Embedding Planar Graphs into Graphs of Treewidth $O(\log^3 n)$*

Hsien-Chih Chang† Vincent Cohen-Addad‡ Jonathan Conroy§ Hung Le¶
Marcin Pilipczuk‖ Michał Pilipczuk**

Abstract

Cohen-Addad, Le, Pilipczuk, and Pilipczuk [CLPP23] recently constructed a stochastic embedding with expected $1+\varepsilon$ distortion of n-vertex planar graphs (with polynomial aspect ratio) into graphs of treewidth $O(\varepsilon^{-1}\log^{13} n)$. Their embedding is the first to achieve polylogarithmic treewidth. However, there remains a large gap between the treewidth of their embedding and the treewidth lower bound of $\Omega(\log n)$ shown by Carroll and Goel [CG04]. In this work, we substantially narrow the gap by constructing a stochastic embedding with treewidth $O(\varepsilon^{-1}\log^3 n)$.

We obtain our embedding by improving various steps in the CLPP construction. First, we streamline their embedding construction by showing that one can construct a low-treewidth embedding for any graph from (i) a stochastic hierarchy of clusters and (ii) a stochastic balanced cut. We shave off some logarithmic factors in this step by using a single hierarchy of clusters. Next, we construct a stochastic hierarchy of clusters with optimal separating probability and hop bound based on shortcut partition [CCL+23, CCL+24]. Finally, we construct a stochastic balanced cut with an improved trade-off between the cut size and the number of cuts. This is done by a new analysis of the contraction sequence introduced by [CLPP23]; our analysis gives an optimal treewidth bound for graphs admitting a contraction sequence.

*Ma.P. is supported by Polish National Science Centre SONATA BIS-12 grant number 2022/46/E/ST6/00143. This work is a part of project BOBR (Mi. P.) that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 948057). Hung Le was supported by the NSF CAREER Award No. CCF-223728, an NSF Grant No. CCF-2121952, and a Google Research Scholar Award.

†Dartmouth College. Email: hsien-chih.chang@dartmouth.edu.
‡Google Research, France. Email: cohenaddad@google.com.
§Dartmouth College. Email: jonathan.conroy.gr@dartmouth.edu.
¶University of Massachusetts Amherst. Email: hungle@cs.umass.edu.
‖Institute of Informatics, University of Warsaw, Poland. Email: m.pilipczuk@uw.edu.pl.
**Institute of Informatics, University of Warsaw, Poland. Email: michal.pilipczuk@mimuw.edu.pl.

1 Introduction

A general algorithmic technique to design approximation algorithms on graphs is *metric embedding*: given an edge-weighted graph G and a problem Π, embed G into a structurally simpler edge-weighted graph H with a small *distortion* and solve Π on H to get an approximate solution for Π in G. The distortion often decides the approximation ratio, while the running time to solve Π depends on the simplicity of the structure of H. An important line of work initiated by Bartal [Bar96, Bar98] culminated in the tree embedding theorem by Fakcharoenphol, Rao, and Talwar [FRT04], who showed that any n-vertex graph can be stochastically embedded into a tree with distortion $O(\log n)$. This result gives polynomial-time $O(\log n)$-approximation algorithms for a host of optimization problems in graphs, for example, buy-at-bulk network design, metric labeling, minimum cost communication network; see Section 4 in [FRT04] for a more comprehensive list of applications.

However, the $O(\log n)$ distortion for embedding into trees is tight [Bar96] even for planar graphs [CG04, CJLV08]; therefore, techniques using embedding into trees cannot give better than $O(\log n)$ approximation ratio. To achieve a better distortion, one has to not only restrict the structure of the input graph G but also extend the class of host graphs H to be broader than trees. Here, we consider a well-studied and natural setting: G is planar or minor-free, and H has a small *treewidth*. Until recently, most results in this direction were negative: (1) the expected distortion remains $\Omega(\log n)$ even if H has *bounded treewidth* [CJLV08]; (2) if the distortion is a constant $c \geq 1$, then the treewidth must be $\Omega(c^{-1} \cdot \log n)$ [CG04]. If one looks for a deterministic embedding, the treewidth of H must be $\Omega(\sqrt{n})$ for a constant distortion [CG04]. However, as observed by Cohen-Addad, Le, Pilipczuk, and Pilipczuk [CLPP23], these lower bounds do not rule out the following conjecture:

Conjecture 1.1. *For every $\varepsilon \in (0,1)$, any edge-weighted n-vertex planar graph can be embedded with expected distortion $1+\varepsilon$ into a distribution of graphs with treewidth $O(\varepsilon^{-1} \cdot \log n)$.*

Conjecture 1.1, if true, will imply, among other things, the first PTAS (polynomial-time approximation scheme) for the k-MST (k-minimum spanning tree) problem in planar graphs: find a spanning tree of minimum weight that contains at least k vertices a graph. After a long line of works [RSM+96, AABV95, BRV99, Gar96, AR98, AK06], Garg [Gar05] showed that this problem admits a 2-approximation in polynomial time in general graphs. In planar graphs, it has been a long-standing open problem to design a PTAS for k-MST; only a QPTAS (quasi-polynomial time approximation scheme) was known by a very recent work of Cohen-Addad [Coh22].

Cohen-Addad, Le, Pilipczuk, and Pilipczuk [CLPP23] gave strong evidence supporting Conjecture 1.1: they constructed the first embedding with polylogarithmic treewidth. The precise treewidth bound is

$$O(\varepsilon^{-1} \cdot \log^6(n\Phi) \cdot \log^5(n\Phi/\varepsilon) \cdot \log^2 n)$$

where $\Phi := \frac{\max_{u,v} d_G(u,v)}{\min_{u \neq v} d_G(u,v)}$ is the *aspect ratio*. Specifically, for graphs with polynomial aspect ratios $\Phi = n^{O(1)}$, the treewidth they obtained is roughly $O(\varepsilon^{-1} \log^{13} n)$. (In many interesting applications, including those considered in [CLPP23], one can assume that the aspect ratio is polynomial.) Clearly there is a big gap between the treewidth upper bound achieved by [CLPP23] and the treewidth bound in Conjecture 1.1. In this work, we substantially narrow the gap, and specifically, for the case of polynomial aspect ratio, we achieve treewidth bound $O(\varepsilon^{-1} \log^3 n)$. Our result extends to any apex-minor-free graphs, which include planar graphs as a subclass.

Theorem 1.1. *Let G be an n-vertex apex-minor-free graph with aspect ratio Φ. For any parameter $\varepsilon \in (0,1)$, G can be stochastically embedded with expected distortion $1+\varepsilon$ into a distribution of graphs of treewidth $O(\varepsilon^{-1} \cdot \log^2 \Phi \cdot \log(n\Phi))$.*

Our Theorem 1.1 immediately leads to faster QPTAS for several problems on planar graphs, including capacitated vehicle routing, capacitated k-median, and capacitated/uncapacitated facility location.

A brief overview of the CLPP embedding. Our construction builds on ideas of [CLPP23], which we briefly review here. For simplicity of exposition, we assume that the aspect ratio $\Phi = n^{O(1)}$ and hence $\log \Phi = O(\log n)$. The key idea in [CLPP23] is the construction of a (τ, ψ)-stochastic balanced cut $\mathfrak{F}$. Roughly speaking, a cut $\mathcal{F}$ in $\mathfrak{F}$ is a (small) set of at most τ clusters taken from a (stochastic) hierarchy $\mathbb{C}$ of clusters with exponentially decreasing diameter, called a *clustering chain*. (This notion of cut is non-standard.) Intuitively a cut $\mathcal{F}$ is *balanced* if $G \setminus \mathcal{F}$, the graph obtained by removing from G all the vertices in the clusters in $\mathcal{F}$, has connected components of size at most $2n/3$. Loosely speaking, a (τ, ψ)-stochastic balanced cut $\mathfrak{F}$ is a distribution of balanced

cuts of size at most τ such that every cluster in $\mathbb{C}$ is sampled with at most $1/\psi$ probability. [CLPP23] showed that for apex-minor-free graphs (and hence for planar graphs), one can construct a (τ, ψ)-stochastic balanced cut with $\tau = \psi \cdot \log^6 n$. (Parameter τ will dictate the treewidth of the embedding, while ψ will dictate the distortion.)

The embedding algorithm is then rather simple: sample a cut $\mathcal{F}$ from $\mathfrak{F}$, recurse on each cluster $C \in \mathcal{F}$ and each connected component of $G \setminus \mathcal{F}$, and then form the embedding of G from the embeddings in the recursions by (i) taking a single representative vertex per cluster in $\mathcal{F}$ and (ii) connecting each representative to every other vertex of the graph.

The depth of the recursion is shown to be $L = O(\log^2 n)$ by the fact that clusters in $\mathcal{F}$ have a diameter at most half of the diameter of G, and the cut is balanced. (A subtle point is that connected components of G might have a diameter bigger than G, and hence, the recursion depth is much larger than $\log n$.) Every time a pair (u, v) is cut by a cluster C of at level i of $\mathbb{C}$ (of diameter 2^i) in the balanced cut $\mathcal{F}$, the additive distortion is $O(2^i)$ since their shortest path $\pi_G(u, v)$ is now rerouted (by the embedding) through the representative of C. (A pair (u, v) is cut by a cluster C if $\pi_G(u, v)$ intersects C.) By constructing $\mathbb{C}$ carefully, in expectation, one could show that the number of clusters at level i intersecting $\pi(u, v)$ is about $O(\log^2 n) \cdot \delta_G(u, v)/2^i$, and hence the *expected additive distortion at one level of the recursion* is:

$$O(2^i) \cdot \frac{1}{\psi} \cdot O(\log^2 n) \cdot \frac{\delta_G(u, v)}{2^i} = \frac{O(\log^2 n) \cdot \delta_G(u, v)}{\psi} , \tag{1.1}$$

where $\frac{1}{\psi} \cdot O(\log^2 n) \cdot \frac{\delta_G(u,v)}{2^i}$ is the probability that a cluster intersecting $\pi(u, v)$ is chosen.

Over L levels of the recursion, the expected additive distortion is roughly $L \cdot O(\log^2 n) \cdot \delta_G(u, v)/\psi$. A technical subtlety is that a single edge (x, y) could be cut by up to $O(\log n)$ clusters in different levels of $\mathbb{C}$ since the depth of $\mathbb{C}$ is $O(\log n)$; this introduces another $O(\log n)$ factor in the distortion. (Indeed, in this paper we suffer from the same extra log factor which is inherent in the framework.) Therefore, the final expected additive distortion is:

$$\frac{L \cdot O(\log^3 n) \cdot \delta_G(u, v)}{\psi} = \frac{O(\log^5 n) \cdot \delta_G(u, v)}{\psi} \tag{1.2}$$

To get an expected additive distortion of $\varepsilon \cdot \delta_G(u, v)$ so to imply an expected *multiplicative* distortion $1 + \varepsilon$, one sets $\psi := \frac{\log^5(n)}{\varepsilon}$, giving $\tau = \log^6(n) \cdot \psi = \varepsilon^{-1} \log^{11} n$. Since every sampled cut has size at most τ, and the recursion depth is L, the final treewidth of the embedding is $O(\tau \cdot L) = O(\varepsilon^{-1} \log^{13} n)$.

One could probably improve one log factor in the analysis of the algorithm by [CLPP23] by showing that the number of clusters at level i of $\mathbb{C}$ intersecting $\pi(u, v)$ is instead $O(\log n) \cdot \frac{\delta_G(u,v)}{2^i}$, which is tight for their algorithm. By redoing the analysis, one could see that this log factor improvement shaves two log factors in the relationship between ψ and τ, giving $\tau = \psi \cdot \log^4 n$. All things together imply an embedding with treewidth $O(\varepsilon^{-1} \log^{10} n)$. Significantly improving over the $O(\log^{10} n)$ treewidth bound seems to require new ideas for every step in the framework.

Our ideas. We propose three new ideas on every component of the CLPP embedding. First, we streamline the framework, showing that one can construct a low-treewidth embedding from two objects: (i) a (stochastic) clustering chain $\mathbb{C}$ with *β-separating property*: the probability of cutting an edge e at level i of $\mathbb{C}$ is $\beta \cdot \|e\|/2^i$ where 2^i is the diameter of clusters at level i, and (ii) a (τ, ψ)-stochastic balanced cut. In this streamlined framework the treewidth of the embedding is $O(\tau + \log n)$ while the expected additive distortion is $O(\beta \log^2 n/\psi) \cdot \delta_G(u, v)$, compared to the original treewidth bound $O(\tau \log^2 n)$ and additive distortion $O(\log^5 n/\psi) \cdot \delta_G(u, v)$. (See Theorem 2.3; notice this theorem holds even for general graphs.) Our key idea here is to use a single clustering chain $\mathbb{C}$ throughout the recursion instead of recomputing the clustering chain for every connected component of $G \setminus \mathcal{F}$ as done by [CLPP23]. At every step of the recursion, the subgraph on which we recurse is a cluster of $\mathbb{C}$, and hence, we are guaranteed to make progress in *every level of the recursion*: either the diameter reduced by a constant factor or the number of important vertices (those that the recursion has to handled) reduced by a constant factor while the diameter remains unchanged. Therefore, the recursion depth is $O(\log n)$.

Next, we show how to construct a (stochastic) β-separating clustering chain $\mathbb{C}$ with $\beta = O(1)$ for any minor-free graphs, see Theorem 2.1. The construction of [CLPP23] can be seen as constructing a β-separating clustering chain $\mathbb{C}$ with $\beta = O(\log^2 n)$. (See Lemma 3.6 in [CLPP23], arXiv version.) Their construction does

not exploit minor-closed property and holds for general graphs. As we remarked above, one could probably improve their analysis to obtain $\beta = O(\log n)$, and this is the best for general graphs. We achieve constant β by exploiting minor-closed property. Although the same β can be achieved by the KPR decomposition (Klein-Plotkin-Rao [KPR93]), for the purpose of constructing balanced cuts in the next step, we need an additional guarantee that the clustering chain $\mathbb{C}$ has *bounded-hop property*. Roughly speaking, $\mathbb{C}$ has a bounded-hop property if, for every cluster C in $\mathbb{C}$, the *cluster contraction graph*, obtained from $G[C]$ by contracting every child cluster of C into a single vertex, has bounded hop-diameter. (See Definition 2.2 for a precise definition.) KPR decomposition (and its variants [AGMW10, AGG+14, Fil19]) does not guarantee the bounded-hop property — this is also why CLPP embedding cannot use KPR in their construction. Here, we build on the shortcut partition in a white-box fashion for minor-free graphs recently introduced by [CCL+24] to achieve the constant hop bound (and constant β). The basic idea is to sample a random radius in the ball carving process of [CCL+24]. The analysis of [CCL+24] already provides a constant hop bound; our job is to show that for every edge, not many random balls can potentially "reach" it, giving a constant β in the separating probability. Clearly, constant β and constant hop bound are the best one can hope for. Note that the original shortcut partition construction is deterministic, whereas our β-separating clustering chain is necessarily stochastic; analyzing such variant requires us to prove certain new property about the shortcut partition that has not been established before. For comparison, recently Filtser [Fil24a] constructed a padded-decomposition based off the buffered cop decomposition [CCL+24], but they emphasized that only a weak diameter guarantee can be achieved. Here we construct a strong-diameter padded decomposition with constant hop guarantee (Lemma 6.1) which is needed for the clustering chain. It turns out the proof is rather technical.[1]

Finally, we construct a (τ, ψ)-stochastic balanced cut for apex-minor-free graphs with $\tau = O(\log n) \cdot \psi$. Recall that in [CLPP23], $\tau = O(\log^6 n) \cdot \psi$ (see Lemma 4.2 in [CLPP23], arXiv version). By plugging the clustering chain with constant hop bound above into the construction of balanced cuts by [CLPP23], we already remove a factor of $\log^4 n$, giving $\tau = O(\log^2 n) \cdot \psi$. One key idea to remove another $\log n$ factor is a new analysis of *contraction sequence*: roughly speaking, an (a, b, c)-*contraction sequence* represents a contraction of a graph G into a single vertex through b rounds, where in each round, we contract a certain number of vertex-disjoint subgraphs of radius at most c; the total number of subgraphs we contract in b rounds is a. [CLPP23] showed that if an apex-minor-free graph G admits an (a, b, c)-contraction sequence, then $\mathrm{tw}(G) = O(bc\sqrt{a})$. Their basic idea is to argue that there exists a set S of a vertices such that every other vertex is within hop distance $O(bc)$ from the set and then apply known techniques to get treewidth bound $O(bc\sqrt{a})$. Here, we improve the treewidth bound by a factor of $\sqrt{b}$, which we show is optimal. First, by a simple reduction, we can assume that $c = 1$. Second, which is also the bulk of the technical details, we show the following structural property: Let Z be any maximal set of vertices in G such that their pairwise (hop) distance in G is $\Omega(b)$, then $|Z| \leq \frac{a}{b}$. Thus, by standard tools, we can conclude that $\mathrm{tw}(G) = O(\sqrt{Z}b) = O(\sqrt{ab})$ (for the case $c = 1$). This, in turn, leads to our improved bound $\tau = O(\log n) \cdot \psi$. Proving this structural property requires an in-depth analysis of the contraction sequence, which might be of independent interest.

By incorporating all the ideas above, we obtain an embedding with treewidth $O(\varepsilon^{-1} \log^3 n)$ as stated in Theorem 1.1. It seems to us that, to further improve any logarithmic factor, one has to significantly deviate from the framework of [CLPP23]. Recall that our expected distortion is $1 + O(\beta \log^2 n/\psi)$. One log factor comes from the fact that we must handle distances in $O(\log n)$ scales; this factor is unavoidable and also reflects in the treewidth lower bound $\Omega(\log n)$ by [CG04]. Another log factor comes from the technical subtlety mentioned in the overview of [CLPP23]: a single edge (x, y) could be cut by up to $\Omega(\log n)$ clusters in different levels of $\mathbb{C}$. This is inherent in any clustering chain: there is always an edge that is cut by $\Omega(\log n)$ clusters. One last log factor comes from the bound $\tau = O(\log n) \cdot \psi$ in (τ, ψ)-balanced cut. Removing this log factor seems to require a completely different technique to construct balanced cuts since the core component of our construction, which is the (a, b, c)-contraction sequence, is the best possible. While we are closer to Conjecture 1.1, its resolution remains elusive.

[1] For the readers who are familiar with the buffered cop decomposition: Roughly speaking, we need to bound the number of supernodes X that are at most distance $O(\Delta)$ away from a fixed vertex $\hat{v}$ within the domain of X. However, it is not with respect to the final domain $\mathrm{dom}(X)$, but the *initial* domain $\mathrm{dom}_0(X)$ at the time when X was initiated. As a result, many more supernodes could be distance $O(\Delta)$ away. See Section 6.3 and Lemma 6.6 for a precise statement.

2 Technical Overview

In this section, we provide key definitions from the discussion above, as well as a high-level overview of our embedding construction. A *clustering* of an edge-weighted graph $G = (V, E, \|\cdot\|)$ is a partition of the vertex set V where each *cluster* in the clustering is a connected subset of vertices. The strong (resp. weak) *diameter* of a cluster C is $\max_{x,y\in C} d_{G[C]}(x, y)$ (resp. $\max_{x,y\in C} d_G(x, y)$). We say that an edge $e \in E(G)$ is *cut* by a clustering $\mathcal{C}$ if the endpoints of the edge belong to two different clusters in $\mathcal{C}$. We now recall the notion of *clustering chain* [CLPP23].

Definition 2.1. (Clustering Chain) *A clustering chain of a connected graph G is a sequence $\mathbb{C} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k)$ of ever coarser clusterings G such that:*

- *$\mathcal{C}_k$ contains a single cluster V.*
- *$\mathcal{C}_0$ consists of vertices in V each as a singleton cluster.*
- *for every i, $\mathcal{C}_i$* refines *$\mathcal{C}_{i+1}$: every cluster of $\mathcal{C}_i$ is contained within a unique cluster of $\mathcal{C}_{i+1}$.*
- *every cluster $C \in \mathcal{C}_i$ has strong diameter at most 2^i.*

A clustering $\mathbb{C}$ naturally forms a hierarchy where the root of the hierarchy corresponds to the (only) cluster in $\mathcal{C}_k$, the leaves are singleton clusters in $\mathcal{C}_0$, scale-i clusters are clusters in $\mathcal{C}_i$, and the parent of a cluster $C \in \mathcal{C}_i$ is the cluster $C' \in \mathcal{C}_{i+1}$ such that $C \subseteq C'$. See Figure 1. We denote by $\mathcal{C}_i[C']$ the set of scale-i clusters in $\mathcal{C}_i$ that make up the scale-$(i+1)$ cluster C'. We often abuse the language and say that a cluster C of G is *in* $\mathbb{C}$ if C belongs to the clustering at some scale of $\mathbb{C}$. For a cluster C in $\mathbb{C}$, we denoted by $\mathbb{C}_{\downarrow C}$ a clustering chain of C obtained by taking all the clusters in $\mathbb{C}$ that are subsets of C (including C). Intuitively, $\mathbb{C}_{\downarrow C}$ is a sub-hierarchy of $\mathbb{C}$ rooted at C.

For our randomized embedding, we need to construct a *distribution* of clustering chains so that the probability of an edge being cut by a clustering $\mathcal{C}_i$ at a specific scale i is inverse-proportional to the diameter of the clusters in $\mathcal{C}_i$. We formalize this via a notion of *β-separating distribution*.

Definition 2.2. (β-Separating Distribution of Clustering Chains with Hop Guarantee) *We say that a distribution $\mathfrak{C}$ of clustering chains of G is β-separating for some parameter $\beta \geq 1$ if for every edge $e \in E(G)$ and every i in $\{0, \ldots, k\}$,*

$$\Pr_{\mathbb{C}\sim\mathfrak{C}}[e \text{ is cut by } \mathcal{C}_i \in \mathbb{C}] \leq \beta \cdot \frac{\|e\|}{2^i}. \tag{2.3}$$

For a clustering chain $\mathbb{C}$, We say that $\mathbb{C}$ has hop bound h if for every i in $\{0, \ldots, k-1\}$ and every cluster $C \in \mathcal{C}_{i+1}$, the graph obtained by starting from $G[C]$ and contracting every cluster in $\mathcal{C}_i$, denoted by $G[C]/\mathcal{C}_i[C]$, has hop-diameter at most h.

[CLPP23] showed that any n-vertex graph admits a β-separating distribution of clustering chains for $\beta = \log\Phi \cdot \log(n)$. Here in Section 6, we show β can be improved to $O(1)$ for minor-free graphs, and furthermore, the clustering chain has a small hop bound.

Theorem 2.1. *Let G be a graph excluding a fixed minor. There exists a β-separating distribution of clustering chains $\mathfrak{C}$ with $\beta = O(1)$. Furthermore, every chain $\mathbb{C} = \{\mathcal{C}_0, \ldots, \mathcal{C}_k\}$ in the support of $\mathfrak{C}$ has hop bound $h = O(1)$. The big-O in both β and h hides a dependence on the minor size.*

Next, we introduce the notion of a balanced cut. A *cut* of G, denoted by $\mathcal{F}$, is a set of vertex-disjoint clusters that are proper subsets of V. (We do not allow the cut to contain a single cluster V.) In this paper, we construct a cut by taking some clusters from a given clustering chain $\mathbb{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ of G: A cut $\mathcal{F}$ *respects* a clustering chain $\mathbb{C}$ if every cluster in $\mathcal{F}$ is in $\mathbb{C} \setminus \{\mathcal{C}_k\}$ (recall that $\mathcal{C}_k$ contains a single cluster V). See Figure 1(c). Let $\omega_V : V \to \mathbb{R}^+$ be a weight function on vertices of G. The *size* of $\mathcal{F}$ is the number of clusters that $\mathcal{F}$ contains. We say that a cut $\mathcal{F}$ of G is *balanced* with respect to ω_V if the total vertex weight of every connected component H of $G \setminus \mathcal{F}$ is at most $W/2$ where $W \coloneqq \sum_{u\in V} \omega_V(u)$.

