Spanning Tree Embeddings Are Not Much Harder than Hierarchical Partitions

Willem Fletcher
Brown University
willem_fletcher@brown.edu

D Ellis Hershkowitz
Brown University
delhersh@gmail.com

Abstract

Embeddings of graph distances into trees is one of the most important techniques in algorithms. Often, applications of these embeddings require that the trees into which distances are embedded are spanning trees of the input graph. However, this requirement makes embedding much more difficult. This is largely due to the fact that embedding graph distances into hierarchical partitions of the graph is well-understood and, in turn, embedding hierarchical partitions into non-spanning trees is trivial. On the other hand, no such embedding of hierarchical partitions into spanning trees is known.

In this work, we show that, generally speaking, hierarchical partitions of graphs embed into spanning trees. In particular, we show that every hierarchical partition embeds into a spanning tree with $O(1)$-distortion as long as the diameter of each level of the hierarchy increases by $\Omega(\log n)$. As an immediate consequence of our embedding, we improve the best-known approximation for universal Steiner trees (USTs) from $O(\log^7 n)$ to $O(\log^5 n)$. Likewise, our embedding gives a new and extremely simple probabilistic tree embedding into spanning trees with distortion $O(\log^2 n)$ whose analysis is essentially the same as the classic (non-spanning) tree embedding of Bartal (FOCS'96, ESA'04). Our embeddings also give the first construction of Ramsey spanning trees from the Ramsey partitions of Mendel and Naor (FOCS'06) and the first non-trivial embeddings of hop-constrained distances into spanning trees.

Contents

1 Introduction

Embeddings of graph distances into trees are among the most flexible and powerful techniques in the modern algorithms toolkit. By allowing an algorithm to "pretend" that the input graph is a tree, such embeddings have paved the way for numerous results across algorithms. For example, such embeddings have found applications in areas as varied as clustering [13], dynamic algorithms[31, 36], network design [5, 32, 38], online algorithms [49, 4, 35, 37], routing algorithms [50, 29], distance oracles [2, 14, 28], solving symmetric diagonally dominant (SDD) linear systems [24] and almost-linear time flow [22], among many others.

The precise sense in which the input graph distances are represented by a tree gives rise to different kinds of embeddings. For example, up to some incurred distortion:

- **Probabilistic tree embeddings** preserve distances in expectations [7, 27] (see Definition 11);
- **Ramsey-type embeddings** preserve distances for a large fraction of nodes [10, 2, 38, 28];
- **Clan embeddings** preserve distances between "copies" of each pair of nodes [9, 29, 37];
- **Tree covers** preserve each pair's distance in one of a small number of trees [11, 18];
- **Universal Steiner trees (USTs)** preserve distances among subsets of nodes, i.e. optimal Steiner tree costs [16, 42, 17] (see Definition 8).

Across these embeddings, it is often desirable that the trees into which graph distances are embedded are not just arbitrary trees but instead *spanning trees* of the embedded graph. In particular, several works which embed graph distances into trees—including those in [35, 2, 20, 51, 43, 45, 17, 16]—only work when the embedding is into a spanning tree of the input graph.

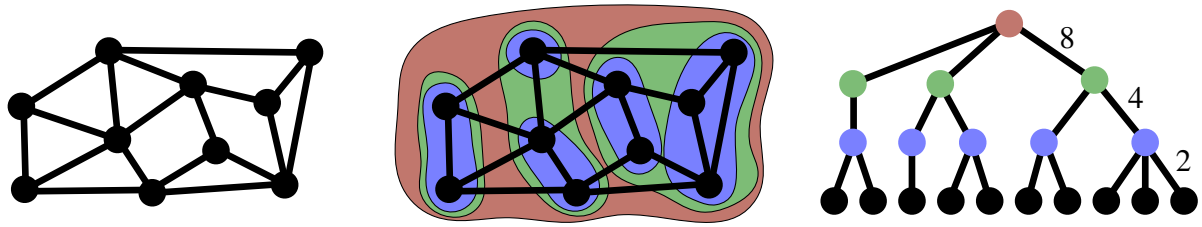
Unfortunately, embedding into spanning trees instead of arbitrary trees typically makes embedding much more difficult. For example, Alon et al. [3] initiated the study of probabilistic tree embeddings, showing that $\exp(O(\sqrt{\log n \log\log n}))$-distortion probabilistic tree embeddings on n node graphs using spanning trees are possible. Shortly thereafter, Bartal [6] dramatically improved this by showing that $\mathsf{poly}(\log n)$ distortion (specifically, $O(\log^2 n)$) is possible *if we embed into arbitrary trees*. However, it took about another 10 years until this $\mathsf{poly}(\log n)$-distortion was shown to be possible for spanning trees: Elkin et al. [26] showed $O(\log^2 n \log\log n)$ distortion with spanning trees is possible which was then improved to $O(\log^2 n)$ by Dhamdhere et al. [25]. Currently, the best-known (and best possible) tree embeddings that embed into arbitrary trees achieve distortion $O(\log n)$ and have a reasonably straightforward analysis based on random permutations [27]. On the other hand, the best probabilistic tree embeddings that embed into spanning trees only achieve distortion $O(\log n \log\log n)$ and require the sophisticated notion of petal-decompositions [1].

Similarly, it has been known for almost 20 years that, if we do not require spanning trees, USTs with $\mathsf{poly}(\log n)$ distortion are possible [42]. However, it was only very recently shown by Busch et al. [17] that $\mathsf{poly}(\log n)$-distortion USTs that are spanning trees are possible. Moreover, the best non-spanning USTs of Gupta et al. [34] achieve distortion $O(\log^2 n)$ whereas the spanning USTs of Busch et al. [17] achieve a much worse $O(\log^7 n)$ distortion.

One notable reason that embedding into trees becomes much more difficult when we require spanning trees is that the most common tree embedding technique does not work for spanning trees. In particular, most tree embedding results—including, but not limited to [6, 2, 42, 27, 37, 38]—are based on first embedding into a *hierarchical partition* and then embedding this hierarchical partition into a tree. A hierarchical partition is a series of vertex partitions where each partition coarsens the previous. Below, we say a partition $\mathcal{C}_i$ of V *coarsens* a partition $\mathcal{C}_{i-1}$ of V if for each $C_i \in \mathcal{C}_i$ and $C_{i-1} \in \mathcal{C}_{i-1}$, if $C_i \cap C_{i-1} \neq \emptyset$ then $C_{i-1} \subseteq C_i$.

Definition 1 (Hierarchical Partition)**.** *Given edge weighted graph $G = (V, E, w)$, a hierarchical partition is a sequence of vertex partitions $\mathcal{H} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k)$ where $\mathcal{C}_0 = \{\{v\} : v \in V\}$ is the singleton vertex partition, $\mathcal{C}_k = \{V\}$ is all vertices and $\mathcal{C}_i$ coarsens $\mathcal{C}_{i-1}$ for every $i > 0$.*

See Figure 1a/1b for an illustration. We will often refer to the parts of $\cup_i \mathcal{C}_i$ as "clusters" of $\mathcal{H}$.



(a) Input graph G. **(b)** Hierarchical Partition $\mathcal{H}$ of $\mathcal{G}$. **(c)** Embedding of $\mathcal{H}$ into tree.

Figure 1: An illustration of an input graph G, a 2-growing hierarchical partition $\mathcal{H}$ of G and an $O(1)$-distortion embedding of $\mathcal{H}$ into a tree that is not a spanning tree of G.

In order to formalize the sense in which graph distances can be embedded into a hierarchical partition, we must first define the distances associated with a hierarchical partition. In particular, typically these hierarchical partitions are growing in the sense that their diameter multiplicatively increases from one level to the next. Below, we say that a vertex partition of edge-weighted graph G has strong diameter at most D if for each $C \in \mathcal{C}$, graph $G[C]$ has diameter at most D.[1]

Definition 2 (γ-Growing Hierarchical Partition)**.** *Given hierarchical partition $\mathcal{H} = (\mathcal{C}_0, \mathcal{C}_1, \ldots)$ of graph $G = (V, E, w)$, we say $\mathcal{H}$ is γ-growing if each $\mathcal{C}_i$ has strong diameter at most γ^i.*

γ-growing hierarchical partitions naturally induce distances between nodes. In particular, the distance induced by a hierarchical partition between two nodes is γ^i where i is the first level where u and v are in the same part of a partition.

Definition 3 (Distances Induced by a Hierarchical Partition)**.** *Given a γ-growing hierarchical partition $\mathcal{H} = (\mathcal{C}_0, \mathcal{C}_1, \ldots)$ of graph (V, E, w), the distance induced by $\mathcal{H}$ between $u, v \in V$ is*

$$d_{\mathcal{H}}(u, v) := \gamma^{i(u,v)}$$

where $i(u, v)$ is the minimum i such that there is a $C \in \mathcal{C}_i$ containing both u and v.

By choosing each vertex partition of a hierarchical partition to satisfy nice properties, one can typically approximate graph distances by those of some hierarchical partition. For example, Bartal [6] takes each partition to be a random "low-diameter decomposition." The result is a distribution over hierarchical partition that approximates distances in expectation. Likewise, Jia et al. [42] constructs USTs by choosing each partition to be a "strong sparse partition." The result is a hierarchical partition that approximates Steiner tree costs.[2]

Once one has embedded into a hierarchical partition, as long as we do not require that the tree into which we embed is a spanning tree, it is trivial to then embed the hierarchical partition $\mathcal{H}$ into a tree. Specifically, we define the distortion of an embedding of a hierarchical partition into a spanning tree as follows.[3]

[1] The diameter of a graph is the maximum distance between two nodes and given edge-weighted graph $G = (V, E, w)$, $G[C] := (C, \{e \in E : e \subseteq C\}, w)$ is the induced graph on C.

[2] We note that for both results a weak diameter rather than strong diameter guarantee as required by Definition 2 is sufficient. $\mathcal{C}$ has weak diameter at most D if for each $C \in \mathcal{C}$, each $u, v \in C$ are at distance at most D in G.

[3] Below and throughout this work we let $d_G(u, v)$ give the shortest-path distance in G between vertices u and v.

Definition 4. *Given a γ-growing hierarchical partition $\mathcal{H}$ of edge-weighted graph $G = (V, E, w)$, we say that spanning tree $T \subseteq G$ is an α-distortion embedding of $\mathcal{H}$ if for all distinct $u, v \in V$, we have*

$$d_T(u, v) \leq \alpha \cdot d_{\mathcal{H}}(u, v).$$

If we do not require that T is a spanning tree then we can embed a a γ-growing hierarchical partition into a tree with an $O(1)$ increase in distance by taking a tree which has a node for each part of each partition where part $C \in \mathcal{C}_i$ has as children the parts of $\mathcal{C}_{i-1}$ which it contains and a parent edge of weight $\approx \gamma^i$. See Figure 1c. Typically, combining this embedding with an embedding of graph distances into a hierarchical partition gives a simple way of embedding graph distances into (non-spanning) trees. For instance, Bartal [6] used this embedding and their above-mentioned embeddings into hierarchical partitions to give probabilistic tree embeddings into non-spanning trees. Likewise, Jia et al. [42] used this fact and their above-mentioned embeddings into hierarchical partitions to give non-spanning USTs.

Unfortunately, embedding a hierarchical partition into a *spanning* tree is significantly more challenging. One might naturally hope to use a strategy similar to the non-spanning tree case by choosing a spanning tree whose structure reflects the structure of the hierarchical partition. However, it is known that there is a hierarchical partition $\mathcal{H}$ such that any spanning tree T that "strictly obeys" the structure of $\mathcal{H}$ must incur distortion $\omega(\mathsf{poly}(\log n))$ (see Theorem 7 of Busch et al. [16]). Here, strictly obeys means that $T[C]$ is connected for every cluster C of $\mathcal{H}$. As such, and unlike the non-spanning tree case, we necessarily must find a spanning tree that deviates from the structure of the hierarchical partition. In fact, to our knowledge, the only known embedding of hierarchical partitions into spanning trees is one which requires that the hierarchical partition be a highly structured hierarchical partition. For example, for a hierarchy of strong sparse partitions (which are parameterized by α and τ; see Definition 49 for a formal definition), it is known that $7\alpha\tau$-distortion embeddings into spanning trees are possible as long as the hierarchical partition is $\Omega(\log n)$-growing (see Lemma 8 of Busch et al. [16]). By Busch et al. [17], α and τ are parameters known to satisfy $\alpha\tau = \Theta(\log^2 n)$. See also [19] for similar results for doubling metrics. However, for general hierarchical partitions, no embeddings into spanning trees are known.

In summary, the typical framework for embedding graph distances into trees first embeds graph distances into a hierarchical partition and then applies the trivial embedding of a hierarchical partition into a (non-spanning) tree. This framework ceases to work when we require an embedding into spanning trees because there is no known embedding of hierarchical partitions into spanning trees. Indeed, there are impossibility results that suggest that the structure of an embedding of hierarchical partitions into spanning trees would have to be somewhat counter-intuitive.

1.1 Our Contributions

In this work, we show that hierarchical partitions embed into spanning trees.

Specifically, we show that any γ-growing hierarchical partition embeds into a spanning tree with distortion $O(1)$ as long as $\gamma = \Omega(\log n)$. We then use this embedding to improve the best universal Steiner tree distortion from $O(\log^7 n)$ to $O(\log^5 n)$. Furthermore, we show how, leveraging the classic hierarchical partitions of Bartal [7, 8], this gives simple $O(\log^2 n)$-distortion probabilistic tree embeddings into spanning trees. Our embedding also allows us to construct Ramsey spanning trees from so-called Ramsey partitions as introduced by Mendel and Naor [47] with an $O(\log n)$ overhead; this gives a much simpler construction of Ramsey spanning trees than that of Abraham et al. [2], albeit with an extra $O(\log n/\log\log n)$ factor. Lastly, using our embedding we give the first non-trivial embedding of hop-constrained distances of a graph into spanning trees—previously such results were only known for embedding hop-constrained distances into non-spanning trees [38, 28].

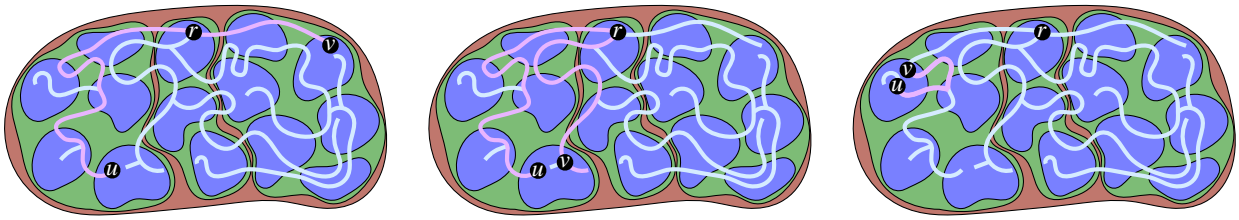
1.1.1 Embedding Hierarchical Partitions into Spanning Trees

More formally, our main result embedding hierarchical partitions into spanning trees is as follows.[4]

Theorem 5 (Hierarchical Partitions Embed into Spanning Trees)**.** *Any γ-growing hierarchical partition of n-node graph $G = (V, E, w)$ where $\gamma \geq 203 \log n$ can be embedded into a spanning tree $T \subseteq G$ with distortion at most 407 (in poly-time with high probability).*

Note that, as mentioned above, a similar but much weaker result was previously shown in [16] and [17]. In particular, if the input hierarchical partition is one of the strong sparse partition hierarchies of Busch et al. [17], then the above result is possible with 407 replaced by $O(\log^2 n)$.

Before moving on to the rest of our results, we discuss the intuition for how we show how to embed an $\Omega(\log n)$-growing hierarchical partition $\mathcal{H} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k)$ into a spanning tree T. Specifically, we discuss two incorrect embeddings, the intuition that can be gotten from them and how we apply this intuition for our embedding.



(a) Good distortion. **(b)** Bad distortion. **(c)** Good distortion.

Figure 2: An illustration of which pairs of vertices u and v a shortest path tree wtih root r provides good or bad distortion for as an embedding of a hierarchical partition. Path between u and v highlighted in pink.

