SIAM J. COMPUT.
Vol. 48, No. 2, pp. 249–278

STEINER POINT REMOVAL WITH DISTORTION $O(\log k)$ USING THE RELAXED-VORONOI ALGORITHM*

ARNOLD FILTSER†

Abstract. In the Steiner point removal problem, we are given a weighted graph $G = (V, E)$ and a set of terminals $K \subset V$ of size k. The objective is to find a minor M of G with only the terminals as its vertex set, such that distances between the terminals will be preserved up to a small multiplicative distortion. Kamma, Krauthgamer, and Nguyen [*SIAM J. Comput.*, 44 (2015), pp. 975–995] devised a ball-growing algorithm with exponential distributions to show that the distortion is at most $O(\log^5 k)$. Cheung [*Proceedings of the 29th Annual ACM/SIAM Symposium on Discrete Algorithms*, 2018, pp. 1353–1360] improved the analysis of the same algorithm, bounding the distortion by $O(\log^2 k)$. We devise a novel and simpler algorithm (called the `Relaxed-Voronoi` algorithm) which incurs distortion $O(\log k)$. This algorithm can be implemented in almost linear time ($O(|E| \log |V|)$).

Key words. Steiner point removal (SPR), distortion, metric embedding, minor graph, randomized algorithm

AMS subject classifications. 41, 60, 68

DOI. 10.1137/18M1184400

1. Introduction.

In graph compression problems the input is usually a massive graph. The objective is to compress the graph into a smaller graph, while preserving certain properties of the original graph, such as distances or cut values. Compression allows us to obtain faster algorithms while reducing the storage space. In the era of massive data, the benefits are obvious. Examples of such structures are graph spanners [37], distance oracles [39], cut sparsifiers [7], spectral sparsifiers [6], and vertex sparsifiers [36].

In this paper we study the *Steiner point removal* (SPR) problem. Here we are given an undirected graph $G = (V, E)$ with positive weight function $w : E \to \mathbb{R}_+$, and a subset of terminals $K \subseteq V$ of size k (the nonterminal vertices are called Steiner vertices). The goal is to construct a new graph $M = (K, E')$ with positive weight function w', with the terminals as its vertex set, such that (1) M is a graph minor of G and (2) the distance between every pair of terminals t, t' is distorted by at most a multiplicative factor of α, formally

$$\forall t, t' \in K, \quad d_G(t, t') \leq d_M(t, t') \leq \alpha \cdot d_G(t, t') \ .$$

Property (1) expresses preservation of the topological structure of the original graph. For example, if G was planar, so will M be. Property (2), however, expresses preservation of the geometric structure of the original graph, that is, distances between terminals. The question is, What is the minimal α (which may depend on k) such that every graph with a terminal set of size k will admit a solution to the SPR problem with distortion α?

The first to study a problem of this flavor was Gupta [24], who showed that given a weighted tree T with a subset of terminals K, there is a tree T' with K as its vertex

*Received by the editors April 30, 2018; accepted for publication January 22, 2019; published electronically March 26, 2019. A preliminary version appeared in *Proceedings of SODA*'18, 2018.
http://www.siam.org/journals/sicomp/48-2/M118440.html
Funding: The research was supported in part by ISF grant (1718/18) and BSF grant 2015813.
†Department of Computer Science, Ben Gurion University of the Negev, Beer Sheva, 8410501, Israel (arnoldf@cs.bgu.ac.il).

set that preserves all the distances between terminals up to a multiplicative factor of 8. Chan et al. [9] observed that the tree T' of Gupta is in fact a minor of the original tree T. They showed that 8 is the best possible distortion and formulated the problem for general graphs. This lower bound of 8 is achieved on the complete unweighted binary tree and is the best known lower bound for the general SPR problem.

Basu and Gupta [5] showed that on outerplanar graphs, the SPR problem can be solved with distortion $O(1)$.

Kamma, Krauthgamer, and Nguyen were the first to bound the distortion for general graphs. They suggested the `Ball-growing` algorithm. Their first analysis provide $O(\log^6 k)$ distortion (conference version [26]), which they later improved to $O(\log^5 k)$ (journal version [27]). Recently, Cheung [11] improved the analysis of the `Ball-growing` algorithm further, providing an $O(\log^2 k)$ upper bound on the distortion.

The `Ball-growing` algorithm constructs a terminal partition, that is, a partition where each cluster is connected and contains a single terminal. The minor is then constructed by contracting all the internal edges in all clusters. The weight of the minor edge $\{t, t'\}$ (if it exists) is defined simply to $d_G(t, t')$. The clusters are generated iteratively. In each round, by turn, each terminal t_j increases the radius R_j of its ball cluster V_j in an attempt to add more vertices to its ball cluster V_j. Once a vertex joins some cluster, it will remain there. In round ℓ, the radii are (independently) distributed according to an exponential distribution, where the mean of the distribution grows in each round. A description of the `Ball-growing` algorithm can be found in Appendix B.

The main contribution of this paper is a new upper bound of $O(\log k)$ for the SPR problem. In a preliminary conference version [20], the author improved the analysis of the `Ball-growing` algorithm, providing an $O(\log k)$ upper bound. In this paper we devise a novel algorithm called the `Relaxed-Voronoi` algorithm. We bound the distortion incurred by the minor produced using the `Relaxed-Voronoi` by $O(\log k)$ as well. Nevertheless, the `Relaxed-Voronoi` algorithm is arguably simpler and more intuitive compared to the `Ball-growing` algorithm. Both algorithms grow clusters around the terminals; the main difference is that the `Ball-growing` algorithm has many iterations, growing slowly from all terminals (almost in parallel), while the `Relaxed-Voronoi` algorithm has one round only (the terminals create clusters by turns. Once a cluster is created it will remain unchanged till the end of the algorithm). The analysis in [20] was built upon [11]. In both papers, a considerable effort was made to lower and upper bound the number of the round in which each nonterminal is clustered. The analysis in this paper is quite similar to [20], while all the round-base analysis simply becomes unnecessary.

Furthermore, we devise an efficient implementation of the `Relaxed-Voronoi` algorithm in almost linear time $O\left(m + \min\{m, nk\} \cdot \log n\right)$ (m (resp., n) here is the number of edges (resp., vertices) in G). While the `Ball-growing` algorithm can be implemented in polynomial time, it is not clear how to do so efficiently.

We show that the analysis of the `Relaxed-Voronoi` algorithm is asymptotically tight. That is, there are graphs for which the `Relaxed-Voronoi` produces a minor which incurs distortion $\Omega(\log k)$. We prove a similar lower bound also for the `Ball-growing` algorithm. However, there we are only able to prove an $\Omega(\sqrt{\log k})$ lower bound on the performance of the algorithm.

1.1. Related work. Englert et al. [17] showed that every graph G admits a distribution $\mathcal{D}$ over terminal minors with expected distortion $O(\log k)$. Formally, for all $t_i, t_j \in K$, it holds that $1 \leq \frac{\mathbb{E}_{M\sim\mathcal{D}}[d_M(t_i,t_j)]}{d_G(t_i,t_j)} \leq O\left(\log k\right)$. Thus, Theorem 3.1 can be

seen as an improvement upon [17], where we replace distribution with a single minor. Englert et al. showed better results for β-decomposable graphs; in particular, they showed that graphs excluding a fixed minor admit a distribution with $O(1)$ expected distortion.

Krauthgamer, Nguyen, and Zondiner [29] showed that if we allow the minor M to contain at most $\binom{k}{2}^2$ Steiner vertices (in addition to the terminals), then distortion 1 can be achieved. They further showed that for graphs with constant treewidth, $O(k^2)$ Steiner points will suffice for distortion 1. Cheung, Goranci, and Henzinger [12] showed that allowing $O(k^{2+\frac{2}{t}})$ Steiner vertices, one can achieve distortion $2t-1$ (in particular distortion $O(\log k)$ with $O(k^2)$ Steiners). For planar graphs, they achieved $1+\epsilon$ distortion with $\tilde{O}((\frac{k}{\epsilon})^2)$ Steiner points.

There is a long line of work focusing on preserving the cut/flow structure among the terminals by a graph minor. See [36, 32, 10, 34, 17, 13, 30, 2, 23, 31].

There are works studying metric embeddings and metric data structures concerning preserving distances among terminals, or from terminals to other vertices, out of the context of minors. See [14, 38, 25, 28, 15, 16, 4].

Finally, there are clustering algorithms similar in nature to the `Relaxed-Voronoi` and `Ball-growing` algorithms [33, 3, 19, 8, 18, 35].

1.2. Technical ideas. The basic approach in this paper, as well as in all previous papers on SPR in general graphs, is to use terminal partitions in order to construct a minor for the SPR problem. Specifically, we partition the vertices into k connected clusters, with a single terminal in each cluster. Such a partition induces a minor by contracting all the internal edges in each cluster. See the preliminaries for more details. Considering such a framework, the most natural idea will be to partition the vertices into the Voronoi cells, i.e., the cluster V_j of the terminal t_j will contain all the vertices v for which t_j is the closest terminal. However, this approach miserably fails and can incur distortion as large as $k-1$. See Figure 1.1 for an illustration.

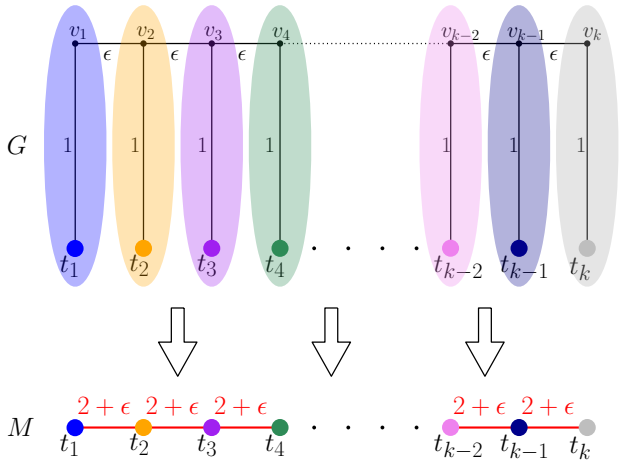

FIG. 1.1. *The graph G consists of a k-path of Steiner vertices $v_1, \ldots, v_k$ with edges of weight ϵ. To each Steiner vertex v_j we add a terminal using a unit weight edge. The Voronoi cell of the terminal t_j is $\{t_j, v_j\}$. The minor M induced by this terminal partition is a path $t_1, \ldots, t_k$ where the weight of each edge equals $2+\epsilon$. The original distance in G between t_1 to t_k is $2+(k-1)\cdot\epsilon$, while the distance in the minor M equals $(k-1)\cdot(2+\epsilon)$. In particular, when ϵ tends to 0, the distortion tends to $k-1$.*

Our idea is to introduce some noise in order to avoid the sharp boundaries between the clusters. Specifically, we order the terminals in an arbitrary order. For each terminal t_j we sample a parameter $R_j \geq 1$ that we will call its *magnitude*. Then, by turn, each terminal will construct a cluster V_j which will be essentially a magnified (by R_j) Voronoi cell (in the remaining graph). However, in order to maintain connectivity, the magnified Voronoi cell is constructed in a "Dijkstra manner" as follows. For every vertex v, denote by $D(v)$ the distance from v to its closest terminal. Initially $V_j = \{t_j\}$. In each step, every unclustered neighboring vertex v of V_j is examined. If $d_G(v, t_j) \leq R_j \cdot D(v)$, then v joins the cluster V_j. The process terminates when no new potential vertices remain. Then we move on to the next terminal and repeat the same process on the remaining graph. Eventually, all of G is partitioned into clusters.

To sample R_j, we first sample g_j according to geometric distribution with parameter $p = \frac{1}{5}$. Then, R_j is set to be $(1+\delta)^{g_j}$, where $\delta = \Theta(\frac{1}{\ln k})$. In particular, all the R_j's are bounded by some universal constant with high probability (w.h.p.).

Next, we provide some intuition for the distortion analysis. Consider a pair of terminals t, t', and let $P_{t,t'}$ be the shortest path between them in the original graph G. When the algorithm terminates, all the vertices in $P_{t,t'}$ are clustered by different terminals. See Figure 4.2 for an illustration. Let $\mathcal{D}_{\ell_1}, \ldots, \mathcal{D}_{\ell_k}$ be the partition of the vertices in $P_{t,t'}$ induced by the partition of all vertices created by the algorithm. i.e., $\mathcal{D}_{\ell_i} = P_{t,t'} \cap V_{\ell_i}$. For simplicity at this stage, we will assume that every $\mathcal{D}_{\ell_j}$ is continuous. In the induced minor graph, there is an edge between any two consecutive terminals t_{ℓ_j} and $t_{\ell_{j+1}}$. Therefore the distance between t and t' in the minor graph can be bounded by $\sum_j d_G(t_{\ell_j}, t_{\ell_{j+1}})$. Let v^{ℓ_j} be the "first" vertex on $P_{t,t'}$ to be covered by t_{ℓ_j}. "First" here is in the following sense: we think about the sampling of R_j in a gradual manner. For a vertex v, let r_v denote the minimal value of R_j such that $v \in V_j$. Then v^j is defined to be the vertex with the minimal value r_v. Using the triangle inequality, $d_G(t_{\ell_j}, t_{\ell_{j+1}}) \leq d_G(t_{\ell_j}, v^{\ell_j}) + d_G(v^{\ell_j}, v^{\ell_{j+1}}) + d_G(v^{\ell_{j+1}}, t_{\ell_{j+1}})$. Therefore $d_M(t,t') \leq \sum_{i=1}^{k'-1} d_G(v^{\ell_i}, v^{\ell_{i+1}}) + 2\sum_{i=1}^{k'} d_G(t_{\ell_i}, v^{\ell_i}) \leq d_G(t,t') + 2\sum_{i=1}^{k'} d_G(t_{\ell_i}, v^{\ell_i})$ (see Figure 4.2 for an illustration).

