Near-Linear ε-Emulators for Planar Graphs[*]

Hsien-Chih Chang[†] Robert Krauthgamer[‡] Zihan Tan[§]

June 21, 2022

Abstract

We study vertex sparsification for distances, in the setting of planar graphs with distortion: Given a planar graph G (with edge weights) and a subset of k terminal vertices, the goal is to construct an *ε-emulator*, which is a small planar graph G' that contains the terminals and preserves the distances between the terminals up to factor $1+\varepsilon$.

We construct the first ε-emulators for planar graphs of near-linear size $\tilde{O}(k/\varepsilon^{O(1)})$. In terms of k, this is a dramatic improvement over the previous quadratic upper bound of Cheung, Goranci and Henzinger, and breaks below known quadratic lower bounds for exact emulators (the case when $\varepsilon = 0$). Moreover, our emulators can be computed in (near-)linear time, which lead to fast $(1+\varepsilon)$-approximation algorithms for basic optimization problems on planar graphs, including multiple-source shortest paths, minimum (s,t)-cut, graph diameter, and offline dynamic distace oracle.

---

[*]This is the full version of the paper "Almost-Linear ε-Emulators for Planar Graphs" that appears in STOC 2022. As indicated in the title change, the main difference is that the emulator size's dependence on k is improved here from $k^{1+o(1)}$ to $k \log^{O(1)} k$.

[†]Department of Computer Science, Dartmouth College. Email: `hsien-chih.chang@dartmouth.edu`. Supported in part by the startup fund at Dartmouth College.

[‡]Weizmann Institute of Science. Work partially supported by ONR Award N00014-18-1-2364, the Israel Science Foundation grant #1086/18, the Weizmann Data Science Research Center, and a Minerva Foundation grant. Email: `robert.krauthgamer@weizmann.ac.il`.

[§]Computer Science Department, University of Chicago. Email: `zihantan@uchicago.edu`. Supported in part by NSF grant CCF-2006464.

Contents

1 Introduction

Graph compression describes a paradigm of transforming a large graph G to a smaller graph G' that preserves, perhaps approximately, certain graph features such as distances or cut values. The algorithmic utility of graph compression is apparent — the compressed graph G' may be computed as a preprocessing step, reducing computational resources for subsequent processing and queries. This general paradigm covers famous examples like spanners, Gomory-Hu trees, and cut/flow/spectral edge-sparsifiers, in which case G' has the same vertex set as G, but fewer edges. Sometimes the compression is non-graphical and comprises of a small data structure instead of a graph G'; famous examples are distance oracles and distance labeling.

We study another well-known genre of compression, called *vertex sparsification*, whose goal is for G' to have a small vertex set. In this setting, the input graph G has a collection of k designated vertices T, called the *terminals*. The compressed graph G' should contain, besides the terminals in T, a small number of vertices and preserve a certain feature among the terminals. Specifically, we are interested in preserving the distances between terminals up to multiplicative factor $\alpha \geq 1$ in an edge-weighted graph (where the weights are interpreted as lengths). Formally, given a graph G with terminals $T \subseteq V(G)$, an *emulator* for G with *distortion* $\alpha \geq 1$ is a graph G' that contains the terminals, i.e., $T \subseteq V(G')$, satisfying

$$\forall x, y \in T, \quad \mathsf{dist}_G(x, y) \leq \mathsf{dist}_{G'}(x, y) \leq \alpha \cdot \mathsf{dist}_G(x, y), \tag{1}$$

where dist_G denotes the shortest-path distance in G (and similarly for G'). In the important case when $\alpha = 1+\varepsilon = e^{\Theta(\varepsilon)}$ for $0 \leq \varepsilon \leq 1$, we simply say G' is an *ε-emulator*.[1] Notice that G' need not be a subgraph or a minor of G (in such two settings G' is known as a *spanner* and a *distance-approximating minor*).

We focus on the case where G is known to be planar, and thus require also G' to be planar (which excludes the trivial solution of a complete graph on T). This requirement is natural and also important for applications, where fast algorithms for planar graphs can be run on G' instead of on G. Such a requirement that G' has structural similarity to G is usually formalized by assuming that both G and G' belong to $\mathcal{F}$ for a fixed graph family $\mathcal{F}$ (e.g., all planar graphs). If $\mathcal{F}$ is a minor-closed family, one can further impose the stronger requirement that G' is a minor of G, and this clearly implies that G' is in $\mathcal{F}$.

Vertex sparsifiers commonly exhibit a tradeoff between accuracy and size, which in our case of an emulator G', are the distortion α and the number of vertices of G'. Let us briefly overview the known bounds for planar graphs. At one extreme of this tradeoff we have the "exact" case, where distortion is fixed to $\alpha = 1$ and we wish to bound the (worst-case) size of the emulator G' [CGH16, CGMW18, GHP20]. For planar graphs, the known size bounds are $O(k^4)$ [KNZ14] and $\Omega(k^2)$ [KZ12, CO20].[2] At the other extreme, we fix the emulator size to $|V(G')| = k$, i.e., zero non-terminals, and we wish to bound the (worst-case) distortion α [BG08, CXKR06, KKN15, Che18, FKT19]. For planar graphs, the known distortion bounds are $O(\log k)$ [Fil18] and lower bound 2 [Gup01].

Our primary interest is in minimizing the size-bound when the distortion α is $1 + \varepsilon$, i.e., ε-emulators, a fascinating sweet spot of the tradeoff. The minimal loss in accuracy is a boon for applications, but it is usually challenging as one has to control the distortion over iterations or recursion. For planar graphs, the known size bounds for a distance-approximating minor are $\tilde{O}((k/\varepsilon)^2)$ [CGH16] and $\Omega(k/\varepsilon)$ [KNZ14]. Improving the upper bound from quadratic to linear in k is an outstanding question that offers a bypass to the aforementioned $\Omega(k^2)$ lower bound for exact emulators ($\alpha = 1$). In fact, no subquadratic-size emulators for planar graphs are known to exist even when we allow the emulators to be arbitrary graphs, except for when the input is unweighted [CGMW18] or for trivial cases like trees.

[1] Our definition in Section 2 differs slightly (allowing two-sided errors), affecting our results only in some hidden constants.

[2] For fixed distortion $\alpha = 1$, every graph G in fact admits a minor of size $O(k^4)$ [KNZ14], but for some planar graphs (specifically grids) every minor [KNZ14] or just planar emulator [KZ12, CO20] must have $\Omega(k^2)$ vertices.

Notation. Throughout the paper, we consider undirected graphs with non-negative edge weights, and denote $n = |V(G)|$ and $k = |T|$. A *plane graph* refers to a planar graph together with a specific embedding in the plane. We suppress poly-logarithmic terms by writing $\tilde{O}(t) = t \cdot \operatorname{poly}\log t$, and multiplicative factors that depend on ε by writing $O_\varepsilon(t) = O(f(\varepsilon) \cdot t)$. We write $\log^* t$ for the iterated logarithm of t.

1.1 Main Result

We design the first ε-emulators for planar graphs that have near-linear size; furthermore, these emulators can be computed in near-linear time. These two efficiency parameters can be extremely useful, and we indeed present a few applications in Section 1.2.

Theorem 1.1. *For every n-vertex planar graph G with k terminals and parameter $0 < \varepsilon < 1$, there is a planar ε-emulator graph G' of size $|V(G')| = \tilde{O}(k/\varepsilon^{O(1)})$. Furthermore, such an emulator can be computed deterministically in time $\tilde{O}(n/\varepsilon^{O(1)})$.*

The result dramatically improves over the previous $\tilde{O}((k/\varepsilon)^2)$ upper bound of Cheung, Goranci and Henzinger [CGH16]. Moreover, it breaks below the aforementioned lower bound $\Omega(k^2)$ for exact emulators ($\alpha = 1$) [KZ12, KNZ14, CO20]. Unsurprisingly, our result is unlikely to extend to all graphs, because for some (bipartite) graphs, every minor with fixed distortion $\alpha < 2$ must have $\Omega(k^2)$ vertices [CGH16]. See Section 1.1 for comparison to prior work.

Distortion	Size (lower)	Size (upper)	Requirement	Reference
1	$\Omega(k^2)$		planar	[KZ12, CO20]
1		$O(k^4)$	minor	[KNZ14]
$1+\varepsilon$	$\Omega(k/\varepsilon)$		minor	[KNZ14]
$1+\varepsilon$		$\tilde{O}((k/\varepsilon)^2)$	minor	[CGH16]
$1+\varepsilon$		$\tilde{O}(k/\operatorname{poly}\varepsilon)$	planar	Theorem 1.1
$O(\log k)$		k	minor	[Fil18]

Table 1. Distance emulators for planar graphs.

1.2 Algorithmic Applications

We present a few applications of our emulators to the design of fast $(1+\varepsilon)$-approximation algorithms for standard optimization problems on planar graphs.

Our first application is to construct an approximate version of the multiple-source shortest paths data structure, called *ε-MSSP*: Preprocess a plane graph G and a set of terminals T on the outerface of G, so as to quickly answer distance queries between terminal pairs within $(1+\varepsilon)$-approximation. The preprocessing time of our data structure is $O_\varepsilon(n)$, which for any fixed $\varepsilon > 0$ is faster than Klein's $O(n \log n)$-time algorithm [Kle05] for the exact setting when $\varepsilon = 0$. Both algorithms have the same query time $O(\log n)$.

Theorem 1.2. *Given a parameter $0 < \varepsilon < 1$, an n-vertex plane graph G with the range of edge weights bounded by $n^{O(1)}$,[3] and a set of terminals T all lying on the boundary of G with $|T| \le O(n/\log^C n)$ for some large enough constant C, one can preprocess an ε-MSSP data structure on G with respect to T in time $O_\varepsilon(n)$, that answers queries in time $O(\log n)$.*

[3] Our algorithm can also handle general weights with a slightly slower $O_\varepsilon(n \operatorname{poly}(\log^* n))$ preprocessing time.

Our second application is an $O_\varepsilon(n)$-time algorithm to compute $(1+\varepsilon)$-approximate minimum (s,t)-cut in planar graphs, which for fixed $\varepsilon > 0$ is faster than the $O(n \log\log n)$-time exact algorithm by Italiano, Nussbaum, Sankowski, and Wulff-Nilsen [INSW11].

Theorem 1.3. *Given an n-vertex planar graph G with two distinguished vertices $s, t \in V(G)$ and a parameter $0 < \varepsilon < 1$, computing $(1+\varepsilon)$-approximate minimum (s,t)-cut in G takes $O_\varepsilon(n)$ time.*

Our third application is an $O_\varepsilon(n \log n)$-time algorithm to compute a $(1+\varepsilon)$-approximate diameter in planar graphs, which for fixed $0 < \varepsilon < 1$ is faster than the $O(n \log^2 n + \varepsilon^{-5} n \log n)$-time algorithm of Chan and Skrepetos [CS19] (which itself improves over Weimann and Yuster [WY16]).

Theorem 1.4. *Given an n-vertex planar graph G and a parameter $0 < \varepsilon < 1$, one can compute a $(1+\varepsilon)$-approximation to its diameter in time $O_\varepsilon(n \log n)$.*

Finally, one important open problems in the field of dynamic algorithms is the existence of efficient $(1+\varepsilon)$-approximate distance oracle on planar graphs. Abboud and Dahlgaard [AD16] provided an $\Omega(n^{1/2-o(1)})$ lower bound on the query and update time for such oracles in the exact setting. Recently, Chen *et al.* [CGH+20] showed that if one can efficiently construct a $(1+\varepsilon)$-*distance-approximating minor* of size $\tilde{O}(k)$ for a planar graph with n nodes and k terminals in $O(n\,\mathrm{poly}(\log n, \varepsilon^{-1}))$ time, then there is an offline dynamic $(1+\varepsilon)$-approximate distance oracle with $O(\mathrm{poly}\log n)$ query and update time.

Here we show that while our ε-emulator is not strictly a $(1+\varepsilon)$-distance-approximating minor, the same distance oracle can still be constructed. This demonstrates that an efficient $(1+\varepsilon)$-approximate distance oracle on planar graphs exists.

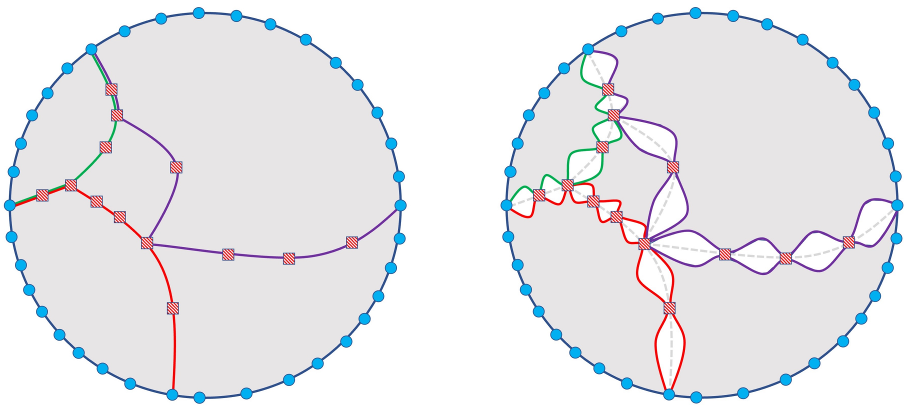
Theorem 1.5. *There is an offline dynamic $(1+\varepsilon)$-approximate distance oracle for any planar graph of size n with $O(\mathrm{poly}\log n)$ query and update time.*

1.3 Technical Contributions

A central technical contribution of this paper is to carry out a *spread reduction* for the all-terminal-pairs shortest path problem when the input graph can be embedded in the plane and the terminals all lie on the outerface; the *spread* is defined to be the ratio between the largest and the smallest distances between terminals. Spread reduction is a crucial preprocessing step for many optimization problems, particularly in Euclidean spaces or on planar graphs [SA12, BG13, KKN15, CFS19, **?**], that replaces an instance with a large spread with one or multiple instances with a bounded spread. In many cases, one can reduce the spread to be at most polynomial in the input size. However, we are not aware of previous work that achieves such a reduction in our context, where many pairs of distances have to be preserved all at once. In fact, even after considerable work we only managed to reduce the spread to be sub-exponential.

We now provide a bird-eye's view of our emulator construction. The emulator problem on plane graphs with an arbitrary set of terminals can be reduced to the same problem on plane graphs, but with the strong restriction that all the terminals lies on a constant number of faces, known as *holes* (cf. Section 5), using a separator decomposition that splits the number of vertices and terminals evenly; such a decomposition (called the *r-division*) can be computed efficiently [Fre87, KMS13]. From there we can further slice the graph open into another plane graph with all the terminals on a single face, which without loss of generality we assume to be the outerface. We refer to it as a *one-hole instance*.

To construct an emulator for a one-hole instance G we adapt a recursive *split-and-combine* strategy (cf. Section 3). We will attempt to split the input instance into multiple one-hole instances along some shortest paths that distribute the terminals evenly (cf. Lemma 3.3). Every time we slice the graph G open along a shortest path P, we compute a small collection of vertices on P called the *portals*, that approximately preserve the distances from terminals in G to the vertices on P. The portals are duplicated during the slicing along P and added to the terminal set (i.e., become terminals) at each piece incident to P, to ensure that further processing will (approximately) preserve their distances as well. We emphasize that the naive idea of placing portals at equally-spaced points along P is not sufficient, as some terminals in G might be arbitrarily close to P. Instead, we place portals at exponentially-increasing intervals from both ends of P. After splitting the original instance into small enough pieces by recursively slicing along shortest paths and computing the portals, we compute exact emulators for each piece using any of the polynomial-size construction [KNZ14, CO20]. Next we glue these small emulators back along the paths by identifying multiple copies of the same portal into one vertex. See Figure 1.



(a) A one-hole instance, a set of paths (shown in red, green and purple curves), and portals (shown as red boxes). Slicing the instance open along these paths gives us smaller pieces.

(b) The one-hole instance obtained from gluing together the emulators for the small pieces at the portals (shown as red boxes).

Figure 1. Illustration of the split-and-combine process for a one-hole instance.

Let U be the set of terminals in the current piece, and let $r := |U|$. We need the portals to be dense enough so that only a small error term, of the form $r^{-\delta}$ (meaning that the distortion increases multiplicatively by $1 + r^{-\delta}$) will be added to the distortion of the emulator after the gluing, as this will eventually guarantee (through more details like the stopping condition of the recursion) that the final distortion is $1 + \varepsilon$ and the final emulator size has polynomial dependency on ε^{-1}. At the same time, the number of portals cannot be too large, as they are added to the terminal set, causing the number of terminals per piece to go down slowly and creating too many pieces, and in the end the size of the combined emulator might be too big. It turns out that the sweet spot is to take roughly $L_r := r/\log^2 r$ portals. Calculations show that in such case the portals preserve distances up to an additive error term $\log \Phi / L_r$, where Φ is the *spread* of the terminal distances (cf. Claim 4.4). When $\Phi \leq 2^{r^{0.9}}$, we will get the polynomially-small $\tilde{O}(r^{-0.1})$ error term needed for the gluing (cf. Section 4.3). However, even when the original input has a polynomial spread to start with, in general we cannot control the spread of all the pieces occurring during the split-and-combine process, because portals are added to the terminal sets. Therefore a new idea is needed.

When $\Phi > 2^{r^{0.9}}$, we need to tackle the spread directly. We perform a *hierarchical clustering* of the terminals (cf. Section 4.4). At each level i, we connect two clusters of terminals from the previous level $i-1$ using an edge if their distance is at most r^{2i}; then we group each connected component into a single cluster. The key to the spread reduction is the idea of *expanding clusters*. A cluster S is *expanding* if its parent cluster $\hat{S}$ is at least $\sim e^{r^{-0.7}}$-factor bigger. Intuitively, if all clusters are expanding, then the number of levels in the hierarchical clustering must be at most $r^{0.7}$, and therefore the spread must be at most sub-exponential. So in the high-spread case some non-expanding cluster must exist.

- If such non-expanding cluster S is of moderate size (that is, in between $r/5$ and $4r/5$) (cf. Section 4.4.1), we construct a collection of *non-crossing* shortest paths between terminals in S (non-crossing means that no two paths with endpoint pairs (s_1, s_2) and (t_1, t_2) have their endpoints in an interleaving order (s_1, t_1, s_2, t_2) on the outerface) in which no two paths intersect except at their endpoints. Again compute portals on the paths from every terminal in $\hat{S} \setminus S$, but now using ε_r-covers [Tho04] for $\varepsilon_r := r^{-0.1}$, and split along the paths to create sub-instances. Because the cluster is non-expanding and has moderate size, the number of terminals in $\hat{S} \setminus S$ is at most $(e^{r^{-0.7}} - 1)|S| \le r^{0.3}$, and thus the number of portals is $O(r^{0.3}/\varepsilon_r) \le O(r^{0.4})$, which is a gentle enough increase in the number of terminals. The hard part is to argue that the portals created are sufficient for the recombined instance to be an emulator. This can be done by observing that terminal pairs among $U \setminus \hat{S}$ are far apart, and similarly when one terminal is from S and the other is from $U \setminus \hat{S}$; hence only terminal pairs involving $\hat{S} \setminus S$ have to be dealt with using properties of ε_r-covers (cf. Claim 4.9).
- If there are no non-expanding clusters with moderate size (cf. Section 4.4.2), we find a non-expanding cluster $\tilde{S}$ of lowest level that contains most of the terminals, and construct a collection of non-crossing shortest paths between terminals in $\tilde{S}$ like the previous case. However this time, after computing the $r^{-0.1}$-covers and splitting along the paths, there might be one instance containing too many terminals. In this case, we find *every* non-expanding cluster S of *maximal level*; such clusters must all lie within $\tilde{O}(r^{0.7})$ levels from $\tilde{S}$ because we cannot have nested expanding clusters for $\tilde{O}(r^{0.7})$ consecutive levels. The Monge property guarantees that the shortest paths generated by the union of these maximal-level non-expanding clusters must be non-crossing because all such clusters are disjoint (cf. Observation 4.7). Now if we split the graph based on the path set generated, each resulting instance either has moderate size, or must have small spread, and we safely fall back to the earlier cases.

Applications. A widely adopted pipeline in designing efficient algorithms for distance-related optimization problems on planar graphs in recent years consists of the following steps:

1. Decompose the input planar graph into small pieces each of size at most r with a small number of boundary vertices and $O(1)$ holes, called an *r-division* (see Frederickson [Fre87] and Klein-Mozes-Sommer [KMS13];
2. Process each piece so that all-pairs shortest paths between boundary vertices within a piece can be extracted efficiently by the *multiple-source shortest paths* algorithm for planar graphs (Klein [Kle05]);
3. Further process each piece into a *compact data structure* that supports efficient min-weight-edge queries and updates (SMAWK [AKM+87], Fakcharoenphol and Rao [FR06]);
4. Compute shortest paths in the original graph in a problem-specific fashion, now with each piece replaced with the compact data structure, using a *modified Dijkstra algorithm* (Fakcharoenphol and Rao [FR06]).

The conceptual role of our planar emulators is an alternative to Step 3. The reason for the development of the aforementioned machinery and complex algorithms is to get around the size lower bound in representing the all-pairs distances for the pieces. The benefit of replacing the data structure with a single planar emulator is that the whole graph stays planar. One can then simply replace Step 4 with the standard Dijkstra algorithm (or even better, with the $O(n)$-time algorithm for planar graphs by

Henzinger *et al.* [HKRS97]). More importantly, one can *recurse* on the resulting graph when appropriate, and compress the graph further and further with small additive errors slowly accumulated (cf. Section 5.3). This allows us to construct near-linear-size ε-emulator in $O_\varepsilon(n \operatorname{poly}\log^* n)$ time and even $O_\varepsilon(n)$ time using a precomputed look-up table for pieces that are tiny compared to n when the spread of the input graph is bounded by a polynomial, which can easily be achieved by standard spread reduction techniques for many optimization problems.

1.4 Related Work

In addition to emulators, there are other lines of research on graph compression preserving distance information. Among them the most studied objects are *spanners* and *preservers* (when the sparsifier is required to be a subgraph of the input graph) and *distance oracles* (a data structure that reports exact or approximate distances between pairs of vertices). We refer the reader to the excellent survey [ABS+20].

There are also rich lines of works for constructing vertex sparsifiers that preserve cut/flow values (known as *cut/flow sparsifiers*) exactly [HKNR98, CSWZ00, KR13, KR14, KPZ17, GHP20, KR20] or approximately [Moi09, CLLM10, Chu12, AGK14, EGK+14, MM16, GR16, GRST21].

2 Preliminaries

All logarithms are to the base of 2. All graphs are simple and undirected. Let G be a connected graph. A vertex $v \in V(G)$ is called a *cut vertex* of G if the graph $G \setminus \{v\}$ is disconnected. The cut vertices of a plane graph G can be computed in time $O(|V(G)| + |E(G)|)$. Let G be a graph with an edge-weight function $w\colon E(G) \to \mathbb{R}_+$. The weight of a path P is defined as $w(P) := \sum_{e \in E(P)} w(e)$. The shortest-path distance between two vertices u and v is denoted by $\mathsf{dist}_G(u,v)$. For a subset S of vertices in G, we define $\mathsf{diam}_G(S) := \max_{u,u' \in S} \mathsf{dist}_G(u,u')$. For a pair of disjoint subsets of vertices (S,S') in G, we define $\mathsf{dist}_G(S,S') := \min_{u \in S, u' \in S'} \mathsf{dist}_G(u,u')$.

Emulators. Throughout, we consider graph G equipped with a special set of vertices T, called *terminals*. We refer to the pair (G,T) as an *instance*. Let (G,T) and (H,T) be a pair of instances with the same set of terminals, and let $\varepsilon \in [0,1]$. We say that H is an *ε-emulator* for G with respect to T, or equivalently, instance (H,T) is an ε-emulator for instance (G,T) if

$$\forall x,y \in T, \quad e^{-\varepsilon} \cdot \mathsf{dist}_G(x,y) \le \mathsf{dist}_H(x,y) \le e^{\varepsilon} \cdot \mathsf{dist}_G(x,y). \tag{2}$$

Throughout, we use Equation (2) as the definition of an ε-emulator instead of Equation (1); but since we restrict our attention to $\varepsilon < 1$, the two definitions are equivalent up to scaling ε by a constant factor. By definition, if (H,T) is an ε-emulator for (G,T), then (G,T) is also an ε-emulator for (H,T). Moreover, if (G,T) is an ε-emulator for (G',T) and (G',T) is an ε'-emulator for (G'',T), then (G,T) is an $(\varepsilon+\varepsilon')$-emulator for (G'',T).

Most instance (G,T) considered in this paper are *planar instances* where graph G is a connected plane graph. We say that a planar instance (G,T) is an *h-hole instance* for an integer $h > 0$ if the terminals lie on at most h faces in the embedding of G. The faces incident to some terminals are called *holes*. Notice that in a one-hole instance (G,T), we can safely assume all the terminals in T lie on the outerface G. By definition, a 0-emulator preserves distances exactly, i.e., $\mathsf{dist}_G(x,y) = \mathsf{dist}_{G'}(x,y)$ for all $x,y \in T$.

Theorem 2.1 (Chang-Ophelders [CO20, Theorem 1]). *Given one-hole instance (G,T) with $n := |V(G)|$ and $k := |T|$, one can compute a 0-emulator (G',T) for (G,T) of size $|V(G')| \le k^2$. The running time of the algorithm is $O((n+k^2)\log n)$.*

Crossing pairs and the Monge property. Let (G,T) be a one-hole instance. Assume that no terminal in T is a cut vertex of G, every terminal appears exactly once as we traverse the boundary of the outerface. Let $(t_1,t_2),(t'_1,t'_2)$ be two terminal pairs whose four terminals are all distinct. We say that the pairs $(t_1,t_2),(t'_1,t'_2)$ are *crossing* if the clockwise order in which these terminals appear on the boundary is either (t_1,t'_1,t_2,t'_2) or (t_1,t'_2,t_2,t'_1); otherwise we say that they are *non-crossing*. A collection $\mathcal{M}$ of pairs of terminals in T is called *non-crossing* if every two pairs in $\mathcal{M}$ is non-crossing. Sometimes we abuse the language and say that a set of shortest paths $\mathcal{P}$ in G is *non-crossing* when the collection of endpoint pairs for the paths is non-crossing. The *Monge property*[4] states that, for every one-hole instance (G,T) and every crossing pairs of terminals (t_1,t_2) and (t'_1,t'_2),

$$\mathsf{dist}_G(t_1,t_2)+\mathsf{dist}_G(t'_1,t'_2)\geq \mathsf{dist}_G(t'_1,t_2)+\mathsf{dist}_G(t_1,t'_2).$$

Well-structured sets of shortest paths. Consider a graph G and a collection $\mathcal{P}$ of shortest paths in G. We say that the set $\mathcal{P}$ is *well-structured* if for every pair of paths (P,P') in $\mathcal{P}$, $P\cap P'$ is a single subpath of both P and P'. It is not hard to see that every collection of shortest paths in G is well-structured if the shortest path between any two vertices in G is unique. Such condition can be enforced with high probability if we perturb the edge-weights in G slightly and apply the *isolation lemma* [MVV87]. If randomization is to be avoided, one can use a *lexicographic perturbation* by redefining the edge weights to be a vector [Cha52, DOW55, HM94], or the *leftmost rule* when choosing a shortest path [EK13] when G is a plane graph. A deterministic lexicographic perturbation scheme that guarantees the uniqueness of shortest paths in an n-vertex plane graph can be computed in $O(n)$ time [EFL18]. Therefore from here on we assume that all the planar graphs we consider have unique shortest path between every pair of vertices, and every collection of shortest paths is well-structured. The proof of the following lemma is provided in Appendix A.1.

Lemma 2.2. *Given a one-hole instance (G,T) and a non-crossing collection $\mathcal{M}$ of pairs of terminals in T, one can compute a well-structured set $\mathcal{P}$ of shortest paths, one for each pair of terminals in T in $O(|E(G)|\cdot\log|\mathcal{M}|)$ time.*

ε-covers. We use the notion of *ε-covers* [KS98, Tho04]. Let $\varepsilon\in(0,1)$ be a parameter. Let G be a graph and let P be a shortest path in G connecting some pair of vertices. Consider now a vertex v in G that does not belong to path P. An *ε-cover* of v on P is a subset S of vertices in P such that, for each vertex $x\in V(P)$, taking the detour from v to some $y\in S$ then to x is a $(1+\varepsilon)$-approximation to the shortest path from v to x, i.e., there exists $y\in S$ for which $\mathsf{dist}_G(v,y)+\mathsf{dist}_G(y,x)\leq(1+\varepsilon)\cdot\mathsf{dist}_G(v,x)$. Small ε-cover of size $O(1/\varepsilon)$ is known to exist.

