Almost-Linear ε-Emulators for Planar Graphs

Hsien-Chih Chang* Robert Krauthgamer† Zihan Tan‡

November 11, 2021

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 design the first ε-emulators for planar graphs of almost-linear size $k^{1+o(1)}/\varepsilon^{O(1)}$. In terms of k, this is a dramatic improvement over the previous quadratic upper bound of Cheung, Goranci and Henzinger [ICALP 2016], and breaks below known quadratic lower bounds for exact emulators ($\varepsilon = 0$).

Moreover, our emulators can be computed in near-linear time whenever $k \le \sqrt{n}$, with applications to fast $(1+\varepsilon)$-approximation algorithms for basic optimization problems on planar graphs such as minimum (s,t)-cut and diameter.

*Department of Computer Science, Dartmouth College. Email: hsien-chih.chang@dartmouth.edu.

†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.

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 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', such as 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 same set of terminals, i.e., $T \subseteq V(G')$, satisfying

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

where dist_G denotes the shortest-path distance in G (and similarly for G'). In the important case that $\alpha = 1 + \varepsilon \approx e^{\varepsilon}$ for $0 \leq \varepsilon \leq 1$, we simply say that G' is an *ε-emulator*.[1]

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 $G \in \mathcal{F}$ for a fixed graph family $\mathcal{F}$ (e.g., all planar graphs) and requiring that also $G' \in \mathcal{F}$. If $\mathcal{F}$ is a minor-closed family, one can further impose the requirement that G' is a minor of G, which is only stronger, as it clearly implies $G' \in \mathcal{F}$. In this case, one can omit the requirement about $\mathcal{F}$, which is equivalent to $\mathcal{F}$ being the family of all graphs. An arbitrary graph G' satisfying (1) is called an *emulator*; notice that G' need not be a subgraph or a minor of G (these two settings are known as a *spanner* and a *distance-approximating minor*).

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 range 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' [KNZ14, CGH16, CGMW18, CO20, GHP20]. For planar graphs, the known size bounds are $O(k^4)$ and $\Omega(k^2)$.[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 [Gup01, BG08, CXKR06, KKN15, Che18, Fil18, FKT19]. For planar graphs, the known distortion bounds are $O(\log k)$ and 2.[3]

[1] Our formal definition in Section 2 differs slightly (allowing two-sided errors), which affects 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.

[3] That is, for fixed emulator size $|V(G')| = k$, every planar graph G (in fact, every graph) admits a minor with distortion $O(\log k)$ [Fil18], but for some planar graphs (specifically stars) the distortion must be at least 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.

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 $\tilde{O}(t) = t \cdot \operatorname{polylog} t$, 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 almost-linear size; furthermore, these emulators can be computed in near-linear time whenever $k \le \sqrt{n}$. 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 $|V(G')| = k^{1+o(1)}/\varepsilon^{O(1)}$. Furthermore, such an emulator can be computed deterministically, for input G and ε, in time $O\big((n+k^2)\log^{O(1)} n/\varepsilon^{O(1)}\big)$.*

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 Table 1 for detailed comparison to prior work.

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

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. Both algorithms have the same query time $O(\log n)$. The preprocessing time of our data structure is $O_\varepsilon(n \operatorname{poly}(\log^* n))$, which for fixed $\varepsilon > 0$ is faster than Klein's $O(n \log n)$-time algorithm [Kle05] for the exact setting when $\varepsilon = 0$.

Theorem 1.2 *Given a parameter $0 < \varepsilon < 1$, an n-vertex plane graph G and a set of terminals T all lying on the boundary of G, one can preprocess an ε-MSSP data structure on G with respect to T in time $O(n \operatorname{poly}(\log^* n)/\varepsilon^{O(1)})$, and then answer queries in time $O(\log n)$.*

Our second application is an $O_\varepsilon(n \log^* n)$-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 $0 < \varepsilon < 1$, one can compute a $(1+\varepsilon)$-approximation to the minimum (s,t)-cut in G in time $O(n \log^* n/\varepsilon^{O(1)})$.*

Our third application is an $O_\varepsilon(n \log n \log^* n)$-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 [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 $O(n \log n \log^* n + n \log n/\varepsilon^{O(1)})$.*

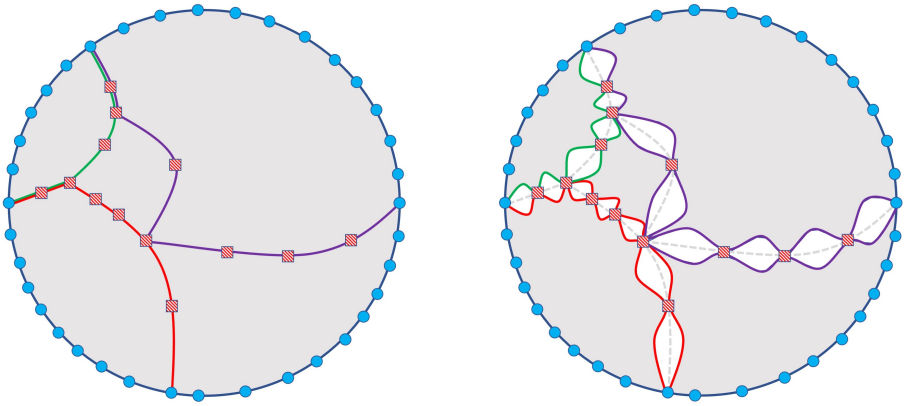
1.3 Technical Contributions

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 is planar and the terminals all lie on the outerface, where *spread* is defined to be the ratio between the largest and the smallest distances between terminals. Spread reduction is a crucial preprocessing step for optimization problems, particularly in Euclidean spaces or on planar graphs [SA12, BG13, KKN15, CFS19, FL20], 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 5.2), 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 (cf. Section 5.1). 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 be (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 the quarter-grid construction [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) (b)

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

Let r be the number of terminals in the current piece. 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 will 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 additive term $\log\Phi/L_r$, where Φ is the *spread* of the terminal distances (cf. Claim 4.5). When $\Phi \le 2^{r^{0.9}}$, 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 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 $|U|/5$ and $4|U|/5$ where U is the set of terminals in the current instance) (cf. Section 4.4.1), we construct *non-crossing* a collection of 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 non-expanding, the number of terminals in $\hat{S} \setminus S$ is at most $(e^{r^{-0.7}} - 1)|S| \leq r^{0.3}$, and thus the number of portals is $O(r^{0.3}/\varepsilon_r) \leq O(r^{0.4})$, 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 for pairs of one terminal from S and one 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 all non-expanding clusters are not moderate in 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 (which is a simple corollary of the Jordan curve theorem) 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 4.8). 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 can 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 that each has $O(\sqrt{r})$ boundary vertices and $O(1)$ holes, called an *r-division* (Frederickson [Fre87], Klein-Mozes-Sommer [KMS13]);
2. Compute all-pairs shortest paths between all boundary vertices within a piece, using the *multiple-source shortest paths* algorithm (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. In a sense, 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 6.1). This allows us to construct ε-emulator that is sub-linear in size of each piece in linear time (up to $O_\varepsilon(\log^* n)$ factors).

1.4 Related Work

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

Another type of vertex sparsifiers that are extensively studied are cut/flow sparsifiers. There are rich lines of works for constructing vertex sparsifiers that preserve cut/flow values 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 $\mathrm{dist}_G(u,v)$. For a subset S of vertices in G, we define $\mathrm{diam}_G(S) := \max_{u,u'\in S} \mathrm{dist}_G(u,u')$. For a pair of disjoint subsets of vertices (S,S') in G, we define $\mathrm{dist}_G(S,S') := \min_{u\in S,u'\in S'} \mathrm{dist}_G(u,u')$.

Emulators. Throughout, we consider graph G equipped with a special set of vertices T, called *terminals*. We will refer 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 a graph 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

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

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 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).

In this paper we mostly consider instance (G,T) where graph G is a plane graph, which we refer to as *planar instances*. 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, Recall that by definition, a 0-emulator preserves distances exactly, i.e., $\mathrm{dist}_G(x,y) = \mathrm{dist}_{G'}(x,y)$ for all $x,y \in T$.

Theorem 2.1 (Chang-Ophelders [CO20, Theorem 1]) *Given a one-hole instance (G,T) with $|V(G)| = n$ and $|T| = k$, 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 and let F be the face of G on which all terminals of T lie. Assume that no terminal in T is a cut vertex of G, every terminal appears exactly once as we traverse the boundary of F. Let $(t_1,t_2),(t_1',t_2')$ be two pairs of terminals 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 of F 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'),

$$\mathrm{dist}_G(t_1,t_2)+\mathrm{dist}_G(t_1',t_2')\geq \mathrm{dist}_G(t_1',t_2)+\mathrm{dist}_G(t_1,t_2').$$

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 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]. The following lemma is proved 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.*

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.

ε-cover. We use the notion of *ε-cover* [KS98, Tho04]. Let $0<\varepsilon<1$ 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 is $y\in S$ for which $\mathrm{dist}_G(v,y)+\mathrm{dist}_G(y,x)\leq(1+\varepsilon)\cdot\mathrm{dist}_G(v,x)$. Small ε-cover of size $O(1/\varepsilon)$ is known to exist.

Theorem 2.3 (Thorup [Tho04, Lemma 3.4]) *Let $0<\varepsilon<1$ be any 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, Lemma 4] 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 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 $0 < \varepsilon < 1$ 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 $0 < \varepsilon < 1$ and one-hole instance (G, T) with $|T| = k$, one can compute an aligned ε-emulator for (G, T) of size $O(k \log^{O(1)} k/\varepsilon^{O(1)})$ in $O\big((n+k^2)\log^{O(1)} n/\varepsilon^{O(1)}\big)$ time.*

We complement the algorithm 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 $\Omega(k/\varepsilon)$.*

In the rest of this section and the next, all the emulators are aligned, and therefore we omit the word "aligned" for 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. For clarity of exposition, in Sections 3.1 and 4, we only provide a polynomial-time algorithm for Theorem 3.1, and then in Section 4.5 we describe how the algorithm can be implemented in near-linear time. The proof of Theorem 3.2 is deferred to Appendix B.1, since it is not relevant to the proof of Theorem 1.1.

3.1 The Algorithm and its Analysis

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

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

Lemma 3.3 *Given one-hole instance (H, U) with $|U| = r$ and threshold $\lambda := \frac{c^* \log^2 k}{\varepsilon^{20}}$, one can compute a collection $\tilde{\mathcal{H}} = \{(H_1, U_1), \ldots, (H_s, U_s)\}$ of one-hole instances, such that*

- $U \subseteq \big(\bigcup_{1 \leq i \leq s} U_i\big)$;
- $|U_i| \leq 9r/10$ *for each* $1 \leq i \leq s$;

- $\sum_{i:|U_i|\le\lambda} |U_i| \le O(r\log^2 r)$; *and*
- $\sum_{i:|U_i|>\lambda} |U_i| \le r\cdot\left(1+O(\frac{1}{\log^2 r})\right)$.

Moreover, given an ε-emulator (H'_i, U_i) for each $(H_i, U_i) \in \tilde{\mathcal{H}}$, algorithm COMBINE *computes for (H, U) an $\left(\varepsilon + O(\frac{\log^4 r}{r^{0.1}})\right)$-emulator (H', U) of size*

$$|V(H')| \le \left(\sum_{1\le i\le s} |V(H'_i)|\right) + O(r\log^2 r).$$

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* τ. 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 $(H_1, U_1), \ldots, (H_s, U_s)$ generated by Lemma 3.3. The leaves of τ are those that are in $\mathcal{H}$ at the end of the first stage. (To avoid ambiguity, we refer to elements in $V(\tau)$ 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 (H', U) for (H, U) using the algorithm from Theorem 2.1.[5]

We then iteratively process the non-leaf nodes in τ inductively in a bottom-up fashion: Given a non-leaf node (H, U) with children $(H_1, U_1), \ldots, (H_s, U_s)$, let (H'_i, U_i) be the emulator computed for (H_i, U_i) by induction. Apply algorithm COMBINE from Lemma 3.3 to the emulators $(H'_1, U_1), \ldots, (H'_s, U_s)$ to obtain an emulator (H', U) for instance (H, U). After all nodes in τ 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 analysis. We need to bound the size of the emulator G' by $O(k\log^{O(1)} k/\varepsilon^{O(1)})$. In what follows, $\mathcal{H}$ denotes this set at the end of the first stage.

- For each node (H, U) in the partitioning tree τ, we define $f(H, U) := \hat{c}\cdot|U|\log^2|U|$, where $\hat{c} > 0$ is a constant that is greater than all constants hidden in the big-O notations in the statement of Lemma 3.3. By Theorem 2.1 and Lemma 3.3, the final emulator has size

$$|V(G')| \le \sum_{(H,U)\in\mathcal{H}} |U|^2 + \sum_{(H,U)\in V(\tau)} f(H, U).$$

- We bound the term $\sum_{(H,U)\in V(\tau)} f(H, U)$ via a charging scheme as follows. Consider a node (H, U) in τ. We charge the value of $f(H, U)$ uniformly to all terminals in U, so each terminal has a charge of at most $\hat{c}\log^2|U| \le \hat{c}\log^2 k$. The height of τ is at most $O(\log k)$, because at every node that is split by Lemma 3.3, the number of terminals decreases by at least a factor of 9/10. So every terminal in $\bigcup_{(H,U)\in\mathcal{H}} U$ is charged at most $O(\log k)$ times, thus has a total charge at most $O(\log^3 k)$.