In order to bound the distortion, we need to bound the sum of "deviations" $\sum_{i=1}^{k'} d_G(t_{\ell_i}, v^{\ell_i})$ from the shortest path. However, these deviations are heavily dependent. Instead of analyzing the deviations directly, we will follow an approach first suggested by [11]. We partition the shortest path $P_{t,t'}$ from t to t' into a set of intervals $\mathcal{Q}$; the idea will be to count for each interval Q how many deviations start from this interval (denoted $X(Q)$). Specifically, for each deviation, we will charge the interval in which this deviation was initiated. Afterward, we will be able to replace the sum of deviations above by a linear combination of the interval charges.

The partition of the shortest path $P_{t,t'}$ into intervals is done such that the length of each interval $Q \in \mathcal{Q}$ will be a $\log k$ fraction of the distance from the interval to its closest terminal. Such interval lengths will ensure the following crucial property: given that some vertex $v \in Q$ joins the cluster V_j (of the terminal t_j), with probability at least $1-p$, all of Q joins V_j.

Using this property alone, one can show that the expected charge on each interval is bounded by a constant. This already will imply an $O(\log k)$ distortion on each pair in expectation. However, as we are interested in $O(\log k)$ distortion on all pairs w.h.p., a more subtle argument is required. We couple the interval charges into a series of independent random variables that dominate the interval charges. Then, a concentration bound on the independent variables implies an upper bound on the sum of interval charges, which provides $O(\log k)$ distortion w.h.p.

1.3. Paper organization. In section 3 we describe the `Relaxed-Voronoi` algorithm and prove some of its basic properties. Then, in section 4 we analyze the distortion incurred by the `Relaxed-Voronoi` algorithm. In section 5 we introduce a small modification to the `Relaxed-Voronoi` algorithm. We prove that the distortion analysis is still valid and explain how the modified algorithm can be efficiently implemented. In section 6 we prove that our analysis of the `Relaxed-Voronoi` algorithm is asymptotically tight (and provide some lower bound on the performance of the `Ball-growing` algorithm). Finally, in section 7 we provide some concluding remarks and discuss further directions.

2. Preliminaries. Appendix C contains a summary of all the definitions and notation we use. The reader is encouraged to refer to this index while reading.

We consider undirected graphs $G = (V, E)$ with positive edge weights $w : E \to \mathbb{R}_{\geq 0}$. Let d_G denote the shortest path metric in G. For a subset of vertices $A \subseteq V$, let $G[A]$ denote the *induced graph* on A. Fix $K = \{t_1, \ldots, t_k\} \subseteq V$ to be a set of *terminals*. For a vertex v, $D(v) = \min_{t \in K} d_G(v, t)$ is the distance from v to its closest terminal. For clarity, we will assume that all metric distances are unique (that is, for $\{v, v'\} \neq \{u, u'\}$, $d_G(v, v') \neq d_G(u, u')$). Moreover, we will assume that for every pair v, u there is a unique shortest path. Otherwise, we can introduce arbitrarily small perturbations.

A graph H is a *minor* of a graph G if we can obtain H from G by edge deletions/contractions and vertex deletions. A partition $\{V_1, \ldots, V_k\}$ of V is called a *terminal partition* (w.r.t. K) if for every $1 \leq i \leq k$, $t_i \in V_i$, and the induced graph $G[V_i]$ is connected. See Figure 2.1 for an illustration. The *induced minor* by terminal partition $\{V_1, \ldots, V_k\}$ is a minor M, where each set V_i is contracted into a single vertex called (abusing notation) t_i. Note that there is an edge in M from t_i to t_j iff there are vertices $v_i \in V_i$ and $v_j \in V_j$ such that $\{v_i, v_j\} \in E$. We determine the weight of the edge $\{t_i, t_j\} \in E(M)$ to be $d_G(t_i, t_j)$. Note that by the triangle inequality, for every pair of (not necessarily neighboring) terminals t_i, t_j, it holds that $d_M(t_i, t_j) \geq d_G(t_i, t_j)$. The *distortion* of the induced minor is $\max_{i,j} \frac{d_M(t_i, t_j)}{d_G(t_i, t_j)}$.

2.1. Probability. For a distribution $\mathcal{D}$, $X \sim \mathcal{D}$ denotes that X is a random variable distributed according to $\mathcal{D}$.

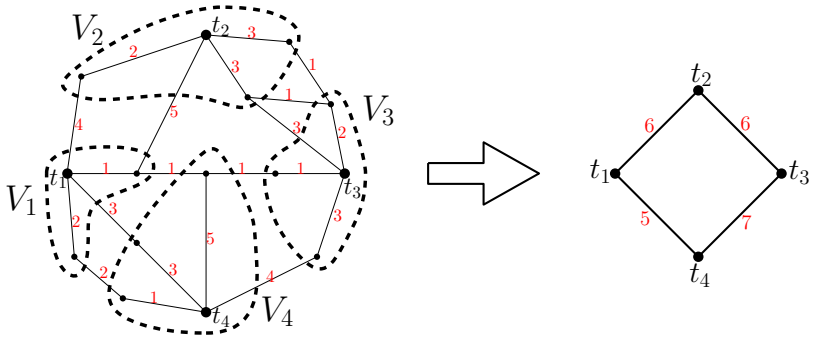

FIG. 2.1. *The left side of the figure contains a weighted graph $G = (V, E)$, with weights specified in red, and four terminals $\{t_1, t_2, t_3, t_4\}$. The dashed black curves represent a terminal partition of the vertex set V into the subsets V_1, V_2, V_3, V_4. The right side of the figure represent the minor M induced by the terminal partition. The distortion is realized between t_1 and t_3, and is $\frac{d_M(t_1,t_3)}{d_G(t_1,t_3)} = \frac{12}{4} = 3$.*

$\mathsf{Geo}(p)$ denotes the *geometric distribution* with parameter p. Here we toss a biased coin with probability p for heads, until the first time we get heads. $\mathsf{Geo}(p)$ is the number of coin tosses. Formally, $\mathsf{Geo}(p)$ is supported in $\{1, 2, 3, \dots\}$, where the probability to get s is $(1-p)^{s-1} \cdot p$.

Exponential distribution is the continuous analogue of geometric distribution. $\mathsf{Exp}(\lambda)$ denotes the exponential distribution with mean λ and density function $f(x) = \frac{1}{\lambda} e^{-\frac{x}{\lambda}}$ for $x \geq 0$. Exponential distribution is closed under scaling, that is, for $X \sim \mathsf{Exp}(\lambda)$, $c \cdot X$ is distributed according to $\mathsf{Exp}(c\lambda)$. We will use the following concentration bound.

LEMMA 2.1. *Suppose $X_1, \dots, X_n$'s are independent random variables, where each X_i is distributed according to $\mathsf{Exp}(\lambda_i)$. Let $X = \sum_i X_i$ and $\lambda_M = \max_i \lambda_i$. Set $\mu = \mathbb{E}[X] = \sum_i \lambda_i$.*

$$\text{For } a \geq 2\mu, \qquad \Pr[X \geq a] \leq \exp\left(-\frac{1}{2\lambda_M}(a - 2\mu)\right) .$$

In Appendix A we prove a more general bound. In particular, Lemma 2.1 above is a special case of Lemma A.1 (which is obtained by choosing parameters $\alpha = \frac{a}{\mu} - 1$ and $t = \frac{1}{2\lambda_M}$).

3. Algorithm. The terminals are ordered in arbitrary order $t_1, t_2, \dots, t_k$. The `Relaxed-Voronoi` algorithm has k rounds, where in the round i, the cluster V_i (containing t_i) is constructed in the graph induced by the non-terminal vertices not clustered so far.

The clusters are created using the `Create-Cluster` procedure. The algorithm provides a random variable $R_j = (1+\delta)^{g_j}$, where g_j is distributed according to geometric distribution with parameter p.

The `Create-Cluster` procedure runs in a Dijkstra-like fashion. During the execution, we maintain three sets: (1) V_j: the currently created cluster (initiated to be $\{t_j\}$). (2) U: the set of vertices that were "refused" to join V_j. (3) N: the set of neighboring vertices to V_j (that are not in U).

While N is nonempty, the algorithm extracts an arbitrary vertex v from N. If $d_G(v, t_j) \leq R(j) \cdot D(v)$ (the distance from t_j to v is at most R_j times the distance from v to its closest terminal), then v joins V_j. Otherwise v joins U. In the case where v joins V_j, all its neighbors (outside of $U \cup V_j$) join N. As each vertex might join N at most once, eventually N becomes empty. Then the procedure ceases and returns V_j.

THEOREM 3.1. *With probability $1 - \frac{1}{k}$, in the minor graph M returned by Algorithm* 3.1*, it holds that for every two terminals t, t', $d_M(t, t') \leq O(\log k) \cdot d_G(t, t')$.*

First we argue that Algorithm 3.1 indeed produces a terminal partition.

LEMMA 3.2. *The sets $V_1, \dots, V_k$ constructed by Algorithm* 3.1 *form a terminal partition.*

Proof. It is straightforward from the description of the algorithm that the sets $V_1, \dots, V_k$ are disjoint and that for every j, $t_j \in V_j$ and $G[V_j]$ is connected. The only nontrivial property we have to show is that every vertex $v \in V$ joins some cluster.

Fix some $v \in V$, let t_j be the closest terminal to v (s.t. $D(v) = d_G(v, t_j)$), and let $P = \{t_j = u_0, u_1, \dots, u_s = v\}$ be the shortest path from t_j to v in G. Note that as P is a shortest path, t_j is also the closest terminal to all the vertices in P. As $t_j = u_0 \in V_j$, at least one vertex from P is clustered during the algorithm. Let $u_{i'}$ be

Algorithm 3.1. $M = \texttt{Relaxed-Voronoi}(G = (V, E, w), K = \{t_1, \ldots, t_k\})$.

1: Set $\delta = \frac{1}{20 \ln k}$ and $p = \frac{1}{5}$.
2: Set $V_\perp \leftarrow V \setminus K$. // $V_\perp$ *is the currently unclustered vertices.*
3: **for** j from 1 to k **do**
4: Choose independently at random g_j distributed according to $\mathsf{Geo}(p)$.
5: Set $R_j \leftarrow (1+\delta)^{g_j}$.
6: Set $V_j \leftarrow \texttt{Create-Cluster}(G, V_\perp, t_j, R_j)$.
7: Remove all the vertices in V_j from $V_\perp$.
8: **end for**
9: **return** the terminal-centered minor M of G induced by $V_1, \ldots, V_k$.

the first clustered vertex from P (w.r.t. time). Denote by $V_{j'}$ the cluster $u_{i'}$ joins to. We argue by induction on $i \geq i'$ that u_i also joins $V_{j'}$. This will imply that $u_s = v$ joins $V_{j'}$ and thus is clustered. Suppose u_i joins $V_{j'}$. It holds that $d_G(u_i, t_{j'}) \leq R_{j'} \cdot D(u_i)$. Moreover, all the neighbors of u_i join N. Therefore u_{i+1} necessarily joined to the set N (at some stage during the execution of the `Create-Cluster` procedure for $V_{j'}$). As

$$\begin{aligned} d_G(u_{i+1}, t_{j'}) &\leq d_G(u_{i+1}, u_i) + d_G(u_i, t_{j'}) \\ &\leq d_G(u_{i+1}, u_i) + R_{j'} \cdot d_G(u_i, t_j) \\ &\leq R_{j'} \cdot d_G(u_{i+1}, t_j) = R_{j'} \cdot D(u_{i+1}) \ , \end{aligned}$$

u_{i+1} will join $V_{j'}$, as required. ∎

3.1. Modification. Let $\hat{\Delta} = \min_{t,t' \in K}\{d_G(t, t')\}$ denote the minimal distance between a pair of terminals. Note that $\hat{\Delta} > 0$. For the sake of analysis we will make a preprocessing step to ensure that every edge e has weight at most $c_w \cdot \hat{\Delta} = \frac{\delta}{24} \cdot \hat{\Delta}$. This can be achieved by subdividing larger edges, i.e., adding additional vertices of degree two in the middle of such edges. Denote by $\hat{G}$ the modified graph G, when we repeatedly subdivide edges until every edge e has small enough weight. We argue that such subdivisions did not affect whatsoever the terminal-centered minor returned by Algorithm 3.1.

