Ramsey Spanning Trees and Their Applications

ITTAI ABRAHAM, VMWare
SHIRI CHECHIK, Tel-Aviv University
MICHAEL ELKIN, ARNOLD FILTSER, and OFER NEIMAN, Ben-Gurion University of the Negev

The *metric Ramsey problem* asks for the largest subset S of a metric space that can be embedded into an ultrametric (more generally into a Hilbert space) with a given distortion. Study of this problem was motivated as a non-linear version of Dvoretzky theorem. Mendel and Naor [29] devised the so-called Ramsey Partitions to address this problem, and showed the algorithmic applications of their techniques to approximate distance oracles and ranking problems.

In this article, we study the natural extension of the metric Ramsey problem to graphs, and introduce the notion of *Ramsey Spanning Trees*. We ask for the largest subset $S \subseteq V$ of a given graph $G = (V, E)$, such that there exists a spanning tree of G that has small stretch for S. Applied iteratively, this provides a small collection of spanning trees, such that each vertex has a tree providing low stretch paths to *all other vertices*. The union of these trees serves as a special type of spanner, a *tree-padding spanner*. We use this spanner to devise the first compact stateless routing scheme with $O(1)$ routing decision time, and labels that are much shorter than in all currently existing schemes.

We first revisit the metric Ramsey problem and provide a new deterministic construction. We prove that for every k, any n-point metric space has a subset S of size at least $n^{1-1/k}$ that embeds into an ultrametric with distortion $8k$. We use this result to obtain the state-of-the-art deterministic construction of a distance oracle. Building on this result, we prove that for every k, any n-vertex graph $G = (V, E)$ has a subset S of size at least $n^{1-1/k}$, and a spanning tree of G, that has stretch $O(k \log \log n)$ between any point in S and any point in V.

CCS Concepts: • **Theory of computation → Graph algorithms analysis**;

Additional Key Words and Phrases: Distortion, metric embedding, spanning trees, compact routing, distance oracles

ACM Reference format:
Ittai Abraham, Shiri Chechik, Michael Elkin, Arnold Filtser, and Ofer Neiman. 2020. Ramsey Spanning Trees and Their Applications. *ACM Trans. Algorithms* 16, 2, Article 19 (March 2020), 21 pages.
https://doi.org/10.1145/3371039

19

S. Chechik supported by the ISF grant No. 1528/15 and the Blavatnik Fund. M. Elkin supported by the ISF grant No. 724/15. A. Filtser and O. Neiman supported in part by ISF grant 1817/17, and by BSF Grant 2015813.
Authors' addresses: I. Abraham, Ampa Building, 5 Sapir St., Ground Floor, POB 12093, Herzliya 4685209, Israel; email: iabraham@vmware.com; S. Chechik, Department of Computer Science, Tel-Aviv University, Tel Aviv 69978, Israel; email: schechik@post.tau.ac.il; M. Elkin, A. Filtser, and O. Neiman, Department of Computer Science, Ben-Gurion University, Beer Sheva, Israel; emails: {elkinm, arnoldf, neimano}@cs.bgu.ac.il.

1549-6325/2020/03-ART19 $15.00
https://doi.org/10.1145/3371039

1 INTRODUCTION

Inspired by the algorithmic success of Ramsey Type Theorems for metric spaces, in this article, we study an analogue Ramsey Type Theorem in a graph setting. The classical Ramsey problem for metric spaces was introduced in Ref. [12] and is concerned with finding "nice" structures in arbitrary metric spaces. Following Ref. [14], Ref. [29] showed that every n-point metric (X, d) has a subset $M \subseteq X$ of size at least $n^{1-1/k}$ that embeds into an ultrametric (and thus also into Hilbert space) with distortion at most $O(k)$, for a parameter $k \geq 1$. In fact, they construct an ultrametric on X that has $O(k)$ distortion for any pair in $M \times X$. Additionally, Ref. [29] demonstrated the applicability of their techniques, which they denoted Ramsey Partitions, to approximate distance oracles and ranking problems.

We introduce a new notion that we call *Ramsey Spanning Trees*, which is a natural extension of the metric Ramsey problem to graphs. In this problem, we wish to find a large subset $S \subseteq V$, and a *spanning tree* of G that has small distortion for $S \times V$. Let $\mathbf{dist}(u, v, G)$ denote the shortest path distance in the graph G between the vertices $u, v \in V$, then our main result is the following.

Theorem 1. *Let $G = (V, E)$ be a weighted graph on n vertices, and fix a parameter $k \geq 1$. There is a polynomial time deterministic algorithm that finds a spanning tree T of G and a subset $S \subseteq V$ of vertices of size at least $n^{1-1/k}$, such that for every $v \in S$ and $u \in V$, it holds that $\mathbf{dist}(v, u, T) \leq O(k \log\log n) \cdot \mathbf{dist}(v, u, G)$.*

We remark that the extra factor of $\log\log n$ in the stretch comes from the state-of-the-art result of $O(\log n \log\log n)$ for low stretch spanning trees [5]. It is quite plausible that if that result is improved to the optimal $O(\log n)$, then the stretch in our result would be only $O(k)$. By applying Theorem 1 iteratively, we can obtain a small collection of trees so that each vertex has small stretch to all other vertices in at least one of the trees.

Theorem 2. *Let $G = (V, E)$ be a weighted graph on n vertices, and fix a parameter $k \geq 1$. There is a polynomial time deterministic algorithm that finds a collection $\mathcal{T}$ of $k \cdot n^{1/k}$ spanning trees of G, and a mapping $\mathbf{home} : V \to \mathcal{T}$, such that for every $u, v \in V$, it holds that $\mathbf{dist}(v, u, \mathbf{home}(v)) \leq O(k \log\log n) \cdot \mathbf{dist}(v, u, G)$.*

A spanner H with stretch t for a graph G, is a sparse spanning subgraph satisfying $\mathbf{dist}(v, u, H) \leq t \cdot \mathbf{dist}(v, u, G)$. Spanners are a fundamental metric and graph-theoretic constructions; they are very well-studied [1, 3, 16, 21, 24, 31, 36] and have numerous applications [2, 7, 21, 23, 27]. Theorem 2 can be viewed as providing a spanner, which is the union of $k \cdot n^{1/k}$ spanning trees, such that every vertex has a tree with low stretch paths to *all other vertices*. We call such a spanner a *tree-padding spanner* of *order* $k \cdot n^{1/k}$. To the best of our knowledge, no previous construction of spanners can be viewed as a tree-padding spanner of order $o(n)$. Until now, even the following weaker question was open: does there exist a spanner that is a union of a sublinear in n number of trees, such that every pair of vertices has a low stretch path in one of these trees.

Having a single tree that provides good stretch for any pair containing the vertex v, suggests that routing messages to or from v could be done on this one tree. Our main application of Ramsey spanning trees is a compact routing scheme that has constant routing decision time and improved label size; see Section 1.1 for more details.

Deterministic Ramsey Partitions. As a first step toward our main result, which is of interest in its own right, we provide a new deterministic Ramsey ultrametric construction. In particular, we show a polynomial time deterministic algorithm, which, given an n-point metric space (X, d) and a parameter $k \geq 1$, finds a set $M \subseteq X$ of size at least $n^{1-1/k}$ and an ultrametric (M, ρ) with distortion at most $8k-2$. That is, for each $v, u \in M$,

$$d(v,u) \le \rho(v,u) \le (8k-2) \cdot d(v,u).$$

The first result of this flavor was by Bartal et al. [10], who showed a deterministic construction with distortion $O(k \log\log n)$. This was improved by Ref. [14] to distortion $O(k \log k)$. Mendel and Naor [29] developed the so-called Ramsey partitions, which had distortion of $128k$ (using a randomized algorithm). Blelloch et. al [13] showed that the (randomized) algorithm of Ref. [26] constructs an ultrametric with distortion $18.5k$ (they also provided a near-linear time implementation of it). The best randomized algorithm is by Naor and Tao [30], who obtained distortion $2ek$. Their techniques are not based on Ramsey partitions, as is ours and other previous works (see Section 3 for more details on Ramsey partitions). In fact, Ref. [30] declared that a Ramsey partition with distortion better than $16k-2$ seems not to be possible with their current techniques. Moreover, Ref. [29] mention as a drawback that their solution is randomized (while Ref. [14] is deterministic). A deterministic construction similar to ours was obtained by Bartal [9].

An application of our improved deterministic Ramsey ultrametric construction is a new distance oracle that has the best space-stretch-query time tradeoff among deterministic distance oracles. See Section 1.1 below.

Techniques. Our construction of Ramsey ultrametrics uses the by-now-standard *deterministic* ball growing approach, e.g., see Refs [4], [6]–[8], and [26]. In this article, we provide tighter and more parameterized analysis of these multi-scale deterministic region growing techniques. Our improved analysis of the deterministic ball growing technique of Refs [8] and [26] obtains a similar type of improvement as the one obtained by the analysis of Mendel and Naor [29] on the randomized partition technique of Refs [20] and [25].

Our construction of Ramsey spanning trees is based on combining ideas from our Ramsey ultrametric construction, with the Petal Decomposition framework of Ref. [5]. The optimal multi-scale partitions of Refs [8] and [26] cannot be used in this petal decomposition framework, so we must revert to partitions based on Refs [22] and [33], which induce an additional factor of $O(\log\log n)$ to the stretch. In addition, the refined properties required by the Ramsey partition make it very sensitive to constant factors (these constants can be ignored in the Ref. [22] analysis of the average stretch, say). In order to alleviate this issue, we consider two possible region growing schemes, and choose between them according to the densities of points that can still be included in M. One of these schemes is a standard one, while the other grows the region "backward," in a sense that it charges the remaining graph, rather than the cluster being created, for the cost of making a cut. See Section 4.3 for more details.

1.1 Applications

Distance Oracles. A distance oracle is a succinct data structure that (approximately) answers distance queries. A landmark result of Ref. [34] states that any metric (or graph) with n points has a distance oracle of size $O(k \cdot n^{1+1/k})$,[1] which can report any distance in $O(k)$ time with stretch at most $2k-1$. A deterministic variant with the same parameters was given by Ref. [32], and this was the state-of-the-art for deterministic constructions. The oracle of Ref. [29] has improved size $O(n^{1+1/k})$ and $O(1)$ query time, but larger stretch $128k$. This oracle was key for subsequent improvements by Refs [18], [19], and [37] the latter gave a randomized construction of an oracle with size $O(n^{1+1/k})$, query time $O(1)$ and stretch $2k-1$ (which is asymptotically optimal assuming Erdos' girth conjecture).

[1] We measure size in machine words; each word is $\Theta(\log n)$ bits.

