weights appropriately. Very loosely speaking:[11] for every edge in π, the image of the edge e in H is a walk with length at most C; we set one edge in this walk to have weight 1 and the rest to have weight 0. Every edge not in the image of π is assigned to have some large weight C^2.

After this reweighting, [38] show that the resulting graph (H, w) has *purely additive* distortion $+\operatorname{poly}(C)$, rather than multiplicative distortion. We sketch the proof: Let u and v be a pair of vertices in G. First, find the vertex u' on π that is closest to u in G. The distance between u and u' is at most $O(C)$ in G, and at most $O(C^2)$ in H because H is a C-distortion planar emulator, and therefore at most $O(C^4)$ in the reweighted graph (H, w). Similarly we find the vertex v' on π that is closest to v in G, so v and v' also has distance at most $O(C^4)$ in (H, w). Finally, consider the shortest path between u' and v' in G. Because the path is effectively straightened in (H, w), there is no multiplicative distortion accumulated when walking along π' in the weighted graph (H, w). This proves $\delta_{(H,w)}(u, v) \le \delta_G(u, v) + \operatorname{poly}(C)$.

Detours Through C-Disjoint Shortest Paths. The argument above shows that it is easy to find additive emulators when G is a spined supernode (i.e., an $O(C)$-neighborhood around some shortest path). What about general string graphs? One idea is to draw inspiration from Section 2.1 and compute a partition of $V(G)$ into disjoint spined supernodes. Suppose we somehow could find a partition such that any shortest path between two vertices u and v intersects only $O(1)$ spined supernodes. In this case we could obtain an emulator with purely additive distortion by straightening each spined supernode independently: walking past each supernode only incurs a $+\operatorname{poly}(C)$ additive distortion, so in total only a constant additive distortion accumulates on the path between u and v. However, requiring that *any* shortest path intersects only $O(1)$ spined supernodes is an incredibly strong requirement. In contrast, the shortcut partition from Section 2.1 essentially says that there is a partition of G into spined supernodes such that a shortest path of length $O(C)$ intersects $O(1)$ spined supernodes; but for constructing an additive emulator, we would need such a guarantee to apply to arbitrarily long paths. [38] show (in different language) that such a partition exists in the special case when G has an $O(1)$-distortion emulator with bounded pathwidth, but in general it is easy to see that an arbitrary string graph G may not admit such a partition, for example when G is a grid graph.[12]

Fortunately, it is often the case that planar graphs (or close-to-planar graphs) have extremely compact distance-sketching tools in the $(1+\varepsilon)$-multiplicative distortion regime, even when there are lower bounds against exact distances sketches. We could still hope for a partition of $V(G)$ into spined supernodes such that, for any pair of vertices u and v, there is a $(1+\varepsilon)$-approximate shortest path between u and v that intersects few supernodes. We draw inspiration from a recent distance-approximating minor construction of Chang and Conroy [14] for planar graphs: we interpret their result as constructing a collection of paths $\boldsymbol{O}$ in a planar graph G that $(1+\varepsilon)$-approximate all pairwise distances in G, such that the paths have few intersections with each other (in some amortized sense). Specifically: for every pair of vertices (u, v) in G, there is a path P that is the concatenation of $\operatorname{poly}\log n$ subpaths P_j in $\boldsymbol{O}$, such that the sum of lengths of all P_j is at most a $(1+\varepsilon)$ multiplicative factor longer than the original distance $\delta_G(u, v)$. We aim to generalize the techniques of [14] to find our desired partition of a string graph G into spined supernodes. There are two key challenges we need to overcome.

(1) Can we adapt the techniques of [14] to find a suitable collection of paths $\boldsymbol{O}$ when G is a *string graph*, not a planar graph?
(2) Can we obtain a collection of paths $\boldsymbol{O}$ that are all (pairwise) at distance $\Omega(C)$ from each other, rather than just sparsely intersecting? If the paths intersect, even with few intersections as in [14], we cannot straighten them independently.

As we describe below, we overcome these two challenges to prove the following lemma.

Lemma 2.3. *Let G be an unweighted n-vertex graph that has a C-distortion planar emulator H with $V(H) = V(G)$. For any small constant $\varepsilon_0 > 0$, there is a set of paths $\boldsymbol{O}$ in G such that:*

[9] There is nothing special about the choice of G being a string graph; as mentioned, our reduction from multiplicative distortion to $(1+\varepsilon)$-mixed-distortion works for any graph class that admits $O(1)$-distortion planar emulators.
[10] Actually, [38] get something slightly different than emulator: they allow distances to *contract* by a small amount, but note that it is easy to remove this contraction by adding some Steiner vertices.
[11] There are technical difficulties because π' may be a walk, not a path. In actuality, not all vertices on the walk π' remain in the straightened path; see the full version of the paper for details.
[12] In this case, every column is a shortest path, so it may intersects only $O(1)$ spined supernodes. But every row is also a shortest path — and if every column intersects only $O(1)$ supernodes, it is not too difficult to see that some row must intersects polynomially many supernodes.

- [$\Theta(C)$-disjointness.] *For any paths $P_1, P_2 \in \mathcal{O}$, we have $\delta_G(P_1, P_2) > 2C$.*
- [Low-hop.] *For any vertices $u, v \in V(G)$, there is a path P between u and v (not necessarily in $\mathcal{O}$): either $\text{len}(P) < O(C\varepsilon_0^{-4} \cdot \log^{17} n)$ (that is, P is short), or $\text{len}(P) \le (1+\varepsilon_0) \cdot \delta_G(u,v)$, such that P can be written as the concatenation*

$$P = Q_1 \circ P_1 \circ Q_2 \circ P_2 \circ \cdots \circ Q_\ell$$

where each P_j is a subpath of some path in $\mathcal{O}$, the sum of lengths of all Q_j is at most $O(\varepsilon_0 \cdot \text{len}(P))$, and $\ell \le \varepsilon_0^{-3} \cdot \log^{18} n$.

Challenge 1: Shortest-path Separators on String Graphs. The construction of [14] relies heavily on shortest-path separators in planar graphs, and we will also need this tool. However, we have an immediate problem: there are no known shortest-path separators construction for string graphs, even if we reinterpret separators to be 1-neighborhood separators of a shortest path. Such separators are only known for unit-disk graphs [28, 43]. Our first challenge is to construct a shortest-path separator for arbitrary string graphs.