CLAIM 3.3. *Let $G = (V, E, w)$ be a weighted graph with terminal set $K = \{t_1, \ldots, t_k\}$. Consider an edge $e = \{v, u\} \in E$ of weight ω. Let $\tilde{G}$ be the graph G with subdivided edge e. Specifically, we add a new Steiner vertex v_e and replace the edge e by two new edges $\{v_e, v\}$, $\{v_e, u\}$, both of weight $\omega/2$.*

Fix $g_1, \ldots, g_k$ and consider Algorithm 3.1*, where the random choices in line* 4 *are $g_1, \ldots, g_k$, respectively. Then the terminal-centered minor M returned on input G is the same as the terminal-centered minor $\tilde{M}$ returned on input $\tilde{G}$.*

Proof. As $g_1, \ldots, g_k$ are fixed, Algorithm 3.1 is now deterministic. Let $V_1, \ldots, V_k$ be the terminal partition induced by Algorithm 3.1 on G, and similarly let $\tilde{V}_1, \ldots, \tilde{V}_k$ be the terminal partition induced by Algorithm 3.1 on $\tilde{G}$. We argue that for all j, $V_j = \tilde{V}_j \setminus \{v_e\}$. Note that this will imply our claim. Indeed, let V_j, $V_{j'}$ be the clusters such that $v \in V_j$ and $u \in V_{j'}$. As each cluster is connected, necessarily $v_e \in V_j \cup V_{j'}$. By the definition of subdivision, this will imply that the terminal-centered minors are indeed identical.

Each Steiner vertex can be clustered only after at least one of its neighbors is clustered. Therefore v_e cannot be clustered before both v and u. Without loss of generality (w.l.o.g.) v joined V_j while u is still unclustered. The vertex v_e wasn't

Algorithm 3.2. $V_j = \texttt{Create-Cluster}(G = (V, E, w), V_\perp, t_j, R_j)$.

1: Set $V_j \leftarrow \{t_j\}$.
2: Set $U \leftarrow \emptyset$. *// U is the set of vertices already denied from V_j.*
3: Set N to be all the neighbors of t_j in $V_\perp$.
4: **while** $N \neq \emptyset$ **do**
5: Let v be an arbitrary vertex from N.
6: Remove v from N.
7: **if** $d_G(v, t_j) \leq R_j \cdot D(v)$ **then**
8: Add v to V_j.
9: Add all the neighbors of v in $V_\perp \setminus (U \cup V_j)$ to N.
10: **else**
11: Add v to U.
12: **end if**
13: **end while**
14: **return** V_j.

examined before the clustering of v. Denote by V'_j (resp., $\tilde{V}'_j$) the set V_j (resp., $\tilde{V}_j$) right after the clustering of v at the execution of Algorithm 3.1 on G (resp., $\tilde{G}$). Note that the order of extraction from N in line 5 of Algorithm 3.2 is determined deterministically. Therefore, up to the clustering of v the algorithm behaved the same on both G and $\tilde{G}$. In particular, for all $j'' < j$, $V_{j''} = \tilde{V}_{j''}$. Moreover, $V'_j = \tilde{V}'_j$. After v joins V_j, v_e joins (for the first time) to the set N (for $\tilde{G}$). Note that

$$D(v_e) = \min\{D(v), D(u)\} + \frac{\omega}{2},$$
$$d_G(t_j, v_e) = \min\{d_G(t_j, v), d_G(t_j, u)\} + \frac{\omega}{2}.$$

As v joined V_j, necessarily $d_G(t_j, v) \leq R_j \cdot D(v)$. Consider the following cases:

- $u \notin V_j$: In the algorithm for G, u was examined (as $v \in V_j$), thus $d_G(t_j, u) > R_j \cdot D(u)$. Therefore u will also not join $\tilde{V}_j$. As v_e has edges only to v and u, v_e has no impact on any other vertex. Therefore the cluster $\tilde{V}_j$ will be constructed in the same manner as V_j (up to maybe containing v_e). Note that all the other clusters will not be affected, as if v_e remained unclustered, it becomes a leaf. We conclude that for every j'', $V_{j''} = \tilde{V}_{j''} \setminus \{v_e\}$.
- $u \in V_j$: It holds that $d_G(t_j, u) \leq R_j \cdot D(u)$. Therefore

$$d_G(t_j, v_e) = \min\{d_G(t_j, v), d_G(t_j, e)\} + \frac{\omega}{2} \leq R_j \cdot \min\{D(v), D(u)\} + \frac{\omega}{2} \leq R_j \cdot D(v_e)\ .$$

  Therefore v_e will join $\tilde{V}_j$, which will ensure that u joins $\tilde{N}$, and afterward to $\tilde{V}_j$. Note that v_e has no other impact. In particular, for every $j'' \neq j$, $V_{j''} = \tilde{V}_{j''}$ while $V_j \cup \{v_e\} = \tilde{V}_j$. ∎

Consider the modified graph $\hat{G}$. Suppose that we proved that with probability at least $1 - \frac{1}{k}$, in the minor graph $\hat{M}$ returned by Algorithm 3.1 for $\hat{G}$, it holds that for every two terminals t, t', $d_{\hat{M}}(t, t') \leq O(\log k) \cdot d_{\hat{G}}(t, t') = O(\log k) \cdot d_G(t, t')$. Then by repetitive use of Claim 3.3 (once for every new vertex), Theorem 3.1 follows. From now on, we will abuse notation and refer to the graph $\hat{G}$ as G. Note that all this is

done purely for the sake of analysis, as by Claim 3.3 we will get the same minor when running Algorithm 3.1 for either G or $\hat{G}$. Thus, in fact, we will execute Algorithm 3.1 on the original graph with no modifications.

4. Distortion analysis.

4.1. Interval and charges. In this section we describe in detail the probabilistic process of breaking the graph into clusters from the viewpoint of the Steiner vertices. The main objective will be to define a charging scheme, which we can later use to bound the distortion.

Consider two terminals t and t'. Let $P_{t,t'} = \{t = v_0, \ldots, v_\gamma = t'\}$ be the shortest path from t to t' in G. We can assume that there are no terminals in $P_{t,t'}$ other than t, t'. This is because if we will prove that for every pair of terminals t, t' such that $P_{t,t'} \cap K = \{t, t'\}$ it holds that $d_M(t, t') \leq O(\log k) \cdot d_G(t, t')$, this property will be implied for all terminal pairs.

For an interval $Q = \{v_a, \ldots, v_b\} \subseteq P_{t,t'}$, the *internal length* is $L(Q) = d_G(v_a, v_b)$, while the *external length* is $L^+(Q) = d_G(v_{a-1}, v_{b+1})$.[1] The distance from the interval Q to the terminals, denoted $D(Q) = D(v_a)$, is simply the distance from its leftmost point v_a to the closest terminal to v_a. Set $c_{\text{int}} = \frac{1}{6}$ ("int" for interval). We partition the vertices in $P_{t,t'}$ into consecutive intervals $\mathcal{Q}$ such that for every $Q \in \mathcal{Q}$,

$$L(Q) \leq c_{\text{int}}\delta \cdot D(Q) \leq L^+(Q) \ . \tag{4.1}$$

Such a partition could be constructed as follows. Sweep along the interval $P_{t,t'}$ in a greedy manner; after partitioning the prefix $v_0, \ldots, v_{h-1}$, to construct the next Q, simply pick the minimal index s such that $L^+(\{v_h, \ldots, v_{h+s}\}) \geq c_{\text{int}}\delta \cdot D(v_h)$. By the minimality of s, $L(\{v_h, \ldots, v_{h+s}\}) \leq L^+(\{v_h, \ldots, v_{h+s-1}\}) \leq c_{\text{int}}\delta \cdot D(v_h)$ (in the case $s = 0$, trivially $L(\{v_h\}) = 0 \leq c_{\text{int}}\delta \cdot D(v_h)$). Note that such s could always be found, as $L^+(\{v_h, \ldots, v_\gamma\}) = d_G(v_{h-1}, t') \geq d_G(v_h, t') \geq D(v_h) = D(Q)$.

In the beginning of Algorithm 3.1, all the vertices of $P_{t,t'}$ are *active*. Consider round j in the algorithm when terminal t_j constructs its cluster V_j. Specifically, it picks g_j and sets $R_j \leftarrow (1+\delta)^{g_j}$. Then, using the `Create-Cluster` procedure it grows a cluster in a "Dijkstra" fashion. If no active vertex joins V_j, we say that t_j doesn't *participate* in $P_{t,t'}$. Otherwise, let $a_j \in P_{t,t'}$ (resp., b_j) be the active vertex that joins to V_j with minimal (resp., maximal) index (w.r.t. $P_{t,t'}$). All the vertices $\{a_j, \ldots, b_j\} \subset P_{t,t'}$ between a_j and b_j (w.r.t. the order induced by $P_{t,t'}$) become inactive. We call this set $\{a_j, \ldots, b_j\}$ a *detour* $\mathcal{D}_j$ from a_j to b_j. See Figure 4.1 for an illustration.

Within each interval Q, each maximal subinterval of active vertices is called a *slice*. We denote by $\mathcal{S}(Q)$ the current number of slices in Q. In the beginning of the algorithm, for every interval Q, $\mathcal{S}(Q) = 1$, while at the end of the algorithm $\mathcal{S}(Q) = 0$.

For an active vertex v, let r_v be the minimal choice of R_j (determined by g_j) that will force v to join V_j. Let v^j be the active vertex with minimal r_v (breaking ties arbitrarily). Note that V_j is monotone with respect to R_j. That is, if v will join V_j for $R_j = r$, it will join V_j for $R_j = r' \geq r$ as well. We denote by $Q_j \in \mathcal{Q}$ the interval containing v^j. Similarly, S_j is the slice containing v^j. We *charge* Q_j for the detour $\mathcal{D}_j$. We denote by $X(Q)$ the number of detours the interval Q is currently charged for. For every detour $\mathcal{D}_{j'}$ which is contained in $\mathcal{D}_j$ (that is, $a_j < a_{j'} < b_{j'} < b_j$ w.r.t. the order induced by $P_{t,t'}$), we erase the detour and its charge. That is, for every

[1] For ease of notation we will denote $v_{-1} = t$ and $v_{\gamma+1} = t'$.

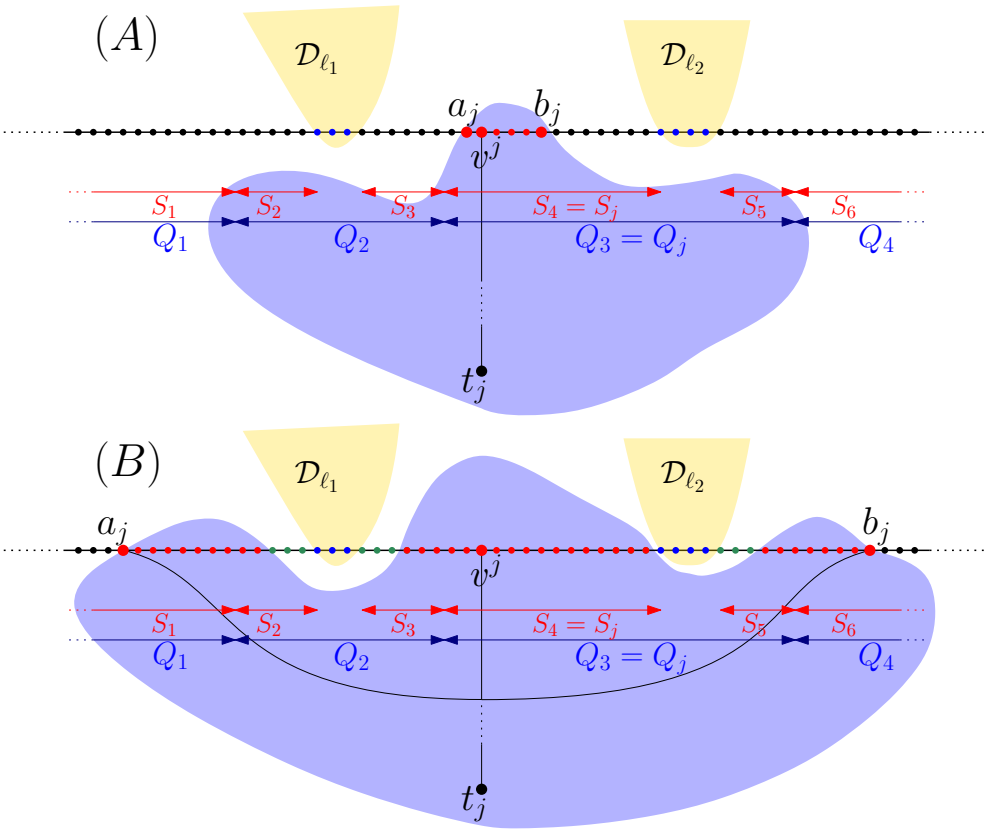

FIG. 4.1. *The figure illustrates round j in Algorithm* 3.1, *when t_j grows the cluster V_j. We present two scenarios for different choices of R_j. The black line is part of $P_{t,t'}$ the shortest path from t to t'. The blue intervals Q_i represent the intervals in $\mathcal{Q}$. The red subintervals S_i represent the slices (maximal continuous subsets of active vertices), where $S_2, S_3 \subset Q_2$ and $S_4, S_5 \subset Q_3$. The yellow areas represent detours $\mathcal{D}_{\ell_1}$ and $\mathcal{D}_{\ell_2}$, where Q_2 (resp., Q_3) is charged for $\mathcal{D}_{\ell_1}$ (resp., $\mathcal{D}_{\ell_2}$). Note that vertices in those areas are inactive. The terminal t_j increases gradually R_j, and the first vertex to be covered is v^j. In scenario* (A), *the growth of R_j terminates immediately after covering v^j and sets the borderline vertices a_j and b_j within the subinterval S_j. In scenario* (B), *the growth of R_j continues for another step, setting both a_j and b_j out of S_j. Vertices already inactive are shown in blue. Vertices that join the cluster V_j are shown in red. The green vertices are vertices which are still uncovered, but nevertheless become inactive. Vertices which remain active after the creation of V_j are colored in black. In scenario* (A) *all the vertices that become inactive, $\mathcal{D}_j$, are included in S_4. Q_3 is charged for $\mathcal{D}_j$. The number of slices in Q_3 is increased by* 1, *and no other changes occur ($X(Q_2) = 1$, $X(Q_3) = 2$). In scenario* (B) *$\mathcal{D}_\ell$ contains all the vertices in S_2, S_3, S_4, S_5 and part of the vertices in S_1, S_6. The number of slices in Q_2 and Q_3 becomes* 0, *while the number of slices in Q_1 and Q_4 remains unchanged. Q_3 is charged for $\mathcal{D}_\ell$, while its charge for $\mathcal{D}_{\ell_2}$ is erased. Additionally, the charge of Q_2 for $\mathcal{D}_{\ell_1}$ is erased. That is, Q_2 will remain uncharged till the end of the algorithm ($\tilde{X}(Q_2) = X(Q_2) = 0$, $X(Q_3) = 1$).*

$Q' \neq Q_j$, $X(Q')$ might only decrease, while $X(Q_j)$ might increase by at most 1 (and can also decrease as a result of deleted detours). We denote by $\tilde{X}(Q)$ the size of $X(Q)$ by the end of Algorithm 3.1. Figure 4.1 illustrates a single step.