Theorem 2.3 (Thorup [Tho04, Lemma 3.4]). *Let $\varepsilon\in(0,1)$ be a constant. For every shortest path P in some graph G and every vertex $v\notin P$, there is an ε-cover of v on P with size $O(1/\varepsilon)$. Moreover, such an ε-cover can be computed in $O(|E(G)|)$ time.*

We emphasize that choosing $O(1/\varepsilon)$ "portals" at equal distance on the path P as in Klein-Subramanian [KS98] is not sufficient, because the distance from v to P might be much smaller than the length of P. The linear-time construction is not stated in Lemma 3.4 of [Tho04], but it can be inferred from their proof. In fact, we will use the following construction that allows us to efficiently compute the union of ε-covers of a subset Y of vertices along the boundary of plane graph; the proof is a simple divide-and-conquer similar to Reif [Rei81], which we omit here.

[4] Technically, this is known as the *cyclic Monge property* [CO20].

Lemma 2.4. *Let $\varepsilon \in (0,1)$ be a constant and G is a plane graph. Given a subset Y of vertices that lie on the same face of G and a shortest path P connecting a pair of vertices in G, we can compute the union of ε-covers of each vertex in Y on P in $O(|E(G)| \cdot \log |Y|)$ time.*

3 Emulators for One-Hole Instances

In this section and the next one we design a near-linear time algorithm for constructing ε-emulators for one-hole instances, as stated in Theorem 3.1. We say that an ε-emulator (G', T) for a one-hole instance (G, T) is *aligned* if (G', T) is also a one-hole instance, and the circular orderings of the terminals on the outerfaces of G and of G' are identical.

Theorem 3.1. *Given a parameter $\varepsilon \in (0,1)$ and a one-hole instance (G, T) with $|T| = k$, one can compute an aligned ε-emulator for (G, T) of size $|V(G')| = \tilde{O}(k/\varepsilon^{O(1)})$ in $\tilde{O}\big((n+k^2)/\varepsilon^{O(1)}\big)$ time.*

We complement the upper bound in Theorem 3.1 with an $\Omega(k/\varepsilon)$ lower bound on the size of aligned ε-emulators for one-hole instances. This lower bound generalizes the $\Omega(k/\varepsilon)$ lower bound of [KNZ14], which holds for one-hole instances too, but only when the emulator is a minor of G (and is thus clearly an aligned emulator).

Theorem 3.2. *For every $k \geq 2$ and $(4/k) < \varepsilon < 1$, there is a one-hole instance (G, T) with $|T| = k$, such that every aligned ε-emulator (G', T) for (G, T) must have size $\Omega(k/\varepsilon)$.*

All emulators we consider are aligned and therefore we omit the word "aligned" from now on. We describe the algorithm and proof for Theorem 3.1 in Section 3.1, with the help of the core decomposition lemma (cf. Lemma 3.3). The proof to Lemma 3.3 itself is deferred to Section 4. The proof of Theorem 3.2 is provided in Appendix A.2, since it is not relevant to the proof of Theorem 1.1.

3.1 The Algorithm and its Analysis

Let (G, T) be the input one-hole instance. The algorithm for Theorem 3.1 consists of two stages. In the first stage, we iteratively decomposes (G, T) into smaller one-hole instances; and in the second stage, we compute emulators for these small instances and then combines them into an emulator for (G, T).

Throughout the algorithm we maintain a collection $\mathcal{H}$ of one-hole instances, that is initialized to be $\mathcal{H} = \{(G, T)\}$. Set $\lambda^* := c^* \log^2 k/\varepsilon^{20}$, where $k := |T|$ and $c^* > 0$ is a large enough constant. In the first stage, we repeatedly replace a one-hole instance $(H, U) \in \mathcal{H}$ where $|U| > \lambda^*$ with smaller one-hole instances obtained by applying the algorithm from Lemma 3.3 to (H, U), until every one-hole instance (H, U) in $\mathcal{H}$ satisfy $|U| \leq \lambda^*$. The core of our construction is the following lemma.

Lemma 3.3. *Given one-hole instance (H, U) with $r := |U|$, one can compute a collection of one-hole instances $\{(H_1, U_1), \ldots, (H_s, U_s)\}$, such that*

- $U \subseteq \big(\bigcup_{1 \leq i \leq s} U_i\big)$;
- $|U_i| \leq 9r/10$ *for each* $1 \leq i \leq s$;
- $\sum_{1 \leq i \leq s} |U_i| \leq O(r)$; *and*
- *for any parameter* $100 < \lambda \leq \log^2 r$, $\sum_{i:|U_i|>\lambda} |U_i| \leq r \cdot \big(1 + O(1/\lambda)\big)$.

Moreover, given an ε-emulator (Z_i, U_i) for each (H_i, U_i), algorithm Combine *computes for (H, U) an $\big(\varepsilon + O(\frac{\log^4 r}{r^{0.1}})\big)$-emulator (Z, U) of size $|V(Z)| \leq \sum_{1 \leq i \leq s} |V(Z_i)|$. The running time of both algorithms is at most $O\big((|V(H)| + r^2) \cdot \log r \cdot \log |V(H)|\big)$.*

We prove this lemma in Section 4, and in the remainder of this subsection we use it to complete the proof of Theorem 3.1.

We associate with the decomposition process a *partitioning tree* $\mathcal{T}$. Its nodes are all the one-hole instances that ever appear in the collection $\mathcal{H}$. Its root node is the initial one-hole instance (G,T), and every tree node (H,U) has children nodes corresponding to the new instances $(H_1,U_1),\dots,(H_s,U_s)$ generated by Lemma 3.3. The leaves of $\mathcal{T}$ are those that are in $\mathcal{H}$ at the end of the first stage. (To avoid ambiguity, we refer to elements in $V(\mathcal{T})$ as *nodes* and elements in $V(H)$ as *vertices*.)

We now describe the second stage of the algorithm. For each one-hole instance (H,U) in $\mathcal{H}$ at the end of the first stage, compute a 0-emulator (Z,U) for (H,U) using the algorithm from Theorem 2.1.[5] We then iteratively process the non-leaf nodes in $\mathcal{T}$ inductively in a bottom-up fashion: Given a non-leaf node (H,U) with children $(H_1,U_1),\dots,(H_s,U_s)$, let (Z_i,U_i) be the emulator computed for (H_i,U_i) by induction. Apply algorithm COMBINE from Lemma 3.3 to the emulators $(Z_1,U_1),\dots,(Z_s,U_s)$ to obtain an emulator (Z,U) for instance (H,U). After all nodes in $\mathcal{T}$ have been processed, output the emulator (G',T) constructed for the root node (G,T).

We proceed to show that the instance (G',T) computed by the above algorithm satisfies all the properties required in Theorem 3.1.

Size Bound. We first show that $|V(G')| = \tilde{O}(k/\varepsilon^{O(1)})$. We denote by $\mathcal{H}$ the collection obtained at the end of the first stage. Note that $|V(G')| \le \sum_{(H,U)\in\mathcal{H}} O(|U|^2) \le O(\max_{(H,U)\in\mathcal{H}}|U|)\cdot\sum_{(H,U)\in\mathcal{H}}|U|$. As $\max_{(H,U)\in\mathcal{H}}|U| \le \lambda^* = O(\log^2 k/\varepsilon^{O(1)})$, it now suffices to bound the total number of terminals in all resulting one-hole instances in $\mathcal{H}$ by $\tilde{O}(k/\varepsilon^{O(1)})$, which we do next via a charging scheme. Let (H,U) be a node in $\mathcal{T}$ with children $(H_1,U_1),\dots,(H_s,U_s)$.

- For instances (H_i,U_i) with $|U_i|\le\lambda^*$ (which will all be in $\mathcal{H}$ at the end of the first stage), charge every vertex in U_i to vertices in U. Since $\sum_i |U_i| \le O(|U|)$, each vertex of U gets a charge of $O(1)$ this way. We call these charge *inactive*.
- For instances (H_i,U_i) with $|U_i|>\lambda^*$, let U' be the set of all new vertices, i.e., they appear in some set U_i but not in U; we have $|U'|\le O(|U|/\log^2|U|)$ by Lemma 3.3. Charge every vertex in U' uniformly to vertices in U, so each vertex gets $O(1/\log^2|U|)$ charge. We call these charge *active*.

The total inactive charge on each vertex of T is $O(\log k)$ because $\mathcal{T}$ has height $O(\log k)$. As for the total active charge to each vertex in T, a quick calculation shows that it is at most $O(1/(\log_{(10/9)}\lambda-1))\le 1/2$. (For a complete proof see Appendix A.3.) Note that this only accounts for the *direct* active charge. For example, some terminal does not belong to the initial one-hole instance (G,T), that was first actively charged to the terminals in T, can in turn be actively charged by some other terminals later. We call such charge *indirect* active charge. The total direct and indirect active charge for each terminal in T is at most $1/2+(1/2)^2+\cdots\le 1$.

Altogether, each terminal in T is charged $O(\log k)$. Therefore, the total number of terminals in all resulting instances in $\mathcal{H}$ is bounded by $O(k\log k)$, which, combined with previous discussion, implies that $|V(G')|\le\tilde{O}(k/\varepsilon^{O(1)})$.

Correctness. It remains to show that (G',T) is an ε-emulator for (G,T). Recall that we have associated with the algorithm in first stage a partitioning tree $\mathcal{T}$. We now define, for each tree node (H,U), a value $\varepsilon_{(H,U)}$ as follows. If (H,U) is a leaf node, we define $\varepsilon_{(H,U)} := 0$. Otherwise, (H,U) is a non-leaf node

[5]This step can use any 0-emulator that has size $\operatorname{poly} k$ and can be constructed in time $\tilde{O}(n+\operatorname{poly} k)$, and we conveniently use Theorem 2.1.

with child nodes in $\mathcal{T}$ be $(H_1, U_1), \ldots, (H_s, U_s)$. Denote $r := |U|$, and let $c > 0$ be a large enough constant that is greater than the constants hidden in all big-O notations in Lemma 3.3 and $c < (c^*)^{1/20}$. We define

$$\varepsilon_{(H,U)} := \frac{c \log^4 r}{r^{0.1}} + \max\{\varepsilon_{(H_1,U_1)}, \ldots, \varepsilon_{(H_s,U_s)}\}.$$

From the properties of the algorithm COMBINE, it is easy to verify that for each node (H, U) in $\mathcal{T}$, the one-hole instance (Z, U) we construct is an $\varepsilon_{(H,U)}$-emulator for (H, U).

We now show that $\varepsilon_{(G,T)} \le \varepsilon$. Observe that there are integers $r_1, \ldots, r_t$ with $r_1 \le k$, $r_t \ge \lambda^*$, such that for each $1 \le i \le t-1$, $r_i \ge (10/9) \cdot r_{i+1}$, $\varepsilon_{(G,T)} = \sum_{1 \le i \le t} c \log^4 r_i / (r_i^{0.1})$. A quick calculation gives us $\varepsilon_{(G,T)} \le c \cdot (\log \lambda^*)^4 / (\lambda^*)^{0.1}$. (For a complete proof see Appendix A.3.) Since c is a constant, and recall that $\lambda^* = c^*/\varepsilon^{20}$ where $c^* > c^{20}$ is large enough, so $\varepsilon_{(G,T)} \le c \cdot (\log \lambda^*)^4 / (\lambda^*)^{0.1} < \varepsilon$, and therefore (G', T) is an ε-emulator for (G, T).

Running Time. Every time we implement the algorithm from Lemma 3.3 to split some instance in $(H, U) \in \mathcal{H}$ with $n' := |H|$ and $r := |U|$, the running time is $O\big((n' + r^2) \log r \log n'\big)$. We charge its running time (and also the time for COMBINE) to vertices in H as follows:

- charge the $O(n' \log r \log n')$ term uniformly to vertices in H (each gets $O(\log k \log n)$ charge);
- charge the $O(r^2 \log r \log n')$ term uniformly to terminals in U (each gets $O(k \log k \log n)$ charge).

Since the depth of the partitioning tree $\mathcal{T}$ is at most $O(\log k)$, each non-terminal vertex in G gets in total $O(\log^2 k \log n)$ charge, and each terminal in the resulting collection $\mathcal{H}$ at the end of the first stage gets in total $O(k \log^2 k \log n)$ charge. Therefore, the total running time of the algorithm is

$$O(\log^2 k \log n) \cdot n + O(k \log^2 k \log n) \cdot \tilde{O}(k/\varepsilon^{O(1)}) = \tilde{O}\big((n + k^2)/\varepsilon^{O(1)}\big).$$

4 Construct Emulator using SPLIT and GLUE: Proof of Lemma 3.3

In this subsection we provide the proof of Lemma 3.3. We first introduce the basic graph operations SPLIT and GLUE in Section 4.1. Then we describe the algorithm and its analysis.

4.1 Splitting and Gluing

In this subsection we introduce building blocks for the divide-and-conquer: procedures SPLIT and GLUE. We will decompose a single one-hole instance (H, U) into many small one-hole instances using procedure SPLIT, compute emulators for each of them, and then glue the collection of small emulators together into an emulator for (H, U) using procedure GLUE. We now introduce the procedures in more detail.

Splitting. The input to procedure SPLIT consists of

- a one-hole instance (H, U);
- a non-crossing set $\mathcal{P}$ of shortest paths in H connecting pairs of terminals in U; and
- a subset Y of vertices on the union of shortest paths in $\mathcal{P}$; set Y must contain all endpoints of paths in $\mathcal{P}$ and all vertices with degree at least three in the graph $\bigcup_{P \in \mathcal{P}} P$ (we call them *branch vertices*).

The output of procedure SPLIT is a collection of one-hole instances constructed as follows. Consider a plane embedding of H where all the terminals in U lying on the outerface of H. We *slice*[6] H open along each path P in $\mathcal{P}$ by duplicating every vertex and edge of P to create another path P' identical to P. The set of edges incident to each vertex on P are split into two sides naturally based on their cyclic order around the vertex. We index the collection of subgraphs of H obtained by the slicing of H along $\mathcal{P}$ by $\mathcal{R}$. Let R be an index in $\mathcal{R}$ that corresponds to the subgraph H_R. The plane embedding of H naturally induces a planar embedding of H_R. Define U_R to be the set of all vertices of H_R that is either a terminal in $H_R \setminus P$ or a vertex in Y. All vertices of U_R appear on the outerface of H_R, and so (H_R, U_R) is a one-hole instance. The output of procedure SPLIT is simply the collection $\{(H_R, U_R) \mid R \in \mathcal{R}\}$ that contains, for each subgraph H_R obtained by slicing H, a one-hole instance defined in the above way. See Figure 2 for an illustration. Note that each vertex $y \in Y$ may now belong to multiple instances in $\mathcal{H}$. We call them *copies* of y.

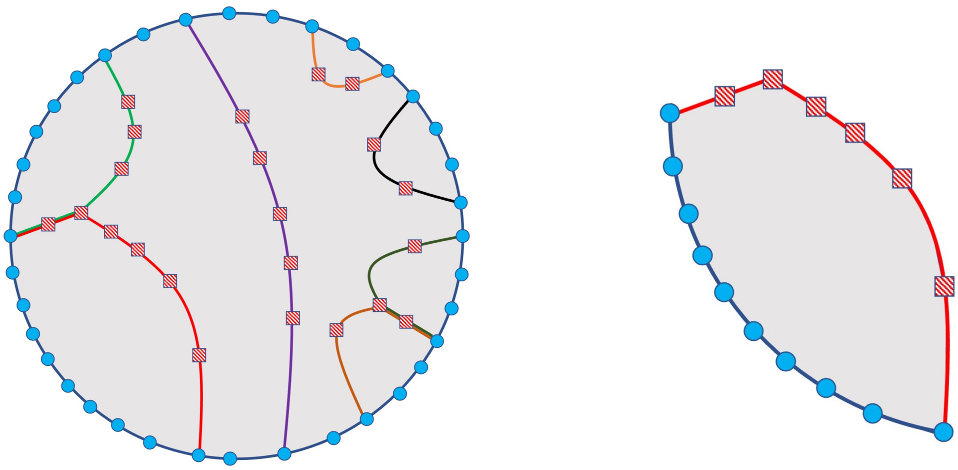


Figure 2. An illustration of splitting a one-hole instance along a path set $\mathcal{P}$. *Left*: Graph H, together with terminals in set U (in blue), paths in set $\mathcal{P}$ (in different colors), and vertices of Y (red boxes). *Right*: An output instance (that corresponds to the left bottom region of H) by procedure Split.

Gluing. We now describe procedure GLUE. Assume that we have applied procedure SPLIT to a one-hole instance (H, U), a non-crossing set $\mathcal{P}$ of shortest paths, and a vertex subset Y to obtain a collection $\mathcal{H} = \{(H_R, U_R) \mid R \in \mathcal{R}\}$ of one-hole instances. The input to procedure GLUE consists of

- one emulator (Z_R, U_R) for each one-hole instance (H_R, U_R) in $\mathcal{H}$; and
- the same vertex subset Y given as the input to procedure SPLIT.

The output of procedure GLUE is an emulator (Z, U) for (H, U), which is constructed as follows. Graph Z is obtained by taking the union of all graphs in $\{Z_R \mid R \in \mathcal{R}\}$, and identifying, for each vertex $y \in Y$, all copies of y. Graph Z is naturally a plane graph by inheriting the embeddings of all Z_Rs. (See Figure 3 for an illustration.) By the assumption that Y contains all the endpoints of paths in $\mathcal{P}$, every vertex in U shows up uniquely on the outerface of Z. Therefore, (Z, U) is a one-hole instance. Moreover, it is easy to observe that $|V(Z)| \le \sum_{R \in \mathcal{R}} |V(Z_R)|$.

[6]The slicing operation, which can be traced back to Reif [Rei81] (when describing the minimum-cut algorithm by Itai-Shiloach [IS79]), is sometimes referred to as *cutting* [**?**] or *incision* [MNNW18] in the literature.

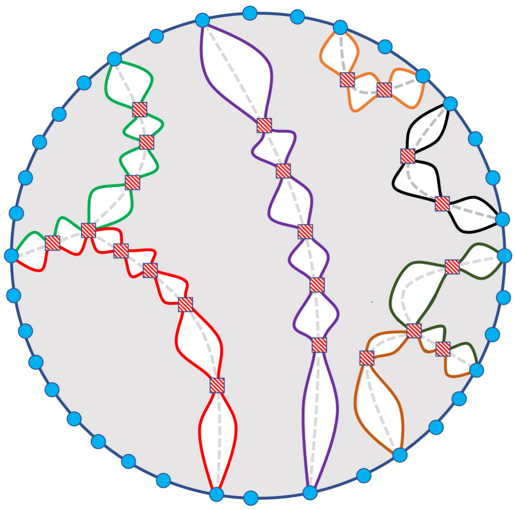


Figure 3. An illustration of gluing one-hole instances at outer-boundaries. Identified vertices of U are shown in blue) and identified vertices of $Y \setminus U$ are shown in red boxes.

One can verify that both procedures Split and Glue can be implemented in $O(|V(H)|)$ time. Now we now summarize the behavior of the procedures with the following claims. The proofs of Claim 4.1 and Claim 4.2 are deferred to Appendix A.4 and A.5 respectively.

Claim 4.1. *Let $\mathcal{H}$ be the output of procedure* Split *applied to a valid input $((H,U),\mathcal{P},Y)$, then*

1. *the number of branch vertices is at most $O(|U|)$; and*
2. *if we denote by Y^* the subset of all branch vertices in Y, then for every parameter $\lambda \geq 100$, $\sum_{(H_R,U_R)\in\mathcal{H}:\ |U_R|\geq\lambda} |U_R| \leq |U|\cdot\big(1+O(1/\lambda)\big)+O(|Y\setminus Y^*|)$.*

Claim 4.2. *Let $\mathcal{H}$ be output collection of procedure* Split *when applied to a valid input $((H,U),\mathcal{P},Y)$. Let $(\hat{H},U)$ be output of procedure* Glue *when applied to the collection $\mathcal{H}$ and set Y. For each instance $(H_R,U_R)\in\mathcal{H}$, let (Z_R,U_R) be an ε-emulator for (H_R,U_R), and let (Z,U) be the output of procedure* Glue *when applied to the collection $\{(Z_R,U_R)\}_R$ and set Y. Then (Z,U) is an ε-emulator for $(\hat{H},U)$.*

4.2 Remove All Cut Vertices in U

Before we proceed with the main ingredient for proving Lemma 3.3, first we describe a reduction on the input instance (H,U) so that no vertex in U is a cut vertex of graph H. The impatient readers may skipped ahead to Section 4.3.

We first compute the set U' of all cut vertices of H in U, and along the way the maximal 2-vertex-connected subgraphs $\hat{H}_1,\ldots,\hat{H}_t$ of H that each contains at least two terminals of U. For each $i\in\{1,\ldots,t\}$, we denote $\hat{U}_i := U\cap V(\hat{H}_i)$, so $(\hat{H}_i,\hat{U}_i)$ is a one-hole instance. Moreover, from Claim 4.2, if we are given an ε-emulator for instance $(\hat{H}_i,\hat{U}_i)$ for each i, then by simply gluing them at terminals in U', we can obtain an ε-emulator for instance (H,U). We use the following claim in order to bound $\sum_{1\leq i\leq t}|\hat{U}_i|$ and $\sum_{|\hat{U}_i|\geq\lambda}|\hat{U}_i|$.

Claim 4.3. $\sum_{1\leq i\leq t}|\hat{U}_i| \leq O(|U|)$*, and* $\sum_{|\hat{U}_i|\geq\lambda}|\hat{U}_i| \leq |U|\cdot(1+O(1/\lambda))$*.*

Proof: Recall that $r := |U|$. Consider the following tree $\mathcal{T}'$: The node set of tree $\mathcal{T}'$ is $U'\cup V'$, where $V' := \{v_i \mid 1\leq i\leq t\}$. The edge set of tree $\mathcal{T}'$ contains, for each $1\leq i\leq t$ and each node $u'\in U'$, an edge (u', v_i) if $u' \in \hat{U}_i$. Since vertices of U' are cut vertices of H, it is easy to verify that the graph $\mathcal{T}'$ constructed above is a tree, and moreover, all leaves of $\mathcal{T}'$ lie in V'.

We partition set V' into three subsets: V'_1 contains all leaf nodes of $\mathcal{T}'$, V'_2 contains all nodes of degree 2 in $\mathcal{T}'$, and $V'_{\geq 3}$ contains all nodes of degree at least 3 in $\mathcal{T}'$. Observe that, for each node $v_i \in V'_1$, since $|\hat{U}_i| \geq 2$, at least one terminal in $\hat{U}_i$ does not belong to any other set in $\{\hat{U}_1, \dots, \hat{U}_t\}$. Therefore, $|V'_1| \leq r$. Since $\mathcal{T}'$ is a tree, $|V'_{\geq 3}| \leq |V'_1| \leq r$. Since for every node in V'_2, both its neighbors lie in U', we get that $|V'_2| \leq |U'| \leq r$. Altogether, $|V(\mathcal{T}')| \leq O(r)$. Note that for every terminal $u' \in U'$, the number of sets $\hat{U}_i$ that contains u is exactly $\deg_{\mathcal{T}'}(u')$. Therefore,

$$\sum_{1 \leq i \leq t} |\hat{U}_i| \leq |U \setminus U'| + \sum_{u' \in U'} \deg_{\mathcal{T}'}(u') \leq |U| + O(|V(\mathcal{T}')|) = O(r).$$

We now upper bound $\sum_{|\hat{U}_i| \geq \lambda} |\hat{U}_i|$ via a charging scheme. We root the tree $\mathcal{T}'$ at an arbitrary node of V', and process the nodes in U' one-by-one as follows. Consider a node $u' \in U'$ such that all its child nodes are leaves in $\mathcal{T}'$. We denote by $v_1, \dots, v_s$ the child nodes of u'. For each $1 \leq i \leq s$, if $|\hat{U}_i| \geq \lambda$, we charge u' (as one unit) uniformly to vertices of $\hat{U}_i \setminus \{u'\}$, so each terminal in $\hat{U}_i \setminus \{u'\}$ is charged at most $2/\lambda$ units. We delete nodes u' and $v_1, \dots, v_s$ from $\mathcal{T}'$ and recurse on the remaining tree, until the tree contains no nodes of U'. It is easy to observe that the value of $\sum_{|\hat{U}_i| \geq \lambda} |\hat{U}_i|$ is at most r plus the total charge. We now show that the total charge is $O(1/\lambda)$. In fact, every terminal in U is directly charged at most $2/\lambda$. Note that it is possible that some terminal in U' was first charged to some other terminals in U', and was later (indirectly) charged for other terminals in U'. It is easy to observe that, the total direct and indirect charge is bounded by $2/\lambda + (2/\lambda)^2 + \dots \leq 4/\lambda$. Therefore, $\sum_{|\hat{U}_i| \geq \lambda} |\hat{U}_i| \leq r \cdot (1 + O(1/\lambda))$. □

Note that we can simply return the collection $\{(\hat{H}_i, \hat{U}_i) \mid 1 \leq i \leq t\}$ of one-hole instances as the output, and it is easy to verify from the algorithm and Claim 4.3 that the output satisfies all properties required in Lemma 3.3 (where the algorithm COMBINE is simply the procedure GLUE), unless some set $\hat{U}_i$ contains more than $(9/10)r$ terminals. However, from Claim 4.3, there is at most one such large instance. Assume without loss of generality that $(\hat{H}_1, \hat{U}_1)$ is the unique large instance. We claim that, if Lemma 3.3 holds for instance $(\hat{H}_1, \hat{U}_1)$, then Lemma 3.3 holds for the input instance (H, U). In fact, we apply the algorithm from Lemma 3.3 to instance $(\hat{H}_1, \hat{U}_1)$ and obtain a collection $\tilde{\mathcal{H}}'$, and we can simply return the collection $\tilde{\mathcal{H}} := \tilde{\mathcal{H}}' \cup \{(\hat{H}_i, \hat{U}_i) \mid 2 \leq i \leq t\}$. It is easy to verify from the above discussion that all conditions of Lemma 3.3 hold for the collection $\tilde{\mathcal{H}}$ as an output for the original instance (H, U).

From now on we focus on proving Lemma 3.3 for the unique large instance $(\hat{H}_1, \hat{U}_1)$. For convenience, we rename this large instance by (H, U), denote $r := |U|$, and treat it as the original input instance. From our algorithm, no vertex in U is a cut vertex of graph H, so if we traverse the outerface of H, then every terminal of U appears exactly once.

4.3 The Small Spread Case

Let (H, U) be a planar instance. The *spread*[7] of the instance (H, U) is defined to be

$$\Phi(H, U) := \frac{\max_{u,u' \in U} \mathsf{dist}_H(u, u')}{\min_{u,u' \in U} \mathsf{dist}_H(u, u')}.$$

For convenience, we denote $\Phi := \Phi(H, U)$. We distinguish between the following two cases, depending on whether Φ is small or large. In this subsection we assume $\Phi \leq 2^{r^{0.9} \log^2 r}$. The large spread case will be discussed in Section 4.4.

[7] sometimes also referred to as *aspect ratio*

We will employ the procedure SPLIT in order to decompose the one-hole instance (H,U) into smaller instances. Throughout this case, we use parameters

$$L_r := r/100\log^2 r \quad \text{and} \quad \varepsilon_r := \log\Phi/L_r,$$

so $\varepsilon_r = O((\log r)^4/r^{0.1})$.

Balanced terminal pairs. Denote $U := \{u_1,\ldots,u_r\}$, where the terminals are indexed according to the order in which they appear on the outerface. We say that a pair of terminals (u_i,u_j) (with $i<j$) is a *c-balanced pair* for some parameter $1/2<c<1$, if and only if $j-i\le c\cdot r$ and $i+r-j\le c\cdot r$. In other words, the terminals u_i and u_j separate the outer boundary into two segments, each contains at most c-fraction (and therefore at least $(1-c)$-fraction) of the terminals.

We first compute the $(3/4)$-balanced pair u,u' of terminals in U that, among all $(3/4)$-balanced pairs of terminals in U, minimizes the distance between them in H. We compute the u-u' shortest path P in H. Let the set Y contain the endpoints of P, together with the following vertices of P: for each $1\le i\le L_r$,

1. among all vertices v of P with $\mathsf{dist}_P(v,u)\le e^{i\varepsilon_r}$, the vertex that maximizes its distance to u;
2. among all vertices v of P with $\mathsf{dist}_P(v,u)\ge e^{i\varepsilon_r}$, the vertex that minimizes its distance to u;
3. among all vertices v of P with $\mathsf{dist}_P(v,u')\le e^{i\varepsilon_r}$, the vertex that maximizes its distance to u';
4. among all vertices v of P with $\mathsf{dist}_P(v,u')\ge e^{i\varepsilon_r}$, the vertex that minimizes its distance to u'.

In other words, if we think of path P as a line, and then mark, for each $1\le j\le L_r$, the point on the line that is at distance $e^{i\varepsilon_r}$ from u, and the point on the line that is at distance $e^{i\varepsilon_r}$ from u', then set Y contains, for all marked points, the vertices of P that are closest to it from both sides. By definition, $|Y|\le 4L_r$.