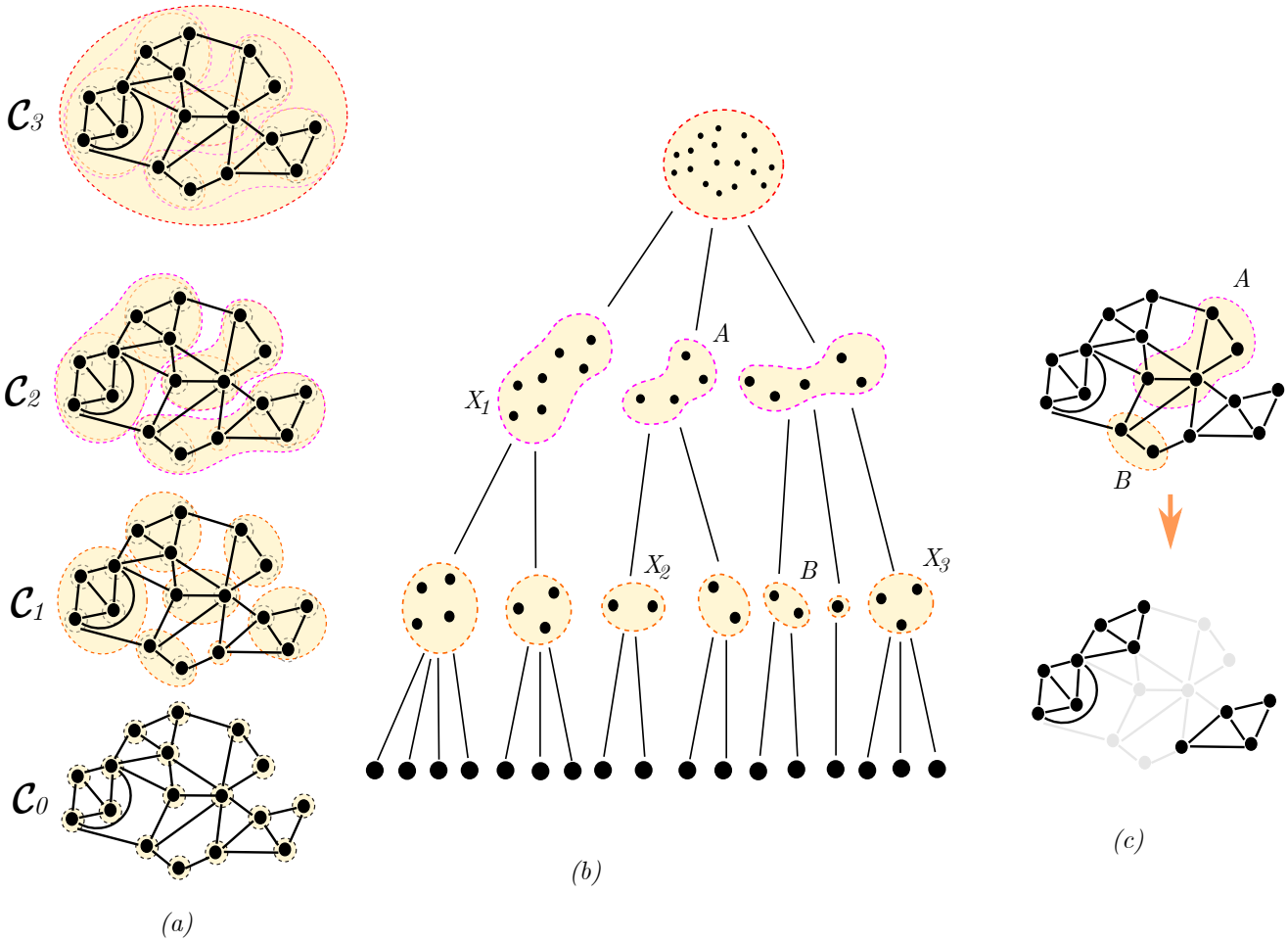


Figure 1: (a) A clustering chain $\mathbb{C} = \{\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$. (b) Viewing $\mathbb{C}$ as a hierarchy represented by a tree. (c) Balanced cut $\mathcal{F} = \{A, B\}$ respects $\mathbb{C}$ since the cut contains two clusters A and B (at different levels) of $\mathbb{C}$; removing this cut results in connected components of size at most $2n/3$ each. If $\mathcal{X} = \{X_1, X_2, X_3\}$ as in (b), then the cut $\mathcal{F}$ in (c) conforms $\mathcal{X}$.

Definition 2.3. ((τ,ψ)-Stochastic Balanced Cuts) *Let $G = (V, E, \|.\|)$ be a graph with vertex weight function ω_V, $\mathbb{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ be a clustering chain for G, and $\mathcal{X}$ be a set of clusters in $\mathbb{C}$. We say $(G, \mathbb{C})$ has a (τ,ψ)-stochastic balanced cut with respect to $(\omega_V, \mathcal{X})$ if there is a distribution $\mathfrak{F}$ of cuts of G:*

- [Respecting.] *Every cut $\mathcal{F}$ in the support of $\mathfrak{F}$ respects $\mathbb{C}$.*
- [Balanced.] *Every cut $\mathcal{F}$ in the support of $\mathfrak{F}$ is balanced with respect to ω_V.*
- [Conforming.] *Every cut $\mathcal{F}$ in the support of $\mathfrak{F}$ has the property that: for every cluster $C \in \mathcal{F}$, C is not strictly contained in a cluster in $\mathcal{X}$. (C may be in or above clusters in $\mathcal{X}$.) We say that $\mathcal{F}$ conforms $\mathcal{X}$.*
- [Small size.] *Every cut $\mathcal{F}$ in the support of $\mathfrak{F}$ contains at most τ clusters.*
- [Low probability.] *For every* non-singleton *cluster C in $\mathbb{C}$ such that $C \notin \mathcal{X}$:*

$$\Pr_{\mathcal{F}\sim\mathfrak{F}}[C \text{ is contained in } \mathcal{F}] \leq 1/\psi. \tag{2.4}$$

We say that $(G, \mathbb{C})$ admits a (τ,ψ)-stochastic balanced cut if for any cluster $C \in \mathbb{C}$, for any given weight function ω_C, and any subset of clusters $\mathcal{X}$ in $\mathbb{C}$, there exists a (τ,ψ)-stochastic balanced cut w.r.t $(G[C], \omega_C, \mathbb{C}_{\downarrow C}, \mathcal{X})$. (Notice that $\mathcal{X}$ can be in $\mathbb{C}$ outside $\mathbb{C}_{\downarrow C}$.)

The [conforming] property in the definition of stochastic balanced cut is somewhat artificial and unintuitive. This property is needed for a purely technical reason: at some step of the embedding construction, we will sample a balanced cut $\mathcal{F}$ from some subgraph H of G. However, we are not allowed to break some "boundary clusters" of H—these are clusters in $\mathbb{C}$ that are adjacent to vertices not in H—in the sense that every cluster in $\mathcal{F}$ either contains a whole boundary cluster or is disjoint from any boundary cluster. In other words, $\mathcal{F}$ has to conform the boundary clusters as defined in the [conforming] property.

When constructing a stochastic balanced cut, the most representative case is when $\mathcal{X} = \varnothing$, and it is simpler to focus on this case.

THEOREM 2.2. *Let G be an apex-minor-free graph, $\mathbb{C}$ be a clustering chain of G that has hop bound h. Then for any parameter $\psi \geq 1$, $(G, \mathbb{C})$ admits a (τ, ψ)-stochastic balanced cut $\mathfrak{F}$ with $\tau = O(h^2 \cdot \log(\Phi)) \cdot \psi$.*

Finally, we show that one can construct a low-distortion embedding into a low-treewidth graph for a graph G if it admits a good clustering chain and a stochastic balanced cut.

THEOREM 2.3. *Let G be an n-vertex edge-weighted graph with aspect ratio Φ. Suppose that (i) G admits a β-separating distribution $\mathfrak{C}$ of clustering chains for some $\beta \geq 1$ and (ii) for every clustering chain $\mathbb{C}$ in the support of $\mathfrak{C}$, $(G, \mathbb{C})$ admits a (τ, ψ)-stochastic balanced cut. Then G can be stochastically embedded into graphs of treewidth $O(\tau + \log(n\Phi))$ and with expected distortion $1 + \frac{O(\beta \log(n\Phi) \log \Phi)}{\psi}$.*

We observe that Theorem 2.1, Theorem 2.2, and Theorem 2.3 together implies Theorem 1.1.

Proof. [of Theorem 1.1] Let $G = (V, E, \|\cdot\|)$ be an apex-minor-free graph. By Theorem 2.1, there exists a $\beta = O(1)$-separating distribution $\mathfrak{C}$ of clustering chains of hop bound $h = O(1)$. We choose

$$\psi := \frac{\beta \log(n\Phi) \log(\Phi)}{\varepsilon}$$

Let $\mathbb{C}$ be a clustering chain sampled from $\mathfrak{C}$. Let $\mathcal{X}$ be any set of clusters in $\mathbb{C}$. Let C be any cluster in $\mathbb{C}$. Since G is apex-minor-free, $G[C]$ is also apex-minor-free. Thus, by Theorem 2.2, $(G[C], \mathbb{C}_{\downarrow C})$ admits a (τ, ψ)-balanced cut conforming $\mathcal{X}$ with $\tau = O(h^2 \log \Phi) \cdot \psi = O(\log^2 \Phi \log(n\Phi)/\varepsilon)$. By Theorem 2.3, we can embed G stochastically into graphs with treewidth $O(\tau + \log(n\Phi)) = O(\log^2 \Phi \log(n\Phi)/\varepsilon)$. The expected distortion of the embedding is:

$$1 + O\left(\frac{\beta \log(n\Phi) \log \Phi}{\psi}\right) = 1 + O(\varepsilon),$$

by our choice of ψ. ∎

3 Preliminaries

We use mostly standard graph terminology. We denote by G an edge-weighted graph with nonnegative length function $\|\cdot\|$ on the edges. For any graph H, we denote by $V(H)$ and $E(H)$ the vertex set and edge set of H, respectively. For a subset of vertices S, we denote by $G[S]$ the subgraph of G induced by S.

A *tree decomposition* of a graph $G = (V, E)$ is a tree $\mathcal{T}$ where each node $x \in \mathcal{T}$ is associated with a subset of V called $\mathsf{bag}(x)$, such that: (i) $\cup_{x \in \mathcal{T}} \mathsf{bag}(x) = V$, (ii) if u and v are adjacent in G, there exists a node x in $\mathcal{T}$ such that $\{u, v\} \subseteq \mathsf{bag}(x)$, and (iii) for every $u \in V$, all the nodes whose bags contains u induce a connected subtree of $\mathcal{T}$. The *width* of a tree decomposition $\mathcal{T}$ is $\max_{x \in \mathcal{T}} |\mathsf{bag}(x)| - 1$ and the *treewidth* of G, denoted by $\mathrm{tw}(G)$, is the minimum width over all valid tree decompositions of G.

Given a graph H, we say that G is *H-minor-free* if G excludes H as a minor. We say that G is *minor-free* if it excludes a graph of constant size as a minor. A graph K is an *apex* graph if there exists a vertex u, called the apex of K, such that $K - \{u\}$ is a planar graph. We say that G is *apex-minor-free* if it excludes an apex graph of constant size as a minor. We will rely on the following lemma in our construction:

LEMMA 3.1. (LEMMA 2.1 IN [CLPP23]) *For every apex graph K, there is a positive integer α_K such that if $G = (V, E)$ is K-(apex-)minor-free and $Z \subseteq V$ is a subset of vertices such that every vertex of G is at hop distance at most d from a vertex of Z, then $\mathrm{tw}(G) \leq \alpha_K \cdot d\sqrt{|Z|}$.*

4 The Embedding Framework: Proof of Theorem 2.3

Let G be a graph satisfying the assumptions in Theorem 2.3. Let $\mathbb{C}$ be a clustering chain for G sampled from the distribution $\mathfrak{C}$. Roughly speaking, the high-level idea is to (i) sample a balanced cut $\mathcal{F}$ to cut G into connected components of size at most $n/2$, (ii) pick a representative vertex v_C for each cluster $C \in \mathcal{F}$ and make a bag from $\{v_C : C \in \mathcal{F}\}$, and (iii) recursively embed vertices in each connected component[2] of $G \setminus \mathcal{F}$ and each cluster in $\mathcal{F}$. As we recurse on each cluster C in $\mathcal{F}$, we will need to sample a balanced cut for C from $\mathbb{C}_{\downarrow C}$; this is possible by the definition of stochastic balanced cut in Definition 2.3.

When we recurse to embed vertices in a connected component, say H, of $G \setminus \mathcal{F}$, clusters in $\mathcal{F}$ become "boundary clusters" of H, meaning that vertices in H can only connect to other vertices in $G \setminus H$ by going through these clusters. In the algorithm below, we will take one representative per boundary cluster and embed these representatives along with vertices in H in the recursion. We will keep the number of such representatives small by the following trick reminiscent to the construction of r-division: whenever the number of boundary clusters is above a certain threshold, we will sample a balanced cut to balance the number of boundary clusters.

Now, we describe the embedding algorithm in more detail. Throughout this section, fix ε in $(0, 1)$. The main embedding procedure $\textsc{Embed}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$ takes four parameters as input:

- the *terminals* T, a subset of the vertices of G that we want to embed,
- the *current piece* P, a subgraph[3] induced by a cluster in $\mathbb{C}$ such that all terminals are within P,
- the *boundary vertices* ∂T, a subset of T that contains representatives of boundary clusters, and
- the *boundary clusters* $\partial\mathcal{C}$, a set of clusters in $\mathbb{C}$ where every cluster C in $\partial\mathcal{C}$ satisfies $|C \cap \partial T| = 1$. (We remark that intuitively $\partial\mathcal{C}$ are clusters that were added to a cut during earlier recursive calls.)

The procedure returns a pair $(\hat{G}, \hat{\mathcal{T}})$, where $\hat{G}$ is a small treewidth graph with vertex set T, and $\hat{\mathcal{T}}$ is a tree decomposition for $\hat{G}$ where every vertex in ∂T appears in the root bag of $\hat{\mathcal{T}}$. Implementation details of $\textsc{Embed}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$ are in Figure 2. The wrapper function $\textsc{Embed}(G)$ sample a clustering chain $\mathbb{C}$ from distribution $\mathfrak{C}$ in the assumption of Theorem 2.3, and produce an embedding of G by calling $\textsc{Embed}_{\mathbb{C}}(V, G, \varnothing, \varnothing)$. See Figure 3 for an illustration. We bound the treewidth of $\hat{\mathcal{T}}$ in Section 4.1 and bound the expected distortion of embedding into $\hat{G}$ in Section 4.2, completing the proof of Theorem 2.3.

We will use a (randomized) helper procedure $\textsc{Cut}_{\mathbb{C}}(S, P, \partial\mathcal{C})$, where $S \subseteq V(P)$ and $\partial\mathcal{C}$ is the set of boundary clusters of graph P with respect to ∂T. This procedure will sample a stochastic random cut $\mathcal{F}$ in P conforming $\partial\mathcal{C}$ to balance the number of vertices in S, achieved by setting the weight $\omega_{V(P)}(u) = 1$ for every $u \in S$ and $\omega_{V(P)}(v) = 0$ for every $v \in V(P) \setminus S$. Since $\mathcal{F}$ conforms $\partial\mathcal{C}$, no cluster in $\mathcal{F}$ will be a proper subset of a cluster in $\partial\mathcal{C}$ (though $\mathcal{F}$ could contain clusters in $\partial\mathcal{C}$). Note that we only call procedure $\textsc{Cut}_{\mathbb{C}}(S, P, \partial\mathcal{C})$ when P is the subgraph of G induced by some cluster $\hat{C} \in \mathbb{C}$; we will show in Observation 4.1 below that this is the case during the course of the embedding algorithm.

Observation 4.1. *For any recursive call* $\textsc{Embed}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$, $P = G[\hat{C}]$ *for some cluster* $\hat{C} \in \mathbb{C}$.

Proof. In the beginning, the observation follows from the fact that $P = G$ and the root of $\mathbb{C}$ is V. Inductively, we assume that in the call $\textsc{Embed}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$, $P = G[\hat{C}]$ for some cluster $\hat{C} \in \mathbb{C}$. The call $\textsc{Embed}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$ spawns two types of recursive calls: (i) $\textsc{Embed}_{\mathbb{C}}(T_C, G[C], \partial T_C, \partial\mathcal{C})$ for a cluster $C \in \mathcal{F}$ in Step 3 and (ii) $\textsc{Embed}_{\mathbb{C}}(T_H, P, \partial T_H, \partial\mathcal{C}_H)$ in Step 4. The observation clearly holds for type (ii). For type (i), the next piece is $G[C]$, so the observation also holds. ∎

[2] More precise, for presentation, the graph considered in the next recursive step to embed a connected component of $G \setminus \mathcal{F}$, say H, remains to be named G. However, we will set the weight $\omega_V(u) = 1$ for every $u \in V(H)$ and 0 otherwise. Therefore, effectively, we only embed vertices of H in the recursion.

[3] we remark that we pass P as input only to help intuition; a suitable P could be inferred from T and $\mathbb{C}$ alone

Embed$_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$**:** Terminal set T, current graph P (called piece), boundary terminals ∂T, and boundary clusters $\partial\mathcal{C}$.

1. *Base case.*

   If T contains at most 4τ vertices, then define $\hat{G}$ to be a clique on T, where the weight between u and v is $\delta_G(u, v)$; define $\hat{\mathcal{T}}$ to be a single bag containing all vertices in T; and return $(\hat{G}, \hat{T})$. If T contains more than 4τ vertices, continue.

2. *Compute a balanced cut $\mathcal{F}$.*

   If ∂T contains more than 4τ vertices, set $S \leftarrow \partial T$; otherwise set $S \leftarrow T$. Sample a cut $\mathcal{F}$ by calling $\textsc{Cut}_{\mathbb{C}}(S, P, \partial\mathcal{C})$. Notice that $\mathcal{F}$ is balanced with respect to S and conforms $\partial\mathcal{C}$.

3. *Recurse on $\mathcal{F}$.*

   For each cluster C in $\mathcal{F}$, let $T_C \coloneqq T \cap C$, and let $\partial T_C \coloneqq \partial T \cap C$. Let $\partial\mathcal{C}_C$ be the set of clusters in $\partial\mathcal{C}$ whose representatives are in ∂T_C. Recursively call $\textsc{Embed}_{\mathbb{C}}(T_C, G[C], \partial T_C, \partial\mathcal{C}_C)$ to find an embedding $(\hat{G}_C, \hat{\mathcal{T}}_C)$.

4. *Recurse on $P \setminus \bigcup \mathcal{F}$.*

   For each cluster C in $\mathcal{F}$ with $C \cap T \neq \varnothing$, select an arbitrary representative terminal v_C from $C \cap T$; let $T_{\mathcal{F}}$ denote the set of representatives for all clusters in $\mathcal{F}$. For each connected component H of $P \setminus \bigcup \mathcal{F}$, let $T_H \coloneqq (T \cap H) \cup T_{\mathcal{F}}$, let $\partial T_H \coloneqq (\partial T \cap H) \cup T_{\mathcal{F}}$, and let $\partial\mathcal{C}_H$ be the set containing all clusters in $\mathcal{F}$ and in $\partial\mathcal{C}$ whose representatives are in ∂T_H. Recursively call $\textsc{Embed}_{\mathbb{C}}(T_H, P, \partial T_H, \partial\mathcal{C}_H)$ to find an embedding $(\hat{G}_H, \hat{\mathcal{T}}_H)$. (Note here that we pass P to the next recursive call instead of H.)

5. *Construct the host graph $\hat{G}$.*

   Take the union of all $\hat{G}_C$ and $\hat{G}_H$ recursively constructed above (identifying all vertices in $T_{\mathcal{F}}$, which may appear in multiple graphs $\hat{G}_H$). For each cluster C in $\mathcal{F}$, add an edge from the representative vertex v_C to every vertex u in C, with weight equal to $\delta_G(v_C, u)$. The resulting graph is $\hat{G}$.

6. *Construct the tree decomposition $\hat{\mathcal{T}}$.*

   The root bag of $\hat{\mathcal{T}}$ contains every vertex in $T_{\mathcal{F}} \cup \partial T$. The children of the root bag are the roots of the tree decompositions for the recursive embeddings ($\hat{\mathcal{T}}_C$ and $\hat{\mathcal{T}}_H$). For every cluster C in $\mathcal{F}$, we add v_C to every bag of $\hat{\mathcal{T}}_C$. The resulting tree is $\hat{\mathcal{T}}$. Return $(\hat{G}, \hat{\mathcal{T}})$.

Embed(G)**:** graph $G = (V, E, \|\cdot\|)$

1. Sample $\mathbb{C}$ from distribution $\mathfrak{C}$ in the assumption of Theorem 2.3.
2. Return $\textsc{Embed}_{\mathbb{C}}(V, G, \varnothing, \varnothing)$.

Figure 2: The embedding algorithm.

