Lower-Stretch Spanning Trees

Michael Elkin*
Department of Computer Science
Ben-Gurion University of the Negev

Y. Emek
Weizmann Institute of Science,
Department of Computer Science
and Mathematics

Daniel A. Spielman†
Department of Mathematics
Massachusetts Institute of Technology

Shang-Hua Teng‡
Department of Computer Science
Boston University and
Akamai Technologies Inc.

February 6, 2005

Abstract

In 1991, Alon, Karp, Peleg, and West proved that every weighted connected graph G contains as a subgraph a spanning tree into which the edges of G can be embedded with average stretch $\exp\left(O(\sqrt{\log n \log\log n})\right)$, and that there exists an n-vertex graph G such that all its spanning trees have average stretch $\Omega(\log n)$. Closing the exponential gap between these upper and lower bounds is listed as one of the long-standing open questions in the area of low-distortion embeddings of metrics (Matousek 2002).

We significantly reduce this gap by constructing a spanning tree in G of average stretch $O((\log n \log\log n)^2)$. Moreover, we show that this tree can be constructed in time $O(m \log^2 n)$ in general, and in time $O(m \log n)$ if the input graph is unweighted. The main ingredient in our construction is a novel graph decomposition technique.

Our new algorithm can be immediately used to improve the running time of the recent solver for diagonally dominant linear systems of Spielman and Teng from

$$m 2^{\left(O(\sqrt{\log n \log\log n})\right)} \log(1/\epsilon)$$

to

$$m \log^{O(1)} n \log(1/\epsilon),$$

and to $O(n(\log n \log\log n)^2 \log(1/\epsilon))$ when the system is planar. Applying a recent reduction of Boman, Hendrickson and Vavasis, this provides an $O(n(\log n \log\log n)^2 \log(1/\epsilon))$ time algorithm for solving the linear systems that arise when applying the finite element method to solve two-dimensional elliptic partial differential equations. Our result can also be used to improve several earlier approximation algorithms that use low-stretch spanning trees.

*Part of this work was done in Yale University, and was was supported by the DoD University Research Initiative (URI) administered by the Office of Naval Research under Grant N00014-01-1-0795. The work was also partially supported by the Lynn and William Frankel Center for Computer Sciences.

†Partially supported by NSF grant CCR-0324914. Part of this work was done at Yale University.

‡Partially supported by NSF grants CCR-0311430 and ITR CCR-0325630.

1 Introduction

Let $G = (V, E, w)$ be a weighted connected graph, where w is a function from E into the positive reals. We define the length of each edge $e \in E$ to be the reciprocal of its weight:

$$d(e) = 1/w(e).$$

Given a spanning tree T of V, we define the distance in T between a pair of vertices $u, v \in V$, $\mathrm{dist}_T(u, v)$, to be the sum of the lengths of the edges on the unique path in T between u and v. We can then define the stretch of an edge $(u, v) \in E$ to be

$$\mathrm{stretch}_T(u, v) = \frac{\mathrm{dist}_T(u, v)}{d(u, v)},$$

and the average stretch over all edges of E to be

$$\text{ave-stretch}_T(E) = \frac{1}{|E|} \sum_{(u,v)\in E} \mathrm{stretch}_T(u, v).$$

Alon, Karp, Peleg and West [1] proved that every weighted connected graph $G = (V, E, w)$ of n vertices and m edges contains a spanning tree T such that

$$\text{ave-stretch}_T(E) = \exp\left(O(\sqrt{\log n \log\log n})\right),$$

and that there exists a collection $\tau = \{T_1, \ldots, T_h\}$ of spanning trees of G and a probability distribution Π over τ such that for every edge $e \in E$,

$$\mathbf{E}_{T\leftarrow\Pi}\left[\mathrm{stretch}_T(e)\right] = \exp\left(O(\sqrt{\log n \log\log n})\right).$$

The result of [1] triggered the study of low-distortion embeddings into *probabilistic tree metrics*. Most notable in this context is the work of Bartal [5, 6] that has shown that if the requirement that the trees T be *subgraphs* of G is abandoned, then the upper bound of [1] can be improved by finding a tree whose distances approximate those in the original graph with average distortion $O(\log n \cdot \log\log n)$. On the negative side, a lower bound of $\Omega(\log n)$ is known for both scenarios [1, 5]. The gap left by Bartal was recently closed by Fakcharoenphol, Rao, and Talwar [10], who have shown a tight upper bound of $O(\log n)$.

However, some graph metric approximation applications require trees that are subgraphs. Until now, no progress had been made on reducing the gap between the upper and lower bounds proved by Alon *et al.* [1] on the average stretch of subgraph spanning trees. Moreover, no better understanding of the problem than provided by the results of [1] had been achieved even for *unweighted* graphs, or even for *unweighted planar* graphs.

In this paper, we significantly narrow this gap by improving the upper bound of [1] from $\exp(O(\sqrt{\log n \log\log n}))$ to $O((\log n \cdot \log\log n)^2)$. Specifically, we devise an algorithm that for every weighted connected graph $G = (V, E, w)$ constructs a spanning tree $T \subseteq E$ that satisfies $\text{ave-stretch}_T(E) = O((\log n \log\log n)^2)$. The running time of our algorithm is $O(m \log^2 n)$ for weighted graphs, and $O(m \log n)$ for unweighted. We begin by presenting a simpler algorithm that guarantees a weaker bound, $\text{ave-stretch}_T(E) = O(\log^3 n)$.

Alon *et al.* [1] prove that the existence of a spanning tree with average stretch $f(n)$ for every weighted graph implies the existence of a distribution of spanning trees in which every edge has expected stretch $f(n)$. Consequently, our result implies that for every weighted connected graph $G = (V, E, w)$ there exists a probability distribution Π over a set $\tau = \{T_1, \ldots, T_h\}$ of spanning trees ($T \subseteq E$ for every $T \in \tau$) such that for every $e \in E$, $\max\{\mathbf{E}_{T \leftarrow \Pi}[\mathrm{stretch}_T(e)]\} = O((\log n \log\log n)^2)$. Furthermore, our algorithm itself can be adapted to produce a probability distribution Π that guarantees a slightly weaker bound of $O(\log^3 n)$ in time $O(m \cdot \log^2 n)$. So far, we have not yet been able to verify whether our algorithm can be adapted to produce the bound of $O((\log n \cdot \log\log n)^2)$ within similar time limits.

1.1 Applications

Our new result can be applied to improve algorithms that use low-stretch spanning trees.

1.1.1 Solving Linear Systems

Boman and Hendrickson [7] were the first to realize that low-stretch spanning trees could be used to solve diagonally dominant linear systems. They applied the spanning trees of [1] to design solvers that run in time $m^{3/2} 2^{O(\sqrt{\log n \log\log n})} \log(1/\epsilon)$. Spielman and Teng [19] improved their results by showing how to use the spanning trees of [1] to solve diagonally-dominant linear systems in time

$$m 2^{O(\sqrt{\log n \log\log n})} \log(1/\epsilon).$$

Unfortunately, the trees produced by the algorithms of Bartal [5, 6] and Fakcharoenphol, Rao, and Talwar [10] cannot be used to improve these linear solvers, and it is currently not known whether it is possible to solve linear systems efficiently using trees that are not subgraphs.

By applying the low-stretch spanning trees developed in this paper, we can reduce the time for solving these linear systems to

$$m \log^{O(1)} n \log(1/\epsilon),$$

and to $O(n(\log n \log\log n)^2 \log(1/\epsilon))$ when the systems are planar. Applying a recent reduction of Boman, Hendrickson and Vavasis [8], one obtains a $O(n(\log n \log\log n)^2 \log(1/\epsilon))$ time algorithm for solving the linear systems that arise when applying the finite element method to solve two-dimensional elliptic partial differential equations.

1.1.2 Alon-Karp-Peleg-West Game

Alon, Karp, Peleg and West [1] constructed low-stretch spanning trees to upper-bound the value of a zero-sum two-player game that arose in their analysis of an algorithm for the k-server problem: at each turn, the *tree player* chooses a spanning tree T and the *edge player* chooses an edge $e \in E$, simultaneously. The payoff to the edge player is 0 if $e \in T$ and $\mathrm{stretch}_T(e) + 1$ otherwise. Alon *et al* showed that if every n-vertex weighted connected multigraph G has a spanning T of average stretch $f(n)$, then the value of this game is at most $f(n) + 1$. Our new result lowers the bound on the value of this graph-theoretical game from $\exp\left(O(\sqrt{\log n \log\log n})\right)$ to $O\left((\log n \log\log n)^2\right)$.