Next, we analyze the change in the number of slices as a result of constructing the cluster V_j. If $R_j < r_{v^j}$, then no active vertex joins V_j and therefore $X(Q)$ and $\mathcal{S}(Q)$ stay unchanged, for all $Q \in \mathcal{Q}$. Otherwise, $R_j \geq r_{v^j}$, a new detour will appear

and will be charged upon Q_j. All the slices S which are contained in $\mathcal{D}_j$ are deleted. Every slice S that intersects $\mathcal{D}_j$ but is not contained in it will be replaced by one or two new slices. If $\mathcal{D}_j \cap S \notin \{\mathcal{D}_j, S\}$, then S is replaced by a single new subslice S'. The only possibility for a slice to be replaced by two subslices is if $\mathcal{D}_j \subseteq S$, and $\mathcal{D}_j$ does not contain an "extremal" vertex in S (see Figure 4.1, scenario (A)). This can happen only at S_j. We conclude that for every $Q' \neq Q_j$, $\mathcal{S}(Q')$ might only decrease, while $\mathcal{S}(Q_j)$ might increase by at most 1.

CLAIM 4.1. *Assuming $R_j \geq r_{v^j}$, all of S_j joins V_j with probability at least $1-p$.*

Proof. As v^j joins V_j for $R_j \geq r_{v^j}$, by line 7 of Algorithm 3.2, necessarily $\frac{d_G(v^j,t_j)}{D(v^j)} \leq r_{v^j}$. We will argue that for every $u \in S_j$, the following inequality holds:

$$\frac{d_G(u,t_j)}{D(u)} \leq \frac{d_G(v^j,t_j)}{D(v^j)}\,(1+\delta) \leq r_{v^j}\,(1+\delta)\ . \tag{4.2}$$

Next, assume that $R_j \geq (1+\delta) r_{v^j}$. Before the execution of the `Create-Cluster` procedure for V_j, all the vertices in S_j belong to $V_\perp$ (as all of them are active). Because $R_j \geq r_{v^j}$, v^j will join V_j (by the definition of r_{v^j}). In particular, additional vertices from S_j (if they exist) will join N. Using inequality (4.2), for every $u \in S_j$, $d_G(u,t_j)/D_u \leq r_{v_j}(1+\delta) \leq R_j$. Therefore every vertex from S_j joining N will also join V_j. In such a way, since S_j is connected in $V_\perp$, all the vertices of S_j will join V_j, as required.

Next, we analyze the probability that indeed $R_j \geq (1+\delta) r_{v^j}$. Recall that $R_j = (1+\delta)^{g_j}$, where g_j is distributed according to geometric distribution with parameter $P_{t,t'}$. Conditioned on the event $R_j \geq r_{v^j}$, we have that

$$\begin{aligned} &\Pr\left[R_j \geq (1+\delta) r_{v^j} \mid R_j \geq r_{v^j}\right] \\ &= \Pr\left[g_j \geq \log_{1+\delta}\left((1+\delta) r_{v^j}\right) \mid g_j \geq \log_{1+\delta} r_{v^j}\right] \\ &= \Pr\left[g_j \geq 1 + \log_{1+\delta} r_{v^j} \mid g_j \geq \log_{1+\delta} r_{v^j}\right] = 1-p\ . \end{aligned} \tag{4.3}$$

It remains to prove inequality (4.2). By the definition of $D(Q_j)$ and the triangle inequality

$$\begin{aligned} L(Q_j) &\overset{(4.1)}{\leq} c_{\text{int}}\delta \cdot D(Q_j) \leq c_{\text{int}}\delta \cdot \left(D(v^j) + L(Q_j)\right) \\ &\leq\ 2c_{\text{int}}\delta \cdot D(v^j) \leq 2c_{\text{int}}\delta \cdot d_G(v^j,t_j)\ . \end{aligned} \tag{4.4}$$

Therefore, for every $u \in S_j$,

$$d_G(u,t_j) \leq d_G(v^j,t_j) + L(Q_j) \overset{(4.4)}{\leq} d_G(v^j,t_j)\,(1+2c_{\text{int}}\delta)\,.$$

Similarly,

$$D(u) \geq D(v^j) - L(Q_j) \geq D(v^j)\,(1-2c_{\text{int}}\delta)\ . \tag{4.5}$$

We conclude that

$$\frac{d_G(u,t_j)}{D(u)} \leq \frac{d_G(v^j,t_j)\,(1+2c_{\text{int}}\delta)}{D(v^j)\,(1-2c_{\text{int}}\delta)} \leq \frac{d_G(v^j,t_j)}{D(v^j)}\,(1+3\cdot 2c_{\text{int}}\delta) = \frac{d_G(v^j,t_j)}{D(v^j)}\,(1+\delta)\ .$$

∎

4.2. Bounding the number of failures. Next, we define a *cost function* $f : \mathbb{R}_+^{|\mathcal{Q}|} \to \mathbb{R}_+$. Intuitively, the cost function is simply a summation over the intervals, where for each interval Q we add its length $L(Q)$ for each time it was charged. Formally, $f(\{x_Q\}_{Q\in\mathcal{Q}}) = \sum_{Q\in\mathcal{Q}} x_Q \cdot L^+(Q)$. Even though our goal will be to bound $f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}})$, we define f as a general function from $\mathbb{R}^{|\mathcal{Q}|}$ in order to use it on other variables as well. Note that the cost function f is linear and monotonically increasing coordinatewise. In subsection 4.3 we show that the distance $d_M(t,t')$ between t and t' in the minor graph M can be bounded by $\log k \cdot f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}})$, the scaled cost function applied on the charges. This section is devoted to proving the following lemma.

LEMMA 4.2. $\Pr[f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}) \geq 43 \cdot d_G(t,t')] \leq k^{-3}$.

Using Claim 4.1, one can show that for every $Q \in \mathcal{Q}$, $\mathbb{E}[\tilde{X}(Q)] = O(1)$, and moreover, w.h.p. $\tilde{X}(Q) = O(\log k)$ for all Q. However, we use a concentration bound on all $\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}$ simultaneously in order to provide a stronger upper bound.

4.2.1. Bounding by independent variables. In our journey to bound $f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}})$, the first step will be to replace $\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}$ with independent random variables. Consider the following process: a *box* B which contains coins of two types, active and inactive. In the beginning, there is a single active coin. In each round, we toss an active coin, which gets 0 (failure) with probability p, and 1 (success) with probability $1-p$. If we get a 0, two additional active coins are added to the box. In any case, the tossed coin becomes inactive. All the coin tosses throughout the process are independent. The process terminates when no active coins remain. Let $\{B_Q\}_{Q\in\mathcal{Q}}$ be a set of $|\mathcal{Q}|$ independent boxes (here the box B_Q resembles the interval Q). For the box B_Q, denote by $Z(Q)$ the number of active coins, by $Y(Q)$ the number of inactive coins, and by $\tilde{Y}(Q)$ the number of inactive coin at the end of the process.

CLAIM 4.3. *For every* $\alpha \in \mathbb{R}_+$,

$$\Pr\left[f\left(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}\right) \geq \alpha\right] \leq \Pr\left[f\left(\{\tilde{Y}(Q)\}_{Q\in\mathcal{Q}}\right) \geq \alpha\right] .$$

Proof. The proof is done by coupling the two processes of Algorithm 3.1 and the coin tosses. We execute Algorithm 3.1, which implicitly induces slices and detour charges. Simultaneously, we will use Algorithm 3.1 to toss coins. Inductively, we will maintain the invariant that $\{Y(Q)\}_{Q\in\mathcal{Q}}$ and $\{Z(Q)\}_{Q\in\mathcal{Q}}$ are no less than $\{X(Q)\}_{Q\in\mathcal{Q}}$ and $\{S(Q)\}_{Q\in\mathcal{Q}}$ (respectively) coordinatewise.

In the beginning $\{X(Q)\}_{Q\in\mathcal{Q}} = \{Y(Q)\}_{Q\in\mathcal{Q}} = \{0\}_{Q\in\mathcal{Q}}$ and $\{S(Q)\}_{Q\in\mathcal{Q}} = \{Z(Q)\}_{Q\in\mathcal{Q}} = \{1\}_{Q\in\mathcal{Q}}$. Consider round j, where the cluster V_j is created for the terminal t_j. If $R_j < r_{v^j}$, then nothing happens, and the invariant holds. Else, $R_j \geq r_{v^j}$, we will make a coin toss from the B_{Q_j} box. Let p' be the probability that not all of S_j joins V_j. By Claim 4.1, $p' \leq p$. If indeed not all of S_j joins V_j, the toss result is set to 0. Otherwise, with probability $\frac{p-p'}{1-p'}$ the toss is set to 0. Note that the probability of 0 is exactly $p' \cdot 1 + (1-p') \cdot \frac{p-p'}{1-p'} = p$.

Next we argue that the invariant is maintained in either case. If not all of S_j joins Q_j, then $S(Q_j)$ might increase by at most one, while the number of active coins Z_{Q_j} increases by exactly one. Otherwise, all of S_j joins Q_j. In this case $S(Q_j)$ necessarily decreases by at least one, while Z_{Q_j} might either decrease or increase by one. For the charge parameter, $X(Q_j)$ might increase by at most one, while the number of inactive coins $Y(Q_j)$ increases by exactly one. For every $Q' \neq Q_j$, $\mathcal{S}(Q')$ and $X(Q')$ might

only decrease, while $Z_{Q'}$ and $Y(Q')$ stay unchanged. We conclude that the invariant holds after the construction of the cluster V_j.

Intuitively speaking, creating a cluster for a terminal t_j is a global processes that can involve many slices in different terminals, the crux being that only the interval Q_j is charged, and only the slice S_j might get splitted. For all other intervals, charges can only get erased and slices eliminated. The process of coin tosses in the boxes imitates charge and slice counting, while ignoring the potential savings.

At the end of the algorithm (when no slices are left), we might still have some active coins. In this case we will simply toss coins until no active coins remain (note that this indeed happens with probability 1). Note that by doing so $\{Y(Q)\}_{Q\in\mathcal{Q}}$ can only grow coordinatewise. As the marginal distribution on $\{\tilde{Y}(Q)\}_{Q\in\mathcal{Q}}$ is exactly identical to the original one, the claim follows. ∎

4.2.2. Replacing coins with exponential random variables. Our next step is to replace each $Y(Q)$ with exponential random variable. This replacement will make the use of concentration bounds more convenient. Consider some box B_Q. An equivalent way to describe the probabilistic process in B_Q is the following. Take a single coin with failure probability p, and toss this coin until the number of successes exceeds the number of failures. The total number of tosses is exactly $\tilde{Y}(Q)$. Note that $\tilde{Y}(Q)$ is necessarily odd. Next we bound the probability that $\tilde{Y}(Q) \geq 2m+1$ for $m \geq 1$. This is obviously upper bounded by the probability that in a series of $2m$ tosses we had at least m failures (as otherwise the process would have stopped earlier). Let χ_i be an indicator for a failure in the ith toss, and $\chi = \sum_{i=1}^{2m} \chi_i$. Note that $\mathbb{E}[\chi] = 2m \cdot p$. A bound on χ follows by the Chernoff inequality.