Bad Embedding 1: Shortest Path Tree. Consider, for a moment, the following natural but bad embedding of $\mathcal{H}$: choose an arbitrary root r and then choose for T a shortest path tree for r (without regard for the structure of $\mathcal{H}$). For such a tree T, we would have that any two nodes u and v, for which V is the smallest cluster containing both u and v, have their distances well-approximated by T. In particular, if u and v satisfied $i(u, v) = k$ (see Definition 3 for the notation), then we would have that $d_{\mathcal{H}}(u, v) = \gamma^k$ and so

$$d_T(u, v) \leq d_T(u, r) + d_T(v, r) = d_G(u, r) + d_G(v, r) \leq 2\gamma^i = 2d_{\mathcal{H}}(u, v),$$

giving $d_T(u, v) \leq 2d_{\mathcal{H}}(u, v)$. See Figure 2a.

Where such a solution potentially fails (possibly arbitrarily badly) is for two nodes for which the smallest cluster containing them is much smaller than V, i.e. for two nodes u and v for which $i(u, v) \ll k$. In particular, if nodes u and v satisfy $i(u, v) \ll k$, but the path connecting them in T goes through r, the distance between u and v in T can be up to $2\gamma^i$ (see Figure 2b), meaning

$$d_{\mathcal{H}}(u, v) = \gamma^{i(u,v)} \ll 2\gamma^i = d_T(u, v).$$

However, a shortest path tree does not fail for all u and v such that $i(u, v) \ll k$. In particular, let C be the smallest cluster containing both u and v in $\mathcal{H}$ and observe that—even if $i(u, v) \ll k$—as

[4] Henceforth, we say a function is $\mathsf{poly}(n)$ if it is at most $O(n^c)$ for some constant c and we let "with high probability" mean with probability at least $1 - 1/\mathsf{poly}(n)$.

long as u and v lie on the same root-to-leaf path in T we will have

$$d_T(u,v) = d_G(u,v) \leq d_{G[C]}(u,v) \leq d_{\mathcal{H}}(u,v)$$

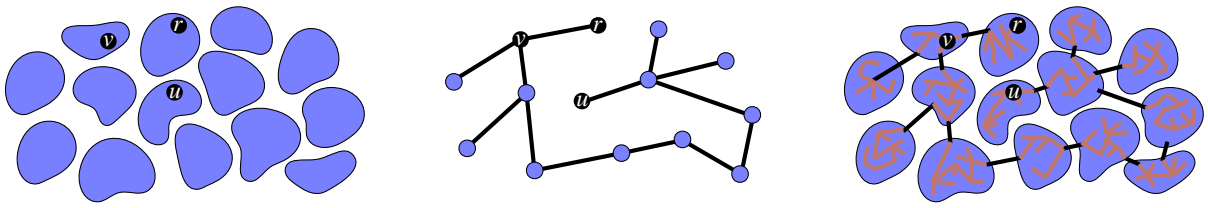
where the first equality follows because a root-to-leaf path of T is a shortest path of G and therefore so is any subpath of it, the second equality holds because $G[C] \subseteq G$ and the last equality holds because $d_{\mathcal{H}}(u,v) = \gamma^{i(u,v)}$ must be an upper bound on the strong diameter of C. See Figure 2c.

Generalizing this, observe that if our spanning tree T could be partitioned into shortest paths of G such that each cluster of $\mathcal{H}$, other than V, were intersected by at most one of these shortest paths, then T would be an $O(1)$-distortion embedding of $\mathcal{H}$. Obtaining exactly this property is too much to ask for, but our embedding, generally speaking, draws on this idea.

Specifically, the rough idea of our approach will be to construct a T that decomposes into shortest paths. These shortest paths will satisfy the property that for any given cluster C of $\mathcal{H}$, this cluster is intersected (in a non-trivial way) by at most one shortest path. However, these will not necessarily be shortest paths in G as above but, more generally, a shortest path in some cluster of $\mathcal{H}$ which contains C and so corresponds to distances at most those of C. The notable remaining challenge, then, is to decide how to select shortest paths while still ensuring that the final output of our algorithm is a spanning tree.

Bad Embedding 2: Recursive Shortest Path Trees. Another natural but bad embedding naively leverages the (laminar) structure of $\mathcal{H}$ for recursion. Specifically, suppose we are currently considering some cluster $C \in \mathcal{C}_{i+1}$ of $\mathcal{H}$. Then, let $\tilde{T}$ be a shortest path tree of $G[C]$ after contracting each cluster of $\mathcal{C}_i$. Return as our solution the union of $\tilde{T}$ and the result of recursing on each cluster of $\mathcal{C}_i$. See Figure 3. Such an embedding is trivially a spanning tree. Furthermore, notice that in the returned spanning tree any two vertices are either in the same cluster of $\mathcal{C}_i$ and so handled by recursion or lie on some shortest path of $\tilde{T}$.

In the latter case these vertices lie on a shortest path of $\tilde{T}$. However, this is a shortest path in a metric that could be distorted multiplicatively by γ as compared to distances in $G[C]$ since we contracted clusters of $\mathcal{C}_i$. Consequently, such a construction would lose a γ factor at each level, leading to a distortion of $\approx \gamma^k$. Summarizing, the basic issue here is that, in going from one node u to another node v in our shortest path tree $\tilde{T}$, we may have to "hop across" the output of a large number of recursive calls. Thus, if we are to use recursion, what is needed is a way of selecting shortest paths that provides control on the total number of traversals of recursive calls.



(a) Input $\mathcal{C}_i$ and root. **(b)** SPT of $\mathcal{C}_i$ contracted. **(c)** Ouput of recursive calls.

Figure 3: An illustration of the problem with the naive recursive embedding. 3a gives the input partition. 3b gives the result of taking a shortest path tree (SPT) after contracting the partition. 3c gives the result after recursing and uncontracting (result of recursion in red). Notice that u and v have to hop across a large number of recursively-computed trees despite being on a shortest path in the shortest path tree.

Our Embedding, Roughly. Our embedding does exactly this. Specifically, as above, our embedding will be recursive. Suppose we are currently considering some cluster $C \in \mathcal{C}_{i+1}$. To construct our solution for this cluster, we will first add to our solution a collection of shortest paths in $G[C]$ to a root such that each cluster (of every partition of our hierarchy $\mathcal{H}$, not just of $\mathcal{C}_i$) is intersected (in a non-trivial way) by at most one chosen shortest path. By our discussion in the first bad embedding, such a collection of paths can be added to our solution in a way consistent with $O(1)$ distortion. We then recurse on each cluster of $\mathcal{C}_i$ contained in C. The paths we add to our solution will be carefully chosen so that the number of recursive calls a node will have to hop across will be bounded, unlike in the second bad embedding.

In more detail, our algorithm for $C \in \mathcal{C}_{i+1}$ is roughly as follows. We receive as input a root $r \in C$. We then choose an arbitrary representative for each cluster of $\mathcal{C}_i$ and let $\mathcal{P}$ be the shortest paths of all representatives to r in $G[C]$.

It is possible that, for any given cluster of $\mathcal{C}_i$, there are many paths of $\mathcal{P}$ which intersect it. Thus, we next "aggregate" the paths of $\mathcal{P}$ so that at most one path (non-trivially) intersects each cluster (see Figure 4). In particular, we repeatedly fix a uniformly random permutation π on the paths of $\mathcal{P}$. If a path P occurs earlier in π than every other path of $\mathcal{P}$ which has a cluster of $\mathcal{C}_i$ in common with P, then we truncate all paths of $\mathcal{P}$ up until they first hit a cluster that also intersects P. Note that a path may get truncated multiple times. We show that the result after this process is a collection of shortest paths where if we were to contract $\mathcal{C}_i$, what we get is not necessarily a tree but a graph in which every node has a path to the root consisting of at most $O(\log n)$ of these truncated shortest paths. As such, the number of hops across clusters of $\mathcal{C}_i$ that we perform is bounded by $O(\log n)$. We abstract this away with what we call the *dipath aggregation problem*—see Section 2—and note that by a binary-tree-like instance, the above $O(\log n)$ is a best-possible bound for this problem.

We then recurse on each cluster of $\mathcal{C}_i$ contained in C. The root we use in each recursive call is the contraction of our above truncated paths. Since we have bounded the number of cluster hops by $O(\log n)$, we know that in order to reach the root in C, a given vertex must hop across at most $O(\log n)$ recursive calls. As long as $\gamma = \Omega(\log n)$, we will be able to afford to pay this many recursive traversals with overall constant distortion (this is why we require $\gamma = \Omega(\log n)$). See Section 3 for our final recursive algorithm which uses our solution to the dipath aggregation problem.

Notably, the final output spanning tree T of our algorithm may deviate significantly from the structure of the input hierarchical partition $\mathcal{H}$. In particular, for a given cluster C of $\mathcal{H}$, it is not necessarily the case that $T[C]$ is even connected; for example, a given selected shortest path could leave and reenter a given cluster C arbitrarily-many times. As such, our embedding does not violate the aforementioned impossibility result of Busch et al. [16]. Nonetheless, we can show that T is a good approximation for the distances of $\mathcal{H}$, even without obeying the structure of $\mathcal{H}$.

1.1.2 Embedding Laminar Families into Spanning Trees

Given a graph G and distances $d_{\mathcal{H}}$ induced on G by a γ-growing hierarchy $\mathcal{H}$, Theorem 5 tells us that we can embed G into a subtree with constant loss. A more general way of inducing distances on a graph is with a laminar family.

Definition 6 (Laminar Family Distance)**.** *A laminar family $\mathcal{L}$ on graph $G = (V, E, w)$ is a set of subsets of $V(G)$ such that if $C_0, C_1 \in \mathcal{L}$, then $C_1 \subseteq C_2$, $C_2 \subseteq C_1$, or $C_1 \cap C_2 = \emptyset$. We will assume $\{G\}$ and all singletons are in $\mathcal{L}$. If $\mathcal{L}$ is a laminar family then it defines distances on $V(G)$ by*

$$d_{\mathcal{L}}(u, v) = \Delta(G[C_{u,v}])$$

where $C_{u,v}$ is the part in $\mathcal{L}$ containing both u and v with smallest strong diameter.

In Section 4 we show that any such laminar family can be turned into a γ-growing hierarchy with distortion γ. Then, as a corollary of Theorem 5, we are able to embed laminar families into trees with $O(\log n)$ distortion.

Theorem 7 (Laminar Families Embed into Spanning Trees)**.** *Any laminar family on n-node graph $G = (V, E, w)$ can be (poly-time with high probability) embedded into a spanning tree $T \subseteq G$ with distortion $O(\log n)$.*

1.1.3 Improved Universal Steiner Trees

Next, we use our embeddings to improve the approximation guarantees of the best universal Steiner trees (USTs) from $O(\log^7 n)$ to $O(\log^5 n)$. The basic idea is that we are able to improve embeddings of strong sparse partition hierarchies into spanning trees by using our embedding.

In particular, given an edge weighted graph $G = (V, E, w)$ and a set of terminals, $S \subseteq V$, a Steiner tree is a tree $T \subseteq G$ connecting all terminals. The weight of a Steiner tree, T, is defined as $w(T) = \sum_{e \in T} w(e)$. For a set of terminals, S, we denote the minimum weight over all Steiner trees connecting S with OPT_S. USTs were introduced by Jia et al. [41] to model designing a network in which repeated broadcast occurs. Roughly speaking, a UST is a single spanning tree that is a good approximation for every instance of Steiner tree. More formally, a UST is as follows.

Definition 8 (ρ-Approximate Universal Steiner Tree)**.** *Given an edge weighted graph $G = (V, E, w)$ and a root $r \in V$, a ρ-approximate universal Steiner tree is a spanning tree T of G such that for all terminal sets $S \subseteq V$ with $r \in S$ we have*

$$w(T\{S\}) \leq \rho \cdot \mathsf{OPT}_S$$

where $T\{S\}$ is the minimal sub-tree of T connecting S.

Note that we must include r in S since, otherwise, the n-node cycle shows that $\rho = o(n)$ is impossible.

Jia et al. [41] showed that if we do not require our UST to be a spanning tree then $\tilde{O}(\log^4 n)$-approximate USTs are possible which was later improved to $O(\log^2 n)$ by Gupta et al. [34]. Later, Busch et al. [16] showed that $2^{O(\sqrt{\log n})}$-approximate USTs that are spanning trees are possible which was recently improved to $O(\log^7 n)$ by Busch et al. [17]. We improve this to $O(\log^5 n)$.

Theorem 9. *For any edge weighted graph $G = (V, E, w)$ and root $r \in V$, there exists an $O(\log^5 n)$-approximate UST (that is poly-time computable with high probability).*

All previous results—ours included—are based on the idea of embedding strong sparse partition hierarchies into spanning trees. We summarize previous work as follows.

Lemma 10 ([16, 17])**.** *If there exists a (poly-time computable) embedding of a γ-growing hierarchical partition for $\gamma = \Theta(\log^2 n)$ into a spanning tree with distortion α, then there exists an $O(\alpha \cdot \log^5 n)$-approximate UST (that is poly-time computable).*

For completeness, we explicitly prove the above Lemma 10 in Appendix A using results of Busch et al. [16] and Busch et al. [17]. Our $O(\log^5 n)$-approximate USTs (Theorem 9) are immediate from the above Lemma 10 and our $O(1)$-distortion embeddings of hierarchical partitions into spanning trees as given by Theorem 5. Note that, without an entirely new approach to USTs or further improvements to strong sparse partition hierarchies, this bound is best-possible.

1.1.4 New Probabilistic Tree Embeddings into Spanning Trees

We next show that our embedding of hierarchical partitions into spanning trees reduce probabilistic spanning tree embeddings to embedding into hierarchical partitions. Combining this with known embeddings into hierarchical partitions gives simple new probabilistic spanning tree embeddings.

We first define a probabilistic embedding where any $d : V \times V \to \mathbb{R}_{\geq 0}$ is a "distance function".

Definition 11 (Probabilistic Embedding)**.** *Given edge-weighted graph $G = (V, E, w)$, an α-distortion probabilistic embedding is a distribution over distance functions $\mathcal{D}$ where*

- ***Non-Contracting:*** *$d_G(u, v) \leq d(u, v)$ for every $d \in \mathcal{D}$ and $u, v \in V$ and;*
- ***α-Distortion:*** *$\mathbf{E}_{d\sim\mathcal{D}}[d(u, v)] \leq \alpha \cdot d_G(u, v)$ for every $u, v \in V$.*

If all of the support of $\mathcal{D}$ are tree metrics, then we say $\mathcal{D}$ is a *probabilistic tree embedding*.[5] If each of the corresponding trees are also spanning trees of G, then we say $\mathcal{D}$ is a *probabilistic tree embedding into spanning trees*. Lastly, if each $d \in \mathcal{D}$ is equal to $d_{\mathcal{H}}$ for some γ-growing hierarchical partition $\mathcal{H}$ of G, then we say that $\mathcal{D}$ is a *probabilistic γ-growing hierarchical partition embedding*. Note that the non-contracting property for hierarchical partitions is immediate from the fact that the definition of γ-growing hierarchical partition distances (Definition 3) is based on strong diameter.

With the above definitions in hand, we can state our reduction of probabilistic tree embeddings into spanning trees to probabilistic hierarchical partition embeddings.

