Split and Join: Strong Partitions and Universal Steiner Trees for Graphs

Costas Busch * Chinmoy Dutta † Jaikumar Radhakrishnan ‡ Rajmohan Rajaraman †
Srivathsan Srinivasagopalan *

December 8, 2011

Abstract

We study the problem of constructing universal Steiner trees for undirected graphs. Given a graph G and a root node r, we seek a single spanning tree T of minimum stretch, where the stretch of T is defined to be the maximum ratio, over all subsets of terminals X, of the ratio of the cost of the sub-tree T_X that connects r to X to the cost of an optimal Steiner tree connecting X to r. Universal Steiner trees (USTs) are important for data aggregation problems where computing the Steiner tree from scratch for every input instance of terminals is costly, as for example in low energy sensor network applications.

We provide a polynomial time UST construction for general graphs with $2^{O(\sqrt{\log n})}$-stretch. We also give a polynomial time polylogarithmic-stretch construction for minor-free graphs. One basic building block in our algorithm is a hierarchy of graph partitions, each of which guarantees small strong cluster diameter and bounded local neighbourhood intersections. Our partition hierarchy for minor-free graphs is based on the solution to a cluster aggregation problem that may be of independent interest. To our knowledge, this is the first sub-linear UST result for general graphs, and the first polylogarithmic construction for minor-free graphs.

*Department of Computer Science, Louisiana State University, Baton Rouge LA 70803, USA. E-mail: {busch,ssrini1}@csc.lsu.edu

†College of Computer and Information Science, Northeastern University, Boston MA 02115, USA. E-mail: {chinmoy,rraj}@ccs.neu.edu. Chinmoy Dutta is supported in part by NSF grant CCF-0845003 and a Microsoft grant to Ravi Sundaram; Rajmohan Rajaraman is supported in part by NSF grant CNS-0915985.

‡School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai 400005, India. E-mail: jaikumar@tifr.res.in

1 Introduction

In this paper, we study universal approximations for the Steiner Tree problem on undirected graphs. In the universal Steiner Tree (UST) problem for graphs, we are given an undirected graph G and a designated root vertex r in G, and the task is to find a *single spanning tree* T of G such that for any set X of terminal vertices, the minimal subtree T_X of T that connects X to r is a good approximation to the optimal Steiner tree connecting X to r in G. The quality of the solution T is measured by its *stretch* which is the maximum ratio of the cost of T_X to the cost of the optimal Steiner tree connecting X to r in G over all terminal sets X.

The universal Steiner tree problem has been studied extensively for the case of metrics where one is allowed to output an "overlay tree", whose edges correspond to paths in the given graph [21, 19, 9, 30]. Equivalently, the case of metrics can be viewed as a complete graph in which all edge weights satisfy the triangle inequality. In fact, for the case of metrics, there have been several important results on extensions of the UST problem and variants seeking sparse network structures that simultaneously approximates the optimal solution for a range of input instances [15, 17, 16, 19].

The focus of this paper is on the UST problem on arbitrary graphs where we require that the solution being sought is a spanning tree, i.e., a subgraph of the given graph. The Minimum Steiner tree problem on a graph can be well-approximated by solving the same problem on the metric induced by the graph and then computing the minimum subtree connecting the terminals. Such an approach, however, does not apply to the UST problem owing to the requirement that the tree *simultaneously* approximate the optimal Steiner tree for *all* terminal sets. Note that this is a much stronger requirement than asking for a probability distribution over spanning trees where for every terminal set, the expected stretch is small. In the latter case, there might not be any single tree in the distribution that is good for all terminal sets, i.e., for every tree there is a set of terminals for which the induced tree has a cost much larger than the optimal steiner tree.

Motivation. Our problem formulation is primarily motivated by information aggregation and data dissemination in sensor and ad-hoc wireless networks [25, 26, 24]. In a sensor network, data is often collected by a central processing agent that periodically queries a subset of sensors for their sensed information. In many applications, the queries seek aggregate information which can be transmitted using a low cost tree that aggregates data at intermediate nodes. This reduces the number of transmissions which is crucial as sensors have limited battery life and wireless transmissions are power intensive. It is not realistic, however, to expect the sensors to compute and store a low cost tree for each potential subset of sensors being aggregated as the sensors have limited memory and computational power. In this setting, a universal tree provides a practical solution where the nodes just need to realize a single tree which approximates optimal aggregation trees for all subsets of sensors. One approach for the above aggregation problem is to employ a universal *overlay* tree. There are several disadvantages of this approach, however. First, aggregation over the overlay tree requires a physical routing infrastructure that supports point-to-point communication among distant nodes in the network. Second, even if such an infrastructure exists, it may not route packets along minimum-cost paths as required by the overlay tree. Furthermore, aggregation over the overlay tree requires synchronization among distant nodes in the network and incurs overhead in terms of delays and storage. Thus, in some resource-constrained applications, we would ideally want to construct a universal spanning tree as opposed to an overlay tree.

Another motivation to study universal approximation algorithms comes from their relation with differential privacy which was recently established by Bhalgat, Chakrabarty and Khanna [9]. They showed that universal solutions such as USTs are differentially private, and argued that a kind of "strong" lower bounds for universal algorithms implies lower bounds for differentially private ones as well.

From a theoretical standpoint, our motivation is to find out whether the results known for UST and related problems in the metric case can, in fact, be achieved using spanning trees of the underlying graph. The analogous question for the problem of approximating metrics by tree metrics has been answered affir-

matively by [13, 1] who showed that nearly logarithmic-stretch spanning trees exist for all graphs, almost matching the best bound achievable by tree metrics [14]. No comparable results are known for the UST problem.

1.1 Our results and techniques

We present UST algorithms for general graphs and for the special class of minor-free graphs. Our main results are the following.

- **UST for general graphs:** We present a polynomial-time algorithm for computing an $2^{O(\sqrt{\log n})}$-stretch spanning tree for any undirected graph.
- **UST for minor-free graphs:** We present a polynomial-time algorithm for computing a polylogarithmic-stretch spanning tree for any graph that is H-minor free for any finite graph H.

While the specific techniques used in the two algorithms are substantially different, both are grounded in a general framework that draws close connections between USTs and certain graph partitions based on strong diameter. We define an (α, β, γ)-partition of a graph G as a partition of the vertices of G into clusters such that each cluster has strong diameter at most $\alpha\gamma$ and for every vertex the ball of radius γ in G intersects at most β clusters. An (α, β, γ)-partition hierarchy is a sequence of partitions starting from the trivial partition in which each vertex forms its own cluster, and the ith partition is an $(\alpha, \beta, \gamma^i)$-partition and coarsens the $(i-1)$th partition. (See Section 2 for formal definitions.) The significance of our framework stems from the following result.

- **From partition hierarchies to USTs:** For any graph G, given an (α, β, γ)-partition hierarchy for G, an $O(\alpha^2\beta^2\gamma \log n)$-stretch UST for G can be constructed in polynomial time. (Section 3.1)

A major consequence of the above result is that a $(\text{polylog}(n), \text{polylog}(n), \text{polylog}(n))$-partition hierarchy implies a $\text{polylog}(n)$-stretch UST. At a high-level, our approach of using partition hierarchies to derive USTs is similar to that of [21]. There is a critical difference, however, since the natural divide-and-conquer approach of constructing the UST by connecting together subtrees recursively computed for lower levels of the hierarchy does not work. In fact, it can be shown that there exist graphs and hierarchies such that any tree that completely obeys the connectivity structure of the hierarchy in the sense that the subgraph of the tree induced by every cluster of the hierarchy is connected will have poor stretch. We show, however, that we can get the desired bound on stretch by guaranteeing that for any cluster in the given hierarchy, even though the tree may be split within the cluster it is joined externally so as to approximately respect the distances within the cluster.

A natural question to ask is whether (α, β, γ)-partitions or the corresponding hierarchies, with low values of α, β, and γ, are *necessary* to achieve low-stretch USTs. We provide a partial affirmative answer to this question with the following result.

- **From USTs to partitions:** If every graph has a σ-stretch UST, then for any real $\gamma > 0$, every graph has an $(O(\sigma^2), O(\sigma), \gamma)$-partition. (Section 3.2)

We next obtain our main results for general graphs and minor-free graphs by constructing suitable partition hierarchies.

- **Partition hierarchies for general graphs:** Every graph G has a polynomial-time computable $(2^{O(\sqrt{\log n})}, 2^{O(\sqrt{\log n})}, 2^{O(\sqrt{\log n})})$-partition hierarchy. (Section 4)
- **Partition hierarchies for minor-free graphs:** Every minor-free graph G has a polynomial-time computable $(O(\log^3 n), O(\log^4 n), O(\log^3 n))$-partition hierarchy. (Section 6)

The partition hierarchy for general graphs is obtained by an iterative procedure in which clusters are continually merged by identifying vertices for which the number of intersecting clusters within a specified distance exceeds the desired bound. The particular order in which the vertices are processed is carefully chosen; a natural greedy approach fails.

Our construction of the partition hierarchy for minor-free graphs is much more complicated. It is based on a separator theorem due to [31, 2] which builds on the influential work of [22] and shows that any minor-free graph can be decomposed into connected components, each of which contains at most half the number of nodes, by removing a sequence of a constant number of shortest paths. A key ingredient of our hierarchical construction for minor-free graphs is a result on cluster aggregation in general graphs, which is of independent interest.

- **Cluster aggregation:** We show that given any partition of G into disjoint clusters each with strong diameter at most D, and a set S of portal vertices, we can aggregate the clusters into disjoint connected components, each component with a distinguished portal from S, such that for any vertex v, the distance, within the component of v, from v to the distinguished portal in the component is at most $O(\log^2 n)D$ more than the distance of v to S in G. (Section 5)

1.2 Related work

Research in network design over the past decade has revealed that it is often possible to derive sparse network structures (e.g., routes, multicast trees) that yield good approximations simultaneously for a range of input instances. One of the earliest examples of such a result is due to Goel and Estrin [15] who introduced the problem of *simultaneous single sink buy-at-bulk* and gave an $O(\log D)$ bound on the simulataneous ratio where D is the total demand. The simultaneous guarantee means that their solution works simultaneously for all fusion cost function f which are concave and monotonically non-deceasing with $f(0) = 0$. In a related paper [16], Goel and Post constructed a distribution over trees such that the expected cost of a tree for any f is within an $O(1)$-factor of the optimum cost for that f. A recent improvement by Goel and Post [17] provides the first constant guarantee on the simultaneous ratio achievable by a tree. This result is incomparable to our results since the set of terminals that are being aggregated in the buy-at-bulk problem are fixed.

Jia et al. [21] introduced the notion of universal approximation algorithms for optimization problems and provided approximation algorithms for TSP, Steiner Tree and set cover problems. For the universal Steiner tree problem, they presented polynomial-time algorithms that construct overlay trees with a stretch of $O(\log^4 n/\log\log(n))$ for arbitrary metrics and logarithmic stretch for doubling, Euclidean, or growth-restricted metrics. They also provided a lower bound of $\Omega(\log n/\log\log n)$ for UST that holds even when all the vertices are on a plane; for general metrics, this can be improved to $\Omega(\log n)$ [19, 9]. Note that these lower bounds extend to the UST problem on graphs. Lower bounds for universal TSP are given in [20, 18]. For earlier work on universal TSP, see [28, 8].

Gupta, Hajiaghayi and Räcke [19] developed an elegant framework to model *oblivious network design* problems and gave algorithms with poly-logarithmic approximation ratios. They give network structures that are simultaneously oblivious to both the link cost functions (subject to them being drawn from a suitable class) and traffic demand. Their algorithms are based on the celebrated tree metric embeddings of Fakcharoenphol et al. [14] and hierarchical cut-based decompostions of Räcke [29]. For the UST problem on metrics, the algorithm of [19] builds a UST as follows: First obtain $O(\log n)$ trees from the distribution of [14]; next assign each non-root vertex to a tree that well-approximates its distances to all other nodes; finally, take the union, over each of the $O(\log n)$ overlay trees, the subtree of the tree induced by the root and the vertices assigned to the tree. The resulting overlay tree is an $O(\log^2 n)$-stretch UST.