Lemma 2.4. *Let G be an unweighted n-vertex graph that has a C-distortion planar emulator H with $V(H) = V(G)$. There is a set of $O(\log n)$ vertex-disjoint shortest paths $\mathcal{P}$ in G such that, letting $S \subseteq V(G)$ denote the set of vertices within distance $2C$ (in G) of a vertex in $\mathcal{P}$, every connected component of $G \setminus S$ has at most $n/2$ vertices.*

Our proof combines the standard path separator construction on planar graphs by Lipton-Tarjan [36] with our $O(1)$-distortion planar emulator from Theorem 1.1. First we compute a single-source shortest-path tree T on string graph G, and consider the planar emulator H of G. Map the tree T into H to obtain a subgraph T^H (which is not necessarily a tree). We then make use of a decycling trick to extract a spanning tree from T^H. The following combinatorial lemma, to our best knowledge, appeared first in [18] in which they dubbed it the *Steiner path aggregation problem*. Consider the set of n paths $\mathcal{P}$ from every vertex to root in T and their corresponding images $\mathcal{P}'$ in H. Given such paths in H (where the paths are allows to self-intersect) that together covers T^H, there is a tree T' that is a spanning subgraph of T^H, such that any leaf-to-root path in T' is a concatenation of $O(\log n)$ paths from the set of paths $\mathcal{P}'$. By [36], some set of $O(1)$ paths in T' form a balanced separator S of H. The corresponding preimage of S is a set of $O(\log n)$ shortest paths in G, and one can show that the C-neighborhood of these $O(\log n)$ paths is a balanced separator of G.

With the separator hierarchy for the string graph G, we are able to carry out the first steps of the construction of [14] for computing a sparsely-intersecting set of paths $\mathcal{O}$. First, for every piece R in the separator hierarchy and every scale $i \in [\log n]$, we compute *scale-i portals* on the internal separator S of R, with the property that consecutive scale-i portals along S are at distance $\Theta(\varepsilon \cdot 2^i)$. The portals are chosen so that any shortest path between (u, v) passing through S can be rerouted through some portal on S at a cost of a multiplicative $1+\varepsilon$ distortion. Using the portals, we then define a set of *canonical paths* [1, 14] between the portals as follows. A pair of portals p and p' are *relevant* to a piece R in the separator hierarchy if p is on the internal separator of R and p' is on the internal separator of some ancestor piece R' of R. A *canonical path* at scale i and region R is shortest path between a pair of scale-i portals that are relevant to R and are within distance $O(2^i)$ of each other; see the full version of the paper for a formal definition. One important property of canonical paths is that, between any pair of vertices u and v, there exists a path P with length at most $(1+\varepsilon) \cdot \delta_G(u,v)$, which is the concatenation of poly $\log n$ canonical paths. Moreover, [14] show that one can slightly "wiggle" the canonical paths so that they intersect sparsely with each other.

Challenge 2: How to Ensure the Spined Supernodes are $\mathcal{O}(C)$-Disjoint. With the analog of separator hierarchy for string graphs, we could obtain something similar to the set $\mathcal{O}$ of sparsely intersecting paths from [14]. However, here we come to another, more difficult, obstacle: Recall that we have the ability to straighten *one* path with its $O(C)$-neighborhood. However if two paths intersect (or even have overlapping $O(C)$-neighborhoods) the corresponding images of the two paths in H could intersect, and as a result we may have two different values we want to reweight an edge of H to. This would be problematic.

Our main technical contribution in the second part of the paper is to devise a novel detour strategy to resolve this issue and obtain a set $\mathcal{O}$ of paths that are at least distance $\Omega(C)$ apart, proving Lemma 2.3. The construction of $\mathcal{O}$ is a simple *carving algorithm*. First compute the separator hierarchy and canonical paths, as described above. Order the canonical paths according to a natural ordering $\preceq$ based on the region and scale of the canonical paths[13], and iterate through each path in $\mathcal{O}$ based on such ordering. Every time before a new path π is inserted in $\mathcal{O}$, we "erase" an $\Theta(C)$-neighborhood around π from the existing paths in $\mathcal{O}$. If some path π' breaks into multiple parts because of the erasure, remove π' from $\mathcal{O}$ and insert the components of the broken paths back into $\mathcal{O}$. Clearly the resulting collection of subpaths in $\mathcal{O}$ are at distance more than C away from each other.[14] To construct our mixed-distortion emulator, we reweight the emulator H to straighten each path in $\mathcal{O}$. It remains to prove the distortion bound.

Bound on Distortion via Detouring. The simplicity of the carving algorithm comes with a cost: the analysis of the distortion bound is the most complicated part of the proof. We outline some of the technical issues together with a sketch of how we overcome them. We want argue that, between any pair of vertices u and v, there is a $(1+\varepsilon)$-approximate shortest path in G that walks along only $\tilde{O}(1)$ paths in $\mathcal{O}$. (This is because, whenever we walk along paths in $\mathcal{O}$ we incur no distortion in H, and whenever we switch between paths in $\mathcal{O}$ we incur some additive distortion $+\text{poly}(C)$ to "jump" the gap of width $O(C)$ between paths. Thus it is crucial to bound the number of jumps we make.) By construction of canonical paths, it suffices to consider the case when u and v are the endpoints of some canonical path π.

Idea 1: Detouring π. We would like to claim that π gets broken into $\tilde{O}(1)$ paths in $\mathcal{O}$, during the carving algorithm; then we could easily find a path between u and v that makes only $\tilde{O}(1)$ jumps between

[13] We order the paths by region containment—that is, if π is a canonical path between portals in region R, and π' is a canonical path between portals in some ancestor region R', we say $\pi \preceq \pi'$. Within a region, we order paths by length—that is, if π is a path between two scale-i portals and π' is a path between two scale-i' portals with $i \le i'$, we say $\pi \preceq \pi'$. The same ordering appears in [14].