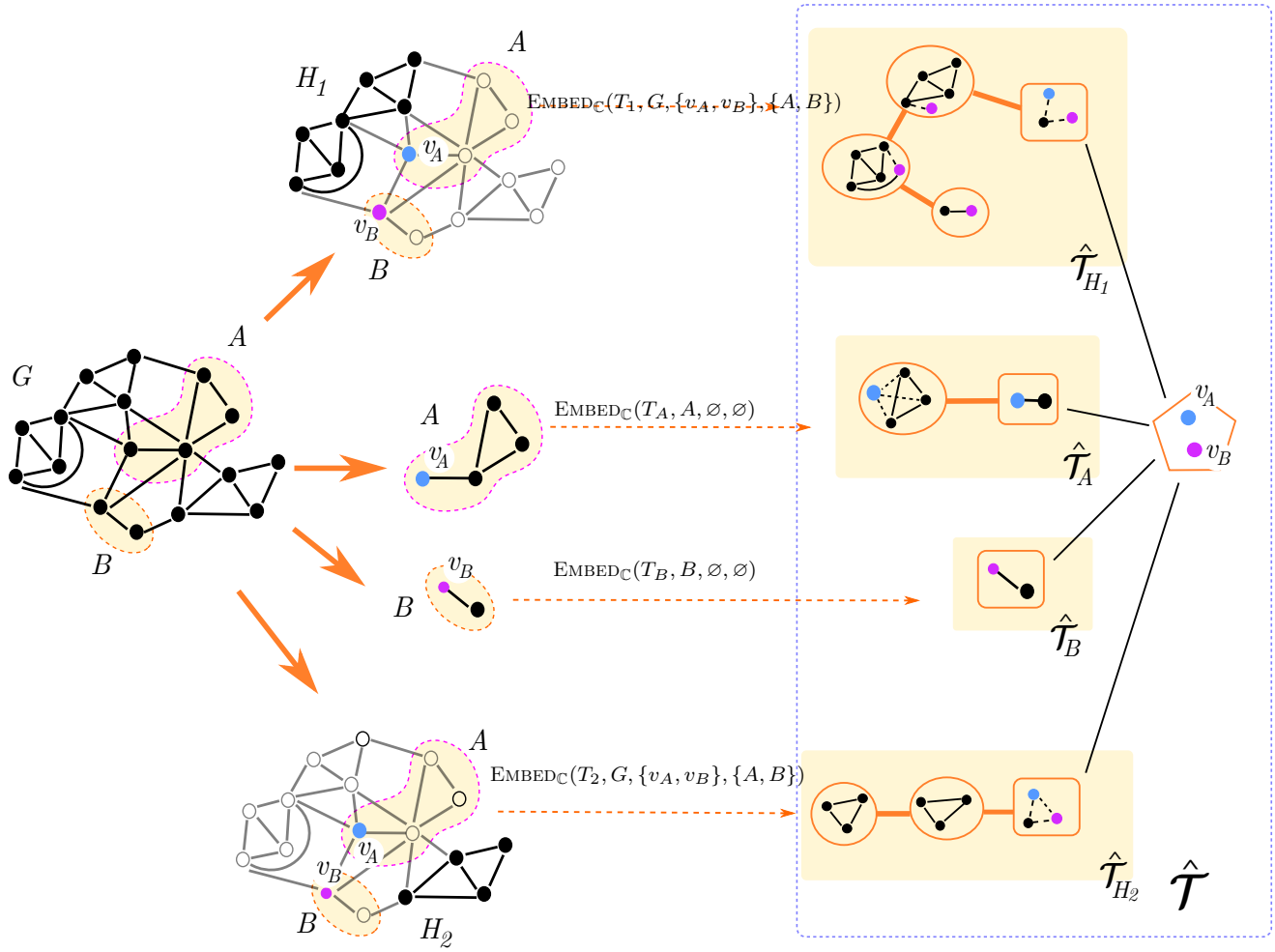


Figure 3: An embedding of G obtained by calling EMBED$_{\mathbb{C}}(V, G, \varnothing, \varnothing)$: the algorithm will call CUT$_{\mathbb{C}}(V, G, \varnothing)$ to sample a cut $\mathcal{F}$; assume that $\mathcal{F} = \{A, B\}$ as in the figure. Then, the algorithm will recursively embed the terminals in components of $G \setminus \mathcal{F}$, which are H_1, H_2. Note that the graphs we recurse on to embed terminals in H_1 and H_2 are G. Terminals are filled vertices, while non-terminals are hollow vertices. The algorithm also recursively embeds clusters in $\mathcal{F}$, which are A and B. The final tree decomposition $\hat{\mathcal{T}}$ is obtained by connecting four tree decompositions from four recursive calls via the root bag $\{v_A, v_B\}$, which are representative vertices of A and B, respectively. Dashed edges are new edges added during the embedding process.

4.1 Bounding the treewidth First, we show that $\hat{\mathcal{T}}$ output by EMBED(G) is a valid tree decomposition of $\hat{G}$.

LEMMA 4.1. *$\hat{\mathcal{T}}$ is a valid tree decomposition of $\hat{G}$.*

Proof. By induction, we assume that in procedure EMBED$_{\mathbb{C}}(T, P, \partial T, \partial \mathcal{C})$, the tree decompositions $\hat{\mathcal{T}}_C$ in Step 3 and $\hat{\mathcal{T}}_H$ in Step 4 are valid tree decompositions of $\hat{G}_C$ and $\hat{G}_H$, respectively. Recall that recursive calls are made on subsets T_H and T_C, which are pairwise disjoint *except* for the vertices of $T_{\mathcal{F}}$. Thus, if a vertex in $\hat{G}$ appears in multiple subtrees, then it is in $T_{\mathcal{F}}$ (and thus in the root bag). Further, $T_{\mathcal{F}}$ appears (if at all) in the root bag of all tree decompositions for the recursive embeddings. Recursively, each subtree is a tree decomposition. Thus, the bags containing any given vertex form a connected subtree.

Let e be an edge in $\hat{G}$. Recall that in Step 5, we only add edges between the representative v_C of C to vertices in C for every cluster $C \in \mathcal{F}$ (aside from edges in $\hat{G}_C$ and $\hat{G}_H$) to form $\hat{G}$. If e is an edge between some v_C and a vertex in C (that is, if it was added in Step 5), then there is a bag containing both endpoints of e (as we added

v_C to every bag of the tree decomposition for cluster C). Otherwise, recursively, there is some bag that contains both endpoints of e. ∎

To bound the width of $\hat{T}$, we must first bound the depth of the recursion tree. Let X be a call $\textsc{Embed}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$ that occurs during the algorithm. Let $\text{scale}(P)$ denote the scale of the cluster $\hat{C}$ inducing $P = G[\hat{C}]$ in $\mathbb{C}$. Note that $\text{scale}(P) \leq \lceil \log \Phi \rceil$. We define the potential function:

$$\phi(T, P, \partial T) := 5 \log_2 |T| + \text{scale}(P) + \frac{|\partial T|}{\tau}.$$

Recall that τ is the parameter in the (τ, ψ)-stochastic balanced cut $\mathcal{F}$ sampled in Step 2. The root call has potential $\phi(T, P, \partial T) = O(\log n) + O(\log \Phi) + 0 = O(\log(n\Phi))$.

LEMMA 4.2. *If call* $X := \textsc{Embed}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$ *makes a recursive call* $X' := \textsc{Embed}_{\mathbb{C}}(T', P', \partial T', \partial\mathcal{C}')$, *then* $\phi(T', P', \partial T') \leq \phi(T, P, \partial T) - 1$. *As a corollary, the recursion tree has depth* $O(\log(n\Phi))$.

Proof. In Step 3, X makes calls $X_C := \textsc{Embed}_{\mathbb{C}}(T_C, G[C], \partial T_C, \partial\mathcal{C})$ for every cluster C in $\mathcal{F}$. We have $|T_C| \leq |T|$ (as $T_C \subseteq T$) and $|\partial T_C| \leq |\partial T|$ (as $\partial T_C \subseteq \partial T$). We have $\text{scale}(G[C]) \leq \text{scale}(P) - 1$ by the definition of a cut. We conclude that $\phi(T_C, C, \partial T_C) \leq \phi(T, P, \partial T) - 1$.

In Step 4, X makes calls $X_H := \textsc{Embed}_{\mathbb{C}}(T_H, P, \partial T_H, \partial\mathcal{C})$ for every connected component H in $P \setminus \bigcup \mathcal{F}$. We consider two cases based on Step 2 of X.

- Case 1: $|\partial T| \leq 4\tau$, so that X selects a balanced cut $\mathcal{F}$ with respect to the terminals T.

  In this case, we have $|T_H| \leq \frac{1}{2} \cdot |T| + |T_{\mathcal{F}}|$. Note that $|T_{\mathcal{F}}| \leq \tau$ since $\mathcal{F}$ is sampled from a (τ, ψ)-stochastic balanced cut. Observe that $|T| \geq 4\tau$ (because X makes a recursive call which means the base case of Step 1 was not reached). Thus, $|T| \geq 4 \cdot |T_{\mathcal{F}}|$, giving $|T_H| \leq \frac{3}{4} \cdot |T|$, and so $5 \log_2 |T_H| \leq 5 \log_2 |T| - 2$. Further, $|\partial T_H| \leq |\partial T| + \tau$ (because we add at $|T_{\mathcal{F}}| \leq \tau$ points into the boundary), and so $\frac{|\partial T_H|}{\tau} \leq \frac{|\partial T|}{\tau} + 1$. We conclude that $\phi(T_H, P, \partial T_H) \leq \phi(T, P, \partial T) - 1$.

- Case 2: $|\partial T| > 4\tau$, so that X selects a balanced cut $\mathcal{F}$ with respect to ∂T.

  In this case, we have $|T_H| \leq |T|$. Further, $|\partial T_H| \leq \frac{1}{2} \cdot |\partial T| + \tau \leq |\partial T| - \tau$ and hence $\frac{|\partial T_H|}{\tau} \leq \frac{|\partial T|}{\tau} - 1$. We conclude that $\phi(T_H, P, \partial T_H) \leq \phi(T, P, \partial T) - 1$.

This concludes the proof. ∎

LEMMA 4.3. $\hat{\mathcal{T}}$ *has width* $O(\tau + \log(n\Phi))$.

Proof. By Lemma 4.2, the recursion tree has depth $O(\log(n\Phi))$. This means that, in total, we add at most $O(\log(n\Phi))$ representative vertices v_C to a bag in Step 6 over the course of the algorithm. Excluding the vertices $\{v_C\}$, every bag contains a set of vertices that is $T_{\mathcal{F}} \cup \partial T$, for some ∂T that is passed as a parameter to EMBED. Thus, every bag contains at most $\tau + |\partial T|$ vertices.

Let $X := \textsc{Embed}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$ be a call that occurs during the algorithm. We show that $|\partial T| \leq 5 \cdot \tau$. We proceed by induction, starting at the root of the recursion tree and working downward. Indeed, let $X' := \textsc{Embed}_{\mathbb{C}}(T', P', \partial T', \partial\mathcal{C}')$ be the parent call of X in the recursion tree (and if X is the root, then $\partial T = \varnothing$ by construction). Let $\mathcal{F}'$ be the cut sampled during X'. If $P = G[C']$ for some cluster $C' \in \mathcal{F}'$ (that is, if X was called recursively during Step 3), then $\partial T \subseteq \partial T'$ and the induction hypothesis proves the claim. Otherwise, X was called recursively during Step 4, and so all terminals T are contained in $H \cup \mathcal{F}'$ for some connected component H of $P' \setminus \bigcup \mathcal{F}'$. In particular, T contains at most τ terminals in $\mathcal{F}'$ (the representative terminals $T_{\mathcal{F}'}$), and so we have $|\partial T| \leq |\partial T' \cap H| + \tau$. If $|\partial T'| \leq 4 \cdot \tau$, then we are done; otherwise, if $|\partial T'| > 4 \cdot \tau$, the cut $\mathcal{F}$ is a balanced separator for $\partial T'$ and so $|\partial T' \cap H| \leq |\partial T'|/2 \leq 2.5 \cdot \tau$ — note here that $|\partial T'| \leq 5 \cdot \tau$ by induction — giving $|\partial T| \leq 5 \cdot \tau$ as desired. ∎

4.2 Bounding the expected distortion It will be helpful to bound the distortion of an *edge* $e = (u, v)$, rather than an arbitrary *path* between u and v. This will simplify the analysis because either a cut $\mathcal{F}$ cuts edge e, or it does not — if we instead consider a path, $\mathcal{F}$ may cut multiple edges along the path. We generalize this notion. Let $\mathcal{X}$ be a set of clusters in a clustering chain $\mathbb{C}$. Let $G/\mathcal{X}$ denote the graph obtained by contracting every cluster in $\mathcal{X}$ into a supernode, and let $\eta : V \to V(G/\mathcal{X})$ be the surjective map that takes a vertex in G to the corresponding vertex in $G/\mathcal{X}$ after contraction. We say that a pair of vertices (u, v) is a *pseudo-edge with respect to $\mathcal{X}$ in G* if there is an edge between $\eta(u)$ and $\eta(v)$ in $G/\mathcal{X}$. We simply say that (u, v) is a pseudo-edge with respect to $\mathcal{X}$ if the graph G is clear from the context.

OBSERVATION 4.2. *Let (u, v) be a pseudo-edge with respect to $\mathcal{X}$ in G where $\mathcal{X}$ is a set of clusters in $\mathbb{C}$. Let $\mathcal{F}$ be a cut conforming $\mathcal{X}$, such that u is in some cluster C_u in $\mathcal{F}$, but v is not in C_u. Then, for every vertex u' in C_u, the pair (u', v) is a pseudo-edge with respect to $\mathcal{X} \cup \{C_u\}$. (Note that C_u might contain some clusters in $\mathcal{X}$ as subclusters.)*

Proof. Let $\eta : V \to V(G/\mathcal{X})$ be as above. Define $\eta' : V \to V(G/(\mathcal{X} \cup \{C_u\}))$ to be the function that sends vertex x to the supernode C_u if $x \in C_u$, and to $\eta(x)$ if $x \notin C_u$. As $\eta(v)$ and $\eta(u)$ are adjacent in $V(G/\mathcal{X})$, they remain adjacent after the cluster C_u is contracted; that is, $\eta'(u)$ and $\eta'(v)$ are adjacent. Now observe that every vertex u' in C_u is mapped to $\eta'(u)$ by η', and so (u', v) is a pseudo-edge. ∎

Let (u, v) be a pair of vertices. We say that a clustering chain $\mathbb{C}$ *splits (u, v) at scale i* if i is the *smallest* number such that, for every scale $s > i$, there is some cluster C in $\mathbb{C}$ at scale s that contains both u and v.

OBSERVATION 4.3. *Let u and v be vertices that are split at scale i by $\mathbb{C}$. Let C be a cluster (at* any *scale) in $\mathbb{C}$ that contains u but not v. Then, for every vertex u' in C, the pair (u', v) is split at scale i by $\mathbb{C}$.*

Proof. This follows from the nested structure of the clusters in $\mathbb{C}$. For every scale $s > i$, there is a cluster C_s that contains both u and v. The nested structure of $\mathbb{C}$ implies that $C \subseteq C_s$, and so C_s contains both u' and v. At scale i, there is a cluster C_i in $\mathbb{C}$ that contains u but not v. We have $C \subseteq C_i$, so C_i contains u' but not v. ∎

The key to the distortion analysis is the following lemma.

LEMMA 4.4. *Let u and v be vertices in G. Let $\mathbb{C}$ be a clustering chain sampled from distribution $\mathfrak{C}$ for G that splits (u, v) at scale i. For any call to* $\text{EMBED}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$ *such that*

- *the set of terminals T contains u and v, and*
- *(u, v) is a pseudo-edge with respect to $\partial\mathcal{C}$,*

the output embedding $\hat{G}$ satisfies

$$\mathbb{E}[\delta_{\hat{G}}(u, v)] \leq \delta_G(u, v) + O\left(2^i \cdot \frac{\log(n\Phi)}{\psi}\right),$$

where the expectation is over the sampled random cuts.

Before proving this lemma, we show that it suffices for a good multiplicative distortion bound.

LEMMA 4.5. *Let $(\hat{G}, \hat{\mathcal{T}})$ be the output of* $\text{EMBED}_{\mathbb{C}}(V, G, \varnothing, \varnothing)$. *For any edge $e = (u, v)$ in G, in expectation we have*

$$\mathbb{E}[\delta_{\hat{G}}(u, v)] \leq \left(1 + O\left(\beta \cdot \frac{\log(n\Phi)\log(\Phi)}{\psi}\right)\right) \cdot \|e\|.$$

Proof. The first step in creating $\hat{G}$ is to sample a clustering chain $\mathbb{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ from a β-separating distribution $\mathfrak{C}$. We split the analysis into cases, based on the scale i in $\{0, \ldots, k\}$ that splits (u, v). There are $k = O(\log \Phi)$

different cases:

$$\begin{aligned}
\mathbb{E}[\delta_{\hat{G}}(u,v)] &= \sum_{i=1}^{k} \mathbb{E}[\delta_{\hat{G}}(u,v) \mid \mathbb{C} \text{ splits } (u,v) \text{ at } i] \cdot \Pr[\mathbb{C} \text{ splits } (u,v) \text{ at } i] \\
&\le \sum_{i=1}^{k} \left(\|e\| + O\left(2^i \cdot \frac{\log(n\Phi)}{\psi}\right)\right) \cdot \Pr[\mathbb{C} \text{ splits } (u,v) \text{ at } i] && \text{(by Lemma 4.4)} \\
&\le \|e\| + \sum_{i=1}^{k} O\left(2^i \cdot \frac{\log(n\Phi)}{\psi}\right) \cdot \Pr[\mathbb{C} \text{ splits } (u,v) \text{ at } i] \\
&\le \|e\| + \sum_{i=1}^{k} O\left(2^i \cdot \frac{\log(n\Phi)}{\psi}\right) \cdot \beta \cdot \frac{\|e\|}{2^i} && \text{(by Definition 2.2)} \\
&\le \|e\| + O\left(\beta \cdot \frac{\log(n\Phi)\log\Phi}{\psi}\right) \cdot \|e\|
\end{aligned}$$

The inequality in the second line follows because every edge (u,v) is a pseudo-edge with respect to $\varnothing$; thus we may apply Lemma 4.4. The last line follows from the fact that $k = O(\log \Phi)$. ∎

It remains to prove Lemma 4.4.

Proof. [of Lemma 4.4] Recall that we defined the potential function

$$\phi(T, P, \partial T) := (5 \log |T|) + \text{scale}(P) + \frac{|\partial T|}{\tau}.$$

To prove the lemma, we prove the stronger statement:

> If $\textsc{Embed}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$ is called, where (1) T contains u and v, and (2) (u,v) is a pseudo-edge with respect to $\partial\mathcal{C}$ in P, and then
>
> $$\mathbb{E}[\delta_{\hat{G}}(u,v)] \le \delta_G(u,v) + \alpha \cdot \frac{2^i \cdot \phi(T,P,\partial T)}{\psi}$$
>
> where i is the scale at which $\mathbb{C}$ splits (u,v), and where α is a sufficiently large constant.

Notice that $\phi(T, P, \partial T) = O(\log(n\Phi))$, so proving this is sufficient to prove the lemma. We proceed by induction on $\phi(T, P, \partial T)$.

Base case. The base case is when $|T| \le 4 \cdot \tau$. (In particular, this case must occur if $\phi(T, P, \partial T) \le 4 \cdot \tau$.) In this case, the graph $\hat{G}$ is a clique on T, and $\delta_{\hat{G}}(u,v) = \delta_G(u,v)$.

Inductive case. In the inductive case, the procedure Embed samples a cut $\mathcal{F}$ that conforms to $\partial\mathcal{C}$ in Step 3 from distribution $\mathfrak{F}$. We will consider three cases below. To find the expected distortion, we will take a weighted sum of the distortion incurred for each of these cases. To simplify notation, we define the *distortion overhead* D to be

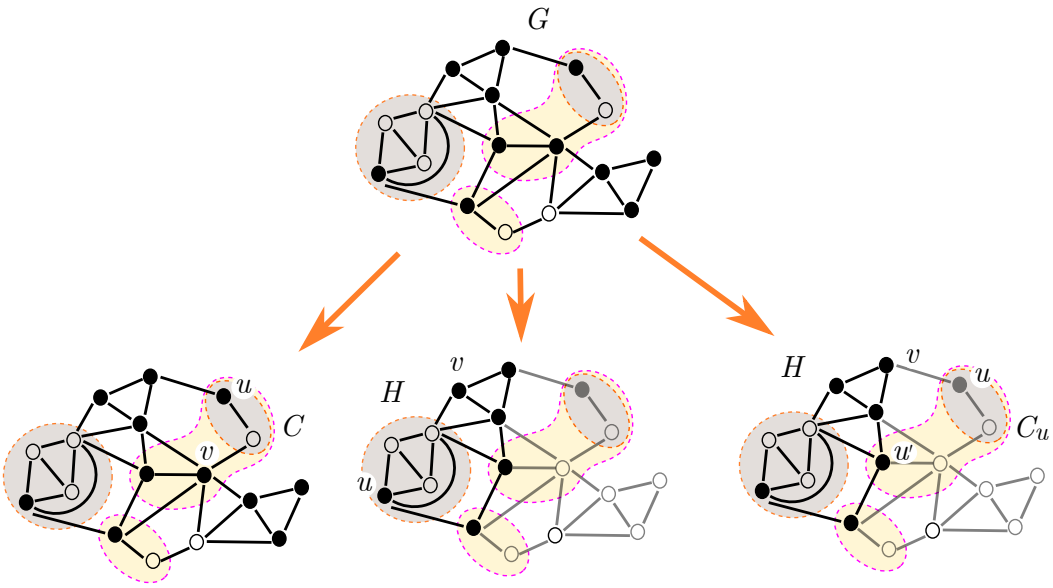
$$D := \delta_{\hat{G}}(u,v) - \left(\delta_G(u,v) + \alpha \cdot \frac{2^i \cdot (\phi(T,P,\partial T) - 1)}{\psi}\right).$$

(Note that the term in the parenthesis will arise naturally when we appeal to the induction hypothesis.) To prove the claim, we show that $\mathbb{E}[D] \le \alpha \cdot \frac{2^i}{\psi}$ where the expectation is over the sampled random cuts.

Case ξ_1: Vertices u and v are assigned to the same cluster C in $\mathcal{F}$; see Figure 4(a).

In this case, the procedure recurses on $\textsc{Embed}(T_C, G[C], \partial T_C, \partial\mathcal{C}_C)$. By Lemma 4.2, $\phi(T_C, G[C], \partial T_C) \le \phi(T, P, \partial T) - 1$. By assumption, both u and v are in T_C, and (u,v) is a pseudo-edge with respect to $\partial\mathcal{C}_C$. Thus, we can apply induction, and conclude that $\mathbb{E}[\delta_{\hat{G}}(u,v) \mid \text{Case } \xi_1 \text{ occurs}] \le \delta_G(u,v) + \alpha \cdot \frac{2^i \cdot (\phi(T,P,\partial T)-1)}{\psi}$. In other words,

$$\mathbb{E}[D \mid \text{Case } \xi_1 \text{ occurs}] = 0.$$



(a) Case 1 (b) Case 2 (c) Case 3

Figure 4: Three cases in the analysis of the distortion overhead D. Filled vertices are terminals. The gray clusters form $\partial\mathcal{C}$. The yellow clusters form $\mathcal{F}$.

Case ξ_2: Vertices u and v are assigned to the same connected component H of $P \setminus \bigcup \mathcal{F}$; see Figure 4(b).

This case is almost the same as the Case ξ_1. In this case, we recurse on $\textsc{Embed}_{\mathbb{C}}(T_H, P, \partial T_H, \partial\mathcal{C}_H)$. By Lemma 4.2, $\phi(T_H, P, \partial T_H) \le \phi(T, P, \partial T) - 1$. Vertices u and v are in T_H, and (u, v) is a pseudo-edge with respect to $\partial\mathcal{C}_H$ (because it is a pseudo-edge with respect to $\partial\mathcal{C} \subseteq \partial\mathcal{C}_H$). By induction, $\mathbb{E}[\delta_{\hat{G}}(u, v) \mid \text{Case } \xi_2 \text{ occurs}] \le \delta_G(u, v) + \alpha \cdot \frac{2^i \cdot (\phi(T,P,\partial T)-1)}{\psi}$. In other words,

$$\mathbb{E}[D \mid \text{Case } \xi_2 \text{ occurs}] = 0.$$

Case ξ_3: Vertices u and v are cut by $\mathcal{F}$; see Figure 4(c).

We claim that (without loss of generality) *there is some cluster* C_u *in* $\mathcal{F}$ *that contains* u *(but not* v*).* Indeed, let $\eta : V \to V(G/\partial\mathcal{C})$ be the map from G to the contracted graph $G/\partial\mathcal{C}$. Because $\mathcal{F}$ conforms $\partial\mathcal{C}$, every cluster C in $\mathcal{F}$ corresponds to a cluster in $G/\partial\mathcal{C}$, denoted by $\eta(C)$, whose preimage is C. (That is, every vertex c with $\eta(c) \in \eta(C)$ satisfies $c \in C$.) Because u and v are cut by $\mathcal{F}$, the corresponding vertices $\eta(u)$ and $\eta(v)$ are also cut by the clusters corresponding to $\mathcal{F}$. As there is an edge between $\eta(u)$ and $\eta(v)$, they can be cut only if there is some cluster $\eta(C_u)$ from $\mathcal{F}$ containing either $\eta(u)$ or $\eta(v)$ (without loss of generality, $\eta(u)$). Thus, C_u contains u.