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

- It suffices to bound the total number of terminals in all resulting one-hole instances in $\mathcal{H}$ by $O(k\log^{O(1)}k)$, because then the total size of emulators (H',U) for all resulting one-hole instances in $\mathcal{H}$ is

$$\sum_{(H,U)\in\mathcal{H}}|U|^2 \le \max_{(H,U)\in\mathcal{H}}\{|U|\}\cdot\sum_{(H,U)\in\mathcal{H}}|U| \le \lambda\cdot O(k\log^{O(1)}k) = O(k\log^{O(1)}k/\varepsilon^{O(1)}).$$

  Therefore,

$$|V(G')| \le \sum_{(H,U)\in\mathcal{H}}\left(|U|^2+|U|\cdot O(\log^3 k)\right) = O\left(k\cdot\log^{O(1)}k/\varepsilon^{O(1)}\right).$$

It remains to bound the total number of terminals in all one-hole instances in $\mathcal{H}$, which we do next via a charging scheme. Let (H,U) be a node in τ with children $(H_1,U_1),\ldots,(H_s,U_s)$.

- For instances (H_i,U_i) with $|U_i|\le\lambda$ (which will all be in $\mathcal{H}$ at the end of first stage), charge every vertex in U_i to vertices in U. Since $\sum_{i:|U_i|\le\lambda}|U_i| \le O(\log^2|U|\cdot|U|) \le O(\log^2 k\cdot|U|)$, each vertex of U gets a charge of $O(\log^2 k)$ 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 $|U'|\le O(|U|/\log^2 r)$ by Lemma 3.3. Charge every vertex in U' uniformly to vertices in U, so each vertex gets $O(1/\log^2 r)$ charge. We call these charge *active*.

The total inactive charge on each vertex of T is $O(\log k)\cdot O(\log^2 k) = O(\log^3 k)$ because τ has height $O(\log k)$. As for the total active charge to each vertex in T, a quick calculation (which can be found in Appendix B.5) shows that it is at most $O(1/(\log_{(10/9)}\lambda-1)) \le 1/2$. Note that this only accounts for the *direct active charge*. For example, some terminal does not belongs 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 $O(\log^3 k)$. Therefore, we conclude that the total number of terminals in all resulting instances in $\mathcal{H}$ is bounded by $O(k\log^3 k)$, which, from the above discussion, implies that the total size of emulators (H',U) for all resulting one-hole instances in $\mathcal{H}$ is $O(k\log^{O(1)}k/\varepsilon^{O(1)})$.

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 (rooted) partitioning tree τ. We now define, for each tree node (H,U), a value $\varepsilon_{(H,U)}$ as follows. For a leaf node (H,U), we define $\varepsilon_{(H,U)} := 0$. For a non-leaf node (H,U), let $(H_1,U_1),\ldots,(H_s,U_s)$ be its child nodes in τ. 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 τ, the one-hole instance (H',U) we construct is an $\varepsilon_{(H,U)}$-emulator for (H,U).

We now show that $\varepsilon_{(G,T)}\le\varepsilon$. It is easy to see that there exist integers $r_1,\ldots,r_t$, where $r_1\le k$, $r_t\ge\lambda$, and for each $1\le i\le t-1$, $r_i\ge(10/9)\cdot r_{i+1}$, such that $\varepsilon_{(G,T)} = \sum_{1\le i\le t} c\log^4 r_i/r_i^{0.1}$. A quick calculation gives us $\varepsilon_{(G,T)}\le c\cdot(\log\lambda)^4/\lambda^{0.1}$. (For a complete proof see Appendix B.5.) Since c is a constant, and recall that we have set $\lambda = \frac{c^*\log^2 k}{\varepsilon^{20}}$ where $c^*>c^{20}$ is large enough, so $\varepsilon_{(G,T)}<\varepsilon$, 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 $|H| = n$ and $|U| = r$, we charge its running time (and also the time for COMBINE): (i) the $O(n \log^{O(1)} r \cdot \log n)$ term uniformly to all vertices in H, so every vertex is charged at most $O(\log^{O(1)} r \cdot \log n)$; and (ii) the $O(r^2 \log^{O(1)} r \cdot \log n)$ term uniformly to all terminals in U, so every terminal in U is charged at most $O(r \log^{O(1)} r \cdot \log n)$. Using similar charging scheme as described above, we get that the total running time is at most $O(\log^{O(1)} k \cdot \log n)$ times the total number of vertices in all resulting one-hole instances in $\mathcal{H}$, which is at most $n + O(k \log^{O(1)} k)/\varepsilon^{O(1)})$, plus $O(k \log^{O(1)} k \cdot \log n)$ times the total number of terminals in all resulting one-hole instances in $\mathcal{H}$, which is at most $O(k \log^{O(1)} k)/\varepsilon^{O(1)})$. Therefore, the total running time of the algorithm is $O\big((n + k^2) \log^{O(1)} k \cdot \log n/\varepsilon^{O(1)}\big)$.

4 Construct Emulator using Split and Glue

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 in the rest of the section.

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 $\mathcal{P}$; Y must contain all endpoints of paths in $\mathcal{P}$.

The output of procedure SPLIT is a collection of one-hole instances constructed as follows.

Consider the plane graph H with all the terminals in U lying on the outerface of H. Because all the paths in $\mathcal{P}$ are shortest paths in H, each path must be simple and connecting two terminals in U. Given any plane graph H and simple path P, we can *slice*[6] H open along the path P by duplicating every vertex and edge of P to create another path P' identical to P, together bounding a common new boundary component. The set of edges incident to each vertex on P are split into two sides naturally based on the cyclic order around the vertex.

The output of procedure SPLIT is simply the collection $\mathcal{H}$ of one-hole instances formed by iteratively slicing (the remaining of) H using shortest paths from $\mathcal{P}$. 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.

[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* [EFN12] or *incision* [MNNW18] in the literature.

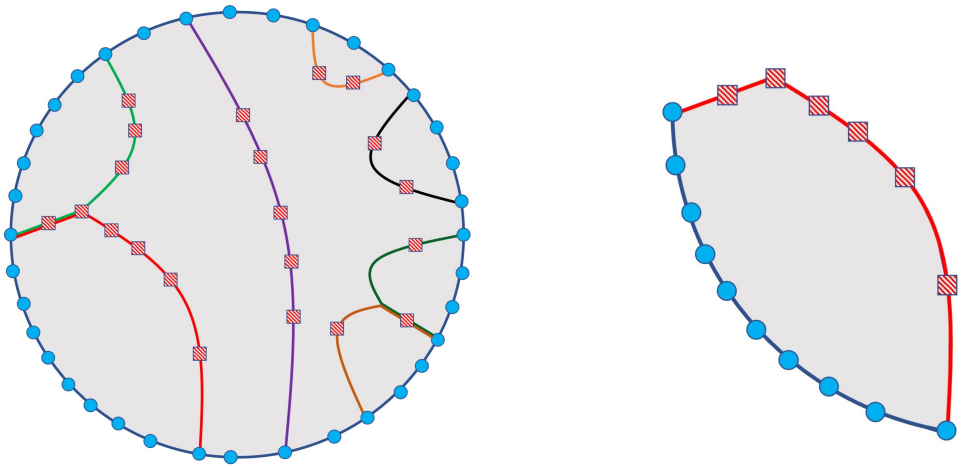


Figure 2. An illustration of splitting a one-hole instance along a path set. *Left*: Graph H, together with terminals in set U (blue), paths in set $\mathcal{P}$ (in different colors), and vertices of Y (red box). *Right*: An output instance (that corresponds to the left bottom region of H) by procedure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

Gluing. We now describe procedure GLUE. Assume that we have applied the procedure SPLIT to the one-hole instance (H, U), and $\mathcal{H} := \{(H_R, U_R)\}$ is the collection of one-hole instances produced. The input to procedure GLUE consists of

- one emulator Z_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 for (H, U), which is constructed as follows.

We start by taking the union of all emulators $\{Z_R\}$, and identifying, for each vertex $y \in Y$, all copies of the vertex. We denote by $\hat{Z}$ the resulting graph. We then add to $\hat{Z}$ all the *branch vertices* of $\mathcal{P}$ vertices that are not in Y (and thus have not been merged), where the *branch vertices* of $\mathcal{P}$ in H are those vertices that has degree at least three in the union of shortest paths in $\mathcal{P}$. For each emulator Z_R, denote the cyclic sequence of all terminals in U_R and the copies of branch vertices appearing around the outerface of Z_R in counter-clockwise order as $\langle v_1, \ldots, v_m \rangle$. For each branch vertex v_i not in U_R, we add an edge connecting v_i to the previous vertex v_{i-1} (might be a terminal or another branch vertex), with the edge-weight $\mathrm{dist}_H(v_i, v_{i-1})$; similarly, add an edge connecting v_i to v_{i+1} with edge-weight $\mathrm{dist}_H(v_i, v_{i+1})$. (See Figure 3 for an illustration.) We denote the resulting graph Z after processing all the emulators Z_R inside $\hat{Z}$. Graph Z is naturally a plane graph by inheriting the embeddings of all Z_Rs. (See Figure 4 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. The size of Z is bounded by $\sum_R |Z_R|$ plus the number of branch vertices.

It is easy to verify that both procedures SPLIT and GLUE can be implemented in $O(|V(H)|)$ time. Now we summarize the behavior of the algorithms with the following three claims; let $((H, U), \mathcal{P}, Y)$ be a valid input to procedure SPLIT and $\mathcal{H}$ be the corresponding output throughout the description. The proofs of Claim 4.1, 4.2, and 4.3 are deferred to Appendix B.2, B.3, and B.4 respectively.

Claim 4.1 *The number of branch vertices in H is at most $O(|U| \cdot \log^2 |U|)$.*

Claim 4.2 *The total number of terminals in all one-hole instances in $\mathcal{H}$ after procedure* SPLIT *is*

$$\sum_{(H_R, U_R) \in \mathcal{H}} |U_R| \leq O\big(|U| \cdot \log^2 |U| + |Y|\big),$$

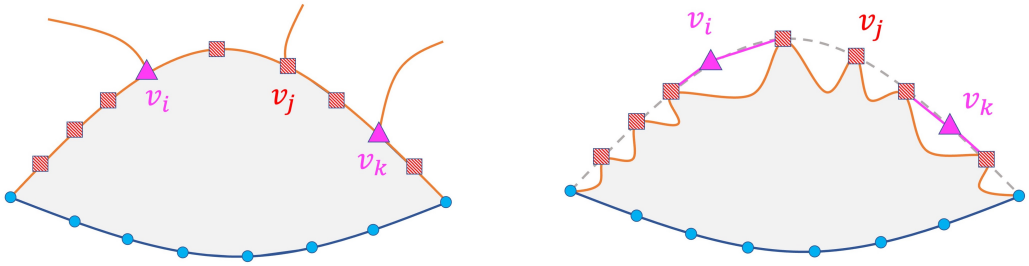


Figure 3. An illustration of adding edges connecting to branch vertices. *Left:* Graph H_R: v_i, v_j, v_k are branch vertices, and only vertices v_i, v_k belong to set $V^* \setminus U_R$. *Right:* Graph Z_R in Z: v_i, v_k do not belong to Z_R but they are connect to vertices of U_R by edges (purple).

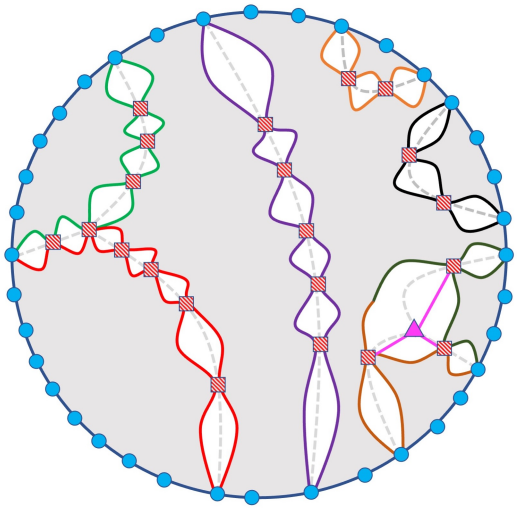


Figure 4. An illustration of gluing one-hole instances at outer-boundaries. Identified vertices of U are shown in blue) and identified vertices of Y are shown in red box. Original boundaries of regions in $\mathcal{R}$ are shown in dash gray.

and for any $2 < \mu < |U|/2$,

$$\sum_{(H_R,U_R)\in\mathcal{H}:|U_R|\geq\mu} |U_R| \leq \big(|U| + O(|Y \setminus U|)\big) \cdot \left(1 + \frac{1}{\mu - 2}\right).$$

Claim 4.3 *Let* $(\hat{H}, U)$ *be the instance obtained by applying the procedure* GLUE *directly to the instances in* $\mathcal{H}$. *The output* (Z, U) *from procedure* GLUE *when applying to the emulators of instances in* $\mathcal{H}$ *is an* ε*-emulator for* $(\hat{H}, U)$.

Later in the proof we will use the divide-and-conquer technique twice. Because the set Y we choose to glue along is different each time, we will postpone the statements that relate $(\hat{H}, U)$ to the original instance (H, U) to the later sections in context. (See Claim 4.5 and Claim 4.9).

4.2 Remove All Cut Vertices

We first compute the set U' of all cut vertices of U in H, and along the way the maximal subgraphs $\hat{H}_1, \ldots, \hat{H}_t$ of H that (i) each contains at least two terminals of U and (ii) do not contain any cut vertex in U'. We define, for each $1 \leq i \leq t$, $\hat{U}_i = U \cap V(\hat{H}_i)$, and it is easy to verify that $(\hat{H}_i, \hat{U}_i)$

is a one-hole instance. Moreover, it is easy to show that if we are given an ε-emulator $(\hat{H}'_i, \hat{U}_i)$ for instance $(\hat{H}_i, \hat{U}_i)$ for each i, then by simply gluing along U', we can obtain an ε-emulator (H', U) by Claim 4.3. We use the following claim in order to bound $\sum_{1\le i\le t}|\hat{U}_i|$ and $\sum_{|\hat{U}_i|\ge\lambda}|\hat{U}_i|$.