[14] In actuality, the $O(C)$-neighborhood is taken to be in a subgraph of G that depends on the path π chosen; nevertheless we can still argue that the distance between subpaths in $\mathcal{O}$ is more than C away in G.

paths in O. Unfortunately, this claim is simply not true. Even if π is only close (ie, within distance C) to a single other canonical path π' with $\pi \prec \pi'$, it may be the case that π is shattered into arbitrarily many subpaths; see Figure 3. However, in this case we can find a short path P between u and v that makes few jumps with the following *detour* procedure: walk along π until the first point where π and π' are within distance C, then jump onto π' and walk until the last point where π and π' are within distance C, and then jump back onto π. Assuming the entire path π' appears in O, we have found a path P between u and v that touches only two paths in O, and P is only longer than $\delta_G(u,v)$ by an additive term $+2C$.

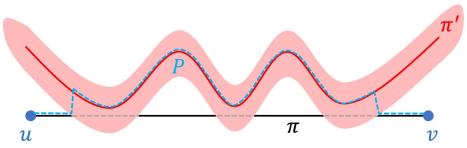


Figure 3: Canonical paths π and π', with $\pi \prec \pi'$. Erasing the C-neighborhood of π' shatters π into many subpaths. The "detoured" path P jumps from π to π' and back to π'.

Idea 2: Controlling the number of detours. The argument above suggests that we would want to prove that there are only $\tilde{O}(1)$ paths π' with $\pi \prec \pi'$ that are close to π. We say that π' *threatens* π if $\pi \prec \pi'$ and π' is within distance C of π. If we could prove that π is threatened by $\tilde{O}(1)$ canonical paths, we could hope to detour π along each such threatening π', ending up with a short path P between u and v that makes $\tilde{O}(1)$ jumps. There are two issues here. First, it is just not true that π is threatened by $\tilde{O}(1)$ canonical paths. Second, even if we had a bound on the number of threateners of π, there is the issue that π' itself could be threatened by some other canonical pair: that is, even if we detour along π', the path π' may not belong to O. We now describe how we circumvent these two issues.

- *Issue: There are many threateners of π.* On one hand, it is relatively straightforward to prove that there are only $\tilde{O}(1)$ paths π' that threaten π and satisfy $\mathrm{len}(\pi') > \varepsilon \cdot \mathrm{len}(\pi)$; that is, there aren't many *long* paths π' that threaten π. (This will follow from the definition of canonical pair and the fact that scale-i portals are spaced out from each other.) On the other hand, however, there could be many *very short* paths π' that threaten π.
  We overcome this issue as follows. To begin with, we compute a path P between the endpoints of π by performing detours along sufficiently long threatening paths π', where $\mathrm{len}(\pi') > \varepsilon \cdot \mathrm{len}(\pi)$. This means that we perform only $\tilde{O}(1)$ detours. Of course, at this point there may still be short paths π' that threaten π, meaning P could still make arbitrarily many jumps between paths in O. However, in this case we argue that P is within distance $\varepsilon \cdot \mathrm{len}(\pi)$ of a separator path of some piece R' in the separator hierarchy, where R' is a proper ancestor of the piece R that contains π. Whenever our path P is within distance $O(\varepsilon \cdot \mathrm{len}(\pi))$ of a separator path $\pi_{R'}$ of some ancestor R', we immediately "jump" to $\pi_{R'}$ and make a detour along $\pi_{R'}$; one can show that $\pi_{R'}$ is itself a canonical path that is not threatened by any short path π' with an endpoint on $\pi_{R'}$. Thus, detouring along separator paths effectively allows us to ignore the short paths that threaten π, while only increasing the length of P by a small amount.
- *Issue: Recursive threateners of π'.* Our next issue is that, even after we detour our path P along a threatening path π', we have no guarantee that π' itself appears in O. We may have to recursively detour π' along some other path π'' that threatens π'. To overcome this issue, we would like to claim that if π'' threatens π', then π'' also threatens π, under some relaxed notation of "threatening". We say a canonical path π' *β-threatens* a path π if, roughly speaking[15], some endpoint of π' is within distance $O(\mathrm{len}(\pi')) + \beta$ of some endpoint of π. Clearly, if π is threatened by π', then π is 0-threatened by π'. As discussed in the previous section, there are only $\tilde{O}(1)$ long paths π' that 0-threaten π, and in fact a similar argument shows that there are only $\tilde{O}(1)$ long paths π' that β-threaten π for any constant $\beta = O(1)$. Moreover, the β-threatening relationship is almost transitive: if π is β-threatened by π', and π' is threatened by another path π'', then π is $(\beta + C)$-threatened by π''. That is, if we recursively detour π along θ different paths π', all these paths π' are $(\theta \cdot C)$-threateners of π.
  To bound the number of recursive detours we make, we want to bound the number of β-threateners of π. But, for what value of β should we bound the number of β-threateners? If we find that there are $f(\beta)$ many β-threateners of π, then we could make as many $f(\beta)$ detours, meaning that subsequent paths we detour against are $f(\beta) \cdot C$-threateners of π. This "self-referential" problem between the number of detours and the radius we can claim π threatens seems circular and may escalate into an uncontrollable number of segments in the final detour path P. Our one final trick is to establish the existence of a θ whose value is well-defined and non-circular, such that the number of (long) canonical paths that θ-threaten π is at most θ. The value of θ ends ups to be around $O(\varepsilon^{-3} \log^3 n)$, which is sufficient for an $(1+\varepsilon, +\mathrm{poly}\log n)$ approximation.

We remark that our detour procedure is somewhat reminiscent of the detours performed by [14]. However, there are substantial differences. The detours in [14] were directly performed to construct O; the detours were only performed against separator paths; and a main challenge was to detour in a way that deals with $O(\log n)$ scales. In contrast, our detours are performed only in the analysis (not in the construction of O); we detour along other canonical paths; and a key challenge is to cope with recursively detouring along canonical paths via the existence of the seemingly self-referential θ.

3 Preliminaries

In this paper, we consider unweighted graph G. We let $E(G)$ denote the edges of G and $V(G)$ denote the vertices of G. For a set of vertices $S \subseteq V(G)$, we will frequently consider the induced subgraph $G[S]$. We will use $\mathrm{diam}(G)$ to denote the diameter of G. In addition, we

[15] see the full version of the paper for the precise definition

work with paths P in graph G. For any two vertices a and b in P we will use $P[a:b]$ to denote the subpath of P between vertices a and b. We will sometimes concatenate two paths P_1 and P_2, and denote the concatenation of the paths by $P_1 \circ P_2$. Note that this may result in a walk (and not a path) that visits some vertices and edges multiple times. The logarithms in this paper are all of base 2.