1.1.3 Minimum Communication Cost Spanning Tree Problem

Our result can be directly used to improve drastically the approximation ratio of the algorithm of Peleg and Reshef [18] for the *minimum communication cost spanning tree* (henceforth, *MCT*) problem. This problem was introduced in [13], and is listed as [ND7] in [11] and [9].

The instance of this problem is a weighted graph $G = (V, E, w)$, and a matrix $\{r(u, v) \mid u, v \in V\}$ of nonnegative requirements. The goal is to construct a spanning tree T that minimizes $c(T) = \sum_{u,v \in V} r(u, v) \cdot \mathrm{dist}_T(u, v)$.

Peleg and Reshef [18] used the result of [1] to show that the MCT problem admits $2^{O(\sqrt{\log n \cdot \log\log n})}$-approximation ratio. Our result can be used directly to produce an efficient $O((\log n \cdot \log\log n)^2)$-approximation algorithm for this problem.

1.1.4 Message-Passing Model

Embeddings into probabilistic tree metrics have been extremely useful in the context of approximation algorithms (to mention a few: buy-at-bulk network design [3], graph steiner problem [12], covering steiner problem [14]). However, all these algorithms seem to be hardly applicable in the message-passing model of distributed computing (see [17]). In this model, every vertex of the input graph hosts a processor, and the processors communicate over the edges of the graph.

Consequently, in this model executing an algorithm that starts by constructing a non-subgraph spanning tree of the network, and then solves a problem whose instance is this tree is very problematic, since direct communication over the links of this "virtual" tree is impossible. This difficulty disappears if the tree in this scheme is a *subgraph* of the graph. We believe that our result will enable the adaptation of the these approximation algorithms to the message-passing model.

1.2 Our Techniques

We build our low-stretch spanning trees by recursively applying a new graph decomposition that we call a *star-decomposition*. A star-decomposition of a graph is a partition of the vertices into sets that are connected into a star: some central set is connected to each of the others (See Figure 1). We show how to find star-decompositions that do not cut too many edges and such that the radius of the graph induced by the star decomposition is not much larger than the radius of the original graph.

Our algorithm for finding a low-cost star-decomposition applies a generalization of the ball-growing technique of Awerbuch [4] to grow *cones*, where a cone induced by a vertex x_0 is a set C that contains every vertex that has a shortest path to x_0 that goes through C.

1.3 The Structure of the Paper

In Section 2, we define our notation. In Section 3, we introduce the star decomposition of a weighted connected graph. We then show how to use this decomposition to construct a subgraph spanning tree with average stretch $O(\log^3 n)$. In Section 4, we present our star decomposition algorithm. In Section 5, we refine our construction and improve the average stretch to $O\left((\log n \log\log n)^2\right)$. Finally, we conclude the paper in Section 6 and list some open questions. Due to space limitations, most proofs are deferred to the appendix.

2 Notation

For two vertices $u, v \in V$, we define $\text{dist}(u, v)$ to be the length of the shortest path between u and v in E. We write $\text{dist}_G(u, v)$ to emphasize that the distance is in the graph G.

For a set of vertices, $S \subseteq V$, $G(S)$ is the subgraph induced by vertices in S. We write $\text{dist}_S(u, v)$ instead of $\text{dist}_{G(S)}(u, v)$ when G is understood. $E(S)$ is the set of edges with both endpoints in S.

The boundary of a set S, $\partial(S)$ is the set of edges with exactly one endpoint in S.

For a vertex $x \in V$, $\text{dist}_G(x, S)$ is the length of the shortest path between x and a vertex in S.

A multiway partition of V is a collection of pairwise-disjoint sets $\{V_1, \ldots, V_k\}$ such that $\bigcup_i V_i = V$. The boundary of a multiway partition, $\partial(V_1, \ldots, V_k)$, is the set of edges with endpoints in different sets in the partition.

For a set of edges F, $\text{vol}(F)$ is the size of the set F.

For a set of edges F, the cost of F, $\text{cost}(F)$, is the sum of the weights of the edges in F.

The volume of a set S of vertices, $\text{vol}(S)$, is the number of edges with at least one endpoint in S.

The ball of radius r around a vertex v, $B(r, v)$, is the set of vertices of distance at most r from v.

For $v \in V$, $\text{rad}_G(v)$ is the smallest r such that every vertex of G is within distance r from v.

3 Using Star-Decompositions to build Low-Stretch Spanning Trees

In this section, we present our first algorithm for constructing a low-stretch spanning tree of a weighted graph. This algorithm generates a spanning tree with average stretch $O(\log^3 n)$. We will improve the average stretch bound to $O\left((\log n \log\log n)^2\right)$ in Section 5.

We first state the properties of the graph decomposition algorithm at the heart of our construction. We then present the construction and analysis of the low-stretch spanning trees. We defer the description of the graph decomposition algorithm and its analysis to Section 4.

3.1 Low-Cost Star-Decomposition

Definition 3.1 (Star-Decomposition). *A multiway partition $\{V_0, \ldots, V_k\}$ is a* star-decomposition *of a weighed connected multigraph G with center $x_0 \in V$ (see Figure 1) if $x_0 \in V_0$ and*

1. *for all $0 \le i \le k$, the subgraph induced on V_i is connected, and*
2. *for all $i \ge 1$, V_i contains an* anchor *vertex x_i that is connected to a vertex $y_i \in V_0$ by an edge $(x_i, y_i) \in E$. We call the edge (x_i, y_i) the `bridge` between V_0 and V_i.*

Let $r = \text{rad}_G(x_0)$, and $r_i = \text{rad}_{V_i}(x_i)$ for each $0 \le i \le k$. For $\delta, \epsilon \le 1/2$, a star-decomposition $\{V_0, \ldots, V_k\}$ is a (δ, ϵ)-star-decomposition if

a. *$\delta r \le r_0 \le (1 - \delta)r$, and*

b. *$r_0 + d(x_i, y_i) + r_i \le (1 + \epsilon)r$, for each $i \ge 1$.*

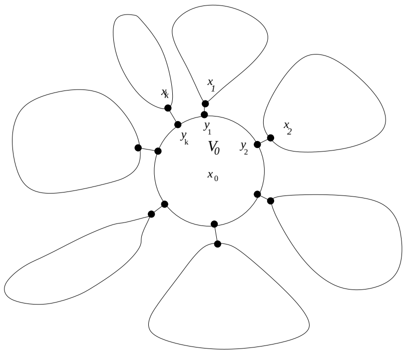

Figure 1: Star Decomposition.

The `cost` *of the star-decomposition is* $cost\left(\partial\left(V_0, \ldots, V_k\right)\right)$.

Note that if $\{V_0, \ldots, V_k\}$ is a (δ, ϵ) star-decomposition of G, then the graph consisting of the union of the induced subgraphs on $V_0, \ldots, V_k$ and the bridge edges (y_k, x_k) has radius at most $(1+\epsilon)$ times the radius of the original graph.

In Section 4, we present an algorithm `starDecomp` that satisfies the following cost guarantee.

Lemma 3.2 (Low-Cost Star Decomposition). *Let $G = (V, E, w)$ be a connected weighted multigraph, and $m = |E|$. Let $x_0 \in V$ be a vertex of G, and let $r = \mathrm{rad}_G(x_0)$. Then*

$$\left(\{V_0, \ldots, V_k\}, x_1, \ldots, x_k, y_1, \ldots, y_k\right) = \texttt{starDecomp}(G, x_0, 1/3, \alpha),$$

in time $O(m \log n)$, returns a $(1/3, \alpha)$-star-decomposition of G with center x_0 of cost

$$cost\left(\partial\left(V_0, \ldots, V_k\right)\right) \leq \frac{4m \log_2 m}{\alpha r}.$$

3.2 A Divide-and-Conquer Algorithm for Low-Stretch Spanning Trees

We begin by showing how to construct low-stretch spanning trees in the case that all edges have length 1. In particular, we will use the fact that in this case the cost of a set of edges equals the number of edges in the set. The formal description of Procedure `simpleLowStretchTree` follows.