Table 1. Different Distance Oracles

Distance Oracle	Stretch	Size	Query time	Is deterministic?
Ref. [34]	$2k-1$	$O(k \cdot n^{1+1/k})$	$O(k)$	no
Ref. [29]	$128k$	$O(n^{1+1/k})$	$O(1)$	no
Ref. [37]	$(2+\epsilon)k$	$O(k \cdot n^{1+1/k})$	$O(1/\epsilon)$	no
Ref. [18]	$2k-1$	$O(k \cdot n^{1+1/k})$	$O(1)$	no
Ref. [19]	$2k-1$	$O(n^{1+1/k})$	$O(1)$	no
Ref. [32]	$2k-1$	$O(k \cdot n^{1+1/k})$	$O(k)$	yes
Ref. [37]	$2k-1$	$O(k \cdot n^{1+1/k})$	$O(\log k)$	yes
This article	$8(1+\epsilon)k$	$O(n^{1+1/k})$	$O(1/\epsilon)$	yes
This article	$2k-1$	$O(k \cdot n^{1+1/k})$	$O(1)$	yes

Similarly to Ref. [29], our deterministic construction of Ramsey ultrametrics can provide a deterministic construction of an approximate distance oracle. While the stretch of the oracle of Ref. [29] is further increased by a large constant factor over the stretch of the Ramsey ultrametric, we use a careful analysis and a collection of oracles in various distance scales, in order to increase the stretch by only a $1+\epsilon$ factor.

Theorem 3. *For any metric space on n points, and any $k > 1$, $0 < \epsilon < 1$, there is an efficient deterministic construction of a distance oracle of size $O(n^{1+1/k})$, which has stretch $8(1+\epsilon)k$ and query time $O(1/\epsilon)$.*

This is the first deterministic construction of an approximate distance oracle with constant query time and small size $O(n^{1+1/k})$.

Moreover, our oracle is an essential ingredient toward de-randomizing the recent distance oracles improvements [18, 19, 37]. Specifically, if we construct Ref. [18] by replacing the distance oracle of Mendel and Naor [29] by our deterministic version, and replacing the distance oracle of Thorup and Zwick [34] by the deterministic version of Roditty, Thorup, and Zwick [32], we immediately get a deterministic distance oracle of $O(k \cdot n^{1+1/k})$ size, $2k-1$ stretch, and $O(1)$ query time. This is a strict improvement over Ref. [32]. In addition, our oracle can be viewed as a first step toward de-randomizing the Ref. [19] oracle. A summary of all the previous and current results in a table form can be found at Table 1.

Stateless Routing with Short Labels and Constant Decision Time. A routing scheme in a network is a mechanism that allows packets to be delivered from any node to any other node. The network is represented as a weighted undirected graph, and each node can forward incoming data by using local information stored at the node, often called a routing table, and the (short) packet's header. The routing scheme has two main phases: in the preprocessing phase, each node is assigned a routing table and a short label. In the routing phase, each node receiving a packet should make a local decision, based on its own routing table and the packet's header (which may contain the label of the destination, or a part of it), where to send the packet. The *routing decision time* is the time required for a node to make this local decision. The *stretch* of a routing scheme is the worst ratio between the length of a path on which a packet is routed, to the shortest possible path. A routing scheme is called *stateless* if the routing decision does not depend on the path traversed so far.

The classical routing scheme of Ref. [35], for a graph on n vertices and integer parameters $k, b > 1$, provides a scheme with routing tables of size $O(k \cdot b \cdot n^{1/k})$, labels of size $(1+o(1))k \log_b n$, stretch $4k-5$, and decision time $O(1)$ (but the initial decision time is $O(k)$). The stretch was improved recently to roughly $3.68k$ by Ref. [17], using a similar scheme as Ref. [35]. With Theorem 2, we devise a stateless compact routing scheme with very short labels, of size only $(1+o(1)) \log_b n$, and with *constant* decision time, while the stretch increases to $O(k \log\log n)$ (and with the same table size as Ref. [35]).

We wish to point out that our construction of a routing scheme is simpler in some sense than those of Refs [17] and [35]. In both constructions, there is a collection of trees built in the preprocessing phase, such that every pair of vertices has a tree that guarantees small stretch. Routing is then done in that tree. In our construction, there are few trees, so every vertex can store information about all of them; in addition, every vertex $v \in V$ knows its home tree, and routing toward v from *any other vertex* on the tree $\mathbf{home}(v)$ has small stretch. In particular, the header in our construction consists of only the label of the destination. In the Ref. [35] scheme, however, there are n trees, and a certain process is used to find the appropriate tree to route on, which increases the initial decision time, and also some information must be added to the header of the message after the tree is found. Finally, our routing scheme is stateless, as opposed to Ref. [35]. (We remark that using ideas from Ref. [18], one can devise a stateless routing scheme based on Ref. [35], but this scheme seems to suffer from larger header and decision time at each node.)

Theorem 4. *Given a weighted graph $G = (V, E)$ on n vertices and integer parameters $k, b > 1$, there is a stateless routing scheme with stretch $O(k \log\log n)$ that has routing tables of size $O(k \cdot b \cdot n^{1/k})$ and labels of size $(1 + o(1)) \log_b n$. The decision time in each vertex is $O(1)$.*

Observe that choosing parameters $2k$ and $b = n^{1/(2k)}$ for Theorem 4 yields a routing scheme with stretch $O(k \log\log n)$ that has tables of size $O(k \cdot n^{1/k})$ and labels of size only $O(k)$. Another interesting choice of parameters is $b = 2$ and $k = \frac{100 \log n}{\log\log n}$; this provides a scheme with stretch $O(\log n)$ that has tables of size $O(\log^{1.01} n)$ and labels of size $O(\log n)$. Compare this to the Ref. [35] scheme, which for stretch $O(\log n)$ has tables of size $O(\log n)$ and labels of size $O(\log^2 n)$.

1.2 Organization

In Section 3, we present our deterministic Ramsey partitions that are used for Ramsey ultrametrics and distance oracles. In Section 4, we show the Ramsey spanning trees and the application to routing. Each section can be read independently.

2 PRELIMINARIES

Let $G = (V, E)$ be a weighted undirected graph. We assume that the minimal weight of an edge is 1. For any $Y \subseteq V$ and $x, y \in Y$, denote by $\mathbf{dist}(x, y, Y)$ the shortest path distance in $G[Y]$ (the graph induced on Y). For $v \in Y$ and $r \geq 0$, let $B(v, r, Y) = \{u \in Y \mid \mathbf{dist}(v, u, Y) \leq r\}$; when $Y = V$, we simply write $B(v, r)$. We may sometimes abuse notation and not distinguish between a set of vertices and the graph induced by them.

An ultrametric (Z, d) is a metric space satisfying a strong form of the triangle inequality, that is, for all $x, y, z \in Z$, $d(x, z) \leq \max\{d(x, y), d(y, z)\}$. The following definition is known to be an equivalent one (see Ref. [15]).

Definition 1. An ultrametric is a metric space (Z, d) whose elements are the leaves of a rooted labeled tree T. Each $z \in T$ is associated with a label $\ell(z) \geq 0$ such that if $q \in T$ is a descendant of z, then $\ell(q) \leq \ell(z)$ and $\ell(q) = 0$ iff q is a leaf. The distance between leaves $z, q \in Z$ is defined as $d_T(z, q) = \ell(\mathrm{lca}(z, q))$ where $\mathrm{lca}(z, q)$ is the least common ancestor of z and q in T.

3 RAMSEY PARTITIONS

Consider an undirected weighted graph $G = (V, E)$, and a parameter $k \geq 1$. Let D be the diameter of the graph and let $\phi = \lceil \log(D + 1) \rceil$. Let $\rho_i = 2^i/(4k)$. We start by presenting a construction for a collection $\mathcal{S}$ of partial partitions $\mathcal{X}_i$ satisfying the following key properties.

ALGORITHM 1: $\mathcal{S}$ = **FPSDHPP**(G, U)

1: Mark all the nodes in U.
2: Set $\mathcal{X}_\phi = \{V\}$ to be the trivial partition. Set $r(V) \in V$ to be the vertex v with maximal $|B_M(v, 2^{\phi-1}, G)|$.
3: **for** i from $\phi - 1$ to 0 **do**
4:     **for** every subset $X \in \mathcal{X}_{i+1}$ **do**
5:         Set H to be the induced graph on X.
6:         **while** H contains a marked vertex **do**
7:             Pick a node $v \in V(H)$ with maximal $|B_M(v, 2^{i-1}, H)|$.
8:             Let $j(v) \geq 0$ be the minimal integer such that
                $|B_M(v, 2^{i-1} + 2(j(v)+1)\rho_i, H)| \leq |B_M(v, 2^{i-1} + 2j(v)\rho_i, H)| \cdot |Z_i(v, H)|^{1/k}$.
9:             Let $X(v) = B\left(v, 2^{i-1} + (2j(v)+1)\rho_i, H\right)$.
10:             Add $X(v)$ to $\mathcal{X}_i$.
11:             Unmark the nodes in $B(v, 2^{i-1} + 2(j(v)+1)\rho_i, H) \setminus B(v, 2^{i-1} + 2j(v)\rho_i, H)$.
12:             Remove all nodes in $X(v)$ from H.
13:         **end while**
14:     **end for**
15: **end for**
16: **set** $\hat{V}$ to be all nodes that remain marked.

For a set X, a *partial-partition* is a set of nonempty subsets of X such that every element $x \in X$ is in at most one of these subsets. For two partial partitions $\mathcal{P}_1$ and $\mathcal{P}_2$, we say that $\mathcal{P}_1$ is a refinement of $\mathcal{P}_2$ if for every $X_1 \in \mathcal{P}_1$ there is $X_2 \in \mathcal{P}_2$ such that $X_1 \subseteq X_2$.

Definition 2 [(G, U, k)-Fully Padded Strong Diameter Hierarchical Partial Partition]. Given a graph $G = (V, E)$, an index k and a set of nodes $U \subseteq V$, a (G, U, k)-Fully Padded Strong Diameter Hierarchical Partial Partition (**FPSDHPP**) is a collection $\{\mathcal{X}_i\}_{i=0}^{\phi}$ of partial partitions of V where each $X \in \mathcal{X}_i$ has a center $r(X) \in V$, such that the following properties hold.

(i) For every $0 \leq i < \phi$, $\mathcal{X}_i$ is a refinement of $\mathcal{X}_{i+1}$.
(ii) For every $0 \leq i \leq \phi$ and every $X \in \mathcal{X}_i$ and every $v \in X$, $\mathbf{dist}(v, r(X), X) < 2^i$.
(iii) There exists a set $\hat{V} \subseteq U$ such that $|\hat{V}| \geq |U|^{1-1/k}$ and for every $v \in \hat{V}$ and every i, there exists a subset $X \in \mathcal{X}_i$ such that $B(v, \rho_i) \subseteq X$.

For a node v and index i, we say that v is i-padded in $\mathcal{S}$, if there exists a subset $X \in \mathcal{X}_i$ such that $B(v, \rho_i) \subseteq X$. We would like to maximize the number of nodes that are padded on all levels. Note that we do not get to choose the set $\hat{V}$ of padded vertices. Nevertheless, Definition 2 allows us to specify a subset $U \subseteq V$ such that the padded vertices will be chosen from U. In such a case, a significant portion of the vertices in U will indeed be padded.

Fully Padded Strong Diameter Hierarchical Partial Partition Construction. Let us now turn to the construction of the collection $\mathcal{S}$ of partial partitions $\mathcal{X}_i$ given a set U.

In the beginning of the algorithm, all nodes in U are set as *marked*. The algorithm iteratively *unmarks* some of the nodes. The nodes that will remain marked by the end of the process are the nodes that are padded on all levels. For a given graph H, let $B_M(v, d, H)$ (M stands for marked) be the set of marked nodes at distance at most d from v in H.