Claim 4.4 *$\sum_{1\le i\le t}|\hat{U}_i| \le O(r)$, and $\sum_{|\hat{U}_i|\ge\lambda}|\hat{U}_i| \le |U|\cdot(1+O(\frac{1}{\log^2 r}))$.*

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

We partition set V' into three subsets: V'_1 contains all leaf nodes of τ', V'_2 contains all nodes of degree 2 in τ', and $V'_{\ge3}$ contains all nodes of degree at least 3 in τ'. Observe that, for each node $v_i\in V'_1$, since instance $(\hat{H}_i, \hat{U}_i)$ contains at least two terminals, at least one terminal in $\hat{U}_i$ does not belong to any other instance in $\{(\hat{H}_1,\hat{U}_1),\ldots,(\hat{H}_t,\hat{U}_t)\}$. Therefore, $|V'_1|\le r$. Since τ' is a tree, it follows that $|V'_{\ge3}|\le|V'_1|\le r$. Since for every node in V'_2, all its neighbors lie in U', we get that $|V'_2|\le|U'|\le r$. Altogether, we get that $|V(\tau')|\le O(r)$. Note that for every terminal $u'\in U'$, the number of sets $\hat{U}_i$ that contains u is exactly $\deg_{\tau'}(u')$. Therefore,

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

We now bound $\sum_{|\hat{U}_i|\ge\lambda}|\hat{U}_i|$ via a charging scheme. We root the tree τ' 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 τ'. We denote by $v_1,\ldots,v_s$ the child nodes of u'. For each $1\le i\le s$, if $|\hat{U}_i|\ge\lambda$, 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 $v_1,\ldots,v_s$ from τ' and recurse on the remaining tree, until the tree contains no nodes of U'. It is easy to observe that the value of $\sum_{|\hat{U}_i|\ge\lambda}|\hat{U}_i|$ is at most r plus the total charge. We now show that the total charge is bounded by $O(1/\log^2 r)$. 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 charged for other terminals in U'. It is easy to observe that, the total direct and indirect charge is bounded by $2/\lambda+(2/\lambda)^2+\cdots\le 4/\lambda$. Therefore, since $\lambda\ge\log^2 r$, we get that $\sum_{|\hat{U}_i|\ge\lambda}|\hat{U}_i| \le r\cdot(1+O(\frac{1}{\log^2 r}))$. □

Note that we can simply return the collection $\{(\hat{H}_i,\hat{U}_i)\mid 1\le i\le t\}$ of instances as output, unless some set $\hat{U}_i$ contains more than $0.9r$ terminals. However, from Claim 4.4, 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 $\tilde{\mathcal{H}}=\tilde{\mathcal{H}}'\cup\{(\hat{H}_i,\hat{U}_i)\mid 2\le i\le t\}$. It is easy to verify from the above discussion that all conditions of Lemma 3.3 hold for the output 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) 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 boundary of the face that contains all terminals in U, then every terminal of U appear exactly once.

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

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

Denote the spread of instance (H, U) by $\Phi := \Phi(H, U)$. We distinguish between the following two cases, depending on whether or not Φ is small or large.

4.3 Small Spread Case

In this case we assume $\Phi \leq 2^{r^{0.9} \log^2 r}$. 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 $\varepsilon_r = O(\log^4 r / r^{0.1})$.

Balanced terminal pairs. Let (H, U) be a one-hole instance with $U = \{u_1, \ldots, u_r\}$, 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 $j - i \leq c \cdot r$ and $i + r - j \leq c \cdot r$. 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 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 in H, and denote it by P. We let set Y contain the endpoints of P, together with the following vertices of P: for each $1 \leq i \leq L_r$,

(i) among all vertices v of P with $\mathrm{dist}_P(v, u) \leq e^{i\varepsilon_r}$, vertex that maximizes its distance to u;

(ii) among all vertices v of P with $\mathrm{dist}_P(v, u) \geq e^{i\varepsilon_r}$, vertex that minimizes its distance to u;

(iii) among all vertices v of P with $\mathrm{dist}_P(v, u') \leq e^{i\varepsilon_r}$, vertex that maximizes its distance to u';

(iv) among all vertices v of P with $\mathrm{dist}_P(v, u') \geq e^{i\varepsilon_r}$, vertex that minimizes its distance to u'.

In other words, if we think of path P as a line, and then mark, for each $1 \leq j \leq L_r$, 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, $|Y| \leq 4L_r$.

We apply the procedure Split to the one-hole instance (H, U), the path set that contains a single path P and vertex set Y defined above. Let (H_1, U_1) and (H_2, U_2) be the one-hole instances we get. We then return the collection $\{(H_1, U_1), (H_2, U_2)\}$.

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 $|Y| \leq 4L_r = r/25 \log^2 r$, we

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

get that $|U_1| \leq (3/4)r + r/25\log^2 r \leq (9/10)r$, and similarly $|U_2| \leq (9/10)r$. Second, note that $|U_1| + |U_2| \leq |U| + O(|Y|) \leq r \cdot (1 + O(L_r/r)) = r \cdot (1 + O(\frac{1}{\log^2 r}))$.

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 instances $(H_1', U_1), (H_2', U_2)$. 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 algorithm COMBINE satisfies the required properties. Note that instance pair $(H_1, U_1), (H_2, U_2)$ is also a valid input for procedure GLUE. Let $(\hat{H}, \hat{U})$ be the one-hole instance that it outputs. It is easy to verify that $\hat{U} = U$. We use the following claim.

Claim 4.5 *Instance $(\hat{H}, U)$ is a $(3\varepsilon_r)$-emulator for instance (H, U).*

We provide the proof of Claim 4.5 right after we complete the analysis for the small spread case. From Claim 4.3, (H', U) is an ε-emulator for $(\hat{H}, U)$. From Claim 4.5, 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).

Proof of Claim 4.5. We will show that, for each pair u_1, u_2 of terminals in U, $\mathrm{dist}_H(u_1, u_2) \leq \mathrm{dist}_{\hat{H}}(u_1, u_2) \leq e^{3\varepsilon_r} \cdot \mathrm{dist}_H(u_1, u_2)$.

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 graph $\hat{H}_1, \hat{H}_2$, respectively. See Figure 5 for an illustration.

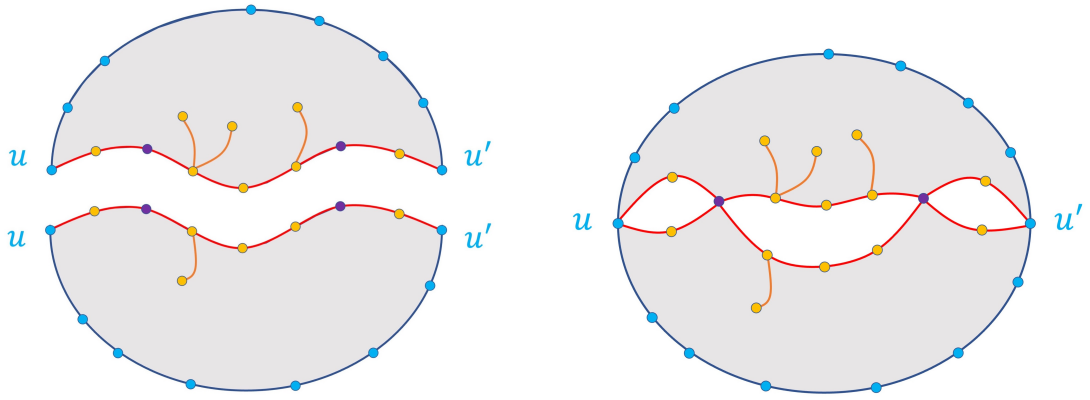


Figure 5. An illustration graphs $\hat{H}$, H_1, and H_2. *Left:* Graphs H_1 (top) graph H_2 (bottom) viewed as individual graphs. *Right:* Subgraphs $\hat{H}$ obtained by gluing graphs H_1 and H_2. Vertices in $Y \setminus \{u, u'\}$ are shown in purple.

We first show that for each pair u_1, u_2 of terminals in U, $\mathrm{dist}_H(u_1, u_2) \leq \mathrm{dist}_{\hat{H}}(u_1, u_2)$. 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 graph H_1, and therefore $\mathrm{dist}_H(u_1, u_2) \leq \mathrm{dist}_{\hat{H}}(u_1, u_2)$. 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

(i) a path Q_1 in $\hat{H}_1$ connecting u_1 to some vertex $x_1 \in V(P^1)$, that does not contain vertices of $V(P^1)$ as inner vertices;

(ii) a subpath R^1 of P^1 connecting x_1 to a vertex $y \in Y$;

(iii) a subpath R^2 of P^2 connecting y to a vertex x_2; and

(iv) a path Q_2 in $\hat{H}_2$ connecting x_2 to u_2 that does not contain vertices of $V(P^2)$ as inner vertices.

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, $\mathrm{dist}_H(u_1,u_2) \leq \mathrm{dist}_{\hat{H}}(u_1,u_2)$.

From now on we focus on showing that, for each pair u_1,u_2 of terminals in U, $\mathrm{dist}_{\hat{H}}(u_1,u_2) \leq e^{3\varepsilon_r} \cdot \mathrm{dist}_H(u_1,u_2)$. 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 $\mathrm{dist}_{\hat{H}}(u_1,u_2) = \mathrm{dist}_H(u_1,u_2)$. 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 path Q and path 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. Denote by Q_1 the subpath of Q between u_1 and x_1, denote by Q_2 the subpath of Q between u_2 and x_2, and denote by Q' the subpath of Q between x_1 and x_2. We consider the following possibilities, depending on the relative locations of points x_1,x_2.

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, $\mathrm{dist}_{\hat{H}}(u_1,u_2) \leq \mathrm{dist}_H(u_1,u_2)$.

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

Observation 4.6 $\mathrm{dist}_H(x_1,u_1) \geq \mathrm{dist}_H(x_1,u)$.

Proof: Assume not, then

$$\mathrm{dist}_H(u_1,u) \leq \mathrm{dist}_H(x_1,u_1) + \mathrm{dist}_H(x_1,u) < 2 \cdot \mathrm{dist}_H(x_1,u) \leq \mathrm{dist}_H(u,u'), \text{ and}$$

$$\mathrm{dist}_H(u_1,u') \leq \mathrm{dist}_H(x_1,u_1) + \mathrm{dist}_H(x_1,u') < \mathrm{dist}_H(x_1,u) + \mathrm{dist}_H(x_1,u') \leq \mathrm{dist}_H(u,u').$$

So $\mathrm{dist}_H(u_1,u), \mathrm{dist}_H(u_1,u') < \mathrm{dist}_H(u,u')$. However, since $|U|/2 \leq |V(H_1) \cap U| \leq (3/4) \cdot |U|$, 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. □

Think of path P as a line connecting u to u'. We now mark, for each $1 \leq j \leq L_r$, 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 $\text{dist}_P(x,u) = e^{i\varepsilon_r}$. Let y be the vertex of Y closest to the landmark x that lies between x and x_1. From the construction of portals, $e^{i\varepsilon_r} \le \text{dist}_P(y,u) < \text{dist}_P(x_1,u), \text{dist}_P(x_2,u) < e^{(i+1)\varepsilon_r}$. Therefore, $\text{dist}_P(x_1,y), \text{dist}_P(x_2,y) \le (e^{\varepsilon_r}-1)\cdot e^{i\varepsilon_r}$. 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

$$\begin{aligned}
& \text{dist}_{\hat{H}_1}(u_1,x_1^1) + \text{dist}_{\hat{H}_1}(x_1^1,y) + \text{dist}_{\hat{H}_2}(x_2^2,y) + \text{dist}_{\hat{H}_2}(u_2,x_2^2) \\
= \;& \text{dist}_H(u_1,x_1) + \text{dist}_P(x_1,y) + \text{dist}_P(x_2,y) + \text{dist}_H(u_2,x_2) \\
= \;& \text{dist}_H(u_1,x_1) + \text{dist}_H(u_2,x_2) + \text{dist}_P(x_1,x_2) + \big(\text{dist}_P(x_1,y) + \text{dist}_P(x_2,y) - \text{dist}_P(x_1,x_2)\big) \\
\le \;& \text{dist}_H(u_1,u_2) + 2\cdot(e^{\varepsilon_r}-1)\cdot e^{i\varepsilon_r} \\
\le \;& \text{dist}_H(u_1,u_2) + 2\cdot(e^{\varepsilon_r}-1)\cdot \text{dist}_H(u,x_1) \\
\le \;& \text{dist}_H(u_1,u_2) + 2\cdot(e^{\varepsilon_r}-1)\cdot \text{dist}_H(u_1,x_1) \qquad \text{(from Observation 4.6)} \\
\le \;& e^{3\varepsilon_r}\cdot \text{dist}_H(u_1,u_2).
\end{aligned}$$

Therefore, $\text{dist}_{\hat{H}}(u_1,u_2) \le e^{3\varepsilon_r}\cdot \text{dist}_H(u_1,u_2)$. This completes the proof of Claim 4.5.

4.4 Large Spread Case

In this case we assume $\Phi > 2^{r^{0.9}\log^2 r}$. Without loss of generality $\min_{u,u'\in U}\text{dist}_G(u,u') = 1$, so $\max_{u,u'\in U}\text{dist}_G(u,u') = \Phi$. 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, $\mathcal{S}_0 = \{\{u\} \mid u \in U\}$.
- 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 $V_{i-1} = \{v_S \mid S \in \mathcal{S}_{i-1}\}$. 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 and only if $\text{dist}_{W_{i-1}}(v_S, v_{S'}) \le \mu^i$, or equivalently $\text{dist}_H(S,S') \le \mu^i$. 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 $S' \subseteq S$.

We use the following simple observation.

Observation 4.7 *For each set S in collection $\mathcal{S}_i$, $\mathrm{diam}_G(S) \leq 2r \cdot \mu^i$.*

Proof: We prove Observation 4.7 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 $0 \leq 2r\mu^0$. 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, $\mathrm{dist}_G(u,u') \leq 2r \cdot \mu^{i-1} \leq 2r \cdot \mu^i$. 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 $b \leq r-2$ (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 $0 \leq j \leq b+1$, 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 $0 \leq j \leq b+1$, $\mathrm{dist}_G(\hat{u}_j, \hat{u}'_j) \leq 2r \cdot \mu^{i-1}$; and
- for each $0 \leq j \leq b$, $\mathrm{dist}_G(\hat{u}'_j, \hat{u}_{j+1}) \leq \mu^i$.