Theorem 12. *Suppose graph $G = (V, E, w)$ has a (poly-time sampleable) α-distortion probabilistic γ-growing hierarchical partition embedding for $\gamma \geq 203 \log n$. Then, G has a (poly-time with high probability sampleable) $O(\alpha)$-distortion probabilistic tree embedding into spanning trees.*

Showing the above is a straightforward matter of applying our embedding of hierarchical partitions into spanning trees (Theorem 5) and the above definitions. We provide a proof in Section 5. Note that our embedding of hierarchical partitions into spanning trees deterministically bound distortion by a constant and so our result shows that, in some sense, the difficulty and need for randomization in probabilistic tree embeddings into spanning trees comes from the hierarchical partitions and not the requirement that we have a spanning tree. If we embed into Laminar families instead of γ-growing hierarchies, it is possible to get lower distortion. However, we then pick up another log when embedding the Laminar family into a spanning tree. Combining Theorem 12 and Theorem 7 we get:

Theorem 13. *Suppose graph $G = (V, E, w)$ probabilistically embeds into laminar families with α-distortion (in poly-time). Then, G has a (poly-time with high probability sampleable) $O(\alpha \cdot \log n)$-distortion probabilistic tree embedding into spanning trees.*

The upshot of the above theorems is that they give simple probabilistic subtree embeddings using the following embeddings of graphs into hierarchical partitions. In Appendix B we show how to get simple $O(\log^3 n/\log\log n)$-distortion probabilistic γ-growing hierarchical partition embedding for $\gamma \geq 203 \log n$ with minor adaptations to a well-known analysis of Bartal [7]. By theorem 12 this gives us simple $O(\log^3 n/\log\log n)$-distortion probabilistic subtree embeddings. In Section 6 we show that with minor changes to Bartal [8], any graph embeds into laminar families with $O(\log n)$-distortion (Lemma 31). Thus, by Theorem 13, we get $O(\log^2 n)$-distortion probabilistic subtree embeddings.

[5] A metric d is a tree metric if there exists a tree T such that $d = d_T$.

Theorem 14. *Every graph $G = (V, E, w)$ has an $O(\log^2 n)$-distortion probabilistic tree embedding into spanning trees (that is poly-time computable).*

While the above do not improve the $O(\log n \log\log n)$ of Abraham and Neiman [1], they are conceptually simple and different than existing $\mathsf{poly}(\log n)$-distortion embeddings into spanning trees [26, 1, 25]. In particular, they are, more or less, the same as the classic analysis of Bartal [7] and Bartal [8]. Furthermore, the above approach provides a promising new direction for closing the gap between the $O(\log n \log\log n)$-distortion probabilistic tree embeddings into spanning trees of Abraham and Neiman [1] and the $O(\log n)$-distortion non-spanning tree embeddings of Fakcharoenphol et al. [27]. Specifically, it shows that $O(\log n)$-distortion probabilistic tree embeddings into spanning trees reduces to $O(\log n)$-distortion embeddings into $\Omega(\log n)$-growing hierarchical partitions.

1.1.5 New Ramsey Spanning Tree Construction

We also show that our embeddings reduce Ramsey spanning trees to embedding into hierarchical partitions. Similar to probabilistic tree embeddings, we can combine this with known hierarchical partitions to get a simple construction of Ramsey spanning trees.

Ramsey spanning trees are similar to probabilistic tree embeddings except instead of being a distribution over spanning trees that approximately preserve distance for all pairs of vertices in expectation, it is a single spanning tree that approximately preserves distances for many pairs of vertices. We define Ramsey spanning trees formally below.

Definition 15 (Ramsey Spanning Tree)**.** *Given an edge-weighted graph $G = (V, E, w)$, $M \subseteq V$, and $k > 1$, an α-distortion M-Ramsey spanning tree is a spanning tree $T \subseteq G$ such that for some $\tilde{M} \subseteq M$ with $|\tilde{M}| \geq |M|^{1-1/k}$, for all $u \in \tilde{M}$ and $v \in V$*

$$d_T(u, v) \leq \alpha \cdot d_G(u, v).$$

The type of hierarchical partition that corresponds to a Ramsey spanning tree is called a fully padded hierarchical partition and is essentially the Ramsey partitions introduced in [47].

Definition 16 ((M, k)-Fully Padded γ-Growing Hierarchical Partition)**.** *Given an edge-weighted graph $G = (V, E, w)$, $M \subseteq V$, and $k > 1$, an (M, k)-fully padded γ-growing hierarchical partition $\mathcal{H} = (\mathcal{C}_0, \mathcal{C}_1, \ldots)$ is a γ-growing hierarchical partition such that for some $\tilde{M} \subseteq M$ with $|\tilde{M}| \geq |M|^{1-1/k}$, for all $u \in \tilde{M}$ and all i there is some $C_i \in \mathcal{C}_i$ such that*

$$B_G(u, \gamma^i/8k) \subseteq C_i.$$

We now state our reduction of fully padded hierarchical partitions to Ramsey spanning trees as Theorem 17. The proof, which consists of a simple application of Theorem 5, is in Section 7.

Theorem 17. *Suppose graph $G = (V, E, w)$ has a (M, k)-fully padded γ-growing hierarchical partition for $\gamma \geq 203 \log n$ (that we can compute in poly-time). Then, G has an $O(k\gamma)$-distortion M-Ramsey spanning tree that we can compute in poly-time with high probability.*

The existence of such hierarchies is proven in [2].

Lemma 18 ([2])**.** *For every graph $G = (V, E, w)$, $M \subseteq V$, and $k > 1$, there exists an (M, k)-fully padded $(203 \log n)$-growing hierarchical partition of G (that is poly-time computable).*

Combining Theorem 17 with Lemma 18 gives low distortion Ramsey spanning trees.

Theorem 19. *For every graph $G = (V, E, w)$, $M \subseteq V$, and $k > 1$, there exists an $O(k \cdot \log n)$-distortion M-Ramsey spanning tree that is poly-time computable with high probability.*

This does not improve on the $O(k \cdot \log \log n)$-distortion Ramsey spanning trees of [2]. However, it is far simpler and gives (to our knowledge) the first construction of Ramsey spanning trees directly from Ramsey partitions.

1.1.6 New Embeddings of Hop-Constrained Distances into Spanning Trees

Lastly, we give the first embeddings of hop-constrained distances into spanning trees. Roughly, hop constraints require that graph connections occur over paths with a small number of edges. As such, hop constraints give a natural way of modeling latency requirements in network design problems; see [15, 44, 40, 23, 38, 28, 21, 38, 28] for examples of works on hop-constrained network design.

Whereas the natural notion of distance in network design problems is the shortest path distance, the natural notion of distance in hop-constrained network design problems is the hop-constrained distance, defined as follows. Given edge-weighted graph $G = (V, E, w)$, let $\mathcal{P}_h(u, v) := \{P = (u, \ldots, v) : |P| \leq h\}$ be all u-v paths with at most h edges. Then, the h-hop-constrained distance between u and v is the minimum weight of a u-v path with at most h edges, namely:

$$d^{(h)}(u, v) := \min_{P \in \mathcal{P}_h(u,v)} w(P).$$

In addition to hop-constrained network design problems, hop-constrained distances are key to recent breakthroughs in near-linear time negative-weight single-source shortest path algorithms [30, 12] and universally optimal distributed algorithms [33, 39].

Notably, whereas shortest path distances are metric, hop-constrained distances are provably inapproximable by a metric [38]. Nonetheless, Haeupler et al. [38] showed that Ramsey-type embeddings of hop-constrained distances into (non-spanning) trees are possible, provided one is allowed to be poly-log-approximate with respect to both weights and hop constraints. Later work of Filtser [28] improved the poly-logs in these embeddings.

We give the first sub-linear quality embeddings of hop-constrained distances into *spanning* trees. We define the hop-diameter of a graph G, $\mathsf{hd}(G)$, to be the minimum h such that all pairs of vertices in G have a path between them of length at most h. The following defines these embeddings.

Definition 20 (Hop-Constrained Ramsey Subtrees)**.** *Given edge-weighted graph $G = (V, E, w)$, an h-hop-constrained Ramsey subtree distribution of G with exclusion probability ϵ, distortion α, and hop-stretch β is a distribution $\mathcal{D}$ over pairs (T, M) where T is a spanning subtree of G and $M \subseteq V(G)$, such that*

- ***Bounded Exclusion Probability:*** *For every $v \in V(G)$, $\mathbb{P}_{(T,M)\sim\mathcal{D}}(v \notin M) \leq \epsilon$ and;*
- ***Low Distortion:*** *For every $(T, M) \in \mathcal{D}$, $u \in M$, and $v \in V(G)$, $d_T(u, v) \leq \alpha \cdot d^{(h)}(u, v)$.*
- ***Low Hop-Diameter:*** *For every $(T, M) \in \mathcal{D}$, $\mathsf{hd}(T) \leq \beta \cdot \mathsf{hd}(G)$.*

Note that the bound on the diameter of these trees is natural in that it makes them non-contracting with respect to hop-constrained distances (up to the hop slack): that is, it implies that for any $u, v \in V$ and T in our support we have $d^{(\mathsf{hd}(G)\cdot\beta)}(u, v) \leq d_T(u, v)$.

The following is our embedding of hop-constrained distances into low diameter spanning trees.

Theorem 21. *For any edge-weighted graph $G = (V, E, w)$, $\epsilon \in (0, 1)$, and $h \leq O\left(\frac{\epsilon \cdot \mathsf{hd}(G)}{\log^{1.5}(n)}\right)$, let $\alpha = \tilde{O}\left(\frac{2^{\sqrt{\log n}}}{\epsilon}\right)$ and $\beta = \log^{O(\sqrt{\log n})}(n)$. Then there is a (poly-time sampleable) h-hop-constrained Ramsey subtree distribution of G with exclusion probability ϵ, distortion α, and hop-stretch β.*

Roughly, we obtain the above result by modifying the hierarchical partitions of Haeupler et al. [38] and then applying our embedding of hierarchical partitions into spanning trees (Theorem 5). Our results are sub-linear rather than poly-logarithmic because we must heavily sparsify these hierarchies in order to control the hop diameter of our spanning trees.

2 Dipath Aggregation

Towards proving our main embedding of hierarchical partitions into spanning trees (Theorem 5), we begin by providing algorithms for what we call the *dipath aggregation problem.*

In the dipath aggregation problem we are given a directed graph $G = (V, A)$ along with a root $r \in V$, and a partition $\mathcal{C}$ of $V \setminus r$. We are also given a set of terminals $T \subseteq G \setminus \{r\}$ and for each $t \in T$, a simple path p_t^* from t to r in G. Our goal is to find a set of paths $P = \{p_t\}_{t \in T}$, where each p_t is a prefix of p_t^*, such that each terminal has a unique path to the root and each cluster has at most one path leaving it. Since a cluster isn't allowed to have multiples paths leaving it, sometimes the path from a terminal to the root will have to *hop* from one path to another in a cluster. See Figure 4.

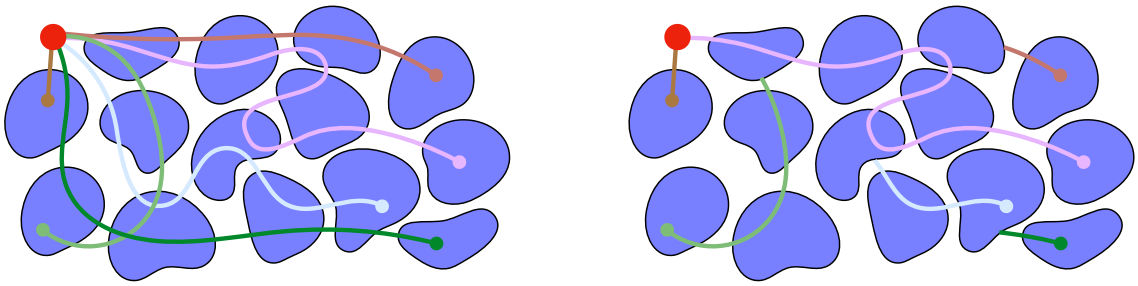


Figure 4: An instance of the dipath aggregation problem along with a 2-hop solution.

To formalize these notions we define a graph $\hat{G}_P$ by contracting each cluster $C \in \mathcal{C}$ into a single vertex. For clusters $C_0, C_1 \in \mathcal{C}$ which correspond to vertices $v_0, v_1 \in V(G_P)$, there is an arc $a = (v_0, v_1)$ of color t in $\hat{G}_P$ if for $p_t \in P$, there is an arc $a \in p_t$ with $\mathsf{tail}(a) \in C_0$ and $\mathsf{head}(a) \in C_1$.

Definition 22 (Dipath Aggregation Problem)**.** *An instance of the dipath aggregation problem consists of a directed graph G, a root $r \in G$, terminals $T \in G \setminus r$, and $P = \{p_t^*\}_{t \in T}$ where each p_t^* is a simple path from t to r. An h-hop solution is a set $P = \{p_t\}_{t \in T}$ where each p_t is a prefix of p_t^* such that:*

- ***Cluster Respecting:*** *Each vertex in $\hat{G}_P$ has arcs of at most a single color leaving it.*

- ***Few Hops:*** *If cluster C_0 contains a terminal, then there is a path in $\hat{G}_p$ from v_0 to r that switches colors at most h times.*

Since multiple arcs of the same color can leave a vertex, there may be multiple paths in $\hat{G}_p$ from v to r. However, the cluster respecting property and the fact that each $p \in P$ is a prefix means that up to monochromatic loops in $\hat{G}_p$, there is a unique path. We show that every instance of the dipath aggregation problem has an $O(\log(|T|))$-hop solution.

Theorem 23. *Every instance of the path aggregation problem has a $200\log(|T|)$-hop solution that can be computed in polynomial time with high probability.*

Note that this is tight up to constants as can be seen by considering a balanced binary tree with singleton clusters whose terminals are the leafs.

2.1 Algorithm for Dipath Aggregation

We now describe Algorithm 1 which we will show gives an $O(\log|T|)$-hop solution to any instance of the dipath aggregation problem with high probability. For readers familiar with Luby's maximal independent set (MIS) algorithm [46]: our analysis is similar in spirit to the analysis of Luby's MIS algorithm but in our case when we take a "node" into the MIS we only get to reduce the degree of its neighbors by a constant in expectation, not delete them outright.

Let $P_0 = \{p_t^*\}_{t\in T}$. We will define P_i for $i > 0$ where each $p_t \in P_i$ is a prefix of $p_t \in P_{i-1}$. We will say two paths $p_1, p_2 \in P_i$ conflict if there exist arcs $a_1 \in p_1$, $a_2 \in p_2$ and a cluster $C \in \mathcal{C}$ such that $\mathsf{tail}(a_1), \mathsf{tail}(a_2) \in C$. For $i = 1$ until there are no more paths that conflict:

- Choose a uniformly random permutation π_i of T.
- We say a path p_t is higher priority than a path p_s if $\pi(t) > \pi(s)$. Let $T_i \subseteq T$ be the set of paths in P that do not conflict with any higher priority path.
- Let $P_i = \text{Trim}(P_{i-1}, T_i)$. In $\text{Trim}(P_{i-1}, T_i)$, for each $s \notin T_i$, replace $p_s \in P_{i-1}$ with its maximal prefix that does not conflict with any path $p_t \in P_{i-1}$ where $t \in T_i$.

We formally describe our algorithm in Algorithm 1.

Algorithm 1 Dipath Aggregation

Input: directed graph $G = (V, A)$, root r, terminals T, $P = \{p_t^*\}_{t\in T}$
Output: $O(\log(|T|))$-hop solution ▷ Theorem 23
$P_0 = \{p_t^*\}_{t\in T}$
$i \leftarrow 0$
while There are conflicting paths in P_i **do**
  $i \leftarrow i + 1$
  $\pi_i \leftarrow$ uniformly random permutation of T
  $T_i \leftarrow \{t : p_t$ has no conflict with a higher priority path$\}$
  $P_i \leftarrow \text{Trim}(P_{i-1}, T_i)$
return P_i

2.2 Analysis of Dipath Aggregation

In the analysis of our algorithm we will define the *conflict graph* which will represent which paths conflict with each other. We will show that the number of edges in the conflict graph decreases in expectation by a constant fraction each round. It follows that Algorithm 1 terminates in $O(\log |T|)$ rounds. The following lemma tells us that if the algorithm terminates after h rounds then it gives an h-hop solution.