For a subgraph G' and a node $v \in V(G')$, let $Z_i(v, G') = |B_M(v, 2^i, G')|/|B_M(v, 2^{i-1}, G')|$. The construction is given in Algorithm 1.

Let $X(v)$ be a set constructed in Line 9 of Algorithm 1 when partitioning $X \in \mathcal{X}_{i+1}$. We say that X is the parent of $X(v)$. Let $H(X(v))$ denote the graph in Algorithm 1 just before $X(v)$ was constructed (note that this is a graph induced on a subset of X). Similarly, let $M(X(v))$ denote the set of marked vertices just before $X(v)$ was constructed. (We can also use $H(X'), M(X')$ without (v), if the set X' is clear). We say that $B(v, 2^{i-1} + 2j(v)\rho_i, H(X(v)))$ is the *interior* part of $X(v)$. We also say that the set $B_{M(X(v))}(v, 2^{i-1} + 2(j(v) + 1)\rho_i, H(X(v)))$ (from Line 8 in stage i of the algorithm), is the *responsibility set* of $X(v)$, hereafter referred to as $\mathbf{Res}(X(v))$. Note that every node u that was marked before the processing of X started belongs to exactly one set $\mathbf{Res}(X(v))$ for $X(v) \in \mathcal{X}_i$.

We now define by induction the term *i-surviving* for $0 \le i \le \phi$: All nodes in U are ϕ-surviving. We say that a node is i-surviving if it is $(i+1)$-surviving and it belongs to the interior part of some subset in $\mathcal{X}_i$, or, equivalently, if it remains marked after the construction of $\mathcal{X}_i$. We denote by S_i the set of i-surviving vertices. Note that $S_\phi = U$. Our goal in the analysis is to show that many vertices are 0-surviving, which is exactly the set $\hat{V}$. For a subset $X \in \mathcal{X}_i$, let $\mathbf{Sur}(X)$ be the set of nodes in X that are 0-surviving, that is $S_0 \cap X$. We now turn to the analysis.

The next auxiliary claim helps in showing that property (ii) holds.

Claim 1. *Consider a subset $X \in \mathcal{X}_i$ centered at some node $v = r(X)$. The index $j(v)$ defined in Line 8 of Algorithm 1, satisfies $j(v) \le k - 1$.*

Proof. Seeking contradiction, assume that for every $0 \le j' \le k-1$, $|B_{M(X)}(v, 2^{i-1} + 2(j'+1)\rho_i, H(X))| > |B_{M(X)}(v, 2^{i-1} + 2j'\rho_i, H(X))| \cdot |Z_i(v, H(X))|^{1/k}$. Then, applying this for $j' = k-1, k-2, \ldots, 0$ we get

$$\begin{aligned}|B_{M(X)}(v, 2^i, H(X))| &= |B_{M(X)}(v, 2^{i-1} + 2k\rho_i, H(X))|\\ &> |B_{M(X)}(v, 2^{i-1} + 2(k-1)\rho_i, H(X))| \cdot |Z_i(v, H(X))|^{1/k}\\ &> \cdots > |B_{M(X)}(v, 2^{i-1}, H(X))| \cdot |Z_i(v, H(X))|^{k/k}\\ &= |B_{M(X)}(v, 2^i, H(X))|,\end{aligned}$$

a contradiction. □

The next lemma shows that the collection $\mathcal{S}$ satisfies the basic properties of an **FPSDHPP**.

Lemma 2. *$\mathcal{S}$ is a collection of partial partitions that satisfies properties (i) and (ii).*

Proof. It is straightforward from Line 12 that $\mathcal{S}$ is a collection of partial partitions. Property (i) holds as each $X(v)$ is selected from the graph $H(X(v))$, which is an induced graph over a subset of X (the parent of $X(v)$). Finally, property (ii) follows from Claim 1, as the radius of $X(v)$ is bounded by $2^{i-1} + (2(k-1)+1)\rho_i < 2^{i-1} + 2k\rho_i = 2^i$. □

Next, we argue that if a vertex is 0-surviving, then it is padded in all the levels.

Lemma 3. *Suppose $x \in \mathbf{Sur}(V)$, then x is padded in all the levels.*

Proof. Fix some $x \in \mathbf{Sur}(V)$. To prove that x is i-padded, we assume inductively that x is j-padded for all $i < j \le \phi$ (the base case $i = \phi$ follows as $B(x, \rho_\phi) \subseteq V$). Let $X \in \mathcal{X}_{i+1}$ such that $x \in X$. Set $B = B(x, \rho_i)$. By the induction hypothesis $B \subseteq B(x, \rho_{i+1}) \subseteq X$. Let $X(v) \in \mathcal{X}_i$ such that $x \in X(v)$.

First, we argue that $B \subseteq H(X(v))$. Seeking contradiction, let $X(v') \in \mathcal{X}_i$ be the first created cluster such that there is $u \in B \cap X(v')$. By the minimality of v', it follows that $B \subseteq H(X(v'))$. Thus, $\mathbf{dist}(v, u, H(X(v'))) = \mathbf{dist}(v, u, G) \le \rho_i$. Let $j(v')$ be the index chosen in Line 8 of Algorithm 1. Then, $X(v') = B(v', 2^{i-1} + (2j(v') + 1)\rho_i, H(X(v')))$ using the triangle inequality $\mathbf{dist}(x, v', H(X(v'))) \le 2^{i-1} + (2j(v') + 2)\rho_i$. Therefore, x was unmarked in Line 11, a contradiction.

It remains to show that $B \subseteq X(v)$. Set $j(v)$ s.t. $X(v) = B(v, 2^{i-1} + (2j(v)+1)\rho_i, H(X(v)))$. As x is part of the interior of $X(v)$, it holds that $\mathbf{dist}(x, v, H(X(v))) \le 2^{i-1} + 2j(v)\rho_i$. Therefore, $B \subseteq B(v, 2^{i-1} + (2j(v)+1)\rho_i, H(X(v))) = X(v)$. □

The next lemma bounds the number of surviving nodes.

Lemma 4. *For every index $0 \le i \le \phi$ and every subset $X = X(v) \in \mathcal{X}_i$, the 0-surviving nodes satisfy $|\mathbf{Sur}(X)| \ge |S_i \cap X| \,/\, \left|B_{M(X)}(r(X), 2^{i-1}, H(X))\right|^{\frac{1}{k}}$.*

Proof. We prove the lemma by induction on i. Consider first the base case where $X = X(v) \in \mathcal{X}_0$. As the subsets in $\mathcal{X}_0$ contain a single node (their radius is less than 1), it holds that $|\mathbf{Sur}(X)| = 1 = \frac{1}{1} = |S_0 \cap X| \,/\, \left|B_{M(X)}(r(X), 1/2, H(X))\right|^{\frac{1}{k}}$ (observe that each cluster in $\mathcal{X}_i$ has at least one marked node, for all $0 \le i \le \phi$). Assume the claim holds for every subset $X' \in \mathcal{X}_i$, and consider $X \in \mathcal{X}_{i+1}$. Let $v = r(X)$. Consider the children $X_1, \ldots, X_{j'}$ of X. For every $1 \le h \le j'$, set $v_h = r(X_h)$.

Note that by definition of $j(v_h)$ in Line 8 of Algorithm 1 and by the construction of X_h in Line 9, we have that $|S_i \cap X_h| \ge |\mathbf{Res}(X_h)|/Z_i(v_h, H(X_h))^{1/k}$. Moreover, by the induction hypothesis, we have that $|\mathbf{Sur}(X_h)| \ge |S_i \cap X| \,/\, \left|B_{M(X_h)}(v_h, 2^{i-1}, H(X_h))\right|^{\frac{1}{k}}$ for every h.

We claim that $|B_{M(X)}(v, 2^i, H(X))| \ge |B_{M(X_h)}(v_h, 2^i, H(X_h))|$ for every $1 \le h \le j'$. To see this, note that v is the node with maximal $|B_{M(X)}(v, 2^i, H(X))|$; hence, $|B_{M(X)}(v, 2^i, H(X))| \ge |B_{M(X)}(v_h, 2^i, H(X))|$. In addition, note that $H(X_h) \subseteq H(X)$ and $M(X_h) \subseteq M(X)$; hence, $|B_{M(X)}M(v_h, 2^i, H(X))| \ge |B_{M(X_h)}(v_h, 2^i, H(X_h))|$.

It follows that $|B_{M(X)}(v, 2^i, H(X))| \ge |B_{M(X_h)}(v_h, 2^i, H(X_h))|$. Therefore, the number of 0-surviving nodes in X_h is at least

$$
\begin{aligned}
|\mathbf{Sur}(X_h)| &\ge \frac{|S_i \cap X_h|}{\left|B_{M(X_h)}(v_h, 2^{i-1}, H(X_h))\right|^{\frac{1}{k}}} \\
&\ge \frac{|\mathbf{Res}(X_h)| \,/\, |Z_i(v_h, H(X_h))|^{\frac{1}{k}}}{\left|B_{M(X_h)}(v_h, 2^{i-1}, H(X_h))\right|^{\frac{1}{k}}} \\
&= \frac{|\mathbf{Res}(X_h)|}{\left|B_{M(X_h)}(v_h, 2^{i}, H(X_h))\right|^{\frac{1}{k}}} \cdot \frac{\left|B_{M(X_h)}(v_h, 2^{i-1}, H(X_h))\right|^{\frac{1}{k}}}{\left|B_{M(X_h)}(v_h, 2^{i-1}, H(X_h))\right|^{\frac{1}{k}}} \\
&\ge \frac{|\mathbf{Res}(X_h)|}{\left|B_{M(X)}(v, 2^{i}, H(X))\right|^{\frac{1}{k}}},
\end{aligned}
$$

we conclude that

$$
\begin{aligned}
|\mathbf{Sur}(X)| &= \sum_{h=1}^{j'} |\mathbf{Sur}(X_h)| \\
&\ge \sum_{h=1}^{j'} \frac{|\mathbf{Res}(X_h)|}{\left|B_{M(X)}(v, 2^{i}, H(X))\right|^{\frac{1}{k}}} \\
&= \frac{|S_{i+1} \cap X|}{\left|B_{M(X)}(v, 2^{i}, H(X))\right|^{\frac{1}{k}}}.
\end{aligned}
$$

□

Using Lemma 4 on V with $i = \phi$, we have $|\mathbf{Sur}(X)| \geq |S_\phi \cap X|/|U|^{\frac{1}{k}} = |U|^{1-\frac{1}{k}}$. Combined with Lemma 3, property (iii) follows.

Lemma 5. *The number of marked nodes $\hat{V}$ by the end of Algorithm 1 is at least $|U|^{1-1/k}$. Moreover, for every $v \in \hat{V}$ and every i, there exists a subset $X \in \mathcal{X}_i$ such that $B(v, \rho_i) \subseteq X$.*

Theorem 5. *For every n-point metric space and $k \geq 1$, there exists a subset of size $n^{1-1/k}$ that can be embedded into an ultrametric with distortion $8k - 2$.*

Proof. The hierarchical partial partition $\mathcal{S} = \{\mathcal{X}_i\}$ naturally induces an ultrametric on $\hat{V}$. The singleton sets of $\hat{V}$ are the leaves, and each $X \in \mathcal{X}_i$ for $0 \leq i < \phi$ will be a tree-node that is connected to its parent. Each set in $\mathcal{X}_i$ for $i \geq 1$ will receive the label $2^{i+1}(1 - 1/(4k))$, while the leaves in $\mathcal{X}_0$ receive the label 0 (recall Definition 1).