The procedure Embed selects an arbitrary terminal u' from $C_u \cap T$ in Step 4, and adds an edge to $\hat{G}$ between u and u', with weight $\delta_G(u', u)$ in Step 5. It also recursively calls $\textsc{Embed}_{\mathbb{C}}(T_H, P, \partial T_H, \partial\mathcal{C}_H)$ to get output $(\hat{G}_H, \hat{T}_H)$. By Observation 4.2, the pair (u', v) is a pseudo-edge with respect to $\partial\mathcal{C}_H$ which contains C_u by definition of $\partial\mathcal{C}_H$ in Step 4 of $\textsc{Embed}_{\mathbb{C}}(T, P, \partial T, \partial\mathcal{C})$. Further, T_H contains both u' and v, and by Lemma 4.2 $\phi(T_H, P, \partial T_H) \le \phi(T, P, \partial T) - 1$. Finally, Observation 4.3 implies that $\mathbb{C}$ splits (u', v) at scale i. Thus, we can apply induction to conclude that, in the output graph $\hat{G}_H$, we have

$$\mathbb{E}[\delta_{\hat{G}_H}(u', v)] \le \delta_G(u', v) + \alpha \cdot \frac{2^i \cdot (\phi(T, P, \partial T) - 1)}{\psi}.$$

We now split into subcases based on C_u.

- *Case* $\xi_3[\varnothing]$: The cluster C_u satisfies $|C_u \cap T| = 1$.

In Case $\xi_3[\varnothing]$, we have $u = u'$. Thus, in this case $\mathbb{E}[\delta_{\hat{G}_H}(u', v) \mid \text{Case } \xi_3[\varnothing] \text{ occurs}] = \delta_G(u, v) + \alpha \cdot \frac{2^i \cdot (\phi(T,P,\partial T)-1)}{\psi}$, and so

$$\mathbb{E}[D \mid \text{Case } \xi_3[\varnothing] \text{ occurs}] = 0.$$

- *Case $\xi_3[s]$* (see Figure 4(c)): The cluster C_u satisfies $|C_u \cap T| > 1$ and is at scale s. (Note that $s \le i$ since (u, v) is split at scale i.)

  In this case, cluster C_u has diameter 2^s. Thus, we have $\delta_G(u', u) \le 2^s$, and $\delta_G(u', v) \le \delta_G(u', u) + \delta_G(u, v) \le 2^s + \delta_G(u, v)$. We conclude:

$$\begin{aligned}\mathbb{E}[\delta_{\hat{G}}(u, v) \mid \text{Case } \xi_3[s] \text{ occurs}] &\le \delta_G(u, u') + \mathbb{E}[\delta_{\hat{G}_H}(u', v)] \\ &\le \delta_G(u, v) + 2^{s+1} + \alpha \cdot \frac{2^i \cdot (\phi(T, P, \partial T) - 1)}{\psi}\end{aligned}$$

  In other words,

$$\mathbb{E}[D \mid \text{Case } \xi_3[s] \text{ occurs}] \le 2^{s+1}.$$

The expected value $\mathbb{E}[\delta_{\hat{G}}(u, v)]$ is a weighted sum of these cases. We have

$$\begin{aligned}\mathbb{E}[\delta_{\hat{G}}(u, v)] &= \sum_{j=1}^{3} \mathbb{E}[\delta_{\hat{G}}(u, v) \mid \text{Case } \xi_j \text{ occurs}] \cdot \Pr[\text{Case } \xi_j \text{ occurs}] \\ &= \delta_G(u, v) + \alpha \cdot \frac{2^i \cdot (\phi(T, P, \partial T) - 1)}{\psi} + \sum_{j=1}^{3} \mathbb{E}[D \mid \text{Case } \xi_j \text{ occurs}] \cdot \Pr[\text{Case } \xi_j \text{ occurs}] \\ &= \delta_G(u, v) + \alpha \cdot \frac{2^i \cdot (\phi(T, P, \partial T) - 1)}{\psi} + \sum_{s=1}^{i} \mathbb{E}[D \mid \text{Case } \xi_3[s] \text{ occurs}] \cdot \Pr[\text{Case } \xi_3[s] \text{ occurs}] \\ &\le \delta_G(u, v) + \alpha \cdot \frac{2^i \cdot (\phi(T, P, \partial T) - 1)}{\psi} + \sum_{s=1}^{i} 2^{s+1} \cdot \Pr[\text{Case } \xi_3[s] \text{ occurs}]\end{aligned}$$

where the second-to-last line follows because $\mathbb{E}[D]$ is only nonzero in case $\xi_3[s]$. (Notice that the sum involves i terms, because by assumption (u, v) is not cut by any clusters with scale larger than i.) Now, observe that for each scale s, there are at most two clusters in $\mathbb{C}$ at scale s that contain u or v. (This is because each clustering in $\mathbb{C}$ is a partition of vertices.) If Case $\xi_3[s]$ occurs, then one of these two clusters was chosen. Since $\mathcal{F}$ is sampled from a (τ, ψ)-stochastic balanced cut, C_u is chosen with probability at most $1/\psi$. (This is by [Low probability] property in Definition 2.3; one can readily check that if some cluster C_u is chosen in Case $\xi_3[s]$, then C_u is not a singleton in $\mathbb{C}$, nor is it in $\partial\mathcal{C}$ (as by definition every cluster in $\partial\mathcal{C}$ contains at most 1 vertex in T.) Thus, for any s, Case $\xi_3[s]$ occurs with probability at most $2/\psi$. We have:

$$\begin{aligned}\mathbb{E}[\delta_{\hat{G}}(u, v)] &\le \delta_G(u, v) + \alpha \cdot \frac{2^i \cdot (\phi(T, P, \partial T) - 1)}{\psi} + \sum_{s=1}^{i} 2^{s+1} \cdot \frac{2}{\psi} \\ &\le \delta_G(u, v) + \alpha \cdot \frac{2^i \cdot (\phi(T, P, \partial T) - 1)}{\psi} + 8 \cdot \frac{2^i}{\psi} \\ &\le \delta_G(u, v) + \alpha \cdot \frac{2^i \cdot \phi(T, P, \partial T)}{\psi}\end{aligned}$$

where the last line follows if α is sufficiently large. (Choosing $\alpha = 8$ suffices.) ∎

5 Stochastic Balanced Cuts: Proof of Theorem 2.2

In this section, we prove Theorem 2.2. A key concept in the construction of a stochastic balanced cut is the notion of (a, b, c)-*contraction sequence* introduced by [CLPP23]. Roughly speaking, it represents a contraction of a graph G into a single vertex through b rounds, where in each round, we contract a certain number of vertex-disjoint subgraphs of radius at most c; the total number of subgraphs we contract in b rounds is a. See Figure 5(a).

DEFINITION 5.1. ((a,b,c)-CONTRACTION SEQUENCE) *For a connected unweighted graph G and integers $a,b,c \geq 0$, an (a,b,c)-contraction sequence consists of:*

- *a sequence of graphs $G_0, G_1, \ldots, G_b$, and*
- *for every $i \in [1\,..\,b]$, a collection of pairwise disjoint subgraphs $H_i^1, \ldots, H_i^{a_i}$ of G_{i-1},*

such that the following conditions hold:

- *$G_0 = G$ and G_b is a one-vertex graph.*
- *Each H_i^j is of radius at most c.*
- *G_i is obtained from G_{i-1} by contracting each of the subgraphs H_i^j, $j \in [1\,..\,a_i]$, to a single vertex.*
- *It holds that $\sum_{i=1}^{b} a_i \leq a$.*

[CLPP23] showed that if an apex-minor-free graph G admits an (a,b,c)-contraction sequence, then $\mathrm{tw}(G) = O(\sqrt{abc})$. Their basic idea is to argue that there exists a set S of a vertices such that all other vertices are within hop distance $O(bc)$ from the set, and then apply Lemma 3.1. The set S is obtained by picking one vertex per contracted subgraph. Here, we improve the treewidth bound by a factor of $\sqrt{b}$ to $O(\sqrt{abc})$, which is best possible; see Remark 5.1. First, by a simple reduction, we can assume that $c = 1$. Second, which is also the bulk of the technical details, we show the following structural property: Let Z be any maximal set of vertices in G such that their pairwise (hop) distance in G is $\Omega(b)$[4], then $|Z| \leq O(\frac{a}{b})$. By Lemma 3.1, we can conclude that $\mathrm{tw}(G) = O(\sqrt{Z}b) = O(\sqrt{ab})$ (for the case $c = 1$), which ultimately leads to our improved bound.

LEMMA 5.1. *For every apex graph K, there exists a constant β_K such that if a K-minor-free graph G admits an (a,b,c)-contraction sequence for some integers $a,b,c \geq 1$, then $\mathrm{tw}(G) \leq \beta_K\sqrt{abc}$.*

Proof. Reduction to the case when $c = 1$. Observe first that if a graph H is of radius at most c, then it admits a $(c,c,1)$-contraction sequence: if $v \in V(H)$ is such that every vertex of H is within distance at most c from v, then iteratively contracting all neighbors of v onto v for c rounds turns H into a one-vertex graph. Consequently, if G admits a (a,b,c)-contraction sequence, then it also admits a $(ac,bc,1)$-contraction sequence: replace every contraction of a graph H_i^j of radius at most c with at most c contractions of radius 1. Since $\beta_K\sqrt{abc} = \beta_K \cdot 1 \cdot \sqrt{ac \cdot bc}$, it suffices to prove the lemma for the case $c = 1$. That is, in the remainder of the proof we assume that a K-minor-free graph G admits a $(a,b,1)$-contraction sequence for some $a,b \geq 1$ and we prove that the treewidth of G is bounded by $\beta_K\sqrt{ab}$ for some constant β_K depending only on K.

Let $G_0, G_1, \ldots, G_b$, $a_1, \ldots, a_b$, and H_i^j for $i \in [1\,..\,b]$ and $j \in [1\,..\,a_i]$ be as in the definition of an $(a,b,1)$-contraction sequence for G, where $G = G_0$ and $V(G_b) = \{r\}$. For $i \in [1\,..\,b]$ and $j \in [1\,..\,a_i]$, let $x_i^j \in V(G_i)$ be the vertex of G_i that is the result of the contraction of H_i^j. Without loss of generality, we can assume that every graph H_i^j has at least two vertices, and that $a = \sum_{i=1}^{b} a_i$ and $a_i > 0$ for every $i \in [1\,..\,b]$ (in particular, $a \geq b$). We view the contraction sequence as a rooted tree T; see Figure 5(b). That is, we set $V(T) = \bigcup_{i=0}^{b} \{i\} \times V(G_i)$, indicate (b,r) as the root of T, and for every $i \in [1\,..\,b]$ and $v \in V(G_{i-1})$:

- If there exists $j \in [1\,..\,a_i]$ such that $v \in V(H_i^j)$, then the parent of $(i-1,v)$ is (i,x_i^j).
- Otherwise, the parent of $(i-1,v)$ is (i,v).

A node of T is *branching* if it has at least two children. Since every H_i^j has at least two vertices, a node (i,v) is branching if and only if there exists $j \in [1\,..\,a_i]$ such that $v = x_i^j$. Consequently, there are exactly a branching nodes of T. The *weight* of a node (i,v) of T, denoted $\mathrm{weight}(i,v)$, is the number of branching vertices in the subtree of T rooted at (i,v) (including possibly the node (i,v) itself). A node is *heavy* if it is of weight at least b and *light* otherwise. Note that all leaves of T are light (as they are of weight 0) while the root is heavy (as it is of weight a, which by assumption is at least b).

[4] known as an *r-net* with $r = \Omega(b)$

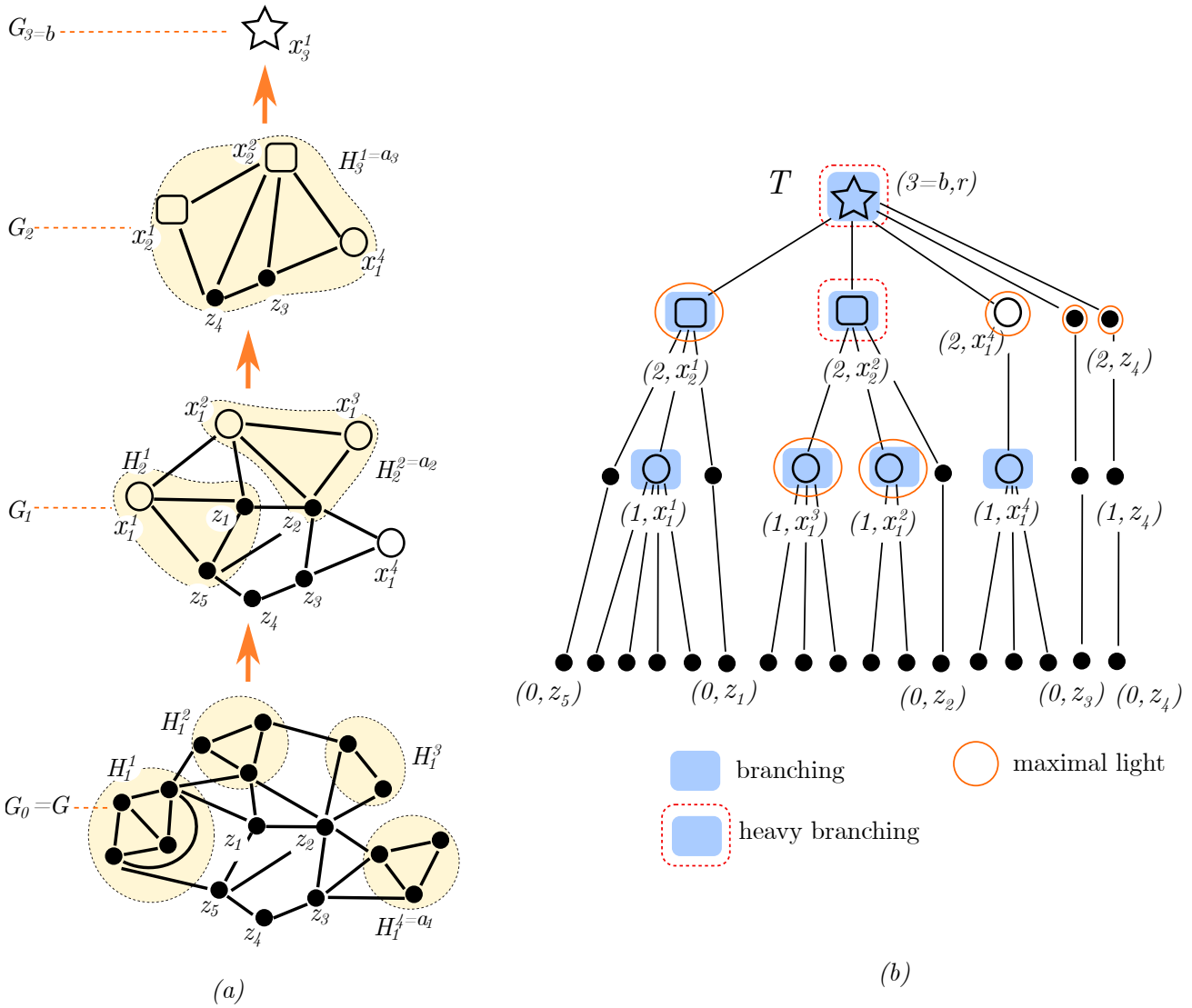


Figure 5: (a) An (a, b, c) contraction sequence with $a = 7, b = 3, c = 1$. (b) A tree T associated with the contraction sequence in the proof of Lemma 5.1. M contains exactly all the nodes at level 2; $(2, x_2^2)$ is minimally heavy and the only node in M_1; the rest of nodes at level 2 are in M_2.

We say that a node is *maximal light* if it is light, but its parent is heavy, and *minimal heavy* if it is heavy, but all its children are light. Note that all children of a minimal heavy node are maximal light, but a parent of a maximal light node is always heavy, but may not be minimal heavy. Let M_1 be the family of all minimal heavy nodes and M_2 be the family of those maximal light nodes, whose parents are not minimal heavy. Let $M := M_1 \cup M_2$. We observe the following.

(5.5) On every root-to-leaf path in T there is exactly one node of M.

Indeed, since all leaves of T are light while the root is heavy, on every root-to-leaf path in T there is exactly light node (i, v) with a heavy parent $(i + 1, w)$. Then either $(i + 1, w)$ is minimal heavy and is in M_1 (but then (i, v) is not in M_2) or $(i + 1, w)$ is not minimal heavy (but then (i, v) is in M_2). This proves (5.5).

We infer that the sum of weights of elements of M is at most a. This in particular implies that

(5.6) $$|M_1| \leq a/b.$$

Furthermore, note that every node of M_1 is branching, as the weight of a non-branching non-leaf node of T is equal to the weight of its only child.

For a node (i,v) of T, let $G[(i,v)]$ be the subgraph of G induced by those vertices w where $(0,w)$ is a descendant of (i,v) in T. In other (less formal) words, $G[(i,v)]$ is the subgraph that is contracted onto v in G_i in the contraction process. Note that (5.5) implies that $\{V(G[(i,v)]) \mid (i,v) \in M\}$ is a partition of V. We observe the following.

$$\text{If } (i,v) \in V(T) \text{ is of weight } p, \text{ then } \operatorname{diam}(G[(i,v)]) \leq 2p. \tag{5.7}$$

Indeed, to see (5.7), observe that a contraction of a subgraph of radius 1 in a connected graph can decrease the diameter by at most 2. By induction, if a graph H can be turned into a one-vertex graph by a series of p contractions of subgraphs of radius 1, then the diameter of H is bounded by $2p$. This proves (5.7).

We infer that for every light (i,v), the diameter of $G[(i,v)]$ is bounded by $2(b-1)$. We now show a bound on the diameter of $G[(i,v)]$ for minimal heavy (i,v).

$$\text{For every } (i,v) \in M_1, \operatorname{diam}(G[(i,v)]) \leq 6b-1. \tag{5.8}$$

Let (i,v) be a minimal heavy node. Recall that (i,v) is branching, and thus $v = x_i^j$ for some $j \in [1\,..\,a_i]$. Hence, the children of (i,v) in T are $(i-1,w)$ for $w \in V(H_i^j)$. Let $w_0 \in H_i^j$ be such that every vertex of H_i^j is w_0 or a neighbor of w_0. As $(i-1,w)$ is light for $w \in V(H_i^j)$, the diameter of $G[(i-1,w)]$ is at most $2(b-1)$. We infer that the diameter of $G[(i,v)]$ is bounded by $3 \cdot 2(b-1) + 2 = 6b-1$. This proves (5.8).

Let G/M be the graph obtained from G by contracting, for every $(i,v) \in M$, the graph $G[(i,v)]$ into a single vertex. We identify $V(G/M)$ with the set M. (Recall that $G/\mathcal{X}$ denote the graph obtained by contracting every cluster in $\mathcal{X}$ into a vertex; here we contract clusters associated with all $(i,v) \in M$.) The crucial observation is the following.

$$\text{For every } (i,v) \in M \text{ there exists } (i',v') \in M_1 \text{ within distance } b \text{ in } G/M. \tag{5.9}$$

Indeed, by the definition of M and the original $(a,b,1)$ contraction sequence, one can turn G/M into a one-vertex graph by iteratively, in b rounds, contracting some neighborhoods of elements of M_1. This proves (5.9).

We fix a threshold $\kappa := 9b$. For a vertex $x \in V$ and a node (i,v) of T, we say that (i,v) is *close* to x if every vertex of $G[(i,v)]$ is within distance at most κ from x in G. Let $\operatorname{Close}(x)$ be the set of nodes close to x. We now prove the following critical property.

$$\text{For every vertex } x, \sum_{(i,v) \in M \cap \operatorname{Close}(x)} \operatorname{weight}(i,v) \geq b. \tag{5.10}$$

To see (5.10), first note that if $(i,v) \in M_1$ is close to x, then already $\operatorname{weight}(i,v)$ contributes at least b to the sum of (5.10). In the remaining case, we observe that, by (5.8) and the fact that $\{V(G[(i,v)]) \mid (i,v) \in M\}$ is a partition of V, every vertex within distance at most $\kappa - 6b$ of x belongs to $G[(i,v)]$ for some $(i,v) \in M_2$.

Let $(i_0,v_0) \in M$ be such that $x \in V(G[(i_0,v_0)])$ and let Q be a shortest path (in hops) in G/M from (i_0,v_0) to a node of M that is not close to x. By (5.9), the length of Q is $\ell \leq b$, as every node of M_1 is not close to x. Let $(i_0,v_0),(i_1,v_1),\ldots,(i_\ell,v_\ell)$ be consecutive vertices of Q where $(i_\ell,v_\ell) \in M_1$ and $(i_j,v_j) \in M_2$ for $j \in [0\,..\,\ell-1]$. By the definition of G/M, there is a path P from x to a vertex of $G[(i_\ell,v_\ell)]$ that goes via $G[(i_0,v_0)]$, $G[(i_1,v_1)]$, $\ldots$, $G[(i_{\ell-1},v_{\ell-1})]$. See Figure 6. By (5.7), the length of P is bounded by

$$\sum_{j=0}^{\ell-1} 1 + 2\operatorname{weight}(i_j,v_j) \leq b + 2\sum_{j=0}^{\ell-1} \operatorname{weight}(i_j,v_j).$$

On the other hand, the definition of close, together with (5.7) and (5.8) ensure that the length of P is at least $\kappa - 6b$. We infer that

$$\sum_{j=0}^{\ell-1} \operatorname{weight}(i_j,v_j) \geq b.$$

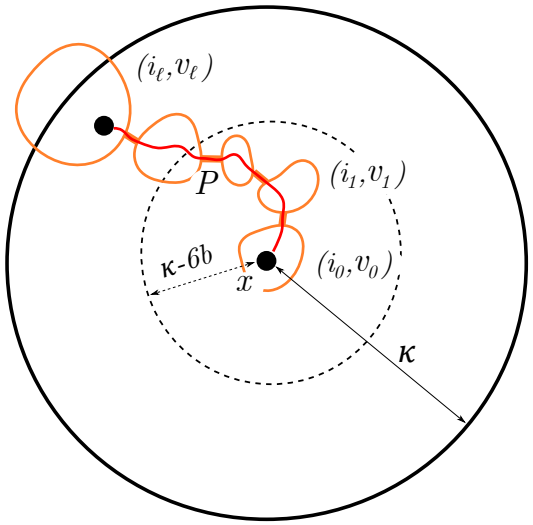


Figure 6: Path P from x to a vertex in $G[(i_\ell, v_\ell)]$.

As every node (i_j, v_j) for $j \in [0\,..\,\ell-1]$ is close to x, this finishes the proof of (5.10).

Let Z be a maximal family of vertices of G within pairwise distance more than $2\kappa = 18b$. Since the total weight of all nodes of M is at most a, we infer from (5.9) that $|Z| \leq a/b$. By the definition of Z, every node of G is within distance at most 2κ from an element of Z. Lemma 3.1 implies that the treewidth of G is bounded by

$$\alpha_K \cdot 2\kappa \cdot \sqrt{|Z|} \leq \alpha_K \cdot 18b \cdot \sqrt{\frac{a}{b}} \leq 18\alpha_K \sqrt{ab}.$$

This finishes the proof of Lemma 5.1 with $\beta_K := 18\alpha_K$, where α_K comes from Lemma 3.1. ∎

Remark 5.1. *The bound of Lemma 5.1 is asymptotically optimal. Pick integers p, q with $2^q \gg p \gg q$ and consider a $pq \times pq$ grid. Let Z be a set of p^2 equidistributed vertices of the grid, so that every vertex of the grid is within distance $2q$ from a vertex of Z. Iteratively, for $2q$ rounds, contract the neighborhood of Z onto Z. This contracts the grid into a grid of sidelength p. Then, within $O(\log p)$ rounds, contract this grid onto a single vertex, by just taking every second vertex of the grid and contracting its neighborhood onto it. This gives a $(O(p^2(q + \log p)), O(q + \log p), 1)$-contraction sequence for the original grid of sidelength pq. If $q > \log p$, this is a matching lower bound to Lemma 5.1.*

We are now ready to prove Theorem 2.2.