A potential approach to solving the UST problem on graphs is to adapt the techniques of [19] with

$O(\log n)$ spanning trees drawn from the distributions of [13] instead of the overlay trees of [14]. A major challenge here is that the paths or subtrees chosen from the different $O(\log n)$ trees may share several vertices and hence create unavoidable cycles when combined. The only prior work on constructing universal Steiner trees for graphs is due to Busch et al. [30] who achieved a stretch of $O(\log^3 n)$ for the restricted class of graphs with bounded doubling dimension by showing how one can continually refine an $O(\log n)$-stretch overlay tree by removing cycles to obtain an $O(\log^3 n)$-stretch UST. Their techniques, however, are closely tied to the particular class of graphs and seem difficult to generalize.

The aforementioned problem of approximating a graph metric by a tree metric has a rich history. Alon et al. [4] showed an upper bound of $2^{\sqrt{\log n \log\log n}}$ for approximating an arbitrary graph metric by a distribution over spanning trees. Bartal [6] showed that an improved $O(\log^2 n)$ approximation is achievable using tree metrics if one drops the requirement that the trees be subgraphs of the underlying graph. Konjevod et al. [23] improved Bartal's result to $O(\log n)$ for planar graphs while Charikar et al. [12] improved it for low dimensional normed spaces. Subsequently, Bartal [7] improved his earlier result to $O(\log n \log\log n)$ and also showed a lower bound of $\Omega(\log n)$ on the distortion for probabilistically embedding an expander graph into a tree. Fakcharoenphol, Rao and Talwar [14] closed the gap between the lower and the upper bound by showing that arbitrary metrics can be approximated by a distribution over tree metrics with distortion $O(\log n)$. More recently, Elkin, Emek, Spielman, and Teng [13] showed an upper bound of $O(\log^2 n \log\log n)$ for approximating an arbitrary graph metric using a distribution of spanning trees, thus significantly improving the result of Alon et al [4]. This result was subsequently improved by Abraham, Bartal, and Neiman [1] who achieved an $O(\log n \log\log n (\log\log\log n)^3)$ bound.

As mentioned in Section 1.1, our universal Steiner trees are based on certain partitions of graphs where we would like to bound the strong diameter of the clusters while maintaining some sparsity constraints. Such partitions have been extensively studied [27, 5]. While nearly optimal partitions based on weak diameter bounds are known in many cases, strong-diameter based decompositions are less understood [27]. There have been recent results on strong-diameter decompositions[1, 3]; while our partitions share some of the ideas (e.g., of stitching together judiciously chosen shortest paths), there are significant differences in the details and the particular partitions being sought. Furthermore, while we seek partition hierarchies with deterministic guarantees, these previous results concerned hierarchies with either probabilistic guarantees or covers where clusters are allowed to overlap.

2 Definitions and notations

Let $G = (V, E, w)$ denote a weighted undirected graph, where V and E are the sets of vertices and edges, respectively, and $w : E \to \mathbb{R}$ is a weight function on edges. We assume, without loss of generality, that the weight of a minimum-weight edge is 1, since otherwise we can scale all the edge weights appropriately. The weight of a path is simply the sum of the weights of the edges in it. For any u and v in V, let the distance between u and v, denoted by $d(u, v)$, be the weight of a shortest path (i.e. smallest weight path) between u and v, according to w. For $v \in V$ and real number ρ, let $B(v, \rho)$ denote the ball of radius ρ centered at v, i.e., $B(v, \rho)$ is the set of all vertices that are at distance at most ρ from v, including v. For any graph G, let *diameter* $\textsc{Diam}(G)$ denote the maximum distance between any two vertices of G. For any graph G and any subset X of vertices in G, let $G[X]$ denote the subgraph of G induced by X. For any subset X of vertices and vertices u and v in X, let $d_X(u, v)$ denote the distance between u and v in $G[X]$.

Universal Steiner trees. We now introduce some notations that help formalize the universal Steiner tree problem. Given a specified *root* vertex $r \in V$ and a set of *terminal* vertices $X \subseteq V$, a Steiner tree T for X is a minimal subgraph of G that connects the vertices of X to the root. The *cost* of tree T, denoted by $\textsc{Cost}(T)$, is the sum of the weights of edges in it. Assume G and r to be fixed. We let $\textsc{Opt}(X)$ denote the cost of the minimum weight steiner tree connecting X to r. Given a spanning tree T of G and terminal set

X, we define its projection on the terminal set X, denoted by T_X, as the minimal subtree of T rooted at r that contains X.

Definition 1 (Universal Steiner tree (UST))**.** *Let G be an undirected weighted graph, and r be a specified root vertex in V. We define the* stretch *of a spanning tree T of G to be $\max_{X \subseteq V} \frac{\text{COST}(T_X)}{\text{OPT}(X)}$. The* **universal Steiner tree** *problem is to find a spanning tree with minimum stretch.*

Partitions. A *partition* $\mathcal{P}$ of V is a collection of disjoint subsets of V whose union equals V. We refer to any subset of vertices, and hence each element of $\mathcal{P}$, as a *cluster* of the graph G. There are two notions for the diameter of a cluster. This paper focuses on the *strong* diameter, which is the diameter of the subgraph induced by the cluster. In contrast, the *weak* diameter of a cluster is simply the maximum distance between any two verices of the cluster in G.

Definition 2 ((α, β, γ)-partition)**.** *For any real $\gamma > 0$, an (α, β, γ)-partition $\mathcal{P}$ of G is a partition of V satisfying the following properties.*

1. **Strong diameter:** *The strong diameter of every cluster C in $\mathcal{P}$ is at most $\alpha\gamma$; i.e.,* $\text{DIAM}(G[C]) \leq \alpha\gamma$.
2. **Cluster-valence:** *For every vertex v in V, $B(v, \gamma)$ has a nonempty intersection with at most β clusters in $\mathcal{P}$. We refer to β as the cluster-valence of $\mathcal{P}$.*

A notion of partition similar to our (α, β, γ)-partition appeared in Jia et al. [21], which required a bound on the weak diameter of clusters.

Definition 3 (Partition hierarchy)**.** *For a given real $\gamma > 1$, an (α, β, γ)-partition hierarchy of a graph G is a sequence $\mathcal{H} = \langle \mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_d \rangle$ of partitions of V, where $d = \lceil \log_\gamma(\frac{\text{DIAM}(G)}{\alpha}) \rceil$, satisfying the following properties.*

1. **Partition:** *For $0 \leq i \leq d$, $\mathcal{P}_i$ is an $(\alpha, \beta, \gamma^i)$-partition of G. Furthermore, $\mathcal{P}_d$ is the collection $\{V\}$. For convenience, we set $\mathcal{P}_{-1}$ to the collection $\{\{v\} \mid v \in V\}$.*
2. **Hierarchy:** *For $0 \leq i < d$, every cluster in $\mathcal{P}_i$ is contained in some cluster in $\mathcal{P}_{i+1}$.*
3. **Root Padding:** *For $0 \leq i \leq d$, the ball $B(r, \gamma^i)$ of radius γ^i around root r is contained in some cluster in $\mathcal{P}_i$.*

For a partition $\mathcal{P}$, let $\mathcal{P}(v)$ denote the cluster of $\mathcal{P}$ that contains the vertex v and $\text{MAXDIAM}(\mathcal{P})$ denote $\max_{C \in \mathcal{P}} \text{DIAM}(C)$. For a subset X of vertices, let $\mathcal{P}[X]$ denote the partition restricted to X; i.e., $\mathcal{P}[X]$ is the collection $\{X \cap C \mid C \in \mathcal{P}\}$. For a partition hierarchy $\mathcal{H}$ and a cluster C that is an element of a partition $\mathcal{P}_i$ in $\mathcal{H}$, we let $\mathcal{H}[C]$ denote the partition hierarchy projected to C; that is,

$$\mathcal{H}[C] = \langle \mathcal{P}_0[C], \ldots, \mathcal{P}_i[C] \rangle.$$

3 Strong partitions and Universal Steiner trees

In this section, we present close connections between the strong partitions of Definition 2 and universal Steiner trees. We first show in Section 3.1 how partition hierarchies yield USTs. Given an (α, β, γ)-partition hierarchy for any graph G, Section 3.1.1 shows how to get an $O((\alpha\beta)^{\log_\gamma n} \gamma\beta^2 \log_\gamma n)$-stretch UST for G, and then Section 3.1.2 presents an improved construction yielding a stretch of $O(\alpha^2\beta^2\gamma \log_\gamma n)$. We next show, in a somewhat weaker sense, that partitions with low strong diameter and low cluster-valence are necessary for deriving low-stretch trees. In particular, Section 3.2 shows that if every graph has a σ-stretch UST, then every graph also has an $(O(\sigma^2), O(\sigma), \gamma)$-partition for all $\gamma > 0$.

3.1 From a partition hierarchy to a universal Steiner tree

Assume graph G and root vertex r to be fixed throughout this subsection. The main result here is an algorithm to construct an $O(\alpha^2\beta^2\gamma \log n)$-stretch UST from an (α, β, γ)-partition hierarchy which is presented in Section 3.1.2. We say that a spanning tree T of G *μ-respects* an (α, β, γ)-partition hierarchy $\langle \mathcal{P}_i \rangle$ if for any i, any cluster C of $\mathcal{P}_i$, and any vertices $u, v \in C$, $d_T(u, v)$ is at most $\mu\alpha\gamma^i$.

Lemma 4. *A spanning tree T that μ-respects an (α, β, γ)-partition hierarchy has a stretch of $O(\mu\alpha\beta\gamma \log n)$.*

Proof. Let $\langle \mathcal{P}_i \rangle$ denote the given (α, β, γ)-partition hierarchy. Fix a non-empty set X of vertices. Note that X is assumed to not contain the root r. For each cluster C in the partition hierarchy such that $C \cap (X \cup \{r\})$ is nonempty, let $v(C)$ denote an arbitrary vertex in $C \cap (X \cup \{r\})$.

We place an upper bound on the cost of T_X, the subgraph of T connecting the vertices in X to the root r, as follows. Let n_i denote the number of clusters in $\mathcal{P}_i$ that $X \cup \{r\}$ intersects. Since we have defined $\mathcal{P}_{-1}$ to be the trivial clustering consisting of a singleton set for each vertex, n_{-1} is simply $|X \cup \{r\}|$. Let j be the smallest integer such that X is a subset of the cluster in $\mathcal{P}_j$ that contains r. In other words, n_j equals 1 and $n_i > 1$ for all $-1 \le i < j$. Fix an i, $-1 \le i < j$. Let C be any cluster of $\mathcal{P}_i$ that intersects $X \cup \{r\}$, and let C' denote the cluster of $\mathcal{P}_{i+1}$ that contains C. Since T μ-respects the partition hierarchy, it follows that the length of the path from $v(C)$ to $v(C')$ in T is at most $\mu\alpha\gamma^{i+1}$. Therefore, the cost of T_X is at most $\sum_{-1 \le i < j} n_i \mu\alpha\gamma^{i+1}$. Let $I = \{i : (i = j) \vee (-1 \le i < j \wedge \exists p : n_i \ge 2^p \wedge n_{i+1} < 2^p)\}$. For $\ell \in I$, let $I_\ell = \{i : (-1 \le i \le \ell) \wedge \neg(\exists \ell' \in I : i \le \ell' < \ell)\}$. We have,

$$\sum_{i \in I_\ell} n_i \mu\alpha\gamma^{i+1} \le \sum_{i \in I_\ell} 2n_\ell \mu\alpha\gamma^{i+1} \le \sum_{-1 \le i \le \ell} 2n_\ell \mu\alpha\gamma^{i+1} = O(n_\ell \mu\alpha\gamma^{\ell+1})$$

We next place a lower bound on $\textsc{Opt}(X)$. Fix an i, $0 \le i < j$. By the cluster-valence property of the hierarchy, any ball of radius γ^i intersects at most β clusters in $\mathcal{P}_i$. Thus, there are at least $\lceil n_i/\beta \rceil$ vertices in X that are at pairwise distance at least γ^i from one another. This implies that $\textsc{Opt}(X)$ is at least $(\lceil n_i/\beta \rceil - 1)\gamma^i$. If $\lceil n_i/\beta \rceil = 1$, we invoke the padding property which says there is at least one vertex in X that is at distance at least γ_i from the root, implying a lower bound of γ^i on $\textsc{Opt}(X)$. Combining the two bounds, we obtain a lower bound of $\Omega(n_i\gamma^i/\beta)$. For $i = -1$, we also have a lower bound of n_{-1} since the minimum edge-weight is 1. Noting that $|I| = O(\log n)$, we get the stretch of $T(G)$ to be

$$O\left(\sum_{\ell \in I} \frac{\sum_{i \in I_\ell} n_i \mu\alpha\gamma^{i+1}}{\textsc{Opt}(X)}\right) = O\left(\sum_{\ell \in I} \frac{n_\ell \mu\alpha\gamma^{\ell+1}}{n_\ell\gamma^\ell/\beta}\right) = O\left(\sum_{\ell \in I} \mu\alpha\gamma^{\ell+1}\beta/\gamma^\ell\right) = O(\mu\alpha\beta\gamma \log n).$$

□

3.1.1 A bottom-up divide-and-conquer algorithm

We first present a bottom-up divide-and-conquer algorithm for constructing a spanning tree T. Though the stretch achieved by the spanning tree is much weaker than what we obtain by a different algorithm, it helps develop our improved algorithm.