Consider two nodes $u, v \in \hat{V}$. Assume the least common ancestor of u, v is $X \in \mathcal{X}_i$, for some $1 \leq i \leq \phi$. Hence, $\mathbf{dist}(u, v, G) \leq 2 \cdot (2^i - 2^i/4k)$ (they are both in the interior of X - a ball with radius $\leq 2^i - 2\rho_i$). Since this is the label of X, we conclude that distances in the ultrametric are no smaller than those in G.

Next, we argue that distances increase by a factor of at most $8k - 2$. Consider any u, v as above, and seeking a contradiction, assume that $\mathbf{dist}(v, u, G) < \frac{2^{i+1}(1-1/(4k))}{8k-2} = \rho_i$. Let P be the shortest path from v to u in G. As v was padded in X, necessarily $P \subseteq X$. Consider the first time a vertex $z \in P$ was added to a cluster $X' \in \mathcal{X}_{i-1}$, then $P \subseteq H(X')$. Let j be such that $X' = B(r(X'), 2^{i-2} + (2j + 1)\rho_{i-1}, H(X'))$. Since P is a shortest path, at least one of u, v must be within distance less than $\rho_i/2 = \rho_{i-1}$ from z, w.l.o.g., assume $\mathbf{dist}(v, z, H(X')) \leq \rho_{i-1}$. This implies that $v \in \mathbf{Res}(X') = B(r(X'), 2^{i-2} + (2j + 2)\rho_{i-1}, H(X'))$, and as v is marked, it must lie in the interior of X', which is $B(r(X'), 2^{i-2} + 2j\rho_{i-1}, H(X'))$. But then the triangle inequality yields that $u \in \mathbf{Res}(X')$. Yet u and v belong to different clusters of $\mathcal{X}_{i-1}$, and so $u \notin X'$, which is a contradiction to the fact that $u \in \hat{V}$. □

3.1 Deterministic Construction of Distance Oracles

We show a distance oracle with $O(n^{1+1/k})$ size, $(8 + \epsilon)k$ worst-case stretch and $O(1/\epsilon)$ query time (which is $O(1)$ for any fixed epsilon). For simplicity, we start by showing a construction similar to the Ref. [29] oracle, with $O(k \cdot n^{1+1/k})$ size, $16k$ stretch, and $O(1)$ query time. We will later see how to reduce the size and stretch. Let D be the diameter of the graph.

Our distance oracle is constructed as follows.

(1) Set $U \leftarrow V$.
(2) Construct the collection of partial partitions $\mathcal{S}(U) = \mathbf{FPSDHPP}(G, U)$ on the graph G and the set U. Remove from U the set of nodes $\hat{V}$ that were padded in all levels in $\mathcal{S}(U)$. Continue this process as long as $U \neq \emptyset$.
(3) Let $\mathcal{M}$ be the set of all collections $\mathcal{S}(U)$ that were constructed by this process.
(4) For every $\mathcal{S} \in \mathcal{M}$, construct a cluster $X(\mathcal{S})$ as follows.
(5) Let $\mathcal{S} = \{\mathcal{X}_0, \ldots, \mathcal{X}_\phi\}$. All nodes V are the leaves (recall that only nodes in $\hat{V}$ are in $\mathcal{X}_0$). For every index i and every set $X \in \mathcal{X}_i$, add an intermediate node. Connect X to its parent set. Connect every node $v \in V$ to the set $X \in \mathcal{X}_i$ of minimal i such that $v \in X$. This completes the construction of $X(\mathcal{S})$.
(6) In addition, we preprocess $X(\mathcal{S})$ so that least common ancestor (LCA) queries could be done in constant time. In order to do that, we invoke any scheme that takes a tree and preprocesses it in linear time so that LCA queries can be answered in constant time (see Refs [11] and [28]).

(7) Finally, note that for every node v, there exists a collection $\mathcal{S} \in \mathcal{M}$, where v is padded in all levels. Denote this collection by $\mathbf{home}(v)$.

The query phase is done as follows. Given two nodes s and t, let $\mathcal{S} = \mathbf{home}(s)$ and let $\mathcal{S} = \{\mathcal{X}_i \mid 1 \leq i \leq \phi\}$. Find the least common ancestor of s and t in $X(\mathcal{S})$, and let i be its level. Namely, let $\mu \in X(\mathcal{S})$ be the least common ancestor of s and t, and let X be the cluster μ represents; the index i is the index such that $X \in \mathcal{X}_i$. Return 2^{i+1} (denoted by $\hat{\mathbf{dist}}(s,t)$).

Lemma 6. $\mathbf{dist}(s,t) \leq \hat{\mathbf{dist}}(s,t) < 16k \cdot \mathbf{dist}(s,t)$.

Proof. Let $d = \mathbf{dist}(s,t,G)$ and let j be the index such that $2^{j-1} < d \leq 2^j$. Let $X_i \in \mathcal{X}_i \in \mathcal{S} = \mathbf{home}(s)$ be the i level subset such that $s \in X_i$. Recall that s is padded in all the subsets X_i for $0 \leq i \leq \phi$.

Note that X_i has diameter smaller than $2 \cdot 2^i$ (follows from property (iii)). Therefore, $t \in X_i$ implies that $\mathbf{dist}(s,t,G) < 2 \cdot 2^i = 2^{i+1}$. In particular, $t \notin X_i$ for every $i < j-1$. Hence, the least common ancestor is at least at level $j-1$. Hence, the minimal distance returned by the algorithm is $\hat{\mathbf{dist}}(s,t) \geq 2^j \geq d$.

It remains to show that $\hat{\mathbf{dist}}(s,t) \leq 16k \cdot d$. Let i be the level of s and t's least common ancestor. Note that $t \notin X_{i-1}$. Also, recall that s is padded in X_{i-1}; thus, $B(s,\rho_{i-1}) \subseteq X_{i-1}$, which implies $d \geq \rho_{i-1} = 2^{i-1}/(4k) = \hat{\mathbf{dist}}(s,t)/(16k)$. □

Let U_i be the set U after constructing the first i collections. Note that $|U_{i+1}| \leq |U_i| - |U_i|^{1-1/k}$. By resolving this recurrence relation, one can show that the number of phases is $O(kn^{1/k})$ (see Ref. [29, Lemma 4.2]) . Notice that for every $\mathcal{S} \in \mathcal{M}$, $T(\mathcal{S})$ is of size $O(n)$. Hence, the size of our data structure is $O(kn^{1+1/k})$.

Reducing the size of the data structure. We now show how to reduce the size of the data structure to $O(n^{1+1/k})$. We only outline the modifications to the algorithm and the analysis and omit the full details.

Here, we will use only the metric structure of the graph G, while ignoring the structure induced by the edges. Specifically, in Line (2) of the algorithm, instead of the graph G, we will use the graph G_U, which is the complete graph over U, where the weight of each edge $\{u,v\}$ is equal to $\mathbf{dist}(u,v,G)$. This change allows us to remove the nodes $\hat{V}$ from G_U after each iteration.

The query algorithm, given two nodes s and t is as follows. Let $\mathcal{S}_s = \mathbf{home}(s)$ and $\mathcal{S}_t = \mathbf{home}(t)$, and assume w.l.o.g. that $\mathcal{S}_s$ was constructed before $\mathcal{S}_t$. Find the least common ancestor of s and t in $X(\mathcal{S}_s)$ and let i be its level. Return 2^{i+1}.

Following the analysis of the previous construction, we can show that properties (i)–(iv) are satisfied and that the stretch is bound by $16k$. The size of the data structure is bounded by $O(n^{1+1/k})$ (see Ref. [29], Lemma 4.2).

Reducing the stretch to $8(1+\epsilon)k$. We now explain how to reduce the stretch to $8(1+\epsilon)k$. Note that we lose a factor of 2 in the stretch since we look on distances in multiples of two. Recall that in the algorithm, for a pair of vertices s,t at distance d, we looked on the minimal index j such that $d \leq 2^j$. It may happen that d is only slightly larger than 2^{j-1}. Note that by just considering all distances $(1+\epsilon)^i$ rather than all distances 2^i, we get that the number of nodes that are padded in all levels is a fraction of $1/n^{1/(\epsilon k)}$ rather than $1/n^{1/k}$, which is dissatisfying. So, instead, we construct $O(1/\epsilon)$ different copies of our data structure, one for each $1+\ell\epsilon$ for $0 \leq \ell < 1/\epsilon$. In the copy ℓ of the data structure, we consider distances $(1+\ell\epsilon)2^i$ for every $0 \leq i \leq \phi$. Specifically, i-clusters have radius bounded by $(1+\ell\epsilon)2^i$, while the padding parameter is $\rho_{\ell,i} = (1+\ell\epsilon)\rho_i$. We denote by $\mathbf{home}_\ell(s)$ the collection $\mathcal{S}$, created for the ℓ's distance oracle, where s is padded in all levels.

The distance estimation of the ℓ's copy (denoted by $\hat{\mathbf{dist}}_\ell(s,t)$), will be $(1+(\ell+1)\epsilon)2^{i_\ell}$, where i_ℓ is the level of the least common ancestor of s and t in $\mathbf{home}_\ell(s)$.

Set $d = \mathbf{dist}(s,t)$. For every ℓ, we have

$$d > \rho_{\ell, i_\ell - 1} = \frac{(1+\ell\epsilon)2^{i_\ell - 1}}{4k} = \frac{(1+\ell\epsilon)}{(1+(\ell+1)\epsilon)} \cdot \frac{\hat{\mathbf{dist}}_\ell(s,t)}{8k} \geq \frac{\hat{\mathbf{dist}}_\ell(s,t)}{8(1+\epsilon)k}. \tag{1}$$

On the other hand, there exist indices ℓ', j such that $(1+\ell'\epsilon)2^{j-1} < d \leq (1+(\ell'+1)\epsilon)2^{j-1}$. Following the analysis above, as t is not separated from s at level $i_{\ell'}$, it holds that $i_{\ell'} \geq j-1$. Therefore,

$$\hat{\mathbf{dist}}_{\ell'}(s,t) = (1+(\ell+1)\epsilon)2^{i_{\ell'}} \geq (1+(\ell+1)\epsilon)2^{j-1} \geq d. \tag{2}$$

In the query phase, we iterate over all $O(1/\epsilon)$ copies, invoke the query algorithm in each copy, and return the largest distance. By Equations (1) and (2), the stretch is $8(1+\epsilon)k$ rather than $16k$. The query time is $O(1/\epsilon)$, which is $O(1)$ for every fixed ϵ.