Proof. [of Theorem 2.2] It suffices to show that $(G, \mathbb{C})$ has a (τ, ψ)-stochastic balanced cut with respect to $(\omega_V, \mathcal{X})$ since the only property of G that we will use in our construction is apex-minor-free. That means the same construction applies to construct a (τ, ψ)-stochastic balanced cut with respect to $(\omega_C, \mathcal{X})$ for $(G[C], \mathbb{C}_{\downarrow C})$ since $G[C]$ is also apex-minor-free. We will construct a set of balanced cut $\mathbb{F}$ such that:

(i) $|\mathbb{F}| = \psi$.

(ii) every cut $\mathcal{F} \in \mathbb{F}$ conforms $\mathcal{X}$ and contains at most τ clusters in $\mathbb{C}$.

(iii) every *non-singleton* cluster $C \in \mathbb{C} \setminus \mathcal{X}$ appears in (some cut in) $\mathbb{F}$ at most once.

Then, the distribution $\mathfrak{F}$ is simply a uniform distribution over cuts in $\mathbb{F}$. That is, one samples a cut $\mathcal{F}$ with probability $1/|\mathbb{F}| = 1/\psi$. As each non-singleton cluster $C \in \mathbb{C} \setminus \mathcal{X}$ appears in $\mathbb{F}$ at most once, the probability that C is contained in a sampled cut $\mathcal{F}$ is at most $1/\psi$. Thus, the existence of $\mathbb{F}$ implies Theorem 2.2.

Henceforth, we focus on constructing $\mathbb{F}$. The construction is the same as in [CLPP23], but uses Lemma 5.1 instead. We include the details tailored to our notation for completeness. Set the threshold for number of clusters

in any $\mathcal{F}$ to be $\tau := \beta' \cdot h^2 \log \Phi \cdot \psi$ for a constant β' chosen later. The algorithm is greedy: starting from $\mathbb{F} = \varnothing$, if $|\mathbb{F}| < \psi$, we will show below that we can add one more cut $\mathcal{F}$ of size at most τ to $\mathbb{F}$. (The size of a cut is the number of clusters within.) Thus, the algorithm will terminate when $\mathbb{F}$ has exactly ψ cuts, and every cut has a size at most τ.

Now, we construct a balanced cut $\mathcal{F}$ of size at most τ. Let $|\mathbb{F}|$ be the number of cuts currently in $\mathbb{F}$. Note that $|\mathbb{F}| < \psi$, implying that:

$$|\mathbb{F}| < \frac{\tau}{\beta' \cdot h^2 \log \Phi}. \tag{5.11}$$

View the clustering chain $\mathbb{C} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k)$ where $k = \lceil \log \Phi \rceil$ as a tree with the root corresponding to a single cluster V (in $\mathcal{C}_k$) with leaves being singletons in $\mathcal{C}_0$. We mark the root of $\mathbb{C}$, and all clusters in $\mathbb{F} \setminus \mathcal{X}$ as *unavailable*. All other clusters are marked *available*; these include (not only) clusters in $\mathcal{X}$ and singleton clusters in $\mathcal{C}_0$. We say that a cluster C is *maximally available* if C is available and its parent in $\mathbb{C}$ is unavailable.

Let $\mathcal{S}$ be the set of maximally available clusters. Observe that $\mathcal{S}$ is a partition V. Let $\check{G}$ be the graph obtained from G by contracting every cluster in $\mathcal{S}$ into a single vertex. For each vertex $\check{v} \in \check{G}$ corresponding to a cluster C_v, we assign a weight $\check{\omega}(\check{v}) := \sum_{u \in C_v} \omega(u)$. The key observation is that:

$$\check{G} \text{ admits a } (\tau \cdot |\mathbb{F}| + 1, k, h)\text{-contraction sequence.} \tag{5.12}$$

To see (5.12): Starting from level 1 (corresponding to $\mathcal{C}_1$), for each level $i \in [1 \,..\, k]$, contract all unavailable clusters at level i into a single vertex. (Think of each contracted unavailable cluster C at level i as the graph $G[C]/\mathcal{C}_{i-1}[C]$ to align with the definition of the contraction sequence.) Clearly, the number of contraction rounds is k, and the total number of contracted clusters is at most $\tau \cdot |\mathbb{F}| + 1$, which is an upper bound on the number of unavailable clusters. (The $+1$ is for including the root cluster, which is always unavailable.) Since the hop bound if $\mathbb{C}$ is h, the radius of each contracted graph is at most h.

By (5.12) and Lemma 5.1, there is a constant β that only depends on the size of the minor such that:

$$\begin{aligned}
\mathrm{tw}(\check{G}) &\le \beta \cdot \sqrt{(\tau \cdot |\mathbb{F}| + 1)k} \cdot h = \beta \cdot \sqrt{(\tau \cdot |\mathbb{F}| + 1) \log \Phi} \cdot h \\
&\le \beta \cdot \sqrt{2\tau \cdot |\mathbb{F}| \log \Phi} \cdot h \\
&\le \beta \cdot \sqrt{2\tau \cdot \log \Phi \cdot \frac{\tau}{\beta' \cdot h^2 \log \Phi}} \cdot h \qquad \text{(by Equation (5.11))} \\
&= \frac{\sqrt{2} \cdot \beta}{\sqrt{\beta'}} \cdot \tau \le \tau
\end{aligned}$$

by choosing a sufficiently large constant $\beta' \ge 2\beta^2$ (that only depends on the size of the minor). Thus, $\check{G}$ admits a balanced separator $\check{S}$ with respect to $\check{\omega}$ of size at most τ. Then $\mathcal{F} = \{C_v : \check{v} \in \check{S}\}$ is a balanced cut of size at most τ.

Clearly, $\mathcal{F}$ conforms $\mathcal{X}$ since clusters in $\mathcal{X}$ are always marked available, and clusters in $\mathcal{F}$ are maximally available clusters. Thus, property (ii) holds. For every cluster $C \in \mathcal{F}$, since C is available, then $C \in \mathcal{X}$ or C is a singleton, or C has not been added to any other cut in $\mathbb{F}$. Thus, property (iii) holds. ∎

6 Separating Distribution of Clustering Chains: Proof of Theorem 2.1

Recall the definition of a *buffered cop decomposition*, introduced in [CCL+24]. For any graph G, a *supernode* η with *skeleton* T_η and *radius* Δ is an induced subgraph of G containing a tree T_η where every vertex in η is within distance Δ of T_η, where distance is measured with respect to the induced shortest-path metric of η. A *buffered cop decomposition* for G is a partition of G into vertex-disjoint supernodes, together with a tree $\mathcal{T}$ called the *partition tree* whose nodes are the supernodes of G. For any supernode η, the *domain* $\mathrm{dom}(\eta)$ is the subgraph induced by the union of all vertices in supernodes in the subtree of $\mathcal{T}$ rooted at η.

Definition 6.1. *A (Δ, γ, w)-buffered cop decomposition for G is a buffered cop decomposition $\mathcal{T}$ for G that satisfies the following properties:*

- [Supernode radius.] *Each supernode η has radius at most Δ.*

- [Shortest-path skeleton.] *For every supernode η, the skeleton T_η is an SSSP tree in* $\mathrm{dom}(\eta)$ *with at most* $w-1$ *leaves (not counting the root). Further, let* $\mathcal{A}_\eta$ *denote the set of ancestor supernodes* η' *of* η *such that there is an edge between* η' *and* $\mathrm{dom}(\eta)$ *in* G*. Then* $|\mathcal{A}_\eta| \le w-1$*, and there is an edge between* T_η *and* η' *in* G *for each* $\eta' \in \mathcal{A}_\eta$*.*[5]

- [Supernode buffer.] *Let* η *be a supernode, and let* X *be another supernode that is an ancestor of* η*. Then either* η *and* X *are adjacent in* G*, or for every vertex* v *in* $\mathrm{dom}(\eta)$*, we have* $\delta_{\mathrm{dom}(X)}(v, X) \ge \gamma$*.*[6]

- [Tree decomposition.] *Define the expansion* $\hat{\mathcal{T}}$ *to be a tree isomorphic to* $\mathcal{T}$*, such that every supernode* η *in* $\mathcal{T}$ *corresponds to a node* B_η *(called the bag at* η*) that contains (all vertices in) the union of* η *and all supernodes in* $\mathcal{A}_\eta$*. The expansion* $\hat{\mathcal{T}}$ *is a tree decomposition for* G*, and every bag of* $\hat{\mathcal{T}}$ *is the union of at most* w *supernodes.*

We say that an edge e is *cut* by a buffered cop decomposition $\mathcal{T}$ if the endpoints of e belong to different supernodes in $\mathcal{T}$. The majority of this section is devoted to proving the following theorem.

THEOREM 6.1. *Let* G *be a* K_r*-minor-free graph, and let* Δ *be a positive number. There exist a constant* β *such that there is a distribution* $\mathbb{T}$ *of* $(4\Delta, \Delta/r, r-1)$*-buffered cop decompositions of* G*, such that for any edge* e*,* $\Pr_{\mathcal{T}\sim\mathbb{T}}[e \text{ is cut by } \mathcal{T}] \le \beta \cdot \|e\|/\Delta$*.*

[CCL+24] also construct a *shortcut partition* (a type of partition first introduced by [CCL+23] as a weaker version of the scattering partition of [Fil24b]). We recall the definition here.

DEFINITION 6.2. *An* (ε, h)*-shortcut partition of* G *is a clustering* $\mathcal{C} = \{C_1, \dots, C_m\}$ *of* G *such that:*

- [Diameter.] *the strong diameter of each cluster* C_i *is at most* $\varepsilon \cdot \mathrm{diam}(G)$*;*

- [Low-hop.] *for any two vertices* u *and* v *in* G*, there is a path* $\check{\pi}$ *in the graph* $G/\mathcal{C}$ *(obtained by contracting every cluster* C_i *into a single vertex) between the two clusters containing* u *and* v*, such that (1)* $\check{\pi}$ *has hop length at most* $\varepsilon h \cdot \lceil \frac{\delta_G(u,v)}{\varepsilon \cdot \mathrm{diam}(G)} \rceil$*, and (2) there is a shortest path* π *in* G *between* u *and* v*, such that every cluster on* $\check{\pi}$ *has nontrivial intersection with* π*.*

We remark that $\mathcal{C}$ is an (ε, h)-shortcut partition, then the graph $G/\mathcal{C}$ has hop-diameter at most $h+1$ (see the proof of Theorem 2.1 below). [CCL+24] used buffered cop decomposition to construct shortcut partition for minor-free graphs. In particular, they showed that for any $\varepsilon < 1$, K_r-minor-free graphs admit $(\varepsilon, O_r(1/\varepsilon))$-shortcut partition. We give a random version of their algorithm, using Theorem 6.1 as a subroutine, to prove the following lemma.

LEMMA 6.1. *Let* G *be a* K_r*-minor-free graph, and let* ε *be some fixed number in* $(0, 1)$*. There exist constants* $h, \beta = O_r(1)$ *such that there is a distribution* $\mathbb{C}$ *of* (ε, h)*-shortcut partitions of* G *such that for each edge* e*,* $\Pr_{\mathcal{C}\sim\mathbb{C}}[e \text{ is cut by } \mathcal{C}] \le \beta \cdot \frac{\|e\|}{\varepsilon \cdot \mathrm{diam}(G)}$*.*

Below, we prove Theorem 2.1 from Lemma 6.1. In Section 6.1, we give a randomized construction for buffered cop decomposition (and prove that the output is in fact a buffered cop decomposition). In Section 6.2, we upper-bound the probability an edge e is cut by the buffered cop decomposition; the proof of a crucial lemma in this analysis is deferred to Section 6.3. This completes the proof of Theorem 6.1. In Section 6.5, we give a randomized construction of shortcut partition, proving Lemma 6.1.

Proof. [of Theorem 2.1] We will build the β-separating distribution $\mathfrak{C}$ of clustering chains level-by-level, from top to bottom. Start with the original graph G and level $i \leftarrow \lceil \log(\mathrm{diam}(G)) \rceil$. At each level i we deal with a graph H_i with diameter upper bounded by $\Delta_i := 2^i$. Set $\varepsilon = \frac{1}{2}$, and sample a shortcut partition $\mathcal{C}_i$ from the distribution of (ε, h)-shortcut partitions for G promised by Lemma 6.1, where h is some constant depending on the size of the minor excluded by G. The decomposition $\mathcal{C}_i$ partitions H_i into vertex-disjoint clusters each of

[5] The second half of this property was not explicitly stated by [CCL+24]; rather, it was stated by [Fil24a], who observed that the construction of [CCL+24] also satisfies this property.

[6] If A and B are two sets, we use $\delta(A, B)$ to denote $\min_{a\in A, b\in B} \delta(a, b)$. When A is a singleton set $\{a\}$ we write $\delta(a, B)$ instead.

diameter $\operatorname{diam}(H_i)/2 \le 2^{i-1}$. Further, the graph $H_i/\mathcal{C}_i$ has hop-diameter at most h; indeed, any two vertices u and v in H_i satisfy $\delta_{H_i}(u,v) \le \operatorname{diam}(H_i)$, so the [low-hop] property of shortcut partition guarantees that the cluster containing u and v are connected in $H_i/\mathcal{C}_i$ by a path of hop-length at most $\varepsilon h \cdot \lceil \frac{\operatorname{diam}(G)}{\varepsilon \cdot \operatorname{diam}(G)} \rceil \le \varepsilon h \cdot \lceil 1/\varepsilon \rceil \le h$. We recursively build a β-separating distribution of clustering chains for each of the clusters, now at level $i-1$, until we reach level 0 where every cluster is a single vertex.

The resulting distribution $\mathfrak{C}$ of clustering chains clearly satisfies all properties of clustering chains, and each clustering chain in the support has hop bound $h = O(1)$. To demonstrate that the distribution $\mathfrak{C}$ is β-separating for some $\beta = O(1)$, we consider an arbitrary edge e and prove $\Pr_{\mathbb{C}\sim\mathfrak{C}}[e \text{ is cut by } \mathcal{C}_i \in \mathbb{C}] \le \beta \cdot \|e\|/2^i$ by induction on i, starting with the highest level ($i = \lceil \log \operatorname{diam}(G) \rceil$). Indeed, if e is cut by the the i-th level of the clustering chain $\mathcal{C}_i$, then either (1) e was cut by $\mathcal{C}_{i+1}$, or (2) e is contained in some cluster $C \in \mathcal{C}_{i+1}$ but e is cut by the shortcut partition of C at level i. Case (1) occurs with probability at most $\beta \cdot \|e\|/2^{i+1}$, by induction. Case (2) occurs with probability at most $\beta' \cdot \|e\|/(\varepsilon \cdot 2^{i+1}) = \beta' \cdot \|e\|/2^i$ for some $\beta' = O(1)$ promised by Lemma 6.1. By choosing $\beta := 2\beta'$ and applying a union bound, we conclude e is cut with probability at most $\beta \cdot \|e\|/2^i$. ∎

6.1 Construction of stochastic buffered cop decomposition We slightly modify the construction of buffered cop decomposition from [CCL+24]. Before giving the full algorithm, we recall some terminology. Throughout the algorithm, we maintain a global variable $\mathcal{S}$, a set of supernodes (which changes over the course of the algorithm). We say a subgraph H *sees* a supernode X in $\mathcal{S}$ if (1) X is disjoint from H and (2) there exists some *witness vertex* v_X in H that is adjacent (in G) to some vertex in X. For any subgraph H, let $\mathcal{S}_{|H}$ denote the set of supernodes in $\mathcal{S}$ that H sees.

Every supernode η in $\mathcal{S}$ is associated with a subgraph of G called the *initial domain* and denoted $\operatorname{dom}_0(\eta)$, as well as a set of supernodes in $\mathcal{S}$ that $\operatorname{dom}_0(\eta)$ sees, denoted $\mathcal{S}_{|\eta}$. We would like to allow the supernodes in $\mathcal{S}_{|\eta}$ *as induced subgraphs* to grow over the course of the algorithm as $\mathcal{S}$ changes, but the *collection of* supernodes will be fixed even when $\operatorname{dom}_0(\eta)$ later intersects some (grown) supernode X in $\mathcal{S}_{|\eta}$ and thus technically no longer "sees" X because of the disjointness requirement is violated. At any point in the algorithm, with supernode assignment $\mathcal{S}$, the subgraph $\operatorname{dom}_{\mathcal{S}}(\eta)$ is defined to be $\operatorname{dom}_0(\eta) \setminus \bigcup \mathcal{S}_{|\eta}$.[7] It will hold that at the end of the algorithm, $\operatorname{dom}_{\mathcal{S}}(\eta) = \operatorname{dom}(\eta)$.

Our new algorithm is described in Figure 7 below; the lines that differ from [CCL+24] are in red. The parameters Δ and r in the algorithm are fixed throughout execution of the algorithm. For intuition behind the algorithm, refer to [CCL+24]. To construct a buffered cop decomposition for a graph G, we run BuildTree$(\varnothing, G)$. If a vertex v is assigned to a supernode X' in Step 2 of a call $C :=$ GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$, then we say that v is *assigned during* C, and X' is *expanded during* C. If the C selects a supernode $X \in \mathcal{X}$ in Step 1, we say that C *processes* X.

To summarize, we have three modifications to the algorithm of [CCL+24]:

1. When initializing a supernode η in Step 1 of BuildTree, we immediately grow it by some random radius in $[0, \Delta/r]$. ([CCL+24] omits this initial growth.)
2. When growing a buffer ($\mathcal{N}H_X$) around old supernodes in Step 1 of GrowBuffer, we choose the buffer to have a random radius in $[\Delta/r, 2\Delta/r]$. (In [CCL+24], the buffer had fixed radius Δ/r.)
3. When assigning each vertex v in the buffer to an existing supernode, in Step 2 of GrowBuffer, we assign v to the closest existing supernode after doing some small random perturbation. (In [CCL+24], v was assigned deterministically to the closest existing supernode.)

[CCL+24] show that, if graph G excludes a K_r-minor, then their procedure outputs a $(\Delta, \Delta/r, r-1)$-buffered cop decomposition. Their proof carries over almost verbatim for our stochastic algorithm, with slightly worse constants. We sketch the differences below.

Lemma 6.2. *If G is a graph excluding a K_r-minor, then for any $\Delta > 0$, the stochastic procedure* BuildTree$(\varnothing, G)$ *outputs a tree $\mathcal{T}$ that is a $(4\Delta, \Delta/r, r-1)$-buffered cop decomposition.*

[7] We remark that our definition of $\operatorname{dom}_{\mathcal{S}}(X)$ makes specific of the definition given in [CCL+24]; their definition is somewhat ambiguous. All the proofs of [CCL+24] also work with our clarified definition of $\operatorname{dom}_{\mathcal{S}}(\eta)$.

BuildTree$(\mathcal{S}, H)$**:** Input a set of supernodes $\mathcal{S}$, and a subgraph H of G consisting of vertices unassigned to a supernode in $\mathcal{S}$. Returns a tree of supernodes.

1. *Initialize a new supernode η.*

   Let v be an arbitrary vertex in H. Let T_η be an SSSP tree in H, connecting v to a witness vertex for every supernode in $\mathcal{S}_{|H}$. Initialize $\eta \coloneqq T_\eta$ to be a new supernode with skeleton T_η, and add η to $\mathcal{S}$. Let $\alpha \sim \text{Unif}[0,1]$. For every vertex v' satisfying $\delta_H(v', T_\eta) \le \alpha \cdot \Delta/r$, assign v' to supernode η. Initialize tree $\mathcal{T}$ with root η.

2. *Assign vertices to existing supernodes, to guarantee the supernode buffer property.*

   For each connected component H' of $H \setminus \eta$: let $\mathcal{X}$ be a list of every supernode seen by H but not H', and call GrowBuffer$(\mathcal{S}, \mathcal{X}, H')$. (The GrowBuffer procedure modifies the global variable $\mathcal{S}$, assigning some vertices in H' to supernodes in $\mathcal{S}$).

3. *Recurse.*

   For each connected component H' of $H \setminus \bigcup \mathcal{S}$: let $\mathcal{T}'$ be the output of BuildTree$(\mathcal{S}, H')$, and attach $\mathcal{T}'$ as a child to the root of $\mathcal{T}$.

GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$**:** Input a set of supernodes $\mathcal{S}$, a subgraph H of G consisting of vertices unassigned to any supernode in $\mathcal{S}$, and a list of supernodes $\mathcal{X}$ in $\mathcal{S}$ that are not seen by H. This procedure modifies the global variable $\mathcal{S}$ by assigning some vertices in H to $\mathcal{S}$.

1. *Grow a buffer around some supernode in $\mathcal{X}$.*

   Let X be an arbitrary supernode in $\mathcal{X}$. (If $\mathcal{X}$ is empty, do nothing and return.) Let $\partial H_{\downarrow X}$ be the set of vertices in $G \setminus H$ that are (1) adjacent to H, and (2) in $\text{dom}_{\mathcal{S}}(X)$. Let $\alpha \sim \text{Unif}[1,2]$, and let $\mathcal{N}H_X$ be the set of vertices in H such that $\delta_{\text{dom}_{\mathcal{S}}(X)}(v, \partial H_{\downarrow X}) \le \alpha \cdot \Delta/r$

2. *Assign the vertices in the buffer to existing supernodes.*

   For each supernode X' seen by H, choose a random value $\alpha_{X'} \sim \text{Unif}[0, \Delta/r]$. For each vertex v in $\mathcal{N}H_X$, assign v to the supernode X' that minimizes $\delta_{\text{dom}_{\mathcal{S}}(X)}(v, X' \cap \partial H_{\downarrow X}) + \alpha_{X'}$. Update $\mathcal{S}$ with these assignments. (Note that this changes the supernode X', and may also $\text{dom}_{\mathcal{S}}(\eta)$ for any supernodes η initialized after X'.)

3. *Update $\mathcal{X}$ to include any newly "cut-off" supernodes, and recurse.*

   For each connected component H' of $H \setminus \bigcup \mathcal{S}$: initialize $\mathcal{X}'$ to be $\mathcal{X} \setminus \{X\}$, then add to $\mathcal{X}'$ all the supernodes in $\mathcal{S}$ that are seen by H but not H', and then call GrowBuffer$(\mathcal{S}, \mathcal{X}', H')$.

Figure 7: The BuildTree and GrowBuffer algorithms.

Proof. The claims and proofs in Section 3.2 of [CCL+24] ("Basic Properties") carry over nearly verbatim; $\mathcal{T}$ satisfies the [shortest-path skeleton] property and the [tree decomposition] property. The only change is in the proof that every supernode is a connected subgraph (Claim 3.4(1)), where we must replace references to "closest supernode in $\partial H_{\downarrow X}$ to v" with "the supernode X' minimizing $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, X' \cap \partial H_{\downarrow X}) + \alpha_{X'}$". The other proofs of Section 3.2 only rely on the definition of the skeleton T_η in BuildTree, and on the fact that calls to BuildTree$(\cdot, H)$ or GrowBuffer$(\cdot, \cdot, H)$ are made only on subgraphs H that are maximal connected components of unassigned vertices.