Let n be the number of vertices in the original graph, and fix $\alpha = (2 \log_{4/3}(n+6))^{-1}$.

$T = \texttt{simpleLowStretchTree}(G, x_0)$.

1. Set $r = \mathrm{rad}_G(x_0)$.
2. If $r \leq (\log n)^2$,
   Set T to be a shortest-path tree in G from x_0.

   else
   - a. Set $\left(\{V_0, \ldots, V_k\}, x_1, \ldots, x_k, y_1, \ldots, y_k\right) = \texttt{starDecomp}(G, x_0, 1/3, \alpha)$
   - b. For $0 \leq i \leq k$, set $T_i = \texttt{simpleLowStretchTree}(G(V_i), x_i)$
   - c. Set $T = \bigcup_i T_i \cup \bigcup_i (y_i, x_i)$.

Theorem 3.3 (Unweighted). *In time $O(m \log n)$, $T = \texttt{simpleLowStretchTree}(G, x_0)$ returns a spanning tree T satisfying*

$$(1)\ \mathrm{rad}_T(x_0) \leq \sqrt{e} \cdot \mathrm{rad}_G(x_0) \quad \textit{and} \quad (2)\ \textit{ave-stretch}_T(E) \leq (152 + o(1)) \cdot \log^3 m.$$

Proof. For our analysis, we define a family of graphs that converges to T. For a graph G, we let

$$(\{V_0, \ldots, V_k\}, x_1, \ldots, x_k, y_1, \ldots, y_k) = \texttt{starDecomp}(G, x_0, 1/3, \alpha)$$

and recursively define

$$R_1(G) = \bigcup_i (y_i, x_i) \cup \bigcup_i G(V_i), \quad \text{and} \quad R_t(G) = \bigcup_i (y_i, x_i) \cup \bigcup_i R_{t-1}(G(V_i)).$$

For convenience we also define $R_0(G) = G$. The graph $R_t(G)$ is what one would obtain if we forced `simpleLowStretchTree` to return its input graph after t levels of recursion.

Because for all $n \geq 0$, $(2\log_{4/3}(n+6))^{-1} \leq 1/12$, we have $(2/3 + \alpha) \leq 3/4$. Thus, the depth of the recursion in `simpleLowStretchTree` is at most $\log_{4/3} n$, and we have $R_{\log_{4/3} n}(G) = T$. We can prove by induction that, for all graphs G,

$$\mathrm{rad}_{R_t(G)}(x_0) \leq (1+\alpha)^t \mathrm{rad}_G(x_0).$$

Claim (1) now follows from the observation that $(1+\alpha)^{\log_{4/3} n} \leq \sqrt{e}$. To prove Claim (2), we note

$$\begin{aligned}
\sum_{(u,v) \in \partial(V_0, \ldots, V_k)} \mathrm{stretch}_T(u,v) &\leq \sum_{(u,v) \in \partial(V_0, \ldots, V_k)} \mathrm{dist}_T(x_0, u) + \mathrm{dist}_T(x_0, v) \\
&\leq \sum_{(u,v) \in \partial(V_0, \ldots, V_k)} 2\sqrt{e} \cdot \mathrm{rad}_G(x_0), \text{ by Claim (1)} \\
&\leq 2\sqrt{e} \cdot \mathrm{rad}_G(x) \left(\frac{4m \log_2 m}{\alpha \mathrm{rad}_G(x)} \right), \text{ by Lemma 3.2}
\end{aligned}$$

Applying this inequality to all graphs at all $\log_{4/3} n$ levels of the recursion, we obtain

$$\sum_{(u,v) \in E} \mathrm{stretch}_T(u,v) \leq 16\sqrt{e} m \log m \log_{4/3} n \log_{4/3}(n+6) \leq (152 + o(1))\, m \log^3 m \qquad \square$$

We now extend our algorithm and proof to general weighted connected graphs. We begin by pointing out a subtle difference between general and unit-weight graphs. In `simpleLowStretchTree`, we used the facts that $\mathrm{rad}_G(x_0) \leq n$ and that each edge length is 1 to show that the depth of recursion is at most $\log_{4/3} n$. In general weighted graphs, the ratio of $\mathrm{rad}_G(x_0)$ to the length of the shortest edge can be arbitrarily large. Thus, the recursion can be very deep. To compensate, we will contract all edges that are significantly shorter than the radius of their component. In this way, we will guarantee that each edge is only active in a logarithmic number of iterations.

Let $e = (u, v)$ be an edge in $G = (V, E, w)$. The contraction of e results in a new multi-graph by identifying u and v as a new vertex whose neighbors are the union of the neighbors of u and v. The edges between u and v do not appear in the contracted graph. We will refer to u and v as the preimage of the new vertex.

We now state algorithm `lowStretchTree` for general weighted graphs and the main result of this section, Theorem 3.4. We prove Theorem 3.4 by extending the proof of Theorem 3.3. The key observation in establishing Claim (2) is that each edge is active (uncontracted and uncut) in at most $2\log_{4/3} n$ levels of the recursion. The proof can be found in Appendix B.

Theorem 3.4 (Low-Stretch Spanning Tree). *On input any weighted connected graph $G = (V, E, w)$ and $x_0 \in V$, in time $O(m \log^2 n)$,* `lowStretchTree` *outputs a spanning tree of G satisfying*

$$(1)\ \mathrm{rad}_T(x) \leq 2e \cdot \mathrm{rad}_G(x), \quad \text{and} \quad (2)\ \textit{ave-stretch}_T(E) = O(\log^3 n).$$

Fix n to be the number of vertices in the original graph and $\alpha = (2 \log_{4/3}(n+6))^{-1}$.

$T =$ `lowStretchTree`$(G = (V, E), x_0, \alpha)$.

1. Set $r = \mathrm{rad}_G(x_0)$.
2. Let $\bar{G} = (\bar{V}, \bar{E})$ be the multigraph obtained by contracting all edges in G of length less than r/n and discarding self-loops.
3. Set $(\{\bar{V}_0, \ldots, \bar{V}_k\}, \bar{x}_1, \ldots, \bar{x}_k, \bar{y}_1, \ldots, \bar{y}_k) =$ `starDecomp`$(\bar{G}, x_0, 1/3, \alpha)$.
4. Let V_i be the preimage under the contraction of step 2 of vertices in $\bar{V}_i$, and $(x_i, y_i) \in V_0 \times V_i$ be the edge of shortest length for which x_i is a preimage of $\bar{x}_i$ and y_i is a preimage of $\bar{y}_i$.
5. For $0 \leq i \leq k$, set $T_i =$ `lowStretchTree`$(G(V_i), x_i, \alpha)$
6. Set $T = \bigcup_i T_i \cup \bigcup_i (y_i, x_i)$.

4 The star decomposition algorithm

Our star decomposition algorithm exploits two algorithms for growing sets. The first, `ballCut`, is the standard ball growing technique introduced by Awerbuch [4], and was the basis of the algorithm of [1]. The second, `coneCut`, is a generalization of ball growing to cones. So that we can analyze this second algorithm, we abstract the analysis of ball growing from the works of [15, 2, 16]. Instead of nested balls, we will consider *concentric systems*, which we now define.

Definition 4.1 (Concentric System). *For a weighted multigraph $G = (V, E, w)$, a* concentric system *in G is a family of vertex sets $\mathcal{L} = \{L_r \subseteq V : r \in \mathbb{R}^+ \cup \{0\}\}$ such that $L_0 \neq \emptyset$,*

1. *$L_r \subseteq L_{r'}$ for all $r < r'$, and*
2. *if a vertex $u \in L_r$ and (u, v) is an edge in E then $v \in L_{r+d(u,v)}$.*

For example, for any vertex $x \in V$, the set of balls $\{B(r, x)\}$ is a concentric system.

The *radius* of a concentric system $\mathcal{L}$ is $\textit{radius}(\mathcal{L}) = \min\{r : L_r = V\}$. For each vertex $v \in V$, we define $\|v\|_{\mathcal{L}}$ to be the smallest r such that $v \in L_r$.