Lemma 24. *If a terminal $t \in T_i$, then in the solution returned by Algorithm 1, t is at most $i-1$ hops from r.*

Proof. We prove the lemma by induction. If $t \in T_1$, then it is zero hops from r. Now suppose the inductive hypothesis holds for all terminals sampled in rounds less than or equal to i. Consider a terminal $t \in T_{i+1}$. Since t does not conflict with any other terminals in T_{i+1}, it takes one hop to get to either the root or a cluster that was claimed by a previous terminal and then, by the inductive hypothesis, at most $i-1$ hops to get to the root. Thus, t is at most i hops from the root as desired. □

It is now only left to show that the algorithm terminates after $200\log(|T|)$ rounds. For each P_i we will define the *conflict graph* X_i. X_i has a vertex for each terminal in T and there is an edge between vertices $s, t \in V(X_i)$ if the corresponding paths $p_s, p_t \in P_i$ conflict.

For terminals $u, s, t \in T$, where p_u conflicts with p_s and p_t in P_i, we will say that s disconnects u from t in round i if, in P_i, every prefix of p_u that conflicts with p_t also conflicts with p_s. Note that if s disconnects u from t in round i and $s \in T_i$, then for all $j > i$, p_u will not conflict with p_t in P_j and so there will be no edge between u and t in X_j. For an edge $e = (s,t) \in E(X_i)$, we will say that e is directed from s to t if s disconnects t from at least a 1/10-fraction of the paths that conflict with it in round i. It is possible for e to be, undirected, bidirected, or directed in either direction.

Lemma 25. *If X_i has m_i edges, then at least $\frac{4m_i}{5}$ edges will be bidirected.*

Proof. For each vertex $v \in X_i$, at least $\frac{9\deg(v)}{10}$ edges are directed towards v. It follows that if we count bidirected edges with multiplicity 2 then the total number of directed edges in X_i is

$$\sum_{v \in V(X_i)} \frac{9\deg(v)}{10} = \frac{18m_i}{10}$$

so at least $\frac{4m_i}{5}$ of the edges must be bidirected. □

Lemma 26. *If X_i has m_i edges, then $\mathbb{E}[m_{i+1}] \leq \frac{24m_i}{25}$.*

Proof. Let $\tilde{X}_i$ be the graph induced by the bidirected edges of X_i. For each edge $e = (s,t) \in E(\tilde{X}_i)$, define an event S_{st} which occurs if $\pi_i(s) \geq \pi_i(r)$ for all $r \in \Gamma_{X_i}(s) \cup \Gamma_{X_i}(t)$ [6]. By definition of $\tilde{X}_i$, at least $\frac{1}{10}\deg_{X_i}(t)$ of the terminals that conflict with t in P_i are disconnected by s. If t_0 is such a terminal and S_{st} occurs, then $f = (t, t_0) \notin X_{i+1}$ and we will say that s blocks f on the side of t. Note that each edge can be blocked at most twice, once from each side. Thus, we can lower bound the number of edges removed in round i by

$$m_i - m_{i+1} \geq \frac{1}{2} \sum_{(s,t) \in E(\tilde{X}_i)} \left(\frac{\deg_{X_i}(t)}{10} S_{st} + \frac{\deg_{X_i}(s)}{10} S_{ts} \right).$$

[6] $\Gamma_{X_i}(v)$ is the set of neighbors of v in X_i.

The probability S_{st} occurs is at least $\frac{1}{\deg_{X_i}(s)+\deg_{X_i}(t)}$. The lemma follows from taking the expectation of both sides of the previous equation.

$$\begin{aligned}
\mathbb{E}[m_i - m_{i+1}] &\geq \frac{1}{2} \sum_{(s,t)\in E(\tilde{X}_i)} \left(\frac{\deg_{X_i}(t)}{10}\mathbb{P}(S_{st}) + \frac{\deg_{X_i}(s)}{10}\mathbb{P}(S_{ts}) \right) \\
&\geq \frac{1}{20} \sum_{(s,t)\in E(\tilde{X}_i)} \left(\frac{\deg_{X_i}(t)}{\deg_{X_i}(s)+\deg_{X_i}(t)} + \frac{\deg_{X_i}(s)}{\deg_{X_i}(t)+\deg_{X_i}(s)} \right) \\
&\geq \frac{|E(\tilde{X}_i)|}{20} \\
&\geq \frac{m_i}{25}
\end{aligned}$$

Where the final inequity comes from Lemma 24 □

Combining Lemma 24 and Lemma 26 we get Theorem 23.

Theorem 23. *Every instance of the path aggregation problem has a* $200\log(|T|)$*-hop solution that can be computed in polynomial time with high probability.*

Proof. By Lemma 26 and the fact that $m_0 \leq |T|^2$ we get

$$\mathbb{E}[m_{\log_{25/24}(|T|^{10})}] \leq \frac{m_0}{|T|^{10}} \leq \frac{1}{|T|^8}.$$

Thus, by Markov's inequality, the probability that there is a conflict after $\log_{25/24}(|T|^{10})$ rounds of the while loop is at most $\frac{1}{|T|^8}$. If there are no remaining edges after $\log_{25/24}(|T|^{10})$, then Lemma 24 tells us that we have a solution of at most $\log_{25/24}(|T|^{10}) < 200\log(|T|)$ hops proving the theorem. □

3 Embedding Hierarchical Partitions into Spanning Trees

We now leverage our algorithm for the dipath aggregation problem (Theorem 23) to show our main result: an $O(1)$-distortion embedding of an $\Omega(\log n)$-growing hierarchical partition into a spanning tree, as summarized below.

Theorem 5 (Hierarchical Partitions Embed into Spanning Trees)**.** *Any* γ*-growing hierarchical partition of* n*-node graph* $G = (V, E, w)$ *where* $\gamma \geq 203\log n$ *can be embedded into a spanning tree* $T \subseteq G$ *with distortion at most* 407 *(in poly-time with high probability).*

3.1 Algorithm for Embedding Hierarchical Partition

Our algorithm takes an edge-weighted graph $G = (V, E, w)$ and a γ-growing hierarchical partition $\mathcal{H}$ of G and outputs a spanning tree F of G. It is based on the recursive algorithm of [17] which gives $\mathsf{poly}(\log)$ distortion for strong sparse partitions (see Definition 49). However, instead of relying on sparsity conditions, our algorithm uses dipath aggregation to better aggregate the parts of each cluster.

Our algorithm is recursive and is described in Algorithm 3. A recursive call will take an edge-weighted graph $G = (V, E, w)$, portals $S \subseteq V$, a γ-growing hierarchical partition $\mathcal{H} =$

$\{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k\}$, and will return a spanning forest F of G. Remember that $\mathcal{C}_k = \{\{V(G)\}\}$ so $\mathcal{C}_{k-1}$ is the coarsest non-trivial partition of G. Let $\mathcal{C}_S = \{C \in \mathcal{C}_{k-1} : C \cap S \neq \emptyset\}$ be the set of all parts in $\mathcal{C}_{k-1}$ that contain a portal. We will not add any edges in $\mathcal{C}_S$ until later recursive calls of Algorithm 3. Let G_S be G with all clusters in $\mathcal{C}_S$ contracted into a single vertex r. For each $C \in \mathcal{C}_{k-1} \setminus \mathcal{C}_S$, arbitrarily choose a representative vertex $\mathbf{rep}(C) \in V(C)$ and let T be the set of all such representatives. Then for each $t \in T$ let p_t^* be a shortest path in G_S from t to r. We then run dipath aggregation on G_S with root r, terminals T, and paths $\{p_t^*\}_{t \in T}$ and get a solution $\{p_t\}_{t \in T}$. We call a path p_t a highway.

We then return F where F is the set of all highway edges along with edges from recursive calls to Algorithm 3 for each of $C \in \mathcal{C}_{k-1}$. For each C we will use the hierarchy $\mathcal{H}[C]$ which is $\mathcal{H}$ restricted to C and portal set S_C where

- if $C \in \mathcal{C}_S$, $S_C = S \cap C$ and
- if $C \notin \mathcal{C}_S$, S_C is the set of all vertices in C contained in some highway edge.

Now, given an edge-weighted graph $G = (V, E, w)$ with a γ-growing hierarchical partition $\mathcal{H}$ of G, we can build a spanning tree which will have low distortion using Algorithm 2. Simply choose an arbitrary vertex $s \in V(G)$ and run Algorithm 3 with G, $\mathcal{H}$, and portal set $\{s\}$.

Algorithm 2 Embedding

Input: Edge-weighted graph $G = (V, E, w)$, γ-growing hierarchical partition $\mathcal{H}$ of G
Output: Spanning tree F of G
$s \leftarrow$ arbitrary vertex in $V(G)$
return Embedding-Helper$(G, \{s\}, \mathcal{H})$

Algorithm 3 Embedding-Helper

Input: Edge-weighted graph $G = (V, E, w)$, portals $S \subseteq V$, γ-growing hierarchical partition $\mathcal{H} = \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k\}$ of G
Output: Spanning forest F of G
if $|V| = 1$ **then**
    return G
$\mathcal{C}_S \leftarrow \{C \in \mathcal{C}_{k-1} : C \cap S \neq \emptyset\}$
$G_S \leftarrow G$ with all clusters in $\mathcal{C}_S$ contracted into a single vertex r
$T \leftarrow \{\mathbf{rep}(C) : C \in \mathcal{C}_{k-1} \setminus \mathcal{C}_S\}$
for $t \in T$ **do**
    $p_t^* \leftarrow$ a shortest path from t to r in G_S
$\{p_t\}_{t \in T} \leftarrow$ solution to dipath aggregation on G_S with root r, terminals T, and paths $\{p_t^*\}_{t \in T}$
for $C \in \mathcal{C}_{k-1}$ **do**
    if $C \in \mathcal{C}_S$ **then**
        $S_C \leftarrow S \cap C$
    else
        $S_C \leftarrow \{v \in C : v \in p_t \text{ for some } t \in T\}$
    $F_C \leftarrow$ Embedding-Helper$(G[C], S_C, \mathcal{H}[C])$
return $\cup_{t \in T} p_t \cup_{C \in \mathcal{C}_{k-1}} F_C$

3.2 Analysis of Embedding

Lemma 27. *The output of Algorithm 2 is a spanning tree of G.*

Proof. We prove by induction that each call to Algorithm 3 returns a forest where each vertex is connected to exactly one portal vertex. Since Algorithm 2 initially calls it with r being the lone portal, this proves the lemma. The base case is when $|V| = 1$ where the algorithm returns G which is a spanning tree of itself.

For the inductive step, assume that each recursive call to Embedding-Helper$(G[C], S_C, \mathcal{H}[C])$ returns a forest of C where each vertex is connected to exactly one element of S_C. Thus, $F[\mathcal{C}_S]$ is a forest of $G[\mathcal{C}_S]$ where each vertex is connected to exactly one vertex of S. Now consider some $C \in \mathcal{C}_{k-1} \setminus \mathcal{C}_S$. Every $v \in C$ will be part of some tree T_v in F_C and T_v will have a unique path p_t that intersects it which will intersect it exactly once and will lead to some vertex in $G[\mathcal{C}_S]$. Thus, v is not part of a cycle and is connected to some $s \in S$. Therefore, there are no cycles in F and each vertex is connected to some $s \in S$. □

Lemma 28. *If $\gamma \geq 203 \log n$, then for any $\mathcal{C}_i \in \mathcal{H}$, $C \in \mathcal{C}_i$, and $u \in C$, $d_F(u, S_C) \leq 203\gamma^i$.*

Proof. We prove the statement by induction. When $i = 0$, it holds since a vertex is distance 0 from itself. Now, for some $i > 0$, suppose it holds for all values less than i.

Consider the unique shortest path P in F from u to S_C. P will consist of highway edges added during the call to Algorithm 3 on C and edges added by recursive calls on child clusters. Let $p_{t_1}, p_{t_2}, \ldots p_{t_x}$ be the highways that P intersects. For each $i \in [x]$, let s_i be the first vertex of p_{t_i} in P and f_i be the last. For $i < x$, we want to show that $d_{p^*_{i+1}}(s_{i+1}, S_C) \leq d_{p^*_i}(f_i, S_C) + \gamma^{i-1}$. This follows since f_i and s_{i+1} are in the same child cluster of C and p_{i+1} is a shortest path to S_C. Thus, the total length of highway edges in P is at most $d_{p^*_1}(s_1, S_C) + x \cdot \gamma^{i-1}$. Since p^*_1 is a shortest path in C, $d_{p^*_1}(s_1, S_C) \leq \gamma^i$ and by Theorem 23, $x \leq 200 \log n$ (with high probability). Therefore, the total length of highway edges in P is $\gamma^i + 200 \log(n) \cdot \gamma^{i-1}$.

Next, we consider the recursively added edges in P. These will be the segments of P connecting u to s_1, f_x to S_C, and, for $i \in [x-1]$, f_i to s_{i+1}. There are $x+1$ such segments and by the inductive hypothesis, each one has length at most $203\gamma^{i-1}$. Thus, the total length of this part of P is at most $(200 \log(n) + 1) \cdot (203\gamma^{i-1})$. Therefore,

$$\begin{aligned} d_F(u, S_C) &\leq \gamma^i + 200 \log(n) \cdot \gamma^{i-1} + (200 \log(n) + 1) \cdot (203\gamma^{i-1}) \\ &\leq \gamma^i + \gamma^i + 201 \log(n) \cdot 203\gamma^{i-1} \\ &\leq \gamma^i + \gamma^i + 201\gamma^i \\ &= 203\gamma^i \end{aligned}$$

proving the lemma. □

We are now ready to use Lemma 27 and Lemma 28 to prove Theorem 5.

Theorem 5 (Hierarchical Partitions Embed into Spanning Trees)**.** *Any γ-growing hierarchical partition of n-node graph $G = (V, E, w)$ where $\gamma \geq 203 \log n$ can be embedded into a spanning tree $T \subseteq G$ with distortion at most 407 (in poly-time with high probability).*

Proof. We will show that the output, F, of Algorithm 2 is a spanning tree with distortion at most 407. By Lemma 27, F is a spanning tree. Consider any $\mathcal{C}_i \in \mathcal{H}$, $C \in \mathcal{C}_i$, and $u, v \in C$. If $q, r \in S_C$,

then q, r belong to some shortest path in an ancestor of C so $d_F(q,r) \leq \gamma^i$. By Lemma 28 we also have that $d_F(u, S_C) \leq 203\gamma^i$ and $d_F(v, S_C) \leq 203\gamma^i$. Putting these together we get

$$d_F(u,v) \leq 203\gamma^i + \gamma^i + 203\gamma^i = 407\gamma^i.$$

Since this is true for all u and v and all i such that u and v are in the same part in $\mathcal{C}_i$, it follows that

$$d_F(u,v) \leq 407 d_{\mathcal{H}}(u,v)$$

for all distinct u and v so F is a 407-distortion embedding of $\mathcal{H}$. □

4 Laminar Family Distance

In this section we prove Theorem 7 that laminar families embed into spanning trees.

Theorem 7 (Laminar Families Embed into Spanning Trees)**.** *Any laminar family on n-node graph $G = (V, E, w)$ can be (poly-time with high probability) embedded into a spanning tree $T \subseteq G$ with distortion $O(\log n)$.*

First, we show that a laminar family can be turned into a γ-growing hierarchy by incurring a distortion of γ.

Lemma 29. *Given a graph $G = (V, E, w)$, laminar family $\mathcal{L}$ on G, and $\gamma > 1$, we can construct in polynomial time a γ-growing hierarchical partition $\mathcal{H}$ of G such that for all $u, v \in V(G)$,*

$$D_{\mathcal{L}}(u,v) \leq D_{\mathcal{H}}(u,v) \leq \gamma \cdot D_{\mathcal{L}}(u,v).$$

Proof. Initialize $\mathcal{C}_0, \mathcal{C}_1 \ldots, \mathcal{C}_h$ as empty. Then run the following recursive algorithm which takes a set C in $\mathcal{L}$ and a height h. Call the maximal subsets of C its children. Starting with $C = G$ and height $h = \lceil log_\gamma(\Delta(G)) \rceil$:

- If $h = 0$, add all vertices of C to $\mathcal{C}_0$ as singletons
- Else if $\Delta(G[C]) > \gamma^h$, recurse on each of the children of C in $\mathcal{L}$ with height h.
- Else if $\gamma^h \geq \Delta(G[C]) > \gamma^{h-1}$, add C to $\mathcal{C}_h$ and recurse on each of the children of C in $\mathcal{L}$ with height $h-1$.
- Else if $\gamma^{h-1} > \Delta(G[C])$, add C to $\mathcal{C}_h$ and recurse on C with height $h-1$.