4 RAMSEY SPANNING TREES

In this section, we describe the construction of Ramsey spanning trees; each tree will be built using the petal decomposition framework of Ref. [5]. Roughly speaking, the petal decomposition is an iterative method to build a spanning tree of a given graph. In each level, the current graph is partitioned into smaller diameter pieces, called petals, and a single central piece, which are then connected by edges in a tree structure. Each of the petals is a ball in a certain metric. The main advantage of this framework is that it produces a spanning tree whose diameter is proportional to the diameter of the graph, while allowing large freedom for the choice of radii of the petals. Specifically, if the graph diameter is Δ, the spanning tree diameter will be $O(\Delta)$, and each radius can be chosen in an interval of length $\approx\Delta$. For the specific choice of radii that will ensure a sufficient number of vertices are fully padded, we use a region growing technique based on ideas from Refs [22] and [33].

4.1 Preliminaries

For subset $S \subseteq G$ and a center vertex $x_0 \in S$, the radius[2] of S w.r.t. x_0, $\Delta_{x_0}(S)$ is the minimal Δ such that $B(x_0, \Delta, S) = S$. (If for every Δ, $B(x_0, \Delta, S) \neq S$, (this can happen iff S is not connected) we say that $\Delta_{x_0}(S) = \infty$.) When the center x_0 is clear from context or is not relevant, we will omit it.

Definition 3. Given a graph $G = (V,E)$, a Strong Diameter Hierarchical Partition (**SDHP**) is a collection $\{\mathcal{A}_i\}_{i\in[\Phi]}$ of partitions of V, where each cluster $X \in \mathcal{X}_i$ contains a center $r(X) \in \mathcal{X}_i$, such that:

- $\mathcal{A}_\Phi = \{V\}$ (i.e., the first partition is the trivial one).
- $\mathcal{A}_1 = \{\{v\}_{v\in V}\}$ (i.e., in the last partition every cluster is a singleton).
- For every $1 \leq i < \Phi$ and $A \in \mathcal{A}_i$, there is $A' \in \mathcal{A}_{i+1}$ such that $A \subseteq A'$ (i.e., $\mathcal{A}_i$ is a refinement of $\mathcal{A}_{i+1}$). Moreover, $\Delta(A) \leq \Delta(A')$.

Definition 4 (Padded, Fully Padded). Given a graph $G = (V,E)$ and a subset $A \subseteq V$, we say that a vertex $y \in A$ is *ρ-padded* by a subset $A' \subseteq A$ (w.r.t. A) if $B(y, \Delta(A)/\rho, G) \subseteq A'$. See Figure 1 for illustration.

We say that $x \in V$ is *ρ-fully-padded* in the SDHP $\{\mathcal{A}_i\}_{i\in[\Phi]}$, if for every $2 \leq i \leq \Phi$ and $A \in \mathcal{A}_i$ such that $x \in A$, there exists $A' \in \mathcal{A}_{i-1}$ such that x is ρ-padded by A' (w.r.t. A).

[2] In the literature, this notion is sometimes called the eccentricity of x_0.

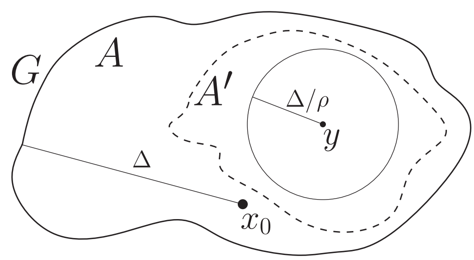

Fig. 1. A is a subset of vertices in the graph G with center x_0. The radius of A (w.r.t x_0) is Δ. A' is a subset of A (denoted by the dashed line). As $B(y, \Delta/\rho, G) \subseteq A'$, we say that the vertex y is padded by A'.

ALGORITHM 2: $T =$ `hierarchical-petal-decomposition`$(G[X], x_0, t, \Delta)$

```
if |X| = 1 then
    return G[X].
end if
Let ({X_j, x_j, t_j, Δ_j}_{j=0}^s, {(y_j, x_j)}_{j=1}^s) = petal-decomposition(G[X], x_0, t, Δ);
for each j ∈ [0, ..., s] do
    T_j = hierarchical-petal-decomposition (G[X_j], x_j, t_j, Δ_j);
end for
Let T be the tree formed by connecting T_0, ..., T_s using the edges {y_1, x_1}, ..., {y_s, x_s};
```

4.2 Petal Decomposition

Here, we will give a concise description of the Petal decomposition algorithm, focusing on the main properties we will use. For proofs and further details we refer to Ref. [5].

The `hierarchical-petal-decomposition` (see Algorithm 2) is a recursive algorithm. The input is $G[X]$ (a graph $G = (V, E)$ induced over a set of vertices $X \subseteq V$), a center $x_0 \in X$, a target $t \in X$, and the radius $\Delta = \Delta_{x_0}(X)$.[3] The algorithm invokes `petal-decomposition` to partition X into $X_0, X_1, \ldots, X_s$ (for some integer s), and also provides a set of edges $\{(x_1, y_1), \ldots, (x_s, y_s)\}$ and targets $t_0, t_1, \ldots, t_s$. The `Hierarchical-petal-decomposition` algorithm now recurses on each $(G[X_j], x_j, t_j, \Delta_{x_j}(X_j))$ for $0 \le j \le s$, to get trees $\{T_j\}_{0 \le j \le s}$, which are then connected by the edges $\{(x_j, y_j)\}_{1 \le j \le s}$ to form a spanning tree T for $G[X]$ (the recursion ends when X_j is a singleton). See Figure 2 for illustration.

Next, we describe the `petal-decomposition` procedure, see Algorithm 3. Initially, it sets $Y_0 = X$, and for $j = 1, 2, \ldots, s$ it carves out the petal X_j from the graph induced on Y_{j-1}, and sets $Y_j = Y_{j-1} \setminus X_j$ (in order to control the radius increase, the first petal is cut using different parameters). The definition of petal guarantees that $\Delta_{x_0}(Y_j)$ is non-increasing (see Ref. [5, Claim 1]), and when at step s it becomes at most $3\Delta/4$, define $X_0 = Y_s$ and then the `petal-decomposition` routine ends. In carving of the petal $X_j \subseteq Y_{j-1}$, the algorithm chooses an arbitrary target $t_j \in Y_{j-1}$ (at distance at least $3\Delta/4$ from x_0) and a range $[lo, hi]$ of size $hi - lo \in \{\Delta/8, \Delta/4\}$, which are provided to the sub-routine `create-petal`.

(One may notice that in Line 15 of the `petal-decomposition` procedure, the weight of some edges is changed by a factor of 2. This can happen at most once for every edge throughout the

[3]Rather than inferring $\Delta = \Delta_{x_0}(X)$ from $G[X]$ and x_0 as in Ref. [5], we can think of Δ as part of the input. We shall allow any $\Delta \ge \Delta_{x_0}(X)$. We stress that, in fact, in the algorithm, we always use $\Delta_{x_0}(X)$ and consider this degree of freedom only in the analysis.

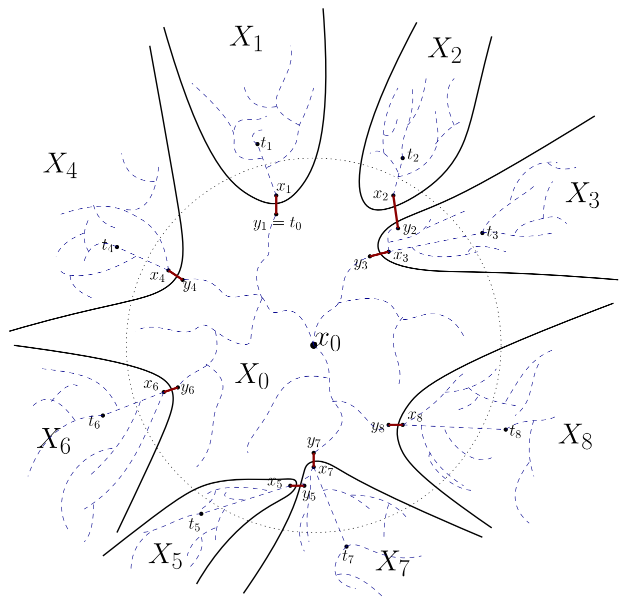

Fig. 2. A schematic depiction of a partition done by the `petal-decomposition` algorithm. The dotted ball contain the vertices within $3\Delta/4$ from the center x_0. The algorithm iteratively picks a target t_j outside this ball and builds a petal X_j that will be connected to the rest of the graph by the edge (x_j, y_j). The dotted lines represent the trees created recursively in each X_j.

`hierarchical-petal-decomposition` execution; thus, it may affect the padding parameter by a factor of at most 2. This re-weighting is ignored for simplicity.)

Both `hierarchical-petal-decomposition` and `petal-decomposition` are essentially the algorithms that appeared in Ref. [5]. The main difference from their work lies in the `create-petal` procedure, depicted in Algorithm 4. It carefully selects a radius $r \in [lo, hi]$, which determines the petal X_j together with a connecting edge $(x_j, y_j) \in E$, where $x_j \in X_j$ is the center of X_j and $y_j \in Y_j$. It is important to note that the target $t_0 \in X_0$ of the central cluster X_0 is determined during the creation of the first petal X_1. It uses an alternative metric on the graph, known as *cone-metric*:

Definition 5 (Cone-metric). Given a graph $G = (V, E)$, a subset $X \subseteq V$ and points $x, y \in X$, define the *cone-metric* $\rho = \rho(X, x, y) : X^2 \to \mathbb{R}^+$ as $\rho(u, v) = |(d_X(x, u) - d_X(y, u)) - (d_X(x, v) - d_X(y, v))|$.

(The cone-metric is, in fact, a pseudo-metric, i.e., distances between distinct points are allowed to be 0.) The ball $B_{(X,\rho)}(y, r)$ in the cone-metric $\rho = \rho(X, x, y)$ contains all vertices u whose shortest path to x is increased (additively) by at most r if forced to go through y.

In the `create-petal` algorithm, while working in a subgraph $G[Y]$ with two specified vertices: a center x_0 and a target t, we define $W_r(Y, x_0, t) = \bigcup_{p \in P_{x_0 t}:\, d_Y(p,t) \le r} B_{(Y, \rho(Y, x_0, p))}(p, \frac{r - d_Y(p,t)}{2})$, which is union of balls in the cone-metric, where any vertex p in the shortest path from x_0 to t of distance at most r from t is a center of a ball with radius $\frac{r - d_Y(p,t)}{2}$.

The following facts are from Ref. [5].