Lemma 4.2 (Concentric System Cutting). *Let $G = (V, E)$ be a connected weighted multigraph, and $m = |E|$, and let $\mathcal{L} = \{L_r\}$ be a concentric system. For all ℓ such that $\ell < \textit{radius}(\mathcal{L})$, there exists an $r \in [\ell/2, \ell]$ such that*

$$\textit{cost}(\partial(L_r)) \leq \frac{2 \mathrm{vol}(L_r) \log_2 \left(m / \mathrm{vol}\left(E(L_{\ell/2})\right)\right)}{\ell},$$

where $m = |E|$. If $\mathrm{vol}\left(E(L_{\ell/2})\right) = 0$, then

$$\textit{cost}(\partial(L_r)) \leq \frac{2(\mathrm{vol}(L_r) + 1) \log_2(m)}{\ell}.$$

We defer the proof of Lemma 4.2 to Appendix D. An analysis of the following standard ball growing algorithm follows immediately by applying Lemma 4.2 to the concentric system $\{B(r, x)\}$.

$r_0 = \texttt{ballCut}(G, x_0, r)$

1. Set $r_0 = r/3$.
2. While $\text{cost}\,(\partial\,(B(r_0, x_0))) > (2(\text{vol}\,(B(r_0, x_0)) + 1)\log_2 |E|)/r$,
   a. Find the vertex $v \notin B(r_0, x_0)$ that minimizes $\text{dist}(x_0, v)$ and set $r_0 = \text{dist}(x_0, v)$.

Corollary 4.3 (Weighted Ball Cutting). *Let $G = (V, E, w)$ be a connected weighted multigraph, let $x \in V$, $r = \text{rad}_G\,(x)$, $r_0 = \texttt{ballCut}(G, x_0, r)$, and $V_0 = B(r_0, x)$. Then $r/3 \le r_0 \le 2r/3$ and*

$$\textit{cost}\,(\partial\,(V_0)) \le 2(\text{vol}\,(V_0) + 1)\log_2 |E|\,/r.$$

We now examine the concentric system that enables us to construct $V_1, \dots, V_k$ in Lemma 3.2.

Definition 4.4 (Ideals and Cones). *For a weighted multigraph $G = (V, E, w)$ and $S \subseteq V$. The set of* forward edges induced by S *is*

$$F = \{(u \to v) : (u, v) \in E, \textit{dist}(u, S) + d(u, v) = \textit{dist}(v, S)\}. \tag{1}$$

For $W \subseteq V$, the ideal of W induced by S, *denoted $I_S(W)$, is the set of vertices reachable from W by directed edges in F, including W itself.*

For each set $W \subset V$ the cone of width l around W induced by S, *$C_S(l, W)$, is the set of vertices in V that can be reached from W by a path, the sum of the lengths of whose edges e that do not belong to F is at most l. Clearly, $C_S(0, W) = I_S(W)$ for all $W \subset V$.*

That is, $I_S(W)$ is the set of vertices that have shortest paths to S that intersect W. Also, $u \in C_S(l, W)$ if there exist $a_0, \dots, a_{k-1}$ and $b_1, \dots, b_k$ such that $a_0 \in W$, $b_k = u$, $b_{i+1} \in I_s(a_i)$, $(b_i, a_i) \in E$, and

$$\sum_i d(b_i, a_i) \le l.$$

The cones $\{C_S(l, W)\}$ form a concentric system (Proposition C.1) with many useful properties. The proof of Proposition 4.5 appears in Appendix C.

Proposition 4.5 (Radius of Cones, II). *Let $G = (V, E, w)$ be a connected weighted multigraph. Let $x_0 \in V$, $r = \text{rad}_G\,(x_0)$, $r_0 < r$ and $V_0 = B(r_0, x_0)$. Let $V' = V - V_0$, $S \subseteq V'$ be all neighbors of V_0 in V', $x_1 \in V'$ and $\rho = r - \textit{dist}_G(x_0, x_1)$. The cones $C_S(l, x_1)$ in the graph $G(V')$ satisfy*

$$\text{rad}_{C_S(l, x_1)}\,(x_1) \le \rho + 2l.$$

Proposition 4.6 (Deleting Cones). *Let $G' = (V', E', w)$ be a connected weighted multigraph, and let $S \subseteq V'$. Let $\rho = \max_{v \in V'} \textit{dist}(v, S)$. For any $x_1 \in S$ and any l, let $V'' = V' - C_S(l, x_1)$ and $S' = S - C_S(l, x_1)$. Then,*

$$\max_{v \in V''} \textit{dist}_{V''}(S', v) \le \rho.$$

Proof. Let $v \in V''$. If the shortest path from v to S intersects $C_S(l, x_1)$, then $v \in C_S(l, x_1)$. So, the shortest path from v to S in V' must lie entirely in V''. $\square$

$r_0 = \texttt{coneCut}(G, W, \delta, S)$

1. Set $l = \delta/2$ and set $\eta = \log_2(m/|E(C_S(\delta/2, W)|)$.
2. While $\mathrm{cost}\left(\partial\left(C_S(l, W)\right)\right) > 2\mathrm{vol}\left(C_S(l, W)\right)\eta/\delta$,
   Find the vertex $v \notin C_S(l, W)$ minimizing $\mathrm{dist}(v, C_S(l, W))$ and set $l = l + \mathrm{dist}(v, C_S(l, W))$.

Corollary 4.7 (Cone Cutting). *Let $G = (V, E, w)$ be a connected weighted multigraph, and let $S \subseteq V$. For $W \subset V$, let r be the least number such that $C_S(r, W) = V$. For any $\delta < r$,* `coneCut`(G, W, δ) *returns a $l \in [\delta/2, \delta]$ such that*

$$\mathit{cost}\left(\partial\left(C_S(l, W)\right)\right) \leq 2\mathrm{vol}\left(C_S(l, W)\right)\log_2\left(m/\mathrm{vol}\left(E(C_S(\ell/2, W))\right)\right)/\ell,$$

where $m = |E|$. If $\mathrm{vol}\left(E(C_S(\ell/2, W))\right) = 0$, then

$$\mathit{cost}\left(\partial\left(C_S(l, W)\right)\right) \leq 2(\mathrm{vol}\left(C_S(l, W)\right) + 1)\log_2\left(m\right)/\ell.$$

The basic idea of `starDecomp` is to first use `ballCut` to construct V_0 and then repeatedly apply `coneCut` to construct $V_1, \ldots, V_k$. We now define algorithm `starDecomp`. Lemma 3.2 follows from Corollaries 4.3 and 4.7 and two other basic properties of the cones $C_S(l, W)$: that we can bound their radius (Proposition C.2), and that their removal does not increase the radius of the graph (Proposition 4.6). See Appendix A for the proof of Lemma 3.2.

$(\{V_0, \ldots, V_k, x_1, \ldots, x_k, y_1, \ldots, y_k\}) = \texttt{starDecomp}(G, x_0, \delta, \alpha)$

1. Set $r = \mathrm{rad}_G(x_0)$; Set $r_0 = \texttt{ballCut}(G, x_0, 3\delta r)$ and $V_0 = B(r_0, x_0)$;
2. Set S be the set of vertices s in $V - V_0$ such that there is an $u \in V_0$ for which there is an edge (s, u) that lies on a shortest path from s to x_0.
3. Set $G' = (V', E', w') = G(V - V_0)$, the weighted graph induced by $V - V_0$;
4. Set $(\{V_1, \ldots, V_k, x_1, \ldots, x_k\}) = \texttt{coneDecomp}(G', S, \alpha r/2)$;
5. for each $i \in [1 : k]$, set y_k to be a vertex in V_0 such that $(x_k, y_k) \in E$ and y_k is on a shortest path from x_0 to x_k.

$(\{V_1, \ldots, V_k, x_1, \ldots, x_k\}) = \texttt{coneDecomp}(G, S, \Delta)$

1. Set $G_0 = G$, $S_0 = S$, and $k = 0$.
2. while S_k is not empty
   a. Set $k = k + 1$; Set x_k to be a vertex of S_{k-1}; Set $r_k = \texttt{coneCut}(G_{k-1}, \{x_k\}, \Delta, S_{k-1})$
   b. Set $V_k = C_{S_{k-1}}(r_k, \{x_k\})$; Set $G_k = G(V - \cup_{i=1}^k V_k)$ and $S_k = S_{k-1} - V_k$.