We apply the procedure SPLIT to the one-hole instance (H,U), the path set $\{P\}$ and the vertex set Y defined above. Let (H_1,U_1) and (H_2,U_2) be the instances we get. We then simply return the collection $\{(H_1,U_1),(H_2,U_2)\}$ as the output of our algorithm.

Analysis of the small spread case. We now show that the output of the algorithm in this case satisfies the properties required in Lemma 3.3. First, from the definition of procedure SPLIT, every terminal in U continues to be a terminal in at least one instance in $\{(H_1,U_1),(H_2,U_2)\}$. Moreover, since the pair (u,u') of terminals is $(3/4)$-balanced, and $|Y|\le 4L_r = r/(25\log^2 r)$, so $|U_1|\le (3/4)r + r/(25\log^2 r)\le (9/10)r$, and similarly $|U_2|\le (9/10)r$. Second, note that $|U_1|+|U_2|\le |U|+2|Y|\le r\cdot(1+O(L_r/r)) = r\cdot(1+O(\frac{1}{\log^2 r})) = r\cdot(1+O(1/\lambda))$, as $\lambda\le\log^2 r$.

We now construct an algorithm COMBINE that satisfies the required properties. Let (H'_1,U_1) be an ε-emulator for (H_1,U_1) and let (H'_2,U_2) be an ε-emulator for (H_2,U_2). The algorithm COMBINE simply applies the procedure GLUE to the collection $\{(H'_1,U_1),(H'_2,U_2)\}$ and set Y. Let (H',U') be the one-hole instance that it outputs. It is easy to verify that $U'=U$. The algorithm COMBINE simply returns the instance (H',U). It remains to show that the output of algorithm COMBINE satisfies the required properties. Note that the collection $\{(H_1,U_1),(H_2,U_2)\}$ and the set Y also constitute a valid input for procedure GLUE. Let $(\hat H,\hat U)$ be the instance output by GLUE when applied to $\{(H_1,U_1),(H_2,U_2)\}$ and Y. It is easy to verify that $\hat U = U$. We use the following claim.

Claim 4.4. *Instance $(\hat H,U)$ is a $(3\varepsilon_r)$-emulator for instance (H,U).*

We provide the proof of Claim 4.4 right after we complete the analysis for the small spread case. From Claim 4.2, (H',U) is an ε-emulator for $(\hat H,U)$. From Claim 4.4, instance $(\hat H,U)$ is a $(3\varepsilon_r)$-emulator for instance (H,U). Altogether, (H',U) is an $(\varepsilon+3\varepsilon_r) = (\varepsilon+O(\frac{\log^4 r}{r^{0.1}}))$-emulator for (H,U). This completes the proof of Lemma 3.3 in the small spread case.

Proof of Claim 4.4. We will show that, for each pair u_1, u_2 of terminals in U,

$$\mathsf{dist}_H(u_1,u_2) \le \mathsf{dist}_{\hat{H}}(u_1,u_2) \le e^{3\varepsilon_r} \cdot \mathsf{dist}_H(u_1,u_2).$$

From the procedure SPLIT, H_1 is the subgraph of H whose image lies in the region surrounded by the image of P and the segment of outer-boundary of H from u clockwise to u' (including the boundary), and H_2 is the subgraph of H whose image lies in the region surrounded by the image of P and the segment of outer-boundary of H from u anti-clockwise to u' (including the boundary), and path P is entirely contained in both H_1 and H_2. We denote by $\hat{H}_1$ the copy of H_1 in graph $\hat{H}$, and we define graph $\hat{H}_2$ similarly, so $V(\hat{H}_1) \cap V(\hat{H}_2) = Y$. We denote by P^1, P^2 the copies of path P in graphs $\hat{H}_1$ and $\hat{H}_2$, respectively. See Figure 4 for an illustration.

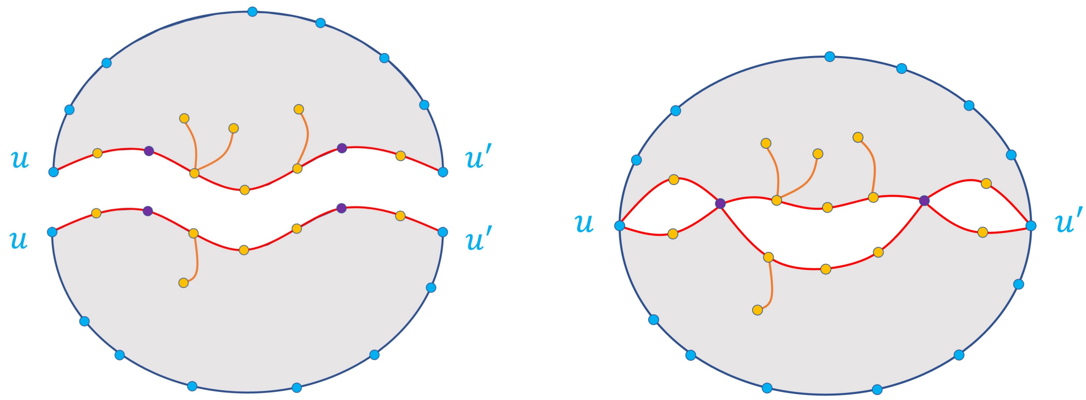


Figure 4. An illustration graphs $\hat{H}$, H_1, and H_2. *Left:* Graphs H_1 (top) graph H_2 (bottom) viewed as individual graphs. *Right:* Subgraphs $\hat{H}$ obtained by gluing graphs H_1 and H_2. Vertices in $Y \setminus \{u,u'\}$ are shown in purple.

We first show that for each pair $u_1,u_2 \in U$, $\mathsf{dist}_H(u_1,u_2) \le \mathsf{dist}_{\hat{H}}(u_1,u_2)$. Consider a pair $u_1,u_2 \in U$. Assume first that u_1,u_2 both belong to H_1 (the case where u_1,u_2 both belong to H_2 is symmetric). Clearly, in graph $\hat{H}$, there is a u_1-u_2 shortest path Q that lies entirely in $\hat{H}_1$. From the construction of $\hat{H}$, the same path belongs to H_1, and therefore $\mathsf{dist}_H(u_1,u_2) \le \mathsf{dist}_{\hat{H}}(u_1,u_2)$. Assume now that $u_1 \in V(H_1) \setminus \{u,u'\}$ and $u_2 \in V(H_2) \setminus \{u,u'\}$ (the case where $u_2 \in V(H_1) \setminus \{u,u'\}$ and $u_1 \in V(H_2) \setminus \{u,u'\}$ is symmetric). It is easy to see that, in graph $\hat{H}$, there exists a u_1-u_2 shortest path that is the sequential concatenation of

1. a path Q_1 in $\hat{H}_1$ connecting u_1 to some vertex $x_1 \in V(P^1)$, that is internally disjoint from P^1;
2. a subpath R^1 of P^1 connecting x_1 to a vertex $y \in Y$;
3. a subpath R^2 of P^2 connecting y to a vertex x_2; and
4. a path Q_2 in $\hat{H}_2$ connecting x_2 to u_2, that is internally disjoint from P^2.

Consider the path in H formed by the sequential concatenation of (i) the copy of Q_1 in H_1; (ii) the subpath R of P connecting the copy of x_1 in P to the copy of x_2 in P; and (iii) the copy of Q_2 in H_2. Clearly, this path connects u_1 to u_2 in P. Moreover, since the weight of R is at most the total weight of paths R^1 and R^2, this path in H has weight at most the weight of the u_1-u_2 shortest path in $\hat{H}$. Therefore, $\mathsf{dist}_H(u_1,u_2) \le \mathsf{dist}_{\hat{H}}(u_1,u_2)$.

From now on we focus on showing that, for each pair $u_1,u_2 \in U$, $\mathsf{dist}_{\hat{H}}(u_1,u_2) \le e^{3\varepsilon_r} \cdot \mathsf{dist}_H(u_1,u_2)$. Assume first that u_1,u_2 both belong to H_1 (the case where u_1,u_2 both belong to H_2 is symmetric). Similar to the previous discussion, the u_1-u_2 shortest path in H is entirely contained in H_1, and so $\mathsf{dist}_{\hat{H}}(u_1,u_2) = \mathsf{dist}_H(u_1,u_2)$. Assume now that $u_1 \in V(H_1) \setminus \{u,u'\}$ and $u_2 \in V(H_2) \setminus \{u,u'\}$ (the case where $u_1 \in V(H_2) \setminus \{u,u'\}$ and $u_2 \in V(H_1) \setminus \{u,u'\}$) is symmetric. Let Q be the u_1-u_2 shortest path in

H. The intersection between Q and P is a subpath of P. Let x_1, x_2 be the endpoints of this subpath, so vertices u_1, x_1, x_2, u_2 appear on path Q in this order. Let Q_1 denote the subpath of Q between u_1 and x_1, Q_2 the subpath of Q between u_2 and x_2, and Q' the subpath of Q between x_1 and x_2. We consider the following possibilities, depending on the locations of vertices x_1, x_2 and vertices in Y.

Possibility 1. There is a vertex in Y between x_1 and x_2. Let y be a vertex of Y between vertices x_1 and x_2. Consider the path $\hat{Q}$ of $\hat{H}$ formed by the sequential concatenation of (i) the copy of Q_1 in $\hat{H}_1$ connecting u_1 to the copy of x_1; (ii) the subpath R^1 of P^1 connecting the copy of x_1 to y; (iii) the subpath R^2 of P^2 connecting y to the copy of x_2; and (iv) the copy of Q_2 in $\tilde{H}_2$ connecting the copy of x_2 to u_2. Since vertex y lies between x_1 and x_2 on path P, from the construction of $\hat{H}$, the path $\hat{Q}$ in $\hat{H}$ constructed above has weight at most the weight of Q in H. Therefore, $\mathsf{dist}_{\hat{H}}(u_1, u_2) \le \mathsf{dist}_H(u_1, u_2)$.

Possibility 2. There is no vertex of Y between x_1 and x_2. Assume without loss of generality that $|V(H_1) \cap U| \ge |U|/2$, and that x_1 is closer to u than to u' in P. We use the following observation.

Observation 4.5. $\mathsf{dist}_H(x_1, u_1) \ge \mathsf{dist}_H(x_1, u)$.

Proof: Assume not, then $\mathsf{dist}_H(u_1, u) \le \mathsf{dist}_H(x_1, u_1) + \mathsf{dist}_H(x_1, u) < 2 \cdot \mathsf{dist}_H(x_1, u) \le \mathsf{dist}_H(u, u')$, and $\mathsf{dist}_H(u_1, u') \le \mathsf{dist}_H(x_1, u_1) + \mathsf{dist}_H(x_1, u') < \mathsf{dist}_H(x_1, u) + \mathsf{dist}_H(x_1, u') \le \mathsf{dist}_H(u, u')$. So both $\mathsf{dist}_H(u_1, u)$ and $\mathsf{dist}_H(u_1, u')$ is less than $\mathsf{dist}_H(u, u')$. However, since $|U|/2 \le |V(H_1) \cap U| \le (3/4) \cdot |U|$, it is easy to verify that at least one of the pairs (u_1, u), (u_1, u') is $(3/4)$-balanced, a contradiction to the fact that u, u' is the closest $(3/4)$-balanced terminal pair in H. $\square$

Think of path P as a line connecting u to u'. We now mark, for each $1 \le j \le L_r$, the point on the line that is at distance $e^{i\varepsilon_r}$ from u, and the point on the line that is at distance $e^{i\varepsilon_r}$ from u', and call these marked points *landmarks*. It is easy to observe that there is no landmark between vertices x_1 and x_2. This is because, if there is landmark between vertices x_1 and x_2, since set Y contains, for all landmark, the vertices of P that are closest to it from both sides, either x_1 or x_2 or some other vertices of P that lie between x_1 and x_2 will be added to vertex set Y, a contradiction. Let x be the landmark closest to x_1 that lies between u and x_1, and assume $\mathsf{dist}_P(x, u) = e^{i\varepsilon_r}$. Let y be the vertex of Y closest to the landmark x that lies between x and x_1. From the construction of portals, $e^{i\varepsilon_r} \le \mathsf{dist}_P(y, u) < \mathsf{dist}_P(x_1, u), \mathsf{dist}_P(x_2, u) < e^{(i+1)\varepsilon_r}$. Therefore, $\mathsf{dist}_P(x_1, y), \mathsf{dist}_P(x_2, y) \le (e^{\varepsilon_r} - 1) \cdot e^{i\varepsilon_r}$. Consider now the u_1-u_2 path in $\hat{H}$ formed by concatenation of (i) the copy of Q_1 in $\hat{H}_1$ connecting u_1 to the copy x_1^1 of x_1; (ii) the subpath of P^1 connecting x_1^1 to y; (iii) the subpath of P^2 connecting y to the copy x_2^2 of x_2; and (iv) the copy of Q_2 in $\hat{H}_2$ connecting x_2^2 to u_2. The total weight of this path is at most

$$\begin{aligned}
& \mathsf{dist}_{\hat{H}_1}(u_1, x_1^1) + \mathsf{dist}_{\hat{H}_1}(x_1^1, y) + \mathsf{dist}_{\hat{H}_2}(x_2^2, y) + \mathsf{dist}_{\hat{H}_2}(u_2, x_2^2) \\
= \; & \mathsf{dist}_H(u_1, x_1) + \mathsf{dist}_P(x_1, y) + \mathsf{dist}_P(x_2, y) + \mathsf{dist}_H(u_2, x_2) \\
= \; & \mathsf{dist}_H(u_1, x_1) + \mathsf{dist}_H(u_2, x_2) + \mathsf{dist}_P(x_1, x_2) + \big(\mathsf{dist}_P(x_1, y) + \mathsf{dist}_P(x_2, y) - \mathsf{dist}_P(x_1, x_2)\big) \\
\le \; & \mathsf{dist}_H(u_1, u_2) + 2 \cdot (e^{\varepsilon_r} - 1) \cdot e^{i\varepsilon_r} \\
\le \; & \mathsf{dist}_H(u_1, u_2) + 2 \cdot (e^{\varepsilon_r} - 1) \cdot \mathsf{dist}_H(u, x_1) \\
\le \; & \mathsf{dist}_H(u_1, u_2) + 2 \cdot (e^{\varepsilon_r} - 1) \cdot \mathsf{dist}_H(u_1, x_1) \qquad \text{(from Observation 4.5)} \\
\le \; & e^{3\varepsilon_r} \cdot \mathsf{dist}_H(u_1, u_2).
\end{aligned}$$

Therefore, $\mathsf{dist}_{\hat{H}}(u_1, u_2) \le e^{3\varepsilon_r} \cdot \mathsf{dist}_H(u_1, u_2)$. This completes the proof of Claim 4.4. $\square$

4.4 The Large Spread Case

Now we assume $\Phi > 2^{r^{0.9}\log^2 r}$. Without loss of generality, we assume that $\min_{u,u'\in U} \mathsf{dist}_H(u,u') = 1$ and $\max_{u,u'\in U} \mathsf{dist}_H(u,u') = \Phi$. In the algorithm for this case, we use the following parameters:

$$\mu = r^2, \quad L = \lceil \log_\mu \Phi \rceil, \quad \varepsilon_r = \frac{\log^4 r}{r^{0.1}}, \quad \varepsilon'_r = \frac{1}{r^{0.7}}.$$

We first compute a hierarchical partitioning $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_L)$ of terminals in U in a bottom-up fashion as follows. We proceed in L iterations. In the ith iteration, we compute a collection $\mathcal{S}_i$ of subsets of U that partition U.

- We start by letting collection $\mathcal{S}_0$ contain, for each terminal $u \in U$, a singleton set $\{u\}$. That is, $\mathcal{S}_0 := \{\{u\} \mid u \in U\}$.
- Consider an index $1 \le i \le L$. Assume we have already computed the collection $\mathcal{S}_{i-1}$ of subsets, we now describe the computation of collection $\mathcal{S}_i$, as follows. First, let graph W_{i-1} be obtained from H by contracting each subset $S \in \mathcal{S}_{i-1}$ into a single *supernode*, that we denote by v_S, and we define $V_{i-1} := \{v_S \mid S \in \mathcal{S}_{i-1}\}$. Recall that H is an edge-weighted graph, and we let every edge of W_{i-1} have the same weight as the corresponding edge in H. Then we construct another auxiliary graph R_{i-1} as follows. Its vertex set is V_{i-1}, and it contains an edge connecting v_S to $v_{S'}$ if $\mathsf{dist}_{W_{i-1}}(v_S, v_{S'}) \le \mu^i$, or equivalently $\mathsf{dist}_H(S, S') \le \mu^i$. Finally, we define $\mathcal{S}_i$ to be the collection that contains, for each connected component C of graph R_{i-1}, the set $\bigcup_{v_S \in V(C)} S$. It is easy to verify that the sets in $\mathcal{S}_i$ partition U.

This completes the description of the hierarchical partitioning $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_L)$. Clearly, collection $\mathcal{S}_L$ contains a single set U. We denote $\mathcal{S} := \bigcup_{0 \le i \le L} \mathcal{S}_i$. So collection $\mathcal{S}$ is a laminar family. That is, for every pair $S, S' \in \mathcal{S}$, either $S \cap S' = \emptyset$, or $S \subseteq S'$, or $S' \subseteq S$.

Observation 4.6. *For each set S in collection $\mathcal{S}_i$, $\mathrm{diam}_H(S) \le 2r \cdot \mu^i$.*

Proof: We prove the observation by induction on i. The base case is when $i = 0$. From the construction, the collection $\mathcal{S}_0$ contains only single-vertex sets, so the diameter of each such set is at most $0 \le 2r \cdot \mu^0$. Assume that the observation holds for $0, 1, \ldots, i-1$. Consider now a cluster $\hat{S} \in \mathcal{S}_i$. From the construction, it is the union of a collection of sets in $\mathcal{S}_{i-1}$. Consider any pair u, u' of vertices in $\hat{S}$. If they belong to the same set of in $\mathcal{S}_{i-1}$, then from the induction hypothesis, $\mathsf{dist}_H(u,u') \le 2r \cdot \mu^{i-1} \le 2r \cdot \mu^i$. Assume now that $u \in S$ and $u' \in S'$ where S, S' are distinct sets in $\mathcal{S}_{i-1}$. Since supernodes v_S and $v_{S'}$ lie in the same connected component of graph R_{i-1}, there exists a path connecting v_S to $v_{S'}$ in R_{i-1}, and we denote it by $(v_S, v_{S_1}, \ldots, v_{S_b}, v_{S'})$, where $b \le r-2$ (since the number of supernodes is at most r). If we further denote $S_0 = S$ and $S_{b+1} = S'$, then there exist, for each $0 \le j \le b+1$, a pair $\hat{u}_j, \hat{u}'_j$ of vertices in S_j, such that

- $u = \hat{u}_0$, $u' = \hat{u}'_{b+1}$;
- for each $0 \le j \le b+1$, $\mathsf{dist}_H(\hat{u}_j, \hat{u}'_j) \le 2r \cdot \mu^{i-1}$; and
- for each $0 \le j \le b$, $\mathsf{dist}_H(\hat{u}'_j, \hat{u}_{j+1}) \le \mu^i$.

Therefore, $\mathsf{dist}_H(u,u') \le r \cdot (2r \cdot \mu^{i-1}) + r \cdot \mu^i \le 2r \cdot \mu^i$, since $\mu = r^2$. □

In order to describe and analyze the algorithm, it would be convenient for us to compute a partitioning tree $\mathcal{T}$ with the hierarchical partitioning $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_L)$, in a natural way as follows. The vertex set of $\mathcal{T}$ is $V(\mathcal{T}) := V_0 \cup \ldots \cup V_L$ (recall that for each i, $V_i = \{v_S \mid S \in \mathcal{S}_i\}$, that is, V_i contains, for each set $S \in \mathcal{S}_i$, the supernode v_S representing S). We call nodes in V_i *level-i nodes* of tree $\mathcal{T}$, and we call sets in $\mathcal{S}_i$ *level-i*

sets. Since $\mathcal{S}_L = \{U\}$, there is only one level-L node in $\mathcal{T}$, that we view as the root of $\mathcal{T}$. The edge set $E(\mathcal{T})$ contains, for each pair $S, \hat{S}$ of sets such that $S \in \mathcal{S}_i, \hat{S} \in \mathcal{S}_{i+1}$ for some i and $S \subseteq \hat{S}$, an edge connecting v_S to $v_{\hat{S}}$, so v_S is a child node of $v_{\hat{S}}$, and in this case we also say that S is a *child set* of $\hat{S}$ and $\hat{S}$ is a *parent set* of S. It is easy to verify from the construction that $\mathcal{T}$ is indeed a tree.

Observation 4.7. *Let S, S' be disjoint sets in $\mathcal{S}$. Let u_1, u_2 be any pair of vertices in S, and let u'_1, u'_2 be any pair of vertices in S'. Then the pairs (u_1, u_2) and (u'_1, u'_2) of terminals are non-crossing in H.*

Proof: Assume for contradiction that the pairs (u_1, u_2) and (u'_1, u'_2) are crossing in H. Assume that S is a level-i set and S' is a level-i' set, and assume without loss of generality that $i \geq i'$.

We first find another two pairs $(u_3, u_4), (u'_3, u'_4)$ of terminals such that $\mathsf{dist}_H(u_3, u_4) \leq \mu^i$, $\mathsf{dist}_H(u'_3, u'_4) \leq \mu^{i'}$ and the pairs (u_3, u_4) and (u'_3, u'_4) are crossing. We start by finding the pair (u_3, u_4). In fact, if we denote by γ_1 the boundary segment clockwise from u'_1 to u'_2 around the outerface of H, and denote by γ_2 the boundary segment clockwise from u'_2 to u'_1 around the outerface of H, then since we have assumed that (u_1, u_2) and (u'_1, u'_2) are crossing, one of u_1, u_2 lies on γ_1 and the other lies on γ_2. Assume without loss of generality that u_1 lies on γ_1 and u_2 lies on γ_2.

From the construction of graphs $R_1, \ldots, R_{i-1}$ and collections $\mathcal{S}_1, \ldots, \mathcal{S}_i$. It is easy to observe that, for every pair u, u' of terminals that belong to the same level-i set, there exists a sequence $u^1, \ldots, u^t$ of terminals in U that all belong to the same level-i set as u and u', such that, if we denote $u = u^0$ and $u' = u^{t+1}$, then for each $0 \leq j \leq t$, $\mathsf{dist}_H(u^j, u^{j+1}) \leq \mu^i$; and for every pair u, u' of terminals do not belong to the same level-i set, $\mathsf{dist}_H(u, u') > \mu^i$.

Consider now the pair u_1, u_2 of terminals. Note that they belong to the same level-i set. From the above discussion, there exists a sequence of terminals in S starting with u_1 and ending with u_2, such that the distance between every pair of consecutive terminals in the sequence is less than μ^i. Since u_1 lies on γ_1 and u_2 lies on γ_2, there must exist a pair (u_3, u_4) of terminals appearing consecutively in the sequence, such that u_3 lies on γ_1 and u_4 lies on γ_2, so pairs (u_3, u_4) and (u'_1, u'_2) are crossing and $\mathsf{dist}_H(u_3, u_4) \leq \mu^i$.

We can then use similar arguments to find another pair (u'_3, u'_4), such that the pairs (u_3, u_4) and (u'_3, u'_4) are crossing and $\mathsf{dist}_H(u'_3, u'_4) \leq \mu^{i'}$. Note that, since $u_3, u_4 \in S$ and $u'_3, u'_4 \notin S$, $\mathsf{dist}_H(u_3, u'_3) > \mu^i$ and $\mathsf{dist}_H(u_4, u'_4) > \mu^i$. Altogether, we get that

$$\mathsf{dist}_H(u'_3, u'_4) + \mathsf{dist}_H(u_3, u_4) \leq \mu^i + \mu^{i'} \leq \mu^i + \mu^i < \mathsf{dist}_H(u_3, u'_3) + \mathsf{dist}_H(u_4, u'_4),$$

a contradiction to the Monge property on the crossing pairs (u_3, u_4) and (u'_3, u'_4). $\square$

Expanding sets. The central notion in the algorithm for the large spread case is the *expanding sets*. Recall that $\varepsilon'_r = r^{-0.7}$. We say that a set $S \in \mathcal{S}$ is *expanding* if $|\hat{S}| \geq e^{\varepsilon'_r} \cdot |S|$, where $\hat{S}$ is the parent set of S (or equivalently, $v_{\hat{S}}$ is the parent node of v_S in $\mathcal{T}$); otherwise it is *non-expanding*. We now distinguish between two cases, depending on whether $\mathcal{S}$ contains a non-expanding set with moderate size.

4.4.1 The Balanced Case: there is a non-expanding set S with $r/5 \leq |S| \leq 4r/5$

We let $\hat{S}$ be the parent set of S. We denote $S^* := \hat{S} \setminus S$, and $S' := U \setminus \hat{S}$, so the sets S^*, S, and S' partition set U. Moreover, we have $r/6 \leq |S|, |S'| \leq 5r/6$ and $|S^*| \leq (e^{\varepsilon'_r} - 1)r$. We will employ the procedure Split in order to decompose the instance (H, U) into smaller instances, for which we need to compute a non-crossing path set and a set of vertices in the path set, as the input to the procedure, as follows.

We say that an ordered pair (u, u') of terminals in S is a *border pair* if the segment on the outer-boundary of H from u clockwise to u' contains no other vertices of S but at least one vertex of $S^* \cup S'$. We compute the set $\mathcal{M}$ of all border pairs in S, and then apply the algorithm from Lemma 2.2 to graph H

and the set of border pairs $\mathcal{M}$, to obtain a set $\mathcal{P}$ of shortest paths connecting pairs in $\mathcal{M}$. We call $\mathcal{P}$ the *border path set* of S. It is easy to verify that set $\mathcal{M}$ is non-crossing, and so path set $\mathcal{P}$ is also non-crossing.

Consider now a border pair (u,u') of terminals and let $P_{u,u'}$ be the u-u' shortest path that we have computed. We apply the algorithm from Lemma 2.4 to graph H, path $P_{u,u'}$ and each vertex $u^* \in S^*$ that lies on the segment of the outer-boundary of H from u clockwise to u', with parameter ε_r, and compute an ε_r-cover of u^* on $P_{u,u'}$. We then let $Y_{u,u'}$ be the union of all vertices in these ε_r-covers and the endpoints of $P_{u,u'}$, so $Y_{u,u'}$ is a vertex set of $P_{u,u'}$. Let Y^* be the set of all vertices that are either an endpoint of a path in $\mathcal{P}$ or have degree at least 3 in the graph $\bigcup_{P\in\mathcal{P}} P$. We then define $Y := Y^* \cup (\bigcup_{(u,u')\in\mathcal{M}} Y_{u,u'})$. From Theorem 2.3,

$$|Y \setminus Y^*| \le O\left(\frac{|S^*|}{\varepsilon_r}\right) \le O\left(\frac{(e^{\varepsilon'_r}-1)\cdot r}{\varepsilon_r}\right) = O\left(\frac{(1/r^{0.7})\cdot r}{\log^4 r/r^{0.1}}\right) = O\left(\frac{r^{0.4}}{\log^4 r}\right).$$

We then apply the procedure Split to the one-hole instance (H,U), the non-crossing path set $\mathcal{P}$, and the vertex set Y. We return the collection $\mathcal{H}$ of one-hole instances output by the procedure Split as the output of our algorithm in this case.

Analysis of the Balanced Case. We now show that the output collection of one-hole instances of the above algorithm satisfies the properties required in Lemma 3.3.

First, we show in the following claim that each instance in $\mathcal{H}$ contains at most $(9/10)r$ terminals.

Claim 4.8. *Each instance in $\mathcal{H}$ contains at most $(9/10)r$ terminals.*

Proof: From the construction of the border path set $\mathcal{P}$, the one-hole instances in $\mathcal{H}$ can be partitioned into two subsets: $\mathcal{H}_1$ contains all instances that corresponds to a region in H surrounded by a segment of outer-boundary of H and the image of some path $P \in \mathcal{P}$; and set $\mathcal{H}_2$ contains all other instances.

Each instance in $\mathcal{H}_1$ contains at most two terminals in S, and so it contains at most $r-|S|+2+|Y\setminus Y^*| \le (9/10)r$ terminals (note that such an instance does not need to contain branch vertices that are not ε_r-cover vertices on its boundary). On the other hand, each instance in $\mathcal{H}_2$ does not contain terminals in S', and so it contains at most $r-|S'|+|Y| \le (9/10)r$ terminals. □

Second, note that $|Y \setminus Y^*| \le O(r^{0.4}/\log^4 r)$, then from Claim 4.1, we get that $\sum_{(H_i,U_i)\in\mathcal{H}} |U_i| \le O(r)$ and $\sum_{(H_i,U_i)\in\mathcal{H}:|U_i|>\lambda} |U_i| \le r\cdot\big(1+O(1/\lambda)\big)$.