Theorem 5. *For any graph G, given an (α, β, γ)-partition hierarchy, an $O((\alpha\beta)^{\log_\gamma n}\gamma\beta^2 \log n)$-stretch USTis computed by Algorithm 1 in polynomial time.*

Proof. Given a graph G, the algorithm builds a spanning tree $T(G)$ by iteratively building spanning trees for each cluster in a (α, β, γ)-partition hierarchy of G, in a bottom-up manner. We first show by induction on i that for any cluster $C \in \mathcal{P}_i$, the strong diameter of $T(C)$ is at most $(\alpha\beta\gamma)^{i+1} - 1$. The induction basis directly follows from the strong diameter property of an $(\alpha, \beta, 1)$-partition.

Algorithm 1 A basic divide-and-conquer algorithm

Require: Undirected graph G, strong (α, β, γ)-partition hierarchy $\{\mathcal{P}_i : 0 \leq i \leq k\}$ for G
Ensure: A spanning tree T of G

```
for level i from 0 to k do
    for cluster C in P_i do
        For an edge e = (C_1, C_2) in Ĝ[P_{i-1}[C]], let m_S(e) denote the edge between C_1 to C_2 in G that has minimum weight. (Recall that P_{i-1}[C] is the partition P_{i-1}, restricted to the set C.)
        Compute a shortest path tree T' from an arbitrary source vertex in G_S.
        Set T(C) to be the union of ∪_{C'∈S} T(C') and {m_S(e) : e ∈ T'}.
    end for
end for
Set T to be T(V) (note that V is the lone cluster in P_k).
```

We now establish the induction step. Let C be a cluster in $\mathcal{P}_i$ and let u and v be two vertices in C. By the hierarchy property, C is the union of a set, say S, of clusters in $\mathcal{P}_{i-1}$. Since $\mathcal{P}_i$ is an $(\alpha, \beta, \gamma^i)$-partition, it follows that the strong diameter of C is at most $\alpha\gamma^i$. Hence, there exists a path P between u and v in C of length at most $\alpha\gamma^i$. By the intersection property, any ball of radius γ^{i-1} intersects at most β clusters in $\mathcal{P}_{i-1}$, and hence at most β clusters in S. Therefore, the path P intersects at most $\alpha\beta\gamma$ clusters in S. Thus, the diameter of G_S is at most $\alpha\beta\gamma$. By the induction hypothesis, it follows that the strong diameter of $T(C)$ is at most $\alpha\beta\gamma - 1 + \alpha\beta\gamma((\alpha\beta\gamma)^i - 1)$, which equals $(\alpha\beta\gamma)^{i+1} - 1$.

Since $(\alpha\beta\gamma)^{i+1}/(\alpha\gamma^i)$ is at most $\alpha^k\beta^{k+1}\gamma$ for $k = \log_\gamma n$, it follows that T $(\alpha^{k-1}\beta^{k+1}\gamma)$-respects the strong partition hierarchy. By Lemma 4, we obtain that T has stretch at most $O((\alpha\beta)^{\log_\gamma n}\gamma\beta^2 \log n)$, completing the proof of the theorem.

□

3.1.2 Split and join: An improved top-down algorithm

The tree returned by Algorithm 1 completely obeys the given partition hierarchy in the sense that for any cluster C of the hierarchy, $T[C]$ is, in fact, a tree; that is, T completely respects the connectivity of each cluster of the hierarchy. In doing so, however, it pays a huge cost in the distances within the cluster. And, in fact, this is unavoidable; one can construct examples of strong partition hierarchies where obeying the connectivity of each cluster in the tree will result in a significant blow-up of stretch.

We now present a much more careful construction of a universal Steiner tree which does not enforce the connectivity constraint within clusters; that is, we use a given partition hierarchy $\mathcal{H}$ to build a tree T in which $T[C]$ may be disconnected. By allowing this disconnectivity within clusters, however, we show that we can build a tree that μ-respects the given hierarchy for a much smaller μ, assuming γ is sufficiently large. The pseudocode is given in Algorithm 2. We have presented the algorithm in a more general context where the goal is to compute a forest rooted at a given set of portals. To obtain a UST, we invoke the algorithm with the portal set being the singleton set consisting of the root.

Before presenting our algorithm, we introduce some useful notation. For a partition $\mathcal{P}$ of a graph H, let $\widehat{H}[\mathcal{P}]$ denote a graph in which the vertex set is $\mathcal{P}$, and there is an edge (C, C') from cluster C to C' if H has an edge between a vertex in C and a vertex in C'; the weight of the edge (C, C') is the weight of the minimum-weight edge between C and C' in H.

Lemma 6. *The output F of the algorithm is a spanning forest, each tree of which contains exactly one vertex in S_G.*

Proof. The proof is by induction on the number of recursive calls to the UST algorithm. For the induction

Algorithm 2 UST: The split and join algorithm

Require: Undirected graph $G = (V, E)$, a nonempty set $S_G \subseteq V$ of portals, a partition hierarchy $\mathcal{H} = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_\ell\}$.

Ensure: A forest F that connects every vertex in V to S_G.

If the graph consists of a single vertex, then simply return the vertex as the forest.
For an edge $e = (C_1, C_2)$ in $\widehat{G}[\mathcal{P}_\ell]$, let $m(e)$ denote the minimum-weight edge from C_1 to C_2 in G.
Let $\widehat{S}$ denote the set of clusters that have a nonempty intersection with S_G.
for cluster C in $\widehat{S}$ **do**
    Set S_C to be $C \cap S$.
end for
Compute a shortest path forest $\widehat{F}$ in $\widehat{G}[\mathcal{P}_\ell]$ rooted at $\widehat{S}$.
for cluster C in $\mathcal{P}_\ell$ in order of decreasing distance from $\widehat{S}$ in $\widehat{F}$ **do**
    if C is a leaf node in $\widehat{F}$ **then**
        Set RANK(C) to be 0, S_C to be {tail of $m(e)$} where e is the edge connecting C to its parent in $\widehat{F}$.
    else
        Let MAXC be $\max\{\text{RANK}(C') \mid C' \text{ is child of } C\}$. Set FAV$(C)$ to be a child of C with rank MAXC. Set HIGHWAY(C) to be a shortest path in C from the head of $m(e)$ to the tail of $m(e')$ where e and e' are the edges connecting FAV(C) to C and C to its parent, respectively, in $\widehat{F}$. Set S_C to be the set of nodes in HIGHWAY(C).
        if there exist at least two children of C in $\widehat{F}$ whose rank equals MAXC **then**
            Set RANK(C) to be MAXC $+ 1$
        else
            Set RANK(C) to be MAXC
        end if
    end if
end for
for each cluster C in $\mathcal{P}_\ell$ **do**
    Compute $F(C) = \text{UST}(G[C], S_C, \mathcal{H}[C])$
end for
Return F to be the union of $\bigcup_{C \in \mathcal{P}_\ell} \text{HIGHWAY}(C)$, $\bigcup_{C \in \mathcal{P}_\ell} F(C)$, and $\{m(e) : e \in \widehat{F}\}$.

base, we consider the case where the graph consists of a single vertex; in this case, the algorithm returns the vertex as the forest, which satisfies the desired claim.

For the induction step, we note that the forest F returned is the union of three sets: (a) union of HIGHWAY(C) over all C in $\mathcal{P}_\ell$; (b) union of $F(C)$ over all C in $\mathcal{P}_\ell$; and (c) $\{m(e) : e \in \widehat{F}\}$. By the induction hypothesis, each $F(C)$ is a forest spanning C, each tree of which contains exactly one vertex of S_C. We distinguish between two kinds of clusters. If C is in $\widehat{S}$, then $F(C)$ is a forest, each tree of which contains exactly one vertex of $S_G \cap C$. Otherwise, $F(C)$ is a forest, each tree of which contains exactly one vertex of HIGHWAY(C). It thus follows that the union of (a) and (b) above gives a forest for each cluster C satisfying the following condition: if C is in $\widehat{S}$, the forest contains a spanning forest of C, each tree of which contains exactly one vertex of $S_G \cap C$; otherwise, the forest contains a spanning tree of C.

Finally, the edges of (c) connect the clusters not in $\widehat{S}$ to the clusters in $\widehat{S}$ via a forest. Consequently, adding these edges to the forest formed by (a) and (b) yields a spanning forest over G, each tree of which contains exactly one vertex in S_G. □

Lemma 7. *Let F be the final forest returned by the algorithm. For any cluster C, when UST is called on cluster C, either S_C is a subset of S_G or for any two vertices u and v in S_C, $d_F(u, v)$ is at most $d_C(u, v)$.*

Proof. We first prove that for any cluster C, the set S_C is exactly one of the following: (i) S_G, if C is G; or (ii) a subset of $S_{C'}$ for the parent cluster C' of C; or (iii) a subset of nodes on a HIGHWAY(C) constructed when processing parent cluster C'. The proof is by induction on the level of the hierarchy. The base case is trivial for C being the whole graph. For the induction step, consider level $i \leq \ell$ of the hierarchy, and let C be a cluster in $\mathcal{P}_i$.

We consider two cases. In the first case, C intersects $S_{C'}$ where C' is the parent cluster for C. In this case, S_C is set to the intersection of C and $S_{C'}$ as desired. In the second case, C is disjoint from $S_{C'}$. In this case, S_C is simply the set of vertices in HIGHWAY(C), again completing the induction step.

To complete the proof of the lemma, consider a cluster C in $\mathcal{P}_i$, for some i. If S_C is a subset of S_G, the lemma trivially follows. If S_C is not a subset of S_G, then by the above claim, S_C equals the set of nodes in HIGHWAY(C). By our construction, HIGHWAY(C) is a shortest path in C. Since HIGHWAY(C) is part of F, it follows that for any two vertices u and v in S_C, $d_F(u, v)$ equals $d_C(u, v)$. □

Lemma 8. *The rank of any cluster C in partition $\mathcal{P}_\ell$ is at most* $\log(|\mathcal{P}_\ell|)$.

Proof. Let $\widehat{F}$ denote the shortest path forest in $\widehat{G}[\mathcal{P}_\ell]$. We show that for any cluster C, the rank of C is at most $\log(m_C)$, where m_C is the number of nodes in the subtree of $\widehat{F}$ rooted at C.