5 Improving the Stretch

In this section, we will improve the average stretch to $O\left((\log n \log\log n)^2\right)$ by introducing a new procedure `impConeDecomp` which refines `coneDecomp`. This new cone decomposition trades off the volume of the cone against the cost of edges on its boundary. Our refined star decomposition algorithm `impStarDecomp` is identical to algorithm `starDecomp`, except that it calls

$$(\{V_1, \ldots, V_k, x_1, \ldots, x_k\}) = \texttt{impConeDecomp}(G', S, \alpha r/2, t, \bar{m})$$

at Step 4. Due to space limitations, we will only state the algorithm and a lemma about its performance. The proof of the lemma can be found in Section E.

$(\{V_1,\ldots,V_k,x_1,\ldots,x_k\}) = \texttt{impConeDecomp}(G,S,\Delta,t,\bar{m})$

1. Set $G_0 = G$, $S_0 = S$, and $k = 0$.
2. while S_k is not empty
   a. Set $k = k+1$, set x_k to be a vertex of S_{k-1}, set $V'_k = C_{S_{k-1}}(0,x_k)$ and set $p = t-1$;
   b. while $p \geq 0$
      i. Set $r_k = \texttt{coneCut}(G_{k-1},V'_k,\Delta/t,S_{k-1})$ and set $V'_k = C_{S_{k-1}}(r_k,V'_k)$.
      ii. **if** $\mathrm{vol}\,(E(V'_k)) \leq \frac{m}{2^{\log^{p/t}\bar{m}}}$ **then exit** the loop; **else** $p = p-1$;
   c. Set $V_k = V'_k$ and set $G_k = G(V - \cup_{i=1}^{k} V_k)$ and $S_k = S_{k-1} - V_k$.

Lemma 5.1 (Improved Low-Cost Star Decomposition). *Let G, x_0, r and α be as in Lemma 3.2, t be a positive integer control parameter, and $\bar{m} \geq m = |E|$. Let*

$$(\{V_0,\ldots,V_k\},x_1,\ldots,x_k,y_1,\ldots,y_k) = \texttt{impStarDecomp}(G,x_0,1/3,\alpha,t,\bar{m}).$$

Then, $V_0,\ldots,V_k$ is a $(1/3,\alpha)$-star-decomposition of G with center x_0 that satisfies

$$cost\,(\partial\,(V_0)) = 2(\mathrm{vol}\,(V_0)+1))\log_2 m/r,$$

and for every index $j \in \{1,2,\ldots,k\}$ there exists $p = p(j) \in \{0,1,\ldots,t-1\}$ such that

$$\mathrm{vol}\,(E(V_j)) \;\leq\; \frac{m}{2^{\log^{p/t}\bar{m}}} \quad \text{and} \quad cost\left(\partial\left(V_j, V - \cup_{i=0}^{j} V_i\right)\right) \;\leq\; t\cdot\frac{4\mathrm{vol}\,(V_j)\log^{(p+1)/t}\bar{m}}{\alpha r}. \qquad (2)$$

Our improved algorithm $\texttt{impLowStretchTree}(G,x_0,t,\alpha,\bar{m})$, is identical to $\texttt{lowStretchTree}$ except that in Step 3 it calls "$\texttt{impStarDecomp}(\bar{G},x_0,1/3,\alpha,t,\bar{m})$," and Step 5 it calls "$\texttt{impLowStretchTree}(G(V_i),x_i,t,\alpha,\bar{m})$". We set $t = \log\log n$, and $\bar{m} = |E|$ throughout the execution of the algorithm. The proof of the following theorem is deferred to Section F.

Theorem 5.2 (Lower–Stretch Spanning Tree). *For every weighted connected graph $G = (V,E,w)$ on n vertices and m edges, algorithm impLowStretchTree, in $O(m\log^2 n)$ time, constructs a spanning tree T satisfying $\mathrm{rad}_T(x) \leq 2e\cdot\mathrm{rad}_G(x)$, and $stretch_T(E) = O\left((\log n\log\log n)^2\right)$.*

6 Open Questions

A natural open question is whether one can improve the stretch bound from $O\left((\log n\log\log n)^2\right)$ to $O(\log n)$. Algorithmically, it is also desirable to improve the running time of the algorithm to $O(m\log n)$. If we can successfully achieve both improvements, then we can use the Spielman-Teng solver to solve planar diagonally dominant linear systems in $O(n\log n\log(1/\epsilon))$ time.

As the average stretch of any spanning tree in a weighted connected graph is $\Omega(1)$, our low-stretch tree algorithm also provides an $O\left((\log n\log\log n)^2\right)$-approximation to the optimization problem of finding the spanning tree with the lowest average stretch. It remains open (1) whether our algorithm has a better approximation ratio and (b) whether one can in polynomial time find a spanning tree with better approximation ratio, e.g., $O(\log n)$ or even $O(1)$.

References

[1] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, February 1995. Also available Technical Report TR-91-066, ICSI, Berkeley 1991.

[2] Yonatan Aumann and Yuval Rabani. An o(log k) approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, 1998.

[3] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Proceedings of the Symp. on Foundations on Theory of Computing*, pages 542–547, 1997.

[4] Baruch Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, 1985.

[5] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, page 184. IEEE Computer Society, 1996.

[6] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 161–168, 1998.

[7] Erik Boman and Bruce Hendrickson. On spanning tree preconditioners. Manuscript, Sandia National Lab., 2001.

[8] Erik Boman, Bruce Hendrickson, and Stephen Vavasis. Solving elliptic finite element systems in near-linear time with support preconditioners. Manuscript, Sandia National Lab. and Cornell, http://arXiv.org/abs/cs/0407022.

[9] P. Crescenzi and V. Kann. A compendium of NP-hard problems. Available online at http://www.nada.kth.se/theory/compendium, 1998.

[10] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455, 2003.

[11] M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to Theory of NP-Completeness*. 1979.

[12] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 253–259, 1998.

[13] T.C. Hu. Optimum communication spanning trees. *SIAM Journal on Computing*, pages 188–195, 1974.

[14] G. Konjevod and R. Ravi. An approximation algorithm for the covering steiner problem. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 338–344, 2000.

[15] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[16] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

[17] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. 2000. SIAM Philadelphia PA.

[18] D. Peleg and E. Reshef. Deterministic polylogarithmic approximation for minimum communication spanning trees. In *Proc. 25th International Colloq. on Automata, Languages and Programming*, pages 670–681, 1998.

[19] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 81–90, 2004.

A Proof of Lemma 3.2

Proof of Lemma 3.2. The basic idea of `starDecomp` is to first use `ballCut` to construct V_0 and then repeatedly apply `coneCut` to construct $V_1, \ldots, V_k$.

By setting $\delta = 1/3$ as in the Lemma, we have $r/3 \leq r_0 \leq (2/3)r$. Applying $\Delta = \alpha r/2$ and Propositions C.2 and 4.6, we can bound for every i, $r_0 + d(x_i, y_i) + r_i \leq r + 2\Delta \leq r + \alpha r$. Thus $\mathtt{starDecomp}(\mathtt{G}, \mathtt{x_0}, \mathtt{1/3}, \alpha)$ returns a $(1/3, \alpha)$-star-decomposition with center x_0.