We now construct an algorithm Combine that satisfies the required properties in Lemma 3.3. Recall that we are given, for each instance $(H_i,U_i)\in\mathcal{H}$, an ε-emulator (Z_i,U_i). The algorithm Combine simply applies Glue to instances $(Z_1,U_1),\dots,(Z_s,U_s)$ and returns instance (Z,U) output by Glue. It remains to show that the algorithm Combine satisfies the required properties. Note that the one-hole instances $(H_1,U_1),\dots,(H_s,U_s)$ also form a valid input for procedure Glue. Let $(\hat{H},\hat{U})$ be the one-hole instance that the procedure Glue outputs when it is applied to instances $(H_1,U_1),\dots,(H_s,U_s)$. It is easy to verify that $\hat{U}=U$. We use the following claim, whose proof is similar to the proof of Claim 4.4, and is deferred to Appendix A.6.

Claim 4.9. *Instance $(\hat{H},U)$ is an $O(\varepsilon_r)$-emulator for instance (H,U).*

Now we complete the proof of Lemma 3.3 for the Balanced Case using Claim 4.9. In fact, since for each $1\le i\le t$, (Z_i,U_i) is an ε-emulator for (H_i,U_i), from Claim 4.2, (Z,U) is an ε-emulator for $(\hat{H},U)$. Then from Claim 4.4 and Claim 4.9, we get that (Z,U) is an $(\varepsilon+O(\varepsilon_r))=(\varepsilon+O(\frac{\log^4 r}{r^{0.1}}))$-emulator for (H,U). Moreover, from the algorithm Glue, it is easy to verify that the instance (Z,U) output by the algorithm Combine satisfies that $|V(Z)|\le\sum_{(H_i,U_i)\in\mathcal{H}}|V(Z_i)|$.

4.4.2 The Unbalanced Case: every set S is either expanding, or $|S| < r/5$, or $|S| > 4r/5$

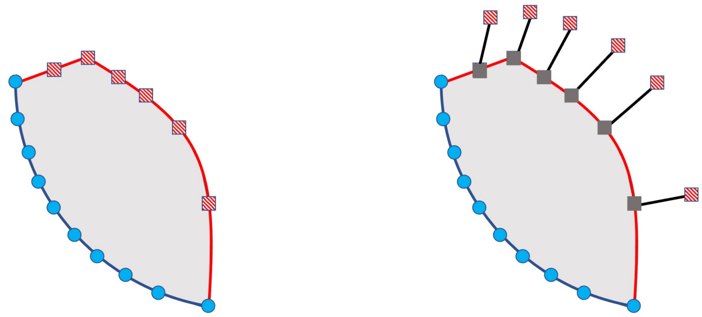
The algorithm in this case consists of two steps. Eventually, we will reduce to the Small Spread Case, and use the algorithm there to complete the decomposition of the instance (H, U).

Step 1: We say that a set $S \in \mathcal{S}$ is *heavy* if $|S| > 4r/5$, and in this case we also say that the node v_S is heavy. Clearly, every level of $\mathcal{T}$ contains at most one heavy node, and all heavy nodes form a path in $\mathcal{T}$ which ends at the root node of $\mathcal{T}$. Let $\hat{S}$ be the non-expanding heavy set that lies on the lowest level. We denote by $\hat{L}$ the level that $\hat{S}$ lies in and let $\check{S}$ be its parent set. Define $\hat{S}^* := \check{S} \setminus \hat{S}$ and $\hat{S}' := U \setminus \check{S}$. So sets $\hat{S}^*, \hat{S}, \hat{S}'$ partition set U, and $|\hat{S}^*| \le (e^{\varepsilon'_r} - 1)r$. We perform the same operations as in the Balanced Case (Section 4.4.1) to graph H with respect to the partition $(\hat{S}, \hat{S}^*, \hat{S}')$. Let $\hat{\mathcal{H}}$ be the collection we obtain. From similar analysis as in Section 4.4.1, we get that $\sum_{(H_i,U_i)\in\hat{\mathcal{H}}} |U_i| \le O(r)$, and $\sum_{(H_i,U_i)\in\hat{\mathcal{H}}:|U_i|>\lambda} |U_i| = r \cdot \big(1 + O(1/\lambda)\big)$. If additionally we have, for each $(H_i, U_i) \in \hat{\mathcal{H}}$, $|U_i| \le (9/10)r$, then we simply return the collection $\hat{\mathcal{H}}$ as the output. Assume now that there exists some instance $(H_{i^*}, U_{i^*}) \in \hat{\mathcal{H}}$ with $|U_{i^*}| > (9/10)r$. Note that we may have only one such instance. It is easy to see from the algorithm Split that no terminal of U_{i^*} is a cut vertex in graph H_{i^*}. Note that it is now enough to prove Lemma 3.3 for the instance (H_{i^*}, U_{i^*}), which we do in the next step. Indeed, if Lemma 3.3 holds for instance (H_{i^*}, U_{i^*}), then we simply apply the algorithm from Lemma 3.3 to instance (H_{i^*}, U_{i^*}) and obtain a collection $\mathcal{H}^*$ instances. We simply return the collection $\tilde{\mathcal{H}} := (\hat{\mathcal{H}} \setminus \{(H_{i^*}, U_{i^*})\}) \cup \mathcal{H}^*$. It is easy to verify that the output collection $\tilde{\mathcal{H}}$ satisfies all conditions in Lemma 3.3 for the original input instance (H, U) (where again we simply set Combine to be Glue).

Step 2: The goal of this step is to further modify and decompose the instance (H_{i^*}, U_{i^*}) into instances with small spread, and eventually apply the algorithm from the Small Spread Case to them. Consider the instance (H_{i^*}, U_{i^*}). From the algorithm Split, the instance (H_{i^*}, U_{i^*}) corresponds to a region of H, that is surrounded by shortest paths connecting terminals in U. Therefore, for every pair v, v' of vertices in H_{i^*} (that are also vertices in H), $\mathsf{dist}_H(v, v') = \mathsf{dist}_{H_{i^*}}(v, v')$. Note that set U_{i^*} can be partitioned into two subsets: set $\tilde{S}$ contains all terminals in $\hat{S}$ that lies in U_{i^*}, and set Y_{i^*} contains all new terminals (which are vertices in ε_r-covers of vertices of $\hat{S}^*$ on paths of $\mathcal{P}$ and the branch vertices) added in Step 1 that lie on the boundary of graph H_{i^*}. Note that the distances between a pair of terminals in Y_{i^*} and the distances between a terminal in Y_{i^*} and a terminal in $\tilde{S}$ could be very small (even much smaller than $\min_{u,u'} \mathsf{dist}_H(u, u')$) at the moment, which makes it hard to bound the spread from above. Therefore, we start by modifying the instance (H_{i^*}, U_{i^*}) as follows.

We let graph $\tilde{H}$ be obtained from H_{i^*} by adding, for each terminal $u \in Y_{i^*}$, a new vertex $\tilde{u}$ and an edge $(\tilde{u}, u)$ with weight $\mu^{\hat{L}-1}$. We then define $\tilde{U} := \tilde{S} \cup \{\tilde{u} \mid u \in Y_{i^*}\}$. This completes the construction of the new instance $(\tilde{H}, \tilde{U})$. We call this operation *terminal pulling*. See Figure 5 for an illustration. It is easy to verify that $(\tilde{H}, \tilde{U})$ is a one-hole instance, and moreover, for each new terminal $\tilde{u}$ in $\tilde{U} \setminus \tilde{S}$, the distance in $\tilde{H}$ from $\tilde{u}$ to any other terminal in $\tilde{U}$ is at least $\mu^{\hat{L}-1}$. We will show later in the analysis that it is now sufficient to prove Lemma 3.3 for the instance $(\tilde{H}, \tilde{U})$.

We now construct the hierarchical clustering $\tilde{\mathcal{S}}$ for instance $(\tilde{H}, \tilde{U})$, in the same way as the hierarchical clustering $\mathcal{S}$ for instance (H, U), that is described at the beginning of the large spread case. Let $\tilde{\mathcal{T}}$ be the partitioning tree associated with $\tilde{\mathcal{S}}$. Recall that for every pair of vertices in H_{i^*}, the distance between them in H_{i^*} is identical to the distance between them in H. From the construction of instance $(\tilde{H}, \tilde{U})$, it is easy to verify that both $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{T}}$ has depth $\hat{L}$, and in levels $\hat{L} - 1, \ldots, 1$, new terminals in $\tilde{U} \setminus \tilde{S}$ only form singleton sets as each of them is at distance at least $\mu^{\hat{L}-1}$ from any other terminal in $\tilde{U}$. Therefore, every non-singleton set in $\tilde{\mathcal{S}}$ is also a set in $\mathcal{S}$.



(a) Before: the instance (H_{i^*}, U_{i^*}). (b) After: the instance $(\tilde{H}, \tilde{U})$.

Figure 5. An illustration of modifying the instance (H_{i^*}, U_{i^*}).

We say that a set S is *good* if

(i) $|S| > 1$;

(ii) S lies on level at most $\hat{L} - 2\log r/\varepsilon'_r$;

(iii) S is non-expanding; and

(iv) for any other set $S' \in \tilde{\mathcal{S}}$ that lies on level at most $\hat{L} - 2\log r/\varepsilon'_r$ and $S \subseteq S'$, S' is expanding.

We denote by $\tilde{\mathcal{S}}_g$ the collection of all good sets in $\tilde{\mathcal{S}}$. Next we show that all (good) sets in $\tilde{\mathcal{S}}_g$ lie on level at least $\hat{L} - O(\log r/\varepsilon'_r)$. From definition of a good set and our assumption for the Unbalanced Case that every set $S \in \mathcal{S}$ with $r/5 \le |S| \le 4r/5$ is expanding, it is easy to see that all good sets S have size at most $r/5$ (we have used the property that every non-singleton set in $\tilde{\mathcal{S}}$ is also a set in $\mathcal{S}$).

Observation 4.10. *Every good set in $\tilde{\mathcal{S}}$ lies on level at least $\hat{L} - 10\log r/\varepsilon'_r$. Every terminal either forms a singleton set on level at least $\hat{L} - 10\log r/\varepsilon'_r$, or belongs to some good set in $\tilde{\mathcal{S}}_g$.*

Proof: Denote $\hat{L}' := \hat{L} - 2\log r/\varepsilon'_r$. Let S be a good set. Assume S lies in level i. Let $S_{i+1}, \ldots, S_{\hat{L}'}$ be the ancestor sets of S on levels $i+1, \ldots, \hat{L}'$, respectively. From the definition of good sets, all sets $S_{i+1}, \ldots, S_{\hat{L}'-1}$ are expanding, so we have

$$1 \le |S| \le |S_{i+1}| \le e^{-\varepsilon_r} \cdot |S_{i+2}| \le \cdots \le e^{-\varepsilon'_r \cdot (\hat{L}'-i-1)} \cdot |S_{L^*}| \le e^{-\varepsilon'_r \cdot (\hat{L}'-i-1)} \cdot r.$$

Therefore, $\varepsilon_r \cdot (\hat{L}' - i - 1) \le \ln r$ and so $i = \hat{L}' - 8\log r/\varepsilon'_r = \hat{L} - 10\log r/\varepsilon'_r$.

Similarly, if a terminal in $\tilde{S}$ does not form a singleton set on level at least $\hat{L} - 10\log r/\varepsilon'_r$, and it does not belong to any good set in $\tilde{\mathcal{S}}_g$, then from the inequality above, its ancestor chain has length at most $8\log r/\varepsilon'_r$, a contradiction. $\square$

Now for each good set S, we compute its border path set $\tilde{\mathcal{P}}_S$ in instance $(\tilde{H}, \tilde{U})$ in the same way as in the Balanced Case (Section 4.4.1). Now define $\tilde{\mathcal{P}} := \bigcup_{S \in \tilde{\mathcal{S}}_g} \tilde{\mathcal{P}}_S$. We show in the next observation that the collection $\mathcal{P}$ of paths is non-crossing.

Observation 4.11. *The collection $\tilde{\mathcal{P}}$ of paths is non-crossing.*

Proof: Assume for contradiction that the collection $\tilde{\mathcal{P}}$ of paths is not non-crossing. Then there exist two distinct sets $S, S' \in \tilde{\mathcal{S}}_g$, a border path P connecting terminals u_1, u_2 in S and a border path P' of S' connecting terminals u'_1, u'_2 in S', such that the pairs $(u_1, u_2), (u'_1, u'_2)$ are crossing. However, from the definition of good sets, $S \cap S' = \emptyset$. Therefore, from Observation 4.7, pairs $(u_1, u_2), (u'_1, u'_2)$ are non-crossing, a contradiction. $\square$

Consider now a good set $S \in \tilde{\mathcal{S}}_g$. We define $S^* := \check{S} \setminus S$, where $\check{S}$ is the parent set of S in $\tilde{\mathcal{S}}_g$. Recall that a pair (u,u') of terminals in S is a border pair, if the outer-boundary of $\tilde{H}$ connecting u to u' contains no other vertices of S but at least one vertex that does not lie in S. Now for each border pair (u,u') of terminals in S, let $P_{u,u'}$ be the u-u' shortest path in $\tilde{\mathcal{P}}_S$ that we have computed. We apply the algorithm from Lemma 2.4 to each vertex $u^* \in S^*$ that lies on the outer-boundary from u clockwise to u' with parameter ε_r, and compute an ε_r-cover of u^* on $P_{u,u'}$. We then let $Y^S_{u,u'}$ be the union of all such ε_r-covers and the endpoints of $P_{u,u'}$. We then let set Y^S be the union of the sets $Y^S_{u,u'}$ for all border pairs (u,u'). Finally, we define Y as the union of $\bigcup_{S\in\tilde{\mathcal{S}}_g} Y^S$ and all branch vertices (which we denote by Y^*), so Y is a vertex set of $V(\tilde{\mathcal{P}})$ that contains all branch vertices $\tilde{\mathcal{P}}$. Moreover, from Theorem 2.3,

$$\begin{aligned}|Y\setminus Y^*| &\le O\bigg(\sum_{S\in\tilde{\mathcal{S}}_g}\frac{|S^*|}{\varepsilon_r}\bigg) \le O\bigg(\frac{(e^{\varepsilon'_r}-1)\cdot\sum_{S\in\tilde{\mathcal{S}}_g}|S|}{\varepsilon_r}\bigg) \le O\bigg(\frac{(e^{\varepsilon'_r}-1)\cdot r}{\varepsilon_r}\bigg)\\ &= O\bigg(\frac{(1/r^{0.7})\cdot r}{\log^4 r/r^{0.1}}\bigg) = O\bigg(\frac{r^{0.4}}{\log^4 r}\bigg).\end{aligned}$$

We now apply the algorithm Split to instance $(\tilde{H},\tilde{U})$, the path set $\tilde{\mathcal{P}}$ and the vertex set Y. Let $\tilde{\mathcal{H}}$ be the collection of one-hole instances we get. If all instances $(\hat{H},\hat{U})$ in $\tilde{\mathcal{H}}$ satisfy that $|\hat{U}| \le (9/10)r$, then we terminate the algorithm and return $\tilde{\mathcal{H}}$. Assume that there is some instance $(\hat{H},\hat{U})$ in $\tilde{\mathcal{H}}$ such that $|\hat{U}| > (9/10)r$. From similar analysis in Step 1, there can be at most one such instance. We denote such an instance by $(\hat{H},\hat{U})$.

We now modify the instance $(\hat{H},\hat{U})$ as follows. Denote $L^* := \hat{L} - 10\log r/\varepsilon'_r$. Let H^* be the graph obtained from $\hat{H}$ by applying the terminal pulling operation to every terminal in $\hat{U}\setminus\tilde{S}$ via an edge of weight μ^{L^*-1}. We then define set U^* to be the union of $(\hat{U}\cap\tilde{S})$ and the set of all new terminals created in the terminal pulling operation. We use the following observation.

Observation 4.12. $\Phi(H^*,U^*) \le 2^{O(\log^2 r/\varepsilon'_r)}$.

Proof: From Observation 4.10, every pair of terminals in U^* has distance at least μ^{L^*-1} in graph H^*. On the other hand, since graph $\hat{H}$ is a subgraph of $\tilde{H}$, every pair of terminals in U^* has distance at most $\mu^{\hat{L}+1}$ in graph H^*. Therefore, $\Phi(H^*,U^*) \le \mu^{\hat{L}-L^*+2} = 2^{O(\log^2 r/\varepsilon'_r)}$ as $\mu = r^2$. □

Since $2^{O(\log^2 r/\varepsilon'_r)} < 2^{r^{0.9}\log^2 r}$ when r is larger than some large enough constant, we apply the algorithm from the Small Spread Case to instance (H^*,U^*) and obtain a collection $\mathcal{H}_{(\hat{H},\hat{U})}$ of instance. The output of the algorithm is the collection $\Big(\tilde{\mathcal{H}}\setminus\{(\hat{H},\hat{U})\}\Big)\cup\mathcal{H}_{(\hat{H},\hat{U})}$ of instances.

Analysis of the Unbalanced Case. Recall that in this step we assume that, after Step 1, there is an instance (H_{i^*},U_{i^*}) with $|U_{i^*}| > (9/10)r$, and we transformed it into another instance $(\tilde{H},\tilde{U})$. We first show that it is sufficient to prove Lemma 3.3 for instance $(\tilde{H},\tilde{U})$. All other conditions can be easily verified. We now show that when applying the algorithm Glue to ε-emulators $\{(\tilde{H}',\tilde{U})\}\cup\{(H'_i,U_i)\}_{i\neq i^*}$, we still obtain an $(\varepsilon+O(\frac{\log^4 r}{r^{0.1}}))$-emulator for (H,U). In fact, we only need to consider the terminal pairs u,u' with $u\in S$ and $u'\notin S$. Note that such a pair u,u' of terminals belongs to different level-$\hat{L}$ clusters in $\mathcal{S}$. From the construction of $\tilde{\mathcal{S}}$, $\mathsf{dist}_H(u,u') \ge \mu^{\hat{L}}$. Therefore, the transformation from instance (H_{i^*},U_{i^*}) to instance $(\tilde{H},\tilde{U})$ adds at most an additive $\mu^{\hat{L}-1}$ to their distance, which is at most $O(\frac{1}{\mu}) = O(\frac{1}{r^2}) \le O(\frac{\log^4 r}{r^{0.1}})$-fraction of their distance in graph H. Therefore, by gluing the ε-emulators $\{(\tilde{H}',\tilde{U})\}\cup\{(H'_i,U_i)\}_{i\neq i^*}$, we still obtain an $(\varepsilon+O(\frac{\log^4 r}{r^{0.1}}))$-emulator for (H,U).

From now on, we focus on proving that the decomposition we computed for instance $(\tilde{H},\tilde{U})$ satisfies all properties in Lemma 3.3. Recall that we have first computed a collection $\tilde{\mathcal{S}}_g$ of good sets, computed a path set $\tilde{\mathcal{P}}$ and a subset Y of vertices in $V(\tilde{\mathcal{P}})$ based on sets in $\tilde{\mathcal{S}}_g$, and then applied the procedure SPLIT to $((\tilde{H},\tilde{U}),\tilde{\mathcal{P}},Y)$ and obtained a collection $\tilde{\mathcal{H}}$ of one-hole instances.

Assume first that all instances $(\hat{H},\hat{U})$ in collection $\tilde{\mathcal{H}}$ satisfies that $|\hat{U}|\le(9/10)r$. Since $|Y\setminus Y^*|\le O\big(\frac{r^{0.4}}{\log^4 r}\big)$, from Claim 4.1, we get that $\sum_{(\hat{H},\hat{U})\in\tilde{\mathcal{H}}}|\hat{U}|\le O(r)$ and $\sum_{(\hat{H},\hat{U})\in\tilde{\mathcal{H}}:|\hat{U}|>\lambda}|\hat{U}|\le r\cdot\big(1+O(1/\lambda)\big)$. We now describe the algorithm COMBINE that, takes as input, for each instance $(\hat{H},\hat{U})\in\mathcal{H}$, an ε-emulator $(\hat{H}',\hat{U})$, computes an $\big(\varepsilon+O(\varepsilon_r)\big)=\big(\varepsilon+O(\frac{\log^4 r}{r^{0.1}})\big)$-emulator for $(\tilde{H},\tilde{U})$. We simply apply the algorithm GLUE to instances $\{(\hat{H}',\hat{U})\mid(\hat{H},\hat{U})\in\tilde{\mathcal{H}}\}$ and return the output instance $(\tilde{H}',\tilde{U})$ of GLUE. The proof that instance $(\tilde{H}',\tilde{U})$ is indeed an $\big(\varepsilon+O(\varepsilon_r)\big)$-emulator for $(\tilde{H},\tilde{U})$ and the proof that $|V(\tilde{H}')|\le\sum_{(\hat{H},\hat{U})\in\tilde{\mathcal{H}}}|V(\hat{H}')|$ use identical arguments in the Balanced Case, and is omitted here.

Assume now that there exists an instance $(\hat{H},\hat{U})$ in collection $\tilde{\mathcal{H}}$ with $|\hat{U}|>(9/10)r$. Denote $\tilde{\mathcal{H}}'=\tilde{\mathcal{H}}\setminus\{(\hat{H},\hat{U})\}$ and denote by $\overline{\mathcal{H}}=\big(\tilde{\mathcal{H}}\setminus\{(\hat{H},\hat{U})\}\big)\cup\mathcal{H}_{(\hat{H},\hat{U})}$ the output collection of instances. First, note that all instances $(\overline{H},\overline{U})$ in collection $\tilde{\mathcal{H}}'$ satisfies that $|\overline{U}|\le(9/10)r$. Since the remaining instances in $\overline{\mathcal{H}}$ is obtained by applying the algorithm from Case 1 to the instance (H^*,U^*), that is obtained from modifying the unique large instance in $(\hat{H},\hat{U})$. From the algorithm in Case 1, we know that each instance in the output collection contains at most $(9/10)r$ terminals. Second, from similar arguments, we get that $\sum_{(\overline{H},\overline{U})\in\overline{\mathcal{H}}}|\overline{U}|\le O(r)$ and $\sum_{(\overline{H},\overline{U})\in\overline{\mathcal{H}}:|\overline{U}|>\lambda}|\overline{U}|\le r\cdot\big(1+O(1/\lambda)\big)$. We now describe the algorithm COMBINE that, takes as input, for each instance $(\overline{H},\overline{U})\in\mathcal{H}$, an ε-emulator $(\overline{H}',\overline{U})$, computes an $\big(\varepsilon+O(\varepsilon_r)\big)=\big(\varepsilon+O(\frac{\log^4 r}{r^{0.1}})\big)$-emulator for $(\tilde{H},\tilde{U})$. First, consider the instances in $\mathcal{H}_{(\hat{H},\hat{U})}$ that are obtained from applying the algorithm in Case 1 to (H^*,U^*). We simply use the algorithm COMBINE described in Case 1 to compute an $\big(\varepsilon+O(\varepsilon_r)\big)$-emulator (H^{**},U^*) for instance (H^*,U^*). Finally, we apply the algorithm GLUE to instances in $\{(\overline{H}',\overline{U})\mid(\overline{H},\overline{U})\in\tilde{\mathcal{H}}'\}\cup\{(H^{**},U^*)\}$ and denote the obtained instance by $(\tilde{H}',\tilde{U})$. Note that, for different sets $S,S'\in\tilde{\mathcal{S}}_g$ such that $S\cap\hat{U}\neq\emptyset, S'\cap\hat{U}\neq\emptyset$ and $S\cap S=\emptyset$, if set S lies on level i and set S' lies on level i', then $\mathsf{dist}_H(S,S')\ge\mu^{(\max\{i,i'\}+1)}\ge\mu^{L^*}$. Therefore, from similar arguments at the beginning of the analysis, the terminal pulling operation only incur an multiplicative factor-$O(1/r)$ error of the distances between terminals in disjoint sets in $\tilde{\mathcal{S}}_g$.

The rest of the proof that instance $(\tilde{H}',\tilde{U})$ is indeed an $\big(\varepsilon+O(\varepsilon_r)\big)$-emulator for $(\tilde{H},\tilde{U})$ uses almost identical arguments in the Balanced Case, and is omitted here.

4.5 Near-linear Time Implementation of Lemma 3.3

Denote $n:=|V(H)|$. In this subsection we show that the algorithm described in this section can be implemented in time $O\big((n+r^2)\cdot\log r\cdot\log n\big)$.

The first step of the algorithm is to split the input instance (H,U) into smaller instances at cut vertices. The cut vertices of the plane graph H are simply the vertices encountered more than once when we traverse the boundary of the outerface of H, and so they can be computed in $O(n)$ time. Therefore, the algorithm in Section 4.2 can be implemented in $O(n)$ time.

Consider now the step in Section 4.3. In this step we first compute the closest $(3/4)$-balanced pair of terminals in U. We show that this can be done in $O(n\log n+r^2\log n)$ time. In fact, we use the algorithm in [Kle05] to compute an MSSP data structure of graph H, which takes time $O(n\log n)$. We then query the distances between every pair of terminals in U, which takes time $O(r^2\log n)$ as the query time of the MSSP data structure is $O(\log n)$. We can then use the acquired information to compute the closest $(3/4)$-balanced pair of terminals in U by simply dropping all the unbalanced pairs and sort. Let this pair be (u,u'). Computing the u-u' shortest-path in H takes $O(n)$ time. Computing portals (vertices of P) takes $O(n)$ time. From Section 4.1, the procedures SPLIT and GLUE can be implemented in $O(n)$ time. Therefore, the total running time of the step in Section 4.3 is $O(n\log n+r^2\log n)$.

Consider next the step in Section 4.4. In this step we first compute a hierarchical clustering of terminals in U, according to their distances in H. This can be done in $O(n\log n+r^2\log n)$ time. In fact, we can similarly use the MSSP data structure in [Kle05] and query the distances between every pair of terminals in U, and then consider the complete graph K_U on U whose edge weights are distances between pairs of its endpoints returned by the MSSP data structure. It is easy to see that, in order to construct the hierarchical clustering $\mathcal{S}$, every edge of K_U needs to be visited at most $O(1)$ times. Therefore, the construction of hierarchical clustering takes in total $O(n \log n + r^2 \log n)$ time. Note that $\mathcal{S}$ is a hierarchical clustering on a collection of r elements, so $\mathcal{S}$ contains at most $O(r)$ distinct sets. Since deciding whether or not a set in $\mathcal{S}$ is expanding or not takes $O(1)$ time, we can tell in $O(r)$ time whether we are in the Balanced Case or the Unbalanced Case.

- In the Balanced Case, the next steps are to compute border pairs, border path sets, ε_r-covers and to use procedure SPLIT to obtain smaller instances. From Theorem 2.2 and Lemma 2.4, all these takes can be done in $O(n \log r)$ time.
- In the Unbalanced Case, the next steps are to first repeat apply the steps in the Balanced Case to the non-expanding set that lies on the lowest level. From the above discussion, this takes in total $O(n \log r)$ time. If we end up with one instance (H_{i^*}, U_{i^*}) with $|U_{i^*}| > (9/10)r$, we need a final step for further splitting this instance. It is easy to verify that the operation of terminal pulling can be done in $O(r)$ time. Constructing the new collection $\tilde{\mathcal{S}}$ takes $O(n \log n + r^2 \log n)$ time. Identifying good sets in $\tilde{\mathcal{S}}$ takes $O(r)$ time. The remaining operations are computing border pairs, border path sets, ε_r-covers and using procedure SPLIT to obtain smaller instances. From the above discussion, all these takes can be done in $O(n \log r)$ time.

Altogether, the running time of the algorithm in this section is $O\big((n + r^2) \cdot \log r \cdot \log n\big)$.

5 Emulator for Edge-Weighted Planar Graphs

In this section we provide the proof of Theorem 1.1. In Section 5.1, we show an algorithm for computing ε-emulators for $O(1)$-hole instances. Then in Section 5.2, we complete the proof of Theorem 1.1 using the results in Section 5.1. We will prove in Section 5.3 that an ε-emulator of size $O_\varepsilon(k \operatorname{polylog} k)$ can be computed in $O_\varepsilon(n)$ time.

5.1 Emulator for $O(1)$-Hole Instances

In this subsection we present a near-linear time algorithm for constructing ε-emulators for $O(1)$-hole instances. We first define *aligned emulators* for $O(1)$-hole instances similarly as aligned emulators for one-hole instances, as follows. Let (G, T) and (G', T) be two h-hole instances. We denote by $\mathcal{F}$ the set of holes in G that contain the images of all terminals, and define $\mathcal{F}'$ for G' similarly, so $|\mathcal{F}| = |\mathcal{F}'| = h$. We say that instances (G, T) and (G', T) are *aligned*, if and only if there is a one-to-one correspondence between faces in $\mathcal{F}$ and faces in $\mathcal{F}'$, such that for every face $F \in \mathcal{F}$, the set $T(F)$ of terminals that it contains is identical to the set $T(F')$ of terminals contained in its corresponding face $F' \in \mathcal{F}'$, and moreover, the circular orderings in which the terminals of $T(F)$ appearing on faces F and F' are identical. If (G, T) and (G', T) aligned and (G, T) is an ε-emulator for (G', T), then we say that (G, T) is an *aligned ε-emulator* for (G', T). Throughout this section, all emulators we construct for various $O(1)$-hole instances are aligned emulators. Therefore, we will omit the word "aligned" and only refer to them by ε-emulators or simply emulators. The main result of this section is the following lemma.