Fact 1 (Chernoff inequality). Let $X_1, \ldots, X_n$ be independent and identically distributed (i.i.d.) indicator variables each with probability p. Set $X = \sum_i X_i$ and $\mu = \mathbb{E}[X] = np$. Then for every $\delta \leq 2e-1$, $\Pr[X \geq (1+\delta)\mu] \leq \exp(-\mu\delta^2/4)$.

$$\begin{aligned}\Pr\left[\tilde{Y}(Q) \geq 2m+1\right] &\leq \Pr[\chi \geq m] = \Pr\left[\chi \geq \left(1+\left(\frac{1}{2p}-1\right)\right)\mathbb{E}[\chi]\right] \\ &\leq \exp\left(-2m\cdot p\cdot\left(\frac{1}{2p}-1\right)^2/4\right) = \exp\left(-\frac{9}{40}m\right) \\ &\leq \exp\left(-\frac{1}{5}m\right)\ .\end{aligned}$$

We conclude that the distribution of $\tilde{Y}(Q)$ is dominated by $1+\mathsf{Exp}(10)$ (as for $W \sim \mathsf{Exp}(10)$, $\Pr[1+W \geq 2m+1] = \exp\left(-\frac{m}{5}\right)$). Let $(\{W(Q)\}_{Q\in\mathcal{Q}})$ be i.i.d. random variables distributed according to $\mathsf{Exp}(10)$; since all the boxes are independent and f is linear and monotone coordinatewise, we conclude as follows.

Claim 4.4. *For every* $\alpha \in \mathbb{R}_+$,

$$\Pr\left[f\left(\left\{\tilde{Y}(Q)\right\}_{Q\in\mathcal{Q}}\right) \geq \alpha\right] \leq \Pr\left[f\left(\{1\}_{Q\in\mathcal{Q}}\right) + f\left(\{W(Q)\}_{Q\in\mathcal{Q}}\right) \geq \alpha\right].$$

Proof. Set $\varphi = |\mathcal{Q}|$. Let $Q^1, Q^2, \ldots, Q^\varphi$ be some arbitrarily fixed ordering of the intervals. For $s \in [\varphi]$, set $f_{\setminus\{s\}}(x_1, \ldots, x_{s-1}, x_{s+1}, \ldots, x_\varphi) = \sum_{i\in[\varphi]\setminus\{s\}} x_i \cdot L^+(Q^i)$. When integrating over the appropriate measure space, it holds that

$$
\begin{aligned}
&\Pr\left[f\left(\tilde{Y}(Q^1),\dots,\tilde{Y}(Q^\varphi)\right)\geq\alpha\right]\\
&=\int_\beta \Pr\left[f_{\setminus\{1\}}\left(\tilde{Y}(Q^2),\dots,\tilde{Y}(Q^\varphi)\right)=\beta\right]\\
&\qquad\qquad\cdot\Pr\left[\tilde{Y}(Q^1)\cdot L^+(Q^1)\geq\alpha-\beta\right]d\beta\\
&\leq\int_\beta \Pr\left[f_{\setminus\{1\}}\left(\tilde{Y}(Q^2),\dots,\tilde{Y}(Q^\varphi)\right)=\beta\right]\\
&\qquad\qquad\cdot\Pr\left[\left(1+W(Q^1)\right)\cdot L^+(Q^1)\geq\alpha-\beta\right]d\beta\\
&=\Pr\left[f\left(1+W(Q^1),\tilde{Y}(Q^2),\dots,\tilde{Y}(Q^\varphi)\right)\geq\alpha\right]\\
&\leq\Pr\left[f\left(1+W(Q^1),1+W(Q^2),\tilde{Y}(Q^3),\dots,\tilde{Y}(Q^\varphi)\right)\geq\alpha\right]\\
&\leq\cdots\leq\Pr\left[f\left(1+W(Q^1),\dots,1+W(Q^\varphi)\right)\geq\alpha\right]\\
&=\Pr\left[f\left(1,\dots,1\right)+f\left(W(Q^1),\dots,W(Q^\varphi)\right)\geq\alpha\right]\ .
\end{aligned}
$$

∎

4.2.3. Concentration. Set $\Delta=d_G(t,t')$. It holds that

$$\Delta\leq\sum_{Q\in\mathcal{Q}}L^+(Q)\leq 2\Delta\ ,$$

as every edge in $P_{t,t'}$ is counted at least once, and at most twice in this sum. In particular $f(\{1\}_{Q\in\mathcal{Q}})\leq 2\Delta$. Recall that by our modification step, every edge in $P_{t,t'}$ is of weight at most $c_w\cdot\Delta$. In particular, for every $Q\in\mathcal{Q}$, $L^+(\mathcal{Q})\leq L(\mathcal{Q})+2c_w\cdot\Delta$. For every vertex v on $P_{t,t'}$, it holds that $D(v)\leq\min\{d_G(v,t),d_G(v,t')\}\leq\frac{\Delta}{2}$. Therefore for every $Q\in\mathcal{Q}$,

$$L^+(\mathcal{Q})\leq L(\mathcal{Q})+2c_w\cdot\Delta\overset{(4.1)}{\leq}c_{\text{int}}\delta\cdot D(Q)+2c_w\cdot\Delta\leq\left(\frac{c_{\text{int}}\delta}{2}+2c_w\right)\cdot\Delta=c_{\text{int}}\delta\cdot\Delta\ .$$

Let $\tilde{W}(Q)\sim L^+(Q)\cdot\mathsf{Exp}(10)$. In particular, $\tilde{W}(Q)\sim\mathsf{Exp}(10\cdot L^+(Q))$. Set $\tilde{W}=\sum_{Q\in\mathcal{Q}}\tilde{W}(Q)$. Then $f(\{W(Q)\}_{Q\in\mathcal{Q}})$ is distributed exactly as $\tilde{W}$. The maximal mean among the $\tilde{W}(Q)$'s is $\lambda_M=\max_{Q\in\mathcal{Q}}10\cdot L^+(Q)\leq 10\cdot c_{\text{int}}\delta\cdot\Delta$. The mean of $\tilde{W}$ is $\mu=\sum_{Q\in\mathcal{Q}}10\cdot L^+(Q)\leq 20\Delta$. Set $c_{\text{con}}=\frac{1}{2}$ (con for concentration). Using Claim 4.3, Claim 4.4, and Lemma 2.1, we conclude

$$
\begin{aligned}
&\Pr\left[f\left(\left\{\tilde{X}(Q)\right\}_{Q\in\mathcal{Q}}\right)\geq(c_{\text{con}}+42)\Delta\right]\\
&\leq\Pr\left[f\left(\left\{\tilde{Y}(Q)\right\}_{Q\in\mathcal{Q}}\right)\geq(c_{\text{con}}+42)\Delta\right]\\
&\leq\Pr\left[f\left(\{W(Q)\}_{Q\in\mathcal{Q}}\right)\geq(c_{\text{con}}+42)\Delta-f\left(\{1\}_{Q\in\mathcal{Q}}\right)\right]\\
&\leq\Pr\left[\tilde{W}\geq(c_{\text{con}}+40)\Delta\right]\\
&\leq\exp\left(-\frac{1}{2\lambda_M}\left(\left(c_{\text{con}}+40\right)\Delta-2\mu\right)\right)\\
&\leq\exp\left(-\frac{1}{2}\cdot\frac{1}{10c_{\text{int}}\delta\Delta}\cdot c_{\text{con}}\Delta\right)=\exp\left(-\frac{c_{\text{con}}}{20\cdot c_{\text{int}}\delta}\right)=k^{-3}\ .
\end{aligned}
$$

Note that $c_{\text{con}}\leq 1$, thus Lemma 4.2 follows.

4.3. Bounding the distortion. Denote by $\mathcal{E}^{\text{fBig}}$ the event that for some pair of terminals t, t', $f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}) \geq 43 \cdot d_G(t,t')$.[2] By Lemma 4.2 and the union bound, $\Pr[\mathcal{E}^{\text{fBig}}] \leq \binom{k}{2} \cdot k^{-3} < \frac{1}{2k}$.

Let $\mathcal{E}^{\text{B}}$ be the event that for some j, $R_j > c_d$, where $c_d = e^2$. Note that if $\mathcal{E}^{\text{B}}$ does not hold, then every vertex v joins to a cluster V_j such that $d_G(v,t_j) \leq c_d \cdot D(v)$.

CLAIM 4.5. $\Pr[\mathcal{E}^{B}] \leq \frac{1}{2k}$.

Proof. Let $\mathcal{E}^{\text{B}}_j$ be the event that $R_j > c_d$. It holds that

$$\Pr[\mathcal{E}^{\text{B}}_j] = \Pr[g_j \geq \log_{1+\delta} c_d] \leq (1-p)^{\log_{1+\delta} c_d - 1} \leq (1-p)^{\frac{2}{\delta}-1} \leq \frac{1}{k^3} ,$$

where the second inequality holds as $\log_{1+\delta} c_d = \frac{\ln c_d}{\ln 1+\delta} \geq \frac{2}{\delta}$. By the union bound, $\Pr[\mathcal{E}^{\text{B}}] \leq \frac{1}{k^2} \leq \frac{1}{2k}$ as required. ∎

LEMMA 4.6. *Assuming $\overline{\mathcal{E}^{B}}$ and $\overline{\mathcal{E}^{fBig}}$, for every pair of terminals t, t', $d_M(t,t') \leq O(\log k) \cdot d_G(t,t')$.*

Proof. Fix some t, t'. By the end of Algorithm 3.1, all the vertices in $P_{t,t'} = \{t = v_0, \ldots, v_\gamma = t'\}$ are divided into consecutive detours[3] $\mathcal{D}_{\ell_1}, \ldots, \mathcal{D}_{\ell_{k'}}$. The detour $\mathcal{D}_{\ell_j}$ was constructed at round ℓ_j by the terminal t_{ℓ_j}. The detour $\mathcal{D}_{\ell_j}$ was charged upon the interval Q_{ℓ_j}, which contains the vertex v^{ℓ_j}. The leftmost vertex in $\mathcal{D}_{\ell_j}$ is called a_{ℓ_j}, while the rightmost vertex is called b_{ℓ_j}. In particular, for every $1 \leq j \leq k'-1$, there is an edge in G between b_{ℓ_j} and $a_{\ell_{j+1}}$, and therefore there is an edge between t_{ℓ_j} to $t_{\ell_{j+1}}$ in the terminal-centered minor M. As $t = v_0$ joins the cluster of itself, necessarily $t_{\ell_1} = t$. Similarly $t_{\ell_{k'}} = t'$. See Figure 4.2 for an illustration. Using the triangle inequality, we conclude

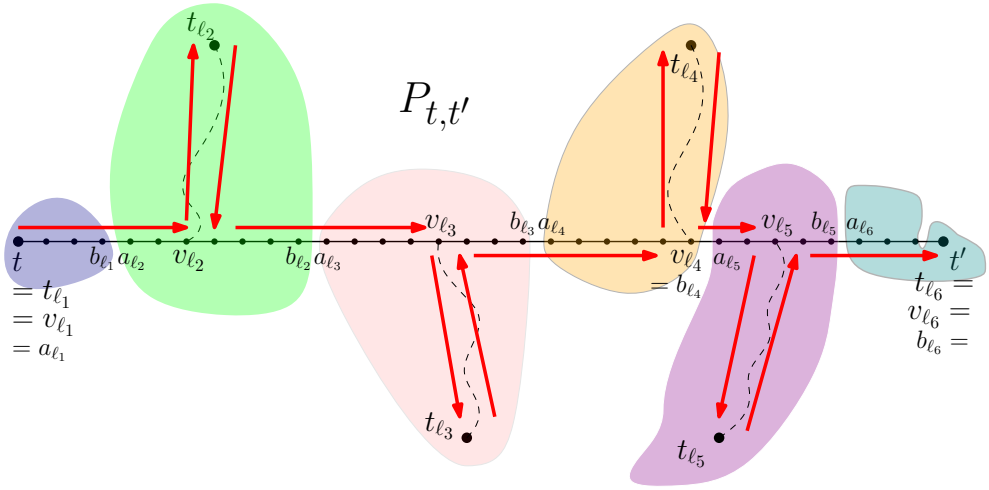

FIG. 4.2. *The vertices $P_{t,t'} = v_0 \ldots v_\gamma$ are divided into consecutive detours $\mathcal{D}_{\ell_1}, \ldots, \mathcal{D}_{\ell_6}$. $t_{\ell_1}, t_{\ell_2}, t_{\ell_3}, t_{\ell_4}, t_{\ell_5}, t_{\ell_6}$ is a path in the terminal-centered minor M of G (induced by $V_1, \ldots, V_k$). The weight of the edge $\{t_{\ell_j}, t_{\ell_{j+1}}\}$ in M is $d_G(t_{\ell_j}, t_{\ell_{j+1}})$, which is bounded by $d_G(t_{\ell_j}, v_{\ell_j}) + d_G(v_{\ell_j}, v_{\ell_{j+1}}) + d_G(v_{\ell_{j+1}}, t_{\ell_{j+1}})$.*

[2] We abuse notation here and use the same $\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}$ for all terminals.