The recursive algorithm returns $\mathcal{H} = \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_h\}$ where $\mathcal{H}$ is a γ-growing hierarchical partition by construction. We have $D_{\mathcal{L}}(u,v) \leq D_{\mathcal{H}}(u,v) \leq \gamma \cdot D_{\mathcal{L}}(u,v)$ since we only add a part C at level h if either $\gamma^h \geq \Delta(G[C]) > \gamma^{h-1}$ or $\gamma^{h-1} > \Delta(G[C])$ where in the latter case C will also be added at level $h-1$. □

Next we show that it is good enough to embed the the new γ-growing hierarchical partition into a spanning tree.

Lemma 30. *If we can embed any γ-growing hierarchical partition of $G = (V, E, w)$ into a spanning tree with $O(1)$ distortion, then we can embed any laminar family on G into a spanning tree with distortion $O(\gamma)$.*

Proof. Let $\mathcal{L}$ be a laminar family on G. By Lemma 29 there is a γ-growing hierarchical partition $\mathcal{H}$ of G such that for all $u, v \in V(G)$, $D_{\mathcal{H}_0}(u, v) \leq \gamma \cdot D_{\mathcal{H}}(u, v)$. By assumption, we can embed $\mathcal{H}$ into a spanning tree $T \subseteq G$ with $O(1)$ distortion. However, then $\mathcal{L}$ embeds into T with $\gamma \cdot O(1) = O(\gamma)$ distortion. □

Combining Theorem 5 with Lemma 30 we get Theorem 7

5 Probabilistic Spanning Tree Embeddings via Hierarchies

As an immediate consequence of our embedding of hierarchical partitions into spanning trees, we get that finding low distortion probabilistic tree embeddings into spanning trees reduces to finding good hierarchical partitions. In particular, we have the following.

Theorem 12. *Suppose graph $G = (V, E, w)$ has a (poly-time sampleable) α-distortion probabilistic γ-growing hierarchical partition embedding for $\gamma \geq 203 \log n$. Then, G has a (poly-time with high probability sampleable) $O(\alpha)$-distortion probabilistic tree embedding into spanning trees.*

Proof. Let $\mathcal{D}$ be the distribution over distance functions that make up the α-distortion probabilistic γ-growing hierarchical partition embedding of G. By Theorem 5, any $\mathcal{H} \in \mathcal{D}$ can be embedded into one or more spanning trees so that for each such spanning tree T and every $u, v \in V(G)$

$$d_T(u, v) \leq 407 \cdot d_{\mathcal{H}}(u, v) \leq 407\alpha \cdot d_G(u, v) = O(\alpha) \cdot d_G(u, v).$$

Thus, G has an $O(\alpha)$-distortion probabilistic tree embedding into spanning trees $\hat{\mathcal{D}}$. Additionally, if we can sample from $\mathcal{D}$ in polynomial time, then we can run Algorithm 2 on the resulting hierarchical partition to sample from $\hat{\mathcal{D}}$ with high probability. □

6 Low Distortion Probabilistic Laminar Families

In this section we adapt the algorithm from [8] for probabilistically embedding metrics into ultrametrics so that it gives probabilistic embeddings of graphs into laminar families.

Lemma 31 (Theorem 2 of [8])**.** *Any graph $G = (V, E, w)$ on n points probabilistically embeds into laminar families with $O(\log n)$-distortion. Furthermore, we can sample such a laminar family in polynomial time.*

6.1 Algorithmic Low Distortion Probabilistic Laminar Families

The algorithm works by growing a ball S in G, removing S from G, and then recursing on S and the connected components of $G \setminus S$. To grow S we start by strategically selecting a vertex v and then choosing a radius r from a probability distribution $p(r)$ and letting $S = B_G(v, r)$. The distribution $p(r)$ will depend on a parameter Δ related to the diameter of the graph. In the metric case, any subset of points will have diameter at most the original set of points. However, since we are interested in strong diameter which can increase, we will use for a subgraph H a parameter Δ set to the minimum of the strong diameter of H, $\Delta(H)$, and the parameter from H's parent in the recursion. This is the main difference between our algorithm and that of [8].

More concretely, given a connected subgraph $H \subseteq G$ and parameter Δ, let $C = 128$, $R = \Delta/C$, and, for all $x \in V(H)$, let $\lambda(x) = \lambda_H^R(x) = |B_H(x, 16R)|/|B_H(x, R)|$. Let $v = \arg\min_{x \in V(H)} \lambda_H^R(x)$, sample $r \in [2R, 4R]$ from the distribution

$$p(r) = p_H^R(r) = \begin{cases} \left(\frac{\lambda_H^R(v)^2}{1-\lambda_H^R(v)^{-2}}\right) \cdot \frac{\ln\left(\lambda_H^R(v)\right)}{R} \cdot \lambda_H^R(v)^{-r/R} & \text{if } r \in [2R, 4R] \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

(which is a probability distribution by Lemma 32), and set $S = B_H(v, r)$. In the case when $\lambda(x) = 1$, $p(r)$ isn't defined so instead set $r = 2R$. The algorithm for growing S is given by Algorithm 4. Let $H_1, \dots H_k$ be the connected components of $G \setminus S$. For each $H \in \{S, H_1, \dots H_k\}$, set Δ_H to be the minimum of $\Delta(H)$ and the parameter Δ from its parent. Then recurse on H with parameter Δ_H. The recursion ends when H is a singleton vertex. We then take $\mathcal{L}$ to be the set of all subgraphs H we consider thoughout the algorithm along with G. The recursive algorithm is given in Algorithm 5. We initiate Algorithm 5 with G and parameter $\Delta(G)$.

Algorithm 4 Bartal Cut

Input: Edge-weighted graph $G = (V, E, w)$, parameter R
Output: Ball $B_G(v, r)$ for some $v \in V(G)$ and $r \in [2R, 4R]$
$v \leftarrow \arg\min_{x \in V(H)} \lambda_H^R(x)$
if $\lambda_H^R(v) > 1$ **then**
  $r \leftarrow$ sample from $p_H^R(r)$ ▷ $p_H^R(r)$ is defined in Equation (1)
else
  $r \leftarrow 2R$
return $B_G(v, r)$

Algorithm 5 Probabilistic LF

Input: Edge-weighted graph $G = (V, E, w)$, parameter Δ
Output: Hierarchical partition $\mathcal{H}$ of G
$R \leftarrow \Delta/C$
$S \leftarrow$ Bartal-Cut(G, R)
$H_1, \dots, H_k \leftarrow$ connected components of $G \setminus S$
for $S \in \{S, H_1, \dots, H_k\}$ **do**
  if $|H| = 1$ **then**
    $\mathcal{H}_H = H$
  else
    $\Delta_H \leftarrow \min\{\Delta(H), \Delta\}$
    $\mathcal{H}_H =$ Probabilistic-LF$(H, \Delta_H/C)$
return $\{G\} \cup_{H \in \{S, H_1, \dots, H_k\}} \mathcal{H}_H$

6.2 Analysis of Low Distortion Probabilistic Laminar Families

In this section we prove Lemma 31. We start by proving that $p(r)$ is in fact a probability distribution.

Lemma 32. *The distribution $p(r)$ from Equation (1) is a probability distribution.*

Proof. We have $p(r) \geq 0$ for all r so it is only left to show that it integrates to 1.

$$\begin{aligned}\int_{2R}^{4R} p(r)dr &= \left(\frac{\lambda(v)^2}{1-\lambda(v)^{-2}}\right)\cdot\frac{\ln(\lambda(v))}{R}\cdot\int_{2R}^{4R}\lambda(v)^{-r/R}dr\\ &= \left(\frac{\lambda(v)^2}{1-\lambda(v)^{-2}}\right)\cdot\left(-\lambda(v)^{-r/R}\Big|_{2R}^{4R}\right)\\ &= \left(\frac{\lambda(v)^2}{1-\lambda(v)^{-2}}\right)\cdot\left(\lambda(v)^{-2}-\lambda(v)^{-4}\right)\\ &= \left(\frac{\lambda(v)^2}{1-\lambda(v)^{-2}}\right)\cdot\left(\frac{\lambda(v)^2-1}{\lambda(v)^4}\right)=1\end{aligned}$$

□

Let P be a shortest path between x and y in the original graph G. Since it is possible for the distance between x and y to increase when we take subgraphs, in our analysis we will pay Δ for x and y being separated the first time P is cut. Before P is cut, we know that the distance between x and y will remain unchanged. This is an upper bound on the actual distance between x and y in our final laminar family since it is possible for P to be cut while x and y remain in the same subgraph. Let A be the event $P\cap\Gamma(S)\neq\emptyset$ that P is cut by $\Gamma(S)$. Let B be the event $P\cap S\neq\emptyset$ that some vertex of P is in S. We want to be able to bound the probability that P is cut and to do so we will find the following lemma useful.

Lemma 33. *For $z\geq 1$, $\ln z\leq z-1/z$.*

Proof. It is enough to show that $f(z)=z^2-1-z\cdot\ln(z)\geq 0$ for $z\geq 1$. We have $f(1)=0$ and we will show that f is increasing on $[1,\infty]$. Taking the derivative we get $f'(z)=2z-\ln(z)-1$. Now, $f'(1)=1$ and $f''(z)=2-1/z$ so $f'(z)$ is positive on our interval proving the lemma. □

The next lemma allows us to bound the probability that P is cut by a single call to Algorithm 4.

Lemma 34. *For any $H\subseteq G$ with $P\subseteq H$ and parameter $R\leq\Delta(H)/C$, Algorithm 4 gives*

$$\mathbb{P}(A)\leq[\mathbb{P}(B)\cdot\ln\lambda(v)+\lambda(v)^{-1}]\cdot\frac{w(P)}{R}.$$

Proof. If $\lambda(v)=1$ then the lemma follows trivially since either $P(A)=0$ or $P(A)=1$ and $w(P)\geq 15R$ so $[\mathbb{P}(B)\cdot\ln\lambda(v)+\lambda(v)^{-1}]\cdot\frac{w(P)}{R}\geq 1$.

Thus, we assume $\lambda(v)>1$. Let $v_0\in V(P)$ be a vertex in P with minimal distance to v, $v_1\in V(P)$ be a vertex in P with maximal distance to v, and $\delta=d_H(v,v_0)$. Then

$$\begin{aligned}\mathbb{P}(A) &= \int_{d_H(v,v_0)}^{d_H(v,v_1)} p(r)dr\\ &\leq \left(\frac{\lambda(v)^2}{1-\lambda(v)^{-2}}\right)\cdot\frac{\ln(\lambda(v))}{R}\cdot\int_{\delta}^{\delta+w(P)}\lambda(v)^{-r/R}dr\\ &= \left(\frac{\lambda(v)^2}{1-\lambda(v)^{-2}}\right)\cdot\left(\lambda(v)^{-\delta/R}-\lambda(v)^{-(\delta+w(P)/R}\right)\\ &= \left(\frac{\lambda(v)^2}{1-\lambda(v)^{-2}}\right)\cdot\lambda(v)^{-\delta/R}\left(1-\lambda(v)^{-w(P)/R}\right)\\ &\leq \left(\frac{\lambda(v)^2}{1-\lambda(v)^{-2}}\right)\cdot\lambda(v)^{-\delta/R}\cdot\frac{w(P)}{R}\cdot\ln\lambda(v) \qquad (2)\end{aligned}$$

where the final inequality holds since

$$1 - \lambda(v)^{-w(P)/R} = 1 - e^{-\frac{w(P)}{R} \cdot \ln \lambda(v)} \leq \frac{w(P)}{R} \cdot \ln \lambda(v).$$

We can assume that $\delta \leq 4R$ since otherwise $\mathbb{P}(A) = 0$ and the lemma is true. Similarly, we can assume $\delta \geq 2R$; however, we will return to this case later. Thus, assuming $2R \leq \delta \leq 4R$, we get

$$\begin{aligned}
\mathbb{P}(B) &= \int_{\delta}^{4R} p(r)dr \\
&= \left(\frac{\lambda(v)^2}{1 - \lambda(v)^{-2}}\right) \cdot \frac{\ln(\lambda(v))}{R} \cdot \int_{\delta}^{4R} \lambda(v)^{-r/R} dr \\
&= \left(\frac{\lambda(v)^2}{1 - \lambda(v)^{-2}}\right) \cdot \left(\lambda(v)^{-\delta/R} - \lambda(v)^{-4}\right) \qquad (3)
\end{aligned}$$

Assuming $2R \leq \delta \leq 4R$ we can use Equation (2) and Equation (3) to get

$$\begin{aligned}
\mathbb{P}(A) - \mathbb{P}(B) \cdot \frac{w(P)}{R} \cdot \ln \lambda(v) &= \left(\lambda(v)^{-\delta/R} - \left(\lambda(v)^{-\delta/R} - \lambda(v)^{-4}\right)\right) \cdot \left(\frac{\lambda(v)^2}{1 - \lambda(v)^{-2}}\right) \cdot \frac{w(P)}{R} \cdot \ln \lambda(v) \\
&= \lambda(v)^{-4} \cdot \left(\frac{\lambda(v)^2}{1 - \lambda(v)^{-2}}\right) \cdot \frac{w(P)}{R} \cdot \ln \lambda(v) \\
&= \lambda(v)^{-1} \cdot \left(\frac{\ln \lambda(v)}{\lambda - \lambda(v)^{-1}}\right) \cdot \frac{w(P)}{R} \\
&\leq \lambda^{-1} \cdot \frac{w(P)}{R}
\end{aligned}$$

where the final inequality comes from Lemma 33. Rearranging then gives the lemma. It is only left to consider the case when $\delta < 2R$. In this case, $\mathbb{P}(B) = 1$ so we want to show that

$$\mathbb{P}(A) \leq [\ln \lambda(v) + \lambda(v)^{-1}] \cdot \frac{w(P)}{R}.$$

Using the same calculation as for Equation (2) we get

$$\begin{aligned}
\mathbb{P}(A) &= \mathbb{P}(r \in [2R, d_H(v, v_1)]) \\
&\leq \left(\frac{\lambda(v)^2}{1 - \lambda(v)^{-2}}\right) \cdot \lambda(v)^{-2R/R} \cdot \frac{d_H(v, v_1) - 2R}{R} \cdot \ln \lambda(v) \\
&\leq \left(\frac{\ln \lambda(v)}{1 - \lambda(v)^{-2}}\right) \cdot \frac{w(P)}{R} \\
&\leq [\ln \lambda(v) + \lambda(v)^{-1}] \cdot \frac{w(P)}{R}
\end{aligned}$$

as desired where the final inequality follows from Lemma 33. □

We are now ready to prove Lemma 31.

Proof of Lemma 31. We will show that the laminar family, $\mathcal{L}$, returned by Algorithm 5 has $O(\log n)$ distortion. Consider any $x, y \in V(G)$ and let P be some shortest path between them. We will bound the expectation of the parameter Δ when P is cut for the first time. This is an upper bound on $\mathbb{E}[d_{\mathcal{H}}(x,y)]$ since x and y may be in the same subgraph even if P is cut and because the parameter is non-increasing on subgraphs.

Let $H \subseteq G$ be any subgraph such that $P \subseteq H$ and let $\Delta_H \leq \Delta(H)$ be the parameter passed to Algorithm 5 with H. We will show by induction that

$$\frac{\mathbb{E}[d_{\mathcal{H}}(x,y)]}{C \cdot w(P)} \leq \ln |B_H(x, 16R)| + \ln |B_H(x, 8R)| \tag{4}$$

where $R = \Delta_H / C$. This proves the lemma by taking $H = G$ and $\Delta_G = \Delta(G)$. The base case, when $H = P$, is immediate.