Fact 1. *Running* `Hierarchical-petal-decomposition` *on input* $(G[X], x_0, t, \Delta_{x_0}(X))$ *will provide a spanning tree* T *satisfying*

$$\Delta_{x_0}(T) \le 4\Delta_{x_0}(X).$$

ALGORITHM 3: $(\{X_j, x_j, t_j, \Delta_j\}_{j=0}^{s}, \{(y_j, x_j)\}_{j=1}^{s})$ = `petal-decomposition`$(G[X], x_0, t, \Delta)$

1: Let $Y_0 = X$; Set $j = 1$;
2: **if** $d_X(x_0, t) \geq \Delta/2$ **then**
3: Let $X_1 =$ `create-petal` $(G[Y_0], [d_X(x_0, t) - \Delta/2, d_X(x_0, t) - \Delta/4], x_0, t)$;
4: $Y_1 = Y_0 \setminus X_1$;
5: Let $\{x_1, y_1\}$ be the unique edge on the shortest path $P_{x_0 t}$ from x_0 to t in Y_0, where $x_1 \in X_1$ and $y_1 \in Y_1$.
6: Set $t_0 = y_1$, $t_1 = t$; $j = 2$;
7: **else**
8: set $t_0 = t$.
9: **end if**
10: **while** $Y_{j-1} \setminus B_X(x_0, \frac{3}{4}\Delta) \neq \emptyset$ **do**
11: Let $t_j \in Y_{j-1}$ be an arbitrary vertex satisfying $d_X(x_0, t_j) > \frac{3}{4}\Delta$;
12: Let $X_j =$ `create-petal`$(G[Y_j], [0, \Delta/8], x_0, t_j)$;
13: $Y_j = Y_{j-1} \setminus X_j$;
14: Let $\{x_j, y_j\}$ be the unique edge on the shortest path $P_{x_j t_j}$ from x_0 to t_j in Y_{j-1}, where $x_j \in X_j$ and $y_j \in Y_j$.
15: For each edge $e \in P_{x_j t_j}$, set its weight to be $w(e)/2$;
16: Let $j = j + 1$;
17: **end while**
18: Let $s = j - 1$;
19: Let $X_0 = Y_s$;
20: **return** $(\{X_j, x_j, t_j, \Delta_{x_j}(X_j)\}_{j=0}^{s}, \{(y_j, x_j)\}_{j=1}^{s})$;

FACT 2. *If the* `petal-decomposition` *partitions X with center x_0 into $X_0, \ldots, X_s$ with centers $x_0, \ldots, x_s$, then for any $0 \leq j \leq s$, we have*
$\Delta_{x_j}(X_j) \leq (3/4) \cdot \Delta_{x_0}(X)$.

We will need the following observation. Roughly speaking, it says that when the `petal-decomposition` algorithm is carving out X_{j+1}, it is oblivious to the past petals $X_1, \ldots, X_j$, edges and targets—it only cares about Y_j and the original diameter Δ.

OBSERVATION 7. *Assume that* `petal-decomposition` *on input* $(G[X], x_0, t, \Delta_{x_0}(X))$ *returns as output* $(X_0, X_1, \ldots, X_s, \{y_1, x_1\}, \ldots, \{y_s, x_s\}, t_0, \ldots, t_s)$. *Then, running* `petal-decomposition` *on input* $(G\left[Y_j\right], x_0, t_0, \Delta_{x_0}(X))$ *will output* $(X_0, X_{j+1} \ldots, X_s, \left\{y_{j+1}, x_{j+1}\right\}, \ldots, \{y_s, x_s\}, t_0, t_{j+1} \ldots, t_s)$.

4.3 Choosing a Radius

Fix some $1 \leq j \leq s$, and consider carving the petal X_j from the graph induced on $Y = Y_{j-1}$ (i.e., Line 12 in the `petal-decomposition` algorithm). While the algorithm of Ref. [5] described a specific way to choose the radius, we require a somewhat more refined choice. The properties of the petal decomposition described above (in Section 4.2), together with Fact 2 and Fact 1, hold for any radius picked from a given interval. We will now describe the method to select a radius that suits our needs. The `petal-decomposition` algorithm provides an interval $[lo, hi]$ of size at least $\Delta/8$, and for each $r \in [lo, hi]$, let $W_r(Y, x_0, t) \subseteq Y$ denote the petal of radius r (usually, we will omit (Y, x_0, t)). The following fact demonstrates that petals are similar to balls.

FACT 3. *For every $y \in W_r$ and $l \geq 0$, $B(y, l, Y) \subseteq W_{r+4l}$.*

Note that Fact 3 implies that W_r is monotone in r, i.e., for $r \leq r'$, it holds that $W_r \subseteq W_{r'}$.

Our algorithm will maintain a set of *marked* vertices $M \subseteq V$, and will update it in any petal creation. Roughly speaking, the marked vertices are those that are fully padded in the (partial) hierarchical partition generated so far by the algorithm. If initially $|M| = m$, we want that at the end of the process, at least $m^{1-1/k}$ vertices will remain marked. In the partition of X to $X_0, \ldots, X_s$, some of the marked vertices will be ρ-padded by a petal X_j (w.r.t. X), and some of the others will be unmarked by the following rule. Fact 3 implies that if we choose a radius r when creating some petal $X_j = W_r$, then all marked vertices in $W_{r-4\Delta/\rho}$ will be ρ-padded by X_j, and, thus, remain marked. All the marked vertices in $W_{r+4\Delta/\rho} \setminus W_{r-4\Delta/\rho}$ are considered unmarked from now on, since their Δ/ρ ball may intersect more than one cluster in the current partition (note that some of these vertices can be outside X_j).

Our algorithm to select a radius is based on region growing techniques, similar to those in Algorithm 1, but rather more involved. Since in the petal decomposition framework we cannot pick as center a vertex maximizing the "small ball," we first choose an appropriate range that mimics that choice (see, e.g., Line 9 in the algorithm below)—this is the reason for the extra factor of $\log\log n$. The basic idea in region growing is to charge the number of marked vertices whose ball is cut by the partition (those in $W_{r+4\Delta/\rho} \setminus W_{r-4\Delta/\rho}$), to those that are saved (in $W_{r-4\Delta/\rho}$). In our setting, we are very sensitive to constant factors in this charging scheme (as opposed to the average stretch considered in Ref. [22]) because these constants are multiplied throughout the recursion. In particular, we must avoid a range in $[lo, hi]$ that contains more than half of the marked vertices, a constraint that did not exist in previous manifestation of this region growing scheme. To this end, if the first half $[lo, mid]$ (with $mid = (hi + lo)/2$) is not suitable, we must "cut backward" in the regime $[mid, hi]$ and charge the marked vertices that were removed from M to the remaining graph Y_j, rather than to those saved in the created cluster X_j.

4.4 Proof of Correctness

Let $z \in V$ be an arbitrary vertex, given a set $M \subseteq V$, let T be the tree returned by calling `Hierarchical-petal-decomposition` on $(G[V], z, z, \Delta_z(V))$ and marked vertices M. There is a natural **SDHP** $\mathcal{X} = \{\mathcal{X}_i\}_{i=1}^{\Phi}$ associated with the tree T, where $\mathcal{X}_i$ consists of all the clusters created in level $\Phi - i$ of the recursion (and $\mathcal{X}_\Phi = \{V\}$). By Fact 2, the radius is always non-increasing. Hence, $\mathcal{X}$ is indeed an **SDHP**, denote by $\mathbf{Sur}(M) \subseteq M$ the set of vertices that remained marked throughout the execution of `Hierarchical-petal-decomposition`.

LEMMA 8. *Suppose $x \in \mathbf{Sur}(M)$, then x is ρ-fully-padded in $\mathcal{X}$.*

PROOF. Fix any $2 \leq i \leq \Phi$. Let $X \in \mathcal{X}_i$ be the cluster containing x in the $(\Phi - i)$-th level of the recursion with $\Delta = \Delta(X)$. Assume X was partitioned by `petal-decomposition` into $X_0, \ldots, X_s$, and let $X_j \subseteq X$ be the cluster containing $x \in X_j$. Assuming (inductively) that x was ρ-padded by X, we need to show that it is also ρ-padded by X_j; that is, $B = B(x, \Delta/\rho, G) \subseteq X_j$. (Note that $B \subseteq X$ since the radii are non-increasing, so x is padded in all higher levels.)

First, we argue that none of the petals $X_1, \ldots, X_{j-1}$ intersects B. Seeking contradiction, assume it is not the case, and let $1 \leq j' < j$ be the minimal such that there exists $y \in X_{j'} \cap B$. By the minimality of j', it follows that $B \subseteq Y' = Y_{j'-1}$; thus, $\mathbf{dist}(x, y, Y') = \mathbf{dist}(x, y, G) \leq \Delta/\rho$. Let r' be the radius chosen when creating the petal $X_{j'} = W'_{r'}$, and Fact 3 implies that

$$x \in B(y, \Delta/\rho, Y') \subseteq W'_{r'+4\Delta/\rho} = W'_{r'+R/(4Lk)},$$

where we recall that $\Delta = 8R$ and $\rho = 2^7 Lk$. This is a contradiction to the fact that $x \in \mathbf{Sur}(X)$: clearly $x \notin W'_{r'-R/(4Lk)}$ since it is not included in $X_{j'} = W'_{r'}$ (and using the monotonicity of W'_r), so it should have been removed from M when creating $X_{j'}$ (in Line 21 of the algorithm).

ALGORITHM 4: $X = \texttt{create-petal}(G[Y], [lo, hi], x_0, t)$

Let $m = |Y \cap M|$;
$L = \lceil 1 + \log\log m \rceil$;
$R = hi - lo$; $mid = (lo + hi)/2 = lo + R/2$;
For every r, denote $W_r = W_r(Y, x_0, t)$, $w_r = |M \cap W_r|$;
if $w_{mid} \leq \frac{m}{2}$ **then**
  if $w_{lo+\frac{R}{2L}} = 0$ **then**
    Set $r = lo + \frac{R}{4L}$;
  else
    Choose $[a, b] \subseteq [lo, mid]$ such that $b - a = \frac{R}{2L}$ and $w_a \geq w_b^2/m$; {see Lemma 11}
    Pick $r \in [a, b]$ such that $w_{r+\frac{b-a}{2k}} \leq w_{r-\frac{b-a}{2k}} \cdot \left(\frac{w_b}{w_a}\right)^{\frac{1}{k}}$; {see Lemma 12}
  end if
else
  For every $r \in [lo, hi]$, denote $q_r = |(Y \setminus W_r) \cap M|$;
  if $q_{hi-\frac{R}{2L}} = 0$ **then**
    Set $r = hi - \frac{R}{4L}$;
  else
    Choose $[b, a] \subseteq [mid, hi]$ such that $a - b = \frac{R}{2L}$ and $q_a \geq q_b^2/m$; {see Lemma 13}
    Pick $r \in [b, a]$ such that $q_{r-\frac{a-b}{2k}} \leq q_{r+\frac{a-b}{2k}} \cdot \left(\frac{q_b}{q_a}\right)^{\frac{1}{k}}$; {see Lemma 14}
  end if
end if
$M \leftarrow M \setminus (W_{r+\frac{R}{4Lk}} \setminus W_{r-\frac{R}{4Lk}})$
return W_r;

For the case $j = 0$, the same reasoning shows B does not intersect any petal $X_1, \ldots, X_s$, and we are done. For $j > 0$, it remains to show that $B \subseteq X_j$, but this follows by a similar calculation. Let r be the radius chosen for creating the petal $X_j = W_r$, and $Y = Y_{j-1}$. We have $B \subseteq Y$, and since $x \in \mathbf{Sur}(X)$ it must be that $x \in W_{r-R/(4Lk)}$. Again, by Fact 3, we have

$$
\begin{aligned}
B = B(x, \Delta/\rho, G) &= B(x, \Delta/\rho, Y) \\
&\subseteq W_{r-R/(4Lk)+4\Delta/\rho} = W_r = X_j.
\end{aligned}
$$
□

Lemma 9. *Consider a vertex $v \in \mathbf{Sur}(M)$; then, for every $u \in V$, $\mathbf{dist}(v, u, T) \leq 8\rho \cdot \mathbf{dist}(v, u, G)$.*

Proof. Let $\mathcal{X} = \{\mathcal{X}_i\}_{i=1}^{\Phi}$ be the **SDHP** associated with T, and for $1 \leq i \leq \Phi$, let $A_i \in \mathcal{X}_i$ be the cluster containing v. Take the minimal $2 \leq i \leq \Phi$ such that $u \in A_i$ (there exists such an i since $u \in A_\Phi = V$ and $u \notin A_1 = \{v\}$). By Lemma 8, v is ρ-fully-padded, so we have that $B(v, \Delta/\rho, G) \subseteq A_{i-1}$, where $\Delta = \Delta(A_i)$. But as $u \notin A_{i-1}$, it must be that $\mathbf{dist}(u, v, G) > \Delta/\rho$. Since both $u, v \in A_i$, Fact 1 implies that the radius of the tree created for A_i is at most 4Δ, so that

$$\mathbf{dist}(u, v, T) \leq 2 \cdot 4\Delta \leq 8\rho \cdot \mathbf{dist}(u, v, G).$$
□

Lemma 10. $|\mathbf{Sur}(M)| \geq |M|^{1-1/k}$.