To bound the cost of the star-decomposition that the algorithm produces, we use Corollaries 4.3 and 4.7.

$$
\begin{aligned}
\operatorname{cost}\left(\partial\left(V_0\right)\right) &= \frac{2(Vol(V_0+1))\log_2 m}{r} \\
\operatorname{cost}\left(\partial\left(V_1, \ldots, V_k\right)\right) &\leq \sum_{i=1}^{k} \operatorname{cost}\left(\partial\left(V_i\right)\right) \leq \frac{4\sum_i^k \operatorname{vol}\left(V_i\right)\log_2 m}{\alpha r}.
\end{aligned}
$$

Thus,

$$
\operatorname{cost}\left(\partial\left(V_0, \ldots, V_k\right)\right) \leq \sum_{i=0}^{k} \operatorname{cost}\left(\partial\left(V_i\right)\right) \leq \frac{4\left|E\right|\log_2 m}{\alpha r}.
$$

We can use priority queue in the implementation of `ballCut` and `coneCut` in finding the next vertex in $O(\log n)$ time. We thus can implement `starDecomp` in $O(m\log^2 n)$ time. ☐

B Proof of Theorem 3.4

Proof of Theorem 3.4. We first establish notation similar to that used in the proof of Theorem 3.3. Our first step is to define a procedure, `SD`, that captures the action of the algorithm in steps 1 through 4. We then define $R_0(G) = G$ and

$$
R_t(G) = \bigcup_i (y_i, x_i) \cup \bigcup_i R_{t-1}(G(V_i)),
$$

where

$$(\{V_0, \ldots, V_k\}, x_1, \ldots, x_k, y_1, \ldots, y_k) = \mathtt{SD}(G, x_0, 1/3, \alpha).$$

We now prove part (1). Let $t = 2\log_{4/3} n$, and let $r_t = \mathrm{rad}_{R_t(G)}(x_0)$. Consider the graph $R_t(G)$. Following the proof of Theorem 3.3, we can show that r_t is at most $e \cdot \mathrm{rad}_G(x_0)$, plus the sum of the lengths of the edges that were not considered at some point in the construction, but which eventually wind up in T. As these edges all have length at most $\mathrm{rad}_G(x_0)/n$, and there are less than n of them, we find

$$r_t \le e \cdot \mathrm{rad}_G(x_0) + \mathrm{rad}_G(x_0) = (e+1)\mathrm{rad}_G(x_0).$$

We can also know that each component of G that remains after $2\log_{4/3} n$ levels of the recursion has radius at most $r(3/4)^{2\log_{4/3} n} \le r/n^2$. We may also assume by induction that, for the graph induced on each of these components, `lowStretchTree` outputs a tree of radius at most $2e(r/n^2)$. As there are at most n of these components, we know that the tree returned by the algorithm has radius at most

$$(e+1) \cdot r + 2e(r(n-1)/n^2) \le 2er,$$

for $n \ge 2$.

We now turn to part (2), the bound on the stretch. In this part, we let $E_t \subseteq E$ denote the set of edges that are active at depth t in the recursion. That is, their endpoints are not identified by the contraction of short edges in step 2, and their endpoints remain in the same component. We now observe that no edge can appear in more than $\lceil \log_{4/3}(2n+1) \rceil$ components. To see this, consider some edge and let t be the first index for which the edge is in E_t. Let r be the radius of the component in which the edge lies at that time. As the endpoints of the edge are not identified under contraction, they are at distance at least r/n from each other. At time $t + \lceil \log_{4/3}(2n+1) \rceil$, each sub-cluster of this cluster will have radius less than $r/2n$, and so the endpoints of the edge must be in different components at that time.

As in the proof of Theorem 3.3, we see that the total contribution to the stretch at depth t is at most

$$O(\mathrm{vol}(E_t) \log^2 m).$$

Therefore, the sum of the stretches over all recursion depths is

$$O\left(\sum_t \mathrm{vol}(E_t) \log^2 m\right) = O\left(\mathrm{vol}(E) \log^3 m\right).$$

We now analyze the running time of the algorithm. Note that an edge contributes to the complexity of an algorithm only when its two endpoints are not identified in contraction and they belong to the sample component. We have already shown above that each edge can appear in at most $\lceil \log_{4/3}(2n+1) \rceil$ components. Thus the complexity of the algorithm is at most $O(m \log n \lceil \log_{4/3}(2n+1) \rceil) = O(m \log^2 n)$. Note that for unweighted graphs G, the running time is smaller by a factor $\log n$, that is, it is $O(m \log n)$.

$\square$

C Properties of Cones

We now establish that these cones form concentric systems.

Proposition C.1 (Cones are concentric). *Let $G = (V, E, w)$ be a weighted multigraph and let $S \subseteq V$. Then for all $W \subset V$, $\{C_S(l, W)\}_l$ is a concentric system in G.*

Proof. Clearly, $C_S(l, W) \subseteq C_S(l', W)$ if $l < l'$. Moreover, suppose $u \in C_S(l, W)$ and $(u, v) \in E$. Then if $(u \to v) \in F$, then $v \in C_S(l, W)$ as well. Otherwise, the path witnessing that $u \in C_S(l, W)$ followed by the edge (u, v) to v is a witness that $v \in C_S(l + d(u, v), W)$. □

We now show two other properties of the cones $C_S(l, W)$: that we can bound their radius (Proposition C.2), and that their removal does not increase the radius of the graph (Proposition 4.6).

Proposition C.2 (Radius of Cones). *Let $G' = (V', E', w)$ be a connected weighted multigraph, and let $S \subseteq V'$. Let $\rho = \max_{v \in V'} dist(v, S)$. Then, for every $x_1 \in S$,*

$$\mathrm{rad}_{C_S(l,x_1)}(x_1) \leq \rho + 2l.$$

Proof. Let u be a vertex in $C_S(l, x_1)$, and let $a_0, \ldots, a_{k-1}$ and $b_1, \ldots, b_k$ be vertices such that $a_0 = x_1$, $b_k = u$, $b_{i+1} \in I_s(a_i)$, $(b_i, a_i) \in E$, and

$$\sum_i d(b_i, a_i) \leq l.$$

These vertices provide a path connecting x_1 to u inside $C_S(l, x_1)$ of length most

$$\sum_i d(b_i, a_i) + \sum_i \mathrm{dist}(b_i, a_{i+1}).$$

As the first term is at most l, we just need to bound the second term by $\rho + l$. To do this, consider the distance of each of these vertices from S. We have the relations

$$\begin{aligned} \mathrm{dist}(S, b_{i+1}) &= \mathrm{dist}(S, a_i) + \mathrm{dist}(a_i, b_{i+1}) \\ \mathrm{dist}(S, a_i) &\geq \mathrm{dist}(S, b_i) - d(b_i, a_i), \end{aligned}$$

which imply that

$$\rho \geq \mathrm{dist}(S, b_k) \geq \sum_i \mathrm{dist}(a_i, b_{i+1}) - d(b_i, a_i) \geq \left(\sum_i \mathrm{dist}(a_i, b_{i+1}) \right) - l,$$

as desired. □

In our proof, we actually use the following slight extension of Proposition C.2. It's proof is similar. The proof of Proposition 4.5 follows the proof of Proposition C.2, replacing distances from S with distances from x_0.

D Proof of Lemma 4.2

Proof of Lemma 4.2. Note that rescaling terms does not effect the statement of the lemma. For example, if all the weights are doubled, then the costs are doubled but the distances are halved. We note that if $\mathrm{vol}\left(E(L_{\ell/2})\right) = 0$, we can reduce the lemma to the case $\mathrm{vol}\left(E(L_{\ell/2})\right) = 1$ by adding a single edge of zero length to a vertex in $L_{\ell/2}$. Hereafter, we assume $\mathrm{vol}\left(E(L_{\ell/2})\right) \geq 1$.

Let $v_1, \dots, v_n$ be the vertices in V in order of increasing norm. Let $r_i = \|v_i\|$, so that $r_1 \leq r_2 \leq \dots \leq r_n$. We may now assume without loss of generality that each edge in the graph has minimal length. That is, an edge from vertex i to vertex j has length $|r_i - r_j|$. The reason we may make this assumption is that it only increases the costs of edges, making our lemma strictly more difficult to prove. (Recall that the cost of an edge is the reciprocal of its length.)

Let $B_i = L_{r_i}$. Our proof will make critical use of a quantity μ_i, which is defined to be

$$\mu_i \;=\; |E(B_i)| + \sum_{(v_j,v_k)\in E: j\leq i<k} \frac{r_i - r_j}{r_k - r_j}.$$

That is, μ_i sums the edges inside B_i, proportionally counting edges that are split by the boundary of the ball. The two properties of μ_i that we exploit are

$$\mu_{i+1} = \mu_i + \mathrm{cost}\left(\partial\left(B_i\right)\right)\left(r_{i+1} - r_i\right), \tag{3}$$

and

$$\mathrm{vol}\left(E(B_i)\right) \leq \mu_i \leq \mathrm{vol}\left(B_i\right). \tag{4}$$

The equality (3) follows from the definition by a straight-forward calculation, as $|E(B_{i+1})| - |E(B_i)| = |\{(v_{i+1}, v_l) : l \in \{1, 2, \dots, i\}|$, and

$$\sum_{(v_j,v_k)\in E: j\leq i<k} \frac{r_i - r_j}{r_k - r_j} - \sum_{(v_j,v_k)\in E: j\leq i+1<k} \frac{r_{i+1} - r_j}{r_k - r_j} \;=\;$$

$$(r_{i+1} - r_i) \cdot \sum_{e=(v_j,v_k)\in E: j\leq i, i+1<k} 1/d(e) - \sum_{j\leq i, e=(v_j,v_{i+1})\in E} \frac{r_i - r_j}{d(e)}\;.$$

Choose a and b so that $r_{a-1} < \ell/2 \leq r_a$ and $r_b < \ell \leq r_{b+1}$. We first consider the trivial case in which $b < a$. In that case, there is no vertex whose distance from v_0 is between $\ell/2$ and ℓ. In that case, every edge crossing $L_{2\ell/3}$ has length at least $\ell/2$, and therefore cost at most $2/\ell$. In this case, setting $r = 2\ell/3$ we obtain

$$\mathrm{cost}\left(\partial\left(L_r\right)\right) \leq \left|\partial\left(L_r\right)\right| \frac{2}{\ell} \leq \mathrm{vol}\left(L_r\right) \frac{2}{\ell},$$

establishing the lemma in this case.