In this extended abstract, we only provide a sketch of our constructions and proofs for the $O(1)$-distortion planar emulator. See the full version of the paper for the complete proofs.

4 O(1)-Distortion Planar Emulator

4.1 Proof of $O(1)$-Distortion Emulator

Here we prove Claim 2.2: the planar emulator H constructed in Section 2.1.1 has $O(1)$-distortion.

Claim 2.2. *H is a $(2 \cdot (\alpha_{\text{diam}} + 1) \cdot \alpha_{\text{hop}})$-distortion planar emulator of $\text{RIG}(G, \mathcal{R})$.*

Proof. Recall that the graph H is defined by contracting every connected cluster $C \in \mathcal{C}$ from Lemma 2.1, connecting adjacent clusters together, and adding edges from vertices representing regions to arbitrary representative clusters. The edges in H have weight $\alpha_{\text{diam}} + 1$. See Figure 2 for an illustration.

First we prove an upper bound on distances in H. It suffices to show that for every edge (R_a, R_b) in the region intersection graph $\text{RIG}(G, \mathcal{R})$, we have $\delta_H(R_a, R_b) \le 2 \cdot (\alpha_{\text{diam}} + 1) \cdot \alpha_{\text{hop}}$. To this end, fix one such edge (R_a, R_b) in $\text{RIG}(G, \mathcal{R})$. There is some vertex $v \in R_a \cap R_b$. Now, let C_a (resp. C_b) be the representative cluster of R_a (resp. R_b). The [scattering] property of Lemma 2.1 implies that there is a path in G between some vertex in C_a to v which intersects at most α_{hop} many clusters, and similarly a path in G from v to some vertex in C_b which intersects at most α_{hop} many clusters: concatenating these paths provides in the cluster graph from C_a to C_b that intersects at most $2\alpha_{\text{hop}} - 1$ clusters. We conclude that there is a path in the contracted graph H between C_a and C_b with at most $2\alpha_{\text{hop}} - 2$ edges, and thus a path from R_a to R_b using at most $2\alpha_{\text{hop}}$ many edges. As each edge has weight $\alpha_{\text{diam}} + 1$, we have that $\delta_H(R_a, R_b) \le 2 \cdot (\alpha_{\text{diam}} + 1) \cdot \alpha_{\text{hop}}$ as desired.

Now we prove a lower bound on distances in H: for any $R_a, R_b \in \mathcal{R}$, we claim $\delta_{\text{RIG}(G,\mathcal{R})}(R_a, R_b) \le \delta_H(R_a, R_b)$. Let us consider two fixed regions $R_a, R_b \in \mathcal{R}$ with $P_H := [R_a, C_1, C_2, \ldots, C_\ell, R_b]$ as a shortest path in H between R_a and R_b; note that every vertex other than the endpoints represents some cluster $C_i \in \mathcal{C}$. Observe that P_H consists of $\ell + 1$ edges of weight $\alpha_{\text{diam}} + 1$, so P_H is a path of length $(\ell + 1) \cdot (\alpha_{\text{diam}} + 1)$. Now for each $(C_i, C_{i+1}) \in E(H)$ there exists a region R'_i and R_{i+1} that are adjacent in $\text{RIG}(G, \mathcal{R})$ such that R'_i intersects C_i and R_{i+1} intersects C_{i+1}. Next, for each i, consider the path in $\mathcal{R}$ from R_i to R'_i. This path is of length at most α_{diam} by the [diameter] property of Lemma 2.1. Consider the walk P_{RIG} in $\text{RIG}(G, \mathcal{R})$ that is the concatenation of the shortest path from R_a to R'_1, followed by the edge from R'_i to R_{i+1} and the shortest path between R_{i+1} to R'_{i+1} for $i = 1, \ldots, \ell - 1$, followed by the edge from R'_ℓ the shortest path from R'_ℓ to R_b. The walk P_{RIG} consists of $\ell + 1$ shortest paths that lie within some cluster, and $\ell - 1$ paths between clusters, which has length at most $(\ell + 1) \cdot \alpha_{\text{diam}} + (\ell - 1)$. As the walk P_{RIG} is at least as long as the shortest path between R_a and R_b, we have that

$$\begin{aligned}\delta_{\text{RIG}(G,\mathcal{R})}(R_a, R_b) &\le (\ell + 1) \cdot \alpha_{\text{diam}} + (\ell - 1) \\ &< (\ell + 1) \cdot (\alpha_{\text{diam}} + 1) \\ &= \delta_H(R_a, R_b),\end{aligned}$$

which proves the lower bound on distances as desired. □

The planar emulator H could have many more vertices than $\text{RIG}(G, \mathcal{R})$. To fix this issue, we apply Steiner point removal [16]. With a small bit of work, one can show that every edge in the planar emulator has $O(1)$ weight (even after Steiner point removal); thus we can obtain an *unweighted* planar emulator for $\text{RIG}(G, \mathcal{R})$ with few Steiner vertices.

Corollary 4.1. *There is an unweighted planar graph H' with $O(n)$ vertices, such that H' is a $O(1)$-distortion emulator for the n-vertex graph $\text{RIG}(G, \mathcal{R})$.*

Proof. Let H be (edge-weighted) $O(1)$-distortion planar emulator of the intersection graph $G_{\mathcal{R}}$, provided by Claim 2.2. It is immediate from the construction that every edge has weight $O(1)$. We define the *terminals* to be the vertices $V(\text{RIG}(G, \mathcal{R})) = \mathcal{R}$, and define the *Steiner vertices* to be $V(H) \setminus V(\text{RIG}(G, \mathcal{R})) = \mathcal{C}$; recall that the terminals represent regions in $\mathcal{R}$, and the Steiner vertices represent clusters from the clustering $\mathcal{C}$ of Lemma 2.1. Note that H could contain many more Steiner vertices, so we now apply the Steiner point removal procedure of [16]. The following claim summarizes the result of applying [16] to the planar graph H:

Claim 4.2 (Theorem 1.2 and Claim 2.7 of [16])**.** *There is a partition of $V(H)$ into connected clusters $\mathcal{S}$ that each contain exactly one terminal vertex $R \in \mathcal{R}$. Letting H' be the weighted cluster graph obtained by contracting every cluster $C_R \in \mathcal{S}$ into a single terminal vertex $R \in \mathcal{R}$, and setting the weight of an edge (R_1, R_2) in H' to be $\delta_H(R_1, R_2)$, we have*

$$\delta_H(R_a, R_b) \le \delta_{H'}(R_a, R_b) \le O(1) \cdot \delta_H(R_a, R_b)$$

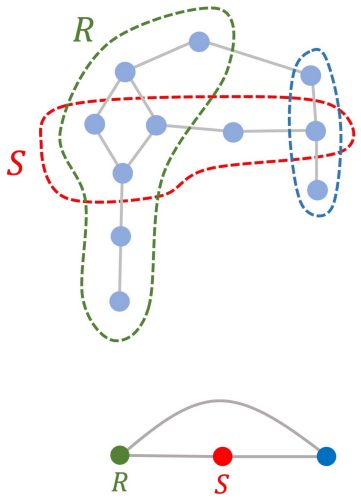
for all pairs of terminals (R_a, R_b) in $\mathcal{R}$. Moreover, every Steiner vertex $C \in \mathcal{C}$ of H belongs to the cluster (in $\mathcal{S}$) containing some terminal $R \in \mathcal{R}$ satisfying $\delta_H(C, R) \le O(1) \cdot \delta_H(C, \mathcal{R})$.

Clearly H' is a $O(1)$-distortion planar emulator for $\text{RIG}(G, \mathcal{R})$. We claim that every edge of H' has weight $O(1)$. Assuming this claim, we can simply subdivide every edge of H' to obtain an unweighted $O(1)$-distortion planar emulator for G with $O(n)$ vertices.

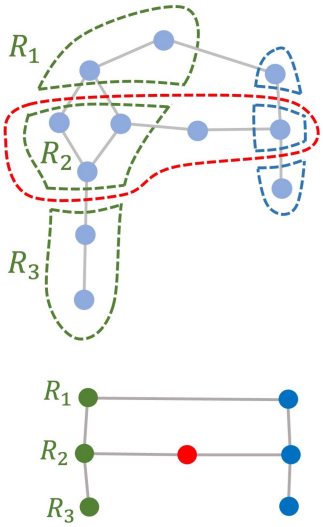
To this end, let (R_1, R_2) be an edge in H' between two terminals. By construction of H', there is some vertex C_1 (resp. C_2) in $V(H)$ which is assigned to the cluster of R_1 (resp. R_2) during Steiner point removal, such that (R_1, R_2) is an edge in H. (Assume WLOG that C_1 and C_2 are Steiner vertices, ie they represent clusters in the clustering $\mathcal{C}$.) The weight of (R_1, R_2) is set to be $\delta_H(R_1, R_2)$. Triangle inequality implies that $\delta_H(R_1, R_2) \le \delta_H(R_1, C_1) + 1 + \delta_H(C_2, R_2)$; by Claim 4.2 we have

$$\delta_H(R_1, R_2) \le O(1) \cdot (\delta_H(C_1, \mathcal{R}) + 1 + \delta_H(C_2, \mathcal{R})).$$

But now observe that, for any Steiner vertex $C \in \mathcal{C}$, we have $\delta_H(C, \mathcal{R}) \le O(1)$. Indeed, let $R \in \mathcal{R}$ be a region that intersects cluster $C \in \mathcal{C}$. Let C_R be the representative cluster of R. By Lemma 2.1, there is a path in H between v and C_R with $O(1)$ hops; as every edge in H has $O(1)$ weight, we have $\delta_H(C, C_R) \le O(1)$. By construction



(a) A set of regions $\mathcal{R}$ on graph G, and the corresponding contact graph $G_{\mathcal{R}}$.



(b) The regions $\mathcal{R}'$ produced by $\textsc{Shatter}(\mathcal{R} - \mathcal{S})$, and the resulting contact graph $G_{\mathcal{R}'}$. The subregions of region R are R_1, R_2, R_3.

Figure 4: An illustration of regions and shattered regions.

$\delta_H(C_R, R) \le O(1)$. As $R \in \mathcal{R}$, we have $\delta_H(C, \mathcal{R}) \le O(1)$ as desired. We conclude that $\delta_H(R_1, R_2) \le O(1)$. □

4.2 Towards Proving Lemma 2.1

This remainder of Section 4 is dedicated to the proof of Lemma 2.1. We fix a plane graph G. All regions are subsets of G.

Shattering. We will frequently require the operation of "shattering" a region; i.e. breaking it apart into smaller components. If R is a region in G and $S \subseteq V(G)$ is a set of vertices of G, then the process of *shattering R by the boundary of S*—denoted $\textsc{Shatter}(R - \mathcal{S})$—returns the set of connected components of $G[R \setminus S]$ together with the connected components of $G[R \cap S]$; see Figure 4. In other words, shattering R by S returns a set of maximal connected regions which do not "cross" the boundary of S and whose union is R. Suppose $R \in \mathcal{R}$ is shattered by S into connected components $R_1, R_2, \ldots, R_k \in \textsc{Shatter}(R - \mathcal{S})$, we say that each R_i is a *subregion* of R. The subregion property is transitive: if R_i' is a subregion of R_i, we also say that R_i' is a subregion of R. If $\mathcal{R}$ is a collection of regions, then $\textsc{Shatter}(\mathcal{R} - \mathcal{S})$ denotes $\bigcup \{\textsc{Shatter}(R - \mathcal{S}) : R \in \mathcal{R}\}$. We say that a set of regions $\mathcal{R}'$ is a *shattering* of $\mathcal{R}$ if it is a subset of some regions produced by an iterative sequence of $\textsc{Shatter}$ operations run on $\mathcal{R}$ (possibly with different S).