Proof. We prove by induction on $|X|$ that if a cluster $X \in \mathcal{X}_i$ (for some $1 \leq i \leq \Phi$) has currently m marked vertices, then at the end of the process, at least $m^{1-1/k}$ of them will remain marked.

The base case when X is a singleton is trivial. For the inductive step, assume we call `petal-decomposition` on $(G[X], x_0, t, \Delta)$ with $\Delta \geq \Delta_{x_0}(X)$ and the current marked vertices $\hat{M}$. Assume that the `petal-decomposition` algorithm does a non-trivial partition of X to $X_0, \ldots, X_s$ (if it is the case that all vertices are sufficiently close to x_0, then no petals will be created, and the `hierarchical-petal-decomposition` will simply recurse on $(G[X], x_0, t, \Delta_{x_0}(X))$, so we can ignore this case). Denote by M_j the marked vertices that remain in X_j (just before the recursive call on X_j), and recall that $\mathbf{Sur}(X_j)$ is the set of vertices of M_j that remain marked until the end of the `hierarchical-petal-decomposition` algorithm. Then, $\mathbf{Sur}(X) = \bigcup_{0 \leq j \leq s} \mathbf{Sur}(X_j)$, and we want to prove that $|\mathbf{Sur}(X)| \geq m^{1-1/k}$.

Let $X_1 = W_r$ be the first petal created by the `petal-decomposition` algorithm, and $Y_1 = X \setminus X_1$. Denote by $\mathbf{Res}(X_1) = W_{r+R/(4Lk)} \cap \hat{M}$ the responsibility set for X_1 (i.e., the marked vertices that are either in M_1 or were removed from $\hat{M}$ when X_1 was created). Define $M' = \hat{M} \setminus \mathbf{Res}(X_1)$, the set of marked vertices that remain in Y_1. By Observation 7, we can consider the remaining execution of `petal-decomposition` on Y_1 as a new recursive call of `petal-decomposition` with input $(G[Y_1], x_0, t_0, \Delta)$ and marked vertices M'. Since $|X_1|, |Y_1| < |X|$, the induction hypothesis implies that $|\mathbf{Sur}(X_1)| \geq |M_1|^{1-1/k}$ and $|\mathbf{Sur}(Y_1)| \geq |M'|^{1-1/k}$.

We now do a case analysis according to the choice of radius in Algorithm 4.

(1) **Case 1:** $w_{mid} \leq m/2$ and $w_{lo+R/(2L)} = 0$. In this case, we set $r = lo + R/(4L)$. Note that $w_{r+R/(4Lk)} \leq w_{lo+R/(2L)} = 0$, so $M' = \hat{M}$; by the induction hypothesis on Y_1, the number of fully padded vertices is $|\mathbf{Sur}(X)| = |\mathbf{Sur}(Y_1)| \geq |\hat{M}|^{1-1/k} = m^{1-1/k}$, as required.

(2) **Case 2:** $w_{mid} \leq m/2$ and $w_{lo+R/(2L)} > 0$. In this case, we pick $a, b \in [lo, hi]$ so that $b - a = R/(2L)$ and

$$w_a > w_b^2/m, \tag{3}$$

and also choose $r \in [a, b]$ such that $w_{r+\frac{b-a}{2k}} \leq w_{r-\frac{b-a}{2k}} \cdot \left(\frac{w_b}{w_a}\right)^{1/k}$. As $\frac{b-a}{2k} = \frac{R}{4Lk}$ and $|M_1| = w_{r-R/(4Lk)}$, we have that

$$|M_1| \geq \mathbf{Res}(X_1) \cdot \left(\frac{w_a}{w_b}\right)^{1/k}. \tag{4}$$

By the induction hypothesis on X_1, we have that

$$\begin{aligned} |\mathbf{Sur}(X_1)| &\geq \frac{|M_1|}{|M_1|^{1/k}} \\ &\overset{(4)}{\geq} |\mathbf{Res}(X_1)| \cdot \left(\frac{w_a}{|M_1| \cdot w_b}\right)^{1/k} \\ &\overset{(3)}{\geq} |\mathbf{Res}(X_1)| \cdot \left(\frac{w_b}{m \cdot |M_1|}\right)^{1/k} \\ &\geq \frac{|\mathbf{Res}(X_1)|}{m^{1/k}}, \end{aligned}$$

where in the last inequality, we use that $|M_1| = w_{r-(b-a)/(2k)} \leq w_b$. Now, by the induction hypothesis on Y_1, we get

$$
\begin{aligned}
|\mathbf{Sur}(X)| &= |\mathbf{Sur}(Y_1)| + |\mathbf{Sur}(X_1)| \\
&\geq |M'|^{1-1/k} + \frac{|\mathbf{Res}(X_1)|}{m^{1/k}} \\
&\geq \frac{|M'| + |\mathbf{Res}(X_1)|}{m^{1/k}} = \frac{|\hat{M}|}{m^{1/k}} = m^{1-1/k}.
\end{aligned}
$$

(3) **Case 3:** $w_{mid} > m/2$ and $q_{hi-R/(2L)} = 0$. In this case, we set $r = hi - R/(4L)$. Note that $q_{r-R/(4Lk)} \leq q_{hi-R/(2L)} = 0$ (recall that q_r is non-increasing in r, by Fact 3), so $M_1 = \hat{M}$; by the induction hypothesis on X_1, $|\mathbf{Sur}(X)| = |\mathbf{Sur}(X_1)| \geq |M_1|^{1-1/k} = m^{1-1/k}$, as required.

(4) **Case 4:** $w_{mid} > m/2$ and $q_{hi-R/(2L)} > 0$. In this case, we pick $a, b \in [lo, hi]$ so that $a - b = R/(2L)$ and

$$q_a > q_b^2/m, \tag{5}$$

and also choose $r \in [b, a]$ such that $q_{r-\frac{b-a}{2k}} \leq q_{r+\frac{b-a}{2k}} \cdot \left(\frac{q_b}{q_a}\right)^{1/k}$. In this case, when we cut "backward," we shift the responsibility for the vertices unmarked by the creation of X_1 to Y_1. This is captured by defining $\mathbf{Res}(Y_1) = \hat{M} \setminus M_1$. Since $|M'| = q_{r+\frac{a-b}{2k}}$ and $|\mathbf{Res}(Y_1)| = q_{r-\frac{a-b}{2k}}$, we have

$$|M'| \geq |\mathbf{Res}(Y_1)| \cdot \left(\frac{q_a}{q_b}\right)^{1/k}. \tag{6}$$

By the induction hypothesis on Y_1, we have that

$$
\begin{aligned}
|\mathbf{Sur}(Y_1)| \geq \frac{|M'|}{|M'|^{1/k}} &\overset{(6)}{\geq} |\mathbf{Res}(Y_1)| \cdot \left(\frac{q_a}{|M'| \cdot q_b}\right)^{1/k} \\
&\overset{(5)}{\geq} |\mathbf{Res}(Y_1)| \cdot \left(\frac{q_b}{m \cdot |M'|}\right)^{1/k} \\
&\geq \frac{|\mathbf{Res}(Y_1)|}{m^{1/k}},
\end{aligned}
$$

where in the last inequality, we use that $|M'| = q_{r+(a-b)/(2k)} \leq q_b$. Now, by the induction hypothesis on X_1, we get

$$
\begin{aligned}
|\mathbf{Sur}(X)| &= |\mathbf{Sur}(Y_1)| + |\mathbf{Sur}(X_1)| \\
&\geq \frac{|\mathbf{Res}(Y_1)|}{m^{1/k}} + |M_1|^{1-1/k} \\
&\geq \frac{|\mathbf{Res}(Y_1)| + |M_1|}{m^{1/k}} = \frac{|\hat{M}|}{m^{1/k}} = m^{1-1/k}.
\end{aligned}
$$
□

From Lemma 9 and Lemma 10, we derive the following theorem.

Theorem 6. *Let $G = (V, E)$ be a weighted graph, fix a set $M \subseteq V$ of size m and a parameter $k \geq 1$. There exists a spanning tree T of G, and a set $\mathbf{Sur}(M) \subseteq M$ of size at least $m^{1-1/k}$, such that for every $v \in \mathbf{Sur}(M)$ and every $u \in V$, it holds that $\mathbf{dist}(v, u, T) \leq O(k \log\log m) \cdot \mathbf{dist}(v, u, G)$.*

We conclude with the proof of our main results.

Proof of Theorem 1. The theorem follows directly from Theorem 6 by choosing $M = V$. □

Proof of Theorem 2. Let $M_1 = V$, and for $i \geq 1$, define $M_{i+1} = M_i \setminus \mathbf{Sur}(M_i)$. We shall apply Theorem 6 iteratively, where M_i is the set of vertices given as input to the i-th iteration, which has size $|M_i| = m_i$. Let T_i be the tree created in iteration i. By Theorem 6, the sizes $m_1, m_2, \ldots$

obey the recurrence $m_1 = n$ and $m_{i+1} \leq m_i - m_i^{1-1/k}$, which implies that after $k \cdot n^{1/k}$ iterations, we will have $m_{k \cdot n^{1/k}+1} < 1$ (see Ref. [29, Lemma 4.2]), and thus every vertex is in $\mathbf{Sur}(M_i)$ for some $1 \leq i \leq k \cdot n^{1/k}$. For each $v \in V$, let $\mathbf{home}(v)$ be the tree T_i such that $v \in \mathbf{Sur}(M_i)$. □

4.5 Routing with Short Labels

In this section, we prove Theorem 4. We first use a result of Ref. [35] concerning routing in trees.

Theorem 7 ([35]). *For any tree $T = (V, E)$ (where $|V| = n$), and integer parameter b, there is a routing scheme with stretch 1 that has routing tables of size $O(b)$ and labels of size $(1 + o(1)) \log_b n$. The decision time in each vertex is $O(1)$.*

Combining Theorem 2 and Theorem 7, we can construct a routing scheme. Let $\mathcal{T}$ be the set of trees from Theorem 2. Each tree $T \in \mathcal{T}$ is associated with a routing scheme given by Theorem 7. Set $L_T(x)$ to be the label of the vertex x in the routing scheme of the tree T.