[3] Note that we consider only detours that inflict a charge by the end of the algorithm. Therefore the detours are disjoint and every vertex in $P_{t,t'}$ belongs to some detour.

$$d_M(t,t') \le \sum_{j=1}^{k'-1} d_G(t_{\ell_j}, t_{\ell_{j+1}}) \le \sum_{j=1}^{k'-1} \left[d_G(t_{\ell_j}, v^{\ell_j}) + d_G(v^{\ell_j}, v^{\ell_{j+1}}) + d_G(v^{\ell_{j+1}}, t_{\ell_{j+1}}) \right]$$
$$\le \sum_{j=1}^{k'-1} d_G(v^{\ell_j}, v^{\ell_{j+1}}) + 2\sum_{j=1}^{k'} d_G(t_{\ell_j}, v^{\ell_j})$$
$$\le d_G(t,t') + 2\sum_{j=1}^{k'} c_d \cdot D(v^{\ell_j}),$$

where the last inequality follows by our assumption $\overline{\mathcal{E}^{\mathrm{B}}}$. By the definition of $D(Q_{\ell_j})$, inequality (4.1) and the triangle inequality, $D(v^{\ell_j}) \le D(Q_{\ell_j}) + L(Q_{\ell_j}) \le (\frac{1}{c_{\mathrm{int}}\delta} + 1)$ $L^+(Q_{\ell_j}) \le \frac{2}{c_{\mathrm{int}}\delta} \cdot L^+(Q_{\ell_j})$. Using the assumption $\overline{\mathcal{E}^{\mathrm{fBig}}}$, we conclude

$$d_M(t,t') \le d_G(t,t') + 2c_d \sum_{i=1}^{k'} \frac{2}{c_{\mathrm{int}}\delta} \cdot L^+(Q_{\ell_i}) \tag{4.6}$$
$$= d_G(t,t') + \frac{4c_d}{c_{\mathrm{int}}\delta} \sum_{Q\in\mathcal{Q}} \tilde{X}(Q) \cdot L^+(Q)$$
$$= d_G(t,t') + \frac{4c_d}{c_{\mathrm{int}}\delta} \cdot f\left(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}\right) = O(\ln k) \cdot d_G(t,t') \ .$$

As $\Pr\left[\overline{\mathcal{E}^{\mathrm{B}}} \wedge \overline{\mathcal{E}^{\mathrm{fBig}}}\right] \ge 1 - (\Pr[\mathcal{E}^{\mathrm{B}}] + \Pr[\mathcal{E}^{\mathrm{fBig}}]) \ge 1 - \frac{1}{2k} - \frac{1}{2k} = 1 - \frac{1}{k}$, Theorem 3.1 follows. ∎

5. `Fast-Relaxed-Voronoi` algorithm. In this section, we describe a slightly modified version of the `Relaxed-Voronoi` algorithm. Then we will show how to implement the modified algorithm in $O(m \log n)$ time.

Given two terminals t_i, t_j, and two clusters $V_i, V_j \subseteq V$ s.t. t_i (resp., t_j) is the unique terminal in V_i (resp., V_j), $d_{G,V_i+V_j}(t_i,t_j)$ denotes the length of the shortest path between t_i and t_j in $G[V_i \cup V_j]$ that uses exactly one crossing edge between V_i and V_j. See Figure 5.1 for an illustration.

In order to allow fast implementation, and avoid costly shortest path computations, we will introduce several modifications:

- In Algorithm 3.1, line 9, we will modify the edge weights in the induced terminal-centered minor. The weight of the edge $\{t_i, t_j\}$ (if exists) will be $d_{G,V_i+V_j}(t_i,t_j)$ instead of $d_G(t_i,t_j)$.

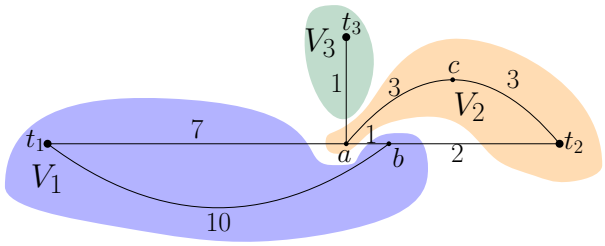

FIG. 5.1. *t_1, t_2, t_3 are terminals. The different color areas describes the terminal partition. The shortest path in G from t_1 to t_2 is t_1, a, b, t_2 and has length $d_G(t_1,t_2) = 10$. Note that all the vertices in this path are in $V_1 \cup V_2$. Nevertheless, the shortest path from t_1 to t_2 that uses only one crossing edge from t_1 to t_2 is $\{t_1, b, t_2\}$ and has length $d_{G,V_1+V_2}(t_1,t_2) = 12$.*

Algorithm 5.1. $M = \texttt{Fast-Relaxed-Voronoi}(G = (V, E, w), K = \{t_1, \ldots, t_k\})$.

1: Set $\delta = \frac{1}{20 \ln k}$ and $p = \frac{1}{5}$.
2: Set $V_\perp \leftarrow V \setminus K$. *// $V_\perp$ is the currently unclustered vertices.*
3: **for** j from 1 to k **do**
4: Choose independently at random g_j distributed according to $\mathsf{Geo}(p)$.
5: Set $R_j \leftarrow (1+\delta)^{g_j}$.
6: Set $V_j \leftarrow \texttt{Fast-Create-Cluster}(G, V_\perp, t_j, R_j)$.
7: Remove all the vertices in V_j from $V_\perp$.
8: **end for**
9: Let M be the minor of G created by contracting all the internal edges in $V_1, \ldots, V_k$. The weight of the edge $\{t_i, t_j\}$ (if it exists) is defined to be $d_{G, V_i + V_j}(t_i, t_j)$.
10: **return** M.

Algorithm 5.2. $V_j = \texttt{Fast-Create-Cluster}(G = (V, E, w), V_\perp, t_j, R_j)$.

1: Set $V_j \leftarrow \{t_j\}$.
2: Set $U \leftarrow \emptyset$. *// U is the set of vertices already denied from V_j.*
3: Set N to be all the neighbors of t_j in $V_\perp$.
4: **while** $N \neq \emptyset$ **do**
5: Let $v \in N$ be the vertex with minimal $d_{G[V_j \cup \{v\}]}(v, t_j)$.
6: Remove v from N.
7: **if** $d_{G[V_j \cup \{v\}]}(v, t_j) \leq R_j \cdot D(v)$ **then**
8: Add v to V_j.
9: Add all the neighbors of v in $V_\perp \setminus U$ to N.
10: **else**
11: Add v to U.
12: **end if**
13: **end while**
14: **return** V_j.

- In Algorithm 3.2, line 5, instead of extracting an arbitrary vertex v from N, we will extract the closest vertex v to t_j in N w.r.t. the shortest path metric induced by $V_j \cup \{v\}$ (i.e., $v \in N$ with minimal $d_{G[V_j \cup \{v\}]}(v, t_j)$, and note that it is a different graph for each vertex).
  Similarly, in line 7, instead of checking whether $d_G(v, t_j) \leq R_j \cdot D(v)$, we will check whether $d_{G[V_j \cup \{v\}]}(v, t_j) \leq R_j \cdot D(v)$.

The pseudocode of the modified algorithm appears in Algorithms 5.1 and 5.2.

THEOREM 5.1. *With probability $1 - \frac{1}{k}$, for the minor graph M returned by Algorithm* 5.1*, it holds that for every two terminals t, t', $d_M(t, t') \leq O(\log k) \cdot d_G(t, t')$. Moreover, executing Algorithm* 5.1 *takes $O(m + \min\{m, nk\} \cdot \log n)$ time.*

We prove Theorem 5.1 in several steps. First, in subsection 5.1 we show that Algorithm 5.1 indeed returns a terminal partition and that similarly to Algorithm 3.1, the edge subdivision does not change the outcome of the algorithm. Then in subsection 5.2 we'll go through the analysis provided in section 4 and verify that it still goes through for Algorithm 5.1 as well. Finally, in subsection 5.3 we describe an efficient implementation of Algorithm 5.1.

5.1. Basic properties. Consider the `Fast-Create-Cluster` procedure (Algorithm 5.2). This is a Dijkstra-like algorithm. For every vertex v, set $\ell_v = d_{G[V_j \cup \{v\}]}(v, t_j)$. Note that for a vertex v, the value ℓ_v is decreasing throughout the algorithm as the set V_j grows. Note also that ℓ_v is defined for all the vertices (but simply has value ∞ for vertices out of $V_j \cup N$). Denote by $\hat{\ell}_v$ the value ℓ_v at the time v is extracted from N at line 6 of Algorithm 5.2 (if such an occasion indeed occurs).

CLAIM 5.2. *Consider the values $\hat{\ell}_v$ of the vertices, extracted from N at line 6 of Algorithm 5.2. Then these values are nondecreasing. That is, if v was extracted before v', then $\hat{\ell}_v \le \hat{\ell}_{v'}$.*

Moreover, after v is extracted, the value ℓ_v remains unchanged till the end of the algorithm.

Proof. The proof of the first property is by induction on the execution of the algorithm. Let v, v' be a pair of vertices such that v' was extracted from N right after v. It will be enough to show that $\hat{\ell}_v \le \hat{\ell}_{v'}$. Consider the time when v was extracted from N. Let $\tilde{V}_j$ denote the set V_j at that time. By minimality, for every $u \in N$, $\hat{\ell}_v = d_{G[\tilde{V}_j \cup \{v\}]}(v, t_j) \le d_{G[\tilde{V}_j \cup \{u\}]}(u, t_j)$. If the value $\ell_{v'}$ did not change, we already have $\hat{\ell}_{v'} = d_{G[\tilde{V}_j \cup \{v'\}]}(v', t_j) \ge \hat{\ell}_v$ (as necessarily $v' \in N$ because it is extracted next). Otherwise, if the value $\ell_{v'}$ decreased, then necessarily v joined V_j and the shortest path from from t_j to v' (in $\tilde{V}_j \cup \{v, v'\}$) goes through v (as otherwise $\ell_{v'}$ would not have changed). In particular, $\hat{\ell}_{v'} = d_{G[\tilde{V}_j \cup \{v,v'\}]}(t_j, v') = d_{G[\tilde{V}_j \cup \{v,v'\}]}(t_j, v) + d_{G[\tilde{V}_j \cup \{v,v'\}]}(v, v') > \hat{\ell}_v$.

For the second property (that after extraction, ℓ_v remains unchanged), seeking contradiction, assume that ℓ_v is updated after some u is extracted from N and joined V_j. This implies that the new shortest path from t_j to v goes through u and thus is of length greater than $\hat{\ell}_u$, a contradiction. ∎

Now we are ready to show that Algorithm 5.1 indeed returns a terminal partition (that is, reprove Lemma 3.2).

LEMMA 5.3. *The sets $V_1, \ldots, V_k$ constructed by Algorithm 5.1 form a terminal partition.*

Proof. It is clear that the clusters $V_1, \ldots, V_j$ are disjoint and that each cluster is connected. It will be enough to argue that every vertex $v \in V$ is clustered. Following along the lines of the proof of Lemma 3.2, let t_j be the closest terminal to v, and let $P = \{t_j = u_0, u_1, \ldots, u_s = v\}$ be the shortest path from t_j to v. Let $u_{i'}$ be the first vertex from $P_{t,t'}$ to be clustered during the algorithm ($u_0 = t_j \in V_j$, so at least one vertex in $P_{t,t'}$ is clustered). Let $V_{j'}$ be the cluster $u_{i'}$ joins to. We argue by induction on $i \ge i'$ that u_i also joins $V_{j'}$. This will imply that $u_s = v$ joins $V_{j'}$ and thus is clustered.

Suppose u_i joins $V_{j'}$. Denote by $V^i_{j'}$ the set $V_{j'}$ right after u_i joins it. As u_i joins $V_{j'}$, $d_{G[V^i_{j'}]}(u_i, t_{j'}) \le R_{j'} \cdot D(u_i)$. In particular, at that stage

$$\begin{aligned}\ell_{u_{i+1}} = d_{G\left[V^i_{j'} \cup \{u_{i+1}\}\right]}(u_{i+1}, t_{j'}) &\le d_{G\left[V^i_{j'}\right]}(u_i, t_{j'}) + w\left(\{u_i, u_{i+1}\}\right) \\ &\le R_{j'} \cdot D(u_i) + d_G(u_i, u_{i+1}) \le R_{j'} \cdot D(u_{i+1}).\end{aligned}$$

As at least one neighbor (u_i) of u_{i+1} joins $V_{j'}$, u_{i+1} joins N at some stage of the algorithm. In particular, by Claim 5.2, when u_{i+1} will be extracted from N, $\hat{\ell}_{u_{i+1}} \leq R_{j'} \cdot D(u_{i+1})$, and thus u_{i+1} will join $V_{j'}$ as required. ∎

We will use the modified graph $\hat{G}$ (with the subdivided edges) for the distortion analysis. In order to prove validity, we will argue that Claim 3.3 still holds.

CLAIM 5.4. *In Claim* 3.3*, if we replace Algorithm* 3.1 *with Algorithm* 5.1*, the claim still holds.*

Proof. We follow the lines of the proof of Claim 3.3. Let $V_1, \ldots, V_k$ (resp., $\tilde{V}_1, \ldots, \tilde{V}_k$) be the terminal partition induced by Algorithm 5.1 on G (resp., $\tilde{G}$). We argue that for all j, $V_j = \tilde{V}_j \setminus \{v_e\}$. As previously, this will imply that the terminal-centered minors have the same edges set. As v_e only subdivides the edge e, it will also hold for all i, j that $d_{G,V_i+V_j}(t_i, t_j) = d_{G,\tilde{V}_i+\tilde{V}_j}(t_i, t_j)$, and thus the edge weights in both minors will also be identical. In particular, the claim will follow.

Suppose w.l.o.g. that v joins V_j while u is still unclustered. Denote by V'_j (resp., $\tilde{V}'_j$) the set V_j (resp., $\tilde{V}_j$) right after the clustering of v at the execution of Algorithm 5.1 on G (resp., $\tilde{G}$). As previously, for all $j'' < j$, $V_{j''} = \tilde{V}_{j''}$, while $V'_j = \tilde{V}'_j$.