Region Contact Graph. Rather than the region intersection graph, we will instead work with the notion of the *region contact graph*—denoted as $G_{\mathcal{R}}$—which is the graph with vertex set $\mathcal{R}$, where there is an edge between regions R_1 and R_2 if and only if there are vertices $v_1 \in R_1$ and $v_2 \in R_2$ such that either $v_1 = v_2$ *or* v_1 *is adjacent to* v_2 *in* G; see Figure 4.

The reason we work with contact graph $G_{\mathcal{R}}$ instead of intersection graph $\text{RIG}(G, \mathcal{R})$ is that a key subroutine in our algorithms is to "shatter" a region R into multiple pieces, say $R = R_1 \sqcup R_2$. In the contact graph, R_1 and R_2 are adjacent, whereas in the intersection graph they are at infinite distance. In other words, distances in the contact graph behave more nicely under the shatter operation.

Note that the class of region contact graphs is precisely equal to the class of region intersection graphs on planar graphs. Given a region intersection graph $\text{RIG}(G, \mathcal{R})$, one can subdivide every edge of G to obtain a new graph G' and a new set of regions $\mathcal{R}'$ where a subdivided vertex is in a region if and only if both end points were in that region. It is clear that the intersection graph $\text{RIG}(G, \mathcal{R})$ is precisely the same graph as the contact graph $G'_{\mathcal{R}'}$.[16] In the rest of this section, we work only with region contact graphs. To justify this, we now describe how to prove Lemma 2.1 even though we only work with contact graphs. For any cluster C in graph G, we define the *$\mathcal{R}$-diameter* of C to be the max, over all pairs of regions $R_1, R_2 \in \mathcal{R}$ that intersect C, of $\delta_{G_{\mathcal{R}}}(R_1, R_2)$.

Observation 4.3. *To prove Lemma 2.1, it suffices to prove the [diameter] bound with respect to the contact graph $G_{\mathcal{R}}$ instead of the intersection graph* $\text{RIG}(G, \mathcal{R})$. *That is, it suffices partition $V(G)$ into clusters $\mathcal{C}$ such that every cluster has $\mathcal{R}$-diameter at most α_{diam} and $\mathcal{C}$ satisfies the [scattering] property of Lemma 2.1.*

Proof. Suppose we are given a planar graph G and set of regions $\mathcal{R}$, and we want to prove Lemma 2.1. First, as described above, we subdivide every edge of G to compute the graph G' and set of regions $\mathcal{R}'$ such that $\text{RIG}(G, \mathcal{R})$ is the same graph as $G'_{\mathcal{R}'}$. We then find a partition of $V(G')$ into clusters $\mathcal{C}'$ that each have $\mathcal{R}'$-diameter at most α_{diam} (with respect to the contact graph $G'_{\mathcal{R}'}$), such that $\mathcal{C}'$ has the [scattering] property. Because G' is a subdivision of G, every cluster C' in $\mathcal{C}'$ can be pulled back naturally to a connected

[16] The converse is also true, every region contact graph is a region intersection graph, though we will not need this fact. Given a region contact graph G, we can again subdivide every edge, and expand each region outwards by one to get the same graph as a region intersection graph.

cluster C on G, by discarding all Steiner vertices $V(G') \setminus V(G)$. If a cluster C' has $\mathcal{R}$-diameter at most α_{diam}, then C satisfies the [diameter] property with respect to $\text{RIG}(G, \mathcal{R})$, because distances in $\text{RIG}(G, \mathcal{R})$ and $G'_{\mathcal{R}'}$ are the same. Finally, the clustering $\mathcal{C}$ satisfies the [scattering] property, because any path P' in G' can be pulled back to a path P in G with $P \subseteq P'$. □

Terminology for Contact Graph. The *support* of a set of regions $\mathcal{R}$, denoted $V(\mathcal{R})$, is the set of vertices in G that are in some region in $\mathcal{R}$. Given a contact graph $G_{\mathcal{R}}$, we assume without loss of generality that $V(\mathcal{R}) = V(G)$, since deleting vertices of G that are not part of any region does not change the contact graph.

Fix any set of regions $\mathcal{R}$. When the base graph G is clear from context, we abuse notation and write $\delta_{\mathcal{R}}(\cdot, \cdot)$ to be the distance function of the contact graph $G_{\mathcal{R}}$; that is, for every pair of regions $R_1, R_2 \in \mathcal{R}$, we define $\delta_{\mathcal{R}}(R_1, R_2)$ to be the length of the shortest path between R_1 and R_2 in $G_{\mathcal{R}}$. In this section we frequently consider an induced subgraph H of G, and a set of regions $\mathcal{R}'$ with $V(\mathcal{R}') = V(H)$, i.e. that the support of the regions of $\mathcal{R}'$ is in $V(H)$. Note that there is no difference between the graph $G_{\mathcal{R}'}$ and $H_{\mathcal{R}'}$, because H is an induced subgraph of G. As such, the notation $\delta_{\mathcal{R}}(\cdot, \cdot)$ is unambiguous, even if we do not explicitly specify whether $\mathcal{R}$ should be understood to be a set of regions on H or on G.

Clustering the Outer Regions. Given a plane graph H (a planar graph along with an embedding of H) and a set of regions $\mathcal{R}$, we say a region $R \in \mathcal{R}$ is an *outer region* if R contains at least one vertex on the outer face of H. We find the clustering of Lemma 2.1 in stages: we first assign (some of) the vertices of G to a (partial) set of clusters $\mathcal{C} = \{C_1, ..., C_k\}$ which includes all vertices that are in outer regions, and then recurse on the subgraph induced by the unassigned vertices. (A similar strategy was used by [15] to find a shortcut partition for planar graphs; they also create a partial set of clusters that cover the outer face and then recurse on the unassigned vertices.) Here we state the lemma with the properties we need for our partial clustering. For notational convenience, we will denote the union of the vertices of that are within our partial cluster as $V(\mathcal{C}) := \bigcup_i C_i$.