We now define

$$\eta = \log_2\left(m/\mathrm{vol}\left(E(L_{a-1})\right)\right).$$

A similarly trivial case is when $[a, b]$ is non-empty, and where there exists an $i \in [a, b]$ such that

$$r_i - r_{i-1} > \frac{\ell}{2\eta}.$$

In this case, every edge in $\partial(B_{i-1})$ has cost at most $2\eta/\ell$, and choosing r to be just slightly smaller than r_i we satisfy

$$\operatorname{cost}(\partial(L_r)) \leq |\partial(L_r)| \frac{2\eta}{\ell} \leq \operatorname{vol}(L_r) \frac{2\eta}{\ell}.$$

In the remaining case that the set $[a,b]$ is non-empty and for all $i \in [a,b]$

$$r_i - r_{i-1} \leq \frac{\ell}{2\eta}, \tag{5}$$

will prove that there exists an $i \in [a,b]$ such that

$$\operatorname{cost}(\partial(B_i)) \leq \frac{2\mu_i \eta}{\ell} = \mu_i 2\eta/\ell,$$

which suffices by (4) and the fact that $L_{\ell/2} = L_{a-1}$.

Assume by way of contradiction that

$$\operatorname{cost}(\partial(B_i)) > \mu_i 2\eta/\ell$$

for all $i \in [a,b]$. We then have

$$\mu_i \geq \mu_{i-1} + \mu_{i-1}(r_i - r_{i-1})2\eta/\ell,$$

which implies

$$\begin{aligned}
\mu_{b+1} &\geq \mu_{a-1} \prod_{i=a}^{b+1} (1 + (r_i - r_{i-1})2\eta/\ell) \\
&\geq \mu_{a-1} \prod_{i=a}^{b+1} (m/\operatorname{vol}(E(L_{a-1})))^{2(r_i - r_{i-1})/\ell}, \text{ by (5) and Proposition D.1} \\
&= \mu_{a-1} \cdot (m/\operatorname{vol}(E(L_{a-1})))^{2(r_{b+1} - r_{a-1})/\ell} \\
&> \mu_{a-1} \cdot (m/\operatorname{vol}(E(L_{a-1}))) \\
&\geq m, \text{ by (4)},
\end{aligned}$$

which is a contradiction. □

Proposition D.1 (Weighed log trick). *If* $\gamma \log_2(x) \leq 1$, *then*

$$1 + \gamma \log_2 x \geq x^\gamma.$$

Proof. Let $y = \gamma \log_2 x$. Then, it suffices to observe that

$$\ln(1+y) \geq \ln(2)y,$$

which holds for all $y \leq 1$. □

E Proof of Lemma 5.1

Proof of Lemma 5.1. We will call $p(\cdot)$ the *index-mapping* of the star-decomposition. For convenience, we set $p(0)$ to -1.

We begin our proof by observing that the set V_k returned by `impConeDecomp` in step 2 has the form $C(l, x_k)$ for some $l \leq \Delta = \alpha r/2$. We can then show that $\{V_0, \ldots, V_k\}$ is a $(1/3, \alpha)$-star decomposition as we did in the proof of Lemma 3.2.

We now bound the cost of the decomposition. Clearly, the bound on $\mathrm{cost}\left(\partial\left(V_0\right)\right)$ remains unchanged from that proved in Lemma 3.2. Below we will use $\Delta = \alpha r/2$ as specified in algorithm `impStarDecomp`.

Fix an index $j \in \{1, 2, \ldots, k\}$, and let $p = p(j)$ be the final value of variable p in the loop above (that is, the value of p when the execution left the loop while constructing V_j). Observe that $p \in \{0, 1, \ldots, t-1\}$, and that $\mathrm{vol}\left(E(V_j)\right) \leq \frac{m}{2^{\log^{p/t} \bar{m}}}$ (in the only case when the loop is aborted due to the condition $p < 0$, this inequality, obviously, holds as well).

For the inequality (2), we split the discussion to two cases. First, $p = t-1$. Then the inequality $\mathrm{cost}\left(\partial\left(V_j, V - S - V_j\right)\right) \leq \frac{2\mathrm{vol}(V_j)\log m}{\delta} = t \cdot \frac{2\mathrm{vol}(V_j)\log m}{\Delta}$ follows directly from Corollary 4.7.

Second, $p < t-1$. In this case observe that in the beginning of the pth iteration of the loop, the set V_j' satisfies $\mathrm{vol}\left(E(V_j')\right) > \frac{m}{2^{\log^{(p+1)/t} \bar{m}}}$ (as otherwise the loop would have been aborted in the previous iteration). Observe also that in the beginning of this iteration the set V_j' is of the form $C(l_{j,t}, \ldots, C(l_{j,p+1}, C(l_{j,p+1}, s_j)) \ldots) = C((\sum_{h=p+1}^{t} l_{j,h}), s_j)$. Denote $l' = \sum_{h=p+1}^{t} l_{j,h}$. Consequently, Corollary 4.7 is applicable with $W = C(l', s_j)$, and it provides us with a number $l_{j,p} \in [\Delta/2t, \Delta/t]$ such that

$$\mathrm{cost}\left(\partial\left(V_j, V - S - V_j\right)\right) \;\leq\; \frac{2\mathrm{vol}\left(V_j, V - S - V_j\right)}{(\Delta/t)} \log\left(\frac{m}{\mathrm{vol}\left(E(C(l' + \frac{l_{j,p}}{2}, s_j))\right)}\right).$$

As $\mathrm{vol}\left(E(C(l' + \frac{l_{j,p}}{2}, s_j))\right) \;\geq\; \mathrm{vol}\left(E(V_j')\right) \;=\; \mathrm{vol}\left(E(C(l', s_j))\right) > \frac{m}{2^{\log^{(p+1)/t} \bar{m}}}$, it follows that

$$\frac{m}{\mathrm{vol}\left(E(C(l' + \frac{l_{j,p}}{2}, s_j))\right)} < 2^{\log^{(p+1)/t} \bar{m}} ,$$

and, consequently, the logarithm of the left-hand side is at most $\log^{(p+1)/t} \bar{m}$, proving the lemma. □

F Proof of Theorem 5.2

Proof of Theorem 5.2. Let $\{V_0, V_1, \ldots, V_k\}$ be the star-decomposition returned by an invocation of algorithm `ImpStarDecomp` on V. For each index $i \in \{0, 1, \ldots, t-1\}$, let $I_i = \{j \in \{1, 2, \ldots, k\} \mid p(j) = i\}$.