In our scheme, the routing table of each vertex will be the collection of its routing tables in all the trees in $\mathcal{T}$. Hence, the table size is $O(b) \cdot |\mathcal{T}| = O(k \cdot b \cdot n^{1/k})$. The label of each $x \in V$ will be $(\mathbf{home}(x), L_{\mathbf{home}(x)}(x))$, i.e., the name of the home tree of x and the label of x in that tree. The label size is $1 + (1 + o(1)) \log_b n = (1 + o(1)) \log_b n$.

The routing is done in a straightforward manner; to route from y to x, we extract $\mathbf{home}(x)$ from the given label of x, and simply use the routing scheme of the tree $\mathbf{home}(x)$. Note that this process takes $O(1)$ time and is independent of the routing path traversed so far. Since all vertices store in their routing table the appropriate routing information for $\mathbf{home}(x)$, the routing can be completed.

APPENDIX

A PROOF OF CORRECTNESS FOR ALGORITHM 4

In this section, we prove that the choices made in the `create-petal` procedure are all legal. In all the Lemmas that follow, we shall use the notation in Algorithm 4.

Lemma 11. *If $w_{mid} \leq \frac{m}{2}$ and $w_{lo+\frac{R}{2L}} \geq 1$, then there is $[a, b] \subseteq [lo, mid]$ such that $b - a = \frac{R}{2L}$ and $w_a \geq w_b^2/m$.*

Proof. Seeking contradiction, assume that for every such a, b with $b - a = \frac{R}{2L}$ it holds that $w_b > \sqrt{m \cdot w_a}$. Applying this on $b = mid - \frac{iR}{2L}$ and $a = mid - \frac{(i+1)R}{2L}$ for every $i = 0, 1, \ldots, L - 2$, we have that

$$w_{mid} > m^{1/2} \cdot w_{mid-\frac{R}{2L}}^{1/2} > \cdots > m^{1-2^{-(L-1)}} \cdot w_{mid-\frac{(L-1)R}{2L}}^{2^{-(L-1)}} \geq m \cdot 2^{-1} \cdot w_{lo+\frac{R}{2L}}^{1/(2\log m)} \geq \frac{m}{2},$$

where we used that $1 + \log\log m \leq L \leq 2 + \log\log m$ and $mid = lo + R/2$. In the last inequality, we also used that $w_a \geq 1$, which follows since $b = a + \frac{R}{2L} \geq lo + \frac{R}{2L}$; thus, $w_b \geq 1$, and, in particular, $w_a \geq w_b^2/m > 0$. The contradiction follows. □

Lemma 12. *There is $r \in [a, b]$ such that $w_{r+\frac{b-a}{2k}} \leq w_{r-\frac{b-a}{2k}} \cdot \left(\frac{w_b}{w_a}\right)^{\frac{1}{k}}$.*

Proof. Seeking contradiction, assume there is no such choice of r; then, applying this for $r = b - (i + 1/2) \cdot \frac{b-a}{k}$ for $i = 0, 1, \ldots, k - 1$, we get

$$w_b > w_{b-\frac{b-a}{k}} \cdot \left(\frac{w_b}{w_a}\right)^{1/k} > \cdots > w_{b-k\cdot\frac{b-a}{k}} \cdot \left(\frac{w_b}{w_a}\right)^{k/k} = w_a \cdot \frac{w_b}{w_a} = w_b,$$

a contradiction. □

The following two lemmas are symmetric to the two lemmas above.

Lemma 13. *If* $w_{mid} > \frac{m}{2}$ *(implies* $q_{mid} \leq \frac{m}{2}$*) and* $q_{hi-\frac{R}{2L}} \geq 1$*, then there is* $[b, a] \subseteq [mid, hi]$ *such that* $a - b = \frac{R}{2L}$ *and* $q_a \geq q_b^2/m$.

Lemma 14. *There is* $r \in [b, a]$ *such that* $q_{r-\frac{a-b}{2k}} \leq q_{r+\frac{a-b}{2k}} \cdot \left(\frac{q_b}{q_a}\right)^{1/k}$.

ACKNOWLEDGMENTS

We are grateful to Yair Bartal for helpful comments.

REFERENCES

[1] Amir Abboud and Greg Bodwin. 2016. The 4/3 additive spanner exponent is tight. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, June 18–21, 2016*. 351–361. DOI: https://doi.org/10.1145/2897518.2897555

[2] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. 1993. Near-linear cost sequential and distribured constructions of sparse neighborhood covers. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, November 3–5, 1993*. 638–647. DOI: https://doi.org/10.1109/SFCS.1993.366823

[3] Ingo Althöfer, Gautam Das, David P. Dobkin, and Deborah Joseph. 1990. Generating sparse spanners for weighted graphs. In *SWAT*. 26–37.

[4] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. 1995. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.* 24, 1 (1995), 78–100.

[5] Ittai Abraham and Ofer Neiman. 2012. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, May 19–22, 2012*. 395–406. DOI: https://doi.org/10.1145/2213977.2214015

[6] Baruch Awerbuch and David Peleg. 1990. Sparse partitions. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science (FOCS)*. 503–513. DOI: https://doi.org/10.1109/FSCS.1990.89571

[7] Baruch Awerbuch. 1984. An efficient network synchronization protocol. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC'84)*. ACM, New York, NY, 522–525. DOI: https://doi.org/10.1145/800057.808722

[8] Yair Bartal. 2004. Graph decomposition lemmas and their role in metric embedding methods. In *Proceedings of the 12th Annual European Symposium on Algorithms*. 89–97.

[9] Yair Bartal. 2011. Lecture notes in Metric Embedding Theory and Its Algorithmic Applications. Retrieved from http://moodle.cs.huji.ac.il/cs10/file.php/67720/GM_Lecture6.pdf.

[10] Yair Bartal, Béla Bollobás, and Manor Mendel. 2006. Ramsey-type theorems for metric spaces with applications to online problems. *J. Comput. System Sci.* 72, 5 (Aug. 2006), 890–921. Special Issue on FOCS 2001.

[11] Michael A. Bender and Martin Farach-Colton. 2000. The LCA problem revisited. In *LATIN*. 88–94.

[12] J. Bourgain, T. Figiel, and V. Milman. 1986. On Hilbertian subsets of finite metric spaces. *Israel J. Math.* 55, 2 (1986), 147–152. ISJMAP DOI: https://doi.org/10.1007/BF02801990

[13] Guy E. Blelloch, Yan Gu, and Yihan Sun. 2016. A new efficient construction on probabilistic tree embeddings. *CoRR* abs/1605.04651 (2016). http://arxiv.org/abs/1605.04651.

[14] Y. Bartal, N. Linial, M. Mendel, and A. Naor. 2003. On metric Ramsey-type phenomena. In *STOC*. 463–472.

[15] Y. Bartal, N. Linial, M. Mendel, and A. Naor. 2005. Some low distortion metric Ramsey problems. *Discrete Comput. Geom.* 33, 2 (2005), 25–41.

[16] S. Baswana and S. Sen. 2003. A simple linear time algorithm for computing a $(2k-1)$-spanner of $O(n^{1+1/k})$ size in weighted graphs. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (LNCS)*, Vol. 2719. Springer, 384–396.

[17] Shiri Chechik. 2013. Compact routing schemes with improved stretch. In *ACM Symposium on Principles of Distributed Computing, PODC'13, Montreal, QC, Canada, July 22–24, 2013*. 33–41. DOI: https://doi.org/10.1145/2484239.2484268

[18] Shiri Chechik. 2014. Approximate distance oracles with constant query time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC'14)*. ACM, New York, NY, 654–663. DOI: https://doi.org/10.1145/2591796.2591801

[19] Shiri Chechik. 2015. Approximate distance oracles with improved bounds. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, June 14–17, 2015*. 1–10. DOI: https://doi.org/10.1145/2746539.2746562

[20] Gruia Călinescu, Howard J. Karloff, and Yuval Rabani. 2004. Approximation algorithms for the 0-extension problem. *SIAM J. Comput.* 34, 2 (2004), 358–372.

[21] Edith Cohen. 1993. Fast algorithms for constructing t-spanners and paths with stretch t. In *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, November 3–5, 1993*. 648–658. DOI: https://doi.org/10.1109/SFCS.1993.366822

[22] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. 2005. Lower-stretch spanning trees. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*. ACM Press, New York, NY, 494–503. DOI: https://doi.org/10.1145/1060590.1060665

[23] M. Elkin. 2001. Computing almost shortest paths. In *Proceedings of the 20th ACM Symp. on Principles of Distributed Computing*. 53–62.

[24] Michael Elkin and David Peleg. 2004. (1+epsilon, beta)-spanner constructions for general graphs. *SIAM J. Comput.* 33, 3 (2004), 608–631. DOI: https://doi.org/10.1137/S0097539701393384

[25] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. 2003. A tight bound on approximating arbitrary metrics by tree metrics. In *STOC'03: Proceedings of the 35th Annual ACM Symposium on Theory of Computing*. ACM Press, 448–455. DOI: https://doi.org/10.1145/780542.780608

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

[27] Anupam Gupta, R. Ravi, Kunal Talwar, and Seeun William Umboh. 2016. Last but not least: Online spanners for buy-at-bulk. *CoRR* abs/1611.00052 (2016). https://arxiv.org/abs/1611.00052.

[28] Dov Harel and Robert Endre Tarjan. 1984. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.* 13, 2 (1984), 338–355.

[29] Manor Mendel and Assaf Naor. 2007. Ramsey partitions and proximity data structures. *J. Eur. Math. Soc.* 9, 2 (2007), 253–275.

[30] Assaf Naor and Terence Tao. 2012. Scale-oblivious metric fragmentation and the nonlinear Dvoretzky theorem. *Israel J. Math.* 192, 1 (2012), 489–504. DOI: https://doi.org/10.1007/s11856-012-0039-7

[31] D. Peleg and A. Schäffer. 1989. Graph spanners. *J. Graph Theory* 13 (1989), 99–116.

[32] Liam Roditty, Mikkel Thorup, and Uri Zwick. 2005. Deterministic constructions of approximate distance oracles and spanners. In *Proceedings of the 32nd International Conference on Automata, Languages and Programming (ICALP'05)*. Springer-Verlag, Berlin, 261–272. DOI: https://doi.org/10.1007/11523468_22

[33] Paul D. Seymour. 1995. Packing directed circuits fractionally. *Combinatorica* 15, 2 (1995), 281–288. DOI: https://doi.org/10.1007/BF01200760

[34] M. Thorup and U. Zwick. 2001. Approximate distance oracles. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*. Hersonissos, Crete, Greece, 183–192.

[35] M. Thorup and U. Zwick. 2001. Compact routing schemes. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. ACM Press, 1–10.

[36] M. Thorup and U. Zwick. 2006. Spanners and emulators with sublinear distance errors. In *Proceedings of Symp. on Discr. Algorithms*. 802–809.

[37] Christian Wulff-Nilsen. 2013. Approximate distance oracles with improved query time. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'13)*. SIAM.

Received July 2018; revised June 2019; accepted October 2019