Lemma 4.4. *There are absolute constants $\alpha \leq 170$ and $\beta \leq 3744$ such that the following holds. Let G be a plane graph. There is a procedure* ClusterOuter$(H, \mathcal{R})$ *that takes as input a connected plane graph H that is an induced subgraph of G (and inherits the planar embedding of G) and a set of regions $\mathcal{R}$ whose support is $V(H)$, and returns a partial partition of $V(H)$ into a set of clusters $\mathcal{C}$ such that:*

1. [Outer-clustered.] *For every outer region $R \in \mathcal{R}$, every vertex in R is assigned to some cluster of $\mathcal{C}$. Moreover, the subgraph $H[V(\mathcal{C})]$, which contains all vertices assigned to clusters, is connected.*
2. [Diameter.] *Every cluster $C \in \mathcal{C}$ has $\mathcal{R}$-diameter at most α.*
3. [Scattering.] *For every region $R \in \mathcal{R}$ and any two vertices $v_a, v_b \in R$, there is a path Γ in G between v_a and v_b that can be written as the concatenation of β paths*

$$\Gamma = \Gamma_1 \circ \Gamma_2 \circ \ldots \circ \Gamma_\beta$$

where each subpath Γ_i is either (1) contained entirely in some cluster $C \in \mathcal{C}$ or (2) consists of unassigned vertices in R.

In the full version of the paper, we prove Lemma 4.4 and then explain how to apply ClusterOuter recursively to find a clustering of the entire graph G.

Acknowledgments

Hsien-Chih Chang and Jonathan Conroy are supported by the U.S. National Science Foundation CAREER Award under the Grant No. CCF-2443017.

References

[1] Ittai Abraham, Shiri Chechik, and Cyril Gavoille. 2012. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 1199–1218. https://doi.org/10.1145/2213977.2214084

[2] Sunil Arya, Gautam Das, David M. Mount, Jeffrey S Salowe, and Michiel Smid. 1995. Euclidean spanners: short, thin, and lanky. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*. 489–498. https://doi.org/10.1145/225058.225191

[3] Baruch Awerbuch and David Peleg. 1992. Routing with polynomial communication-space trade-off. *SIAM Journal on Discrete Mathematics* 5, 2 (1992), 151–162. https://doi.org/10.1137/0405013

[4] Sayan Bandyapadhyay, Fedor V. Fomin, and Tanmay Inamdar. 2023. Coresets for Clustering in Geometric Intersection Graphs. In *39th International Symposium on Computational Geometry (SoCG 2023)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 10–1. https://doi.org/10.4230/LIPIcs.SoCG.2023.10

[5] Yair Bartal, Nova Fandina, and Ofer Neiman. 2019. Covering metric spaces by few trees. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, Patras, Greece, July 9-12, 2019*, Vol. 132. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 20:1–20:16. https://doi.org/10.4230/LIPIcs.ICALP.2019.20

[6] Ahmad Biniaz. 2020. Plane hop spanners for unit disk graphs: Simpler and better. *Computational Geometry* 89 (2020), 101622. https://doi.org/10.1016/j.comgeo.2020.101622

[7] Karl Bringmann, Sándor Kisfaludi-Bak, Marvin Künnemann, André Nusser, and Zahra Parsaeian. 2022. Towards Sub-Quadratic Diameter Computation in Geometric Intersection Graphs. In *38th International Symposium on Computational Geometry*. Schloss Dagstuhl, 1–16. https://doi.org/10.4230/LIPIcs.SoCG.2022.21

[8] Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. 2014. Sparse Covers for Planar Graphs and Graphs That Exclude a Fixed Minor. *Algorithmica* 69, 3 (jul 2014), 658–684. https://doi.org/10.1007/s00453-013-9757-4

[9] Nicolas Catusse, Victor Chepoi, and Yann Vaxès. 2010. Planar hop spanners for unit disk graphs. In *International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*. Springer, 16–30. https://doi.org/10.1007/978-3-642-16988-5_2

[10] Timothy M. Chan, Hsien-Chih Chang, Jie Gao, Sándor Kisfaludi-Bak, Hung Le, and Da Wei Zheng. 2025. Truly subquadratic time algorithms for diameter and related problems in graphs of bounded VC-dimension. In *2025 IEEE 66th Annual Symposium on Foundations of Computer Science—FOCS 2025*. IEEE Comput. Soc. Press, Los Alamitos, CA, 2728–2765. https://doi.org/10.1109/FOCS63196.2025.00140

[11] Timothy M. Chan and Zhengcheng Huang. 2023. Constant-hop spanners for more geometric intersection graphs, with even smaller size. In *Proc. 39th International Symposium on Computational Geometry (SoCG 2023)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.SoCG.2023.23

[12] Timothy M. Chan and Dimitrios Skrepetos. 2019. All-Pairs Shortest Paths in Geometric Intersection Graphs. *Journal of Computational Geometry* 10, 1 (2019), 27–41. https://doi.org/10.20382/jocg.v10i1a2

[13] Timothy M. Chan and Dimitrios Skrepetos. 2019. Approximate shortest paths and distance oracles in weighted unit-disk graphs. *Journal of Computational Geometry* 10, 2 (2019), 3–20. https://doi.org/10.20382/jocg.v10i2a2

[14] Hsien-Chih Chang and Jonathan Conroy. 2025. Distance Approximating Minors for Planar and Minor-Free Graphs. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025*. IEEE, 734–754. https://doi.org/10.1109/FOCS63196.2025.00039

[15] Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenkovic, Shay Solomon, and Cuong Than. 2023. Covering planar metrics (and beyond): O(1) trees suffice. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2231–2261. https://doi.org/10.1109/FOCS57990.2023.00139

[16] Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenković, Shay Solomon, and Cuong Than. 2024. Shortcut partitions in minor-free graphs: Steiner point removal, distance oracles, tree covers, and more. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 5300–5331. https://doi.org/10.1137/1.9781611977912.191

[17] Hsien-Chih Chang, Jie Gao, and Hung Le. 2024. Computing Diameter+2 in Truly-Subquadratic Time for Unit-Disk Graphs. In *40th International Symposium on Computational Geometry (SoCG 2024)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 38–1. https://doi.org/10.4230/LIPIcs.SoCG.2024.38

[18] Da Qi Chen, Daniel Hathcock, D. Ellis Hershkowitz, and R. Ravi. 2025. The Steiner Path Aggregation Problem. *Inform. Process. Lett.* (2025), 106608. https://doi.org/10.1016/j.ipl.2025.106608