The proof is by induction on the height of C. The induction basis is immediate for the leaves of $\widehat{F}$. We now consider the induction step. For cluster C, let r denote the rank of the child with highest rank among all children of C. Let Z denote the set of children of C that have rank r. We note that m_C is at least $1 + \sum_{C' \in Z} m_{C'}$. Furthermore, by the induction hypothesis, $m_{C'}$ is at least 2^r, for each C' in Z. We consider two cases. If $|Z|$ is 1, then the rank of C equals r, which is at most $\log(m_C)$ by the induction hypothesis. In $|Z| \geq 2$, then the rank of C equals $r + 1$; since m_C is at least $1 + 2 \cdot 2^r > 2^{r+1}$, the induction step follows, completing the proof. □

Lemma 9. *Let F be the final forest returned by the algorithm. If $\gamma \geq \log n$, then for any cluster C in $\mathcal{P}_i$ and vertex u in C, $d_F(u, S_C)$ is at most $3\alpha^2\beta\gamma^i$.*

Proof. We prove by induction on level i that $d_F(u, S_C)$ is at most $3\alpha^2\beta\gamma^i$, with the base case being $i = 0$. In this case, the cluster and its portal set are the same singleton vertex set, trivially yielding the desired claim. For the induction step, we consider $i > 0$. Let C be a cluster of $\mathcal{P}_i$. For any vertex u in C, let C_u denote $\mathcal{P}_{i-1}(u)$, that is, the cluster in partition $\mathcal{P}_{i-1}$ that contains u.

As in the algorithm, let $\widehat{S}$ denote the set of clusters in the partition of C that intersect S_C. Let $C_u = C_0, C_1, \ldots, C_k$, where $C_k \in \widehat{S}$, denote the sequence of clusters in the unique path from C_u to $\widehat{S}$ in $\widehat{\mathcal{P}_\ell}[]$, which we refer to as the supergraph in the following argument. Note that C_i is the parent of C_{i-1} in the supergraph. By our argument in the proof of Theorem 5, we know that k is at most $\alpha\beta\gamma$. We now argue that there are at most $\log n$ elements C_i in the sequence such that C_i is not FAV(C_{i+1}). To see this, we note that if C_i is not FAV(C_{i+1}), then RANK(C_{i+1}) strictly exceeds RANK(C_i). Since the rank of any cluster is at most $\log n$ by Lemma 8, the desired claim holds.

This sequence of clusters induces a path from u to S_C, which consists of (a) the connecting edges in the supergraph, (b) the highway in each cluster C_i in the sequence, (c) for each cluster C_i such that C_{i-1} is not a favorite of C_i, the unique path in $F(C_i)$ (and, hence, in F) that connects the head of the edge connecting C_{i-1} and C_i to S_{C_i}. Since the number of clusters in the sequence is at most $\alpha\beta\gamma$, and the highway in each cluster is a shortest path, the total length of the paths in (a) and (b) is at most $2\alpha^2\beta\gamma^i$. The number of clusters in (c) is at most $\log n$, and by the induction hypothesis, the length of each path in (c) is at most $3\alpha^2\beta\gamma^{i-1}$. We thus have,

$$\begin{aligned} d_F(u, S_C) &\leq 2\alpha^2\beta\gamma^i + 2\log n\alpha^2\beta\gamma^{i-1} \\ &\leq 3\alpha^2\beta\gamma^i \end{aligned}$$

for $\gamma \geq 2 \log n$, thus completing the proof of the lemma. □

Lemma 10. *For any cluster C in $\mathcal{P}_i$, and vertices u, v in C, $d_F(u, v)$ is at most $7\alpha^2\beta\gamma^i$.*

Proof. By Lemma 9, $d_F(u, S_C)$ and $d_F(v, S_C)$ are both at most $3\alpha^2\beta\gamma^i$. By Lemma 7, for any two nodes x and y in S_C, $d_F(x, y)$ is at most the strong diameter of C, which is at most $\alpha\gamma^i$. Putting these three distances together, we obtain that $d_F(u, v)$ is at most $7\alpha^2\beta\gamma^i$. □

Theorem 11. *Given an undirected graph G, portal set $S_G = \{r\}$, where r is an arbitrary vertex of G, and (α, β, γ)-partition $\mathcal{H}$ of G as input, Algorithm 2 returns an $O(\alpha^2\beta^2\gamma \log n)$-stretch UST.*

Proof. By Lemma 6, the output F is a spanning forest, each tree of which contains exactly one vertex of S_G. Since S_G has only one vertex, the forest F returned is a tree. By Lemma 10, for any cluster C in any partition at level i of $\mathcal{H}$, and any two vertices u and v in C, we have $d_F(u, v)$ is at most $7\alpha^2\beta\gamma^i$. It thus follows that F $(7\alpha\beta)$-respects $\mathcal{H}$. By Lemma 4, we obtain that F has stretch $O(\alpha^2\beta^2\gamma \log n)$. □

3.2 From universal Steiner trees to strong partitions

We show how to construct partitions with low strong diameter and low cluster-valence for all graphs given an algorithm to construct low-stretch USTsfor all graphs.

Theorem 12. *Given an algorithm $\mathcal{A}$ to construct a σ-stretch UST for all graphs in polynomial time, we can obtain a polynomial-time algorithm $\mathcal{A}'$ to construct an $(O(\sigma^2), O(\sigma), \gamma)$-partition for all graphs and all $\gamma > 0$ which uses $\mathcal{A}$ as a black box.*

Proof. Assume we have algorithm $\mathcal{A}$ that finds a σ-stretch UST for all graphs in polynomial time. The algorithm $\mathcal{A}'$ works as follows. Given graph $G = (V, E, w)$, it constructs graph $G' = (V', E', w')$ where $V' = V \cup \{r\}$, $E' = E \cup \{(r, v) : v \in V\}$ and w' extends w to E' by simply assigning $w((r, v)) = 2\sigma\gamma$ for all $v \in V$. Here r is an additional vertex not in V. $\mathcal{A}'$ invokes $\mathcal{A}$ with graph G' and root vertex r as inputs. Let T be the tree rooted at r output by $\mathcal{A}$ and $T_1, \ldots, T_k$ be the subtrees of T connected directly to the root r by single edges. $\mathcal{A}'$ simply outputs the partition $\mathcal{P} = \{C_1, \ldots, C_k\}$, where C_i is the set of vertices in T_i. We now argue that $\mathcal{P}$ is a $(O(\sigma^2), O(\sigma), \gamma)$-partition of G.

Lemma 13. *The strong diameter of each C_i is at most $4\sigma(\sigma - 1)\gamma$.*

Proof. Fix a C_i. It is enough for us to prove that the height of the tree T_i is at most $2\sigma(\sigma - 1)\gamma$ as we can reach any vertex in C_i from any other while remaining within C_i by going through the root of T_i. Assume not. Then there is a vertex v in tree T_i whose distance in this tree from the root of T_i is more than $2\sigma(\sigma-1)\gamma$. Consider the graph G' with the root vertex r for which $\mathcal{A}$ returned T. $\textsc{Cost}(T_{\{v\}})$ is more than $2\sigma\gamma + 2\sigma(\sigma - 1)\gamma = 2\sigma^2\gamma$, while $\textsc{Opt}(\{v\})$ is $2\sigma\gamma$. Thus $\frac{\textsc{Cost}(T_{\{v\}})}{\textsc{Opt}(\{v\})} > \sigma$, which contradicts the fact that T is a σ-stretch UST for G'. □

Lemma 14. *For any vertex $v \in V$, $B(v, \gamma)$ intersects at most 2σ clusters of $\mathcal{P}$.*

Proof. The proof is by contradiction. Suppose there is a vertex v such that $B(v, \gamma)$ intersects $d > 2\sigma$ clusters of $\mathcal{P}$. We select one vertex from each of these d different clusters such that the selected vertices lie in $B(v, \gamma)$, and call this set S. Now consider the graph G' with the root vertex r for which $\mathcal{A}$ returned T. Since each vertex in S lies in a different T_i in T, $\textsc{Cost}(T_S)$ is at least $2\sigma\gamma d$. On the other hand, $\textsc{Opt}(S)$ is at most $2\sigma\gamma + d\gamma = (2\sigma + d)\gamma$ as v is at a distance $2\sigma\gamma$ from r and each of the d vertices in S are at most a distance γ away from v. Thus $\frac{\textsc{Cost}(T_{\{v\}})}{\textsc{Opt}(\{v\})} = \frac{2\sigma d}{2\sigma + d} > \sigma$ by our choice of d, which again contradicts the fact that T is a σ-stretch USTfor G'. □

The theorem follows from the above two lemmas. □

4 Partition hierarchy for general graphs

In this section we present our algorithm for obtaining a partition hierarchy for general graphs. As mentioned in section 2, we start the hierarchy at level -1 by defining $\mathcal{P}_{-1}$ as the trivial partition where every vertex is in its own cluster. Our main result is the following.

Theorem 15. *For any graph G and any integer $k \geq 0$, a hierarchical $((\frac{4}{3}+\epsilon)4^{k-1}-\frac{4}{3}, kn^{\frac{1}{k}}, \gamma)$-partition can be constructed in polynomial time for $\gamma \geq \frac{1}{\epsilon}((\frac{4}{3}+\epsilon)4^{k-1}-\frac{4}{3})$. In paricular, setting $k = \lceil\sqrt{\log n}\rceil$, we obtain a hierarchical $(2^{O(\sqrt{\log n})}, 2^{O(\sqrt{\log n})}, 2^{O(\sqrt{\log n})})$-partition for any graph in polynomial time.*

Algorithm. For $i = 0, \ldots, \lceil\log_\gamma \frac{\text{DIAM}(G)}{\alpha}\rceil$, we build the ith level of the hierarchy, $\mathcal{P}_i$, after building the previous levels. Assuming that the level $i-1$ partition $\mathcal{P}_{i-1}$ has been constructed, we construct $\mathcal{P}_i$ as follows.

Clusters of $\mathcal{P}_i$ are formed in successive stages starting from stage 0. We assign a rank to each cluster based on the stage in which it is created: a cluster formed in stage j gets the rank j. (All the clusters of level -1 are assigned the rank 0.) We will denote the set of clusters of rank j (of level i) by S_j^i. A cluster of a higher rank is formed by merging clusters of lower ranks. At all times, we maintain a partitioning of the graph, i.e., we guarantee that each vertex of the graph is contained in exactly one cluster of level i.

In stage 0, we simply add all the clusters of $\mathcal{P}_{i-1}$ to S_0^i. For $j \geq 1$, stage j works in two phases as follows. In the first phase, we repeatedly look for a vertex *contained in a cluster of rank at most $j-1$* such that the ball of radius γ^i around it, $B(v,\gamma^i)$, intersects more than $n^{\frac{1}{k}}$ clusters of rank precisely $j-1$. If we find such a vertex v, we merge the cluster containing v with all the clusters of rank $j-1$ that $B(v,\gamma^i)$ intersects. This newly created cluster is assigned the rank j and added to S_j^i while all the clusters that were merged to form it are deleted from their respective $S_{j'}^i$'s. The first phase ends when we can no longer find any such vertex v.

In the second phase, we repeat a simlar procedure for vertices *contained in clusters of rank j*. As long as we can find a vertex v in a cluster of rank j such that $B(v,\gamma^i)$ intersects more than $n^{\frac{1}{k}}$ clusters of rank $j-1$, we merge all these clusters of rank $j-1$ with the cluster containing v to form a new cluster of rank j. We include this new cluster in S_j^i and delete all the clusters that were merged to form it from their respective $S_{j'}^i$'s. The second phase, and also the stage j, ends when we cannot find any such vertex v, and the next stage begins.

If no new cluster gets formed in the first phase of a stage, the construction of level i of the hierarchy finishes and $\mathcal{P}_i$ is defined as simply the union of all the non empty S_j^i's.

Remark. Although the two phases of a stage are quite similar and one might be tempted to do away with this particular ordering of mergings, the naive approach without the ordering does not work. Having a careful order in which mergings are carried out enables us to control the growth of the strong diameter of the clusters. To see this, consider a cluster formed in the second phase of some stage j. It contains a unique cluster that was formed in the first phase of stage j (call it the core). Our ordering ensures that only the vertices in the core can lead to mergings in the second phase of stage j. This is because for any vertex v outside the core, $B(v,\gamma^i)$ intersects at most $n^{\frac{1}{k}}$ clusters of rank $j-1$, otherwise the first phase would not have ended. Thus the mergings of the second phase cannot increase the diameter much as the new vertices are always "close" to the core.

We now analyze the algorithm, which is presented in pseudocode in Algorithm 3. We have the following claims that bound the size and diameter of the clusters of level i.