Moving on to the inductive step, for the sake of analysis whenever we grow a ball and break up a graph into its connected components, we will treat this as a series of cuts where each cut removes one connected component. Since each of these imaginary cuts does not cut any edges, they satisfy Lemma 34. If S is the cut of H in our algorithm and $\bar{S} = H \setminus S$ is its compliment then

$$\begin{aligned}\mathbb{E}[d_{\mathcal{H}}(x,y)] \leq\ & \mathbb{P}(P \cap \Gamma(S) \neq \emptyset) \cdot \Delta_H \\ & + \mathbb{P}(P \subseteq S) \cdot \mathbb{E}[d_{\mathcal{H}}(x,y) | P \subseteq S] + \mathbb{P}(P \subseteq \bar{S}) \cdot \mathbb{E}[d_{\mathcal{H}}(x,y) | P \subseteq \bar{S}].\end{aligned} \tag{5}$$

We will make the following assumptions:

- $\mathbb{P}(P \cap \Gamma(S) \neq \emptyset) \neq 0$
- $w(P) \leq R/2$

If the first isn't true then we are done by the inductive hypothesis. If the second isn't true we are done since Equation (4) is satisfied. Together, they guarantee that $d_H(v,x) < 5R$ so $B_H(v,R) \subsetneq B_H(x,8R)$. Note that the inclusion being proper follows from the assumptions. By the inductive hypothesis we have

$$\begin{aligned}\frac{\mathbb{E}[d_{\mathcal{H}}(x,y) | P \subseteq \bar{S}]}{C \cdot w(P)} &\leq \ln |B_H(x,16R) - S| + \ln |B_H(x,8R) - S| \\ &\leq \ln |B_H(x,16R)| + \ln(|B_H(x,8R)| - |B_H(v,R)|)\end{aligned} \tag{6}$$

Since $\Delta(S) \leq 8R$ we also have

$$\frac{\mathbb{E}[d_{\mathcal{H}}(x,y) | P \subseteq S]}{C \cdot w(P)} \leq \ln |B_H(x,R)| + \ln |B_H(x,R/2)|.$$

By our first assumption, $B_H(v,2R)$ does not include both x and y so by our second assumption $d_H(v,x) \geq 2R - d_H(x,y) \geq 3/2 \cdot R$. Thus, $B_H(x,R/2) \subseteq B_H(x,8R) - B(v,R)$ so

$$\frac{\mathbb{E}[d_{\mathcal{H}}(x,y) | P \subseteq S]}{C \cdot w(P)} \leq \ln |B_H(x,R)| + \ln(|B_H(x,8R)| - |B_H(v,R)|). \tag{7}$$

We will now use Lemma 34 to bound the probability that P is cut by S. Since $d_H(v,x) < 5R$ we have that $B_H(x,8R) \subseteq B_H(v,16R)$ so $\lambda(v)^{-1} = \frac{|B_H(v,R)|}{|B_H(v,16R)|} \leq \frac{|B_H(v,R)|}{|B_H(x,8R)|}$. Also, by choice of v, $\lambda(v) \leq \lambda(x) = \frac{|B_H(x,16R)|}{|B_H(x,R)|}$. Thus,

$$\mathbb{P}(P \cap \Gamma(S) \neq \emptyset) \leq \left[\mathbb{P}(B) \cdot \ln \frac{|B_H(x,16R)|}{|B_H(x,R)|} + \frac{|B_H(v,R)|}{|B_H(x,8R)|}\right] \cdot \frac{w(P)}{R} \tag{8}$$

Dividing Equation (5) by $C \cdot w(P)$ and plugging in Equation (6), Equation (7), and Equation (8) we get

$$\begin{aligned}
\frac{\mathbb{E}[d_{\mathcal{H}}(x,y)]}{C \cdot w(P)} \leq & \mathbb{P}(P \cap \Gamma(S) \neq \emptyset) \cdot \frac{R}{w(P)} \\
& + \mathbb{P}(P \subseteq S) \cdot \frac{\mathbb{E}[d_{\mathcal{H}}(x,y) | P \subseteq S]}{C \cdot w(P)} \\
& + \mathbb{P}(P \subseteq \bar{S}) \cdot \frac{\mathbb{E}[d_{\mathcal{H}}(x,y) | P \subseteq \bar{S}]}{C \cdot w(P)} \\
\leq & \left[\mathbb{P}(B) \cdot \ln \frac{|B_H(x,16R)|}{|B_H(x,R)|} + \frac{|B_H(v,R)|}{|B_H(x,8R)|}\right] \\
& + \mathbb{P}(P \subseteq S) \cdot (\ln |B_H(x,R)| + \ln(|B_H(x,8R)| - |B_H(v,R)|)) \\
& + \mathbb{P}(\bar{B}) \cdot (\ln |B_H(x,16R)| + \ln(|B_H(x,8R)| - |B_H(v,R)|)) \\
= & (\mathbb{P}(B) + \mathbb{P}(\bar{B})) \cdot |B_H(x,16R)| \\
& + (\mathbb{P}(P \subseteq S) - \mathbb{P}(B)) \cdot \ln |B_H(x,R)| \\
& + (\mathbb{P}(P \subseteq S) + \mathbb{P}(\bar{B})) \cdot \ln(|B_H(x,8R)| - |B_H(v,R)|) + \frac{|B_H(v,R)|}{|B_H(x,8R)|} \\
\leq & \ln |B_H(x,16R)| + \ln(|B_H(x,8R)| - |B_H(v,R)|) + \frac{|B_H(v,R)|}{|B_H(x,8R)|} \\
= & \ln |B_H(x,16R)| + \ln |B_H(x,8R)| + \ln\left(1 - \frac{|B_H(v,R)|}{|B_H(x,8R)|}\right) + \frac{|B_H(v,R)|}{|B_H(x,8R)|} \\
\leq & \ln |B_H(x,16R)| + \ln |B_H(x,8R)|
\end{aligned}$$

where the second inequality comes from $\mathbb{P}(P \subseteq \bar{S}) = \mathbb{P}(\bar{B})$, the third inequality comes from $\mathbb{P}(P \subseteq S) \leq \mathbb{P}(B)$ and $\mathbb{P}(P \subseteq S) + \mathbb{P}(\bar{B}) \leq 1$, and the fourth inequality comes from $\ln(1-z) + z \leq 0$ for $z \in (0,1)$ and $|B_H(v,R)| < |B_H(x,8R)|$. This gives Equation (4) proving the lemma. □

7 Constructing Ramsey Spanning Trees

As another consequence of our embedding of hierarchical partitions into spanning trees, we get that finding low distortion Ramsey spanning trees reduces to finding good hierarchical partitions.

Theorem 17. *Suppose graph $G = (V, E, w)$ has a (M, k)-fully padded γ-growing hierarchical partition for $\gamma \geq 203 \log n$ (that we can compute in poly-time). Then, G has an $O(k\gamma)$-distortion M-Ramsey spanning tree that we can compute in poly-time with high probability.*

Proof. Let $\mathcal{H}$ be an (M, k)-fully padded γ-growing hierarchical partition of G. Let $u \in \tilde{M}$ and $v \in V$. For all i, if $d_G(u,v) \geq \gamma^i/8k$, then by definition of $\mathcal{H}$, $u, v \in C_i$ for some C_i in $\mathcal{C}_i$. Thus, if $8k \cdot d_G(u,v) \geq \gamma^i$ then $d_{\mathcal{H}}(u,v) \leq \gamma^i$ so

$$d_{\mathcal{H}}(u,v) < 8k \cdot \gamma \cdot d_G(u,v).$$

If we run Algorithm 2 on $\mathcal{H}$, then by Theorem 5 we get a spanning tree T such that for all $u \in \tilde{M}$ and $v \in V$ we have

$$d_T(u,v) \leq 407 \cdot d_{\mathcal{H}}(u,v) < 407 \cdot 8k \cdot \gamma \cdot d_G(u,v) = O(k\gamma) \cdot d_G(u,v)$$

so T is an $O(k\gamma)$-distortion M-Ramsey spanning tree of T. □

8 Hop-Constrained Ramsey Subtrees

Approximating hop-constrained distances with partial tree metrics is introduced in [38]. We are interested in doing the same, except by using subrtrees of low hop diameter instead of partial tree metrics.

Definition 20 (Hop-Constrained Ramsey Subtrees)**.** *Given edge-weighted graph $G = (V, E, w)$, an h-hop-constrained Ramsey subtree distribution of G with exclusion probability ϵ, distortion α, and hop-stretch β is a distribution $\mathcal{D}$ over pairs (T, M) where T is a spanning subtree of G and $M \subseteq V(G)$, such that*

- ***Bounded Exclusion Probability:*** *For every $v \in V(G)$, $\mathbb{P}_{(T,M)\sim\mathcal{D}}(v \notin M) \leq \epsilon$ and;*
- ***Low Distortion:*** *For every $(T, M) \in \mathcal{D}$, $u \in M$, and $v \in V(G)$, $d_T(u,v) \leq \alpha \cdot d^{(h)}(u,v)$.*
- ***Low Hop-Diameter:*** *For every $(T, M) \in \mathcal{D}$, $\mathsf{hd}(T) \leq \beta \cdot \mathsf{hd}(G)$.*

Our main theorem for this section is that such trees with low distortion and hop-diameter exist.

Theorem 40. *For any edge-weighted graph $G = (V, E, w)$, $\epsilon \in (0, 1)$, and $h \leq O\left(\frac{\epsilon \cdot \mathsf{hd}(G)}{\log^{1.5}(n)}\right)$, let $\alpha = \tilde{O}\left(\frac{2^{\sqrt{\log n}}}{\epsilon}\right)$ and $\beta = \log^{O(\sqrt{\log n})}(n)$. Then there is a (poly-time sampleable) h-hop-constrained Ramsey subtree distribution of G with exclusion probability ϵ, distortion α, and hop-stretch β.*

To prove Theorem 40, we will define hop-constrained Ramsey hierarchical partition distributions and then embed such hierarchical partitions into trees. For a subgraph $H \subseteq G$, define the hop constrained diameter $\Delta^{(h)}(H)$ to be the maximum of $d_H^{(h)}(u,v)$ over all pairs $u, v \in H$ where distance and hops are computed using only edges in $G[H]$. We introduce the following definition.

Definition 35 (Hop-Constrained Ramsey Hierarchical Partition Distribution)**.** *Given edge-weighted graph $G = (V, E, w)$, an h-hop-constrained Ramsey hierarchical partition distribution of G with exclusion probability ϵ, growth rate γ, distortion α, and hop-stretch β is a distribution $\mathcal{D}$ over pairs $(\mathcal{H}, M)$ where $\mathcal{H}$ is a γ-growing hierarchical partitions of G and $M \subseteq V(G)$, such that*

- ***Bounded Exclusion Probability:*** *For every $v \in V(G)$, $\mathbb{P}_{(\mathcal{H},M)\sim\mathcal{D}}(v \notin M) \leq \epsilon$ and;*
- ***Low Distortion:*** *For every $(\mathcal{H}, M) \in \mathcal{D}$, $u \in M$, and $v \in V(G)$, $d_{\mathcal{H}}(u,v) \leq \alpha \cdot d^{(h)}(u,v)$.*
- ***Low Hop-Constrained Diameter:*** *For all $\mathcal{C}_i \in \mathcal{H}$ and $C \in \mathcal{C}_i$, $\Delta^{(h\cdot\beta)}(C) \leq \gamma^i$.*

We show that every graph has good hop-constrained Ramsey hierarchical partition distributions. In particular, we can construct such a distribution given $h \geq 1$, $\gamma > 1$, and $\epsilon \in (0,1)$. The only constraint is that $\mathsf{hd}(G) \leq h \cdot \beta$ since otherwise $d^{(h\cdot\beta)}(u,v) = \infty$ for some u and v, violating the low hop-constrained diameter property. By our assumption that all edge weights are at least one and at most some polynomial we then have $\Delta^{h\cdot\beta}(G) = O(\mathsf{poly}(n))$. Define $\kappa = \log_\gamma(\mathsf{poly}(n))$ for an appropriate $\mathsf{poly}(n)$ so that κ is the number of levels in our hierarchy. The following theorem generalizes what is shown in [38]. Note that ρ_0 is a parameter from padded decomposition (which we will define later) which is know to be $O(\log n)$.

Theorem 36 (Theorem 1 of [38]). *Given $h \geq 1$, $\gamma > 1$, and $\epsilon \in (0,1)$, let $\kappa = \log_\gamma(\mathsf{poly}(n))$, $\alpha = \frac{2\cdot\rho_0\cdot\kappa\cdot\gamma}{\epsilon}$, and $\beta = \frac{2\cdot\rho_0\cdot\kappa}{\epsilon}$ where $\rho_0 = O(\log n)$. For any edge-weighted graph $G = (V,E,w)$ with $\mathsf{hd}(G) \leq \beta \cdot h$, there exists an h-hop-constrained Ramsey hierarchical partition distribution of G with exclusion probability ϵ, growth rate γ, distortion α, and hop-stretch β.*

In particular, letting $\gamma = 2$ ($\kappa = O(\log n)$) recovers the result of [38]. However, we will be interested in the case when $\gamma = 2^{\sqrt{\log n}}$ and $\kappa = O(\sqrt{\log n})$.

8.1 Constructing Hop-Constrained Ramsey Subtrees

We start by assuming Theorem 36 to prove Theorem 40. The algorithm for embedding hop-constrained Ramsey hierarchical partitions into hop-constrained Ramsey subtrees is nearly identical to the algorithm for embedding hierarchical partitions into subtrees from Section 3 except we take hop constrained shortest paths. Namely, in Algorithm 3 (the recursive embedding algorithm) we take highways that are shortest paths. However, for hop-constrained Ramsey subtrees, we will instead take shortest hop-constrained paths as given in Algorithm 7. Thus, given a graph with appropriate parameters, we can use Theorem 36 to sample from a hop-constrained Ramsey hierarchical partition distribution and then run Algorithm 7 to build a subtree from it. The low hop-constrained diameter property of the hierarchical partitions guarantees that we can always find an appropriate hop-constrained path. The algorithm is given in Algorithm 6.