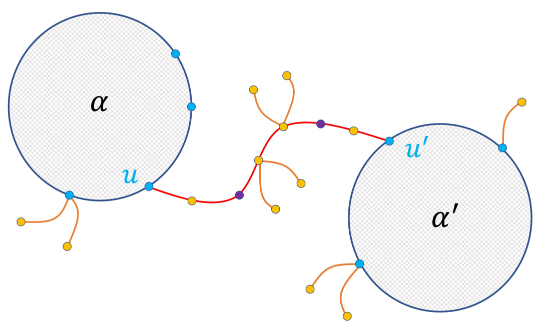
Lemma 5.1. *For any $0 < \varepsilon < 1$ and any h-hole instance (H, U) with $n := |H|$ and $r := |U|$, there exists an h-hole instance (H', U) that is an ε-emulator for (H, U) with size $|V(H')| \le r \cdot (ch \log r/\varepsilon)^{ch}$ for some universal constant c. Moreover, such an emulator can be computed in time $O\big((n + r^2) \cdot (h \log n/\varepsilon)^{O(h)}\big)$.*

The remainder of this subsection is dedicated to the proof of Lemma 5.1. We first introduce basic algorithms $\textsc{Split}_h$ and $\textsc{Glue}_h$ for splitting and gluing h-hole instances that are similar to the algorithms $\textsc{Split}$ and $\textsc{Glue}$ for splitting and gluing one-hole instances in Section 4.1.

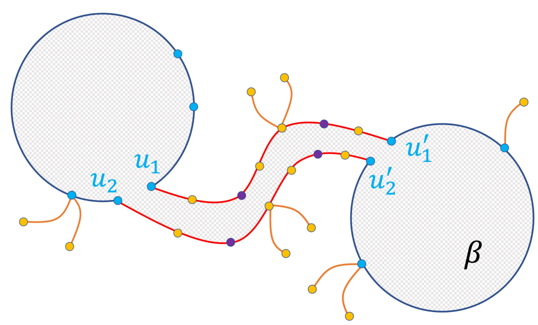
Splitting and Gluing. The input to procedure $\textsc{Split}_h$ (for some integer $h > 1$) consists of:

- an h-hole instance (H, U);
- a path P connecting a pair of terminals lying on two different holes; and
- a set $Y \subseteq V(P)$ of vertices that contains both endpoints of P.

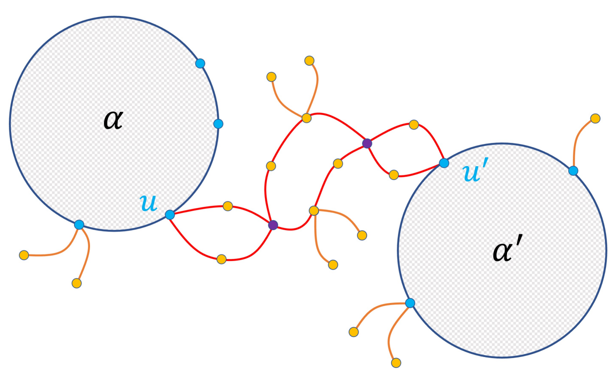
The output of $\textsc{Split}_h$ is an $(h-1)$-hole instance. Intuitively, $\textsc{Split}_h$ slices the graph H open along the path P connecting two separate holes in the graph, as illustrated in Figure 6(b). We denote by $(\tilde{H}, \tilde{U})$ the $(h-1)$-hole instance obtained by applying procedure $\textsc{Split}_h$ to instance (H, U), path P, and vertex set Y. Intuitively, procedure $\textsc{Glue}_h$ takes as input an emulator for $(\tilde{H}, \tilde{U})$, and outputs an emulator for the original instance (H, U) by identifying the two copies in $\tilde{H}$ of every vertex in Y, as illustrated in Figure 6(c). A complete description of these procedures is provided in Appendix B.1.



(a) Graph H: holes α, α' (shaded gray), terminals on α and α' (blue), path P (red), vertices of Y that are not endpoints of P (purple).



(b) Graph $\tilde{H}$: the new hole β (shaded gray), terminals on β (blue and purple), and the new u_1-u'_1 path and u_2-u'_2 path (red).



(c) An illustration of the output instance of $\textsc{Glue}_h$, when the input is the $(h-1)$-hole instances in Figure 6(b). Holes α and α' are restored.

Figure 6. An illustration of splitting and gluing an h-hole instance along a path.

Note that instance $(\tilde{H}, \tilde{U})$ is also a valid input for procedure $\textsc{Glue}_h$. Let $(\hat{H}, \hat{U})$ be the h-hole instance obtained by applying procedure $\textsc{Glue}_h$ to instance $(\tilde{H}, \tilde{U})$. Clearly, $\hat{U} = U$. We use the following claim, whose proof is similar to Claim 4.2 and thus is deferred to Appendix B.2.

Claim 5.2. *Let (Z,U) be the instance obtained by applying procedure* $\textsc{Glue}_h$ *to an ε-emulator $(\tilde{Z},\tilde{U})$ of $(\tilde{H},\tilde{U})$. Let $(\hat{H},U)$ be the instance obtained by applying procedure* $\textsc{Glue}_h$ *to $(\tilde{H},\tilde{U})$. Then (Z,U) is an ε-emulator for $(\hat{H},U)$.*

We now complete the proof of Lemma 5.1 by induction on h. The base case (when $h=1$) follows from Theorem 3.1. Consider now the case where the input (H,U) is an h-hole instance for $h>1$. We first compute a pair of terminals (u,u') that lie on different holes, and a shortest path P in H connecting u to u', such that P does not contain any terminal as internal vertices. Then for each $\hat{u}\in U\setminus\{u,u'\}$, we use the algorithm from Theorem 2.3 and parameter $\varepsilon' := \varepsilon/h$ to compute an ε'-cover of $\hat{u}$ on path P. Let Y be the union of all such ε'-covers together with the endpoints of P, so $Y\subseteq V(P)$. Note that from Theorem 2.3 we have $|Y|\le O(|U|/\varepsilon')\le O(rh/\varepsilon)$, and by using the algorithm from Lemma 2.4, Y can be computed in $O(h\cdot n\log r)$ time.

Let c be a large enough constant that is greater than all hidden constants in Theorem 3.1. We then apply the procedure $\textsc{Split}_h$ to the h-hole instance (H,U), the path P and the vertex set Y. Let $(\tilde{H},\tilde{U})$ be the $(h-1)$-hole instance $\textsc{Split}_h$ returns. From procedure $\textsc{Split}_h$, $|\tilde{U}|\le|U|+2|Y|\le c\cdot rh/\varepsilon$, since c is large enough. Recall that instance $(\hat{H},U)$ is obtained by applying the procedure $\textsc{Glue}_h$ to instance $(\tilde{H},\tilde{U})$. We use the following claim, whose proof is similar Claim 4.4, and is deferred to Appendix B.3.

Claim 5.3. *Instance $(\hat{H},U)$ is an ε'-emulator for instance (H,U).*

Consider the $(h-1)$-hole instance $(\tilde{H},\tilde{U})$. From the induction hypothesis, if we set $\varepsilon'' := \varepsilon(1-\frac{1}{h})$, then there is another $(h-1)$-hole instance $(\tilde{H}',\tilde{U})$ that is an ε''-emulator for $(\tilde{H},\tilde{U})$, such that

$$\begin{aligned}
|V(\tilde{H}')| &\le |\tilde{U}|\cdot\left(\frac{ch\cdot\log|\tilde{U}|}{\varepsilon''}\right)^{c(h-1)}\\
&\le \frac{crh}{\varepsilon}\cdot\left(\frac{ch\cdot\log(crh/\varepsilon)}{\varepsilon\cdot(1-1/h)}\right)^{c(h-1)}\\
&\le r\cdot\left(\frac{ch}{\varepsilon}\right)^{c(h-1)+1}\cdot\left(\frac{\log(crh/\varepsilon)}{(1-h)}\right)^{c(h-1)}\\
&\le r\cdot\left(\frac{ch}{\varepsilon}\right)^{ch}\cdot\left(\log r+\log(crh/\varepsilon)\right)^{c(h-1)}\\
&\le r\cdot\left(\frac{ch\log r}{\varepsilon}\right)^{ch}.
\end{aligned}$$

where we have used the fact that $(1-\frac{1}{h})^{-c(h-1)}\le e^c<c^{c-1}$, as c is large enough.

We apply procedure $\textsc{Glue}_h$ to instance $(\tilde{H}',\tilde{U})$, and let (H',U) be the h-hole instance we get. From the procedure $\textsc{Glue}_h$, $|V(H')|\le|V(\tilde{H}')|\le r\cdot(ch\log r/\varepsilon)^{ch}$. On the other hand, since instance $(\tilde{H}',\tilde{U})$ is an ε''-emulator for $(\tilde{H},\tilde{U})$, from Claim 5.2, instance (H',U) is an ε''-emulator for $(\hat{H},U)$. Since $(\hat{H},U)$ is an ε'-emulator for instance (H,U) (from Claim 5.3), using the fact that $\varepsilon''+\varepsilon'=\varepsilon(1-1/h)+\varepsilon/h=\varepsilon$, we conclude that (H',U) is an ε-emulator for instance (H,U).

Note that the above proof also gives an algorithm for constructing an ε-emulator of (H,U) of size at most $r\cdot(ch\log r/\varepsilon)^{ch}$. Specifically, if (H,U) is the input h-hole instance, then we slice it open along some shortest path P that connects a pair of terminals lying on different holes, add ε'-covers of terminals in U on P, get an $(h-1)$-hole instance $(\tilde{H},\tilde{U})$, and then we recursively construct an ε''-emulator for $(\tilde{H},\tilde{U})$ and glue it along P to get an ε-emulator for (H,U). The following claim completes the proof of Lemma 5.1.

Claim 5.4. *The running time of the above algorithm is $O\big((n+r^2)\cdot(h\log n/\varepsilon)^{O(h)}\big)$.*

Proof: We prove the claim by induction on h. The base case is when $h = 1$. From Theorem 3.1, the running time of the above algorithm is at most $(n + r^2) \cdot (c \log n/\varepsilon)^c$ on an n-vertex graph when $h = 1$. Consider the inductive case: The $\textsc{Split}_h$ and $\textsc{Glue}_h$ algorithms runs in time at most cn. Since the input to the algorithm $\textsc{Split}_h$ is an n-vertex graph, $\textsc{Split}_h$ produces a graph $(\tilde{H}, \tilde{U})$ with at most $2n$ vertices. Therefore, from the induction hypothesis, the construction of an ε'-emulator for $(\tilde{H}, \tilde{U})$ takes at most $(2n + r^2) \cdot (c(h-1) \log n/\varepsilon'')^{c(h-1)}$ time. Therefore, the total running time of the algorithm is at most

$$(2n + r^2) \cdot \left(\frac{c(h-1)\log n}{\varepsilon''}\right)^{c(h-1)} + 2cn \le (n + r^2) \cdot \left(\frac{ch \log n}{\varepsilon}\right)^{ch}. \qquad \square$$

5.2 Algorithm for General Planar Graphs: Proof of Theorem 1.1

Separators and recursive decomposition. Let r be any positive integer. An *r-division with few holes* [Fre87, KMS13] of a n-vertex connected plane graph G is a collection $\mathcal{G}$ of connected subgraphs of G, called the *pieces*, such that

- every edge in G belongs to at least one piece in $\mathcal{G}$;
- $|\mathcal{G}| = O(n/r)$;
- the number of vertices in H is at most r for each piece $H \in \mathcal{G}$;
- the number of *boundary vertices* in H (that is, vertices in $V(H)$ that also belong to some other piece in $\mathcal{G}$) is $O(\sqrt{r})$; and
- for each piece $H \in \mathcal{G}$, there are $O(1)$ faces, called *holes*, whose boundaries contain all boundary vertices of H (when considered as a plane graph).

We often refer to an r-division with few holes as an *r-division*. A standard r-division can be computed in linear time for any r [KMS13]. However in our application we need to compute r-divisions of instances that evenly distribute the terminals among pieces. In particular, we need the following lemma, whose proof is deferred to Appendix B.4.

Lemma 5.5. *Given an instance (G, T) with $n := |V(G)|$ and $k := |T|$ computing an r-division for graph G takes in $O(n)$ time, where each piece contains $O(1 + kr/n)$ terminals.*

We use the following lemma, which is crucial for the proof of Theorem 1.1.

Lemma 5.6. *Given a planar instance (H, U) with $n := |V(H)|$ and $k := |U|$, and a parameter $0 < \varepsilon < 1$, computing an ε-emulator (H', U) for (H, U) with $|V(H')| \le O\left(\sqrt{nk} \cdot (\log n/\varepsilon)^{c'}\right)$ takes $O\left(n \cdot (c' \log n/\varepsilon)^{c'}\right)$ time for some large enough universal constant c'. Furthermore, if (H, U) is an h-hole instance, then (H', U) is also an h-hole instance.*

Proof: Let c' be a constant that is greater than c and all other hidden constants in Lemma 5.1. We first compute an r-division for H, with parameter $r := n/k$ using the algorithm from Lemma 5.5. Let $\mathcal{R}$ be the collection of pieces in H that we obtain. From Lemma 5.5,

- $|\mathcal{R}| = O(k)$;
- the number of vertices in each piece in $\mathcal{R}$ is at most $O(n/k)$;
- the number of boundary vertices in each piece in $\mathcal{R}$ is at most $O(\sqrt{n/k})$;
- the number of terminals in T in each piece in $\mathcal{R}$ is $O(1)$; and
- there are $O(1)$ holes in each piece in $\mathcal{R}$.

For each graph piece R in $\mathcal{R}$, let U_R be the set that contains all boundary vertices of R and all terminals in U. Observe that (R, U_R) is an h-hole instance for some constant h. We apply the algorithm from Lemma 5.1 to instance (R, U_R), and let (R', U_R) be the ε-emulator we get, so $|V(R')| \le |U_R| \cdot (ch \log(n/k))/\varepsilon)^{ch}$. Also, such an emulator can be computed in at most $(|V(R)| + |U_R|^2) \cdot (h \log n/\varepsilon)^{ch}$ time. Therefore, all emulators in $\{(R', U_R) \mid R \in \mathcal{R}\}$ can be computed in time

$$\sum_{R\in\mathcal{R}} O\left(\left(|V(R)| + |U_R|^2\right) \cdot \left(\frac{h \log n}{\varepsilon}\right)^{ch}\right) \le O\left(n \cdot \left(\frac{h \log n}{\varepsilon}\right)^{ch}\right) \le O\left(n \cdot \left(\frac{c' \log n}{\varepsilon}\right)^{c'}\right),$$

as $\sum_{R\in\mathcal{R}} |V(R)| \le O(k) \cdot (n/k) = O(n)$, $\sum_{R\in\mathcal{R}} |U_R|^2 \le O(k) \cdot (\sqrt{n/k})^2 \le O(n)$, and c' is large enough. We then glue the emulators together via a process similar to GLUE and GLUE$_h$, and eventually obtain an ε-emulator (H', U) for (H, U), with size

$$|V(H')| \le \sum_{R\in\mathcal{R}} |U_R| \cdot \left(\frac{ch \log(n/k)}{\varepsilon}\right)^{ch} \le O\left(k \cdot \sqrt{\frac{n}{k}}\right) \cdot \left(\frac{ch \log n}{\varepsilon}\right)^{ch} \le O\left(\sqrt{nk} \cdot \left(\frac{\log k}{\varepsilon}\right)^{c'}\right),$$

as both c and h are constants. □

Algorithm for Theorem 1.1. Let G be the input n-vertex plane graph and let T be the set of terminals of size k. We first preprocess the graph G into a new graph G_0 as follows. If $n < k^2$, then we set $G_0 = G$. If $n \ge k^2$, we use the algorithm in [CGH16, Theorem 6.9] with parameter $\varepsilon/2$ to compute an $(\varepsilon/2)$-emulator G_0 for G with size $O(k^2 \log^2 k/\varepsilon^2)$. This can be done in time $\tilde{O}(n/\varepsilon^{O(1)})$ by a slight modification of the algorithm in [CGH16] (in particular, we remove their preprocessing step that reduces the number of vertices to k^4). Either way, we obtain an $(\varepsilon/2)$-emulator G_0 for G, and $|V(G_0)| = O(k^2 \log^2 k/\varepsilon^2)$.

We then set $L := \log\log k$ and $\varepsilon' := \varepsilon/2L$. Now sequentially for each $0 \le i \le L-1$, we apply the algorithm from Lemma 5.6 to instance (G_i, T) and parameter ε' to obtain an ε'-emulator (G_{i+1}, T) for (G_i, T). Finally, we return $(G', T) = (G_L, T)$ as the output. Note that $\varepsilon' L = (\varepsilon/2)$ and thus (G_L, T) is an $\varepsilon/2$-emulator of (G_0, L), and is therefore an ε-emulator for (G, T). From Lemma 5.6, the running time of our algorithm is $\tilde{O}(n/\varepsilon^{O(1)})$. In order to complete the proof of Theorem 1.1, it suffices to show that $|V(G')| \le O(k \cdot (\log k/\varepsilon)^{O(1)})$, which follows immediately from the next claim (by setting $i = L$).

Claim 5.7. *For each $0 \le i \le L$, $|V(G_i)| \le k^{1+2^{-i}} \cdot (\log k/\varepsilon')^{2c' - c'/2^i}$.*

Proof: We prove the claim by induction on i. The base case is when $i = 0$. From the preprocessing step, $|V(G_0)| \le O(k^2 \log^2 k/\varepsilon^2) \le k^2 (\log k/\varepsilon')^2$, so the claim holds, as c' is large enough. Consider the inductive case. From Lemma 5.6,

$$\begin{aligned}
|V(G_i)| &\le \sqrt{|V(G_{i-1})| \cdot k} \cdot \left(\frac{\log k}{\varepsilon'}\right)^{c'} \\
&\le \sqrt{\left(k^{1+2^{-(i-1)}} \cdot (\log k/\varepsilon')^{2c' - c'/2^{(i-1)}}\right) \cdot k} \cdot \left(\frac{\log k}{\varepsilon'}\right)^{c'} \\
&\le k^{(1+2^{-(i-1)}+1)/2} \cdot \left(\frac{\log k}{\varepsilon'}\right)^{(2c' - c/2^{(i-1)})/2 + c'} \\
&= k^{1+2^{-i}} \cdot (\log k/\varepsilon')^{2c' - c'/2^i}.
\end{aligned}$$

Therefore the claim holds for all i. □

5.3 Bootstrapping

Perhaps surprisingly, we can further reduce the running time for constructing an ε-emulator to be linear to the size of the graph whenever k is "sufficiently" sublinear and the range of the edge weights (that is, the ratio between the smallest and largest weights) are polynomially bounded, using the idea of *bootstrapping* combining with a precomputed look-up table.

Theorem 5.8. *Given any parameter $0 < \varepsilon < 1$ and any instance (H,U) with $n := |H|$ and $k := |U|$ satisfying $k \le n/\log^D n$ for some big enough constant D, and the range of the edge weights are bounded by polynomial in n, computing an emulator (Z,U) for (H,U) of size $|V(Z)| \le O(k \operatorname{polylog} k/\varepsilon^{O(1)})$ takes $O_\varepsilon(n)$ time. Furthermore, if (H,U) is an h-hole instance, then (Z,U) is an h-hole instance.*

Proof: We apply r-division iteratively with exponentially-growing values of r; intuitively each time we shrink the graph by a very small amount, just enough to absorb the logarithmic terms required to compute the emulators.

- First compute r-division of H for $r := (\log\log\log n)^{6C}$ that evenly distribute the terminals in U using Lemma 5.5, where C is bigger than the number of logs we need in the running time of Theorem 1.1. Replace each piece in the r-division by an ε-emulator with respect to the boundary vertices and terminals using Theorem 1.1; every piece contains $O(r^{1/2} + k(\log\log\log n)^{6C}/n) \le O(r^{1/2})$ boundary vertices and terminals. The total time on the emulator construction is

$$O\left(r \cdot \left(\frac{\log r}{\varepsilon}\right)^{O(1)}\right) \cdot O\left(\frac{n}{r}\right) \le O\left(\frac{n \cdot (\log\log\log\log n)^{O(1)}}{\operatorname{poly} \varepsilon}\right);$$

  and the new graph H' has size

$$O\left(r^{1/2}\left(\frac{\log r^{1/2}}{\varepsilon}\right)^{C}\right) \cdot O\left(\frac{n}{r}\right) \le O\left(\frac{n}{\varepsilon^C (\log\log\log n)^{2C}}\right).$$

- Now the graph is about $(\log\log\log n)^{2C}$-factor smaller than original, we can compute another r'-division for $r' := (\log\log n)^{6C}$, and replace each piece in the r'-division by an ε-emulator with respect to the boundary vertices and terminals; every piece contains $O(r'^{1/2} + k(\log\log n)^{6C}/n) \le O(r'^{1/2})$ boundary vertices and terminals. This way, instead of spending $O_\varepsilon(n(\log\log\log n)^{O(1)})$ time if we perform r'-division directly on the original graph, now it takes time

$$O_\varepsilon\left(\frac{n}{(\log\log\log n)^{2C}} \cdot (\log\log\log n)^{C}\right) \le O_\varepsilon(n).$$

  The new graph H'' has size about $O_\varepsilon(n/(\log\log n)^{2C})$.

- Now the graph is about $(\log\log n)^{2C}$-factor smaller than original, we can compute another r''-division for $r'' := (\log n)^{6C}$, and replace each piece in the r''-division by an ε-emulator with respect to the boundary vertices and terminals; every piece contains $O(r''^{1/2} + k(\log n)^{6C}/n) \le O(r''^{1/2})$ boundary vertices and terminals, and this takes time

$$O_\varepsilon\left(\frac{n}{(\log\log n)^{2C}} \cdot (\log\log n)^{C}\right) \le O_\varepsilon(n).$$

  The new graph H''' has size about $O_\varepsilon(n/(\log n)^{2C})$.

- Finally, compute an ε-emulator for H''' with respect to the terminals. This takes time

$$O_\varepsilon\left(\frac{n}{(\log n)^{2C}} \cdot (\log n)^C\right) \le O_\varepsilon(n)$$

  The final emulator has size $O(k \operatorname{polylog} k/\varepsilon^{O(1)})$.

The accumulated distortion in distance is 4ε. Overall the bottleneck is to compute the first set of emulators for pieces in the r-division, which takes $O(n \cdot (\log\log\log\log n)^{O(1)})$ time. We can avoid spending super-linear time to compute the first set of emulators; instead, we precompte a look-up table for every graph up to size $r = (\log\log\log n)^{6C}$, every possible subset of terminals, and every edge-weight functions rounded to the closest power of $1+\varepsilon$.

Look-up table. Now we can describe the construction of the look-up table.

- There are $2^{O(r)}$ plane graphs K up to size r.
- There are 2^r possible choices for the terminal subset U_K.
- The spread of any instance (K, U_K) is at most $n^{O(1)}$ because the range of the edge weights is polynomial in n; so if we round the weight of each edge to the closest power of $1+\varepsilon$, there are $\log_{1+\varepsilon} n^{O(1)} \le O(\log n/\varepsilon)$ possible weight values per edge, and thus $O(\log n/\varepsilon)^{2^{O(r)}}$ many different (rounded) edge-weight functions (because ε is a constant).
- Computing an ε-emulator for each instance (K, U_K) takes $r^{O(1)}$ time.

Overall, it takes

$$2^{O(r)} \cdot 2^r \cdot O(\log n/\varepsilon)^{2^{O(r)}} \cdot r^{O(1)} \le 2^{2^{O_\varepsilon((\log\log\log n)^{6C})}} \le o_\varepsilon(n)$$

time to precompute a look-up table, so that for any instance (K, U_K) from the pieces of the first r-division, one can round the edge weights of K and find the ε-emulator for (K, U_K) directly from the look-up table. Rounding the edge-weights to the closest powers of $(1+\varepsilon)$ will introduce at most $O(\varepsilon)$ distortion. As a result, an ε-emulator of size $O(k \operatorname{polylog} k/\varepsilon^{O(1)})$ for (H, U) can be computed in $O_\varepsilon(n)$ time. □

6 Applications

In this section we present efficient ε-approximate algorithms to several optimization problems on planar graphs that beat their exact counterparts, inclusing multiple-source shortest paths, minimum (s,t)—cut, graph diameter, and offline dynamic distance oracle. To put emphasis on the new ideas presented, we assume the readers are familiar with the various tools for optimization on planar graphs and only provide citations to the earlier literature.

6.1 Approximate Multiple-Source Shortest Paths

The approximate multiple-source shortest paths data structure (ε-*MSSP*) can achieve the following task: Preprocess a plane graph P and a set of terminals U on the outerface of P (that is, a one-hole instance (P, U)), and answer distance queries between terminal pairs within $(1+\varepsilon)$-approximation.

To prove Theorem 1.2, apply Theorem 5.8 on (P, U) to construct another one-hole instance (P', U) that is an ε-emulator of (P, U), which has size

$$O\left(\frac{(n/\log^C n) \cdot \operatorname{poly}\log n}{\varepsilon^{O(1)}}\right) = O\left(\frac{n}{\varepsilon^{O(1)} \operatorname{poly}\log n}\right)$$

and takes $O_\varepsilon(n)$ time. Now construct the MSSP data structure on P' using Klein's algorithm [Kle05], which takes $O\left(\frac{n}{\varepsilon^{O(1)}\operatorname{poly}\log n}\cdot\log n\right) = O(n/\varepsilon^{O(1)})$ time; MSSP answers queries in time $O(\log n)$, which is an ε-approximation to the actual distance between the pairs due to the fact that (P', U) is an ε-emulator. This proves Theorem 1.2.

6.2 Approximate Minimum Cut

Here we briefly summarize the minimum (s,t)-cut algorithm on planar graphs with non-negative weights by Italiano, Nussbaum, Sankowski, and Wulff-Nilsen [INSW11]. Many details and edge-cases are omitted for the clarity of presentation. Let G be the input plane graph, and two vertices s and t.

1. Compute the dual graph G^* of G; it is sufficient to compute a shortest cycle in G^* that separates the *faces* s^* and t^*. Find a shortest s^*-t^* path π in G^*. This step takes $O(n)$ time [HKRS97].
2. Construct r-division in G^* respecting π where $r := \log^6 n$. Cut π open; now each vertex on π has a copy. This step takes $O(n)$ time [KMS13].
3. Compute MSSP [Kle05] for each piece in the r-division with respect to the boundary vertices. Prepare the Monge heap data structures [FR06], and represent each piece as a *dense distance graph*. This step takes $O(n\log r) = O(n\log\log n)$ time for the MSSP [Kle05], and $O(n\log\log n)$ time to set up the Monge heap data structures and dense distance graphs [FR06].
4. Denote the length of π as p. Compute $p/\log p$ shortest paths between the two copies of each evenly spaced points on π, using Reif's divide-and-conquer strategy [Rei81]; each shortest path is computed by FR-Dijkstra [FR06] on the dense distance graphs. Now the graph is cut into $p/\log p$ *slabs*. This step takes $\tilde{O}(n/\sqrt{r}\cdot\log(p/\log p)) \le O(n)$ time.
5. Apply Reif's strategy directly on each slab which now has only $O(\log p)$ vertices from π, so it takes $O(n\log p) = O(n\log\log n)$ time.

Overall the algorithm takes $O(n\log\log n)$ time, with Step 3 being the bottleneck.

We can safely truncate the edge weights to have polynomial range in linear time when solving the minimum (s,t)-cut problem. Now by simply choosing $r := \log^C n$ with a bigger C and replacing Step 3 with an ε-emulator per piece using Theorem 5.8, the new graph has size $O(\frac{n}{r}\cdot\sqrt{r}\operatorname{poly}\log r/\varepsilon^{O(1)}) = O(n/\varepsilon^{O(1)}\operatorname{poly}\log n)$. We can now compute p shortest paths (instead of $p/\log p$) in Step 4 without recursion in Step 5 using Reif's divide-and-conquer strategy directly on the emulators without preparing the MSSP and Monge heap data structures in Step 3 and FR-Dijkstra in Step 4. Therefore the total running time is now $O_\varepsilon(n)$, proving Theorem 1.3.