Recall that $\hat{\ell}_v = d_{G[V'_j](t_j,v)}$ (resp., $\tilde{\hat{\ell}}_v$) denotes the distance between t_j to v at the time of the extraction of v from N (resp. $\tilde{N}$). Note that $\hat{\ell}_v = \tilde{\hat{\ell}}_v$. As v joins V_j, necessarily $\hat{\ell}_v \leq R_j \cdot D(v)$. In the rest of the proof we consider the following cases:

- $\hat{\ell}_u > R_j \cdot D(v)$: In this case u will not join V_j. As v_e has edges only to v and u, v_e has no impact on any other vertex. In particular, $\hat{\ell}_u \leq \tilde{\hat{\ell}}_u$. Therefore $\tilde{V}_j$ will be constructed in the same manner as V_j (up to maybe containing v_e). Note that all the other clusters will not be affected, as if v_e remained unclustered, it becomes a leaf. We conclude that for every j', $V_{j'} = \tilde{V}_{j'} \setminus \{v_u\}$.
- $\hat{\ell}_u \leq R_j \cdot D(v)$: Recall that ω is the weight of e. There are two subcases:
  - $\hat{\ell}_u = \hat{\ell}_v + \omega$. After v joins $\tilde{V}_j$, the label of v_e is updated to $\hat{\ell}_{v_e} \leftarrow \tilde{\hat{\ell}}_v + \frac{\omega}{2}$. It holds that

$$\begin{aligned}\tilde{\hat{\ell}}_{v_e} \leq \tilde{\ell}_{v_e} = \tilde{\hat{\ell}}_v + \frac{\omega}{2} = \hat{\ell}_v + \frac{\omega}{2} = \frac{1}{2}\left(\hat{\ell}_v + \hat{\ell}_u\right) \\ \leq \frac{1}{2} \cdot R_j \left(D(v) + D(u)\right) \leq R_j \cdot D(e_v) \ .\end{aligned}$$

    In particular, v_e will join $\tilde{V}_j$, and $\tilde{\ell}_u$ will be updated to $\tilde{\hat{\ell}}_{v_e} + \frac{\omega}{2} = \tilde{\hat{\ell}}_v + \omega$. From this point on, the two algorithms will behave in the same way. In particular, for every $j'' \neq j$, $V_{j''} = \tilde{V}_{j''}$ while $V_j \cup \{v_e\} = \tilde{V}_j$.
  - $\hat{\ell}_u < \hat{\ell}_v + \omega$. It holds that u joins V_j. However, the shortest path in V_j from t_j to u did not goes through v. Therefore, as v_e did not affect any vertex (other than v, u), the execution will proceed in the same way in both algorithms, and u will join $\tilde{V}_j$. As each cluster is connected and all the vertices are clustered, necessarily v_e will join $\tilde{V}_j$ as well. We conclude that for every $j'' \neq j$, $V_{j''} = \tilde{V}_{j''}$ while $V_j \cup \{v_e\} = \tilde{V}_j$. ∎

5.2. Distortion analysis. We will follow the distortion analysis of Algorithm 3.1 given in section 4. Consider two terminals t, t'. We will use the exact same notation (the reader is referred to Appendix C in order to recall notation and definitions). We start by reproving Claim 4.1.

CLAIM 5.5. *During the execution of Algorithm* 5.1*, assuming* $R_j \geq r_{v^j}$*, all of* S_j *joins* V_j *with probability at least* $1-p$*.*

Proof. Denote $S_j = \{u_{j-q'}, \ldots, u_j, \ldots, u_{j+q}\} \subseteq Q_j \subseteq P_{t,t'}$ where $v^j = u_j$. Denote by V_j' the cluster V_j right after u_j joins. As u_j joined, necessarily $\frac{d_{G[V_j' \cup \{u_j\}]}(u_j, t_j)}{D(u_j)} \leq r_{v^j} \leq R_j$. We will denote by $\bar{V}_j$ the cluster V_j at the end of the algorithm. Following inequality (4.3), with probability $1-p$, $R_j \geq (1+\delta) r_{v^j}$. We will show that if this event indeed occurs, then $S_j \subseteq \bar{V}_j$.

We argue by induction on i that $u_{j+i} \in \bar{V}_j$. The proof that $u_{j-i} \in \bar{V}_j$ is symmetric. Assume that $\{u_i, u_{i+1}, \ldots, u_{j+i-1}\} \subseteq \bar{V}_j$. Following inequalities (4.4) and (4.5), $L(Q_j) \leq 2c_{\text{int}}\delta \cdot D(v^j)$ and $D(u_{j+i}) \geq D(v^j)\,(1-2c_{\text{int}}\delta)$. As $u_{i+j-1} \in \bar{V}_j$, u_{j+i} necessarily joins N at some stage. In particular, at the time u_{j+i} was extracted from N,

$$\hat{\ell}_{u_{j+i}} = d_{G[\bar{V}_j \cup \{u_{j+i}\}]}(t_j, u_{j+i}) \leq d_{G[V_j']}(t_j, v^j) + L(Q_j) \leq d_{G[V_j']}(t_j, v^j)\,(1+2c_{\text{int}}\delta)\ ,$$

where the first equality follows by Claim 5.2, as $\hat{\ell}_{u_{j+i}}$ remains unchanged after extraction. We conclude that

$$\frac{\hat{\ell}_{u_{j+i}}}{D(u_{j+i})} \leq \frac{d_{G[V_j']}(t_j, v^j)\,(1+2c_{\text{int}}\delta)}{D(v^j)\,(1-2c_{\text{int}}\delta)} \leq \frac{d_{G[V_j']}(t_j, v^j)}{D(v^j)}\,(1+3\cdot 2c_{\text{int}}\delta) \leq (1+\delta)\,R_j\ .$$

We conclude that u_{j+i} joins V_j as required. ∎

In subsection 4.2 we defined charge function $f(\{x_Q\}_{Q\in\mathcal{Q}}) = \sum_{Q\in\mathcal{Q}} X(Q)\cdot L^+(Q)$, and in Lemma 4.2 we upper bounded its value (w.h.p.). In that analysis we exploit only Claim 4.1. Replacing it with Claim 5.5, the analysis still hold. That is, $\Pr[f(\{\tilde{X}(Q)\}_{Q\in\mathcal{Q}}) \geq 43\cdot d_G(t,t')] \leq k^{-3}$. Denote by $\mathcal{E}^{\text{fBig}}$ the event that for some pair of terminals t, t', $f(\tilde{X}(Q^1), \ldots, \tilde{X}(Q^\varphi)) \geq 43\cdot d_G(t,t')$. As previously, by union bound $\Pr[\mathcal{E}^{\text{fBig}}] < \frac{1}{2k}$. Denote by $\mathcal{E}^{\text{B}}$ the event that for some j, $R_j > c_d$. By Claim 4.5, $\Pr[\mathcal{E}^{\text{B}}] \leq \frac{1}{2k}$. We argue that assuming $\overline{\mathcal{E}^{\text{B}}}$ and $\overline{\mathcal{E}^{\text{fBig}}}$ (which happens with probability $1-\frac{1}{k}$), the distance between every pair of terminals t, t' in the minor returned by Algorithm 5.1 bounded by $O(\log k)\cdot d_G(v,u)$. This will conclude the proof of the distortion argument in Theorem 5.1. Recall that in contrast to Algorithm 3.1, the weight of the edge $\{t_i, t_j\}$ (if it exists) is $d_{G, V_i+V_j}(t_i, t_j)$ rather than $d_G(t_i, t_j)$; this will force some changes to our analysis. Recall the notation we used in Lemma 4.6: the path $P_{t,t'}$ is divided into consecutive detours $\mathcal{D}_{\ell_1}, \ldots, \mathcal{D}_{\ell_{k'}}$. The leftmost (resp., rightmost) vertex in $\mathcal{D}_{\ell_j}$ is denoted by a_{ℓ_j} (resp., b_{ℓ_j}). Both a_{ℓ_j}, b_{ℓ_j} belong to V_{ℓ_j}, the cluster of t_{ℓ_j}. In particular, the graph G contains an edge between b_{ℓ_j} to $a_{\ell_{j+1}}$. Recall also that $t_{\ell_1} = t$ and $t_{\ell_k'} = t'$ (as each terminal covers itself). It holds that

$$
\begin{aligned}
d_M(t,t') &\le \sum_{j=1}^{k'-1} d_{G,V_{\ell_j}+V_{\ell_{j+1}}}(t_{\ell_j}, t_{\ell_{j+1}}) \\
&\le \sum_{j=1}^{k'-1} \left[d_{G[V_{\ell_j}]}(t_{\ell_j}, b_{\ell_j}) + d_G(b_{\ell_j}, a_{\ell_{j+1}}) + d_{G[V_{\ell_{j+1}}]}(a_{\ell_{j+1}}, t_{\ell_{j+1}}) \right] \\
&\le c_d \cdot \sum_{j=1}^{k'-1} \left[d_G(t_{\ell_j}, b_{\ell_j}) + d_G(b_{\ell_j}, a_{\ell_{j+1}}) + d_G(a_{\ell_{j+1}}, t_{\ell_{j+1}}) \right] \\
&\le c_d \cdot \sum_{j=1}^{k'-1} \left[d_G(t_{\ell_j}, v^{\ell_j}) + d_G(v^{\ell_j}, b_{\ell_j}) + d_G(b_{\ell_j}, a_{\ell_{j+1}}) \right. \\
&\qquad\qquad \left. + d_G(a_{\ell_{j+1}}, v^{\ell_{j+1}}) + d_G(v^{\ell_{j+1}}, t_{\ell_{j+1}}) \right] \\
&\le c_d \cdot \left(\sum_{j=1}^{k'-1} d_G(v^{\ell_j}, v^{\ell_{j+1}}) + 2 \sum_{j=1}^{k'} d_G(t_{\ell_j}, v^{\ell_j}) \right) \\
&\le c_d \cdot \left(d_G(t,t') + 2c_d \cdot \sum_{j=1}^{k'} D(v^{\ell_j}) \right) \\
&= O(\ln k) \cdot d_G(t,t') \ .
\end{aligned}
$$

The third inequality follows by our assumption $\overline{\mathcal{E}^{\mathrm{B}}}$, as for every index j and vertex $v \in V_j$, it holds that $d_{G[V_j]}(t_j, v) \le c_d \cdot D(v) \le c_d \cdot d_G(t_j, v)$. The fifth inequality follows as all $v^{\ell_j}, b_{\ell_j}, a_{\ell_{j+1}}, v^{\ell_{j+1}}$ lie on the same shortest path $P_{t,t'}$. The sixth inequality follows by $\overline{\mathcal{E}^{\mathrm{B}}}$ as $d_G(t_{\ell_j}, v^{\ell_j}) \le d_{G[V_{\ell_j}]}(t_{\ell_j}, v^{\ell_j}) \le c_d \cdot D(v^{\ell_j})$. The equality follows by inequality (4.6) and $\overline{\mathcal{E}^{\mathrm{fBig}}}$.

5.3. Runtime. For the implementation of Algorithm 5.1 and the `Fast-Create-Cluster` procedure we will use two basic data structures. The first one is a binary array to determine set membership of the vertices. It is folklore (see, for example, [1]) that an array could be initialized in constant time to be the all 0 array (that is, the empty set). Changing entry (that is, adding or deleting an element) also takes constant time. The second data structure is the Fibonacci heap (see [22]). Here each element has a key (some real number), and we can add a new element or decrease the value of the key in constant time. Finding the minimal element in the heap and deleting it takes $O(\log h)$ time (assuming there are currently h elements in the heap).

Before the execution of Algorithm 5.1, we compute the values $D(v)$ for all $v \in V$. This is done using an auxiliary graph G' where we add new vertex s with edges of weight 0 to all the terminals. Note that for every vertex v, the distance from s exactly equals $D(v)$. Thus we can simply run the Dijkstra algorithm from s to determine $D(v)$ for all $v \in V$. The runtime is $O(m + n \log n)$ (see [22]).

Next we give a detailed implementation of the `Fast-Create-Cluster` procedure. The sets V_j, U, and $V_\perp$ are stored using the arrays described above ($V_\perp$ will be a global variable). The set N will be stored using the Fibonacci heap, where the key value of $v \in N$ will be ℓ_v (i.e., $d_{G[V_j \cup \{v\}]}(v, t_j)$). Denote by $\mathcal{N}_j$ all the elements that belong to N at any stage of the execution of the `Fast-Create-Cluster` procedure (which created V_j). Let m_j denote the number of edges incident on vertices of V_j.

Each iteration of the while loop starts by deleting an element v with minimal key (of value $\hat{\ell}_v$) from N ($O(\log|\mathcal{N}_j|)$ time). Then we examine whether to add v to V_j (in $O(1)$ time). If v is rejected, we add v to U (in $O(1)$ time). Otherwise, v is added to V_j. In the latter case we go over each neighbor u of v. If $u \in U$ we do nothing. If $u \in N$, its key ℓ_u is updated to be $\min\{\ell_u, \ell_v + w(\{v,u\})\}$. Finally, if $u \in V_\perp \setminus (U \cup N)$, then u is added to N with the key $\ell_u \leftarrow \ell_v + w(\{v,u\})$. It is easy to verify that all the keys are indeed maintained with the correct values. Note that all this processing for u takes only $O(1)$ time. In particular, processing all neighbors throughout the `Fast-Create-Cluster` procedure takes $O(m_j)$ time. All the deletion of elements from the heap N takes $O(|\mathcal{N}_j|\log|\mathcal{N}_j|)$ time.