Algorithm 3 Algorithm to obtain a partition hierarchy for general graphs

Require: A weighted graph $G = (V, E, w)$, integer k, $\gamma \geq \frac{1}{\epsilon}((\frac{4}{3}+\epsilon)4^{k-1} - \frac{4}{3})$

Ensure: A hierarchical $(\alpha = (\frac{4}{3}+\epsilon)4^{k-1} - \frac{4}{3}, \beta = kn^{\frac{1}{k}}, \gamma)$-partition of G

Define $\mathcal{P}_{-1}$ to be the trivial partition where each vertex of V is in its own cluster, i.e., $\mathcal{P}_{-1} = \{\{v\} : v \in V\}$.
for level i from 0 to $\lceil \log_\gamma(\frac{\text{DIAM}(G)}{\alpha}) \rceil$ **do**
  $S_0^i = \mathcal{P}_{i-1}$.
  $S_j^i = \emptyset$ for all $1 \leq j \leq k-1$.
  $j \leftarrow 1$.
  while $j < k$ and $S_{j-1}^i \neq \emptyset$ **do**
    while there exists a v such that $v \in C_v$ for some $C_v \in S_{j_v}^i$ and $j_v < j$, and $B(v, \gamma^i)$ intersects more than $n^{\frac{1}{k}}$ clusters from S_{j-1}^i **do**
      Delete C_v from $S_{j_v}^i$, i.e., $S_{j_v}^i = S_{j_v}^i \setminus \{C_v\}$.
      Delete all the clusters of S_{j-1}^i that $B(v, \gamma^i)$ intersects from it, i.e., $S_{j-1}^i = S_{j-1}^i \setminus \{C : C \in S_{j-1}^i \wedge B(v, \gamma^i) \cap C \neq \emptyset\}$.
      Merge C_v and all the clusters deleted from S_{j-1}^i and add to S_j^i, i.e., $S_j^i = S_j^i \cup C_v \cup \left(\bigcup_{C \in \mathcal{X}} C\right)$, where $\mathcal{X}$ equals $\{C \in S_{j-1}^i : B(v, \gamma^i) \cap C \neq \emptyset\}$.
    end while
    while there exists a v such that $v \in C_v$ for some $C_v \in S_j^i$, and $B(v, \gamma^i)$ intersects more than $n^{\frac{1}{k}}$ clusters from S_{j-1}^i **do**
      Delete C_v from S_j^i, i.e., $S_j^i = S_j^i \setminus \{C_v\}$.
      Delete all the clusters of S_{j-1}^i that $B(v, \gamma^i)$ intersects from it, i.e., $S_{j-1}^i = S_{j-1}^i \setminus \{C : C \in S_{j-1}^i \wedge B(v, \gamma^i) \cap C \neq \emptyset\}$.
      Merge C_v and all the clusters deleted from S_{j-1}^i and add to S_j^i, i.e., $S_j^i = S_j^i \cup C_v \cup \left(\bigcup_{C \in \mathcal{Y}} C\right)$, where $\mathcal{Y}$ equals $\{C \in S_{j-1}^i : B(v, \gamma^i) \cap C \neq \emptyset\}$.
    end while
    $j = j + 1$.
  end while
  $\mathcal{P}_i = \cup_{t=0}^{t=k-1} S_t^i$.
end for
Output $(\mathcal{P}_0, \ldots, \mathcal{P}_{\lceil \log_\gamma(\frac{\text{DIAM}(G)}{\alpha}) \rceil})$.

Lemma 16. *The size (number of vertices) of a cluster of rank j at any level is at least $n^{\frac{j}{k}}$.*

Proof. We prove the claim using induction on j. For $j = 0$, the claim follows trivially as each cluster of any rank has size at least 1. For the induction step, observe that a cluster of rank j contains more than $n^{\frac{1}{k}}$ clusters of rank $j-1$ which all have size at least $n^{\frac{j-1}{k}}$ by the induction hypothesis. □

Corollary 17. *The rank of any cluster of any level can be at most $k-1$.*

Proof. From the previous lemma it follows that at any level there can be at most $\frac{n}{n^{\frac{k-1}{k}}} = n^{\frac{1}{k}}$ clusters of rank $k-1$ which immediately implies that no cluster of rank k gets formed. □

Lemma 18. *The strong diameter of every cluster of level i and rank j is at most $\gamma^i((\frac{4}{3}+\epsilon)4^j - \frac{4}{3})$, provided $\gamma \geq \frac{1}{\epsilon}((\frac{4}{3}+\epsilon)4^{k-1} - \frac{4}{3})$.*

Proof. We prove the claim by induction on i and j. The case for $i = -1$ is trivially true. For the case of $i \geq 0$, assume the claim to be true for every cluster of level $i-1$. Since a cluster of level i and rank 0 is simply one of these clusters, its diameter is bounded by $\gamma^{i-1}((\frac{4}{3}+\epsilon)4^{k-1}-\frac{4}{3})$ by the induction hypothesis and the above corollary. This is at most $\gamma^i((\frac{4}{3}+\epsilon)4^0-\frac{4}{3}) = \gamma^i\epsilon$ by our assumption that $\gamma \geq \frac{1}{\epsilon}((\frac{4}{3}+\epsilon)4^{k-1}-\frac{4}{3})$ which proves the claim for level i and rank 0.

Now assume that the claim is true for level i and all rank at most $j-1$, and consider a cluster C at level i and rank j. There are two cases to consider depending upon whether C was formed in the first or second phase of stage j.

If C was formed in the first phase, it is the union of the cluster containing v and all the clusters of rank $j-1$ that the ball $B(v,\gamma^i)$ intersects, where v is contained in a cluster of rank at most $j-1$. By the induction hypothesis, the strong diameters of all these clusters which were merged to form C are bounded by $\gamma^i((\frac{4}{3}+\epsilon)4^{j-1}-\frac{4}{3})$. This implies that any vertex in C is at most a distance $\gamma^i((\frac{4}{3}+\epsilon)4^{j-1}-\frac{4}{3})+\gamma^i$ from v. Thus the strong diameter of C is at most $2\gamma^i((\frac{4}{3}+\epsilon)4^{j-1}-\frac{4}{3}+1) \leq \gamma^i((\frac{4}{3}+\epsilon)4^j-\frac{4}{3})$ as $j \geq 1$.

If C was formed in the second phase, it implies that there was a cluster C' of rank j which was formed in the first phase of stage j and got merged with other clusters to form C in the second phase. By the argument above, the strong diameter of C' was at most $2\gamma^i((\frac{4}{3}+\epsilon)4^{j-1}-\frac{4}{3})+1)$. Furthermore, we know that any vertex in C either comes from C' or from some cluster of rank $j-1$ which intersects the ball $B(v,\gamma^i)$ for a vertex v contained in C'. From the above facts and the induction hypothesis, we conclude that the strong diameter of C is bounded by $2\gamma^i((\frac{4}{3}+\epsilon)4^{j-1}-\frac{4}{3})+1)+2\gamma^i+2\gamma^i((\frac{4}{3}+\epsilon)4^{j-1}-\frac{4}{3}) = \gamma^i((\frac{4}{3}+\epsilon)4^j-\frac{4}{3})$. □

Now we are in a position to prove Theorem 15 which gives the partition hierarchy for general graphs.
Proof of T: he bound on cluster diameter is given by Lemma 18. For the intersection bound, observe that for any level i of the hierarchy and any vertex v, the ball $B(v,\gamma^i)$ can intersect at most $n^{\frac{1}{k}}$ clusters of a given rank. This implies that $B(v,\gamma^i)$ can intersect at most $kn^{\frac{1}{k}}$ clusters in total from level i as every cluster has rank between 0 and $k-1$. □

Corollary 19. *A $2^{O(\sqrt{\log n})}$-stretch universal Steiner tree can be computed in polynomial time for any undirected graph.*

5 The Cluster Aggregation Problem

In this section, we define the Cluster Aggregation problem which arises when building partition hierarchies for minor-free graphs (see Section 6). Our problem formulation and algorithm, however, apply to arbitrary graphs and may be of independent interest. Indeed, our cluster aggregation algorithm is useful for building other strong-diameter based hierarchical partitions with applications to distributed computing [10].

Definition 20 (Cluster Aggregation)**.** *Given a graph $G = (V, E)$, partition $\mathcal{P}$ of G, set $S \subseteq V$ of* portals*, a* cluster aggregation *is a function* $\text{DEST} : \mathcal{P} \to S$*. The function* DEST *naturally induces a new partition $\mathcal{Q} = \{\bigcup_{C:\text{DEST}(C)=s} C \mid s \in S\}$ that coarsens $\mathcal{P}$. For each vertex v in V, we define the* detour $\text{DTR}_{\text{DEST}}(v)$ *for v under* DEST *to be the difference between the distance from v to S in G and the distance from v to* $\text{DEST}(\mathcal{P}(v))$ *in subgraph of G induced by the cluster in $\mathcal{Q}$ that contains v; i.e.,* $\text{DTR}_{\text{DEST}}(v) = (d_{G[\mathcal{Q}v]}(v, \text{DEST}(C)) - d(v,S))$*. We define the detour of* DEST *to be* $\max_{v\in V} \text{DTR}_{\mathcal{Q}}(v)$*. The goal of the cluster merging problem is to find a cluster aggregation with minimum detour.*

Our algorithm for the Cluster Aggregation problem proceeds in $O(\log n)$ phases. Each phase has a number of iterations. Each iteration aggregates a subset of the clusters in $\mathcal{P}$ and assigns the same DEST value for each of them. The selection of clusters in a particular iteration is based on how shortest paths from these clusters to S proceed through the graph. The interaction of these shortest paths is captured by means of auxiliary directed graph. For any directed graph K and set A of vertices in K, let $\text{in}_K(A)$ (resp.,

$\text{out}_K(A)$) denote the set of vertices that have an edge into (resp., from) any vertex in A. The pseudocode for our algorithm appears in Algorithm 4.

Algorithm 4 The Cluster Aggregation algorithm

Require: An undirected graph G, partition $\mathcal{P}$, set S of portals.
Ensure: A cluster aggregation DEST

For each set X in $\mathcal{P}$, let p_X denote a shortest path from X to S, and let P_X denote the sequence of clusters visited in p_X.
For a cluster Y that appears in P_X, define the *position* of a cluster Y in P_X to be ℓ if the number of distinct clusters that P_X visits before first visiting Y is $\ell - 1$.
Construct an auxiliary directed graph D whose vertices are the clusters of $\mathcal{P}$. For vertices X and Y, D has an edge from X to Y if P_X contains Y; furthermore, we label the edge (X, Y) with the position of Y in P_X.
Set i to be 0 and V_0 to be the set of vertices in D.
repeat {Begin Phase i}
  Let D_i denote the subgraph of D induced by V_i. Let E_i denote the set of edges in D_i. Set V_{i+1} to $\emptyset$ and $\widehat{D}$ to D_i.
  repeat
    Let v be an arbitrary vertex in $\widehat{D}$.
    if $i = 0$ **then**
      Set $\text{DEST}(v)$ to be the vertex in S nearest to v;
    else
      Set $\text{DEST}(v)$ to be $\text{DEST}(x)$ where x is a vertex in $V_{i-1} - V_i$ and the label of (v, x) is the least among all edges from v to $V_{i-1} - V_i$.
    end if
    Let T denote $\{v\} \cup \text{out}_{\widehat{D}}(\{v\})$.
    repeat {iteration}
      For each u in $\widehat{D} - T$, and each edge (u, w) in $\widehat{D}$, remove (u, w) from $\widehat{D}$ if there exists an edge (u, x) in $\widehat{D}$ with $x \in T$ such that the label of (u, x) is smaller than the label of (u, w).
      For each u in $\text{in}_{\widehat{D}}(T) \cup \text{out}_{\widehat{D}}(T \cup \text{in}_{\widehat{D}}(T))$, set $\text{DEST}(u)$ to be equal to $\text{DEST}(v)$. Set T equal to $T \cup \text{in}_{\widehat{D}}(T) \cup \text{out}_{\widehat{D}}(T \cup \text{in}_{\widehat{D}}(T))$.
    until $|\text{in}_{\widehat{D}}(T)| < |T|$.
    Set V_{i+1} to $V_{i+1} \cup \text{in}_{\widehat{D}}(T)$ and remove $T \cup \text{in}_{\widehat{D}}(T)$ from $\widehat{D}$.
  until $\widehat{D}$ is empty
  Increment i {End Phase i}
until V_i is $\emptyset$

We now show that Algorithm 4 solves the Cluster Aggregation problem for a given partition $\mathcal{P}$ with a detour of $O(\log^2(|\mathcal{P}|)\text{MAXDIAM}(\mathcal{P}))$. We first establish the following simple lemma that bounds the number of phases.