For a subset $U \subseteq V$ of vertices, let $AS(U)$ denote the average stretch that the algorithm guarantees for the edges of $E(U)$. Then by Lemma 5.1, the following recursive formula applies.

$$AS(V_0) = \left(\sum_{j=0}^{k} AS(V_j)\right) + \frac{1}{|E|}\left(3\log m \cdot \mathrm{vol}\,(V_0) + 3(t/\alpha)\sum_{p=0}^{t-1} \log^{(p+1)/t} \bar{m} \sum_{j \in I_p} \mathrm{vol}\,(V_j)\right) . \quad (6)$$

Fix an edge $e \in E$, and let $\sigma(e)$ be a sequence of non-negative integer indices, determined in the following way. If $e \in E(V_j)$ for some $j = j_1 \in \{0, 1, \ldots, k = k_1\}$, then $\sigma_1(e) = j_1$. Otherwise, the sequence is empty. Suppose that for some $h \in \{1, 2, \ldots\}$, the first h elements of the sequence $\sigma_1(e), \ldots, \sigma_h(e)$ are already determined. This means that $e \in E(V_{(\sigma_1(e),\ldots,\sigma_h(e))})$. Let $V_{(\sigma_1(e),\ldots,\sigma_h(e),0)}, V_{(\sigma_1(e),\ldots,\sigma_h(e),1)}, \ldots, V_{(\sigma_1(e),\ldots,\sigma_h(e),k_{h+1})}$ be the star decomposition of $V_{(\sigma_1(e),\ldots,\sigma_h(e))}$. Let p_{h+1} denote the index-matching of this decomposition. If $e \in E(V_{(\sigma_1(e),\ldots,\sigma_h(e),j_{h+1})})$ for some $j_{h+1} \in \{0, 1, \ldots, k_{h+1}\}$ then $\sigma_{h+1}(e) = j_{h+1}$; otherwise, $\sigma(e) = (\sigma_1(e), \sigma_2(e), \ldots, \sigma_h(e))$.

Observe that if $|\sigma(e)| = h$ (the length of the sequence is h), it means that $e \in \partial\left(V_{(\sigma_1(e),\ldots,\sigma_h(e),0)}, V_{(\sigma_1(e),\ldots,\sigma_h(e),1)}, \ldots, V_{(\sigma_1(e),\ldots,\sigma_h(e),k_{h+1})}\right)$. Let $h = h(e)$ denote the length of $\sigma(e)$. Let j_e, j'_e denote the two indices such that $e \in \partial\left(V_{(\sigma_1(e),\ldots,\sigma_h(e),j_e)}\right) \cap \partial\left(V_{(\sigma_1(e),\ldots,\sigma_h(e),j'_e)}\right)$. (Observe that there are precisely two indices like that.) Let $\hat{\sigma}(e)$ denote the pair containing the sequence $\sigma(e)$ as its first element, and the pair $\{j_e, j'_e\}$ as its second. Let $\pi(e)$ denote the sequence of the same length as $\sigma(e)$, defined by $\pi_i(e) = p_i(\sigma_i(e))$ for every index $i \in \{1, 2, \ldots, h\}$. Similarly, let $\hat{\pi}(e)$ denote the pair containing the sequence $\pi(e)$ as its first element, and the pair $\{p_{h+1}(j_e), p_{h+1}(j'_e)\}$ as the second.

Observe that due to edge contractions, paradoxically, it may happen that $\mathrm{vol}\left(E(V_{(\sigma_1(e),\ldots,\sigma_i(e),\sigma_{i+1}(e))})\right)$ is greater than $\mathrm{vol}\left(E(V_{(\sigma_1(e),\ldots,\sigma_i(e))})\right)$, even though $V_{(\sigma_1(e),\ldots,\sigma_i(e),\sigma_{i+1}(e))} \subseteq V_{(\sigma_1(e),\ldots,\sigma_i(e))}$. (The reason for this anomaly is the different radii of those sets; the radii, in turn, determine the ranges of the lengths of edges that are counted for these two volumes.) For $i \in \{1, 2, \ldots, h(e) - 1\}$, let the *growth rate* of the edge e on recursion level i be defined by

$$g_i(e) = \min\left\{1, \frac{\mathrm{vol}\left(E(V_{(\sigma_1(e),\ldots,\sigma_{i+1}(e))})\right)}{\mathrm{vol}\left(E(V_{(\sigma_1(e),\ldots,\sigma_i(e))})\right)}\right\} .$$

For each index $p \in \{0, 1, \ldots, t-1\}$, let $l_p = l_p(e)$ denote the number of times the index p appears in the sequence $\pi(e)$.

Lemma F.1.

$$\sum_{e \in E} \sum_{i=1}^{h(e)-1} g_i(e) \ \le\ \bar{m} \cdot \log \bar{m} .$$

Proof. For a fixed edge e, and index $i \in \{1, 2, \ldots, h(e)-1\}$, consider the vertex set $C = V_{(\sigma_1(e),\ldots,\sigma_i(e))}$. Note that for every edge $e' \in E(C)$, $g_i(e') = g_i(e)$. Consequently,

$$\sum_{e' \in E(C)} g_i(e') \ = \ |E(C)| \cdot g_i(e) \ \le\ |E(V_{(\sigma_1(e),\ldots,\sigma_{i+1}(e))})| .$$

Hence $\sum_{e \in E} g_i(e) \le \bar{m}$. The desired upper bound on $\sum_{i-1}^{h(e)-1} \sum_{e \in E} g_i(e)$ follows now as for every edge $e \in E$, $h(e) \le \log n$. $\square$

We next use Lemma F.1 to derive an upper bound on average $l_p(e)$ (average over the edges $e \in E$).

Lemma F.2.

$$\sum_{e\in E} l_p(e) \;=\; O(\bar{m}\log^{1-(p/t)}\bar{m}) \ .$$

Proof. Note that each time the index p appears in the sequence $\pi(e)$, the volume of the edge set on which the algorithm recurses and that contains e decreases by a factor $2^{\log^{p/t}\bar{m}}$ (see (2) in Lemma 5.1). Consequently,

$$\left(2^{\log^{p/t}\bar{m}}\right)^{l_p(e)} \;\le\; \bar{m}\cdot\prod_{i=1}^{h(e)-1} g_i(e) \ ,$$

and so $l_p(e)\log^{p/t}\bar{m} \le \log\bar{m} + \sum_{i=1}^{h(e)-1} g_i(e)$.

Summing up over all edges $e \in E$, we derive:

$$\log^{p/t}\bar{m}\sum_{e\in E} l_p(e) \;\le\; \bar{m}\log\bar{m} + \sum_{e\in E}\sum_{i=1}^{h(e)-1} g_i(e) \ .$$

By Lemma F.1 the second term of the right-hand side is at most $\bar{m}\log\bar{m}$, completing the proof. □

For a fixed edge $e \in E$, and an index $p \in \{0, 1, \ldots, t-1\}$, each time the index p appears in $\pi(e)$ it reflects a contribution of $O(t/\alpha)\cdot\log^{(p+1)/t}\bar{m}$ to the sum $|E|\cdot AS(V_0)$ of the stretches of all the edges. Consequently, by Lemma F.2, the overall contribution of the appearances of the index p in the sequences $\pi(e)$ for all edges $e \in E$, is at most

$$O(t/\alpha)\cdot\log^{(p+1)/t}\bar{m}\sum_{e\in E} l_p(e) = O(t\cdot\log^{2+1/t}\bar{m} \ .$$

Since there are t different indices $p \in \{0, 1, \ldots, t-1\}$, their overall contribution is $O(t^2/\alpha)\cdot\log^{1+(1/t)}\bar{m} = O(t^2\cdot\log^{2+1/t}\bar{m})$.

Also, each time that $p = -1$ appears (recall that it corresponds to the edge e belonging to the central (V_0) part of the star-decomposition), it contributes $O(\log\bar{m})$ (see the second term of the right-hand side expression in (6)). Also, $|\pi(e)| = O(\log n)$, and so the appearances of $p = -1$ contribute altogether an additive term of $O(\log^2 n)$ to the coefficient of $\mathrm{vol}(\{e\})$.

Finally, recall that $e \in \partial\left(V_{(\sigma(e),j_e)}\right)\cap\partial\left(V_{(\sigma(e),j'_e)}\right)$, where $\hat{\sigma}(e) = \{\sigma(e), \{j_e, j'_e\}\}$. Consequently, e is counted in $\mathrm{vol}\left(V_{(\sigma(e),j_e)}\right)$ and in $\mathrm{vol}\left(V_{(\sigma(e),j'_e)}\right)$. However, the coefficients of both these terms are $O(\log^2 n)$, and thus, these terms do not affect the asymptotic expression for the average stretch more than by a constant factor.

Hence, the overall average stretch is $O(t^2\cdot\log^{2+1/t}\bar{m}) = O(t^2\cdot\log^2 n)$. To optimize this expression we set $t = \log\log n$, and obtain the desired bound of $O((\log n\cdot\log\log n)^2)$.

□