The [CCL+24] proof of the [supernode buffer] property relies on the fact that, during any call to GrowBuffer, the set $\mathcal{N}H_X$ includes every vertex within distance Δ/r of $\partial H_{\downarrow X}$. As Step 2 of our modified GrowBuffer chooses $\alpha \geq 1$, we also have this property. The rest of their proof carries over verbatim.

To adapt the [CCL+24] proof of the [supernode radius] property, we need to make two changes. First, in the [CCL+24] algorithm, each supernode η was initialized with radius 0, and expands only when GrowBuffer is called. In our modified algorithm, supernode η is initialized with radius up to Δ/r; thus, our final radius bound is weaker by $+\Delta/r$ than the bound of [CCL+24]. Secondly, we need to modify Claim 3.11 of [CCL+24], which states that each time a supernode η is expanded by some call to GrowBuffer, the radius of η increases by at most Δ/r. Because of our modifications in Step 1 and 2 of GrowBuffer, we instead prove that the radius increases by at most $3\Delta/r$; thus, our final radius bound is weaker by a factor 3 than the bound of [CCL+24].

> Suppose that v is assigned to a supernode η during a call $C :=$ GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$. Let X denote the supernode processed during C, and let $\partial H_{\downarrow X}$ denote the boundary vertices. Let $\tilde{v}$ be the closest vertex in $\partial H_{\downarrow X} \cap \eta$ to v (with respect to $\mathrm{dom}_{\mathcal{S}}(X)$). Then $\delta_\eta(v, \tilde{v}) \leq 3\Delta/r$ (with respect to the final η).

To prove the claim, let $\mathcal{N}H_X$ denote the set of points assigned during C. Let P be a shortest path between v and $\tilde{v}$ in $\mathrm{dom}_{\mathcal{S}}(X)$. Every vertex in P (other than $\tilde{v}$) is in $\mathcal{N}H_X$. Because we assign every vertex v' in $\mathcal{N}H_X$ to the supernode X' minimizing $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v', X' \cap \partial H_{\downarrow X}) + \alpha_{X'}$, every vertex in P is assigned to η. Furthermore, we claim P has length at most $3\Delta/r$. Indeed, v is within distance $2\Delta/r$ of $\partial H_{\downarrow X}$ (as $\alpha \leq 2$ in Step 1 of GrowBuffer), so there is some supernode X' with $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, X' \cap \partial H_{\downarrow X}) \leq 2\Delta/r$. As $\alpha_{X'} \leq \Delta/r$, we have $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, X') + \alpha_{X'} \leq 3\Delta/r$. By choice of η, $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, \eta) + \alpha_\eta \leq 3\Delta/r$, and thus $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, \eta) \leq 3\Delta/r$. We conclude that $\|P\| \leq 3\Delta/r$, and so $\delta_\eta(v, \tilde{v}) \leq 3\Delta/r$.

The rest of the [supernode radius] proof from [CCL+24] applies for our modified algorithm. Because of the two modifications described above, we end up with a radius bound of $3\Delta + \Delta/r \leq 4\Delta$. This concludes the proof that $\mathcal{T}$ is a $(4\Delta, \Delta/r, r-1)$-buffered cop decomposition. ∎

6.2 Bounding the cut probability Let $e = (u, v)$ be an edge in G. Let C_i be a random variable that denotes the i-th call, whether it is to BuildTree or GrowBuffer, that is made during the execution of the algorithm. For any i, we define three events below. Observe that if u and v are in different supernodes, then one of these events occurs.

- ξ^i_{build}: call C_i is a call BuildTree$(\mathcal{S}, H)$, such that both $u, v \in H$, and exactly one of u and v is assigned to a supernode during C.
- ξ^i_{buffer}: call C_i is a call GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$, such that both $u, v \in H$, and exactly one of u and v is in $\mathcal{N}H_X$.
- ξ^i_{split}: call C_i is a call GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$, such that $u, v \in \mathcal{N}H_X$, but u and v are assigned to different supernodes during step 3 of GrowBuffer.

Let ξ_i be the event that either ξ^i_{build}, ξ^i_{buffer}, or ξ^i_{split} occurs. If ξ_i occurs, we say that *call C_i cuts e*. For each i, we define an indicator random variable (1 or 0) that indicates whether call C_i "threatens" e, that is, whether edge e could possibly be cut during call C_i.

- X^i_{build}: indicator random variable that is 1 if $C_i =$ BuildTree$(\mathcal{S}, H)$ initializes some supernode X with skeleton T_X, such that $\delta_H(\{u, v\}, T_X) \leq 2\Delta$ and $u, v \in H$.

- X^i_{buffer}: indicator random variable that is 1 if $C_i = \textsc{GrowBuffer}(\mathcal{S}, \mathcal{X}, H)$ such that $u, v \in H$ and $\delta_H(\{u, v\}, \mathcal{N}H_X) \le 2\Delta$.
- X^i_{split}: indicator random variable that is 1 if $C_i = \textsc{GrowBuffer}(\mathcal{S}, \mathcal{X}, H)$ such that $u, v \in \mathcal{N}H_X$.

We define X_i to be the indicator random variable that is 1 if X^i_{build}, X^i_{buffer}, or X^i_{split} is 1. If $X_i = 1$, we say that *call C_i threatens e*. In Section 6.3, we prove two lemmas that immediately let us bound the number of threatening calls, under *any* execution of the algorithm.

LEMMA 6.3. *There is a constant $\mu_{\text{build}} = O_r(1)$ such that for any vertex $\hat{v}$, there are at most μ_{build} calls $\textsc{BuildTree}(\mathcal{S}, H)$ that initializes some supernode X with skeleton T_X, such that $\delta_H(\hat{v}, T_X) \le 2\Delta$.*

LEMMA 6.4. *There is a constant $\mu_{\text{buffer}} = O_r(1)$ such that for any vertex $\hat{v}$, there are at most μ_{buffer} calls $\textsc{GrowBuffer}(\mathcal{S}, \mathcal{X}, H)$ such that $\hat{v} \in H$ and $\delta_H(\hat{v}, \mathcal{N}H_X) \le 2\Delta$.*

CLAIM 6.1. *For any execution of the algorithm, $\sum_{i \in \mathbb{N}} X_i \le O_r(1)$.*

Proof. For any vertex set A, the condition $\delta(\{u, v\}, A) \le 2\Delta$ implies that either $\delta(u, A) \le 2\Delta$ or $\delta(v, A) \le 2\Delta$. Thus, by applying Lemma 6.3 twice (first with $\hat{v} = u$, and then with $\hat{v} = v$), we conclude that $\sum_{i \in \mathbb{N}} X^i_{\text{build}} \le 2 \cdot \mu_{\text{build}} = O_r(1)$. Similarly Lemma 6.4 implies that $\sum_i X^i_{\text{buffer}} \le 2 \cdot \mu_{\text{buffer}} = O_r(1)$. Finally, we claim that $\sum_i X^i_{\text{split}} = 1$. Indeed, observe that if some vertex is in $\mathcal{N}H_X$ for some call $C_i = \textsc{GrowBuffer}(\mathcal{S}, \mathcal{X}, H)$, then that vertex is assigned by C_i; the claim follows from the fact that every vertex is assigned only once through the course of the algorithm. ∎

We now bound the probability that any specific threatening call cuts e.

CLAIM 6.2. *For any i, we have $\Pr[\xi_i \mid X_i = 1] < O_r(\|e\|/\Delta)$ and $\Pr[\xi_i \mid X_i = 0] = 0$.*

Proof. First observe that $\Pr[\xi^i_{\text{build}} \mid X^i_{\text{build}} = 0] = \Pr[\xi^i_{\text{build}} \mid X^i_{\text{buffer}} = 0] = \Pr[\xi^i_{\text{split}} \mid X^i_{\text{split}} = 0] = 0$. For the remaining cases, we prove that the probability is upper-bounded by $O_r(\|e\|/\Delta)$.

Case 1, $\boldsymbol{\xi^i_{\text{build}}}$: Suppose that $X^i_{\text{build}} = 1$, and call $C_i = \textsc{BuildTree}(\mathcal{S}, H)$. The skeleton T_η constructed by C_i is a random variable depending on the randomness of the previous calls C_j with $j < i$. Crucially, however, T_η is independent of the random value α chosen in Step 1 of BUILDTREE. Recall that even ξ^i_{build} occurs if exactly one of u and v is assigned to supernode η by C_i; that is, ξ^i_{build} occurs if some α is chosen such that $\delta_H(v, T_\eta) \le \alpha \cdot \Delta/r$ and $\delta_H(u, T_\eta) > \alpha \cdot \Delta/r$, or if $\delta_H(u, T_\eta) \le \alpha \cdot \Delta/r$ and $\delta_H(v, T_\eta) > \alpha \cdot \Delta/r$. By triangle inequality, if $\alpha \cdot \Delta/r < \delta_H(v, T_\eta) - \|e\|$, then both $\delta_H(v, T_\eta)$ and $\delta_H(u, T_\eta)$ are smaller than $\alpha \cdot \Delta/r$; on the other hand, if $\alpha \cdot \Delta/r > \delta_H(v, T_\eta) + \|e\|$, then both $\delta_H(v, T_\eta)$ and $\delta_H(u, T_\eta)$ are larger than $\alpha \cdot \Delta/r$. Thus, ξ^i_{build} occurs only if

$$\alpha \in \left(\delta_H(v, T_\eta) \cdot \frac{r}{\Delta} - \|e\| \cdot \frac{r}{\Delta}, \delta_H(v, T_\eta) \cdot \frac{r}{\Delta} + \|e\| \cdot \frac{r}{\Delta}\right)$$

This is an interval of width $2\|e\| \cdot r/\Delta = O_r(\|e\|/\Delta)$. As α is uniformly distributed and independent of $\delta_H(v, T_\eta)$, event ξ^i_{build} occurs with probability at most $O_r(\|e\|/\Delta)$.

Case 2, $\boldsymbol{\xi^i_{\text{buffer}}}$: Suppose that $X^i_{\text{buffer}} = 1$, and call $C_i = \textsc{GrowBuffer}(\mathcal{S}, \mathcal{X}, H)$. As above, the set $\partial H_{\downarrow X}$ is a random variable that depends only on the randomness of previous calls C_j with $j < i$, and so α is independent from $\partial H_{\downarrow X}$. By the same argument from Case 1, the probability that α is chosen such that exactly one of u and v is in $\mathcal{N}H_X$ is at most $O_r(\|e\|/\Delta)$.

Case 3, $\boldsymbol{\xi^i_{\text{split}}}$: Suppose that $X^i_{\text{split}} = 1$, and call $C_i = \textsc{GrowBuffer}(\mathcal{S}, \mathcal{X}, H)$. The sets $\mathcal{S}$ and $\mathcal{N}H_X$ are random variables that depends on the randomness of previous calls C_j with $j < i$ and on the value α chosen is Step 1 of the C_i call to GROWBUFFER; in particular, for any vertex $v' \in \mathcal{N}H_X$ and any supernode X' seen by H, the distance $\delta_{\text{dom}_{\mathcal{S}}(X)}(v', X' \cap \partial H_{\downarrow X})$ is independent of the $\alpha_{X'}$ values chosen during Step 2 of the C_i call. (For ease of presentation, define $X''_1 \coloneqq X'_1 \cap \partial H_{\downarrow X}$ and $X''_2 \coloneqq X'_2 \cap \partial H_{\downarrow X}$.) For any two supernodes X'_1 and X'_2 seen by H, we claim that u can be assigned to X'_1 and v assigned to X'_2 only if

$$\delta_{\text{dom}_{\mathcal{S}}(X)}(u, X''_1) + \alpha_{X'_1} \in \left(\delta_{\text{dom}_{\mathcal{S}}(X)}(v, X''_2) + \alpha_{X'_2} - \|e\|, \delta_{\text{dom}_{\mathcal{S}}(X)}(v, X''_2) + \alpha_{X'_2} + \|e\|\right). \tag{6.13}$$

Indeed, suppose that $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(u, X_1'') + \alpha_{X_1'} < \delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, X_2'') + \alpha_{X_2'} - \|e\|$. Then triangle inequality implies that $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, X_1'') + \alpha_{X_1'} \leq \|e\| + \delta_{\mathrm{dom}_{\mathcal{S}}(X)}(u, X_1'') + \alpha_{X_1'} < \delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, X_2'') + \alpha_{X_2'}$, and thus v is not assigned to X_2'. Similarly, if $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(u, X_1'') + \alpha_{X_1'} > \delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, X_2'') + \alpha_{X_2'} + \|e\|$, then one can show $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(u, X_1'') + \alpha_{X_1'} > \delta_{\mathrm{dom}_{\mathcal{S}}(X)}(u, X_2') + \alpha_{X_2''}$, and so u is not assigned to X_1'.

The interval in (6.13) has length $2\|e\|$. Recall that $\alpha_{X_1'}$ is uniformly chosen from an interval of length Δ/r, independent of $\alpha_{X_2'}$ and $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, X_2'')$ random variable. Thus for any fixed X_1' and X_2' seen by H, u is assigned to X_1' and v is assigned to X_2' with probability at most $\|e\|/(\Delta/r)$. Applying union bound over all $O(r^2)$ pairs of supernodes seen by H, we conclude that u and v are assigned to different supernodes by Step 2 of C_i with probability at most $O_r(\|e\|/\Delta)$. ∎

We now bound the total probability that edge e is cut.

LEMMA 6.5. *Let G be a graph excluding a fixed minor, and let $\mathcal{T}$ be a buffered cop decomposition constructed by the stochastic procedure* BUILDTREE$(\varnothing, G)$. *Then there is a constant β such that, for any edge e in G,* $\Pr[e \text{ is cut by } \mathcal{T}] \leq \beta \cdot \|e\|/\Delta$.

Proof. By Lemma 6.1, $\sum_{i\in\mathbb{N}} X_i \leq \mu = O_r(1)$. Then clearly $\mathbb{E}[\sum_{i\in\mathbb{N}} X_i] \leq \mu$, and linearity of expectation implies that $\sum_{i\in\mathbb{N}} \Pr[X_i = 1] \leq \mu$. We have

$$\begin{aligned}
\Pr[e \text{ is cut by } \mathcal{T}] &= \Pr[\bigcup_{i\in\mathbb{N}} \xi_i] \leq \sum_{i\in\mathbb{N}} \Pr[\xi_i] \\
&= \sum_{i\in\mathbb{N}} \Pr[\xi_i \mid X_i = 1] \cdot \Pr[X_i = 1] + \Pr[\xi_i \mid X_i = 0] \cdot \Pr[X_i = 0] \\
&\leq \sum_{i\in\mathbb{N}} O_r(1) \cdot \frac{\|e\|}{\Delta} \cdot \Pr[X_i = 1] \qquad \text{(by Claim 6.2)} \\
&\leq \mu \cdot O_r(1) \cdot \frac{\|e\|}{\Delta} = O_r(1) \cdot \frac{\|e\|}{\Delta}.
\end{aligned}$$

This concludes the proof. ∎

6.3 Bounding the number of threateners Our goal in this section is to prove Lemma 6.3 and 6.4 from the previous subsection. Both lemmas turn out to depend on bounding the number of threateners X to any vertex $\hat{v}$ with respect to $\mathrm{dom}_0(X)$.

LEMMA 6.6. *For any execution of* BUILDTREE$(\varnothing, G)$, *and for any vertex $\hat{v}$, there are $O_r(1)$ supernodes X such that $\delta_{\mathrm{dom}_0(X)}(\hat{v}, X) \leq 2\Delta$. (The dependency on r is around $r^{O(r^2)}$.)*

We will prove Lemma 6.3 and 6.4 using Lemma 6.6 after some properties of the buffered cop decomposition are introduced in Section 6.3.1. We remark that if one follows the original construction of buffered cop decomposition, Filtser [Fil24a, §4.2] showed that there are $O_r(1)$ supernodes X such that $\delta_{\mathrm{dom}(X)}(\hat{v}, X) \leq 2\Delta$; notice that the distance is measured in the final domain $\mathrm{dom}(X)$ after the buffered cop decomposition is built, as oppose to $\mathrm{dom}_0(X)$ which is possibly bigger. As a result, we cannot use the threatener bound from Filtser [Fil24a, §4.2] directly; in fact, we need to perform an involved charging argument based on the recursive call structure of GROWBUFFER.

To bound the number of threateners X with respect to $\mathrm{dom}_0(X)$, intuitively we will prove the existence of a *threatening sequence* $(X_0, \ldots, X_\ell)$ from X to the supernode η containing $\hat{v}$, such that (1) $X_0 = X$, (2) X_{i-1} threatens X_i for each i, and (3) X_ℓ threatens η, where each "threatening" relation is with respect to the final domain of the threatener. This way we can apply the bound from Filtser [Fil24a, §4.2] iteratively. Unfortunately, we cannot actually prove the second property of the threatening sequence; instead, we can only guarantee the existence of another supernode "related" to X_i that is threatened by X_{i-1}. (See Claim 6.11.) This leads to many complications, and along the way we need to re-establish and strengthen several properties from the buffered cop decomposition, including the supernode buffer property (Lemma 6.7) and the threatener bound (Claim 6.8 and Corollary 6.1), both with respect to a modification of the final domain.

Charging scheme. For every supernode X_0 with $\delta_{\mathrm{dom}_0(X_0)}(\hat{v}, X_0) \leq 2\Delta$, let $(X_0, \ldots, X_\ell)$ denote the threatening sequence of supernodes from X_0 to the supernode η containing $\hat{v}$ guaranteed by Claim 6.11, such that X_i is a *victim* of X_{i-1} for every i, and η is a victim of X_ℓ. (The exact statement of the threatening sequence or the meaning of victim is not important at the moment; one only has to know that ℓ, the length of any sequence, is less than r.) For each supernode X, create r pairs (X, i), one for each $i \in [0\,..\,r-1]$. For each threatening sequence, add a *charge* to the pair $(X_0, 0)$. We then move the charge along every threatening sequence from $(X_{i-1}, i-1)$ to (X_i, i) until the charge reaches (X_ℓ, ℓ). Finally, for every sequence we move the charge on (X_ℓ, ℓ) to η (length ℓ could be different from one sequence to another). Intuitively speaking, the amount of charge on the pair (X, i) across all threatening sequences counts the number of times a supernode X serves as the i-th step threatener in some threatening sequence. And the total amount of charge on η is equal to the number of supernode X_0 such that $\delta_{\mathrm{dom}_0(X_0)}(\hat{v}, X_0) \leq 2\Delta$.

Claim 6.3. *For any $i \in \mathbb{N}$ and any supernode X, the pair (X, i) receives at most μ^i charges, where $\mu = (r-1)\cdot(r-2)\cdot\binom{7r-1}{r-1}$. The supernode η containing $\hat{v}$ receives at most μ^r charges.*

We postpone the proof of Claim 6.3, and proceed to prove Lemma 6.6 using the charging bound.

Proof. [of Lemma 6.6] Fix a vertex $\hat{v}$. We need to count the number of supernodes X satisfying $\delta_{\mathrm{dom}_0(X)}(\hat{v}, X) \leq 2\Delta$. This is equal to the amount of charge on all pairs $(X, 0)$, which is also equal to the final charge received by η. By Claim 6.3, η receives at most μ^r charges, where $\mu = (r-1)\cdot(r-2)\cdot\binom{7r-1}{r-1}$. In total there are

$$\left((r-1)\cdot(r-2)\cdot\binom{7r-1}{r-1}\right)^r \leq O_r(1)$$

many supernodes X satisfying $\delta_{\mathrm{dom}_0(X)}(\hat{v}, X) \leq 2\Delta$. ∎

The rest of the section is devoted to the existence of threatening sequence (Claim 6.11) and a proof of the charging bound (Claim 6.3).

6.3.1 Properties of buffered cop decomposition

Let $\hat{v}$ be a fixed vertex. Imagine fixing an arbitrary sequence of random bits, so that there is a well-defined and deterministic notion of "the set of calls to BuildTree and GrowBuffer made during the execution of BuildTree$(\varnothing, G)$". Define $\mathrm{out}(\hat{v})$ to be the set of all vertices assigned by some call GrowBuffer$(\cdot, \cdot, H)$ where the subgraph H does not contain $\hat{v}$. We need a few observations from [CCL+24] that extends to the presence of $\mathrm{out}(\hat{v})$. The main reason we introduce $\mathrm{out}(\hat{v})$, which may not be clear at the moment, is for a technical but crucial claim for the threatening sequence (see Claim 6.10).

We elaborate a bit on the motivation for $\mathrm{out}(\hat{v})$. In the threatening sequence sketched at the start of Section 6.3, we would like to guarantee that X_{i-1} threatens X_i (or, more precisely, threatens some supernode related to X_i) with respect to $\mathrm{dom}(X_{i-1})$. However, the amount of charge on X_i would depend on the number of possible threateners X_{i-1}, which is tied to the number of calls to GrowBuffer that expand X_i. The number of such calls could be unbounded.[8] We only have a bound on the number of calls GrowBuffer$(\cdot, \cdot, H)$ that expand X_i *and satisfy* $\hat{v} \in H$, for any fixed vertex $\hat{v}$; see Claim 6.6. To exploit this bound, we construct a threatening sequence with the weaker guarantee that X_{i-1} threatens (a supernode related to) X_i with respect to $\mathrm{dom}(X_{i-1}) \cup \mathrm{out}(\hat{v})$, rather than with respect to $\mathrm{dom}(X_{i-1})$.

The following observation is essentially a rephrasing of Invariant 3.3 in [CCL+24], and follows from the fact that a call to GrowBuffer or BuildTree is made on a subgraph H only if H is a maximal connected component of unassigned vertices.

Observation 6.1. (Rephrasing of [CCL+24] Invariant 3.3) *Suppose that some call C, whether it is* BuildTree$(\mathcal{S}, H)$ *or* GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$*, occurs during execution of the algorithm. Then every vertex*

[8] While the out-of-box [supernode radius] property from [CCL+24, §3.4] implies that X_i can expand at most $O(1)$ times "in one direction", there could be an unbounded number of GrowBuffer calls that each expand X_i in different directions. Indeed, imagine a long-star graph with a large number of paths attached to a single center vertex x, which we initialized to be a supernode X. For each path P hanging off x, there are $O(1)$ GrowBuffer calls that assign vertices in P to be part of X; however there could be unbounded many calls to GrowBuffer that expands X along different paths P.

x in $G \setminus H$ that is adjacent to H was assigned in $\mathcal{S}$ at the time C was called. Further, x was assigned during some call to BuildTree$(\mathcal{S}', H')$ *or* GrowBuffer$(\mathcal{S}', \mathcal{X}', H')$ *for which H is a subgraph of H'. In particular, this implies that if $\hat{v} \in H$, then every such vertex x cannot be in* $\mathrm{out}(\hat{v})$.