Lemma 21. *If V_i and V_{i+1} are the set of vertices in D_i and D_{i+1} at the start of phase i and $i+1$, respectively, then $|V_{i+1}| \leq |V_i|/2$.*

Proof. We first note that $V_{i+1} \subseteq V_i$. Furthermore, in each iteration of the ith phase, when we add $\text{in}_{\widehat{D}}(T)$ to V_{i+1}, $|\text{in}_{\widehat{D}}(T)|$ is less than $|T|$, where T is a subset of $V_i - V_{i+1}$. Thus, $|V_i| - |V_{i+1}| \geq |V_{i+1}|$, yielding the desired claim. □

For each r_i in S, let $C(r_i)$ denote the union of the clusters X such that $\text{DEST}(X) = r_i$. Note that $C(r_i)$

may vary as the algorithm progresses.

Theorem 22. *The detour for any vertex v in G in the cluster merger returned by Algorithm 4 is at most* $\log^2(|\mathcal{P}|)\textsc{MaxDiam}(\mathcal{P})$.

Proof. Let m equal $|\mathcal{P}|$, the number of clusters in $\mathcal{P}$. Fix a portal r in S. We will show that at the end of iteration j of phase i, the following holds:

- For any Z in $\mathcal{P}$, if $\textsc{dest}(Z)$ equals r, then for each vertex v in Z, there is a path in $G[C(r)]$ from v to $\textsc{dest}(Z)$ of weight at most $2((i-1)\log(|\mathcal{P}|)+j)\textsc{MaxDiam}(\mathcal{P})$ more than $d(Z,S)$.

Before we establish the above claim, we show how the statement of the theorem follows. By Lemma 21, the number of phases is at most $\log m$. Furthermore, the number of iterations of the inner repeat loop in each phase is at most $\log m$ since the size of T at least doubles in each iteration. Therefore, at termination, the detour for each cluster in $\mathcal{P}$ is at most $2(\log^2 m)\textsc{MaxDiam}(\mathcal{P})$, yielding the desired claim.

Consider an iteration j of phase i. In the following, T and $\widehat{D}$ refer to the variables in the above algorithm at the start of the iteration. The set of clusters for which we set the $\textsc{dest}$ values in the iteration is given by $\text{in}_{\widehat{D}}(T) \cup \text{out}_{\widehat{D}}(T \cup \text{in}_{\widehat{D}}(T))$, where T corresponds to the value of the variable at the start of the iteration. Every cluster in T shares the same $\textsc{dest}$ value, say x. By the induction hypothesis, at the start of iteration j of phase i, each cluster Y in the set of clusters with $\textsc{dest}$ equal to x has a path q_Y in $G[C(x)]$ from Y to x of weight at most $2((i-1)\log m + (j-1))\textsc{MaxDiam}(\mathcal{P})$ more than $d(Y,S)$.

Consider a vertex Z in $\text{in}_{\widehat{D}}(T)$. Since Z is in $\text{in}_{\widehat{D}}(T)$, its path p_Z contains a cluster Y in T. Let p' denote the prefix of the path p_Z that connects Z to the first occurrence of Y in p_Z; and let p'' denote the remainder of the path p_Z. We note that every cluster that appears in p' is in $\text{out}_{\widehat{D}}(\{z\})$, and is, hence, also in $\text{out}_{\widehat{D}}(T \cup \text{in}_{\widehat{D}}(T))$. Thus, at the end of iteration j, p' is fully contained in $G[C(x)]$ the subgraph of G induced by the set of vertices with $\textsc{dest}$ equal to x. The weight of p_Z equals the sum of the weights of p' and p''. The weight of p_Y is at most the weight of p''. Thus, the path from Z to x consisting of p', followed by a shortest path to p_Y in Y, and followed by the path q_Y is entirely contained in $G[C(x)]$ and has weight at most $2((i-1)\log m + j)\textsc{MaxDiam}(\mathcal{P})$ more than the length of p_Z. (This is because the weight of any shortest path in Y is at most $\textsc{MaxDiam}(\mathcal{P})$.) This completes the induction step of the proof. □

6 Partition Hierarchy for Minor-free Graphs

A weighed graph G is H-minor free if zero or more edge contractions on G does not give a graph isomorphic H. Minor-free graphs are special cases of k-path separable graphs. A graph G is *k-path separable* [2] if there exists a subgraph S, called the *k-path separator*, such that: (i) $S = S_1 \cup S_2 \cup \cdots \cup S_l$, where for each $1 \leq i \leq l$, subgraph S_i is the union of k_i paths where each path is shortest in $G \setminus \bigcup_{1 \leq j < i} S_j$ with respect to its end points; (ii) $\sum_i k_i \leq k$; (iii) either $G \setminus S$ is empty, or each connected component of $G \setminus S$ is k-path separable and has at most $n/2$ nodes.

Thorup [31] shows that any planar graph G is 3-path separable, where all paths in the separator S belong in S_1, that is, they are shortest paths in G. Abraham and Gavoille [2] generalize the result to any H-minor free graph, for fixed size H, is k-path separable, for some $k = k(H)$, and the k-path separator can be computed in polynomial time. Interesting classes of H-minor free graphs are: planar graphs, which exclude K_5 and $K_{3,3}$; outerplanar graphs, which exclude K_4 and $K_{2,3}$; series-parallel graphs, which exclude K_4; and trees, which exclude K_3. They also show that the path separator can be computed in polynomial time.

6.1 The algorithm

Consider now an arbitrary weighted H-minor free graph G, for fixed size H. (You may also take G to be an arbitrary k-path separable graph.) We will construct a hierarchical (α, β, γ)-partition of G which is based

on forming clusters around the path separators of G. The concept of creating clusters around path separators has been introduced by Busch *et al.* [11] in the context of sparse covers in k-path separable graphs. Here, we extend that technique to hierarchical partitions.

We build the hierarchical partition bottom up by coarsening clusters. Suppose we are given a $(\alpha, \beta, \gamma^{i-1})$-partition $\mathcal{P}_{i-1}$. We describe how to build a $(\alpha, \beta, \gamma^i)$-partition $\mathcal{P}_i$, such that $\mathcal{P}_{i-1}$ is a refinement of $\mathcal{P}_i$.

The first clusters of partition $\mathcal{P}_i$ are formed around a k-path separator of G by appropriately merging clusters of $\mathcal{P}_{i-1}$ close to the separator paths. We then remove the k-path separator and repeat the clustering process recursively to each residual connected component, until no nodes are left in the graph.

Consider a connected component Φ which appears during the recursive decomposition of G. Let $S = S_1 \cup S_2 \cup \cdots \cup S_l$ be the path separator of Φ. We process the paths of S in sequence starting from the paths in S_1, then the paths in S_2, and so on. We maintain the new formed clusters in a set $\mathcal{N}$, which is updated every time we process a new path.

Consider now the path $p \in S_\chi$. Let Ψ be the connected component of $\Phi \setminus \bigcup_{1 \leq j < r} S_\chi$ in which p resides. Denote $\mathcal{P}_{i-1}^{\Psi} \subseteq \mathcal{P}_{i-1}$ the *integral* clusters of $\mathcal{P}_{i-1}$ which are completely contained in Ψ. We define the following subsets $\mathcal{A}$ and $\mathcal{B}$ of $\mathcal{P}_{i-1}^{\Psi}$ such that: $\mathcal{A}$ contains all clusters of $\mathcal{P}_{i-1}^{\Psi}$ not yet included in $\mathcal{N}$ within distance $2\gamma^i$ from p in Ψ; $\mathcal{B}$ contains all the clusters of $\mathcal{A}$ which are adjacent to clusters in $\mathcal{N}$ (where $\mathcal{N}$ are the clusters which have been formed so far form paths processed before p).

Let Ψ' be the sub-graph induced by $\mathcal{A}$ (note that Ψ' may not be connected). Combine the clusters in $\mathcal{A}$ by invoking the cluster aggregation algorithm of Section 5. to each connected component of Ψ'. We define two sets of nodes L and U in Ψ' which will serve as portals around which new merged clusters will be formed. Set L contains the *leaders* of path p, which is a maximal set of nodes in $p \cap \Psi'$, such for any pair $u, v \in L$, $d_p(u, v) \geq \gamma^i$, and u and v cannot belong to the same cluster of $\mathcal{A}$. Set U contains one arbitrary node from each cluster in $\mathcal{B}$. Combine the clusters in $\mathcal{A}$ by invoking the algorithm of Section 5. to each connected component of Ψ' for the induced clusters from $\mathcal{A}$ and the induced portal nodes in $L \cup U$.

Let $\mathcal{R}$ contain all resulting clusters from invoking Algorithm 4. We can write $\mathcal{R} = \mathcal{I}_p \cup \mathcal{K}_p$ where $\mathcal{I}_p$ consists of clusters that contain a node of L, and $\mathcal{K}_p$ consists of clusters that contain a node of U. Each cluster $X \in \mathcal{K}_p \setminus \mathcal{I}_p$ merges further with at most one arbitrary adjacent cluster $Y \in \mathcal{N}$, for which there is an edge $(u, v) \in E(\Psi)$ such that $u \in X$, $v \in Y$, and $v \notin \Psi'$. We insert the merged cluster from X and Y back to $\mathcal{N}$. The returned set of clusters from processing path p is $\mathcal{N} = \mathcal{N} \cup \mathcal{I}_p$.

The algorithm is initially invoked with $\Phi = G$ and $\mathcal{N} = \emptyset$. The resulting partition $\mathcal{P}_i$ is the final $\mathcal{N}$ that we obtain after we recursively process all the path separators in G.

6.2 The analysis

Consider a minor-free graph G with n nodes. The recursive process of removing path separators defines a decomposition tree T of G. Each node $t \in T$ corresponds to a connected component of G, which we will denote $G(t)$. The root π of T corresponds to G, namely, $G(\pi) = G$. Denote $S(t)$ the path separator for the respective graph $G(t)$. If $G(t) \setminus S(t) = \emptyset$, then t is a leaf of T. Otherwise, for each connected component $\Phi \in G(t) \setminus S(t)$ there is a node $w \in T$ such that w is a child of t and $G(w) = \Phi$.

Consider a node $t \in T$. Each path $p \in S(t)$ has a respective processing order in $S(t)$, denoted $order(p)$, which is an integer between 1 and k. The set of *previous* paths of p, denoted $Q(p)$, is defined to include those paths in $S(t)$ which have smaller order, or the paths in the ancestors of t:

$$Q(p) = \{q \in S(t) : order(q) < order(p)\} \cup \{q \in S(w) : w \text{ is ancestor of } t\}.$$

According to the algorithm, after a new cluster is created (when a path is processed) it may get larger when new clusters merge into it (when subsequent paths are processed). Once a cluster is created it can never shrink or be removed. Consider a path $p \in S(t)$, for some $t \in T$. We say that a *cluster belongs* to

Algorithm 5 Component clustering in minor-free graph

Require: Connected component Φ of minor-free graph G, strong $(\alpha, \beta, \gamma^{i-1})$-partition $\mathcal{P}_{i-1}$ of G, set $\mathcal{N}$ with coarsen clusters of $\mathcal{P}_{i-1}$.

Ensure: Coarsening the $\mathcal{P}_{i-1}$ clusters in Φ, which are then inserted in $\mathcal{N}$.