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

It would be convenient for us to associate a partitioning tree τ with the hierarchical partitioning $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_L)$ we have computed, in a natural way as follows. Its vertex set $V(\tau)$ is $V(\tau) = V_0 \cup \ldots \cup V_L$ (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 τ, and we call sets in $\mathcal{S}_i$ *level-i sets*. Since $\mathcal{S}_L = \{U\}$, there is only one level-L node in τ, that we view as the root of τ. The edge set $E(\tau)$ of τ 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 τ is indeed a tree.

Observation 4.8 *Let S, S' be distinct sets in $\mathcal{S}$ with $S \cap S' = \emptyset$. 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) are non-crossing.*

Proof: Assume for contradiction that the pairs (u_1, u_2) and (u'_1, u'_2) are crossing. 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 in S such that $\mathrm{dist}_G(u_3, u_4) \leq \mu^i$, $\mathrm{dist}_G(u'_3, u'_4) \leq \mu^{i'}$ 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 $0 \leq j \leq t$, $\mathrm{dist}_G(u^j, u^{j+1}) \leq \mu^i$; and for every pair u, u' of terminals do not belong to the same level-i set, $\mathrm{dist}_G(u,u') > \mu^i$.

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 $\mathrm{dist}_G(u_3,u_4)\leq\mu^i$.

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 $\mathrm{dist}_G(u_3',u_4')\leq\mu^{i'}$. Note that, since $u_3,u_4\in S$ and $u_3',u_4'\notin S$, $\mathrm{dist}_G(u_3,u_3')>\mu^i$ and $\mathrm{dist}_G(u_4,u_4')>\mu^i$. Altogether, we get that

$$\mathrm{dist}_G(u_3',u_4')+\mathrm{dist}_G(u_3,u_4)\leq\mu^i+\mu^{i'}\leq\mu^i+\mu^i<\mathrm{dist}_G(u_3,u_3')+\mathrm{dist}_G(u_4,u_4'),$$

a contradiction to the Monge property on the crossing pairs (u_3,u_4) and (u_3',u_4'). □

Expanding sets. The central notion in the algorithm for the large spread case is the notion of *expanding sets*. Recall that $\varepsilon_r'=\frac{1}{r^{0.7}}$. We say that a set $S\in\mathcal{S}$ is *expanding*, if and only 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 τ), otherwise we say it is *non-expanding*. We now distinguish between two cases, depending on whether or not $\mathcal{S}$ contains a non-expanding set with moderate size.

4.4.1 Balanced Case: There are non-expanding S such that $r/5\leq|S|\leq 4r/5$.

We let $\hat{S}$ be the parent set of S. We denote $S^*=\hat{S}\setminus S$, and $S'=U\setminus\hat{S}$, so sets S^*,S,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 walking 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 Theorem 2.2 to graph H and the set $\mathcal{M}$ of pairs, and 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 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 such ε_r-covers and the endpoints of $P_{(u,u')}$, so $Y_{(u,u')}$ is a vertex set of $P_{(u,u')}$. Denote $Y=\bigcup_{(u,u')}Y_{u,u'}$, so Y is a vertex set of $V(\mathcal{P})$ that contains all endpoints of paths in $\mathcal{P}$. Moreover, from Lemma 2.4,

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

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 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, from the construction of the border path set $\mathcal{P}$, the 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. Recall that $r/6 \leq |S|, |S'| \leq 5r/6$ and $|Y| \leq O(r^{0.4}/\log^4 r)$. On the one hand, each instance in $\mathcal{H}_1$ contains at most two terminals in S, and so it contains at most $r - |S| + 2 + |Y| \leq (9/10)r$ terminals. On the other hand, each instance in $\mathcal{H}_2$ does not contain terminals in S', and so it contains at most $r - |S'| + |Y| \leq (9/10)r$ terminals.

Second, note that $|Y| \leq O\big(\frac{r^{0.4}}{\log^4 r}\big)$ and $\lambda = \frac{c^* \log^2 k}{\varepsilon^{20}}$, then from Claim 4.2 (by setting $\mu = \lambda$), we get that $\sum_{(H_i,U_i)\in\mathcal{H}} |U_i| \leq O(r \log^2 r)$ and $\sum_{(H_i,U_i)\in\mathcal{H}:|U_i|>\lambda} |U_i| \leq |U| \cdot \big(1 + O(\frac{1}{\log^2 r})\big)$.

We now construct an algorithm Combine that satisfies the required properties. Recall that we are given, for each instance $(H_i, U_i) \in \mathcal{H}$, an ε-emulator (H_i', U_i) for instance (H_i, U_i). The algorithm Combine simply applies the procedure Glue to instances $(H_1', U_1), \ldots, (H_s', U_s)$ and let (H', U) be the output one-hole instance that it outputs. The algorithm Combine simply returns instance (H', U). It remains to show that the algorithm Combine satisfies the required properties. Note that one-hole instances $(H_1, U_1), \ldots, (H_s, U_s)$ is also 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 $(H_1, U_1), \ldots, (H_s, U_s)$. It is easy to verify that $\hat{U} = U$. We use the following claim, whose proof comes right after the proof of Lemma 3.3.

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 $1 \leq i \leq t$, (H_i', U_i) is an ε-emulator for (H_i, U_i), from Claim 4.3, (H', U) is an ε-emulator for $(\hat{H}, U)$. Then from Claim 4.5 and Claim 4.9, we get that (H', 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, the number of vertices in graph H' that does not belong to graphs $H_1', \ldots, H_s'$ are the branch vertices in graph H (vertices that appear on the boundary of at least three regions when applying the procedure Split to it), and from Claim 4.1 the number of such vertices is bounded by $O(r \log^2 r)$. Therefore, $|V(H')| \leq \big(\sum_{1\leq i\leq s} |V(H_i')|\big) + O(\log^2 r \cdot |U|)$.

Proof (of Claim 4.9): We denote by L the level that set S belongs to. We use the following simple observations.

Observation 4.10 *For every pair $u, u' \in U$ with $u \in S$ and $u' \in S'$, $\mathrm{dist}_H(u, u') \geq \mu^{L+1}$. For every pair u, u' of terminals in S' that do not belong to the same graph in $\{H_R \mid R \in \mathcal{R}\}$, $\mathrm{dist}_H(u, u') \geq \mu^{L+1}$.*

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 $(L+1)$-level set, and so $\mathrm{dist}_H(u, u') > \mu^{L+1}$. Consider now a pair u, u' of terminals in S' that do not belong to the same region in $\mathcal{R}$. From the construction of the regions in $\mathcal{R}$, 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,

$$\mathrm{dist}_H(u, u') \geq \mathrm{dist}(u, \hat{u}) + \mathrm{dist}(u', \hat{u}') - \mathrm{dist}(\hat{u}, \hat{u}') \geq \mu^{L+1} + \mu^{L+1} - 2r\mu^L > \mu^{L+1},$$

where we have used the fact (from Observation 4.7) that $\mathrm{dist}(\hat{u}, \hat{u}') \leq 2r\mu^L$. □

Let u, u' be terminals in U. We will show that $e^{\varepsilon_r} \cdot \mathrm{dist}_{\hat{H}}(u, u') \leq \mathrm{dist}_H(u, u') \leq e^{\varepsilon_r} \cdot \mathrm{dist}_{\hat{H}}(u, u')$. 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, from Theorem 2.2 it is easy to see that $\mathrm{dist}_H(u, u') = \mathrm{dist}_{\hat{H}}(u, u')$.

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 each instance in $\mathcal{H}$ corresponds to a region of (H, U), separated by the paths in $\mathcal{P}$. Recall that all vertices in H that appear on the boundary of at least three regions in $\mathcal{R}$ are called branch vertices, and according to the procedure GLUE the set V^* of all branch vertices belong to graph $\hat{H}$. We say that an instance $(H_R, U_R) \in \mathcal{H}$ is a *regular* instance, if and only if the region R that it corresponds to is surrounded by (i) a contiguous segment of the outer-boundary of H; and (ii) the image of a single path in $\mathcal{P}$. From Theorem 2.2, 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 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. $v \neq v'$**.** Since the intersection between Q with any other graph $H_{\hat{R}}$ is a segment of the boundary of $\hat{R}$, it is easy to see that both v and v' are branch vertices. From the construction of graph $\hat{H}$, it is easy to verify that the entire path Q is also contained in graph $\hat{H}$, so $\text{dist}_{\hat{H}}(u, u') \leq \text{dist}_H(u, u')$. 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 $\text{dist}_{\hat{H}}(u, u') \geq \text{dist}_H(u, u')$. Therefore, in this case we have that $\text{dist}_{\hat{H}}(u, u') = \text{dist}_H(u, u')$.

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 4.10 and the fact (from Observation 4.7) that the boundary path of R and R' have total length at most $2r\mu^L$, it is easy to verify that

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

If both u, u' belong to S, then from the construction of $\hat{H}$, $\text{dist}_H(u, u') \leq \text{dist}_{\hat{H}}(u, u')$. 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 $\hat{v}$ that either belongs to $Y \cap U_R$ or is a branch vertex, such that $\text{dist}_H(u, \hat{v}) + \text{dist}_H(\hat{v}, v) \leq e^{\varepsilon} \text{dist}_H(u, v)$. 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

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

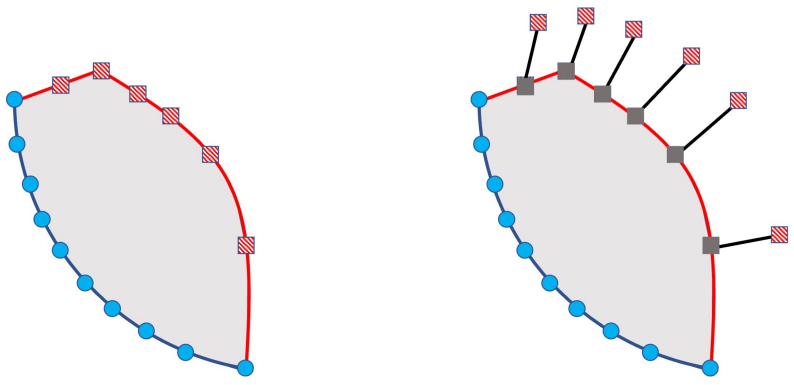
4.4.2 Unbalanced Case: Every S is either expanding, or $|S| < r/5$, or $|S| > 4r/5$.

Step 1: Reducing the spread from above. We say that a set $S \in \mathcal{S}$ is *heavy* if and only if $|S| > 4r/5$, and in this case we also say that the node v_S is heavy. Clearly, every level of τ contains at most one heavy node, and the set of heavy nodes induce a path in τ which ends at the root node of τ. Let $\hat{S}$ be the non-expanding heavy set that lies on the lowest level. We denote by $\tilde{L}$ 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 $|\hat{S}^*| \leq (e^{\varepsilon'_r} - 1)r$. 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 of instances we obtain. From similar analysis as in Section 4.4.1, we get that $\sum_{(H_i,U_i)\in\hat{\mathcal{H}}}|U_i| \leq O(r\log^2 r)$, and $\sum_{(H_i,U_i)\in\hat{\mathcal{H}}:|U_i|>\lambda}|U_i| \leq r\cdot\left(1+O(\frac{1}{\log^2 r})\right)$. If additionally we have, for each $(H_i,U_i)\in\hat{\mathcal{H}}$, $|U_i|\leq 0.9r$, 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 $|U_{i^*}|>0.9r$. Since $\sum_{(H_i,U_i)\in\hat{\mathcal{H}}:|U_i|>\lambda}|U_i| \leq r\cdot\left(1+O(\frac{1}{\log^2 r})\right)$, 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 second 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).

Step 2: Reducing the spread from below. The goal of this final step is to further modify and decompose the instance (H_{i^*},U_{i^*}) into instances with bounded spread, and eventually apply the algorithm from Case 1 to them. Consider the instance (H_{i^*},U_{i^*}). From the algorithm Split, the instance (H_{i^*},U_{i^*}) corresponds to a region of graph 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), $\mathrm{dist}_H(v,v')=\mathrm{dist}_{H_{i^*}}(v,v')$. 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 $\mathcal{P}$) 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 $\min_{u,u'}\mathrm{dist}_H(u,u')$) 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 $\mu^{\tilde{L}-1}$. 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 6 for an illustration. It is easy to verify that $(\tilde{H},\tilde{U})$ is a one-hole instance, and moreover, for each new terminal in $\tilde{U}\setminus\tilde{S}$, the distance in $\tilde{H}$ from it to any other terminal in $\tilde{U}$ is at least $\mu^{\tilde{L}-1}$. We denote $\tilde{Y}=\{\tilde{u}\mid u\in Y_{i^*}\}$, so $\tilde{U}=\tilde{S}\cup\tilde{Y}$. We will show later in the analysis that it is now sufficient to prove Lemma 3.3 for the instance $(\tilde{H},\tilde{U})$.



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

Figure 6. An illustration of modifying the instance (H_{i^*},U_{i^*}).

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 $\tilde{\tau}$ 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 $\tilde{\tau}$ has depth $\tilde{L}$, and in levels $\tilde{L}-1, \ldots, 1$, new terminals in $\tilde{U} \setminus \tilde{S}$ only form singleton sets as each of them is at distance at least $\mu^{\tilde{L}-1}$ from any other terminal in $\tilde{U}$. Therefore, every non-singleton set in $\tilde{\mathcal{S}}$ is also a set in $\mathcal{S}$.