6.3 Approximate Diameter

Here we summarize the $(1+\varepsilon)$-approximate algorithm to compute the diameter of planar graphs with non-negative edge weights by Weimann-Yuster [WY16] and Chan-Skrepetos [CS19]. Again we omit some details about marking/unmarking vertices in the actual algorithm to emphasize on core concepts. Let G be the input planar graph. Given three graphs H, H' and H'', denote $\operatorname{diam}_H(H', H'')$ the longest shortest-path distance with respect to H between a vertex in H' and a vertex in H''.

1. Compute a *shortest-path* cycle separator C in G and splits G into A and B, where $A\cup B = G$ and $A\cap B = C$, using the algorithm by Thorup [Tho04]. This step takes $O(n)$ time.
2. Construct an auxillary graph G^+ by selecting $O(1/\varepsilon)$ evenly-spaced *portals* on C; run single-source shortest path algorithm on each portal p to get maximum distance out of all paths from p, denoted as ℓ; add edges from every vertex in A and B to the portals, with the edge-weights being their

distances rounded to multiples of $\varepsilon\ell$. This step takes $O(n \cdot (1/\varepsilon))$ time using the linear-time single-source shortest path algorithm by Henzinger-Klein-Rao-Subramanian [HKRS97].

3. Approximate $\text{diam}_{G^+}(A, B)$. This step takes $O(n/\varepsilon) + 2^{O(1/\varepsilon)}$ time using brute-force [WY16], or $O(n \cdot (1/\varepsilon)^5)$ time using the farthest Voronoi diagram [CS19].
4. Build another auxillary graph A^+ from G by first adding *denser portals* on C, computing shortest paths between denser portals on C with respect to B, then planarizing the union of all the shortest paths between dense portal pairs so that A^+ remains planar. Following Chan-Skrepetos [CS19], the number of denser portals can be set to $|G|^{1/8}/\varepsilon$; compute all-pairs shortest paths between dense portals in B takes $O(|B| \log n + \log n \cdot \sqrt{|B|}/\varepsilon^4)$ time using MSSP [Kle05]; A^+ has size $|A| + O(|A|^{1/2}/\varepsilon^4)$. Build the graph B^+ similarly by switching the roles of A and B.
5. Approximate $\text{diam}_{A^+}(A, A)$ and $\text{diam}_{B^+}(B, B)$ recursively; the recursion depth is $O(\log n)$.
6. Return the maximum of $\text{diam}_{G^+}(A, B)$, $\text{diam}_{A^+}(A, A)$, and $\text{diam}_{B^+}(B, B)$.

Overall the algorithm takes $O(n \log^2 n + n \log n \cdot (1/\varepsilon)^5)$ time.

Again we can safely truncate the edge weights to have polynomial range when solving the diameter problem. Now we can substitute the construction of A^+ and B^+ using planarized shortest paths in Step 4 with two ε-emulators using Theorem 5.8, which only takes $O_\varepsilon(|A| + |B|)$ time to construct and has size $O_\varepsilon((|A|^{1/8} + |B|^{1/8}) \operatorname{poly} \log n)$. Thus we improve the total running time to $O_\varepsilon(n \log n)$, proving Theorem 1.4.

6.4 Offline Dynamic Approximate Distance Oracle

Here we describe the crucial step in the algorithm by Chen *et al.* [CGH+20] to construct an offline dynamic $(1+\varepsilon)$-approximate distance oracle with $O(\operatorname{poly} \log n)$ query and update time, assuming that a $(1+\varepsilon)$-*distance-approximating minor* of size $\tilde{O}(k)$ for a planar graph of size n and k terminals can be computed in $O(n \operatorname{poly}(\log n, \varepsilon^{-1}))$ time. Given a sequence of graphs $G_0 \subseteq G_1 \subseteq \cdots \subseteq G_\ell$, denote $H_p := G_p \setminus G_{p-1}$ for any $p \in \{1, \ldots, \ell\}$. The proof of Theorem 4.15 in Chen *et al.* [CGH+20] iteratively constructs graphs $G'_1, \ldots, G'_\ell$ in the following way:

$$G'_p := \textsc{Emulator}(G'_{p-1} \cup H_p, T_p)$$

for some terminal set T_p (irrelevant to the discussion here), where $\textsc{Emulator}(G, T)$ returns an ε-emulator of G with respect to terminal set T. When $\textsc{Emulator}(G, T)$ guarantees to return a minor of the input graph G, one can argue that G'_p must be a minor of $G'_{p-1} \cup H_p$, which by induction is a minor of $\bigcup_{1 \le k \le p-1} H_k \cup H_p = G_p$ which must be planar [CGH+20, Lemma 4.16].

To prove Theorem 1.5, we follow the algorithm by Chen *et al.* [CGH+20] almost verbatim; the only missing piece is to prove that G'_p remains planar in our setting. Observe that our emulator construction solely relies on the $\textsc{Split}$ and $\textsc{Glue}$ procedures introduced in Section 4.1. (The base case from Theorem 2.1 can be replaced by the $O(k^4)$-size distance-approximating minor [KNZ14].) While the emulator G' produced by split-and-glue is technically not a minor of the input graph G, there is another planar supergraph $\hat{G}$ modified from G such that G' is a minor of $\hat{G}$. Now we can proceed to prove that G'_p is planar using our construction for $\textsc{Emulator}(G, T)$.

Claim 6.1. *For any $p \in \{1, \ldots, \ell\}$, G'_p is planar when* $\textsc{Emulator}(G, T)$ *is implemented using Theorem 1.1.*

Proof: We will prove the following stronger statement by induction on p: there is a planar graph $\hat{G}_p$ constructed from G_p by vertex spitting (the reverse operation to edge contraction), edge subdivision (by breaking an edge into two using a degree-2 node), and edge duplications (by creating multiedges from an existing edge), and contains G'_p as a minor. We say a plane graph H is a *topological minor* of some graph $\hat{H}$ if $\hat{H}$ is constructed from H by vertex spitting, edge subdivision and edge duplications. (Notice that this is difference from the standard terminology; in fact it is a topological minor *in the dual*.) Notice the crucial property that if plane graph H is a topological minor of $\hat{H}$, then $\hat{H}$ must also be a plane graph.

First we introduce an operation that we will later use in the construction of $\hat{G}_p$. Recall that we can *slice* a graph H open along some path P by duplicating every vertex and edge of P to create another path P' identical to P. The set of edges incident to each vertex on P are split into two sides naturally based on their cyclic order around the vertex. Now we also add an edge between each vertex on P and its copy in P'. We call this operation a *pizza slice*. A pizza slice of a graph H must contain H as a topological minor. Every graph constructed from slice-and-gluing H along a set of paths is a minor of some pizza slice of H.

By induction hypothesis, there is a planar graph $\hat{G}_{p-1}$ containing G'_{p-1} as a minor and G_{p-1} as a topological minor. Now because the endpoints of all edges in H_p can still be found in G'_{p-1} and $\hat{G}_{p-1}$, $G'_{p-1} \cup H_p$ is a minor of $\hat{G}_{p-1} \cup H_p$. We know by induction hypothesis that $\hat{G}_{p-1}$ contains G_p as a topological minor, so edges in H_p can be safely added to $\hat{G}_{p-1}$ without destroying planarity; therefore $\hat{G}_{p-1} \cup H_p$ is still planar, and so does $G'_{p-1} \cup H_p$. Therefore $G'_p := \textsc{Emulator}(G'_{p-1} \cup H_p, T_p)$ is also planar from the emulator construction.

Now we describe the construction of $\hat{G}_p$ from G_p and G'_p. As G'_p is constructed using split-and-glue from $Z_p := G'_{p-1} \cup H_p$ by Theorem 1.1, there is a pizza slice $\hat{Z}_p$ of Z_p that contains G'_p as a minor. Using the lifting property that a topological minor commutes with a minor, there is another plane graph $\hat{G}_p$ that contains $\hat{G}_{p-1} \cup H_p$ as a topological minor; one can indeed construct $\hat{G}_p$ from $\hat{G}_{p-1} \cup H_p$ using pizza slices on a set of paths mimicking the one used during the slice-and-glue operations to obtain G'_p from Z_p. Now $\hat{G}_p$ contains G'_p as a minor because $\hat{G}_p$ contains $\hat{Z}_p$ as a minor and $\hat{Z}_p$ contains G'_p as a minor by construction. $\hat{G}_p$ also contains G_p as a topological minor because $\hat{G}_p$ contains $\hat{G}_{p-1} \cup H_p$ as a topological minor, which by induction contains $G_{p-1} \cup H_p$ as a topological minor. Therefore the existence of $\hat{G}_p$ is established.

The base case is clear: Define $\hat{G}_1$ to be the pizza slice of $G'_0 \cup H_1 = G_0 \cup H_1 = G_1$ that contains G'_1 as a minor from the emulator construction. Thus the claim is proved. $\square$

Acknowledgements

We thank the anonymous reviewers for their helpful comments, as well as pointing out the result by Chen *et al.* [CGH+20] on the offline dynamic approximate distance oracles.

References

[ABS+20] Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Computer Science Review*, 37:100253, 2020. doi:10.1016/j.cosrev.2020.100253.

[AD16] Amir Abboud and Søren Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 477–486, October 2016. doi:10.1109/FOCS.2016.58.

[AGK14] Alexandr Andoni, Anupam Gupta, and Robert Krauthgamer. Towards $(1+\varepsilon)$-approximate flow sparsifiers. In *25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 279–293, 2014. doi:10.1137/1.9781611973402.20.

[AKM+87] Alok Aggarwal, Maria M Klawe, Shlomo Moran, Peter Shor, and Robert Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, November 1987. doi:10.1007/BF01840359.

[BG08] Amitabh Basu and Anupam Gupta. Steiner point removal in graph metrics. Unpublished manuscript, available from http://www.math.ucdavis.edu/~abasu/papers/SPR.pdf, 2008.

[BG13] Yair Bartal and Lee-Ad Gottlieb. A linear time approximation scheme for Euclidean TSP. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 698–706, Berkeley, CA, USA, October 2013. IEEE. doi:10.1109/FOCS.2013.80.

[Cab12] Sergio Cabello. Many distances in planar graphs. *Algorithmica*, 62(1-2):361–381, February 2012. doi:10.1007/s00453-010-9459-0.

[CFS19] Vincent Cohen-Addad, Andreas Feldmann, and David Saulpic. Near-linear time approximations schemes for clustering in doubling metrics. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 540–559, November 2019. doi:10.1109/FOCS.2019.00041.

[CGH16] Yun Kuen Cheung, Gramoz Goranci, and Monika Henzinger. Graph minors for preserving terminal distances approximately – lower and upper bounds. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP*, pages 131:1–131:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.131.

[CGH+20] Li Chen, Gramoz Goranci, Monika Henzinger, Richard Peng, and Thatchaphol Saranurak. Fast dynamic cuts, distances and effective resistances via vertex sparsifiers. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1135–1146, Durham, NC, USA, November 2020. IEEE. doi:10.1109/FOCS46700.2020.00109.

[CGMW18] Hsien-Chih Chang, Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Near-Optimal Distance Emulator for Planar Graphs. In *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ESA.2018.16.

[Cha52] A. Charnes. Optimality and degeneracy in linear programming. *Econometrica*, 20(2):160–170, 1952. doi:10.2307/1907845.

[Che18] Yun Kuen Cheung. Steiner point removal: Distant terminals don't (really) bother. In *29th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 1353–1360. SIAM, 2018. doi:10.1137/1.9781611975031.89.

[Chu12] Julia Chuzhoy. On vertex sparsifiers with Steiner nodes. In *44th symposium on Theory of Computing*, pages 673–688. ACM, 2012. doi:10.1145/2213977.2214039.

[CLLM10] Moses Charikar, Tom Leighton, Shi Li, and Ankur Moitra. Vertex sparsifiers and abstract rounding algorithms. In *51st Annual Symposium on Foundations of Computer Science*, pages 265–274. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.32.

[CO20] Hsien-Chih Chang and Tim Ophelders. Planar emulators for Monge matrices. In *Proceedings of the 32nd Canadian Conference on Computational Geometry (CCCG 2020)*, pages 141–147, 2020. URL: https://vga.usask.ca/cccg2020/papers/Proceedings.pdf.

[CS19] Timothy M. Chan and Dimitrios Skrepetos. Faster approximate diameter and distance oracles in planar graphs. *Algorithmica*, 81(8):3075–3098, August 2019. doi:10.1007/s00453-019-00570-z.

[CSWZ00] S. Chaudhuri, K. V. Subrahmanyam, F. Wagner, and C. D. Zaroliagis. Computing mimicking networks. *Algorithmica*, 26:31–49, 2000. doi:10.1007/s004539910003.

[CXKR06] T. Chan, Donglin Xia, Goran Konjevod, and Andrea Richa. A tight lower bound for the Steiner point removal problem on trees. In *9th International Workshop on Approximation, Randomization, and Combinatorial Optimization*, volume 4110 of *Lecture Notes in Computer Science*, pages 70–81. Springer, 2006. doi:10.1007/11830924_9.

[DOW55] George Dantzig, Alexander Orden, and Philip Wolfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195, June 1955. doi:10.2140/pjm.1955.5.183.

[EFL18] Jeff Erickson, Kyle Fox, and Luvsandondov Lkhamsuren. Holiest minimum-cost paths and flows in surface graphs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing - STOC 2018*, pages 1319–1332, Los Angeles, CA, USA, 2018. ACM Press. doi:10.1145/3188745.3188904.

[EGK+14] M. Englert, A. Gupta, R. Krauthgamer, H. Räcke, I. Talgam-Cohen, and K. Talwar. Vertex sparsifiers: New results from old techniques. *SIAM Journal on Computing*, 43(4):1239–1262, 2014. arXiv:1006.4586, doi:10.1137/130908440.

[EK13] David Eisenstat and Philip N. Klein. Linear-time algorithms for max flow and multiple-source shortest paths in unit-weight planar graphs. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13*, page 735, Palo Alto, California, USA, 2013. ACM Press. doi:10.1145/2488608.2488702.

[Fil18] Arnold Filtser. Steiner point removal with distortion $O(\log k)$. In *29th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, page 1361–1373. Society for Industrial and Applied Mathematics, 2018. doi:10.1137/1.9781611975031.90.

[FKT19] Arnold Filtser, Robert Krauthgamer, and Ohad Trabelsi. Relaxed Voronoi: A simple framework for terminal-clustering problems. In *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*, volume 69, pages 10:1–10:14, 2019. doi:10.4230/OASIcs.SOSA.2019.10.

[FR06] Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72(5):868–889, August 2006. doi:10.1016/j.jcss.2005.05.007.

[Fre87] Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, December 1987. doi:10.1137/0216064.

[GHP20] Gramoz Goranci, Monika Henzinger, and Pan Peng. Improved guarantees for vertex sparsification in planar graphs. *SIAM Journal on Discrete Mathematics*, 34(1):130–162, 2020. doi:10.1137/17M1163153.

[GR16] Gramoz Goranci and Harald Räcke. Vertex sparsification in trees. In *Approximation and Online Algorithms – 14th International Workshop, WAOA*, pages 103–115, 2016. doi:10.1007/978-3-319-51741-4_9.

[GRST21] Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. The expander hierarchy and its applications to dynamic graph algorithms. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*, pages 2212–2228. SIAM, 2021. doi:10.1137/1.9781611976465.132.

[Gup01] Anupam Gupta. Steiner points in tree metrics don't (really) help. In *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 220–227. SIAM, 2001. URL: http://dl.acm.org/citation.cfm?id=365411.365448.

[HKNR98] Torben Hagerup, Jyrki Katajainen, Naomi Nishimura, and Prabhakar Ragde. Characterizing multi-terminal flow networks and computing flows in networks of small treewidth. *J. Comput. Syst. Sci.*, 57:366–375, 1998. doi:10.1006/jcss.1998.1592.

[HKRS97] Monika R. Henzinger, Philip Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.*, 55(1):3–23, 1997. URL: http://dx.doi.org/10.1006/jcss.1997.1493.

[HM94] David Hartvigsen and Russell Mardon. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *SIAM Journal on Discrete Mathematics*, 7(3):403–418, May 1994. doi:10.1137/S0895480190177042.

[INSW11] Giuseppe F. Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proceedings of the 43rd annual ACM symposium on Theory of computing (STOC '11)*, pages 313–322, San Jose, California, USA, 2011. doi:10.1145/1993636.1993679.

[IS79] Alon Itai and Yossi Shiloach. Maximum flow in planar networks. *SIAM J. Comput.*, 8(2):16, May 1979. doi:10.1137/0208012.

[KKN15] Lior Kamma, Robert Krauthgamer, and Huy L. Nguyen. Cutting corners cheaply, or how to remove Steiner points. *SIAM Journal on Computing*, 44(4):975–995, 2015. doi:10.1137/140951382.

[Kle05] Philip N. Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 146–155, 2005. doi:10.5555/1070432.1070454.

[KMS13] Philip N. Klein, Shay Mozes, and Christian Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13*, page 505, Palo Alto, California, USA, 2013. ACM Press. doi:10.1145/2488608.2488672.

[KNZ14] Robert Krauthgamer, Huy L. Nguyen, and Tamar Zondiner. Preserving terminal distances using minors. *SIAM Journal on Discrete Mathematics*, 28(1):127–141, 2014. doi:10.1137/120888843.

[KPZ17] Nikolai Karpov, Marcin Pilipczuk, and Anna Zych-Pawlewicz. An exponential lower bound for cut sparsifiers in planar graphs. *12th International Symposium on Parameterized and Exact Computation, IPEC*, pages 24:1–24:11, 2017. doi:10.4230/LIPIcs.IPEC.2017.24.

[KR13] Robert Krauthgamer and Inbal Rika. Mimicking networks and succinct representations of terminal cuts. In *24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1789–1799. SIAM, 2013. doi:10.1137/1.9781611973105.128.

[KR14] Arindam Khan and Prasad Raghavendra. On mimicking networks representing minimum terminal cuts. *Inf. Process. Lett.*, 114(7):365–371, 2014. doi:10.1016/j.ipl.2014.02.011.

[KR20] Robert Krauthgamer and Havana (Inbal) Rika. Refined vertex sparsifiers of planar graphs. *SIAM Journal on Discrete Mathematics*, 34(1):101–129, 2020. doi:10.1137/17M1151225.

[KS98] P. N. Klein and S. Subramanian. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica*, 22(3):235–249, November 1998. doi:10.1007/PL00009223.

[KZ12] Robert Krauthgamer and Tamar Zondiner. Preserving terminal distances using minors. In *39th International Colloquium on Automata, Languages, and Programming*, volume 7391 of *Lecture Notes in Computer Science*, pages 594–605. Springer, 2012. doi:10.1007/978-3-642-31594-7_50.

[MM16] Konstantin Makarychev and Yury Makarychev. Metric extension operators, vertex sparsifiers and Lipschitz extendability. *Israel Journal of Mathematics*, 212(2):913–959, 2016. doi:10.1007/s11856-016-1315-8.

[MNNW18] Shay Mozes, Cyril Nikolaev, Yahav Nussbaum, and Oren Weimann. Minimum cut of directed planar graphs in $O(n \log \log n)$ time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 477–494, New Orleans, Louisiana, January 2018. arXiv:1512.02068.

[Moi09] Ankur Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *50th Annual Symposium on Foundations of Computer Science*, FOCS, pages 3–12. IEEE, 2009. doi:10.1109/FOCS.2009.28.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, March 1987. doi:10.1007/BF02579206.

[Rei81] John H Reif. Minimum s-t cut of a planar undirected network in $O(n \log^2(n))$ time. page 12, 1981. doi:10.1007/3-540-10843-2_5.

[SA12] R. Sharathkumar and Pankaj K. Agarwal. Algorithms for the transportation problem in geometric settings. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 306–317. Society for Industrial and Applied Mathematics, Philadelphia, PA, January 2012. doi:10.1137/1.9781611973099.29.

[Tho04] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, November 2004. doi:10.1145/1039488.1039493.

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

A Missing Proofs in Section 2 and Section 3

A.1 Proof of Lemma 2.2

Let $w : E(G) \to \mathbb{R}^+$ be the edge weight function of graph G. We slightly perturb w to obtain another function $w' : E(G) \to \mathbb{R}^+$, such that for every pair P, P' of distinct paths in G: $w'(P) \neq w'(P')$; and if $w'(P) > w'(P')$, then $w(P) \geq w(P')$. Therefore, for each pair v, v' of vertices in G, there is a unique v-v' shortest path in G under the weight function w', and this path is also a v-v' shortest path in G under the weight function w [MVV87, Cab12].

The algorithm uses the technique of divide-and-conquer. We now describe the recursive step.

We first construct an auxiliary planar graph H as follows. Its vertex set is $V(H) = T$, and its edge set $E(H)$ contains, for each pair $(t_1, t_2) \in \mathcal{M}$, an edge connecting t_1 to t_2. Graph H inherits a planar embedding from G and is therefore an outerplanar graph. Denote by $\mathcal{F}$ the set of bounded faces in H lying inside a disc D. We construct a graph R as follows. Its vertex set is $V(R) = \{u_F \mid F \in \mathcal{F}\}$, and its edge set $E(R)$ contains, for every pair $F, F' \in \mathcal{F}$, an edge $(u_F, u_{F'})$ if and only if faces F and F' share a segment of non-zero length on their boundaries. It is easy to verify that R is a tree, and $|V(R)| = |\mathcal{M}| + 1$. (In other words, R is the *weak-dual* of an outerplanar graph.) We can now efficiently compute a vertex u_F of R, such that every connected component of graph $R \setminus \{u_F\}$ contains no more than $|V(R)|/2$ vertices. Denote this vertex by u_{F^*}. Consider now the face F^* of H. Since in graph R, every connected component of graph $R \setminus \{u_F\}$ contains no more than $|V(R)|/2$ vertices, it is easy to see that we can find a pair t_i, t_j of terminals on the intersection of the boundary of D and the boundary of F^*, such that, if we draw a straight line segment connecting t_i, t_j, and denote by D_1, D_2 the discs obtained by cutting D along this segment, then each edges of H is drawn either inside D_1 or inside D_2, and each of D_1, D_2 contains the image of at most $3/4$-fractions of edges in H.

Consider now the one-hole instance (G, T). We compute a t_i-t_j shortest path P in G, and cut the graph G into two subgraphs G_1, G_2 along path P (so $G_1 \cap G_2 = P$). Define $\mathcal{M}_1$ to be the subset of $\mathcal{M}$ that contains all pairs whose corresponding edge in graph H is drawn inside D_1 in H, and we define subset $\mathcal{M}_2$ similarly, so sets $\mathcal{M}_1, \mathcal{M}_2$ partition $\mathcal{M}$, and $|\mathcal{M}_1|, |\mathcal{M}_2| \leq (3/4) \cdot |\mathcal{M}|$. We now recurse on graph G_1 for computing the shortest paths connecting pairs of $\mathcal{M}_1$ and graph G_2 for computing the shortest paths connecting pairs of $\mathcal{M}_2$. This completes the description of the algorithm.

It is easy to verify that the running time of the algorithm is $O(\log |\mathcal{M}| \cdot |E(G)|)$, since in every recursive layer, every edge of the original graph G appears in at most two of the graphs that lie on this layer. To complete the proof of Theorem 2.2, it suffices to show that, in a recursive step described above, for every pair $(t_1, t_2) \in \mathcal{M}_1$, the unique shortest path in G under w' lies entirely in graph G_1 (the case for $\mathcal{M}_2$ and G_2 is symmetric), and the set of resulting shortest paths that we computed is well-structured.

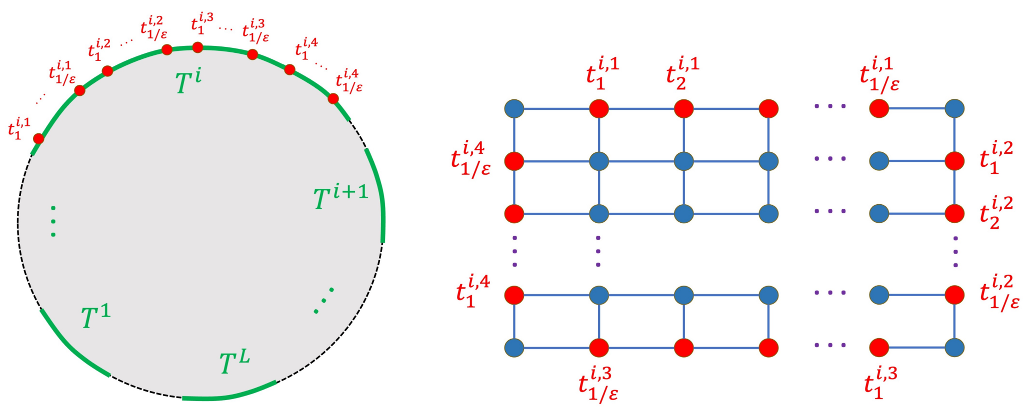
Assume for contradiction that the t_1-t_2 shortest-path P' in G does not lie entirely in G_1. We view P' as being directed from t_1 to t_2. Let v (v', resp.) be the first (last, resp.) vertex of P' that lies on P and denote by $\hat{P}$ ($\hat{P}'$, resp.) the subpath of P (P', resp.) between v and v'. Therefore, some inner vertex of $\hat{P}'$ does not belong to G_1 and therefore does not belong to P, and so $\hat{P} \neq \hat{P}'$. However, since both P and P' are shortest paths under w', $w'(\hat{P}) = w'(\hat{P}')$, a contradiction to the fact that every pair of distinct paths have different weight in w'. Via similar arguments we can also show that set of resulting shortest paths that we computed is well-structured.

A.2 Proof of Theorem 3.2

In this subsection we provide the proof of Theorem 3.2. Our example is inspired by the hard example constructed in [KNZ14]. Assume that $1/\varepsilon$ is an integer and k is a multiple of $1/\varepsilon$. This will only cause an additional constant factor in the size bound and will not influence the bound in Theorem 3.2.

We first construct a circular ordering σ and a metric d on the terminals. From [CO20], if d satisfies the Monge property (under the circular ordering σ), then there exists a one-hole instance (G,T) with terminals in T appearing on the boundary in the order σ.

The set T is partitioned into $L=\varepsilon k/4$ groups $T=\bigcup_{1\le i\le L}T^i$, where each group contains $4/\varepsilon$ terminals. Each group T^i is then partitioned into four subgroups $T^i=T^{i,1}\cup T^{i,2}\cup T^{i,3}\cup T^{i,4}$, each containing $1/\varepsilon$ terminals. We denote $T^{i,j}=\{t^{i,j}_1,\dots,t^{i,j}_{1/\varepsilon}\}$, for each $1\le j\le 4$. The circular ordering σ on terminals of T is defined as follows. The groups $T^1,\dots,T^L$ appear clockwise in this order; within each group T^i, the subgroups $T^{i,1},T^{i,2},T^{i,3},T^{i,4}$ appear clockwise in this order; and within each subgroup $T^{i,j}$, the vertices $t^{i,j}_1,\dots,t^{i,j}_{1/\varepsilon}$ appear clockwise in this order. See Figure 7(a) for an illustration. The metric d on T is defined as follows. For every pair t,t' of terminals that belong to different groups, $d(t,t')=1/\varepsilon^2$. Consider now a group T_i. The metric between terminals in T_i is defined as follows. Consider the $(\frac{1}{\varepsilon}+2)\times(\frac{1}{\varepsilon}+2)$ grid with unit edge weight. We place each terminal in T at a boundary vertex of H, in the way shown in Figure 7(b). Now for each pair $t^{i,j}_r,t^{i,j'}_{r'}$ of terminals in T^i, we define $d(t^{i,j}_r,t^{i,j'}_{r'})=\mathsf{dist}_H(t^{i,j}_r,t^{i,j'}_{r'})$. It is easy to verify that d is a metric and satisfies the Monge property.



(a) An illustration of ordering σ. (b) An illustration of metric d within a group T^i of terminals.

Figure 7. Illustrations of circular ordering σ and metric d within a group T^i of terminals.

Consider now a one-hole instance (G',T) such that the circular ordering in which terminals in T appear on the outer boundary of G' is σ and for each pair $t,t'\in T$, $e^{-\varepsilon/3}\cdot\mathsf{dist}_{G'}(t,t')\le d(t,t')\le e^{-\varepsilon/3}$. For each $1\le i\le L$, we define G'_i to be the subgraph of G' induced by the set of all vertices in G' that have distance at most $10/\varepsilon$ from terminal $t^{i,1}_1$. Since in d, the distance between every pair of terminals in $\{t^{1,1}_1,\dots,t^{L,1}_1\}$ is $1/\varepsilon^2$, it is easy to see that the graphs $\{G'_1,\dots,G'_L\}$ are mutually vertex-disjoint. On the other hand, it is easy to verify that, for every $1\le i\le L$ and every pair t,t' of terminals in T^i, the shortest path in G' connecting t to t' is entirely contained in G'_i. Therefore, for each $1\le i\le L$, (G'_i,T^i) is an aligned $\varepsilon/3$-emulator for (G,T^i). From similar arguments in [KNZ14], we get that $|V(G'_i)|\ge\Omega(|T^i|^2)=\Omega(1/\varepsilon^2)$. Therefore, $|V(G')|\ge\sum_{1\le i\le L}|V(G'_i)|\ge L\cdot\Omega(1/\varepsilon^2)=\Omega(k/\varepsilon)$. This shows that any aligned $(\varepsilon/3)$-emulator for (G,T) has size at least $\Omega(k/\varepsilon)$. Theorem 3.2 now follows by scaling.