Let $S = S_1 \cup S_2 \cup \cdots \cup S_l$ be a k-path separator of Φ.
for χ from 1 to l **do**
  for each path $p \in S_\chi$ **do**
    Let Ψ be the connected component of $\Phi \setminus \bigcup_{1 \leq j < r} S_\chi$ in which p resides.
    Let $\mathcal{P}_{i-1}^{\Psi} = \{X \in \mathcal{P}_{i-1} : X \subseteq V(\Psi)\}$ be the *integral* clusters of $\mathcal{P}_{i-1}$ which are completely contained within Ψ;
    Let $\mathcal{A} = \{X \in \mathcal{P}_{i-1}^{\Psi} : (d_\Psi(X, p) \leq 2\gamma^i) \wedge (X \cap V(\mathcal{N}) = \emptyset)\}$ be the all integral clusters of Ψ which have not yet been coarsen (do not belong in $\mathcal{N}$) and are within distance $2\gamma^i$ from p in Ψ.
    Let $\mathcal{B} = \{X \in \mathcal{A} : \exists (u, v) \in E(\Psi), u \in X \wedge v \in V(\mathcal{N}) \cup (V(\Psi) \setminus V(\mathcal{P}_{i-1}^{\Psi})\}$, contains all the clusters of $\mathcal{A}$ which are adjacent to clusters in $\mathcal{N}$ or adjacent to non-integral clusters in Ψ.
    Let $\Psi' = \Psi \cap V(\mathcal{A})$ be the sub-graph of Ψ induced by $V(\mathcal{A})$ (note that Ψ' may not be connected).
    Let L be the *leaders* of path p, which is a maximal set of nodes in $p \cap \Psi'$, such for any pair $u, v \in L$, $d_p(u, v) \geq \gamma^i$, and u and v cannot belong to the same cluster of $\mathcal{A}$.
    Let U be the set that contains one arbitrary node from each cluster in $\mathcal{B}$.
    Combine the clusters in $\mathcal{A}$ by invoking Algorithm 4 to each connected component of Ψ' for the induced clusters from $\mathcal{A}$ and the induced portal nodes in $L \cup U$.
    Let $\mathcal{R}$ be the union of the resulting set of clusters from Algorithm 4.
    Write $\mathcal{R} = \mathcal{I}_p \cup \mathcal{K}_p$ where $\mathcal{I}_p$ consists of clusters that contain a node of L, and $\mathcal{K}_p$ consists of clusters that contain a node of U.
    for each cluster $X \in \mathcal{K}_p \setminus \mathcal{I}_p$ **do**
      X merges with at most one arbitrary adjacent cluster $Y \in \mathcal{N}$ such that there is an edge $(u, v) \in E(\Psi)$, $u \in X$, $v \in Y$, and $v \notin \Psi'$.
      We insert the merged cluster from X and Y back to $\mathcal{N}$.
    end for
    Update $\mathcal{N} = \mathcal{N} \cup \mathcal{I}_p$.
  end for
end for
for each connected component $\Upsilon \in \Psi \setminus S$ **do**
  Invoke (recursively) Algorithm 5 with parameters Υ, $\mathcal{P}_{i-1}$, and $\mathcal{N}$.
  Update $\mathcal{N}$ to be the result of the recursive invocation.
end for
Return $\mathcal{N}$.

p if it contains a leader of p. It is easy to verify that a cluster in $\mathcal{P}_i$ does not belong to more than one path (we will actually show in Lemma 28 that each cluster in $\mathcal{P}_i$ belongs to exactly one path). Let $\mathcal{I}_p$ denote the clusters that belong to p immediately after p is processed. Let $\widehat{\mathcal{I}}_p$ denote the final clusters of p in $\mathcal{P}_i$.

In the analysis below, assume that $\gamma \geq \alpha$, and define $\alpha' = c_1 \lambda k \log n$, for a constant c_1. The parameter λ denotes the impact of the detour of Algorithm 4, on the ratio of the cluster diameter before and after the merger merger. From Theorem 22, $\lambda = O(\log^2 n)$.

Lemma 23. *In Ψ' every cluster of $\mathcal{A}$ is within distance at most $3\gamma^i$ to a node in $L \cup U$.*

Proof. Consider a cluster $X \in \mathcal{A}$. Let $u \in X$ be the closest to a node $v \in p$ in graph Ψ. From definition of $\mathcal{A}$, $d_\Psi(u, v) \leq 2\gamma^i$. Let q be a shortest path in Ψ connecting u to v.

If q uses a cluster outside A, then that cluster must be either a cluster in $\mathcal{N}$ or a non-integral cluster of $\mathcal{P}_{i-1}$. Therefore, q has to cross a cluster in $\mathcal{B}$. Let $\ell \in V(\mathcal{B}) \cap U$. Since $\alpha\gamma^{i-1} \leq \gamma^i$, $d_{\Psi'}(u,\ell) \leq 2\gamma^i + \alpha\gamma^{i-1} \leq 3\gamma^i$.

Consider now the case where q uses only clusters in A. Let p' be the subpath of p which consists of the nodes within distance γ^i from u, with respect to Ψ.

Suppose that p' uses only clusters in $\mathcal{A}$. Suppose, for the sake of contradiction, that none of the nodes in p' is a leader in L. Let $Y \in \mathcal{A}$ be the cluster that contains v. We have that the closest leader to u (if it exists), must be at distance greater than γ^i from v. Since the diameter of Y is at most $\alpha\gamma^{i-1} \leq \gamma^i$, then L is not maximal because v is a valid possible leader. Therefore, p' must contain a leader $\ell \in L$. Thus, $d_{\Psi'}(u,\ell) \leq 2\gamma^i + \gamma^i \leq 3\gamma^i$.

If p' doesn't use a cluster in $\mathcal{A}$, then it has to use a cluster in $\mathcal{B}$. By selecting a node $\ell \in V(\mathcal{B}) \cap U$, we get $d_{\Psi'}(u,\ell) \leq 2\gamma^i + \alpha\gamma^{i-1} \leq 3\gamma^i$. □

Lemma 24. *Every cluster of $\widehat{\mathcal{I}}_p$ has diameter at most $\alpha'\gamma^i$.*

Proof. From Lemma 23, each cluster in $\mathcal{A}$ is within distance $\sigma = 3\gamma^i$ from a node in $L \cup U$. Since, the diameter of each cluster in $\mathcal{A}$ is bounded by $\alpha\gamma^{i-1} \leq \gamma^i < \sigma$, Algorithm 4 produces new clusters around the nodes in $L \cup U$, so that each new cluster has diameter at most $\lambda\sigma \leq 3\lambda\gamma^i$. Thus the cluster in $\mathcal{I}_p$ have diameter most $\zeta = 3\lambda\gamma^i$. Similarly, the clusters in $\mathcal{K}_p$ have also diameter at most ζ.

The clusters in $\mathcal{I}(p)$ may increase in diameter, when they merge with $\mathcal{K}_q$ clusters from some path q processed after p. This path q may belong to $S(t)$, or it may belong to $S(w)$, where w is a descendant in the sub-tree T' rooted in t.

Each path $q \in S(t)$ with order after p, increases the diameter of $\mathcal{I}_p$ by at most 2ζ, since newly merged clusters from $\mathcal{K}_q$ add at most one layer of clusters into $\mathcal{I}_p$, and any two clusters in the layer can reach each other through the previous instance of $\mathcal{I}_p$. Thus, when we process the last path in $S(t)$, we have added at most k layers, and the increase in the diameter of the new $\mathcal{I}_p$ will be at most $2\zeta k$.

Similarly, any node in the sub-tree T', contributes at most k new layers to $\mathcal{I}_p$. However, all the nodes of T' in the same level contribute in total k layers, since clusters in them are formed independent of each other. Since the sub-tree T has at most $1+\log n$ levels (including t), we have in total $k(1+\log n)$ additional layers in $\mathcal{I}_p$, contributing increase at most $2\zeta k(1+\log n)$ to the diameter of $\mathcal{I}(p)$. Therefore, the diameter of $\widehat{\mathcal{I}}(p)$ is at most $2\zeta k(1+\log n) + \zeta \leq c_1\lambda k\gamma^i \log n$, for some constant c_1. □

For a path $q \in Q(p)$, let $\mathcal{I}'_q$ be the clusters of q just before processing path p, and $\mathcal{I}''_q$ be the clusters of a path q just after processing path p. Let $\mathcal{Z}'(p) = \bigcup_{q\in Q(p)} \mathcal{I}'_q$ and $\mathcal{Z}''(p) = \bigcup_{q\in Q(p)} \mathcal{I}'_q$. Define $\mathcal{Z}(p) = \mathcal{I}_p \cup \mathcal{Z}''(p)$. For any set of nodes Y let $\Gamma(Y)$ denote the set of clusters in $\mathcal{P}_{i-1}$ which are within distance $2\gamma^i$ from p, namely, $\Gamma(Y) = \{X \in \mathcal{P}_{i-1} : d_G(X,Y) \leq 2\gamma^i\}$.

Lemma 25. $\Gamma(p) \subseteq \mathcal{Z}(p)$.

Proof. We prove the claim by induction on $|Q(p)|$. For the basis case, $|Q(p)| = 0$, path p is the first to be processed by the algorithm with $Q(p) = \emptyset$. Therefore, $\Gamma(p) = \mathcal{A}(p) = \mathcal{I}_p = \mathcal{Z}(p)$.

Assume now that the claim holds for $|Q(p)| \leq \sigma$. Consider now the case $|Q(p)| = \sigma + 1$. From induction hypothesis, for each path $q \in Q(p)$, $\Gamma(q) \subseteq \mathcal{Z}(q)$. Let $\mathcal{N}$ be the new formed clusters of the algorithm just before we process p. Since $\mathcal{Z}(q) \subseteq \mathcal{N}$, we have that $\Gamma(q) \subseteq \mathcal{N}$.

First, we show that just before we process path p the cluster of $\mathcal{N}$ that intersect Ψ can only be those in $\mathcal{Z}'(p) \cap \mathcal{N}$. Suppose, for the sake of contradiction, that there is a cluster $X \in \mathcal{N} \setminus \mathcal{Z}'(p)$ which intersects Ψ. Cluster X must contain a node $y \notin V(\Psi)$, since any integral cluster in Ψ can only have been built by a path in $Q(p) \cap S_\chi$, where $p \in S_\chi$. Take a node $u \in X \cap V(\Psi)$. Any path from u to y must cross one of the paths in $Q(p)$ whose removal from G contributed to the formation of Ψ. However, from induction hypothesis all

the nodes in the paths in $Q(p)$ belong to clusters in $\mathcal{Z}'(p)$. Consequently, y cannot exist, and hence neither does X.

Next, we show that any non-integral cluster $Y \in \mathcal{P}_{i-1}$, $Y \notin \mathcal{P}_{i-1}^{\Psi}$, which intersects Ψ is used in a cluster of $\mathcal{Z}'(p)$. Note that Y must have be crossed by at least a path $q \in Q(p)$ whose removal from G contributed to the creation of Ψ. Since the diameter of Y is bounded by $\alpha\gamma^{i-1} \leq \gamma^i$, we have that $Y \in \Gamma(q) \subseteq \mathcal{Z}(q)$. Therefore, $Y \in \mathcal{Z}'(p)$.

We continue now with the main claim. Let $X \in \Gamma(p)$. There are the following possibilities:

- $X \in \mathcal{P}_{i-1}^{\Psi}$: we examine the following sub-cases.
  - $X \in \mathcal{N}$: Since before processing p only clusters in $\mathcal{Z}'(p)$ intersect Ψ, $X \in \mathcal{Z}'(p)$. Therefore, after processing p, X will remain in the same cluster as in $\mathcal{Z}'(p)$. Thus, $X \in \mathcal{Z}(p)$.
  - $X \in \mathcal{A}$: from the algorithm, after processing p there are two possibilities. First possibility is $X \in \mathcal{I}_p$ and hence, $X \in \mathcal{Z}(p)$. Second possibility is $X \in \mathcal{K}_p \setminus \mathcal{I}_p$ and X is either (i) adjacent to some node in $\mathcal{N}$, or (ii) adjacent to some non integral cluster in Ψ. In case (i) X merges with a cluster in $\mathcal{N}$, and since only clusters of $\mathcal{Z}'(p)$ can be in Ψ, we immediately have $X \in \mathcal{Z}(p)$. In case (ii), as we have shown above any non-integral cluster is a member of $\mathcal{Z}'(p) \subseteq \mathcal{N}$, and thus X merges with a cluster of $\mathcal{Z}'(p)$, which implies that $X \in \mathcal{Z}(p)$.
- $X \notin \mathcal{P}_{i-1}^{\Psi}$: Then, X must contain a node $u \notin \Psi$. If X intersects Ψ, then we have shown above that $X \in \mathcal{Z}'(p)$, and thus $X \in \mathcal{Z}(p)$. If X does not intersect Ψ, any path from p to X must intersect a path $q \in Q(p)$, since otherwise X wouldn't reside in a different component. Since $d_G(p, X) \leq 2\gamma^i$, we have that $d_G(q, X) \leq 2\gamma^i$. Therefore, $X \in \Gamma(q) \subseteq \mathcal{Z}(q)$. Consequently, $X \in \mathcal{Z}(p)$.