We say that a set is *good* if

(i) $|S| > 1$;

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

(iii) S is non-expanding; and

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

We denote by $\tilde{\mathcal{S}}_g$ the collection of all good sets in $\tilde{\mathcal{S}}$. We prove in the following observation that all good sets in $\tilde{\mathcal{S}}_g$ lie on level at least $\tilde{L} - O(\log r/\varepsilon_r')$. From property (ii), and our assumption for Case 2 that any set $S \in \mathcal{S}$ with $r/5 \leq |S| \leq 4r/5$ 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 4.11 *All good sets in $\tilde{\mathcal{S}}_g$ lie on level at least $\tilde{L} - 10\log r/\varepsilon_r'$. Every terminal either forms a singleton set on level at least $\tilde{L} - 10\log r/\varepsilon_r'$, or belongs to some good set in $\tilde{\mathcal{S}}_g$.*

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

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

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

Similarly, if a terminal in $\tilde{S}$ does not form a singleton set on level at least $\tilde{L} - 10\log r/\varepsilon_r'$, 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. □

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 4.12 *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 4.8, pairs $(u_1, u_2), (u_1', u_2')$ are non-crossing, a contradiction. □

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 and only 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 set $Y = \bigcup_{S\in\tilde{\mathcal{S}}_g} Y^S$, so Y is a vertex set of $V(\tilde{\mathcal{P}})$ that contains all endpoints of paths in $\tilde{\mathcal{P}}$. Moreover, from Lemma 2.4,

$$|Y| \leq O\Big(\sum_{S\in\tilde{\mathcal{S}}_g}\frac{|S^*|}{\varepsilon_r}\Big) \leq O\Big(\frac{(e^{\varepsilon'_r}-1)\cdot\sum_{S\in\tilde{\mathcal{S}}_g}|S|}{\varepsilon_r}\Big) \leq O\Big(\frac{(e^{\varepsilon'_r}-1)\cdot r}{\varepsilon_r}\Big) = O\Big(\frac{(1/r^{0.7})\cdot r}{\log^4 r/r^{0.1}}\Big) = O\Big(\frac{r^{0.4}}{\log^4 r}\Big).$$

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 $|\hat{U}|\leq 0.9r$, 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 $|\hat{U}|>0.9r$. From similar analysis in Step 1, there can be at most one such instance. We denote such an instance by $(\hat{H},\hat{U})$.

From the algorithm Split, instance $(\hat{H},\hat{U})$ corresponds to a region of $\tilde{H}$ surrounded by segments of paths in $\tilde{\mathcal{P}}$. Let $Q_1,\ldots,Q_k$ be the segments of the boundary of $\hat{H}$. If there are two segments Q_i,Q_j that are subpaths of shortest paths in the same set $\tilde{\mathcal{P}}_S$, then from the definition of border pairs and border path sets, set $\hat{U}$ may not contain terminals in any good set of $\tilde{\mathcal{S}}_g$ other than S. However, since $|S|\leq r/5$ and $|Y|\leq O\big(\frac{r^{0.4}}{\log^4 r}\big)$, this is a contradiction to the assumption that $|\hat{U}|>0.9r$. Therefore, for each set $S\in\tilde{\mathcal{S}}_g$, the boundary of $\hat{H}$ contains at most one segment of some border path of $\tilde{\mathcal{P}}_S$.

We now modify the instance $(\hat{H},\hat{U})$ as follows. Denote $L^* = \tilde{L} - 10\log r/\varepsilon'_r$. 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 4.13 $\Phi(H^*,U^*)\leq 2^{O(\log^2 r/\varepsilon'_r)}$.

Proof: From Observation 4.11, 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 $\mu^{\tilde{L}+1}$ in graph H^*. Therefore, $\Phi(H^*,U^*)\leq\mu^{\tilde{L}-L^*+2} = 2^{O(\log^2 r/\varepsilon'_r)}$ 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 Case 1 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 Unbalanced Case. Recall that in this step we assume that, after Step 1, there is an instance (H_{i^*},U_{i^*}) with $|U_{i^*}|>0.9r$, 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-$\tilde{L}$ clusters in $\mathcal{S}$. From the construction of $\tilde{\mathcal{S}}$, $\mathrm{dist}_H(u,u')\geq\mu^{\tilde{L}}$. Therefore, the transformation from instance (H_{i^*},U_{i^*}) to instance $(\tilde{H},\tilde{U})$ adds at most an additive $\mu^{\tilde{L}-1}$ to their distance, which is at most $O(\frac{1}{\mu}) = O(\frac{1}{r^2}) \leq O(\frac{\log^4 r}{r^{0.1}})$-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 $|\hat{U}| \leq 0.9r$. Since $|Y| \leq O\big(\frac{r^{0.4}}{\log^4 r}\big)$, from Claim 4.2, $\sum_{(\hat{H},\hat{U}) \in \tilde{\mathcal{H}}} |\hat{U}| \leq O(r \log^2 r)$ and $\sum_{(\hat{H},\hat{U}) \in \tilde{\mathcal{H}}: |\hat{U}| > \lambda} |\hat{U}| \leq r \cdot \big(1 + O(\frac{1}{\log^2 r})\big)$. 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 denote the obtained instance by $(\tilde{H}', \tilde{U})$. The proof that instance $(\tilde{H}', \tilde{U})$ is indeed a $\big(\varepsilon + O(\varepsilon_r)\big)$-emulator for $(\tilde{H}, \tilde{U})$ and the proof that $|V(\tilde{H}')| \leq \big(\sum_{1 \leq i \leq s} |V(\hat{H}'_i)|\big) + O(\log^2 r \cdot |U|)$ 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 $|\hat{U}| > 0.9r$. 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 $|\overline{U}| \leq 0.9r$. 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 $0.9r$ terminals. Second, from similar arguments, we get that $\sum_{(\overline{H},\overline{U}) \in \overline{\mathcal{H}}} |\overline{U}| \leq O(r \log^2 r)$ and $\sum_{(\overline{H},\overline{U}) \in \overline{\mathcal{H}}: |\overline{U}| > \lambda} |\overline{U}| \leq r \cdot \big(1 + O(\frac{1}{\log^2 r})\big)$. 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 $\mathrm{dist}_(S, S') \geq \mu^{(\max\{i,i'\}+1)} \geq \mu^{L^*}$. 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 a $\big(\varepsilon + O(\varepsilon_r)\big)$-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

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

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 $|U_{i^*}| > 0.9r$, 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. Recall that we are given a planar instance (G, T) with $|V(G)| = n$ and $|T| = k$. Intuitively, our algorithm first computes an $O(n/k)$-division of graph G using the algorithm from Lemma 5.4, reducing it to a collection of $O(1)$-hole instances each containing $O(1)$ many terminals in T. For each instance we further decompose it into one-hole instances. Computes ε-emulator for each one-hole instance using Theorem 3.1, and finally glues all these ε-emulators together to get the ε-emulator for the input instance (G, T).

We will first show an algorithm for computing ε-emulators for $O(1)$-hole instances in Section 5.1, and then show in Section 5.2 an algorithm for computing an ε-emulator (G', T) for a general planar instance (G, T) with $|V(G')| = k^{1+o(1)}/\varepsilon^{O(1)}$, followed by describing how the algorithm in Section 5.2 can be implemented in near-linear time.

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

In this section we present a near-linear time algorithm for constructing 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 emulators or ε-emulators. The main result of this section is the following lemma.

Lemma 5.1 *For any parameter $0<\varepsilon<1$ and any h-hole instance (H,U) with $|H|=n$ and $|U|=r$, there exists another h-hole instance (H',U) that is an ε-emulator for (H,U), such that $|V(H')|\leq O(h^{2h}\cdot r\log^{O(h)}r/\varepsilon^{O(h)})$. The running time of the algorithm is $O\big((n+r^2)\cdot(h\log n/\varepsilon)^{O(h)}\big)$.*

We first introduce basic algorithms for splitting and gluing h-hole instances that are similar to the basic algorithms for splitting and gluing one-hole instances in Section 4.1.

Procedure $\textsc{Split}_h$ takes the following items as input:

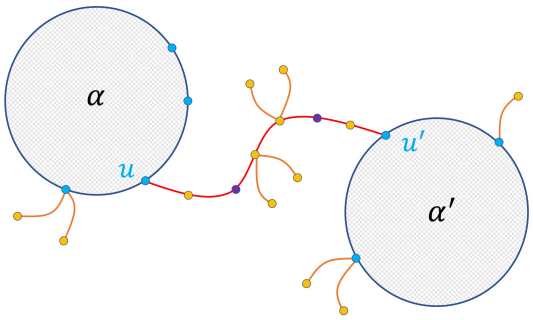
- an h-hole instance (H,U) (with $h>1$);
- a path P connecting a pair of terminals lying on two different holes; and
- a set of vertices Y on the path.

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 7(b). Procedure $\textsc{Glue}_h$ applies on an emulator computed for the $(h-1)$-hole instance $(\tilde{H},\tilde{U})$ that $\textsc{Split}_h$ generated by slicing along P between the two holes α and α', and output an emulator for the original instance (H,U) by identifying all the copies of vertices in Y on the path P, as illustrated in Figure 7(c). A complete description of these procedures are deferred to Appendix C.1.

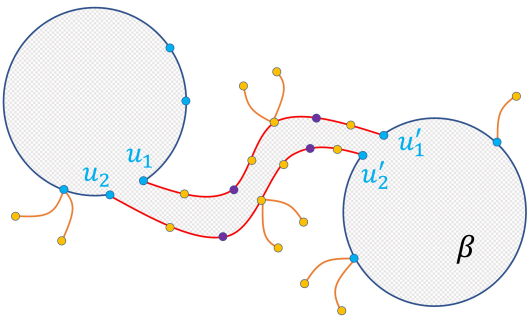
Note that instance $(\tilde{H},\tilde{U})$ is also a valid input for procedure $\textsc{Glue}_h$. Let $(\hat{H},\hat{U})$ be the h-hole instance obtained by applying procedure $\textsc{Glue}_h$ to instance $(\tilde{H},\tilde{U})$. Clearly, $\hat{U}=U$. We use the following claim, whose proof is similar to Claim 4.3 and thus is deferred to Appendix C.2.

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

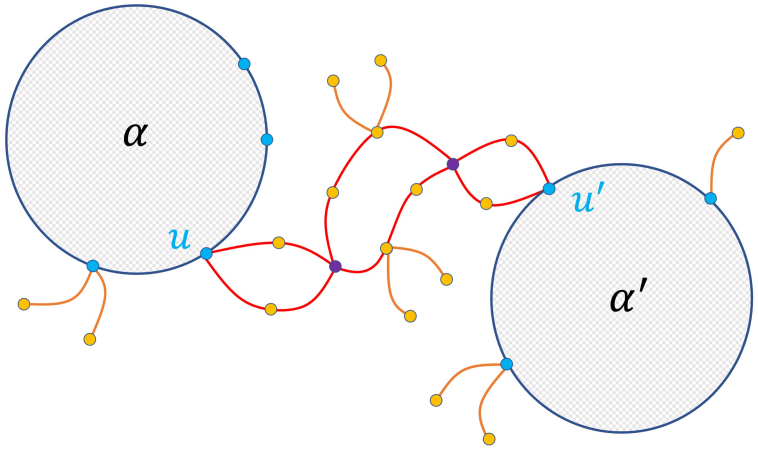
Proof of Lemma 5.1. We now complete the proof of Lemma 5.1 using the procedures previously described. We will prove the lemma by induction on h. The base case (when $h=1$) follows from Theorem 3.1. Consider now the case where the input (H,U) is an h-hole instance. We first compute a pair of terminals (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 inner 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, so Y is a vertex subset of



(a) Graph H: holes α, α' (shaded gray), terminals on α and α' (blue), path P (red), vertices of Y (purple), and other vertices (yellow).



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



(c) An illustration of gluing instances at boundary of holes. The output when the input is the $(h-1)$-hole instances in Figure 7(b). Holes α and α' are restored.

Figure 7. An illustration of splitting and gluing an h-hole instance along a path.

$V(P)$. Moreover, from Theorem 2.3, $|Y| \leq O(|U|/\varepsilon') \leq O(rh/\varepsilon)$. Set Y can be computed in $O(h \cdot n \log r)$ time using Lemma 2.4.

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 it returns. From the description of procedure $\textsc{Split}_h$, $|\tilde{U}| \leq |U| + 2|Y| \leq c \cdot rh/\varepsilon$, 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 to the proof of Claim 4.5, and is deferred to Appendix C.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 parameter $\varepsilon'' := \varepsilon(1 - \frac{1}{h})$, then there is another $(h-1)$-hole instance $(\tilde{H}', \tilde{U})$ that is an ε''-emulator for

$(\tilde{H}, \tilde{U})$, such that

$$
\begin{aligned}
|V(\tilde{H}')| &\le (ch)^{2(h-1)} \cdot |\tilde{U}| \cdot \frac{(\log|\tilde{U}|)^{c(h-1)}}{(\varepsilon'')^{c(h-1)}} \\
&\le (ch)^{2(h-1)} \cdot \frac{crh}{\varepsilon} \cdot \frac{\left(\log\left(\frac{crh}{\varepsilon}\right)\right)^{c(h-1)}}{\left(\varepsilon(1-\frac{1}{h})\right)^{c(h-1)}} \\
&\le (ch)^{2h-1} \cdot r \cdot \frac{1}{\varepsilon^{ch-c+1}} \cdot \frac{1}{(1-\frac{1}{h})^{h-1}} \cdot \left(\log\left(\frac{crh}{\varepsilon}\right)\right)^{c(h-1)} \\
&\le (ch)^{2h-1} \cdot r \cdot \frac{1}{\varepsilon^{ch}} \cdot c \cdot \left(\log r + \log\left(\frac{ch}{\varepsilon}\right)\right)^{c(h-1)} \\
&\le (ch)^{2h} \cdot r \cdot \frac{1}{\varepsilon^{ch}} \cdot \left(\log r\right)^{ch}.
\end{aligned}
$$

We now apply the procedure $\textsc{Glue}_h$ to instance $(\tilde{H}', \tilde{U})$, and let (H', U) be the h-hole instance we get. From the procedure $\textsc{Glue}_h$, $|V(H')| \le |V(H)| \le (ch)^{2h} \cdot r \cdot (\log r)^{ch}/\varepsilon^{ch}$. 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). This completes the proof of Lemma 5.1.