[19] V. Chepoi, F. F. Dragan, I. Newman, Y. Rabinovich, and Y. Vaxès. 2012. Constant Approximation Algorithms for Embedding Graph Metrics into Trees and Outerplanar Graphs. *Discrete & Computational Geometry* 47, 1 (jan 2012), 187–214. https://doi.org/10.1007/s00454-011-9386-0

[20] Vincent Cohen-Addad, Arnold Filtser, Philip N. Klein, and Hung Le. 2020. On light spanners, low-treewidth embeddings and efficient traversing in minor-free graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 589–600. https://doi.org/10.1109/FOCS46700.2020.00061

[21] Jonathan Conroy and Csaba Tóth. 2023. Hop-spanners for geometric intersection graphs. *Journal of Computational Geometry* 14, 2 (Dec. 2023), 26–64. https://doi.org/10.20382/jocg.v14i2a3

[22] James Davies. 2025. String graphs are quasi-isometric to planar graphs. *arXiv preprint arXiv:2510.19602* (2025). https://arxiv.org/abs/2510.19602

[23] Arnold Filtser. 2024. Scattering and sparse partitions, and their applications. *ACM Transactions on Algorithms* 20, 4 (2024), 1–42. https://doi.org/10.1145/3672562

[24] Arnold Filtser and Hung Le. 2022. Low treewidth embeddings of planar and minor-free metrics. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1081–1092. https://doi.org/10.1109/FOCS54457.2022.00105

[25] Eli Fox-Epstein, Philip N. Klein, and Aaron Schild. 2019. Embedding planar graphs into low-treewidth graphs with applications to efficient approximation schemes for metric problems. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1069–1088. https://doi.org/10.1137/1.9781611975482.66

[26] Zachary Friggstad, Mohsen Rezapour, Mohammad R. Salavatipour, and Hao Sun. 2025. A QPTAS for Facility Location on Unit Disk Graphs. In *19th International Symposium on Algorithms and Data Structures (WADS 2025) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 349)*, Pat Morin and Eunjin Oh (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 27:1–27:18. https://doi.org/10.4230/LIPIcs.WADS.2025.27

[27] Anupam Gupta, Amit Kumar, and Rajeev Rastogi. 2005. Traveling with a pez dispenser (or, routing issues in MPLS). *SIAM J. Comput.* 34, 2 (2005), 453–474. https://doi.org/10.1137/S0097539702409927

[28] Elfarouk Harb, Zhengcheng Huang, and Da Wei Zheng. 2024. Shortest Path Separators in Unit Disk Graphs. In *32nd Annual European Symposium on Algorithms (ESA 2024) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 308)*, Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 66:1–66:14. https://doi.org/10.4230/LIPIcs.ESA.2024.66

[29] D. Ellis Hershkowitz and Jason Li. 2022. $O(1)$ Steiner Point Removal in Series-Parallel Graphs. In *30th Annual European Symposium on Algorithms (ESA 2022)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 66–1. https://doi.org/10.4230/LIPIcs.ESA.2022.66

[30] Alice Kerr. 2023. Tree approximation in quasi-trees. *Groups Geom. Dyn.* 17, 4 (2023), 1193–1233. https://doi.org/10.4171/ggd/733

[31] Jan Kratochvíl and Jiří Matoušek. 1991. String graphs requiring exponential representations. *Journal of Combinatorial Theory, Series B* 53, 1 (1991), 1–4. https://doi.org/10.1016/0095-8956(91)90050-T

[32] Neeraj Kumar, Daniel Lokshtanov, Saket Saurabh, and Subhash Suri. 2021. A constant factor approximation for navigating through connected obstacles in the plane. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 822–839. https://doi.org/10.1137/1.9781611976465.52

[33] Neeraj Kumar, Daniel Lokshtanov, Saket Saurabh, Subhash Suri, and Jie Xue. 2022. Point separation and obstacle removal by finding and hitting odd cycles. In *38th International Symposium on Computational Geometry, SoCG 2022, Berlin, Germany, June 7-10, 2022*, Vol. 224. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 52:1–52:14. https://doi.org/10.4230/LIPIcs.SoCG.2022.52

[34] James R. Lee. 2017. Separators in Region Intersection Graphs. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 1–1. https://doi.org/10.4230/LIPIcs.ITCS.2017.1

[35] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. 2002. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3. IEEE, 1268–1277. https://doi.org/10.1109/INFCOM.2002.1019377

[36] Richard J Lipton and Robert Endre Tarjan. 1979. A separator theorem for planar graphs. *SIAM J. Appl. Math.* 36, 2 (1979), 177–189. https://doi.org/10.1137/0136016

[37] Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, Jie Xue, and Meirav Zehavi. 2024. A 1.9999-approximation algorithm for vertex cover on string graphs. In *40th International Symposium on Computational Geometry (SoCG 2024)*, Vol. 293. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 72. https://doi.org/10.4230/LIPIcs.SoCG.2024.72

[38] Tung Nguyen, Alex Scott, and Paul Seymour. 2025. Asymptotic structure. II. Path-width and additive quasi-isometry. *arXiv preprint arXiv:2509.09031* (2025). https://arxiv.org/abs/2509.09031

[39] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. 2003. Recognizing String Graphs in NP. *J. Comput. System Sci.* 67, 2 (sep 2003), 365–380. https://doi.org/10.1016/S0022-0000(03)00045-X

[40] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. 2007. Spiralling and Folding: The Topological View. In *Proceedings of the 19th Annual Canadian Conference on Computational Geometry, CCCG 2007, August 20-22, 2007, Carleton University, Ottawa, Canada*. Carleton University, Ottawa, Canada, 73–76. http://cccg.ca/proceedings/2007/03b3.pdf

[41] Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. 2011. Spiraling and Folding: The Word View. *Algorithmica* 60, 3 (jul 2011), 609–626. https://doi.org/10.1007/s00453-009-9362-8

[42] Puheng Weng. 2025. *Tree Covers of Unit Disk Graphs*. Master's thesis. Rutgers University - School of Graduate Studies. https://doi.org/10.7282/t3-nb7c-3w28

[43] Chenyu Yan, Yang Xiang, and Feodor F. Dragan. 2012. Compact and low delay routing labeling scheme for unit disk graphs. *Computational Geometry* 45, 7 (2012), 305–325. https://doi.org/10.1016/j.comgeo.2012.01.015