One immediate consequence of Observation 6.1 is:

Observation 6.2. *Let X and η be supernodes, where* $\mathrm{dom}_0(\eta)$ *sees X at the time η is initialized. Then X is a proper ancestor of η in the partition tree.*

Claim 6.4. (Rephrasing of [CCL+24] Claim 3.4(2)) *Suppose that call C, whether it is* BuildTree$(\mathcal{S}, H)$ *or* GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$*, occurs at some point in the execution of the algorithm. Over the course of execution, every vertex in H is either assigned to a supernode initialized by or below C, or is assigned to some supernode that H sees (at the time C is called). In particular, if $C =$* BuildTree$(\mathcal{S}, H)$ *which initializes some supernode η, then every vertex x in* $\mathrm{dom}_0(\eta)$ *but not in* $\mathrm{dom}(\eta)$ *is assigned to some supernode in $\mathcal{S}_{|\eta}$.*

Claim 6.5. (Implicit from [CCL+24] Claim 3.5) *Suppose* BuildTree$(\mathcal{S}, H)$ *or* GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$ *is called during the algorithm. Let $\mathcal{S}_{|H}$ be the set of supernodes in $\mathcal{S}$ seen by H. Then $\mathcal{S}_{|H}$ contains at most $r-2$ supernodes; furthermore, the supernodes in $\mathcal{S}_{|H}$ are pairwise adjacent.*

Claim 6.6. (Implicit from [CCL+24, §3.4] on supernode radius property) *For any supernode η, over the course of execution of the algorithm, there are at most $r-1$ calls $C =$* GrowBuffer$(\cdot, \cdot, H)$ *such that C expands η and $\hat{v} \in H$.*

Proof. First observe that if two calls $C =$ GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$ and $\tilde{C} =$ GrowBuffer$(\tilde{\mathcal{S}}, \tilde{\mathcal{X}}, \tilde{H})$ are made during the execution of the algorithm with $\hat{v}$ being in both H and $\tilde{H}$, then either H is a subgraph of $\tilde{H}$, or $\tilde{H}$ is a subgraph of H. This follows form the fact that calls to GrowBuffer are only made on maximal connected components of unassigned vertices. As a result, all calls to GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$ that expands η where $\hat{v} \in H$ can be put into a linear order based on the containment relationship of H. Among such calls, the contrapositive of Claim 3.12 in [CCL+24] implies that, for any call $C =$ GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$ that processes a supernode X, there is no earlier call GrowBuffer$(\tilde{\mathcal{S}}, \tilde{\mathcal{X}}, \tilde{H})$ with $H \subseteq \tilde{H}$ that processes X. Thus, for any supernode X, there is at most one call to GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$ with $\hat{v} \in H$.

Finally, it was shown in Claim 3.13 of [CCL+24] that if some supernode η is expanded during a GrowBuffer call that processes X, then $\mathrm{dom}_0(\eta)$ sees X at the time η was initialized. It follows from the [shortest-path skeleton] property that $\mathrm{dom}_0(\eta)$ sees only $r-1$ supernodes at the time η was initialized. This proves the claim. ∎

Before we continue, first we prove Lemma 6.3 and Lemma 6.4 from the previous subsection.

Proof. [of Lemma 6.3] By Lemma 6.6, for any vertex $\hat{v}$, there are $O_r(1)$ supernodes X such that $\delta_{\mathrm{dom}_0(X)}(\hat{v}, X) \leq 2\Delta$; these includes those supernode X satisfying $\delta_{\mathrm{dom}_0(X)}(\hat{v}, T_X) \leq 2\Delta$. ∎

Proof. [of Lemma 6.4] For some i, suppose C_i is a call to GrowBuffer$(\mathcal{S}, \mathcal{X}, H)$ such that $\hat{v} \in H$ and $\delta_H(\hat{v}, \mathcal{N}H_X) \leq 2\Delta$. By definition of $\mathcal{N}H_X$, there is some vertex $x \in \mathcal{N}H_X$ such that $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(\hat{v}, x) \leq 2\Delta$. Let x be the vertex that minimizes this distance. Let η be the supernode containing x. Observe that (1) call C_i expands η, and (2) $\hat{v} \in H$. We further claim that (3) $\delta_{\mathrm{dom}_0(\eta)}(\hat{v}, \eta) \leq 2\Delta$; indeed, by choice of x, there is a path from v to x in $H \cup \{x\}$ with length at most 2Δ, and Observation 6.1 implies that H is a subgraph of $\mathrm{dom}_0(\eta)$.

By Lemma 6.6, there are $O_r(1)$ supernodes η such that $\delta_{\mathrm{dom}_0(\eta)}(v, \eta) < 2\Delta$. For each such supernode η, Claim 6.6 implies that there are at most $r-1$ calls to GrowBuffer$(\cdot, \cdot, H)$ with $\hat{v} \in H$ that expand the supernode η. We conclude that there are at most $O_r(1)$ calls that satisfy properties (1–3). ∎

6.3.2 Supernode buffer property We now strengthen the [supernode buffer] property proved by [CCL+24] to work with $\mathrm{out}(\hat{v})$.

Claim 6.7. *Let η be a supernode $\hat{v} \in \mathrm{dom}_0(\eta)$. Let X be a supernode above*[9] *η in the partition tree. If η is not adjacent to X in G, then $\delta_{\mathrm{dom}(X) \cup \mathrm{out}(\hat{v})}(v, X) > \Delta/r$ for every vertex v in* $\mathrm{dom}(\eta)$.

[9] that is, X is a proper ancestor of η in the partition tree

Proof. Lemma 3.10 in [CCL+24] proves a similar claim, with a guarantee of $\delta_{\mathrm{dom}(X)}(v, X) > \Delta/r$ instead of $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(v, X) > \Delta/r$. The proof from [CCL+24] carries over almost verbatim for our new statement, with two modifications. First, [CCL+24] introduce a claim that they proved inductively:

> Let $C' \coloneqq \textsc{BuildTree}(\mathcal{S}', H')$ be a call that is below (in the recursion tree) the call that initialized X. Either H' sees X (at the time C' is called), or $\delta_{\mathrm{dom}(X)}(v, X) > \Delta/r$ for every vertex v in H'.

For our proof, we slightly modify the claim:

> Let $C' \coloneqq \textsc{BuildTree}(\mathcal{S}', H')$ be a call that is below (in the recursion tree) the call that initialized X, with $\hat{v} \in H'$. Either H' sees X (at the time C' is called), or $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(v, X) > \Delta/r$ for every vertex v in H'.

The additional assumption that $\hat{v} \in H'$ can be added without breaking the proof of [CCL+24]; indeed, they only apply the inductive claim to the call $C_\eta \coloneqq \textsc{BuildTree}(\mathcal{S}_\eta, H_\eta)$ that initialized η (for which we have $\hat{v} \in \mathrm{dom}_0(\eta) = H_\eta$ by assumption) or to a call $\textsc{BuildTree}(\mathcal{S}', H')$ that is an ancestor of C_η in the recursion tree (and thus H_η is a subgraph of H', and $\hat{v} \in H'$).

The second modification occurs in the proof of the claim introduced above. We first recall some details from the proof of [CCL+24]. During their proof, they consider a shortest path P in $\mathrm{dom}(X)$ between X and v, and show that $\|P\| > \Delta/r$ if η is not adjacent to X. They consider inductively a certain call $C = \textsc{GrowBuffer}(\mathcal{S}, \mathcal{X}, H)$ where H', the graph from the call $C' \coloneqq \textsc{BuildTree}(\mathcal{S}', H')$ described above, is a subgraph of H and H is a subgraph of $\mathrm{dom}_0(X)$, such that C processes X. They define x to be some vertex on P that is in $G \setminus H$ and is adjacent to H (they show that such an x exists, and without loss of generality we may assume that it is the first such vertex along P when travelling from v to X). Recall that $\partial H_{\downarrow X}$ is defined as the set of vertices in $G \setminus H$ that are adjacent to H and are in $\mathrm{dom}_{\mathcal{S}}(X)$. They need to show that $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, \partial H_{\downarrow X}) \le \|P\|$. (From this, they show that $\|P\| \le \Delta/r$ implies that v must have been assigned by the call C and thus $v \notin H'$, a contradiction; thus $\|P\| > \Delta/r$. We emphasize that this argument holds with our stochastic version of GrowBuffer — as we always choose $\alpha \ge 1$ in Step 1 of GrowBuffer, every vertex within distance Δ/r of $\partial H_{\downarrow X}$ is assigned during C.) Their proof that $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, \partial H_{\downarrow X}) \le \|P\|$ is the only place that uses the assumption that P is in $\mathrm{dom}(X)$, and it is as follows: (1) P is a path in $\mathrm{dom}(X)$ and thus in $\mathrm{dom}_{\mathcal{S}}(X)$ (as domains only shrink over the course of the algorithm); and (2) as P is in $\mathrm{dom}_{\mathcal{S}}(X)$, the vertex x is in $\mathrm{dom}_{\mathcal{S}}(X)$ and thus in $\partial H_{\downarrow X}$; thus $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, \partial H_{\downarrow X}) \le \delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, x) \le \|P\|$.

To prove our new claim, we instead let P be a shortest path in $\mathrm{dom}(X) \cup \mathrm{out}(\hat{v})$ between X and v; we show that $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, \partial H_{\downarrow X}) \le \|P\|$. (Notice that the distance is still measured in $\mathrm{dom}_{\mathcal{S}}(X)$ as GrowBuffer algorithm expands supernode based on it.) We observe that $\hat{v}$ is in H; this is because $\hat{v} \in H'$ (by assumption) and H' is a subgraph of H. It follows from Observation 6.1 that vertex x is not in $\mathrm{out}(\hat{v})$. Thus, $x \in \mathrm{dom}_{\mathcal{S}}(X)$, and so $x \in \partial H_{\downarrow X}$. By choice of x, the prefix of P from v to x, denoted $P[v : x]$, is contained in H. But H is a subgraph of $\mathrm{dom}_{\mathcal{S}}(X)$, as H is a subgraph of $\mathrm{dom}_0(X)$ (by assumption on the call C) and every vertex of H is unassigned in $\mathcal{S}$. Thus, $\delta_{\mathrm{dom}_{\mathcal{S}}(X)}(v, \partial H_{\downarrow X}) \le \|P\|$. By following the proof of [CCL+24], we conclude that $\|P\| > \Delta/r$ unless η is adjacent to X. ∎

Following Filtser [Fil24a], for any buffered cop decomposition we define a *dag* $\vec{G}$ whose vertices are the supernodes, and there is an edge from supernode η to supernode η' if (1) η and η' are adjacent in G, and (2) η' is an ancestor of η in the partition tree. As observed by [Fil24a], the [shortest-path skeleton] property of buffered cop decomposition implies that every supernode in $\vec{G}$ has out-degree at most $r - 1$.

Claim 6.8. ([Fil24a, Lemma 1], with $w = r - 1$) *For any supernode η in $\vec{G}$ and any $q \in \mathbb{N}$, there are at most $\mu = \binom{q+r-1}{r-1}$ supernodes X such that $\vec{G}$ contains an path from η to X of length at most q.*

The next claim is similar in spirit to [Fil24a, Lemma 2], which is an analogous claim with $\delta_{\mathrm{dom}(X)}(\eta, X) \le q \cdot \Delta/r$ instead of $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(\eta, X) \le q \cdot \Delta/r$. The details of the proof are rather different, and we provide a complete proof here.

Claim 6.9. *Let η be the supernode with $\hat{v} \in \mathrm{dom}_0(\eta)$, let X be a supernode with skeleton T_X such that X is above η in the partition tree, and let $q \in \mathbb{N}$ with $q \ge 1$. If $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(\eta, T_X) \le q \cdot \Delta/r$, then $\vec{G}$ contains a path from η to X of length at most $2q$.*

Proof. We prove the claim by induction on q. If $q = 1$, then the claim immediately follows from our stronger supernode buffer property (Claim 6.7).

In the inductive case when $q > 1$, if there is a path in $\vec{G}$ from η to X of length at most 2, then we are done. Otherwise, let P be a shortest-path in $\mathrm{dom}(X) \cup \mathrm{out}(\hat{v})$ between some vertex $a \in \eta$ and a vertex $b \in T_X$, with $\|P\| \leq q \cdot \Delta/r$. We claim:

(6.14) There is some supernode $\hat{\eta}$ such that (1) some vertex on P is in $\hat{\eta}$, (2) there is *not* an edge from η to $\hat{\eta}$ in $\vec{G}$, but (3) there *is* a 2-hop path from η to $\hat{\eta}$ in $\vec{G}$, (4) $\hat{v} \in \mathrm{dom}_0(\hat{\eta})$, and (5) X is above $\hat{\eta}$ in the partition tree.

We will prove (6.14) later. Now we complete the inductive step assuming (6.14). Indeed, let x be some vertex on $P \cap \hat{\eta}$ from (6.14)(1). By (6.14)(2) and the buffer property of Claim 6.7, we have $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(\eta, \hat{\eta}) > \Delta/r$. As P passes through $\hat{\eta}$, we have $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(\hat{\eta}, X) \leq \|P\| - \Delta/r \leq (q-1) \cdot \Delta/r$. By the inductive hypothesis (which we may apply due to conditions (6.14)(4–5), $\vec{G}$ contains a path from $\hat{\eta}$ to X of length at most $2q - 2$. Thus by (6.14)(3), $\vec{G}$ contains a path from η to X of length at most $2q$.

We now prove (6.14). Define $\eta_0 \coloneqq \eta$ and $a_0 \coloneqq a$. For every $i > 0$, inductively define a_i to be the first vertex along the subpath $P[a_{i-1} : b]$ (when moving from a_{i-1} to b) that is not in $\mathrm{dom}(\eta_{i-1}) \cup \mathrm{out}(\hat{v})$, and let η_i be the supernode containing a_i. We make three observations about η_i.

- *There is an edge from η_{i-1} to η_i in $\vec{G}$, and η_i is a proper ancestor of η_0 in the partition tree.* It suffices to show that $\mathrm{dom}_0(\eta_{i-1})$ sees η_i at the time η_{i-1} is initialized: in this case, by construction there is an edge between η_{i-1} and η_i in G, and Observation 6.2 implies that η_i is a proper ancestor of η_{i-1} (and thus of η_0 as well by induction) in the partition tree. To this end, consider the subpath $P[a_{i-1} : a_i]$ of path P that starts at a_{i-1} and ends at a_i. By choice of a_i, path $P[a_{i-1} : a_i]$ is contained in $\mathrm{dom}(\eta_{i-1}) \cup \mathrm{out}(\hat{v}) \cup \eta_i$. Consider the path $P[a_{i-1} : a_i]$ at the time η_{i-1} was initialized by some call $\textsc{BuildTree}(\mathcal{S}, H)$. As $a_{i-1} \in \eta_{i-1}$, we have $a_{i-1} \in \mathrm{dom}_0(\eta_{i-1}) = H$. Now, there are two cases. If $a_i \in H$, then a_i is assigned to some supernode seen by η_{i-1}, by Claim 6.4, and we are done. Otherwise, $a_i \notin H$. As one endpoint of $P[a_{i-1} : a_i]$ is in H and the other is not in H, there exists some vertex y on $P[a_{i-1} : a_i]$ that lies in $G \setminus H$ but is adjacent to H. As $\hat{v} \in \mathrm{dom}_0(\eta_0)$ by assumption, and η_{i-1} is a proper ancestor of η_0 in the partition tree by induction, we have $\hat{v} \in \mathrm{dom}_0(\eta_{i-1}) = H$, so Observation 6.1 implies that $y \notin \mathrm{out}(\hat{v})$. But $y \notin \mathrm{dom}(\eta_{i-1})$ as it is not in H, and so $y \in \eta_i$. We conclude that $\mathrm{dom}_0(\eta_{i-1})$ sees η_i.

- *For every η_i, we have $\hat{v} \in \mathrm{dom}_0(\eta_i)$, and either $X = \eta_i$ or X is above η_i in the partition tree.* We have $\hat{v} \in \mathrm{dom}_0(\eta_i)$ immediately from the fact that every η_i is above η_{i-1} in the partition tree, and $\hat{v} \in \mathrm{dom}_0(\eta_0)$. For the latter claim, observe that $a_i \notin \mathrm{out}(\hat{v})$ (as, by definition, it is the first vertex not in $\mathrm{dom}(\eta_{i-1}) \cup \mathrm{out}(\hat{v})$) and so it is in $\mathrm{dom}(X)$ (as P is a path in $\mathrm{dom}(X) \cup \mathrm{out}(\hat{v})$); the fact that $a_i \in \eta_i$ and $a_i \in \mathrm{dom}(X)$ implies that either $\eta_i = X$ or X is above η_i in the partition tree.

- Let $i_{\max}$ be the largest number such that $a_{i_{\max}}$ exists; that is, every vertex on the subpath $P[a_{i_{\max}} : b]$ is in $\mathrm{dom}(\eta_{i_{\max}}) \cup \mathrm{out}(\hat{v})$. *We argue that $\eta_{i_{\max}} = X$.* Indeed, vertex b is in the skeleton T_X, so it is assigned during a call to $\textsc{BuildTree}$ and not during a call to $\textsc{GrowBuffer}$. Thus, $b \notin \mathrm{out}(\hat{v})$, and so (by assumption on $P[a_{i_{\max}} : b]$) we have $b \in \mathrm{dom}(\eta_{i_{\max}})$. As $b \in T_X$ and $b \in \mathrm{dom}(\eta_{i_{\max}})$, either $\eta_{i_{\max}} = X$ or $\eta_{i_{\max}}$ is above X in the partition tree. On the other hand, we showed above that either $\eta_i = X$ or X is above η_i in the partition tree for every η_i. We conclude that $\eta_{i_{\max}} = X$.

We can now prove (6.14). Let k be the smallest number such that there is *not* an edge from η to η_k in $\vec{G}$. (If no such supernode exists, then there is an edge from η to $\eta_{i_{\max}} = X$ in $\vec{G}$. This contradicts our assumption that there is no path from η to X with length at most 2.) (1) Clearly, there is some vertex on P in η_k (namely, a_k), and (2) there is no edge from η to η_k in $\vec{G}$. Finally, there is an edge in $\vec{G}$ from η_{k-1} to η_k by construction, and (by assumption on k) there is an edge in $\vec{G}$ from η to η_{k-1}; thus, (3) there is a 2-hop path from η to η_k in $\vec{G}$. As show above, (4) $\hat{v} \in \mathrm{dom}_0(\eta_k)$, and (5) either $X = \eta_k$ or X is above η_k in the partition tree (and if $X = \eta_k$, $\vec{G}$ contains a 2-hop path from η to X, contradicting our assumption). Thus, choosing $\hat{\eta} \coloneqq \eta_k$ satisfies (6.14). ∎

It is easy to generalize the claim to apply to the case where $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(\eta, X)$ is bounded. We say that supernode η is a *q-step victim of a supernode X* (*victim* for short) if (i) $\hat{v} \in \mathrm{dom}_0(\eta)$, (ii) X is an ancestor of η in the partition tree, where possibly $X = \eta$, and (iii) $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(\eta, X) \leq q \cdot \Delta/r$.

COROLLARY 6.1. *Let integer $q \geq 1$. Let η be a supernode that is a q-step victim of another supernode X. Then $\vec{G}$ contains a path from η to X of length at most $2(q+r)$.*

Proof. The statement is trivial if $X = \eta$. Let T_X denote the skeleton of X. Every vertex in X is within distance Δ of T_X, by the [supernode radius] property. Thus, $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(\eta, X) \leq q \cdot \Delta/r + \Delta = (q+r) \cdot \Delta/r$, and the claim follows from Claim 6.9. ∎

COROLLARY 6.2. *Let η_i be a supernode. There are at most $\binom{2(q+r)+r-1}{r-1}$ supernodes X_{i-1} that has η_i as a q-step victim.*

Proof. Corollary 6.1 (where $\eta = \eta_i$ and $X = X_{i-1}$) shows that if X_{i-1} has η_i as a q-step victim, then there is a path from η_i to X_{i-1} of length at most $2(q+r)$ in dag $\vec{G}$. By Claim 6.8, there are at most $\binom{2(q+r)+r-1}{r-1}$ supernodes X_{i-1} such that $\vec{G}$ contains an path from η_i to X_{i-1} of length at most $2(q+r)$. ∎

6.3.3 Threatening sequence So far, we have bounded the number of supernodes X for which $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(\hat{v}, X) \leq 2\Delta$. However, we really want to bound the number of supernodes X for which $\delta_{\mathrm{dom}_0(X)}(\hat{v}, X) \leq 2\Delta$. We now show the following lemma, Claim 6.10, which is the heart of the proof: if $\delta_{\mathrm{dom}_0(X)}(\hat{v}, X) \leq 2\Delta$ but $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(\hat{v}, X) > 2\Delta$, we can "charge" this to some $C = \textsc{GrowBuffer}(\cdot, \cdot, H)$ call and replace X with some supernode X' expanded by C. In this way, we bound the number of supernodes X for which $\delta_{\mathrm{dom}_0(X)}(\hat{v}, X) \leq 2\Delta$. The complication we mentioned at the start of the section is that X' is not a victim (and thus a descendant) of X, but is in fact an ancestor of X; we have to choose a victim from the supernodes seen by H during the $\textsc{GrowBuffer}(\cdot, \cdot, H)$ call instead. (In fact, it is crucial that X' is an ancestor of X, as this lets us prove that the threatening sequence has length at most $r-1$; see Claim 6.11.) See Figure 8 for an example of Claim 6.10.

CLAIM 6.10. *Let X be a supernode and $q \in \mathbb{N}$. Let P be a path in $\mathrm{dom}_0(X)$ from X to $\hat{v}$ with length at most $q \cdot \Delta/r$. Then either P is already in $\mathrm{dom}(X) \cup \mathrm{out}(\hat{v})$; or there is a call $C = \textsc{GrowBuffer}(\mathcal{S}, \mathcal{X}, H)$ such that*

(1) $\hat{v} \in H$,

(2) there is some q-step victim η' of X such that H sees η' at the time C is called,

(3) there is some supernode X' expanded by C and some suffix path P' of P, such that P' is a path from X' to $\hat{v}$ in $\mathrm{dom}_0(X')$, and X' is a proper ancestor of X in the partition tree.

Proof. Suppose that P is not in $\mathrm{dom}(X) \cup \mathrm{out}(\hat{v})$. We will show that some appropriate call C exists. Let *vertex x* be the first one along P (walking from X to $\hat{v}$) that is not in $\mathrm{dom}(X) \cup \mathrm{out}(\hat{v})$. As x is on P which is in $\mathrm{dom}_0(X)$, Claim 6.4 implies that x was assigned to some supernode X' that was seen by $\mathrm{dom}_0(X)$. Let $C = \textsc{GrowBuffer}(\mathcal{S}, \mathcal{X}, H)$ be the call during which x was assigned. (In particular, x is in H.) We show that C satisfies the three conditions required by the claim.

(1) By choice of x, we have $x \notin \mathrm{out}(\hat{v})$, and so $\hat{v} \in H$ by definition of $\mathrm{out}(\hat{v})$.

(2) Consider the prefix $P[X : x]$ of P that runs from X to x. One endpoint x of $P[X : x]$ is in H. Suppose the whole prefix $P[X : x]$ is in H. The other endpoint x' of $P[X : x]$ must *eventually* be assigned to X. Notice that X cannot be initialized by or after C, because the whole path P, including x, is in $\mathrm{dom}_0(X)$. Every vertex in $\mathrm{dom}_0(X)$ is unassigned at the time X is initialized; as call C assigns x, call C cannot occur before X is initialized. Claim 6.4, applied on C and $x' \in H$, then implies that x' is assigned to a supernode (which must be X) that H sees at the time C is called. We choose $\eta' \coloneqq X$, and η' is trivially a q-step victim of itself.