5.2 Algorithm for General Planar Graphs: Proof of Theorem 1.1

Separators and recursive decomposition. For any $r \ge 0$, an *r-division* [Fre87] of a planar graph G is a decomposition of G into edge-disjoint subgraphs of G, called the *pieces*, such that

- there are at most $O(|V(G)|/r)$ pieces,
- each piece has size at most r,
- the number of *boundary vertices* is at most $O(\sqrt{r})$ per piece, where a *boundary vertex* is a vertex that belongs to multiple pieces, and
- the number of holes in each piece is bounded by $O(1)$.

An r-division can be computed in linear time [KMS13]. In this paper, we compute the r-division on planar graphs with terminals, and we want the r-division to evenly distribute the terminals among the pieces.

Lemma 5.4 *Given a planar graph P with n vertices and t terminals together with a threshold $r \le n/t$, one can compute in $O(n)$-time an r-division for P such that each piece has $O(1)$ terminals.*

Proof: 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 $O(tr/n)$, which is $O(1)$ given that $t \le n/r$. □

Lemma 5.5 *For any planar instance (G,T) with $|G| = n$ and $|T| = k$ and any parameter $0 < \varepsilon < 1$, there is a planar ε-emulator for G with respect to T of size $\tilde{O}(k^{3/2}/\varepsilon^{O(1)})$, which can be constructed in $O\big((n+k^2)\log^{O(1)} n/\varepsilon^{O(1)}\big)$ time.*

Proof: First, without loss of generality we can replace G with an ε-emulator of size $N := O(k^2 \log^2 n/\varepsilon^2)$ using a slight modification of the construction of Cheung, Goranci, and Henzinger [CGH16, Theorem 6.9], by removing the processing step of reducing the graph size to $O(k^4)$. This has the benefit to reduce the construction time.

Apply r-division from Lemma 5.4 to G for $r := O(N/k) = \tilde{O}(k/\varepsilon^2)$ to even distribute the vertices, boundary vertices, holes, and terminals into the pieces. Each piece in the r-division now has $r = O(N/k)$ vertices, $\sqrt{r} = O(N^{1/2}/k^{1/2})$ boundary vertices, $O(1)$ number of holes, and $O(1)$ number of terminals. Now apply Lemma 5.1 on each piece P to obtain a planar emulator on the set of boundary vertices and terminals in P. Stitching the emulators for all the pieces together into a new graph G', which has size

$$O\left(\frac{N}{N/k} \cdot \frac{N^{1/2}\operatorname{poly}\log N}{\varepsilon^{O(1)}k^{1/2}}\right) = O\left(\frac{N^{1/2}\operatorname{poly}\log N \cdot k^{1/2}}{\varepsilon^{O(1)}}\right) = \tilde{O}\left(\frac{k^{3/2}}{\varepsilon^{O(1)}}\right).$$

Graph G' is in fact a planar emulator of G with respect to the terminals T.

A careful analysis to the construction of ε-emulator by Cheung-Goranci-Henzinger shows that it takes $O(n \log n/\varepsilon)$ time. Constructing the emulators for the pieces using Lemma 5.1 takes time $O((N/r) \cdot r(\log r/\varepsilon)^{O(1)}) = O(N \cdot (\log N/\varepsilon)^{O(1)})$. Therefore overall the running time is $O(n \log n/\varepsilon + k^2 \log^2 n \log^{O(1)} k/\varepsilon^{O(1)}) = O\big((n+k^2)\log^{O(1)} n/\varepsilon^{O(1)}\big)$. □

Lemma 5.6 *For any planar graph G with k terminals and any parameter $0 < \varepsilon < 1$, there is a planar $\alpha\varepsilon$-emulator for G of size $\tilde{O}(k^{1+2^{-\alpha}}/\varepsilon^{O(\alpha)})$ for any positive integer α, which can be constructed in $O\big((n+k^2)\log^{O(1)} n/\varepsilon^{O(1)}\big)$ time.*

Proof: We follow the proof of Lemma 5.5 almost verbatim, except for the initial choice of replacement of G, which we bootstrap from the construction for the case when α is one smaller. The size estimate becomes

$$\tilde{O}\left(\frac{(k^{1+2^{-(\alpha-1)}}/\varepsilon^{O(\alpha-1)})^{1/2} \cdot k^{1/2}}{\varepsilon^{O(1)}}\right) = \tilde{O}\left(\frac{k^{1+2^{-\alpha}}}{\varepsilon^{O(\alpha)}}\right).$$

The new running time is now α times $O\big((n+k^2)\log^{O(1)} n/\varepsilon^{O(1)}\big)$. □

6 Applications

We will prove in Section 6.1 that an ε-emulator for any $O(1)$-hole instance of size $O_\varepsilon(k \operatorname{polylog} k)$ can be computed in $O_\varepsilon(n \operatorname{poly}(\log^* n))$ time. Then in the rest of the section we present efficient ε-approximate algorithms to several optimization problems on planar graphs that beats their exact counterparts.

6.1 Bootstrapping

Theorem 6.1 *Given any parameter $0 < \varepsilon < 1$ and any planar graph P with n vertices and k terminals where $k \le \sqrt{n}$ all lying on a constant number of faces of P, one can compute an ε-emulator of P with respect to the terminals of size $O(k \operatorname{polylog} k/\varepsilon^{O(1)})$ in $O(n \log^{(c)} n/\varepsilon^{O(1)} + cn)$ time for any integer $c \ge 1$, where $\log^{(c)} n = \log(\ldots \log(\log n))$, where the logarithm is applied c times.*

Proof: To get an intuition of the proof, let's look at the case when $c = 2$, and ignore the restriction on the number of terminals and the ε-factors in the running time for a moment:

- First compute r-division of P for $r = (\operatorname{poly}\log\log n)^4$, where here the poly is equal to the number of logs we need in the running time of Lemma 5.1. Replace each piece in the r-division by an ε-emulator with respect to the boundary vertices and terminals using Lemma 5.1, which in total takes time $O(n \operatorname{poly}\log\log\log n)$ and the new graph P' has size $n/\varepsilon^{O(1)}(\operatorname{poly}\log\log n)^2$.
- Now because the graph is $\operatorname{poly}\log\log n$-factor smaller than original, we can compute another r'-division for $r' = (\operatorname{poly}\log n)^4$, and replace each piece in the r'-division by an ε-emulator with respect to the boundary vertices and terminals. This way, instead of spending $O(n \operatorname{poly}\log\log n)$ time if we perform r'-division directly on the original graph, now it takes

  $$O((n/\varepsilon^{O(1)}(\operatorname{poly}\log\log n)^2) \cdot \operatorname{poly}\log\log n) = O(n/(\varepsilon^{O(1)} \operatorname{poly}\log\log n))$$

  time which is sublinear. The new graph P'' has size $n/\varepsilon^{O(1)}(\operatorname{poly}\log n)^2$.
- Finally, compute an ε-emulator for P'' with respect to the terminals, which takes

  $$O((n/\varepsilon^{O(1)}(\operatorname{poly}\log n)^2) \cdot \operatorname{poly}\log n) = O(n/(\varepsilon^{O(1)} \operatorname{poly}\log n))$$

  time which is sublinear. The final emulator has size $O(k \operatorname{polylog} k/\varepsilon^{O(1)})$.
- Overall the bottleneck is the compute the first set of emulators in the r-division, which takes $O(n \operatorname{poly}\log\log\log n)$ time. The accumulated distortion in distance is 3ε.

Now we present the full proof. We will bootstrap on the following statement for ever bigger values of c, starting from 1 to C: Given a planar graph P with n vertices and k terminals on the boundary where $k \sim \sqrt{\varepsilon^{(C-c)\cdot O(1)} n (\log^{(c)} n)^{(C-c)\cdot O(1)}}$, one can compute an emulator of P with respect to the terminals in time $O(n \operatorname{poly}\log^{(c+1)} n/\varepsilon^{O(1)} + cn)$ time. (When $c = 0$, the statement follows immediately from Lemma 5.1.)

- First compute an r-division of P. Set $r = (\log^{(c)} n)^{O(1)}$. By Lemma 5.4 each piece has $\sqrt{r}$ boundary vertices. Here P has $t \sim \sqrt{\varepsilon^{(C-c)\cdot O(1)} n (\log^{(c)} n)^{(C-c)\cdot O(1)}}$ terminals when n is big enough. We have $\sqrt{r} + t \le n/r$.
- For each piece of the r-division, construct an e^{ε}-emulator of the boundary nodes and terminals using Lemma 5.1, which has size $O(\sqrt{r} \operatorname{poly}\log r/\varepsilon^{O(1)})$. This takes $O(n \operatorname{poly}\log r/\varepsilon^{O(1)})$ time. The new graph $\hat{P}$ now has size $\hat{n} = O(n \operatorname{poly}\log r/\varepsilon^{O(1)}\sqrt{r}) \sim n/\varepsilon^{O(1)}(\log^{(c)} n)^{O(1)}$ and is an e^{ε}-emulator for the boundary vertices of P. The number of terminals in $\hat{P}$ is at most

  $$\sqrt{\varepsilon^{(C-c)\cdot O(1)} n (\log^{(c)} n)^{(C-c)\cdot O(1)}}$$

$$= \sqrt{\varepsilon^{(C-c)\cdot O(1)}(n/\varepsilon^{O(1)}(\log^{(c)} n)^{O(1)}) \cdot \varepsilon^{O(1)}(\log^{(c)} n)^{O(1)} \cdot (\log^{(c)} n)^{(C-c)\cdot O(1)}}$$

$$\lesssim \sqrt{\varepsilon^{(C-c+1)\cdot O(1)}\hat{n}(\log^{(c)} \hat{n})^{(C-c+1)\cdot O(1)}}.$$

- Compute ε-emulator for $\hat{P}$ with respect to the terminals recursively from bootstrapping with parameter $c-1$, which takes time

$$O(\hat{n}\operatorname{poly}\log^{(c)} \hat{n} + (c-1)\hat{n}) = O\left((n/\varepsilon^{O(1)}(\log^{(c)} n)^{O(1)}) \cdot \operatorname{poly}\log^{(c)} n + (c-1)n\right) = O(cn).$$

Overall it takes $O(n\operatorname{poly}\log r/\varepsilon^{O(1)} + cn) = O(n\operatorname{poly}\log^{(c+1)} n/\varepsilon^{O(1)} + cn)$ time. The accumulated distortion in distance is $(c+1)\varepsilon$. □

By taking $c = \log^* n$, we have the following immediate corollary.

Corollary 6.2 *Given any parameter $0 < \varepsilon < 1$ and any planar graph P with n vertices and k terminals where $k \le \sqrt{n}$ on $O(1)$ many faces, one can compute an ε-emulator of P with respect to the terminals of size $O(k\operatorname{poly}\log k/\varepsilon^{O(1)})$ in time $O(n\operatorname{poly}\log^* n/\varepsilon^{O(1)})$ time.*

6.2 Approximate Multiple-Source Shortest Paths

Theorem 6.3 *Given any parameter $0 < \varepsilon < 1$ and any planar graph P with n vertices and $\sqrt{n}$ boundary vertices, one can preprocess an ε-MSSP data structure on P with respect to the boundary vertices in $O(n\log^{(c)} n + cn)$ time[8] for any integer $c \ge 1$, and answer queries in time $O(\log n)$.*

Proof: Apply Theorem 6.1 on P to construct an ε-emulator of P, which has size $O(\sqrt{n}\operatorname{poly}\log n/\varepsilon^{O(1)})$ and takes $O(n\log^{(c)} n/\varepsilon^{O(1)})$ time. Now construct the MSSP data structure on the emulator using Klein's algorithm [Kle05], which takes $O(\sqrt{n}\operatorname{poly}\log n/\varepsilon^{O(1)} \cdot \log n) = O(n)$ time; MSSP answers queries in time $O(\log n)$. □

By taking $c = \log^* n$, we have the following immediate corollary.

Corollary 6.4 *Given any parameter $0 < \varepsilon < 1$ and any planar graph P with n vertices, one can prepare an ε-MSSP data structure on P with respect to the terminals in time $O(n\operatorname{poly}\log^* n/\varepsilon^{O(1)})$.*

6.3 Approximate Minimum Cut

A high-level description of the minimum (s,t)-cut algorithm on planar graphs with non-negative weights is the following. Let G be the input planar graph, with two vertices s and t.

0. Find a shortest s^*-t^* path π of length p.
1. Construct r-division respecting π where $r = \log^6 n$. Cut π open, each point on π now has a duplicate.
2. Compute MSSP for each piece with respect to the boundary vertices. Prepare the Monge heap data structures, and represent each piece as a *dense distance graph*.
3. Compute $p/\log p$ shortest paths between the two copies, one for each evenly spaced points on π, using FR-Dijkstra on the dense distance graphs. Now the graph is cut into *slabs*.

[8] $\log^{(c)} n = \log(\ldots\log(\log n))$, where the logarithm is applied c times

4. Recurse on each slab which now has only $O(\log p)$ vertices from π, thus the recursion depth is $O(\log^* p)$.

Step 0 takes $O(n)$ time by Henzinger-Klein-Rao-Subramanian [HKRS97]. Step 1 takes $O(n)$ time [KMS13]. Step 2 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]; this step is the bottleneck. Step 3 takes $\tilde{O}(n/\sqrt{r} \cdot \log(k/\log k)) = O(n)$ time. Thus in total it takes $O(n \log\log n)$ time.

Now by simply replacing Step 2 with an ε-emulator per piece using Corollary 6.2, the new graph has size $O(n \log\log n/\varepsilon^{O(1)} \log^3 n)$. we can compute the required $p/\log p$ shortest paths using Reif's divide-and-conquer strategy which takes $O(n/\varepsilon^{O(1)} \operatorname{poly}\log n)$ time. Therefore improve the total running time to $O(n \operatorname{poly}\log^* n/\varepsilon^{O(1)})$ (due to construction time of the emulators), proving Theorem 1.3.