□

Lemma 26. *Any ball of radius γ^i in G intersects with at most $2\alpha' + 3$ clusters of $\widehat{\mathcal{I}}_p$.*

Proof. We start by showing that we only need to consider balls of radius γ^i in Ψ. Let $G' = G \setminus \Psi$. Let Y denote the set of nodes in G' such that each $x \in Y$ is adjacent to a node in Ψ. It must be that $Y \subseteq V(Q(p))$, where $V(Q(p))$ denotes the nodes of all the paths in $Q(p)$. Let $\mathcal{F}$ be all the clusters in $\mathcal{P}_{i-1}^{\Psi}$ which are at distance at most $2\gamma^i$ from Y, namely, $\mathcal{F} = \{X \in \mathcal{P}_{i-1}^{\Psi} : d_{\Psi}(X, Y) \leq 2\gamma^i\}$. Clearly, $\mathcal{F} = \mathcal{P}_{i-1}^{\Psi} \cap \Gamma(Y)$.

From Lemma 25, each cluster in $\Gamma(Y)$ has been used in the clusters of some path of $Q(p)$ that goes through Y. Therefore, the clusters in $\mathcal{F}$ are all used in clusters of paths in $Q(p)$. Consequently, the clusters of p, $\mathcal{I}_p$, cannot possibly belong in $\mathcal{F}$, namely $\mathcal{I}_p \cap \mathcal{F} = \emptyset$. When we further process paths in Ψ in node t (paths ordered after p in $S(t)$), and then descendants of p, we have that each of the clusters in I_p grows, however they will never intersect $\mathcal{F}$. Thus, $\widehat{\mathcal{I}}_p \cap \mathcal{F} = \emptyset$.

Consequently, any cluster of $\widehat{\mathcal{I}}_p$ is at distance at least $2\gamma^i$ from G'. Therefore, any ball of radius γ^i that intersects clusters of $\widehat{\mathcal{I}}_p$ has to be a sub-graph of Ψ. Thus, in order to prove the main claim, we only need to focus on graph Ψ.

Consider a ball $B = B(u, \gamma^i)$ within Ψ. Suppose that $\xi \geq 2$ clusters of $\widehat{\mathcal{I}}_p$ intersect p. Path p is a shortest path in Ψ. Each cluster in $\widehat{\mathcal{I}}_p$ has a distinct leader in p. The leaders are at distance at least γ^i apart in p. Therefore, there are two clusters intersecting B, whose respective leaders, ℓ_1 and ℓ_2, are at distance at least $d_{\Psi}(\ell_1, \ell_2) \geq (\xi - 1)\gamma^i$. Ball B provides an alternative path between ℓ_1 and ℓ_2 through u, with total length is bounded by $d_{\Psi}(\ell_1, \ell_2) \leq d_{\Psi}(\ell_1, u) + d_{\Psi}(u, \ell_2)$. Since the cluster of ℓ_1 intersects B, we obtain from Lemma 24 that $d_{\Psi}(\ell_1, u) \leq \alpha'\gamma^i + \gamma^i = (\alpha' + 1)\gamma^i$. Similarly, $d_{\Psi}(u, \ell_2) \leq (\alpha' + 1)\gamma^i$. Therefore, $d_{\Psi}(\ell_1, \ell_2) \leq 2(\alpha' + 1)\gamma^i$. Therefore, it has to be $\xi - 1 \leq 2(\alpha' + 1)$, or equivalently, $\xi \leq 2\alpha' + 3$. □

Lemma 27. *Any ball of radius γ^i in G intersects with at most $c_2\alpha' k \log n$ clusters of $\mathcal{P}_i$, for a constant c_2.*

Proof. Consider a node $v \in G$ and the ball $B = B(v, \gamma^i)$. Each node $v \in G$ belongs to a path $p \in S(w)$, of some path separator $S(w)$, $w \in T$, in the recursive decomposition of G. Clearly, $B(v, \gamma^i) \subseteq \Gamma(p)$. From Lemma 25, we have that $B \subseteq \mathcal{Z}(p)$. Since $\mathcal{Z}(p)$ consists only of clusters that belong to $Q' = p \cup Q(p)$. All the paths in Q' appear in path separators of T between the root and w. Since the depth of T is at most $1 + \log n$, the total number of path separators involved in Q' is at most $1 + \log n$, each contributing k paths. Therefore, $|Q'| \leq k(1 + \log n)$.

From Lemma 26, B intersects with at most $(2\alpha' + 3)$ clusters of each path $q \in Q'$. Thus, the total number of clusters of $\mathcal{P}_i$ intersecting B is at most $(2\alpha' + 3)k(1 + \log n) \leq c_2\alpha' k \log n$, for a constant c_2, as needed. □

Lemma 28. *$\mathcal{P}_i$ is a $(\alpha', c_2\alpha' k \log n, \gamma^i)$-partition.*

Proof. Every node in G belongs to a path in some path separator used by the algorithm. From Lemma 25, each node in a path p must be a member of some cluster which either belongs to p or to a path $q \in Q(p)$. Consequently, each node $v \in G$ will appear in some cluster of $\widehat{\mathcal{I}}_q$ of some path q. Therefore, $\mathcal{P}_i$ is a partition of G.

From Lemmas 24, the diameter of any $\widehat{\mathcal{I}}_q$ is bounded by $\alpha'\gamma^i$. Therefore, the diameter of each cluster in $\mathcal{P}_i$ is at most $\alpha'\gamma^i$. From Lemma 27, each ball of radius γ^i intersects at most $c_2\alpha' k \log n$ clusters of $\mathcal{P}_i$. □

Theorem 29. *We can obtain a hierarchical $(O(\log^3 n), O(\log^4 n), \Theta(\log^3 n))$-partition of minor-free graph G in polynomial time.*

Proof. From Lemma 28, since $k = O(1)$, we can build a hierarchy of clusters by choosing $\alpha = \alpha' = O(\log^3 n)$. Further, for each level i, we can create the necessary padding around a root node $r \in G$ of radius γ^i, by creating a cluster that contains the ball $B(r, \gamma^i)$. We can do this by using either of two methods. In the first method, we can explicitly add r to the first separator in G, as an artificial path (with one node) that needs to be processed first. This causes the size of the first separator to be of size $k + 1$, and in the analysis we replace k with $k + 1$. In the second method, we can merge all the clusters in $B(r, \gamma^i)$ created by the algorithm, giving a new cluster whose diameter is no more than three times the diameter of the old cluster. Either way, the impact to the parameters of the clustering is a constant factor, giving the desired hierarchical partition. It is easy to verify that all the steps of the algorithm can be performed in polynomial time with respect to the size of G and the parameters of the problem. □

Corollary 30. *A polylog(n)-stretch universal Steiner tree can be computed in polynomial time for any minor-free graph.*

References

[1] I. Abraham, Y. Bartal, and O. Neiman. Advances in metric embedding theory. In *Proceedings of ACM STOC*, 2006.

[2] I. Abraham and C. Gavoille. Object location using path separators. In *Proceedings of ACM PODC*, pages 188–197, 2006.

[3] I. Abraham, C. Gavoille, D. Malkhi, and U. Wieder. Strong-diameter decompositions of minor free graphs. In *Proceedings of ACM SPAA*, 2007.

[4] N. Alon, R. M. Karp, D. Peleg, and D. West. A graphtheoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995.

[5] B. Awerbuch and D. Peleg. Sparse partitions. In *Proceedings of IEEE FOCS*, pages 503–513, 1990.

[6] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of IEEE FOCS*, pages 184–193, 1996.

[7] Y. Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of ACM STOC*, pages 161–168, 1998.

[8] D. Bertsimas and M. Grigni. On the space-filling curve heuristic for the euclidean traveling salesman problem. *Operations Research Letters*, 8:241–244, 1989.

[9] A. Bhalgat, D. Chakrabarty, and S. Khanna. Optimal lower bounds for universal and differentially private steiner trees and tsps. In *Proceedings of APPROX*, pages 75–86, 2011.

[10] T. Birk, I. Keidar, L. Liss, A. Schuster, and R. Wolff. Veracity radius - capturing the locality of distributed computations. In *Proceedings of ACM SIGACT-SIGOPS PODC*, 2006.

[11] C. Busch, R. LaFortune, and S. Tirthapura. Improved sparse covers for graphs excluding a fixed minor. In *Proceedings of ACM PODC*, pages 61–70, 2007.

[12] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S.A. Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proceedings of IEEE FOCS*, pages 379–388, 1998.

[13] M. Elkin, Y. Emek, D. Spielman, and S. Teng. Lower-stretch spanning trees. *special issue of SIAM Journal on Computing for STOC'05*, 38(2):608–628, 2008.

[14] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of ACM STOC*, pages 448–455, 2003.

[15] A. Goel and D. Estrin. Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. In *Proceedings of ACM-SIAM SODA*, pages 499–505, 2003.

[16] A. Goel and I. Post. An oblivious o(1)-approximation for single source buy-at-bulk. In *Proceedings of IEEE FOCS*, pages 442–450, 2009.

[17] A. Goel and I. Post. One tree suffices: A simultaneous o(1)-approximation for single-sink buy-at-bulk. In *Proceedings of IEEE FOCS*, pages 593–600, 2010.

[18] I. Gorodezky, R. D. Kleinberg, D. B. Shmoys, and G. Spencer. Improved lower bounds for the universal and a priori tsp. In *Proceedings of APPROX-RANDOM*, pages 178–191, 2010.

[19] A. Gupta, M. T. Hajiaghayi, and H. Räcke. Oblivious network design. In *Proceedings of ACM-SIAM SODA*, pages 970–979, 2006.

[20] M. T. Hajiaghayi, R. D. Kleinberg, and F. T. Leighton. Improved lower and upper bounds for universal tsp in planar metrics. In *Proceedings of ACM-SIAM SODA*, pages 649–658, 2006.

[21] L. Jia, G. Lin, G. Noubir, R. Rajaraman, and R. Sundaram. Universal approximations for tsp, steiner tree, and set cover. In *Proceedings of ACM STOC*, pages 386–39, 2005.

[22] P. Klein, S. A. Plotkin, and S. Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of ACM STOC*, 1993.

[23] G. Konjevod, R. Ravi, and F. Salman. On approximating planar metrics by tree metrics. *Information Processing Letters*, 80(4):213–219, 2001.

[24] B. Krishnamachari, D. Estrin, and S. Wicker. Modelling data-centric routing in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, 2002.

[25] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad hoc sensor networks. In *OSDI*, 2002.

[26] S. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Proceedings of IEEE WMCSA*, 2002.

[27] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

[28] L. K. Platzman and III J. J. Bartholdi. Spacefilling curves and the planar travelling salesman problem. *Journal of the ACM*, 36(4):719–737, 1989.

[29] H. Räcke. Minimizing congestion in general networks. In *Proceedings of IEEE FOCS*, page 4352, 2002.

[30] S. Srinivasagopalan, C. Busch, and S.S. Iyengar. An oblivious spanning tree for single-sink buy-at-bulk in low doubling-dimension graphs. *IEEE Transactions on Computers*, 99, 2011.

[31] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of ACM*, 51(6):993–1024, 2004.