A.3 Calculations for size and error bounds in Section 3

For convenience, we denote $\lambda=\lambda^*$. We prove the following observations.

Observation A.1. *Let $r_1,\dots,r_t$ be a sequence of integers, such that $r_1\le k$, $r_t\ge\lambda$, and for each $1\le i\le t-1$, $r_i\ge(10/9)\cdot r_{i+1}$. Then $\sum_{1\le i\le t}(\log_{(10/9)}r_i)^{-2}\le 1/(\log_{(10/9)}\lambda-1)$.*

Proof: Since for each $1 \le i \le t-1$, $r_{i+1} \le (9/10)\cdot r_i$, $\log_{(10/9)} r_{i+1} \le \log_{(10/9)} r_i - 1$. Therefore,

$$\sum_{1\le i\le t} \frac{1}{(\log_{(10/9)} r_i)^2} \le \sum_{j\ge \log_{(10/9)}\lambda} \frac{1}{j^2} \le \sum_{j\ge \log_{(10/9)}\lambda} \left(\frac{1}{j-1}-\frac{1}{j}\right) \le \frac{1}{\log_{(10/9)}\lambda - 1}. \qquad \square$$

Observation A.2. *Let $r_1,\ldots,r_t$ be a sequence of integers, such that $r_1 \le k$, $r_t \ge \lambda$, and for each $1 \le i \le t-1$, $r_i \ge (10/9)\cdot r_{i+1}$. Then $\sum_{1\le i\le t} (\log r_i)^4/r_i^{0.1} \le 101(\log\lambda)^4/\lambda^{0.1}$.*

Proof: Consider any index $1 \le i \le t-1$. Denote $x = \log r_i/\log r_{i+1}$, so $r_i = (r_{i+1})^x$. Assume first that $x < 1+10^{-100}$, then since $r_i \ge (10/9)\cdot r_{i+1}$, we get that

$$\left(\frac{(\log r_i)^4}{r_i^{0.1}}\right)\Big/\left(\frac{(\log r_{i+1})^4}{r_{i+1}^{0.1}}\right) = \frac{r_{i+1}^{0.1}}{r_i^{0.1}}\cdot\left(\frac{\log r_i}{\log r_{i+1}}\right)^4 \le \frac{99}{100}\cdot x^4 \le \frac{100}{101}.$$

Assume now that $x \ge 1+10^{-100}$, then since $r_{i+1} \ge \lambda$ and from the definition of λ,

$$\left(\frac{(\log r_i)^4}{r_i^{0.1}}\right)\Big/\left(\frac{(\log r_{i+1})^4}{r_{i+1}^{0.1}}\right) = \frac{r_{i+1}^{0.1}}{r_i^{0.1}}\cdot\left(\frac{\log r_i}{\log r_{i+1}}\right)^4 = \frac{x^4}{(r_{i+1})^{\frac{x-1}{10}}} \le \frac{x^4}{\lambda^{\frac{x-1}{10}}} \le \frac{100}{101}.$$

and so

$$\sum_{1\le i\le t} \frac{(\log r_i)^4}{r_i^{0.1}} < \frac{(\log\lambda)^4}{\lambda^{0.1}}\cdot\left(1+\frac{100}{101}+\left(\frac{100}{101}\right)^2+\cdots\right) \le \frac{101(\log\lambda)^4}{\lambda^{0.1}}. \qquad \square$$

A.4 Proof of Claim 4.1

Item 1 of Claim 4.1. We define the graph $\tilde{H}$ as the union of (i) all paths in $\mathcal{P}$; and (ii) the cycle that connects all vertices of U in the order that they appear on the outer-boundary of the drawing associated with H, so $\tilde{H}$ is a planar graph, and the drawing of H naturally induces a planar drawing of $\tilde{H}$. Let $\tilde{H}'$ be the graph obtained from $\tilde{H}$ by suppressing all degree-2 vertices, so the planar drawing of $\tilde{H}$ naturally induces a planar drawing of $\tilde{H}'$. Since $\tilde{H}'$ has no degree-2 vertices, the number of faces, edges and vertices are all within a constant factor. Therefore, to show that the number of branch vertices is $O(|U|)$, it suffices to show that the number of vertices in $\tilde{H}'$ is $O(|U|)$, and therefore it suffices to show that the number of faces in the planar drawing of $\tilde{H}'$ is $O(|U|)$.

We first construct an outerplanar graph X on U as follows. The edge set of X is the union of (i) all edges of the cycle that connects all vertices of U in the order that they appear on the outerface; and (ii) for each path in $\mathcal{P}$, an edge connecting its endpoints in U. Clearly, X has $|U|$ vertices and $O(|U|)$ edges. The circular ordering on vertices of U naturally defines a drawing of X. Clearly, the number of faces in this drawing is $O(|U|)$, and moreover, the total size of all faces is $O(|U|)$ (where the size of a face is the number of vertices that lie on the boundary of the face).

Let F be a face in the drawing of X defined above. We denote by $|F|$ the number of vertices that lie on the boundary of F. We now show that this face gives birth to at most $O(|F|)$ faces in $\tilde{H}'$. Let Y be a graph defined as follows. The vertex set $V(Y)$ contains, for each boundary edge e of F, a node y_e representing e. The edge set $E(Y)$ contains, for every pair $y_e, y_{e'}$ of vertices, an edge connecting them iff the corresponding paths (in $\mathcal{P}$) of edge e and e' either share an edge or share an internal vertex that does not belong to any other path in $\mathcal{P}$. Since $\mathcal{P}$ is well-structured and non-crossing, the graph Y is an outerplanar graph, and so $|E(Y)| = O(|V(Y)|) = O(|F|)$. Since the number of faces in $\tilde{H}'$ that F gives birth to is at most the number of edges in Y plus one, we get that the number of faces in $\tilde{H}'$ that F gives birth is at most $O(|F|)$.

Therefore, the total number of faces in $\tilde{H}'$ in at most a constant times the total size of all faces in X, which is $O(|U|)$. This completes the proof of 1.

Item 2 of Claim 4.1. For convenience, we rename $Y \setminus Y^*$ as Y. In other words, set Y only contains vertices that belong to exactly two paths of $\mathcal{P}$, so each vertex of Y is contained in at most two instances in $\mathcal{H}$, contributing at most 2 to the sum $\sum_{(H_R,U_R)\in\mathcal{H}:\,|U_R|\geq\lambda}|U_R|$.

We denote by $\mathcal{R}$ the set of regions in H obtained by the procedure SPLIT. Recall that, for each region $R \in \mathcal{R}$, set U_R contains all branch vertices and vertices of $U \cup Y$ that lie on the boundary of R. Therefore, if we denote by U'_R the set that contains all branch vertices and vertices of U lying on the boundary of R, then it suffices to show that

$$\sum_{R\in\mathcal{R}:\,|U'_R|\geq\lambda/2} |U'_R| \leq |U|\cdot\left(1+O\left(\frac{1}{\lambda}\right)\right). \tag{3}$$

This is because, for each $R \in \mathcal{R}$, if $|U'_R| < \lambda/2$ while $|U_R| \geq \lambda$, then $|Y \cap U_R| \geq \lambda/2 \geq |U'_R|$ and so $|U_R| \leq 2\cdot|Y \cap U_R|$, and since every vertex of Y appears on the boundaries of at most two regions in $\mathcal{R}$, we get that

$$\sum_{R\in\mathcal{R}:\,|U'_R|<\lambda/2,\,|U_R|\geq\lambda} |U_R| \leq \sum_{R\in\mathcal{R}:\,|U'_R|<\lambda/2,\,|U_R|\geq\lambda} 2\cdot|Y\cap U_R| \leq O(|Y|).$$

Combined with Inequality 3 and the above discussion, this completes the proof of Claim 4.1.

The remainder of this section is dedicated to the proof of Inequality 3. Using similar arguments in the proof of Claim 4.4, we can show that it suffices to prove Inequality 3 when no vertex of U is a cut vertex of H. In other words, when we traverse the outerface of graph H, every terminal in U will be visited once, and so we get a circular ordering on terminals in U.

Denote $\lambda' := \lambda/2$. We say that a region $R \in \mathcal{R}$ is *big* if $|U'_R| \geq \lambda'$, otherwise we say it is *small*. We need the following observation: if all regions in $\mathcal{R}$ are big, then Claim 4.1 holds.

Observation A.3. *Let $\hat{\lambda} > 10$ be any integer. If for all $R \in \mathcal{R}$, $|U'_R| \geq \hat{\lambda}$, then*

$$\sum_{R\in\mathcal{R}} |U'_R| \leq |U|\cdot\left(1+O(1/\hat{\lambda})\right).$$

Proof: Denote $U = \{u_1,\dots,u_r\}$, where the terminals are indexed according to the circular ordering in which they appear on the outerface of H. We define a graph W as follows. We start from the graph obtained by taking the union of all paths in $\mathcal{P}$. We then suppress all degree-2 non-terminals. Finally, we add the cycle $(u_1,\dots,u_r,u_1)$. Clearly, W is a planar graph, and the planar drawing of H naturally defines a drawing of W: start with the planar drawing of all paths in $\mathcal{P}$ induced by the planar drawing of H, contracting degree-2 non-terminals, and finally draw every edge (u_i,u_{i+1}) along the boundary of the disc in which the one-hole instance (H,U) lies in. Note that each region $R \in \mathcal{R}$ corresponds to a face in the planar drawing of W, that we denote by F_R. Moreover, the vertices lying on the boundary of F_R are exactly the vertices of U'_R.

Consider now the dual graph W^* of W with respect to the planar drawing defined above. Clearly, every node in W^* corresponds to a region in $\mathcal{R}\cup\{R_\infty\}$, where R_∞ is the region outside the disc in which the one-hole instance (H,U) lies in. We denote $V(W^*) = \{v_R \mid R \in \mathcal{R}\}\cup\{v_\infty\}$.

On the one hand, for each $R \in \mathcal{R}$, $|U'_R|$ is equal to the number of edges on the boundary of face F_R, which is then equal to the degree of vertex v_R in W^*. Therefore,

$$\sum_{R\in\mathcal{R}} |U'_R| = \sum_{v\in V(W^*),\,v\neq v_\infty} \deg_{W^*}(v).$$

Recall that every region $R \in \mathcal{R}$ satisfies that $|U'_R| \geq \hat{\lambda}$, so $\deg_{W^*}(v) \geq \hat{\lambda}$ for all $v \in V(W^*), v \neq v_\infty$.

On the other hand, since the paths in $\mathcal{P}$ are well-structured and non-crossing, and we have suppressed all degree-2 vertices, it is easy to observe that the subgraph of W^* induced by all vertices of $\{v_R \mid R \in \mathcal{R}\}$ is a simple graph. In other words, all edges that have a parallel copy in W^* must be incident to v_∞.

Since the number of edges in W^* incident to v_∞ is $|U|$, if we subdivide every edge incident to v_∞ by a new vertex, then the resulting graph, which we denote by $\hat{W}^*$, is a planar simple graph, and so $|E(\hat{W}^*)| \le 3 \cdot |V(\hat{W}^*)|$. Therefore,

$$|U| + 2 \cdot |U| + \sum_{v \in V(W^*), v \neq v_\infty} \deg_{W^*}(v) \le 2 \cdot |E(\hat{W}^*)| \le 6 \cdot |V(\hat{W}^*)| \le 6 \cdot (|U| + |V(W^*)|),$$

so $3|U| + (|V(W^*)| - 1) \cdot \hat{\lambda} \le 6(|U| + |V(W^*)|)$, and so $|V(W^*)| \le (3|U| + \hat{\lambda})/(\hat{\lambda} - 6) \le O(|U|/\hat{\lambda})$.

Altogether, we get that

$$\sum_{R \in \mathcal{R}} |U'_R| = \sum_{v \in V(W^*), v \neq v_\infty} \deg_{W^*}(v) = |U| + \sum_{v \in V(W^*), v \neq v_\infty} \deg_{W^* \setminus v_\infty}(v) \le |U| + O(|U|/\hat{\lambda}).$$ □

We now proceed to prove Inequality 3 using Theorem A.3. Let W be the plane graph defined in the proof of Theorem A.3, and we say that graph W is *generated* by the set $\mathcal{P}$ of paths. We prove the following observation.

Observation A.4. *Let P be a path in $\mathcal{P}$, let F be a face, and let C be the boundary cycle of F. Then either $P \cap C = \emptyset$, or the intersection between P and C is a subpath of both P and C.*

Proof: Assume that $P \cap C \neq \emptyset$; and furthermore, $P \cap C$ contains at least two vertices (since otherwise a single vertex is a subpath of both P and C, and we are done). Assume for contradiction that $P \cap C$ is not a subpath of P. It is easy to verify that there are two vertices u, u', such that $u, u' \in V(P) \cap V(C)$, but every vertex in P between u and u' does not belong to C. Denote by P' the subpath of P connecting u to u'. Note that u, u' separates C into two path, that we denote by C_1, C_2. Assume without loss of generality that the region surrounded by $P' \cup C_1$ does not contain the outerface. Let e be an edge of C_1. Since graph W is generated by paths in $\mathcal{P}$, edge e must belong to some path $P'' \in \mathcal{P}$. However, since both endpoints of P'' lie outside of the region surrounded by $P' \cup C_1$, and since C_1 is a segment of a face, path P'' must contain two vertices of P', and the subpath of P'' between these two vertices contains the edge e, which does not belong to P'. Therefore, paths P' and P'' are not well-structured, a contradiction. □

Let P be a path and let F be a face, such that P and the boundary cycle C_F of F intersect, and the intersection $P \cap C_F$ is a subpath of both P and C_F. Let u, u' be the endpoints of this subpath. We define $P_{\oplus F}$ as the path obtained from P by replacing the subpath between u and u' with the other subpath of C_F connecting u to u' that does not belong to P.

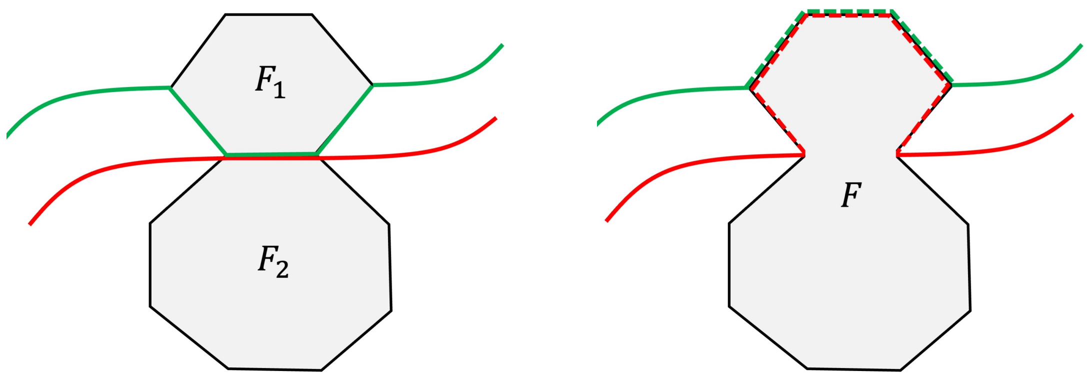
Recall that we only need to prove Inequality 3 for W, where the left hand side $\sum_{R \in \mathcal{R}:\, |U'_R| \ge \lambda'} |U'_R|$, which we denote by $\mathsf{bs}(W)$, is the sum of the sizes of all big faces. We will first iteratively modify W until we are unable to do so, such that the value $\mathsf{bs}(W)$ never decreases. Then we will show that the value $\mathsf{bs}(\tilde{W})$ of the resulting graph $\tilde{W}$ is bounded by $|U| \cdot (1 + O(1/\lambda'))$ using Theorem A.3.

We now describe the algorithm that iteratively modifies the graph W. Throughout, we maintain a plane graph $\hat{W}$, that is initialized to be W, and a set $\hat{\mathcal{P}}$ of paths, that is initialized to be $\mathcal{P}$. We will always ensure that $\hat{\mathcal{P}}$ is a well-structured set of paths, and graph $\hat{W}$ is generated by $\hat{\mathcal{P}}$. When the algorithm proceeds, the plane graph $\hat{W}$ evolves, and so does the set of faces in its planar drawing. We say that a face is big (small, resp.) iff its boundary contains at least (less than, resp.) λ' vertices.

We say that a tuple (e, F_1, F_2) *critical*, iff (i) e is an edge in $\hat{W}$, F_1 is a small face, and F_2 is a big face, such that e is incident to F_1 and F_2; and (ii) no vertex of F_1 is incident to any other big face than F_2. We say that a pair (P, P') of paths in $\hat{\mathcal{P}}$ is a *blocking pair* for a critical tuple (e, F_1, F_2), iff (i)

$e \in E(P)$, $e \notin E(P')$; and (ii) the pair $P_{\oplus F_1}, P'$ of paths are not well-structured. We distinguish between the following cases.

Case 1: There is a critical tuple (e, F_1, F_2) with no blocking pairs, and the degree of at least one endpoint of e is at least 4. In this case, we simply replace each path $P \in \hat{\mathcal{P}}$ that contains the edge e with path $P_{\oplus F_1}$, and then update $\hat{W}$ to be the graph generated by the resulting set $\hat{\mathcal{P}}$ of paths. See Figure 8.



(a) Before: Faces F_1 and F_2 share an edge e. Two paths containing e in $\hat{\mathcal{P}}$ are shown in green and red.

(b) After: Faces F_1 and F_2 are merged into F. The modified segment of two paths are shown in dashed lines.

Figure 8. An illustration of graph and path modification in Case 1.

It is clear that the invariant that $\hat{W}$ is generated by $\hat{\mathcal{P}}$ still holds in this case. Also, since there is no blocking pair for the critical tuple (e, F_1, F_2), the resulting path set $\hat{\mathcal{P}}$ is still well-structured. Moreover, since no path in the resulting set $\hat{\mathcal{P}}$ contains the edge e, the resulting graph $\hat{W}$ no longer contains the edge e, either. Since the resulting graph $\hat{W}$ may not contain any new edge, the number of faces in $\hat{W}$ decreases by at least 1 (as faces F_1 and F_2 are merged into a single face). We now show that the value $\mathsf{bs}(\hat{W})$ does not decrease.

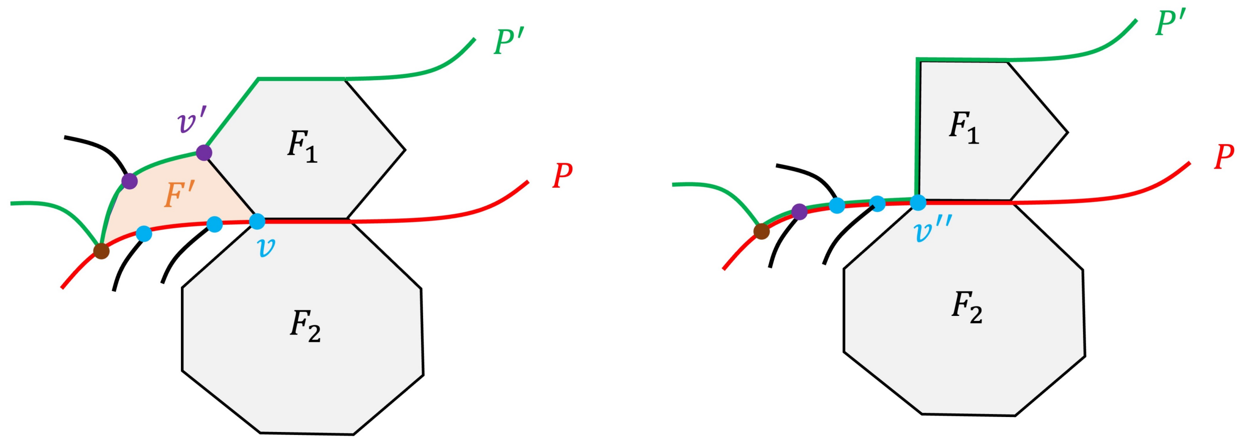
First, since the modification of paths in $\hat{\mathcal{P}}$ only involves edges and vertices in C_{F_1}, the boundary cycle of face F_1, the graph $\hat{W} \setminus C_{F_1}$ remain unchanged, so every big face other than F_2 remain unchanged as well, and so is their contribution to $\mathsf{bs}(\hat{W})$. Second, consider the resulting face F into which F_1 and F_2 are merges. Note that F contains all original vertices of F_2 as branch vertices. This is because all vertices of $F_2 \setminus F_1$ remain unchanged, and since at least one of the endpoints of e has degree at least 4 in $\hat{W}$ before this iteration, this endpoint remain as branch vertices in the resulting graph $\hat{W}$, and the face F contains at least one more branch vertex. Therefore, face F contains at least the same number of branch vertices as the previous big face F_2. It follows that the value $\mathsf{bs}(\hat{W})$ does not decrease.

Case 2: There is a critical tuple (e, F_1, F_2) with no blocking pairs, where F_1 contains more than 3 vertices, and the degrees of both endpoints of e are 3. In this case, we update the path set $\hat{\mathcal{P}}$ and graph $\hat{W}$ in the same way as the previous case. Via similar arguments, we can show that the number of faces decreases by at least 1, and the value $\mathsf{bs}(\hat{W})$ does not decrease.

Case 3: There is a critical tuple (e, F_1, F_2) and a blocking pair (P, P') for it. Since paths P and P' are well-structured, but paths $P_{\oplus F_1}$ and P' are not, from Theorem A.4, there must be two disjoint subpaths P'_1, P'_2 of P', such that $P'_1 = P \cap P'$ and $P'_2 = C_{F_1} \cap P'$. We first give both paths P and P' a direction, such that P'_1 appears before P'_2 on P', and P'_1 appears before edge e on P. Let u be the last vertex of P'_1, let v' be the first vertex of P'_2, and let v be the first vertex of $C_{F_1} \cap P$ that appears on P.

We first show that v and v' must be adjacent on C_{F_1}. Assume not, let X be the segment of C_{F_1} between u and v' that does not contain e, and let x be an inner vertex of X. Since $\deg(x) \geq 3$, we let e_x be an edge incident to x, such that $e_x \notin C_{F_1}$. Consider the region R surrounded by (i) the subpath of P between u and v; (ii) the subpath of P' between u and v'; and (iii) path X. It is clear that e_x must lie entirely in R. On the other hand, let P_x be a path in $\hat{\mathcal{P}}$ that contains the edge e_x, so both endpoints of P_x lie outside R. Since paths in $\hat{\mathcal{P}}$ are non-crossing and well-structured, path P_x must exit region R at v and v', but since $e_x \notin E(C_{F_1})$, the intersection between P_x and C_{F_1} is neither a subpath of C_{F_1} nor a subpath of P_x, a contradiction to Theorem A.4. Via similar arguments, we can show that no edge may lie inside the interior of region R. In other words, region R is in fact a face, which we denote by F' (see Figure 9(a)). Moreover, since vertices v, v' are not incident to any other big faces, F' is a small face.

We now "suppress" the face F' as follows. We first contract the edge (v, v') of C_{F_1}, while identifying vertices v and v' into a single vertex v''. We then "identify" the subpath of P between u and v (which we denote by $\tilde{P}$) with the subpath of P' between u and v' (which we denote by $\tilde{P}'$). Specifically, if originally $\tilde{P} = (u, y_1, \ldots, y_s, v)$ and $\tilde{P}' = (u, y'_1, \ldots, y'_t, v')$, then we replace these two paths with a new path $\tilde{P}'' = (u, y_1, \ldots, y_s, y'_1, \ldots, y'_t, v'')$, and we do not modify the incident edges of any y_i or y'_j (see Figure 9(b)). We update $\hat{W}$ to be the resulting graph after this step.



(a) Before: Vertex u is shown in brown. Paths P, P' are shown in red, green respectively. Face F' is shown in orange.

(b) After: Face F' is suppressed, vertices v, v' are contracted into v'', and the two subpaths are identified.

Figure 9. An illustration of graph and path modification in Case 3.

This face suppression naturally defines a way of modifying the paths in $\hat{\mathcal{P}}$, as follows. Denote by $C_{F'}$ the boundary cycle of face F'. For every path $P \in \hat{\mathcal{P}}$:

- if $P \cap C_{F'} = \emptyset$, then we do not modify it;
- if $P \cap V(C_{F'}) \subseteq \{v, v'\}$, then we let it contain the new vertex v'' at the same location;
- if $P \cap C_{F'}$ is a subpath of $\tilde{P}$ or a subpath of $\tilde{P}'$, then we replace that subpath of P with the corresponding subpath of $\tilde{P}''$.

It is easy to verify that the resulting set $\hat{\mathcal{P}}$ is non-crossing and well-structured, and it still generates the resulting graph $\hat{W}$. Also, the number of faces in $\hat{W}$ decreases by 1 in this case. We now show that the value $\mathsf{bs}(\hat{W})$ does not decrease. Note that the degree of every vertex except for v, v' does not change, and the degree of the new vertex v'' obtained from contracting (v, v') has degree at least 3 in the resulting graph, so all big faces remain unchanged, and so are their contribution to $\mathsf{bs}(\hat{W})$.

We denote by $\tilde{W}$ the graph $\hat{W}$ when none of the Cases 1-3 described above happens. We are then guaranteed that, for each small face F in $\tilde{W}$, either

- it does not share a vertex with any big faces; or
- it contains exactly 3 vertices, it shares a vertex with exactly one big face, and both endpoints of the edge that it shares with that big face has degree exactly 3; or
- it shares a vertex with at least two big faces (in this case we call it a *bridge* face).

We call vertices that are shared by a bridge face and a big face *bridge vertices*, and we call vertices that belong to at least two big faces *interface vertices*. Clearly, bridge vertices and interface vertices must be branch vertices. Consider now any big face F, and let V'_F be the set of its bridge vertices and interface vertices. We prove the following observation.

Observation A.5. *Let F be a big face and let u, u' be a pair of vertices in V'_F that appear consecutively on C_F. That is, there is a subpath Q of C_F connecting u to u' that does not contain any other vertex of V'_F. Then the number of branch vertices that is an internal vertex of Q is at most 2.*

Proof: Consider any edge e in path Q that is not incident to u or u'. Let F' be the other face that e is incident to, so F' is a small face. Since both endpoints of e are not in V'_F, face F' do not share vertex with any other big faces. From the above discussion, face F' has to contain exactly three vertices, and the degrees of both endpoints of e are exactly 3. Let z_e be the other vertex of face F'. Note that, via similar arguments we can show that all internal vertices of Q have degree exactly 3. Therefore, the vertex $z_{e'}$ defined for every other edge e' of Q that is not incident to u or u' has to coincide with z_e. But if the number of branch vertices that is an internal vertex of Q is greater than 2, then there exists a vertex $u'' \in V(Q)$ that is not adjacent to either u or u''. Now the existence of edge (z_e, u'') can be shown to cause a contradiction to the well-structuredness of $\hat{\mathcal{P}}$, using similar arguments in the proof of Theorem A.4. $\square$

Similarly, we can prove the following observation.

Observation A.6. *Let F, F' be a pair of big faces, and let $\hat{F}, \hat{F}'$ be a pair of bridge faces, such that both $\hat{F}, \hat{F}'$ share vertices with both F, F'. Then if we denote by R the region outside $F, F', \hat{F}, \hat{F}'$ surrounded by the boundaries of $F, F', \hat{F}, \hat{F}'$ that does not contain the outerface, then the boundary of R contains at most 8 bridge vertices.*

Consider now the dual graph $\tilde{W}^*$ of the resulting graph $\tilde{W}$. From similar arguments in the proof of Theorem A.3, we know that in order to show $\sum_{R\in\mathcal{R}} |U'_R| \le |U| \cdot \big(1 + O(1/\lambda')\big)$, it suffices to show that $\sum_{v\in V(\tilde{W}^*), v\ne v_\infty} \deg_{\tilde{W}^*\setminus v_\infty}(v) \le O(|U|/\lambda')$. We denote by $\check{W}$ the subgraph of $\tilde{W}^*$ induced by all nodes corresponding to big faces and bridge faces. From the above two observations, we know that, it suffices to show that $\sum_{v\in V(\check{W})} \deg_{\check{W}}(v) \le O(|U|/\lambda')$.