6.4 Approximate Diameter

A high-level description of computing $(1+\varepsilon)$-approximate diameter on planar graphs with non-negative weights is the following. Let G be the input planar graph.

0. Compute a *shortest-path* cycle separator C and splits the input planar graph into A and B, where $A \cap B = C$, using Thorup [Tho04].
1. Construct G^+ by selecting $O(1/\varepsilon)$ portals on C; run SSSP on each portal p to get maximum distance out of all paths from p, denoted as ℓ; and round the edge-lengths to multiples of $\varepsilon\ell$.
2. Approximate $\operatorname{diam}_{G^+}(A, B)$.
3. Build A^+ and B^+ by adding denser portals on C, shortcutting the other side and build a planar graph by planarizing the union of all the shortest paths between dense portal pairs.
4. Approximate $\operatorname{diam}_{A^+}(A, A)$ and $\operatorname{diam}_{B^+}(B, B)$ recursively.
5. Return max of $\operatorname{diam}_G(A, B)$, $\operatorname{diam}_{A^+}(A, A)$, and $\operatorname{diam}_{B^+}(B, B)$

Step 0 takes $O(n)$ time by Thorup [Tho04]. Step 1 takes $O(n \cdot (1/\varepsilon))$ time by Henzinger-Klein-Rao-Subramanian [HKRS97]. Step 2 takes $2^{O(1/\varepsilon)}$ time by brute-force [WY16] or $O(n \cdot (1/\varepsilon)^5)$ using farthest Voronoi diagram [CS19]. Step 3 chooses $|G|^{1/8}/\varepsilon$ dense portals [CS19]; compute APSP between dense portals in either A^+ or B^+, which takes $O(|B| \log n + \log n \cdot \sqrt{|B|}/\varepsilon^4)$ time; A^+ has size $|A| + O(|A|^{1/2}/\varepsilon^4)$. Thus in total it takes $O(n \log^2 n + n \log n (1/\varepsilon)^5)$ time.
Now we can substitute the planarized shortest paths in Step 3 with two ε-emulators using Corollary 6.2, which only takes $O((|A| + |B|) \log^* n)$ time to construct and has size $O((|A|^{1/8} + |B|^{1/8}) \operatorname{poly}\log n)$. Thus we improve the total running time to $O(n \log n \log^* n + n \log n \operatorname{poly}(1/\varepsilon))$, proving Theorem 1.4.

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.

[AGK14] A. Andoni, A. Gupta, and R. 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] A. Basu and A. 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.

[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 A.E. Ophelders. Planar emulators for Monge matrices. In *Proc. 32nd Canadian Conference on Computational Geometry (CCCG)*, pages 141–147, 2020.

[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.

[EFN12] Jeff. Erickson, Kyle. Fox, and Amir. Nayyeri. Global minimum cuts in surface embedded graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, Proceedings, pages 1309–1318. Society for Industrial and Applied Mathematics, January 2012. doi:10.1137/1.9781611973099.103.

[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 of *OpenAccess Series in Informatics (OASIcs)*, pages 10:1–10:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/OASIcs.SOSA.2019.10.

[FL20] Kyle Fox and Jiashuai Lu. A near-linear time approximation scheme for geometric transportation with arbitrary supplies and spread. page 18, 2020.

[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 (SODA)*, 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 multiterminal 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.

[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.

[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] R. Krauthgamer, H. Nguyen, and T. 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.

[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

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 an embedding from G and must be 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.

B Missing Proofs in Section 3

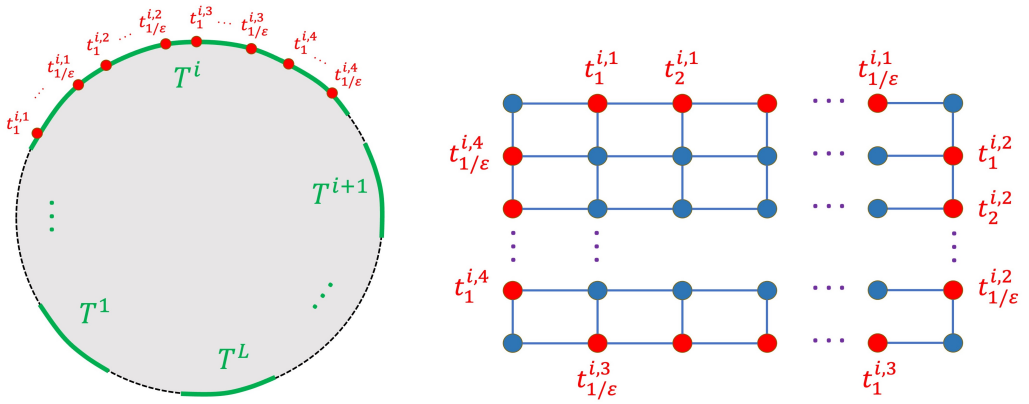
B.1 Proof of Theorem 3.2

In this subsection we provide the proof of Theorem 3.2. Our hard example is inspired by the hard example constructed in [KNZ14].

We first construct an circular ordering σ on the terminals in T and a metric d on the terminals. From [CO20], if d satisfies the Monge property (under the circular ordering σ), then there exists an 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 $T = T^1 \cup \cdots \cup T^L$, 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 $T^{i,j} = \{t_1^{i,j}, \ldots, t_{1/\varepsilon}^{i,j}\}$, for each $1 \leq j \leq 4$. The circular ordering σ on terminals of T is defined as follows. The groups $T^1, \ldots, T^L$ 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 $t_1^{i,j}, \ldots, t_{1/\varepsilon}^{i,j}$ appear clockwise in this order. See Figure 8(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 8(b). Now for each pair $t_r^{i,j}, t_{r'}^{i,j'}$ of terminals in T^i, we define $d(t_r^{i,j}, t_{r'}^{i,j'}) = \mathrm{dist}_H(t_r^{i,j}, t_{r'}^{i,j'})$. It is easy to verify that d is a metric and satisfies the Monge property.

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 $t, t' \in T$, $e^{-\varepsilon/3} \cdot \mathrm{dist}_{G'}(t, t') \leq d(t, t') \leq e^{-\varepsilon/3}$. For each $1 \leq i \leq L$, 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 $t_1^{i,1}$. Since in d, the distance between every pair of terminals in $\{t_1^{1,1}, \ldots, t_1^{L,1}\}$ is $1/\varepsilon^2$, it is easy to see that the graphs $\{G'_1, \ldots, G'_L\}$ are mutually vertex-disjoint. On the other hand, it is easy to verify that, for every $1 \leq i \leq L$ 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 $1 \leq i \leq L$, (G'_i, T^i) is an aligned $\varepsilon/3$-emulator for (G, T^i). From similar arguments in [KNZ14], we get that $|V(G'_i)| \geq \Omega(|T^i|^2) = \Omega(1/\varepsilon^2)$. Therefore, $|V(G')| \geq \sum_{1 \leq i \leq L} |V(G'_i)| \geq L \cdot \Omega(1/\varepsilon^2) = \Omega(k/\varepsilon)$. 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) An illustration of ordering σ. (b) An illustration of metric d within a group T^i of terminals.

Figure 8. Illustrations of circular ordering σ and metric d within a group T^i of terminals.

B.2 Proof of Claim 4.1

Denote $|U| = r$. Denote $\lambda = \log^2 r$. We will show that the number of branch vertices is at most $O(r\lambda)$. Let $\tilde{H}$ be the graph obtained from taking 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 $O(r\lambda)$, it suffices to show that the number of vertices in $\tilde{H}'$ is $O(r\lambda)$, and therefore it suffices to show that the number of face in the planar drawing of $\tilde{H}'$ is $O(r\lambda)$.

Assume without loss of generality that the set vertices that lie on the outer-boundary of $\tilde{H}'$ is U', so $|U'| \leq |U|$. Denote $r' = |U'|$. We view terminals of U' as points lying on the boundary of a disc D (which we denote by ∂D). Note that every path in $\mathcal{P}$ naturally corresponds to a path in graph $\tilde{H}'$, that we denote by P'. We define $\mathcal{P}' = \{P' \mid P \in \mathcal{P}\}$.We view the images of paths in $\mathcal{P}'$ as a set Γ of simple curves in D connecting pairs of points on the boundary of D.

Let Σ be a set of internally disjoint segments of the boundary of D, that we call *active segments*, and we call a point on the boundary of D *inactive* if and only if it does not lie in any of the active segments. We say that a simple curve γ is *well-placed* with respect to the active segments, if and only if (i) both endpoints of γ are either inactive points or endpoints of active segments; (ii) γ does not internally contain any inactive point; and (iii) the intersection between γ and any active segment $\sigma \in \Sigma$ is a contiguous subcurve of both γ and σ. See Figure 9 for an illustration.

To bound the number of branch vertices, we consider the following process. We think of the paths of Γ as coming one-by-one to be added in D. We let D be a disc where all their endpoints lie on. Initially there is no active segment on the boundary of D. When the first curve γ_1 comes, it partitions the disc D into two discs D_1 and D_2, and we say that γ_1 serves as an active segment for both D_1 and D_2. Consider now some curve γ_i comes, and it lies in some disc $D' \subseteq D$, and all active segments of on ∂D are segments of curves in Γ that come before γ_i. Since the curves in Γ are well-structured, the curve γ_i must be well-placed with respect to the boundary segments of the disc that it lies in. Then it further partitions disc D' into smaller ones, and certain segments of γ_i serve as active segments as those newly-created discs.

For every pair $x, y > 0$ of integers, define $f(x, y)$ to be the maximum number of faces that

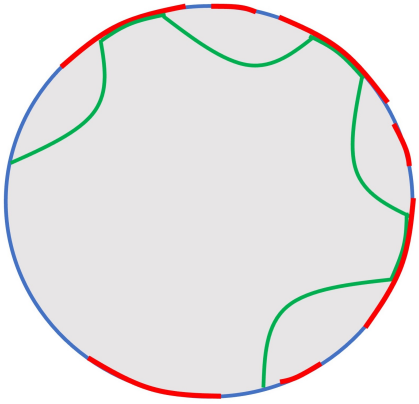


Figure 9. A curve (green) is well-placed with respect to active segments (red) on the boundary (blue) of disc D.

can be separated by a collection of x well-structured curves well-placed in a disc that initially contains y active segments. It suffices to show that $f(r',0) \leq O(r'\lambda)$. Clearly, for all integers x, y, $f(x,y) \leq 2(x+y)^2$ (since every pair of curves and segments create at most two branch vertices); and for all $x \leq x'$ and y, $f(x,y) \leq f(x',y)$.

Assume now that we have a disc D that contains y boundary active segments. Assume that one curve γ out of x curves to be placed in disc D comes. Assume that curve γ intersects with segments $\sigma_1, \ldots, \sigma_t$, where the intersection between γ and each σ_i is a subcurve σ_i' of σ_i. Consider now the small discs separated by γ. If we denote, for each $1 \leq i \leq t$ by $a_i - 1$ the number of active segments that lie clockwise between σ_i and σ_{i+1} (where we set $t+1=1$), then we get that $\sum_{1\leq i\leq t} a_i = y+1$, and the small discs are characterize by pairs in the collection $\{(x_i, a_i+2)\}_i$ (since γ contributes one active segment to each of the small discs), where $\sum_{1\leq i\leq t} a_i = y+1$ and $\sum_{1\leq i\leq t} x_i = x-1$. Let $\{x_i, a_i\}_{1\leq i\leq t}$ be the collection of integers that, among all collections of integers that satisfy constraints $\sum_{1\leq i\leq t} a_i = y+1$ and $\sum_{1\leq i\leq t} x_i = x-1$, maximizes the value of $\sum_{1\leq i\leq t} f(x_i, a_i+2)$.

We now bound $f(r',0)$ from above via a charging scheme as follows. Throughout the process we maintain a collection $\mathcal{M}$ of pairs of integers, that initially contains a single pair $(r',0)$. In each iteration, a pair (x,y) from the current collection $\mathcal{M}$ is replaced by the collection $\{(x_i, a_i+2)\}$ of pairs, with the guarantee that the value of $\sum_{(x',y')\in\mathcal{M}} f(x',y')$ stays the same all the time. The process continues to be executed until all pairs (x,y) in the collection $\mathcal{M}$ satisfies that $x+y \leq \lambda$. Consider now an iteration in which the pair (x,y) is replaced by the collection $\{(x_i, a_i+2)\}$ of pairs, For each i such that $x_i + a_i > (x+y)/10$, we charge 2 (the additional two active segments in this pair) to the curve γ. We also charge the additional $1 = \left(\sum a_i\right) - y$ to curve γ, and we call this *type-1* charge. Clearly, γ is type-1 charged at most 21 in this iteration, and after that it becomes segments and will never be type-1 charged again. Consider now a pair (x_i, a_i+2) with $\lambda - 2 < x_i + a_i \leq (x+y)/10$. We charge 2 uniform to all x_i curves and a_i old segments in this disc, and call it *type-2* charge, so every curve and segment is type-3 charged at most $2/(\lambda-2) \geq 3/\lambda$. Therefore, every time a curve of a segment is type-2 charged, the number of curves and segments in the disc that it lies in decreases by at least a factor of 10. Therefore, every curve is type-2 charged at most $\log_{10} r \cdot (3/\lambda) \leq 1/\log r$ in total. This only counts the direct charge. The total direct and indirect charge can be bounded as at most $21 + (1/\log r) + (1/\log r)^2 + \cdots \leq 22$. Therefore, if we denote by $\mathcal{M}^*$ the collection we obtain

when the process terminates, $\sum_{(x,y)\in\mathcal{M}^*} x+y \le 22r'$. Therefore,

$$f(r',0) = \sum_{(x,y)\in\mathcal{M}^*} f(x,y) \le \sum_{(x,y)\in\mathcal{M}^*} (x+y)\cdot\lambda \le 22r'\lambda \le 22r\lambda.$$

This completes the proof of Claim 4.1.