Algorithm 6 Hop-Constrained Embedding

Input: $h \geq 1$, $\gamma \geq 203 \cdot \log n$, $\epsilon \in (0,1)$ (let $\kappa = \log_\gamma(\mathsf{poly}(n))$ and $\beta' = \frac{2\cdot\rho_0\cdot\kappa}{\epsilon}$) and $G = (V,E,w)$ with $\mathsf{hd}(G) \leq \beta' \cdot h$
Output: Sample (T,M) from hop-constrained Ramsey subtree distribution
$(\mathcal{H}, M) \leftarrow$ h-hop-constrained Ramsey hierarchical partition $\mathcal{H} = \{\mathcal{C}_0, \ldots, \mathcal{C}_\kappa\}$ of G with exclusion probability ϵ, growth rate γ, distortion α, and hop-stretch β' ▷ Theorem 36
$s \leftarrow$ arbitrary vertex in $V(G)$
$T \leftarrow$ Hop-Constrained-Embedding-Helper$(G, \{s\}, h \cdot \beta')$
return (T,M)

Algorithm 7 Hop-Constrained Embedding-Helper

Input: Edge-weighted graph $G = (V, E, w)$; h-hop-constrained Ramsey hierarchical partition distribution $\mathcal{H} = \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_k\}$ of G with exclusion probability ϵ, growth rate γ, distortion α, and hop-stretch β'; and hop-diameter $h \cdot \beta'$
Output: Spanning forest F of G
if $|V| = 1$ **then**
  return G
$\mathcal{C}_S \leftarrow \{C \in \mathcal{C}_{k-1} : C \cap S \neq \emptyset\}$
$G_S \leftarrow G$ with all clusters in $\mathcal{C}_S$ contracted into a single vertex r
$T \leftarrow \{\mathbf{rep}(C) : C \in \mathcal{C}_{k-1} \setminus \mathcal{C}_S\}$
for $t \in T$ **do**
  $p_t^* \leftarrow$ a shortest path from t to r in G_S of length at most $h \cdot \beta'$
$\{p_t\}_{t \in T} \leftarrow$ solution to dipath aggregation on G_S with root r, terminals T, and paths $\{p_t^*\}_{t \in T}$
for $C \in \mathcal{C}_{k-1}$ **do**
  if $C \in \mathcal{C}_S$ **then**
    $S_C \leftarrow S \cap C$
  else
    $S_C \leftarrow \{v \in C : v \in p_t \text{ for some } t \in T\}$
  $F_C \leftarrow$ Embedding-Helper$(G[C], S_C, \mathcal{H}[C])$
return $\cup_{t \in T} p_t \cup_{C \in \mathcal{C}_{k-1}} F_C$

8.2 Analysis of Hop-Constrained Ramsey Subtrees

Lemma 37. *Algorithm 6 returns a spanning tree T such that $d_T(u, v) \leq O(1) \cdot d_{\mathcal{H}}(u, v)$ for all $u, v \in V(G)$.*

Proof. First, we note that the low hop-constrained diameter property of the hierarchical partitions guarantees that for any $C \in \mathcal{C}_i$, $\Delta^{(h \cdot \beta')}(C) \leq \gamma^i$. Thus, Algorithm 7 is well defined since we can always find an appropriate hop-constrained path. To see that Algorithm 6 returns a spanning tree, note that Lemma 27 does not require the highways to be shortest paths. To see that $d_T(u, v) \leq O(1) \cdot d_{\mathcal{H}}(u, v)$, note that Lemma 28 and Theorem 5 only require that the highways added in C have length at most γ^i. □

Lemma 38. *For every $(T, M) \in \mathcal{D}$, $\mathsf{hd}(T) \leq 3 \cdot (203 \log(n))^{\kappa} \cdot h \cdot \beta'$.*

Proof. For any $u, v \in V(G)$ let $P(u, v)$ be the unique path in T between u and v and let $\delta(u, v)$ be the number of edges in $P(u, v)$. We will show by induction that if $u \in C$ for $C \in \mathcal{C}_i$ then $\delta(u, T_C) \leq (203 \log(n))^i \cdot h \cdot \beta'$ where T_C is the terminal set of C. We have two base cases. The first, when $C \in \mathcal{C}_0$ is trivial since $\mathcal{C}_0$ is the singleton partition. Then second is when $C \in \mathcal{C}_1$. Let $P = P(u, T_C)$. By Theorem 23 we can break P into $200 \log n + 1$ subpaths, each with at most $h \cdot \beta'$ edges. Thus, $|P| \leq (200 \log n + 1) \cdot h \cdot \beta' \leq (203 \log(n))^1 \cdot h \cdot \beta'$ as desired.

We now move on to the induction step. Let $i \geq 2$ and assume $\delta(v, T_{C'}) \leq (203 \log(n))^{i-1} \cdot h \cdot \beta'$ for all $C' \in \mathcal{C}_{i-1}$ and $v \in C'$. Let $C \in \mathcal{C}_i$, $u \in C$, and $P = P(u, T_C)$. We can break P into subsections that were added as highway on C and subsections from recursive calls that connect one highway to the next. By Theorem 23, there are at most $200 \cdot \log n + 1$ of the former and $200 \cdot \log n + 2$ of the latter. The highway subsections will have at most $h \cdot \beta'$ edges each by construction and the recursive subsections will have at most $(203 \log(n))^{i-1} \cdot h \cdot \beta'$ by inductive hypothesis. Thus,

$$\begin{aligned}\delta(u, T_C) &\leq (200 \cdot \log n + 1) \cdot h \cdot \beta' + (200 \cdot \log n + 2) \cdot (203 \log(n))^{i-1} \cdot h \cdot \beta' \\ &\leq 201 \cdot \log n \cdot h \cdot \beta' + 202 \cdot \log n \cdot (203 \log(n))^{i-1} \cdot h \cdot \beta' \\ &\leq 201 \cdot \log n \cdot h \cdot \beta' + (203 \log(n))^{i} \cdot h \cdot \beta' - \log n \cdot (203 \log(n))^{i-1} \cdot h \cdot \beta' \\ &\leq (203 \log(n))^{i} \cdot h \cdot \beta'\end{aligned}$$

where the final line follow since $i \geq 2$. Therefore, for any $u, v \in C$ where $C \in \mathcal{C}_i$,

$$\delta(u, v) \leq (203 \log(n))^{i} \cdot h \cdot \beta' + h \cdot \beta' + (203 \log(n))^{i} \cdot h \cdot \beta' \leq 3 \cdot (203 \log(n))^{i} \cdot h \cdot \beta'.$$

Since there are κ levels in $\mathcal{H}$, we get that $\mathsf{hd}(T) \leq 3 \cdot (203 \log(n))^{\kappa} \cdot h \cdot \beta'$. □

Lemma 39. *Given $h \geq 1$, $\gamma \geq 203 \log n$, and $\epsilon \in (0, 1)$, let $\kappa = \log_\gamma(\mathsf{poly}(n))$, $\alpha = \frac{2 \cdot \rho_0 \cdot \kappa \cdot \gamma}{\epsilon}$, $\beta' \geq \frac{2 \cdot \rho_0 \cdot \kappa}{\epsilon}$, and $\beta = 3 \cdot (203 \log(n))^{\kappa} \cdot \beta' \cdot \frac{h}{\mathsf{hd}(G)}$. For any edge-weighted graph $G = (V, E, w)$ with $\mathsf{hd}(G) \leq h \cdot \beta'$, there exists an h-hop-constrained Ramsey subtree distribution of G with exclusion probability ϵ, distortion α, and hop-stretch β that we can sample from in polynomial time.*

Proof. Run Algorithm 6 on G to get (T, M). The bounded exclusion property, for every $v \in V(G)$, $\mathbb{P}_{(T,M) \sim \mathcal{D}}(v \notin M) \leq \epsilon$, is inherited from hop-constrained Ramsey hierarchical partition distribution. The low distortion property, for every $(T, M) \in \mathcal{D}$, $u \in M$, and $v \in V(G)$, $d_T(u, v) \leq \alpha \cdot d^{(h)}(u, v)$, is similarly inherited from the hierarchical partitions by Lemma 37. Lastly, the low hop-diameter property, for every $(T, M) \in \mathcal{D}$, $\mathsf{hd}(T) \leq \mathsf{hd}(G) \cdot \beta$, follows from Lemma 38. □

Theorem 40. *For any edge-weighted graph $G = (V, E, w)$, $\epsilon \in (0, 1)$, and $h \leq O\left(\frac{\epsilon \cdot \mathsf{hd}(G)}{\log^{1.5}(n)}\right)$, let $\alpha = \tilde{O}\left(\frac{2^{\sqrt{\log n}}}{\epsilon}\right)$ and $\beta = \log^{O(\sqrt{\log n})}(n)$. Then there is a (poly-time sampleable) h-hop-constrained Ramsey subtree distribution of G with exclusion probability ϵ, distortion α, and hop-stretch β.*

Proof of Theorem 40. The proof comes from Lemma 39 by setting $\gamma = 2^{\sqrt{\log n}}$. We then get

- $\kappa = O(\sqrt{\log n})$,
- $\alpha = O\left(\frac{2^{\sqrt{\log n}} \cdot \log^{1.5} n}{\epsilon}\right) = \tilde{O}\left(\frac{2^{\sqrt{\log n}}}{\epsilon}\right)$,
- $\beta' \geq O\left(\frac{\log^{1.5} n}{\epsilon}\right)$, and
- $\beta = 3 \cdot (203 \log(n))^{O(\sqrt{\log n})} \cdot \beta' \cdot \frac{h}{\mathsf{hd}(G)} = \log^{O(\sqrt{\log n})}(n) \cdot \beta' \cdot \frac{h}{\mathsf{hd}(G)}$.

The theorem then follows from setting $\beta' = \frac{\mathsf{hd}(G)}{h}$ which we may do as long as long as $h \leq O\left(\frac{\epsilon \cdot \mathsf{hd}(G)}{\log^{1.5}(n)}\right)$. □

8.3 Algorithmic Hop-Constrained Ramsey Hierarchical Partition Distributions

We now return to proving Theorem 36. Our algorithm follows the same strategy as in [38]. Namely, we define a "mixture metric" which increases the weight of each edge by a set amount so that shortest paths take few edges. We then use a padded decomposition to break G into components and recurse on each component. A vertex will be in the set M if it is not close to the boundary of its component in any level. To make this more formal we start by defining what a padded decomposition is.

Given a graph $G = (V, E, w)$, a set of vertices $S \subseteq V(G)$, and a partition $\mathcal{C}$ of G, we will say that S is cut by $\mathcal{C}$ if there is no part $C \in \mathcal{C}$ such that $S \subseteq C$. We will write $S \not\subseteq \mathcal{C}$ if S is cut by $\mathcal{C}$ and $S \subseteq \mathcal{C}$ otherwise.

Definition 41 (Padded Decomposition)**.** *Given edge-weighted graph $G = (V, E, w)$, a (ρ, Δ)-padded decomposition is a distribution $\mathcal{D}$ over partitions $\mathcal{C}$ of G where*

- ***Bounded Diameter:*** *$d_C(u, v) \leq \Delta$ for every $\mathcal{C} \in \mathcal{D}$, $C \in \mathcal{C}$ and $u, v \in C$ and;*
- ***Paddedness:*** *For every $r > 0$ and $v \in V(G)$, $\mathbb{P}_{\mathcal{C}\sim\mathcal{D}}(B_G(v, r) \not\subseteq \mathcal{C}) \leq \frac{r\cdot\rho}{\Delta}$.*

It is known how to construct padded decompositions with $\rho = O(\log n)$. We will let $\rho_0 = O(\log n)$ be the specific value.

Lemma 42 ([6])**.** *Every edge-weighted graph $G = (V, E, w)$ admits a (ρ_0, Δ)-padded decomposition for $\rho_0 = O(\log n)$ and every $\Delta > 0$ which can be sampled from in polynomial time with high probability.*

We are now ready to define the mixture metric.

Definition 43 (Mixture Metric)**.** *Given a graph $G = (V, E, w)$ and parameters $\Delta > 0, h \geq 1, \epsilon \in (0, 1)$ and $\gamma > 1$, let $\kappa = \log_\gamma(\mathsf{poly}(n))$ and set*

$$w_\Delta(e) = \frac{\epsilon \cdot \Delta}{2 \cdot \rho_0 \cdot \kappa \cdot \gamma \cdot h} + w(e)$$

for each $e \in E$. Let $d_\Delta(u, v)$ be the distance between u in v in $G_\Delta = (G, E, w_\Delta)$.

The recursive algorithm is given in Algorithm 8. Initially it will be called with $\Delta = \gamma^\kappa$. The hierarchical partition $\mathcal{H} = \{\mathcal{C}_0, \ldots, \mathcal{C}_\kappa\}$ that gets returned in the end is the collection of all padded decompositions where the union of all such decomposition in level i make up $\mathcal{C}_i$.

Algorithm 8 Hop-Constrained Hierarchy

Input: Edge-weighted graph $G = (V, E, w)$, parameters $\Delta, h, \gamma, \kappa$
Output: Hierarchical partition $\mathcal{H}$ of G and $M \subseteq V(G)$
$w_\Delta(e) \leftarrow \frac{\epsilon\cdot\Delta}{2\cdot\rho_0\cdot\kappa\cdot\gamma\cdot h} + w(e)$, $G_\Delta \leftarrow (V, E, w_\Delta)$
$\mathcal{C} \leftarrow (\rho_0, \Delta/\gamma)$-padded decomposition of G
if $\Delta > 1$ **then**
  for $C \in \mathcal{C}$ **do**
    $(\mathcal{H}_C, M_C) \leftarrow$ Hop-Constrained Hierarchy$(G[C], \frac{\Delta}{\gamma}, h, \gamma, \kappa)$
else
  $(\mathcal{H}_C, M_C) \leftarrow (\{\{V(G)\}\}, \{V(G)\})$ ▷ In this case G is a singleton
$\mathcal{H} \leftarrow \{G\} \cup_{C\in\mathcal{C}} \mathcal{H}_C$
$M \leftarrow \{\cup_{C\in\mathcal{C}} M_C\} \cap \{v \in V(G) \mid B_G\left(v, \frac{\epsilon\cdot\Delta}{\rho_0\cdot\kappa\cdot\gamma}\right) \subseteq \mathcal{C}\}$
return $(\mathcal{H}, M)$

8.4 Analysis of Hop-Constrained Ramsey Hierarchical Partition Distributions

We start with two useful lemmas that allow us to bound the hop distance in G according to the mixture metric. The lower bound is first.

Lemma 44. *If* $d_\Delta(u,v) \geq \frac{\epsilon\cdot\Delta}{\rho_0\cdot\kappa\cdot\gamma}$ *then* $d^{(h)}(u,v) \geq \frac{\epsilon\cdot\Delta}{2\cdot\rho_0\cdot\kappa\cdot\gamma}$.

Proof. The proof is by contrapositive. Suppose $d^{(h)}(u,v) < \frac{\epsilon\cdot\Delta}{2\cdot\rho_0\cdot\kappa\cdot\gamma}$. Then there exists some u,v path P such that $w(P) < \frac{\epsilon\cdot\Delta}{2\cdot\rho_0\cdot\kappa\cdot\gamma}$ and $|P| \leq h$. Thus,

$$d_\Delta(u,v) \leq w_\Delta(P) \leq h\cdot\frac{\epsilon\cdot\Delta}{2\cdot\rho_0\cdot\kappa\cdot\gamma\cdot h} + w(P) < \frac{\epsilon\cdot\Delta}{\rho_0\cdot\kappa\cdot\gamma}.$$

□

Next, we prove the upper bound.

Lemma 45. *If* $d_\Delta(u,v) \leq \Delta/\gamma$ *then* $d^{(2\cdot\rho_0\cdot\kappa\cdot h/\epsilon)}(u,v) \leq \Delta/\gamma$.

Proof. The proof is again by contrapositive. Suppose $d^{(2\cdot\rho_0\cdot\kappa\cdot h/\epsilon)}(u,v) > \Delta/\gamma$. Let P be any u,v path. If $|P| \leq 2\cdot\rho_0\cdot\kappa\cdot h/\epsilon$ then $w_\Delta(P) > w(P) > \Delta/\gamma$. Otherwise, $w_\Delta(P) > (2\cdot\rho_0\cdot\kappa\cdot h/\epsilon)\cdot\frac{\epsilon\cdot\Delta}{2\cdot\rho_0\cdot\kappa\cdot\gamma\cdot h} = \Delta/\gamma$. Thus, $d_\Delta(u,v) > \Delta/\gamma$. □

We now show the bounded exclusion probability property of the hop-constrained Ramsey hierarchical partition distributions.

Lemma 46. *For every* $v \in V(G)$, $\mathbb{P}_{(\mathcal{H},M)\sim\mathcal{D}}(v \notin M) \leq \epsilon$.

Proof. Let $v \in V(G)$ and consider a single iteration of Algorithm 8. If $B_G(v, \frac{\epsilon\cdot\Delta}{\rho_0\cdot\kappa\cdot\gamma}) \subseteq \mathcal{C}$, then v is not removed from M in this iteration. Thus, setting $r = \frac{\epsilon\cdot\Delta}{\rho_0\cdot\kappa\cdot\gamma}$ we get

$$\mathbb{P}(B(v,r) \not\subseteq \mathcal{C}) \leq \frac{r\cdot\rho_0}{\Delta/\gamma} = \frac{\epsilon}{\kappa}$$

from the guarantees of padded decompositions. Taking a union bound over the κ iterations we get that $v \in M$ except with probability at most ϵ. □

Next, we show the low distortion property.

Lemma 47. *For every* $(\mathcal{H},M) \in \mathcal{D}$, $u \in M$, *and* $v \in V(G)$, $d_{\mathcal{H}}(u,v) \leq \alpha\cdot d^{(h)}(u,v)$ *for* $\alpha = \frac{2\cdot\rho_0\cdot\kappa\cdot\gamma}{\epsilon}$.