If some vertex on $P[X : x]$ is not in H, then there is some *vertex x'* on $P[X : x] \setminus \{x\}$ that is in $G \setminus H$ but is adjacent to H. We choose η' to be the supernode containing x'; because $x' \in G \setminus H$, η' must have included x' at the time C is called. We show that η' is a q-step victim of X and H sees η'. By definition of x' and η', the graph H sees η' at the time C is called. (i) As H sees η', it follows (by repeatedly applying Observation 6.1) that η' was initialized by some call to $\textsc{BuildTree}(\mathcal{S}', H')$ where $H \subseteq H'$; in particular, by (1), $\hat{v} \in H \subseteq H' = \mathrm{dom}_0(\eta')$. (ii) By Observation 6.1, $x' \notin \mathrm{out}(\hat{v})$; by the choice of x, $P[X : x']$ is

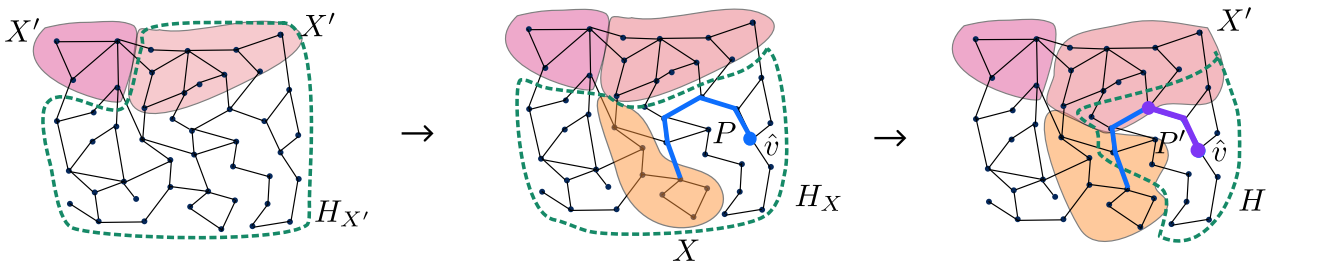


Figure 8: A series of BuildTree and GrowBuffer calls, demonstrating one case of Claim 6.10. The calls C_1 and C_2 initialize supernodes X' and X, respectively. Supernode X' is an ancestor of X, so vertices in X' are not in $\mathrm{dom}(X)$. The blue path P is a shortest-path from X to $\hat{v}$, in the graph $\mathrm{dom}_0(X) \cup \mathrm{out}(\hat{v})$. The GrowBuffer call C_3 expands supernode X', causing P to not be contained in $\mathrm{dom}(X) \cup \mathrm{out}(\hat{v})$. The call C_3 satisfies the conditions of Claim 6.10: $\hat{v}$ is in H, supernode $\eta' \coloneqq X$ is seen by H, and supernode X' is connected to $\hat{v}$ by a subpath P' of P.

contained in $\mathrm{dom}(X) \cup \mathrm{out}(\hat{v})$, thus we have $x' \in \mathrm{dom}(X)$. This implies that either $X = \eta'$ or X is above η' in the partition tree. (iii) Again, because the path $P[X : x']$ is contained in $\mathrm{dom}(X) \cup \mathrm{out}(\hat{v})$, we have $\delta_{\mathrm{dom}(X)\cup\mathrm{out}(\hat{v})}(\eta', X) \le q \cdot \Delta/r$.

(3) By definition, X' was expanded by C. Further, consider the subpath $P' \coloneqq P[x : \hat{v}]$ of P. This is a path from X' to $\hat{v}$. Further, as X' is a proper ancestor of X in the partition tree (by Observation 6.2), $\mathrm{dom}_0(X)$ is a subgraph of $\mathrm{dom}_0(X')$. As P is a path in $\mathrm{dom}_0(X)$, it follows that $P[x : \hat{v}]$ is a path in $\mathrm{dom}_0(X')$. Finally, observe that $x \notin \mathrm{dom}(X)$ and x was assigned to X', so $X \neq X'$.

This concludes the proof. ∎

By applying this claim repeatedly, we can prove the existence of a threatening sequence.

Claim 6.11. *Let $q \in \mathbb{N}$, and let X be a supernode with $\delta_{\mathrm{dom}_0(X)}(\hat{v}, X) \le q \cdot \Delta/r$. Define $X_0 \coloneqq X$. For some $\ell < r - 1$ (possibly $\ell = 0$), there is a sequence of ℓ calls $C_i = \textsc{GrowBuffer}(\mathcal{S}_i, \mathcal{X}_i, H_i)$ and a sequence of supernodes X_i expanded by C_i such that:*

(1) for every $i \in [1 \,..\, \ell]$, $\hat{v} \in H_i$;

(2) for every $i \in [1 \,..\, \ell]$, there is a q-step victim η_i of X_{i-1} such that H_i sees η_i at the time C_i is called;

(3) denote $\eta_{\hat{v}}$ the supernode containing $\hat{v}$, then $\eta_{\hat{v}}$ is a q-step victim of X_ℓ.

Proof. Let P_0 be a path in $\mathrm{dom}_0(X_0)$ from X_0 to $\hat{v}$, with $\|P_0\| \le q \cdot \Delta/r$. For every $i > 0$, we either decide to end the algorithm by setting $\ell = i - 1$ or we inductively define a call C_i, a supernode X_i, and suffix path P_i, using supernode X_{i-1} and path P_{i-1}. We maintain the following invariant: Let $\mathcal{S}_{|X_0} \coloneqq \{S_1, \ldots, S_k\}$ denote the set of supernodes seen by $\mathrm{dom}_0(X_0)$ at the time X_0 is created, where $k \le r - 2$ by Claim 6.5.

P_i is a path in $\mathrm{dom}_0(X_i)$ from X_i to $\hat{v}$ that is a suffix path of P_0, and lies in the union of $\mathrm{dom}(X_i) \cup \{\text{at most } k - i \text{ supernodes from } \mathcal{S}_{|X_0}\}$, and the call C_i satisfies conditions (1–2).

For the base case when $i = 0$, P lies in $\mathrm{dom}(X) \cup \bigcup \mathcal{S}_{|X_0}$ by Claim 6.4.

Apply Claim 6.10 (choosing $X = X_{i-1}$ and $P = P_{i-1}$). Either P_{i-1} is already in $\mathrm{dom}(X_{i-1}) \cup \mathrm{out}(\hat{v})$ and thus $\hat{v} \in \mathrm{dom}(X_{i-1})$ (because $\hat{v}$ is never in $\mathrm{out}(\hat{v})$), or there is a call $C_i = \textsc{GrowBuffer}(\mathcal{S}_i, \mathcal{X}_i, H_i)$ satisfying (1–3) of Claim 6.10. In the first case, we show that $\eta_{\hat{v}}$ is a q-step victim of $X_\ell \coloneqq X_{i-1}$. We have $\hat{v} \in \mathrm{dom}_0(\eta_{\hat{v}})$ and X_{i-1} is above $\eta_{\hat{v}}$ in the partition tree (as $\hat{v} \in \mathrm{dom}(X_{i-1})$), and

$$\delta_{\mathrm{dom}(X_{i-1})\cup\mathrm{out}(\hat{v})}(\eta_v, X_{i-1}) \le \|P_{i-1}\| \le q \cdot \Delta/r.$$

Thus, we may choose $\ell := i-1$, and $\eta_{\hat{v}}$ and X_ℓ satisfy condition (3). Notice that ℓ must be less than k, because by the invariant P_i lies in the union of $\mathrm{dom}(X_i)$ and at most $k-i$ supernodes from $\mathcal{S}_{|X_0}$; after up to k iterations P_i lies in $\mathrm{dom}(X_i)$ completely.

In the other case, we choose C_i to be the call GrowBuffer$(\mathcal{S}_i, \mathcal{X}_i, H_i)$ guaranteed by Claim 6.10, choose X_i to be the supernode expanded by C_i (called X' in the statement of Claim 6.10), and choose P_i to be the suffix path of P_{i-1} that connects X_i and $\hat{v}$ in $\mathrm{dom}_0(X_i)$ (called P' in the statement of Claim 6.10). Conditions (1–2) follow immediately from Claim 6.10(1–2). It remains to show that P_i is in the union of $\mathrm{dom}(X_i)$ and at most $k-i$ supernodes from $\mathcal{S}_{|X_0}$. By the invariant, P_{i-1} is in the union of $\mathrm{dom}(X_{i-1})$ and at most $k-i+1$ supernodes from $\mathcal{S}_{|X_0}$, without loss of generality to be $\{S_1, \ldots, S_{k-i+1}\}$. By Claim 6.10(3), supernode X_i is a proper ancestor of X_{i-1} in the partition tree, and thus X_i is not in $\mathrm{dom}(X_{i-1})$; as some vertex of P_{i-1} lies on X_i, we thus have $X_i \in \{S_1, \ldots, S_{k-i+1}\}$. Without loss of generality suppose that $X_i = S_{k-i+1}$. Now, because X_i is an ancestor of X_{i-1} in the partition tree, $\mathrm{dom}(X_{i-1})$ is a subgraph of $\mathrm{dom}(X_i)$ (and clearly $X_i = S_{k-i+1}$ is a subgraph of $\mathrm{dom}(X_i)$). We conclude that P_i is a path in the union of $\mathrm{dom}(X_i)$ and $\{S_1, \ldots, S_{k-i}\}$. ∎

6.3.4 Proof of charging bound Now we are ready to prove stronger version of Claim 6.3; Claim 6.3 follows by setting $q = 2r$. To handle the technical issue that X_{i-1} does not threaten X_i directly but threatens instead some η_i seen by H_i where $C_i = \text{GrowBuffer}(\mathcal{S}_i, \mathcal{X}_i, H_i)$ was called, we argue that at most $r-1$ supernodes X_i are expanded by C_i and thus at most $r-1$ supernodes η_i can be seen by H_i (at the time C_i is called) using Claim 6.6.

Claim 6.12. *For any $i \in \mathbb{N}$ and any supernode X, the pair (X, i) receives at most μ^i charges, where $\mu = (r-1)\cdot(r-2)\cdot\binom{2(q+r)+r-1}{r-1}$. The supernode $\eta_{\hat{v}}$ containing $\hat{v}$ receives at most μ^r charges.*

Proof. We will prove the statement by induction on i. For the base case when $i = 0$, every supernode X induces one threatening sequence with $X_0 = X$ which charges $(X_0, 0)$ once. For the inductive case, consider any supernode X_0 with $\delta_{\mathrm{dom}_0(X_0)}(\hat{v}, X_0) \le q\cdot\Delta/r$, such that the sequence of supernodes $(X_0, \ldots, X_\ell)$ guaranteed by Claim 6.11 contains the supernode X in question as X_i. Let $C_i = \text{GrowBuffer}(\mathcal{S}_i, \mathcal{X}_i, H_i)$ be the call that expands X_i for each i.

For any supernode $X = X_i$ expanded by $C_i = \text{GrowBuffer}(\cdot, \cdot, H_i)$, Claim 6.11(1) ensures $\hat{v} \in H_i$. Consequently, at most $r-1$ calls of the form GrowBuffer$(\cdot, \cdot, H_i)$ can expand X_i where $\hat{v} \in H_i$, by Claim 6.6. For each of these calls on H_i, there are at most $r-2$ many supernodes η_i seen by H_i by Claim 6.5. For each such η_i, there are at most $\binom{2(q+r)+r-1}{r-1}$ supernodes X_{i-1} that has η_i as a q-step victim, by Corollary 6.2. Claim 6.11(2) ensures there is a q-step victim η_i of X_{i-1} such that H_i sees η_i. By induction, each $(X_{i-1}, i-1)$ receives is at most μ^{i-1} charges. In total, (X, i) receives at most

$$(r-1)\cdot(r-2)\cdot\binom{2(q+r)+r-1}{r-1}\cdot\mu^{i-1} = \mu^i$$

charges, if we set $\mu = (r-1)\cdot(r-2)\cdot\binom{2(q+r)+r-1}{r-1}$.

As for the total amount of charges on $\eta_{\hat{v}}$, Claim 6.11(3) says that $\eta_{\hat{v}}$ is a victim of X_ℓ. There are at most $\binom{2(q+r)+r-1}{r-1}$ supernodes X_ℓ that has $\eta_{\hat{v}}$ as a victim, by Corollary 6.2. Each (X_ℓ, ℓ) receives is at most μ^ℓ. The index ℓ for the threatening sequence to $\hat{v}$ may range from 0 to $r-1$. As a result, $\eta_{\hat{v}}$ receives at most $\sum_{0\le\ell\le r-1}\mu^\ell \le \mu^r$ charges. ∎

6.4 Stochastic shortcut partition [CCL+24] construct a shortcut partition from buffered cop decomposition. We slightly modify their algorithm; see Figure 9. As before, we highlight our modification in red.

Claim 6.13. *For any K_r-minor-free graph G and parameter ε in $(0,1)$, the procedure* Shortcut(G, ε) *outputs an $(O(\varepsilon), O_r(1)/\varepsilon)$-shortcut partition for G.*

Proof. The proof of correctness is almost exactly the same as in Section 4 of [CCL+24]. There is one change. Lemma 4.2 of [CCL+24] states that each cluster created by the Shortcut procedure has (strong) diameter at most 4Δ. With our stochastic procedure, this is no longer true; instead, we now show a diameter bound of 12Δ. Indeed, let η be a supernode and let v be a supernode that is assigned to some cluster C_x. Let P be a shortest-path

Shortcut(G, ε)**:** Input a K_r-minor-free graph G, and a number ε in $(0, 1)$. Output an $(O(\varepsilon), O_r(1/\varepsilon))$-shortcut partition of G.

1. *Construct buffered cop decomposition.*

   Let $\Delta \coloneqq \varepsilon \cdot \mathrm{diam}(G)$. Let $\mathcal{T}$ be a stochastic $(4{\cdot}\Delta, \Delta/r, r-1)$-buffered cop decomposition sampled from the distribution of Theorem 6.1.

2. *Choose net points.*

   For each supernode η in $\mathcal{T}$, let T_η be the skeleton of η. Greedily construct a point set N_η to be a Δ-net of T_η; that is, N_η is a subset of vertices in T_η such that (1) for all vertices v in T_η, we have $\delta_{T_\eta}(v, N_\eta) \le \Delta$, and (2) for any two vertices x_1 and x_2 in N_η, we have $\delta_{T_\eta}(x_1, x_2) \ge \Delta$.

3. *Assign the vertices to the closest net point, after a small random perturbation.*

   For each supernode η, for each net point x in N_η, choose a random value $\alpha_x \sim \mathrm{Unif}[0, \Delta]$, and initialize a new cluster C_x. We say that x is the *center* of C_x. For each vertex v in η, assign v to the cluster C_x whose center x minimizes $\delta(v, x){+}\alpha_x$, breaking ties consistently.

Figure 9: The Shortcut algorithm.

between v and x. Because we break ties consistently, every vertex along P is also assigned to the cluster C_x. We will show that $\|P\| \le 6\Delta$. By the [supernode radius] property, v is within distance 4Δ of some point v' on the skeleton T_η. By definition of the net N_η, v' is within distance Δ of some point in N_η; thus, triangle inequality implies that there is some net point x' such that $\delta_\eta(v, x') \le 5\Delta$. As $\alpha_{x'} \le \Delta$, we have $\delta_\eta(v, x') + \alpha_{x'} \le 6\Delta$. By choice of x, we have $\delta_\eta(v, x) \le 6\Delta$. We conclude that cluster C_x has strong diameter at most 12Δ.

The rest of the proof of correctness from Section 4 in [CCL+24] carries over almost exactly, except that some constant factors increase because we only have a diameter bound of 12Δ for each cluster. In particular, original value of $9r$ in Claim 4.4 of [CCL+24] is now replaced with a bound of $25r$; the original bound of $(54r)^k$ in Lemma 4.5 of [CCL+24] is now replaced with a bound of $(150r)^k$; and finally, this larger constant is absorbed into the Big-O notation of Theorem 1.2 of [CCL+24]. ∎

Claim 6.14. *For any K_r-minor-free graph G, the stochastic procedure* Shortcut(G, ε) *produces a clustering $\mathcal{C}$ such that for any edge e in G,* $\Pr[e \text{ is cut by } \mathcal{C}] \le \beta \cdot \|e\|/(\varepsilon \cdot \mathrm{diam}(G))$ *for constant β.*

Proof. Let $\mathcal{T}$ be the stochastic buffered cop decomposition sampled in Step 1 of the Shortcut procedure. Let $e = (u, v)$ be an edge in G. If e is cut by $\mathcal{C}$, then either e is cut by $\mathcal{T}$, or both u and v belong to the same supernode η but are assigned to different clusters in Step 3 of Shortcut. By Theorem 6.1, edge e is cut by $\mathcal{T}$ with probability at most $\beta' \cdot \|e\|/(\varepsilon \cdot \mathrm{diam}(G))$, for some constant β'. Thus, it suffices to bound the probability that e is cut by $\mathcal{C}$, given that u and v belong to the same supernode in $\mathcal{T}$.

Let η denote the supernode in $\mathcal{T}$ that contains both u and v. By the diameter bound proven above in the proof of Claim 6.13, u (resp. v) is assigned to some cluster C_x with cluster center satisfying $\delta_\eta(u, x) \le 6\Delta$ (resp. $\delta_\eta(v, x) \le 6\Delta$). We first claim that there are at most $13(r-2) = O(r)$ cluster centers within distance 6Δ of u (resp. v), where distance is measured with respect to η — these are the "threatening" cluster centers. Suppose otherwise, for contradiction. As the skeleton T_η consists of at most $r-2$ shortest paths, pigeonhole principle implies that at least 14 cluster centers $\{x_1, \ldots, x_{14}\}$ within distance 6Δ of u lie on a single shortest path in T_η. By definition of the net N_η, the distance between any two cluster center is at least Δ; thus, the distance between furthest two cluster centers in $\{x_1, \ldots, x_{14}\}$ is at least 13Δ. However, triangle inequality implies that all these cluster centers are within distance 12Δ of each other (as each one is within distance 6Δ of v), a contradiction.

Now let C_{x_1} and C_{x_2} be two clusters with $C_{x_1} \ne C_{x_2}$, such that the cluster center x_1 (resp. x_2) is within distance 6Δ of u (resp. v). We claim that u is assigned to C_1 and v is assigned to C_2 with probability at most $\|e\|/(\varepsilon \cdot \mathrm{diam}(G))$. Indeed, if u is assigned to C_1 and v is assigned to C_2, we must have

$$\delta_\eta(u, x_1) + \alpha_{x_1} \in (\delta_\eta(v, x_2) + \alpha_{x_2} - \|e\|, \delta_\eta(v, x_2) + \alpha_{x_2} + \|e\|)\,.$$

Otherwise, following the argument from Case 3 of Claim 6.2, triangle inequality implies that either $\delta_\eta(v, x_1)+\alpha_{x_1} < \delta_\eta(v, x_2) + \alpha_{x_2}$ and so v is not assigned to C_2), or $\delta_\eta(u, x_2) + \alpha_{x_2} < \delta_\eta(u, x_1) + \alpha_{x_1}$ and so u is not assigned to C_1. This interval has length $2\|e\|$; as α_{x_1} is chosen (independently of α_{x_2}) uniformly from an interval of length $\Delta = \varepsilon \cdot \mathrm{diam}(G)$, u is assigned to C_1 and v is assigned to C_2 with probability at most $\|e\|/(\varepsilon \cdot \mathrm{diam}(G))$.

We argued above that there are $O(r)$ possible choices for C_{x_1} and $O(r)$ possible choices for C_{x_2}. By applying a union bound over all $O(r^2)$ pairs, we conclude that e is cut by $\mathcal{C}$ with probability at most $O(r^2) \cdot \|e\|/(\varepsilon \cdot \mathrm{diam}(G))$. This proves the claim. □

Claims 6.13 and 6.14 together prove Lemma 6.1.

References

[AABV95] Baruch Awerbuch, Yossi Azar, Avrim Blum, and Santosh Vempala. Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. In *Proceedings of the twenty-seventh annual ACM Symposium on Theory of Computing*, STOC '95, page 277–283, 1995. doi:10.1145/225058.225139.

[AGG+14] I. Abraham, C. Gavoille, A. Gupta, O. Neiman, and K. Talwar. Cops, robbers, and threatening skeletons: Padded decomposition for minor-free graphs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, page 79–88, 2014. doi:10.1145/2591796.2591849.

[AGMW10] I. Abraham, C. Gavoille, D. Malkhi, and U. Wieder. Strong-diameter decompositions of minor free graphs. *Theory of Computing Systems*, 47(4):837–855, 2010. doi:10.1007/s00224-010-9283-6.

[AK06] Sanjeev Arora and George Karakostas. A $2 + \epsilon$ approximation algorithm for the k-MST problem. *Mathematical Programming*, 107(3):491–504, 2006. doi:10.1007/s10107-005-0693-1.

[AR98] Sunil Arya and Hariharan Ramesh. A 2.5-factor approximation algorithm for the k-MST problem. *Information Processing Letters*, 65(3):117–118, 1998. doi:10.1016/S0020-0190(98)00010-6.

[Bar96] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 184–193, 1996. doi:10.1109/SFCS.1996.548477.

[Bar98] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 161–168, 1998. doi:10.1145/276698.276725.

[BRV99] Avrim Blum, R Ravi, and Santosh Vempala. A constant-factor approximation algorithm for thek-mst problem. *Journal of Computer and System Sciences*, 58(1):101–108, 1999. doi:10.1006/jcss.1997.1542.

[CCL+23] Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenkovic, Shay Solomon, and Cuong Than. Covering planar metrics (and beyond): O(1) trees suffice. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2231–2261, 2023. arXiv:2306.06215, doi:10.1109/FOCS57990.2023.00139.

[CCL+24] Hsien-Chih Chang, Jonathan Conroy, Hung Le, Lazar Milenković, Shay Solomon, and Cuong Than. *Shortcut Partitions in Minor-Free Graphs: Steiner Point Removal, Distance Oracles, Tree Covers, and More*, page 5300–5331. SODA '24. 2024. arXiv:2308.00555, doi:10.1137/1.9781611977912.191.

[CG04] Douglas E. Carroll and Ashish Goel. Lower bounds for embedding into distributions over excluded minor graph families. In *Algorithms – ESA 2004, 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004, Proceedings*, pages 146–156, 2004. doi:10.1007/978-3-540-30140-0_15.

[CJLV08] Amit Chakrabarti, Alexander Jaffe, James R. Lee, and Justin Vincent. Embeddings of topological graphs: Lossy invariants, linearization, and 2-sums. In *The Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS' 08.*, 2008. doi:10.1109/focs.2008.79.

[CLPP23] Vincent Cohen-Addad, Hung Le, Marcin Pilipczuk, and Michał Pilipczuk. Planar and minor-free metrics embed into metrics of polylogarithmic treewidth with expected multiplicative distortion arbitrarily close to 1. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science*, FOCS '23, pages 2262–2277, 2023. arXiv:2304.07268, doi:10.1109/FOCS57990.2023.00140.

[Coh22] Vincent Cohen-Addad. Bypassing the surface embedding: approximation schemes for network design in minor-free graphs. In *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 343–356. ACM, 2022. doi:10.1145/3519935.3520049.

[Fil19] Arnold Filtser. On strong diameter padded decompositions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, pages 6:1–6:21, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.6.

[Fil24a] Arnold Filtser. On sparse covers of minor free graphs, low dimensional metric embeddings, and other applications, 2024. arXiv:2401.14060.

[Fil24b] Arnold Filtser. Scattering and sparse partitions, and their applications. *ACM Trans. Algorithms*, Jun 2024. doi:10.1145/3672562.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004. doi:10.1016/j.jcss.2004.04.011.

[Gar96] Naveen Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 302–309, 1996. doi:10.1109/SFCS.1996.548489.

[Gar05] Naveen Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In *Proceedings of the 37th annual ACM symposium on Theory of Computing*, STOC '05, page 396–402. ACM, 2005. doi:10.1145/1060590.1060650.

[KPR93] Philip Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the 25th annual ACM symposium on Theory of Computing, STOC' 93*, 1993. doi:10.1145/167088.167261.

[RSM+96] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees—short or small. *SIAM Journal on Discrete Mathematics*, 9(2):178–200, May 1996. doi:10.1137/s0895480194266331.