B.3 Proof of Claim 4.2

From Claim 4.1, the number of branch vertices (vertices that appear on the boundary of at least three regions in $\mathcal{R}$) is at most $\tilde{O}(|U|\cdot\log^2|U|)$. Since every vertex in Y either is a branch vertex or belongs to exactly two regions in $\mathcal{R}$, we get that $\sum_{(H_R,U_R)\in\mathcal{H}}|U_R| \le \tilde{O}(|U|\log^2|U|+|Y|)$. From now on we focus on the proof that $\sum_{(H_R,U_R)\in\mathcal{H}:|U_R|\ge\mu}|U_R| \le \big(|U|+O(|Y\setminus U|)\big)\cdot(1+\frac{1}{\mu-2})$.

We pre-process the set $\mathcal{P}$ of paths as follows. If some path $P\in\mathcal{P}$ contains terminals in U as inner vertices, then we cut the path at these terminals into smaller paths. Specifically, consider some path $P\in\mathcal{P}$ and denote $V(P)\cap U=\{u_0,\ldots,u_L\}$, where the terminals are indexed according to the order in which they appear on the path. Then for each $0\le i\le L-1$, we define path P_i to be the subpath of P between vertices u_i to u_{i+1}. We then replace the path P in $\mathcal{P}$ by the new paths $P_0,\ldots,P_{L-1}$. Let $\tilde{\mathcal{P}}$ be the resulting set of paths. Since $((H,U),\mathcal{P},Y)$ is a valid input to procedure Split, it is easy verify that $\tilde{\mathcal{P}}$ is also a well-structured non-crossing set of shortest paths connecting pairs of terminals in U in H, and so $((H,U),\tilde{\mathcal{P}},Y)$ is also a valid input to procedure Split, and the output of procedure Split on the new input $((H,U),\tilde{\mathcal{P}},Y)$ is also the collection $\mathcal{H}$.

Recall that the graph H is drawn inside disc D with all terminals in U lying on the boundary of D, and the procedure Split computes a partitioning of D into a collection $\mathcal{R}$ of regions before returning the final collection of instances. We now prove by on $|\tilde{\mathcal{P}}|$ that $\sum_{(H_R,U_R)\in\mathcal{H}:|U_R|\ge\mu}|U_R| \le \big(|U|+2\cdot(|Y\setminus U|)\big)\cdot(1+\frac{1}{\mu-2})$. The base case is when $|\tilde{\mathcal{P}}|=1$. In this case $\mathcal{R}$ contains two regions and all vertices of Y belong to both regions. Therefore,

$$\sum_{(H_R,U_R)\in\mathcal{H}:|U_R|\ge\mu}|U_R| \le \sum_{(H_R,U_R)\in\mathcal{H}}|U_R| \le \Big(|U|+2+2|Y\setminus U|\Big) \le \Big(|U|+2|Y\setminus U|\Big)\cdot\left(1+\frac{1}{\mu-2}\right).$$

Assume that the claim holds for all valid inputs $((H,U),\tilde{\mathcal{P}},Y)$ with $|\tilde{\mathcal{P}}|\le p-1$. Consider the case where $|\tilde{\mathcal{P}}|=p$. Since paths in $\tilde{\mathcal{P}}$ no longer contain terminals as inner vertices, it is easy to see that there exist a path $P\in\tilde{\mathcal{P}}$, such that one of the two regions of D separated by P belongs to the collection $\mathcal{R}$. Denote $U=\{u_1,\ldots,u_r\}$, where the terminals are indexed according to the order in which they appear clockwise on the boundary of D. Denote by u_i,u_j the endpoints of P. Denote by γ_P the image of P. Denote by ζ_P the boundary segment of D from u_i clockwise to u_j (that contains the images of $u_{i+1},\ldots,u_{j-1}$), denote by ζ'_P the boundary segment of D from u_i anti-clockwise to u_j, and assume without loss of generality that the region surrounded by γ_P and ζ_P, denoted by R_P belongs to $\mathcal{R}$. Denote by Y_P the set of all vertices of $Y\setminus U$ that lie on P.

Consider now a new instance (H',U') defined as follows. Graph H' is obtained from H by deleting all vertices and edges drawn in the interior of region R_P and deleting all terminals $u_{i+1},\ldots,u_{j-1}$ and their incident edges. Set $U'=(U\setminus\{u_{i+1},\ldots,u_{j-1}\})\cup Y_P$. It is easy to verify that (H',U') is a one-hole instance, as the associated drawing of H naturally induces a drawing of H' (inside $D\setminus R_P$), with vertices of U' lying on the boundary of $D\setminus R_P$. Define $\tilde{\mathcal{P}}'=\tilde{\mathcal{P}}\setminus\{P\}$, so $|\tilde{\mathcal{P}}'|=q-1$, and it is easy to verify that the set $\tilde{\mathcal{P}}'$ of paths is well-structured and non-crossing in (H',U'). Finally, define $Y'=(Y\setminus Y_P)\cup\{u_i,u_j\}$, so Y is a set of vertices in $V(\tilde{\mathcal{P}}')$. Therefore,

$((H',U'),\tilde{\mathcal{P}}',Y')$ is a valid input to SPLIT, with $|\tilde{\mathcal{P}}'|=p-1$. And moreover, it is easy to see that the output of procedure SPLIT on $((H',U'),\tilde{\mathcal{P}}',Y')$ is exactly $\mathcal{H}\setminus\{(H_{R_P},U_{R_P})\}$.

Note that $|U'|=|U|-(j-i-1)+|Y_P|$ and $|Y'\setminus U'|=|Y\setminus U|-|Y_P|$. From the induction hypothesis,

$$\begin{aligned}\sum_{(H_R,U_R)\in\mathcal{H}:R\neq R_P,|U_R|\geq\mu}|U_R| &\leq \Big(|U'|+2|Y'\setminus U'|\Big)\cdot\left(1+\frac{1}{\mu-2}\right)\\ &\leq \Big(|U|-(j-i-1)+|Y_P|+2|Y\setminus U|-2r'\Big)\cdot\left(1+\frac{1}{\mu-2}\right)\\ &\leq \Big(|U|+2|Y\setminus U|\Big)\cdot\left(1+\frac{1}{\mu-2}\right)-\Big((j-i-1)+|Y_P|\Big)\left(1+\frac{1}{\mu-2}\right).\end{aligned}$$

It remains to consider the instance (H_{R_P},U_{R_P}). Note that $U_{R_P}=\{u_i,\ldots,u_j\}\cup Y_P$. If $|U_{R_P}|\geq\mu$, then $j+1-i+|Y_P|\geq\mu$, so $\big((j+1-i+|Y_P|)-2\big)\cdot\big(1+\frac{1}{\mu-2}\big)\geq(j+1-i+|Y_P|)$. Combined with the above inequality, we get that $\sum_{(H_R,U_R)\in\mathcal{H}:|U_R|\geq\mu}|U_R|\leq\big(|U|+2\cdot(|Y\setminus U|)\big)\cdot(1+\frac{1}{\mu-2})$. If $|U_{R_P}|<\mu$, then the above inequality directly implies that $\sum_{(H_R,U_R)\in\mathcal{H}:|U_R|\geq\mu}|U_R|\leq\big(|U|+2\cdot(|Y\setminus U|)\big)\cdot(1+\frac{1}{\mu-2})$.

B.4 Proof of Claim 4.3

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

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 $\{u_1,\ldots,u_k\}$ 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 $0\leq i\leq k$, either one of u_i,u_{i+1} is a branch vertex and so $\mathrm{dist}_Z(u_i,u_{i+1})=\mathrm{dist}_{\hat{H}}(u_i,u_{i+1})$, or u_i,u_{i+1} are both vertices of Y and belong to the same instance in $\mathcal{H}$ and so $\mathrm{dist}_{\hat{H}}(u_i,u_{i+1})\geq e^{-\varepsilon}\cdot\mathrm{dist}_Z(u_i,u_{i+1})$. Thus, if we set, for each $0\leq i\leq k$, H_{R_i} to be the graph in $\mathcal{H}$ that vertices u_i,u_{i+1} belong to, then

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

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 $\{u'_1,\ldots,u'_k\}$ 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 $0\leq i\leq k$, either one of u'_i,u'_{i+1} is a branch vertex and so $\mathrm{dist}_Z(u'_i,u'_{i+1})=\mathrm{dist}_{\hat{H}}(u'_i,u'_{i+1})$, or u'_i,u'_{i+1} are both vertices of Y and belong to the same instance in $\mathcal{H}$ and so $\mathrm{dist}_Z(u'_i,u'_{i+1})\geq e^{-\varepsilon}\cdot\mathrm{dist}_{\hat{H}}(u'_i,u'_{i+1})$. Thus, if we set, for each $0\leq i\leq k$, H_{R_i} to be the graph in $\mathcal{H}$ that vertices u'_i,u'_{i+1} belong to, then

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

B.5 Calculations for size and error bounds

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

Proof (of Observation B.1): Since for each $1 \leq i \leq t-1$, $r_{i+1} \leq (9/10) \cdot r_i$, we get that $\log_{(10/9)} r_{i+1} \leq \log_{(10/9)} r_i - 1$. Therefore, it is easy to observe that

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

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

Proof (of Observation B.2): Consider any index $1 \leq i \leq t-1$. 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 $r_i \geq (10/9) \cdot r_{i+1}$, we get that

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

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

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

and so

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

C Missing Proofs in Section 5

C.1 Complete Description of Procedures Split$_h$ and Glue$_h$

Splitting. First we describe an algorithm called Split$_h$, whose input consists of

- an h-hole instance (with $h > 1$);
- a path connecting a pair of its terminals that lie on different holes; and
- a set of vertices in the path.

The output of the algorithm Split$_h$ is an $(h-1)$-hole instance.

Let (H, U) be the given input h-hole instance. Denote P a simple path connecting a pair of terminals u to u' in H, and let $Y \subseteq V(P)$ be a set of vertices in P. We denote by γ the curve representing the image of path P in H, and view it as a curve that is being directed from u to u'. For each $v \in V(P)$, we define $\delta_1(v)$ ($\delta_2(v)$, resp.) to be the set that contains all incident edges of v in graph H, whose image's first segment lies on the left (right, resp.) side of curve γ, as we traverse along γ from u to u'. We now modify the graph H as follows. We 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 $\delta_2(v)$. Lastly, 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 $\tilde{H}$. Graph $\tilde{H}$ naturally inherits an embedding from H. See Figure 7(a) and Figure 7(b) for an illustration.

We now define $\tilde{U}$ to be the set obtained from U by replacing u with u_1 and u_2, replacing u' with u'_1 and u'_2, and adding, 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. Consider now $\tilde{H}$. 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 $(U(\alpha) \setminus \{u\}) \cup (U(\alpha') \setminus \{u'\}) \cup \{u_1, u_2, u'_1, u'_2\} \cup \{y_1, y_2 \mid y \in Y\}$. Therefore $(\tilde{H}, \tilde{U})$ is a valid $(h-1)$-hole instance.

Gluing. We next describe an procedure called $\textsc{Glue}_h$, 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 $Y \subseteq V(P)$ 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 denote by u_1, u_2 (u'_1, u'_2, resp.) the terminals in $\tilde{U}$ obtained by splitting the terminal u (u', resp.). Denote $Y = \{y_1, \ldots, y_L\}$, where the vertices are indexed according to the order in which they appear on path P, with y_1 being closest to u and y_L being closest to u'. We then denote, for each $1 \leq i \leq L$, by y_i^1 and y_i^2 the two terminals in $\tilde{U}$ obtained by splitting y_i. 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 u_1 and u_2 (and name the new vertex u), identifying u'_1 and u'_2 (and name the new vertex u'), and identifying, for each $1 \leq i \leq L$, vertex y_i^1 with vertex y_i^2 (and name the new vertex y_i). Denote $\tilde{Y} = \{y_i^1, y_i^2 \mid 1 \leq i \leq L\}$. 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). Graph H' naturally inherits a planar embedding from $\tilde{H}'$, and (H', U) must be an h-hole instance. Moreover, it is easy to verify that instance (H', U) is aligned with instance (H, U). See Figure 7(c) for an illustration.

C.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 $e^{-\varepsilon'} \cdot \mathrm{dist}_Z(u, u') \leq \mathrm{dist}_{\hat{H}}(u, u') \leq e^{\varepsilon'} \cdot \mathrm{dist}_Z(u, u')$.

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 $E(Q) \cap E_1 = \emptyset$ or $E(Q) \cap E_2 = \emptyset$ 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

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

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. 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

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

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

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

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 $y^1\ldots,y^r$ be the vertices of $Y\cap V(Q)$, where the vertices are indexed according to their appearance on Q. For each $0\leq j\leq r$, 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 $0\leq j\leq r$, 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

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

Altogether, we get that $e^{-\varepsilon'}\cdot\mathrm{dist}_Z(u,u')\leq\mathrm{dist}_{\hat{H}}(u,u')\leq e^{\varepsilon'}\cdot\mathrm{dist}_Z(u,u')$.

C.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 $e^{-\varepsilon'}\cdot\mathrm{dist}_{\hat{H}}(u,u')\leq\mathrm{dist}_H(u,u')\leq\mathrm{dist}_{\hat{H}}(u,u')$.

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 $\mathrm{dist}_{\hat{H}}(u,u')\leq\mathrm{dist}_H(u,u')$. 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 $\mathrm{dist}_H(u,y)+\mathrm{dist}_H(y,v)\leq e^{\varepsilon}\cdot\mathrm{dist}_H(u,v)$; and similarly since set Y contains an ε'-cover of u' on Q, there exists some vertex $y'\in Y$, such that $\mathrm{dist}_H(u',y')+\mathrm{dist}_H(y',v')\leq e^{\varepsilon}\cdot\mathrm{dist}_H(u',v')$.

From the construction of graph $\hat{H}$, we get that

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

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 $\mathrm{dist}_H(u,u')\leq\mathrm{dist}_{\hat{H}}(u,u')$. 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

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