Proof. Let $u \in M$ and let P be the shortest u,v path in G with $|P| \leq h$. Assume such a path exists since otherwise the lemma follows trivially by $d^{(h)}(u,v) = \infty$. As long as P is not broken by the padded decomposition of Algorithm 8, $d^{(h)}(u,v)$ remains unchanged. Suppose P is broken for the first time with parameter Δ. Then $d_{\mathcal{H}}(u,v) \leq \Delta$. However, because $u \in M$, P can only have been broken if $w_\Delta(P) \geq \frac{\epsilon\cdot\Delta}{\rho_0\cdot\kappa\cdot\gamma}$ so by Lemma 44 we have $d^{(h)}(u,v) \geq \frac{\epsilon\cdot\Delta}{2\cdot\rho_0\cdot\kappa\cdot\gamma}$. Combining these we get $d_{\mathcal{H}}(u,v) \leq \frac{2\cdot\rho_0\cdot\kappa\cdot\gamma}{\epsilon}\cdot d^{(h)}(u,v)$ as desired. □

Lastly, we show the low hop-constrained diameter property.

Lemma 48. *For all* $\mathcal{C}_i \in \mathcal{H}$ *and* $C \in \mathcal{C}_i$, $\Delta^{(h\cdot\beta)}(C) \leq \gamma^i$ *for* $\beta = \frac{2\cdot\rho_0\cdot\kappa}{\epsilon}$.

Proof. By assumption, $\Delta^{(h\cdot\beta)}(G) \leq \gamma^\kappa$ initially. If $C \in \mathcal{C}_i$ then C is a part of a $(\Delta/\gamma,\rho_0)$-padded decomposition where $\Delta = \gamma^{i+1}$. By the guarantees of the padded decomposition, C has strong diameter at most Δ/γ in the mixture metric. Thus, by Lemma 45, $\Delta^{(h\cdot\beta)}(C) \leq \Delta/\gamma = \gamma^i$. □

Combining Lemma 46, Lemma 47, and Lemma 48 proves Theorem 36.

A Summarizing Prior Work on Strong Sparse Partition Hierarchies

In this section we prove the following by using results of Busch et al. [16] and Busch et al. [17].

Lemma 10 ([16, 17]). *If there exists a (poly-time computable) embedding of a γ-growing hierarchical partition for $\gamma = \Theta(\log^2 n)$ into a spanning tree with distortion α, then there exists an $O(\alpha \cdot \log^5 n)$-approximate UST (that is poly-time computable).*

Our result is based on the idea of a strong sparse partition hierarchy, as first introduced by Jia et al. [42]. In order to define a strong sparse partition hierarchy, we must first define a strong sparse partition. Roughly speaking, these are low diameter vertex partitions where each "reasonable" ball intersects a small number of parts of the partition.

Definition 49 (Strong Δ-Diameter (α, τ)-Sparse Partition). *Given an edge-weighted graph $G = (V, E, w)$, a strong Δ-diameter (α, τ)-sparse partition is a partition $\mathcal{C}$ of V such that:*

- ***Low Strong Diameter:*** *$\forall C \in \mathcal{C}$, the induced graph $G[C]$ has diameter at most Δ;*
- ***Ball Preservation:*** *$\forall v \in V$, the ball $B_G(v, \frac{\Delta}{\alpha})$ intersects at most τ clusters from $\mathcal{C}$.*

A strong sparse partition hierarchy is simply a hierarchical partition where each vertex partition is a strong sparse partition.

Definition 50 (γ-Hierarchy of Strong (α, τ)-Sparse Partitions). *Given an edge weighted graph $G = (V, E, w)$, a γ-hierarchy of strong (α, τ)-sparse partitions is a γ-growing hierarchical partition $\mathcal{H} = (\mathcal{C}_0, \mathcal{C}_1, \ldots)$ where $\mathcal{C}_i$ is a γ^i-diameter (α, τ)-sparse partition for every i.*

Busch et al. [16] showed that computing good USTs reduces to computing strong sparse partition hierarchies and then embedding them into a spanning tree.

Lemma 51 (Lemma 5 of Busch et al. [16]). *Given an edge weighted graph $G = (V, E, w)$ with root $r \in V$ and a γ-hierarchy of strong (α, τ)-sparse partitions $\mathcal{H}$ of G, then a tree that is a μ-distortion embedding of $\mathcal{H}$ is an $O(\mu\alpha\tau\gamma \log n)$-approximate Universal Steiner tree of G with root r.*

Furthermore, Busch et al. [17] showed how to compute strong sparse partition hierarchies with good parameters as follows.

Theorem 52 (Theorem 6.1 of Busch et al. [17]). *Every edge-weighted graph $G = (V, E, s)$ has a (poly-time computable) γ-hierarchy of (α, τ)-sparse strong partitions for $\alpha = \tau = O(\log n)$ and $\gamma = O(\log^2 n)$.*

Combining Lemma 51 and Theorem 52 immediately shows Lemma 10.

B Simply Embedding Graph Distances into Hierarchical Partitions

In this section we adapt the classic analysis of Bartal [6] to show that every graph embeds into a γ-growing hierarchical partition for $\gamma = \Omega(\log n)$, as summarized below.

Lemma 53. *Every graph $G = (V, E, w)$ has a $O(\log^2 n)$-distortion probabilistic γ-growing hierarchical partition for $\gamma \geq 203 \log n$ (that is poly-time computable).*

B.1 Algorithm for Embedding Graph Distances into Hierarchical Partitions

We will be making use of a low diameter decomposition (LDD) algorithm such as that of [48]. An LDD algorithm takes a weighted graph and returns a partition of the vertices so that each part of the partition has low strong diameter and each edge gets cut[7] with small probability.

Definition 54 (LDD)**.** *Given an edge weighted graph $G = (V, E, w)$ and some $\Delta > 0$, a low diameter decomposition is a randomized algorithm that returns a partition $\mathcal{C}$ of V such that*

- ***Low Strong Diameter:*** *$\forall C \in \mathcal{C}$, the induced graph $G[C]$ has diameter at most Δ except with probability at most $\frac{1}{n^3}$;*
- ***Few Edge Cuts:*** *$\forall e \in E(G)$, $\mathbb{P}(e \text{ is cut by } \mathcal{C}) \leq O\left(\frac{w(e)\cdot \log n}{\Delta}\right)$.*

We construct a hierarchy of partitions that preserves pairwise distance in expectation. Our construction is top down where in each level we use an LDD algorithm to further partition each part. In particular, we start with our largest partition which is the entire graph. Then, until our partition is the singleton clusters, we run LDD on each part of the clusters in the previous level independently with the diameter decreased by $O(\log n)$. That is, if $\mathcal{C}_i$ is our partition in layer i, then we get $\mathcal{C}_{i-1}$ by taking the union of $\mathsf{LDD}[G[C], (203 \log n)^i]$ over all $C \in \mathcal{C}_i$.

Algorithm 9 Distance Preserving Hierarchy

Input: Edge-weighted graph $G = (V, E, w)$
Output: $\mathcal{H} = \{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_h\}$
$h \leftarrow \lceil \log_{203 \log n} \Delta \rceil$, $i \leftarrow h$
$\mathcal{C}_h \leftarrow \{V(G)\}$
while $i > 0$ **do**
  $i \leftarrow i - 1$
  $\mathcal{C}_i \leftarrow \cup_{C \in \mathcal{C}_{i+1}} \mathsf{LDD}[G[C], (407 \log n)^i]$
return $\{\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_h\}$

B.2 Analysis of Embedding of Graph Distances into Hierarchical Partitions

Lemma 55. $\mathbb{E}[d_{\mathcal{H}}(u, v)] = d_G(u, v) \cdot O\left(\frac{\log^3 n}{\log \log n}\right)$

Proof. Let P be a shortest path in G from u to v. Suppose $P \subseteq C$ for some part $C \in \mathcal{C}_i$. Then by union bounding over the edges of P, the probability that P is cut in $\mathcal{C}_{i-1}$ is $O\left(\frac{d_G(u,v)\cdot \log n}{(203 \log n)^{i-1}}\right)$. Additionally, $d_{\mathcal{H}}(u, v) \leq (203 \log n)^i$ because $u, v \in C$. Summing over all layers of our hierarchy and paying the diameter of the smallest cluster where P is not cut we get

[7] An edge is cut if its endpoints are in different parts of the partition.

$$\begin{aligned}
\mathbb{E}[d_{\mathcal{H}}(u,v)] &\leq \sum_{i=1}^{h} (203\log n)^i \cdot \mathbb{P}(P \text{ cut in } \mathcal{C}_{i-1} \mid P \text{ not cut in } \mathcal{C}_i) \\
&= \sum_{i=1}^{h} (203\log n)^i \cdot O\left(\frac{d_G(u,v)\cdot \log n}{(203\log n)^{i-1}}\right) \\
&= \sum_{i=1}^{h} d_G(u,v)\cdot O(\log^2 n) \\
&= d_G(u,v)\cdot O\left(\frac{\log^3 n}{\log\log n}\right)
\end{aligned}$$

as required. □

We get Lemma 53 from Lemma 55.

Lemma 53. *Every graph $G = (V, E, w)$ has a $O(\log^2 n)$-distortion probabilistic γ-growing hierarchical partition for $\gamma \geq 203\log n$ (that is poly-time computable).*

Proof. We get a random $(203\log n)$-growing hierarchical partition from running Algorithm 9. Lemma 55 tells us that the resulting distribution over these partitions has $O\left(\frac{\log^3 n}{\log\log n}\right)$ distortion. □

References

[1] I. Abraham and O. Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2012.

[2] I. Abraham, S. Chechik, M. Elkin, A. Filtser, and O. Neiman. Ramsey spanning trees and their applications. 2020.

[3] N. Alon, R. M. Karp, D. Peleg, and D. West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, 1995.

[4] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms (TALG)*, 2(4):640–660, 2006.

[5] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Annual ACM Symposium on Theory of Computing (STOC)*, 1997.

[6] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Symposium on Foundations of Computer Science (FOCS)*, 1996.

[7] Y. Bartal. On approximating arbitrary metrices by tree metrics. In *Annual ACM Symposium on Theory of Computing (STOC)*, 1998.

[8] Y. Bartal. Graph decomposition lemmas and their role in metric embedding methods. In *Annual European Symposium on Algorithms (ESA)*, 2004.

[9] Y. Bartal and M. Mendel. Multiembedding of metric spaces. *SIAM Journal on Computing*, 34 (1):248–259, 2004.

[10] Y. Bartal, N. Linial, M. Mendel, and A. Naor. On metric ramsey-type phenomena. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2003.

[11] Y. Bartal, N. Fandina, and O. Neiman. Covering metric spaces by few trees. *arXiv preprint arXiv:1905.07559*, 2019.

[12] A. Bernstein, D. Nanongkai, and C. Wulff-Nilsen. Negative-weight single-source shortest paths in near-linear time. *Communications of the ACM*, 68(2):87–94, 2025.

[13] G. E. Blelloch, A. Gupta, and K. Tangwongsan. Parallel probabilistic tree embeddings, k-median, and buy-at-bulk network design. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2012.

[14] G. E. Blelloch, Y. Gu, and Y. Sun. Efficient construction of probabilistic tree embeddings. *arXiv preprint arXiv:1605.04651*, 2016.

[15] M. Boehm, R. Hoeksma, N. Megow, L. Nooelke, and B. Simon. On hop-constrained steiner trees in tree-like metrics. *SIAM Journal on Discrete Mathematics*, 36(2):1249–1273, 2022.

[16] C. Busch, C. Dutta, J. Radhakrishnan, R. Rajaraman, and S. Srinivasagopalan. Split and join: Strong partitions and universal steiner trees for graphs. In *Symposium on Foundations of Computer Science (FOCS)*.

[17] C. Busch, D. Chen, A. Filtser, D. Hathcock, D. Hershkowitz, and R. Rajaraman. One tree to rule them all: Poly-logarithmic universal steiner tree. In *Symposium on Foundations of Computer Science (FOCS)*, 2023.

[18] H.-C. Chang, J. Conroy, H. Le, L. Milenkovic, S. Solomon, and C. Than. Covering planar metrics (and beyond): O (1) trees suffice. In *Symposium on Foundations of Computer Science (FOCS)*, 2023.

[19] H.-C. Chang, J. Conroy, H. Le, S. Solomon, and C. Than. Light tree covers, routing, and path-reporting oracles via spanning tree covers in doubling graphs. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 2257–2268, 2025.

[20] S. Chechik and T. Zhang. Dynamic low-stretch spanning trees in subpolynomial time. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2020.

[21] C. Chekuri and R. Jain. Approximation algorithms for hop constrained and buy-at-bulk network design via hop constrained oblivious routing. *Annual European Symposium on Algorithms (ESA)*, 2024.

[22] L. Chen, R. Kyng, Y. P. Liu, R. Peng, M. P. Gutenberg, and S. Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *Symposium on Foundations of Computer Science (FOCS)*, 2022.

[23] M. Chimani and J. Spoerhase. Network Design Problems with Bounded Distances via Shallow-Light Steiner Trees. In *International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 238–248, 2015.

[24] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu. Solving sdd linear systems in nearly m log1/2 n time. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2014.

[25] K. Dhamdhere, A. Gupta, and H. Räcke. Improved embeddings of graph metrics into random trees. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[26] M. Elkin, Y. Emek, D. A. Spielman, and S.-H. Teng. Lower-stretch spanning trees. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2005.

[27] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2003.

[28] A. Filtser. Hop-constrained metric embeddings and their applications. In *Symposium on Foundations of Computer Science (FOCS)*, 2022.

[29] A. Filtser and H. Le. Clan embeddings into trees, and low treewidth graphs. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2021.

[30] J. T. Fineman. Single-source shortest paths with negative real weights in o(mn8/9) time. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 3–14, 2024.

[31] S. Forster, G. Goranci, and M. Henzinger. Dynamic maintenance of low-stretch probabilistic tree embeddings with applications. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2021.

[32] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000.

[33] M. Ghaffari, B. Haeupler, and G. Zuzic. Hop-constrained oblivious routing. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 1208–1220, 2021.

[34] A. Gupta, M. T. Hajiaghayi, and H. Räcke. Oblivious network design. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[35] A. Gupta, R. Krishnaswamy, and R. Ravi. Online and stochastic survivable network design. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2009.

[36] V. Gupta, R. Krishnaswamy, and S. Sandeep. Permutation strikes back: The power of recourse in online metric matching. *arXiv preprint arXiv:1911.12778*, 2019.

[37] B. Haeupler, D. E. Hershkowitz, and G. Zuzic. Deterministic tree embeddings with copies for algorithms against adaptive adversaries. *arXiv preprint arXiv:2102.05168*, 2021.

[38] B. Haeupler, D. E. Hershkowitz, and G. Zuzic. Tree embeddings for hop-constrained network design. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2021.

[39] B. Haeupler, D. Wajc, and G. Zuzic. Universally-optimal distributed algorithms for known topologies. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 1166–1179, 2021.

[40] M. T. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour. Approximating buy-at-bulk and shallow-light k-steiner trees. *Algorithmica*, 53:89–103, 2009.

[41] L. Jia, G. Lin, G. Noubir, R. Rajaraman, and R. Sundaram. Universal approximations for tsp, steiner tree, and set cover. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 386–395, 2005.

[42] L. Jia, G. Lin, G. Noubir, R. Rajaraman, and R. Sundaram. Universal approximations for tsp, steiner tree, and set cover. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2005.

[43] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2013.

[44] M. R. Khani and M. R. Salavatipour. Improved approximations for buy-at-bulk and shallow-light k k-steiner trees and (k, 2)(k, 2)-subgraph. *Journal of Combinatorial Optimization*, 31: 669–685, 2016.

[45] I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving sdd linear systems. *SIAM Journal on Computing*, 43(1):337–354, 2014.

[46] M. Luby. A simple parallel algorithm for the maximal independent set problem. In *Annual ACM Symposium on Theory of Computing (STOC)*, 1985.

[47] M. Mendel and A. Naor. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society*, 9(2):253–275, 2007.

[48] G. L. Miller, R. Peng, and S. C. Xu. Parallel graph decompositions using random shifts. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2013.

[49] J. Naor, D. Panigrahi, and M. Singh. Online node-weighted steiner tree and related problems. In *Symposium on Foundations of Computer Science (FOCS)*, 2011.

[50] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2008.

[51] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Annual ACM Symposium on Theory of Computing (STOC)*, 2004.