Let $\hat{F}$ be a bridge face. We denote by $F_1, \ldots, F_t$ the big faces that share a vertex with $\hat{F}$, where the faces are indexed according to the circular ordering in which they intersect with $\hat{F}$. Then, it is easy to see that, if we replace, for each bridge face $\hat{F}$, all edges incident to node $v_{\hat{F}}$ (the node in $\check{W}$ that corresponds to face $\hat{F}$) with edges $(v_{F_1}, v_{F_2}), \ldots, (v_{F_t}, v_{F_1})$, then the resulting graph $\check{W}$ is still a planar graph, with each every having at most one parallel copy. Using similar arguments in the proof of Theorem A.3, we can show that $\sum_{v\in V(\check{W})} \deg_{\check{W}}(v) \le O(|U|/\lambda')$. This completes the proof of Claim 4.1.

A.5 Proof of Claim 4.2

Let u, u' be terminals in U. We will show that $e^{-\varepsilon} \cdot \mathsf{dist}_Z(u,u') \le \mathsf{dist}_{\hat{H}}(u,u') \le e^{\varepsilon} \cdot \mathsf{dist}_Z(u,u')$.

On the one hand, let Q be the u-u' shortest path in $\hat{H}$. We view path Q as being directed from u to u'. Let $\{u_1, \dots, u_k\}$ be the set of all inner vertices of Q that belongs to $V^* \cup Y$ (recall that V^* is the set of branch vertices), where the vertices are indexed according to the order in which they appear on Q. Therefore, if we set $u_0 = u$ and $u_{k+1} = u'$, then for each $0 \le i \le k$, either one of u_i, u_{i+1} is a branch vertex and so $\mathsf{dist}_Z(u_i,u_{i+1}) = \mathsf{dist}_{\hat{H}}(u_i,u_{i+1})$, or u_i, u_{i+1} are both vertices of Y and belong to the same instance in $\mathcal{H}$ and so $\mathsf{dist}_{\hat{H}}(u_i,u_{i+1}) \ge e^{-\varepsilon} \cdot \mathsf{dist}_Z(u_i,u_{i+1})$. Thus, if we set, for each $0 \le i \le k$, H_{R_i} to be the graph in $\mathcal{H}$ that vertices u_i, u_{i+1} belong to, then

$$\begin{aligned}\mathsf{dist}_{\hat{H}}(u,u') = \sum_{0\le i\le k} \mathsf{dist}_{\hat{H}}(u_i,u_{i+1}) &\ge \sum_{0\le i\le k} \mathsf{dist}_{H_{R_i}}(u_i,u_{i+1}) \\ &\ge \sum_{0\le i\le k} e^{-\varepsilon}\cdot \mathsf{dist}_{Z_{R_i}}(u_i,u_{i+1}) \ge \sum_{0\le i\le k} e^{-\varepsilon}\cdot \mathsf{dist}_{Z}(u_i,u_{i+1}) \ge e^{-\varepsilon}\cdot\mathsf{dist}_Z(u,u').\end{aligned}$$

On the other hand, let Q' be the u-u' shortest path in Z. We view path Q' as being directed from u to u'. Let $\{u'_1, \dots, u'_k\}$ be the set of all inner vertices of Q' that belongs to $V^* \cup Y$ (recall that V^* is the set of branch vertices), where the vertices are indexed according to the order in which they appear on Q'. Therefore, if we set $u'_0 = u$ and $u'_{k+1} = u'$, then for each $0 \le i \le k$, either one of u'_i, u'_{i+1} is a branch vertex and so $\mathsf{dist}_Z(u'_i,u'_{i+1}) = \mathsf{dist}_{\hat{H}}(u'_i,u'_{i+1})$, or u'_i, u'_{i+1} are both vertices of Y and belong to the same instance in $\mathcal{H}$ and so $\mathsf{dist}_Z(u'_i,u'_{i+1}) \ge e^{-\varepsilon} \cdot \mathsf{dist}_{\hat{H}}(u'_i,u'_{i+1})$. Thus, if we set, for each $0 \le i \le k$, H_{R_i} to be the graph in $\mathcal{H}$ that vertices u'_i, u'_{i+1} belong to, then

$$\begin{aligned}\mathsf{dist}_{Z}(u,u') = \sum_{0\le i\le k} \mathsf{dist}_{Z}(u'_i,u'_{i+1}) &\ge \sum_{0\le i\le k} \mathsf{dist}_{Z_{R_i}}(u'_i,u'_{i+1}) \\ &\ge \sum_{0\le i\le k} e^{-\varepsilon}\cdot \mathsf{dist}_{H_{R_i}}(u'_i,u'_{i+1}) \ge \sum_{0\le i\le k} e^{-\varepsilon}\cdot \mathsf{dist}_{\hat{H}}(u'_i,u'_{i+1}) \ge e^{-\varepsilon}\cdot\mathsf{dist}_{\hat{H}}(u,u').\end{aligned}$$

A.6 Proof of Claim 4.9

We denote by ℓ the level that set S belongs to. We use the following simple observations.

Observation A.7. *For every pair (u,u') with $u \in S$ and $u' \in S'$, $\mathsf{dist}_H(u,u') \ge \mu^{\ell+1}$. For every pair (u,u') of terminals in S' that do not belong to the same graph in $\mathcal{H}$, $\mathsf{dist}_H(u,u') \ge \mu^{\ell+1}$.*

Proof: From the construction of the collection $\mathcal{S}$ and the definition of sets S, S', S^*, if $u \in S$ and $u' \in S'$, then u, u' do not belong to the same $(\ell+1)$-level set, and so $\mathsf{dist}_H(u,u') > \mu^{\ell+1}$. Consider now a pair u, u' of terminals in S' that do not belong to the same graph in $\mathcal{H}$. From the construction of the graphs in $\mathcal{H}$, there must exist a pair $\hat{u}, \hat{u}'$ of terminals in S, such that the pairs $(\hat{u}, \hat{u}')$ and (u,u') are crossing. Therefore, from Monge property,

$$\mathsf{dist}_H(u,u') \ge \mathsf{dist}(u,\hat{u}) + \mathsf{dist}(u',\hat{u}') - \mathsf{dist}(\hat{u},\hat{u}') \ge \mu^{\ell+1} + \mu^{\ell+1} - 2r\mu^{\ell} > \mu^{\ell+1},$$

where we have used the fact (from Observation 4.6) that $\mathsf{dist}(\hat{u},\hat{u}') \le 2r\mu^{\ell}$. □

Let u, u' be terminals in U. We will show that $\mathsf{dist}_{\hat{H}}(u,u') \le \mathsf{dist}_H(u,u') \le e^{\varepsilon_r} \cdot \mathsf{dist}_{\hat{H}}(u,u')$. If vertices u, u' belong to the same instance in $\mathcal{H}$, then since the instances in $\mathcal{H}$ is obtained by cutting along shortest paths in H, it is easy to see that $\mathsf{dist}_H(u,u') = \mathsf{dist}_{\hat{H}}(u,u')$. Therefore, we assume from now on that that terminals u, u' do not belong to the same instance in $\mathcal{H}$. We denote by Y the set of all vertices that belongs to more than one instances in $\mathcal{H}$.

Recall that, in the procedure SPLIT, we have sliced H open along a set of shortest paths in H. Let $\mathcal{R}$ be the collection of regions (of H) that we get. Recall that each instance in $\mathcal{H}$ corresponds to a region in $\mathcal{R}$. We say that an instance $(H_R, U_R) \in \mathcal{H}$ is a *regular* instance if the corresponding region R is surrounded by (i) a contiguous segment of the outer-boundary of H and (ii) the image of a single path in $\mathcal{P}$. Since the paths are well-structured, when we consider a u-u' shortest path Q in H, we can assume that, for each regular instance $(H_R, U_R) \in \mathcal{H}$ with $u, u' \notin V(H_R)$, the intersection between Q and H_R is a subpath of the path in $\mathcal{P}$ that surrounds the region R and both endpoints of this subpaths are branch vertices.

Consider now the u-u' shortest path Q in H. Assume that $u \in H_R$ and $u' \in H_{R'}$. We view path Q as being directed from u to u'. Let v be the last vertex of Q that belongs to H_R, and let v' be the first vertex of Q after v that belongs to $H_{R'}$. We distinguish between the following cases.

Case 1. $v \neq v'$**.** From the construction of graph $\hat{H}$ and the above discussion, it is easy to verify that the entire path Q is also contained in graph $\hat{H}$, so $\mathsf{dist}_{\hat{H}}(u,u') \le \mathsf{dist}_H(u,u')$. On the other hand, it is easy to verify that any shortest path in $\hat{H}$ connecting u to u' is also entirely contained in H, so $\mathsf{dist}_{\hat{H}}(u,u') \ge \mathsf{dist}_H(u,u')$. Therefore, $\mathsf{dist}_{\hat{H}}(u,u') = \mathsf{dist}_H(u,u')$.

Case 2. $v = v'$**.** This means that path Q only touches two regions, R and R'. If one of u, u' belongs to set S', then from Observation A.7 and the fact (from Observation 4.6) that the boundary path of R and R' have total length at most $2r\mu^{\ell}$, it is easy to verify that

$$\mathsf{dist}_H(u,u') \le \mathsf{dist}_{\hat{H}}(u,u') \le (1+O(1/r)) \cdot \mathsf{dist}_H(u,u') \le e^{\varepsilon_r} \cdot \mathsf{dist}_H(u,u').$$

If both u, u' belong to S, then from the construction of $\hat{H}$, $\mathsf{dist}_H(u,u') \le \mathsf{dist}_{\hat{H}}(u,u')$. It remains to consider the case where at least one of u, u' belongs to set S^*. Assume without loss of generality that $u \in S^*$. Since the set $Y \cap U_R$ contains an ε_r-cover of u on the boundary path of R, there exists a vertex $\hat{v} \in Y \cap U_R$, such that $\mathsf{dist}_H(u,\hat{v}) + \mathsf{dist}_H(\hat{v}, v) \le e^{\varepsilon_r} \cdot \mathsf{dist}_H(u, v)$. In this case we denote by v_1 the copy of v in H_R and by v_2 the copy of v in $H_{R'}$, then

$$\begin{aligned}
\mathsf{dist}_H(u,u') \le \mathsf{dist}_{\hat{H}}(u,u') &\le \mathsf{dist}_{\hat{H}}(u,\hat{v}) + \mathsf{dist}_{\hat{H}}(\hat{v}, v_2) + \mathsf{dist}_{\hat{H}}(v_2, u') \\
&\le \mathsf{dist}_H(u,\hat{v}) + \mathsf{dist}_H(\hat{v}, v_2) + \mathsf{dist}_H(v', u') \\
&\le e^{\varepsilon_r} \cdot \mathsf{dist}_H(u,v) + \mathsf{dist}_H(v,u') \le e^{\varepsilon_r} \cdot \mathsf{dist}_H(u,u').
\end{aligned}$$

B Missing Proofs in Section 5

B.1 Complete Description of Procedures SPLIT$_h$ and GLUE$_h$

Splitting. The input to procedure SPLIT$_h$ consists of

- an h-hole instance (H, U);
- a path P connecting a pair of its terminals that lie on different holes; and
- a set Y of vertices in P that contains both endpoints of P.

The output of procedure SPLIT$_h$ is an $(h-1)$-hole instance $(\tilde{H}, \tilde{U})$ that is constructed as follows. Let u, u' be the endpoints of P. We denote by γ the curve representing the image of path P in H, and view it as being directed from u to u'. For each $v \in V(P)$, we define $\delta_1(v)$ ($\delta_2(v)$, resp.) as the set of all incident edges of v in graph H, whose image lie on the left (right, resp.) side of γ, as we traverse along γ from u to u'. We now modify the graph H as follows. Replace each vertex $v \in V(P)$ by two new vertices v_1 and v_2, where v_1 is incident to all edges in $\delta_1(v)$ and v_2 is incident to all edges in $\delta_2(v)$. Then we add, for each edge (v, v') of path P, an edge (v_1, v_1') and an edge (v_2, v_2'). The resulting graph is denoted by $\tilde{H}$.

We naturally construct a planar drawing of graph $\tilde{H}$, as follows. We start from the drawing ϕ associated with instance (H, U). We first erase from it the images of all vertices and edges of P. Denote by α (α', resp.) the hole in ϕ whose boundary contains the image of u (u', resp.). Let S be a thin strip around the curve γ. We draw the new vertices u_1, u_2 at the intersections of S and the boundary of hole α, where u_1 lies on the left of γ and u_2 lies on the right of γ. Similarly, we draw the new vertices u'_1, u'_2 at the intersections of S and the boundary of hole α', where u'_1 lies on the left of γ and u'_2 lies on the right of γ. Now for every other vertex $v \in V(P)$, we draw the new vertex v_1 (v_2, resp.) on the boundary of S just to the left (right, resp.) of the old image of v in ϕ. The images of other vertices remain the same as in ϕ. For each vertex $v \in V(P)$ and each edge $e \in \delta_1(v)$ ($\delta_2(v)$, resp.), we slightly modify the image of e to make it direct to v_1 (v_2, resp.). Lastly, for each edge $(v, v') \in P$, we draw the image of new edge (v_1, v'_1) ((v_2, v'_2), resp.) as the segment of the boundary of strip S between the points representing the images of v_1, v'_1 ((v_2, v'_2), resp.). This completes the construction of a planar drawing of $\tilde{H}$, that we denote by $\tilde{\phi}$. See Figure 6(a) and Figure 6(b) for an illustration.

We now define $\tilde{U}$ to be the set obtained from U by replacing for each vertex $y \in Y$, two new vertices y_1 and y_2 (since such a vertex y belongs to path P), so $|\tilde{U}| = |U| + 2|Y|$. The instance $(\tilde{H}, \tilde{U})$ is the output of procedure $\textsc{Split}_h$. We now show that it is indeed an $(h-1)$-hole instance.

We define area $\beta = \alpha \cup S \cup \alpha'$. It is easy to observe that no vertices or edges are drawn inside the interior of area β, and if we denote by $U(\alpha)$ the set of terminals in H that lie on the boundary of α, and define set $U(\alpha')$ similarly, then in $\tilde{H}$, the boundary of β contains the images of terminals in $(U(\alpha) \setminus \{u\}) \cup (U(\alpha') \setminus \{u'\}) \cup \{y_1, y_2 \mid y \in Y\}$. Therefore $(\tilde{H}, \tilde{U})$ is a valid $(h-1)$-hole instance.

Gluing. We next describe the procedure $\textsc{Glue}_h$, which is intuitively a reverse process of procedure called $\textsc{Split}_h$. Assume that we have applied the procedure $\textsc{Split}_h$ to some h-hole instance (H, U), some path P connecting a pair u, u' of terminals in U that lie on holes α, α' respectively, and a subset Y of vertices in P. Let $(\tilde{H}, \tilde{U})$ be the $(h-1)$-hole instance that the procedure $\textsc{Split}_h$ outputs, where holes α, α' are merged into hole β. We then denote, for each $y \in Y$, by y^1 and y^2 the two terminals in $\tilde{U}$ obtained by splitting y. The procedure $\textsc{Glue}_h$ takes as input an emulator $(\tilde{H}', \tilde{U})$ for instance $(\tilde{H}, \tilde{U})$, and works as follows.

We let graph H' be obtained from graph $\tilde{H}'$ by identifying, for each $y \in Y$, vertex y^1 with vertex y^2 (and name the obtained vertex y). Denote $\tilde{Y} = \{y^1, y^2 \mid y \in Y\}$. We then set $U' = (\tilde{U} \setminus \tilde{Y}) \cup \{u, u'\}$. Clearly, $U' = U$. The output of algorithm $\textsc{Glue}_h$ is instance (H', U).

We associate with instance (H', U) a planar drawing with terminals of U drawn on the boundary of h holes as follows. We denote by γ the boundary segment of hole β from u^2 to u^1 that does not contain any other vertex of $\tilde{Y}$, and denote by γ' the boundary segment of hole β from $(u')^1$ to $(u')^2$ that does not contain any other terminal of $\tilde{Y}$. We now compute, for each $y \in Y$, a curve γ_y connecting y^1 to y^2, such that the curves $\{\gamma_y \mid y \in Y\}$ all lie in hole β and are mutually disjoint. We now move, for each $y \in Y$, the images of y^1 and y^2 along the curve γ_y towards each other until they are identified. Now γ becomes a closed curve that surrounds a region which does not contain the image of any vertices or edges in its interior. We designate this region by hole α. We define hole α' for the closed curve γ' similarly. It is easy to verify that all terminals of U' that previously lied on the boundary of hole β now lie on the boundary of either hole α or hole α'. See Figure 6(c) for an illustration. Therefore, (H', U) is a valid h-hole instance, and it is easy to verify that instance (H', U) is aligned with instance (H, U).

B.2 Proof of Claim 5.2

For convenience, we rename the selected terminals u, u' by $\hat{u}, \hat{u}'$, respectively. Throughout the proof, we will use u, u' to denote some pair of terminals in U, and we will show that $e^{-\varepsilon'} \cdot \mathsf{dist}_Z(u, u') \le \mathsf{dist}_{\hat{H}}(u, u') \le e^{\varepsilon'} \cdot \mathsf{dist}_Z(u, u')$.

On the one hand, let Q be the shortest path in $\hat{H}$ connecting u to u'. We view the path Q as being directed from u to u'. Recall that in graph $\hat{H}$, for each vertex $y \in Y$, we have denoted by $\delta_1(y)$ the incident edges of y that lie on one side of path P, and denote by $\delta_2(y)$ the incident edges of y that lie on the other side of path P. We denote $E_1 = \bigcup_{y\in Y} \delta_1(y)$ and $E_2 = \bigcup_{y\in Y} \delta_2(y)$. If either $E(Q)\cap E_1 = \varnothing$ or $E(Q)\cap E_2 = \varnothing$ holds, then it is immediate to verify that path Q is entirely contained in graph $\tilde{H}$. Since $(\tilde{Z},\tilde{U})$ is an ε-emulator for instance $(\tilde{H},\tilde{U})$, we get that

$$\mathsf{dist}_{\hat{H}}(u,u') = \mathsf{dist}_{\tilde{H}}(u,u') \ge e^{-\varepsilon}\cdot \mathsf{dist}_{\tilde{Z}}(u,u') \ge e^{-\varepsilon}\cdot \mathsf{dist}_{Z}(u,u').$$

Assume now that $E(Q')\cap E_1 \neq \varnothing$ or $E(Q')\cap E_2 \neq \varnothing$. Recall that graph $\hat{H}$ contains two copies P_1, P_2 of path P that corresponds to the sides of E_1, E_2, respectively. We can assume without loss of generality that path Q is the concatenation of (i) a path Q_1 connecting u to some vertex $x_1 \in V(P_1)$, that is internally disjoint from P_1; (ii) a subpath P_1' of P_1 connecting x_1 to some vertex $y \in Y$; (iii) a subpath P_2' of P_2 connecting y to some vertex $x_2' \in V(P_2)$; and (iv) a path Q_2' connecting x_2' to some vertex u', that is internally disjoint from P_2. Recall that $(\tilde{Z},\tilde{U})$ is an ε-emulator for instance $(\tilde{H},\tilde{U})$, and instance (Z,U) is obtained by applying the procedure $\textsc{Glue}_h$ to instance $(\tilde{Z},\tilde{U})$. We denote by y_1, y_2 the copies of y in graph $\tilde{H}$, where $y_1 \in V(P_1)$ and $y_2 \in V(P_2)$. Then

$$\begin{aligned}
\mathsf{dist}_{\hat{H}}(u,u') &= \mathsf{dist}_{\hat{H}}(u,x_1) + \mathsf{dist}_{P_1}(x_1,y) + \mathsf{dist}_{P_2}(y,x_2') + \mathsf{dist}_{\hat{H}}(x_2',u)\\
&\ge \mathsf{dist}_{\tilde{H}}(u,x_1) + \mathsf{dist}_{\tilde{H}}(x_1,y_1) + \mathsf{dist}_{\tilde{H}}(y_2,x_2') + \mathsf{dist}_{\tilde{Z}}(x_2',u)\\
&\ge e^{-\varepsilon}\cdot\left(\mathsf{dist}_{\tilde{Z}}(u,x_1) + \mathsf{dist}_{\tilde{Z}}(x_1,y_1) + \mathsf{dist}_{\tilde{Z}}(y_2,x_2') + \mathsf{dist}_{\tilde{Z}}(x_2',u)\right)\\
&\ge e^{-\varepsilon}\cdot\left(\mathsf{dist}_{Z}(u,x_1) + \mathsf{dist}_{Z}(x_1,y) + \mathsf{dist}_{Z}(y,x_2') + \mathsf{dist}_{Z}(x_2',u)\right)\\
&\ge e^{-\varepsilon}\cdot \mathsf{dist}_{Z}(u,u').
\end{aligned}$$

On the other hand, let Q' be the shortest path in Z connecting u to u'. We view the path Q' as being directed from u to u'. Via similar analysis, we can easily show that, if Q' does not contain vertices of Y, then

$$\mathsf{dist}_{Z}(u,u') = \mathsf{dist}_{\tilde{Z}}(u,u') \ge e^{-\varepsilon}\cdot \mathsf{dist}_{\tilde{H}}(u,u') \ge e^{-\varepsilon}\cdot \mathsf{dist}_{\hat{H}}(u,u').$$

We assume from on now that Q' contains some vertices of Y. In graph $\tilde{Z}$, we denote by $\tilde{E}_1$ the set of edges incident to some vertex of $Y_1 = \{y_1 \mid y \in Y\}$, and define set $\tilde{E}_2$ for set $Y_2 = \{y_2 \mid y\in Y\}$ similarly. Let $y^1 \dots, y^r$ be the vertices of $Y \cap V(Q)$, where the vertices are indexed according to their appearance on Q. For each $0 \le j \le r$, we denote by Q_j the subpath of Q between vertices y^j and y^{j+1} (where we set $y^0 = u$ and $y^{r+1} = u'$). For each $0\le j\le r$, we set $a(j)$ to be 1 (2, resp.) if the first edge of Q_j belongs to $\tilde{E}_1$ ($\tilde{E}_2$, resp.), and set set $b(j)$ to be 1 (2, resp.) if the last edge of Q_j belongs to $\tilde{E}_1$ ($\tilde{E}_2$, resp.). Since $(\tilde{Z},\tilde{U})$ is an ε-emulator for instance $(\tilde{H},\tilde{U})$, and instance (Z,U) is obtained by applying the procedure $\textsc{Glue}_h$ to instance $(\tilde{Z},\tilde{U})$, we get that

$$\begin{aligned}
\mathsf{dist}_{Z}(u,u') = \sum_{0\le j\le r} \mathsf{dist}_{Z}(y^j,y^{j+1}) &= \sum_{0\le j\le r} \mathsf{dist}_{\tilde{Z}}(y^j_{a(j)}, y^{j+1}_{b(j)})\\
&\ge \sum_{0\le j\le r} e^{-\varepsilon}\cdot \mathsf{dist}_{\tilde{H}}(y^j_{a(j)}, y^{j+1}_{b(j)}) \ge \sum_{0\le j\le r} e^{-\varepsilon}\cdot \mathsf{dist}_{\hat{H}}(y^j,y^{j+1}) \ge e^{-\varepsilon}\cdot \mathsf{dist}_{\hat{H}}(u,u').
\end{aligned}$$

Altogether, we get that $e^{-\varepsilon'}\cdot \mathsf{dist}_{Z}(u,u') \le \mathsf{dist}_{\hat{H}}(u,u') \le e^{\varepsilon'}\cdot \mathsf{dist}_{Z}(u,u')$.

B.3 Proof of Claim 5.3

For convenience, we rename the selected terminals u,u' by $\hat{u},\hat{u}'$, respectively. Throughout the proof, we will use u,u' to denote some pair of terminals in U, and we will show that $e^{-\varepsilon'}\cdot \mathsf{dist}_{\hat{H}}(u,u') \le \mathsf{dist}_{H}(u,u') \le \mathsf{dist}_{\hat{H}}(u,u')$.

On the one hand, let Q be a shortest path in H connecting $\hat{u}$ to $\hat{u}'$. We view Q as being directed from u to u'. If $V(Q) \cap V(P) = \emptyset$, then it is immediate to verify that path Q is entirely contained in graph $\hat{H}$, so $\mathsf{dist}_{\hat{H}}(u,u') \le \mathsf{dist}_H(u,u')$. Assume now that $V(Q) \cap V(P) \neq \emptyset$. Since Q and P are shortest paths in H, $Q \cap P$ is a subpath of both Q and P. Let v, v' be the endpoints of this path where v is closer to u and v' is closer to u' on Q (note that it is possible that $v = v'$). Since set Y contains an ε'-cover of u on P, there exists some vertex $y \in Y$, such that $\mathsf{dist}_H(u,y) + \mathsf{dist}_H(y,v) \le e^{\varepsilon} \cdot \mathsf{dist}_H(u,v)$; and similarly since set Y contains an ε'-cover of u' on Q, there exists some vertex $y' \in Y$, such that $\mathsf{dist}_H(u',y') + \mathsf{dist}_H(y',v') \le e^{\varepsilon} \cdot \mathsf{dist}_H(u',v')$. From the construction of graph $\hat{H}$, we get that

$$\begin{aligned}
\mathsf{dist}_H(u,u') &= \mathsf{dist}_H(u,v) + \mathsf{dist}_H(v,v') + \mathsf{dist}_H(u',v') \\
&\ge e^{-\varepsilon'} \cdot (\mathsf{dist}_H(u,y) + \mathsf{dist}_H(y,v)) + \mathsf{dist}_H(v,v') + e^{-\varepsilon} \cdot (\mathsf{dist}_H(u',y') + \mathsf{dist}_H(y',v')) \\
&\ge e^{-\varepsilon'} \cdot (\mathsf{dist}_{\hat{H}}(u,y) + \mathsf{dist}_{\hat{H}}(y,v) + \mathsf{dist}_{\hat{H}}(v,v') + \mathsf{dist}_{\hat{H}}(u',y') + \mathsf{dist}_{\hat{H}}(y',v')) \\
&\ge e^{-\varepsilon} \cdot \mathsf{dist}_{\hat{H}}(u,u').
\end{aligned}$$

On the other hand, let Q' be a shortest path in $\hat{H}$ connecting $\hat{u}$ to $\hat{u}'$. We view Q' as being directed from u to u'. Recall that in graph H, for each vertex $y \in Y$, we have denoted by $\delta_1(y)$ the incident edges of y that lie on one side of path P, and denote by $\delta_2(y)$ the incident edges of y that lie on the other side of path P. We denote $E_1 = \bigcup_{y \in Y} \delta_1(y)$ and $E_2 = \bigcup_{y \in Y} \delta_2(y)$. If either $E(Q') \cap E_1 = \emptyset$ or $E(Q') \cap E_2 = \emptyset$ holds, then it is immediate to verify that path Q' is entirely contained in graph H, so $\mathsf{dist}_H(u,u') \le \mathsf{dist}_{\hat{H}}(u,u')$. Assume now that $E(Q') \cap E_1 \neq \emptyset$ or $E(Q') \cap E_2 \neq \emptyset$. Recall that graph $\hat{H}$ contains two copies P_1, P_2 of path P that corresponds to the sides of E_1, E_2, respectively. We can assume without loss of generality that path Q' is the concatenation of (i) a path Q'_1 connecting u to some vertex $x_1 \in V(P_1)$, that is internally disjoint from P_1; (ii) a subpath P'_1 of P_1 connecting x_1 to some vertex $y \in Y$; (iii) a subpath P'_2 of P_2 connecting y to some vertex $x'_2 \in V(P_2)$; and (iv) a path Q'_2 connecting x'_2 to some vertex u', that is internally disjoint from P_2. Let x be the original copy of x_1 in graph H, and let x' be the original copy of x_1 in graph H. From the construction of graph $\hat{H}$, we get that

$$\begin{aligned}
\mathsf{dist}_{\hat{H}}(u,u') &= \mathsf{dist}_{\hat{H}}(u,x_1) + \mathsf{dist}_{P_1}(x_1,y) + \mathsf{dist}_{P_2}(y,x'_2) + \mathsf{dist}_{\hat{H}}(u',x'_2) \\
&\ge \mathsf{dist}_H(u,x) + \mathsf{dist}_H(x,x') + \mathsf{dist}_H(u',x') \ge \mathsf{dist}_H(u,u').
\end{aligned}$$

B.4 Proof of Theorem 5.5

Similar to Frederickson [Fre87] and Klein-Mozes-Sommer [KMS13], we recursively find balanced cycle separators to subdivide the input graph. To control the number vertices, boundary vertices, holes, and terminals within each piece simultaneously, we ask the cycle separator to balance these quantities in rounds. Specifically, at recursive level ℓ:

- If $\ell \bmod 4 = 0$, balance the vertices.
- If $\ell \bmod 4 = 1$, balance the boundary vertices.
- If $\ell \bmod 4 = 2$, balance the holes by inserting one *supernode* per hole.
- If $\ell \bmod 4 = 3$, balance the terminals.

We terminate the recursion four rounds after a piece has size at most r. The depth of the recursion tree is $\log(n/r)$, and a similar analysis as in Klein-Mozes-Sommer [KMS13] shows that the number of terminals within each piece is $O(kr/n)$.