Next we bound the total cost of the k calls to the `Fast-Create-Cluster` procedure. $|\mathcal{N}_j|$ can be bounded from above by both m_j and n. Moreover, $\sum_j m_j \le 2m$, as every edge is incident on only two vertices. We provide two upper bounds on the running time:

$$O(n) + \sum_{j=1}^{k} O(m_j + |\mathcal{N}_j|\log|\mathcal{N}_j|) \le O\left(m + \sum_{j=1}^{k} m_j \log n\right) = O(m\log n)\ ,$$

$$O(n) + \sum_{j=1}^{k} O(m_j + |\mathcal{N}_j|\log|\mathcal{N}_j|) \le O\left(m + \sum_{j=1}^{k} n\log n\right) = O(m + nk\log n)\ .$$

Thus the total running time of these k calls is bounded by $O(m + \min\{m, nk\} \cdot \log n)$. Finally we bound the total runtime of Algorithm 5.1 without the calls to the `Create-Cluster`. It is straightforward that up line 9, where we create the minor M given the clusters, all computations took $O(n)$ time.[4] Using Claim 5.2, by the end of the for loop in Algorithm 5.1, for every j and $v \in V_j$ it holds that $\hat{\ell}_v = d_{G[V_j]}(t_j, v)$. In order to create the minor graph M, we go over all the edges iteratively, for every edge $\{v,u\} \in E$, such that $v \in V_j$, $u \in V_i$, and $i \neq j$. We add an edge $\{t_i, t_j\}$ to M (if it does not exist already). The weight of the edge updated to be the minimum between the current weight (∞ if it does not exist yet) and $\hat{\ell}_v + w(\{v,u\}) + \hat{\ell}_u$ (the keys at the time of extraction from N). It is straightforward that by the end of this procedure we will indeed compute the minor M, and each edge $\{t_i, t_j\}$ in M will have weight $d_{G,V_i+V_j}(t_i, t_j)$. This iterative process takes $O(m)$ time. Theorem 5.1 now follows.

6. Lower bounds on the performance of the algorithms. Chan et al. [9] gave a lower bound of 8 for the distortion in the SPR problem. This lower bound has not been improved since. This section is dedicated to lower bounding the performance of the various algorithms which were suggested for the problem. That is, while we do not provide better lower bounds for the SPR problem itself, we are able to lower bound the performance of the algorithms used so far.

In subsection 6.1 we prove that our analysis of the `Relaxed-Voronoi` algorithm (Algorithms 3.1 and 5.1) is asymptotically tight. That is, there is a graph family on which the achieved distortion is $\Theta(\log k)$. Next, in subsection 6.2, we provide a lower bound on the performance of the `Ball-growing` algorithm studied by [27, 11, 20]. Specifically, we provide (the same) graph family on which the `Ball-growing` algorithm incurs $\Omega(\sqrt{\log k})$ distortion. Recall that in [20], the author proved that the `Ball-growing` algorithm finds a minor with distortion $O(\log k)$. That is, while the

[4] In fact, the sampling of $g_1, \ldots, g_k$ takes $O(k)$ time only w.h.p. But we will ignore this issue.

analysis of the `Ball-growing` algorithm still might be improved, it cannot be pushed further than $\Omega(\sqrt{\log k})$.

First, we show that the *expected* distortion incurred by the minor returned by the algorithms is large. Then, we deduce that with constant probability the (usual worst-case) distortion is also large. Formally, both algorithms are randomized and thus can be viewed as producing a distribution $\mathcal{D}$ over graph minors. Given such distribution $\mathcal{D}$, the expected distortion of the pair t, t' is $\mathbb{E}_{M\sim\mathcal{D}}\left[\frac{d_M(t,t')}{d_G(t,t')}\right]$. The overall expected distortion is the maximal expected distortion among all terminal pairs.

A final remark. Both algorithms used an arbitrary order over the terminals, in contrast to similar algorithms for other problems [8, 19] which consider a random order. Our lower bounds will still hold even if one replaces the arbitrary order with a random one.

6.1. Lower bound on the performance of the `Relaxed-Voronoi` algorithm. The following theorem provides a lower bound on the expected distortion incurred by Algorithm 3.1. The graphs which we will use for the lower bound are trees. As both Algorithm 3.1 and Algorithm 5.1 are identical where the input graph is a tree, the lower bound will also hold on Algorithm 5.1.

THEOREM 6.1. *Fix some $k \in \mathbb{N}$. There is a graph $G = (V, E, w)$ with terminal set K of size k such that the expected distortion of the minor returned by Algorithm* 3.1 *is* $\Omega(\log k)$.

Proof. We will assume that k is large enough, as otherwise $1 = \Omega(\log k)$ and hence every graph with k terminals provides a valid lower bound. Let G_k be the graph described in Figure 1.1 with parameter $\epsilon = 14\delta = \Theta(\frac{1}{\log k})$. Let X_j be an indicator for the event $v_j \in V_j$, that is, t_j covers v_j. For X_j to occur, it is enough that for every $i \neq j$, $d_G(t_i, v_j) > R_i \cdot D(v_j)$. That is, $R_i < 1 + |i-j| \cdot \epsilon$. By the definition of R_i,

$$\Pr\left[R_i \geq 1 + |i-j|\,\epsilon\right] = \Pr\left[g_i \geq \log_{1+\delta}\left(1 + |i-j|\,\epsilon\right)\right] = (1-p)^{\left\lceil \log_{1+\delta}(1+|i-j|\epsilon)-1\right\rceil} .$$

For i such that $|i-j| < \frac{1}{\epsilon}$, it holds that $\log_{1+\delta}\left(1 + |i-j|\,\epsilon\right) = \frac{\ln(1+|i-j|\epsilon)}{\ln(1+\delta)} \geq \frac{|i-j|\epsilon/2}{\delta}$, while for i such that $|i-j| \geq \frac{1}{\epsilon}$, $\log_{1+\delta}\left(1 + |i-j|\,\epsilon\right) \geq \frac{\ln 2}{\ln 1+\delta} \geq \frac{1}{2\delta}$. We conclude

$$\begin{aligned}
\Pr\left[X_i\right] &\geq \Pr\left[\forall_{j\neq i}\left(R_j < 1 + |i-j|\,\epsilon\right)\right] \\
&\geq 1 - \sum_{j\neq i} \Pr\left[R_j \geq 1 + |i-j|\,\epsilon\right] \\
&\geq 1 - 2\sum_{i=1}^{\left\lfloor \frac{1}{\epsilon}\right\rfloor} \left(\left(1-p\right)^{\frac{i\epsilon/2}{\delta}-1}\right) - k\left(1-p\right)^{\frac{1}{2\delta}-1} .
\end{aligned}$$

Now, $\sum_{i=1}^{\left\lfloor \frac{1}{\epsilon}\right\rfloor}\left(1-p\right)^{\frac{i\epsilon/2}{\delta}} \leq \sum_{i=1}^{\infty}((1-p)^7)^i \leq \sum_{i=1}^{\infty}\frac{1}{4^i} = \frac{1}{4}\frac{1}{1-\frac{1}{4}} = \frac{1}{3}$, while $k\left(1-p\right)^{\frac{1}{2\delta}} = k\left(\frac{4}{5}\right)^{10\ln k} = k^{1-10\ln\frac{5}{4}} \leq \frac{1}{k}$. In particular $\Pr\left[X_i\right] \geq 1 - (1-p)^{-1}\cdot\left(2\cdot\frac{1}{3} + \frac{1}{k}\right) = \Omega(1)$.

Set $X = \sum_{i=2}^{k-1} X_i$. By linearity of expectation, $\mathbb{E}[X] = \Omega(k)$. Note that the distance from t_1 to t_k in the minor graph M_k equals $2 + (k-1)\,\epsilon + 2X$. We conclude

$$\mathbb{E}\left[\frac{d_{M_k}(t_1,t_m)}{d_{G_k}(t_1,t_m)}\right] = \frac{2+(k-1)\,\epsilon + 2\mathbb{E}\left[X\right]}{2+(k-1)\,\epsilon} = \frac{\Omega(k)}{O(k\epsilon)} = \Omega\left(\frac{1}{\epsilon}\right) = \Omega(\log k) .$$ ∎

COROLLARY 6.2. *Fix some $k \in \mathbb{N}$. There is a graph $G = (V, E, w)$ with terminal set K of size k such that with constant probability, the distortion incurred by the minor returned by Algorithm* 3.1 *is $\Omega(\log k)$.*

Proof. We will use the graph and notation from the proof of Theorem 6.1. Set $\mu = \mathbb{E}[\frac{d_{M_k}(t_1,t_m)}{d_{G_k}(t_1,t_m)}] = \Omega(\log k)$. Note the largest possible distortion is $\frac{2k-2+(k-1)\epsilon}{2+(k-1)\epsilon} = c \cdot \mu$ for some constant $c \geq 1$ (this distortion occurred exactly when each vertex v_j belongs to V_j). Denote by χ the event that $\frac{d_{M_k}(t_1,t_m)}{d_{G_k}(t_1,t_m)} \geq \frac{1}{2}\mu$. Then

$$\mu = \mathbb{E}\left[\frac{d_{M_k}(t_1,t_m)}{d_{G_k}(t_1,t_m)}\right] \leq \Pr[\chi] \cdot c\mu + (1 - \Pr[\chi]) \cdot \frac{1}{2}\mu\,,$$

therefore

$$\Pr[\chi] \geq \frac{1-\frac{1}{2}}{c-\frac{1}{2}} \geq \frac{1}{2c} = \Omega(1)\,.$$

Therefore, with constaint probability, the distortion is at least $\frac{1}{2}\mu = \Omega(\log k)$. ∎

6.2. Lower bound on the performance of the `Ball-Growing` algorithm. In this subsection we provide a lower bound on the performance of the `Ball-Growing` algorithm. For completeness, we give in Appendix B a full description of the `Ball-Growing` algorithm as it appeared in [20]. In particular, we will use the notation defined there. The `Ball-Growing` as described in [20] also had a modification step. As our lower bound example is a tree, this modification has no impact on the minor returned by the algorithm, and thus we can ignore it. Formally, a claim similar to Claim 3.3 can be proven.

THEOREM 6.3. *Fix some $k \in \mathbb{N}$. There is a graph $G = (V, E, w)$ with terminal set K of size k such that the expected distortion of the minor returned by the `Ball-Growing` algorithm is $\Omega(\sqrt{\log k})$.*

Proof. We will use the graph described in Figure 1.1 with modified parameters: the weight of an edge between terminal to Steiner vertex will be $2-\epsilon$, while the weight of an edge between two Steiner vertices will be 2ϵ for ϵ to be specified later. Note that the `Ball-Growing` algorithm assumes that the minimal distance between a terminal to a Steiner vertex in the input graph is exactly 1. In order to satisfy this condition we will add an additional Steiner vertex as a leaf connected to t_1 via an edge of unit weight. Note that this new vertex has no impact on the resulting minor whatsoever and therefore can be completely ignored.

As previously, we denote by X_j the indicator for the event $v_j \in V_j$. Following the analysis of Theorem 6.3, if we prove that $\Pr[X_j] = \Omega(1)$ (for arbitrary j) it will imply expected distortion of $\Omega(\frac{1}{\epsilon})$.

Let $\mathcal{R}_j$ be equal to R_j (the magnitude of t_j) at the end of the $m = \log_r 3 - 1$ round. For simplicity we will assume that m is an integer; otherwise the analysis will go through after slight modification of the parameters. Recall that $\mathcal{R}_j = \sum_{\ell=0}^{m} q_j^\ell$ where q_j^ℓ is distributed according to $\mathrm{Exp}(D \cdot r^\ell)$. Here $r = 1 + \frac{\delta}{\ln k}$, $\delta = \frac{1}{20}$, $D = \frac{\delta}{\ln k}$, and all the q_j^ℓ are independent. It holds that

$$\mathbb{E}\left[\mathcal{R}_j\right] = \sum_{\ell=0}^{m} D \cdot r^\ell = D \cdot \frac{r^{m+1}-1}{r-1} = 2\,,$$

$$\mathbb{V}\left[\mathcal{R}_j\right] = \mathbb{V}\left[\sum_{\ell=0}^{m} q_j^\ell\right] = \sum_{\ell=0}^{m} \mathbb{V}\left[q_j^\ell\right] = \sum_{\ell=0}^{m} \left(D \cdot r^\ell\right)^2$$
$$= D^2 \cdot \frac{r^{2(m+1)}-1}{r^2-1} = \left(\frac{\delta}{\ln k}\right)^2 \cdot \frac{9-1}{2 \cdot \frac{\delta}{\ln k} + \left(\frac{\delta}{\ln k}\right)^2} \le 4 \cdot \frac{\delta}{\ln k} = O\left(\frac{1}{\ln k}\right)\,,$$

where we used linearity of expectation and independence. In order that X_j will occur, it is enough that $\mathcal{R}_j \ge d(t_j, v_j)$, while for every $j' \ne j$, $\mathcal{R}_j < d(t_{j'}, v_j)$. Using the Chebyshev inequality,

$$\Pr\left[\mathcal{R}_j \ge d(t_j, v_j)\right] = \Pr\left[\mathcal{R}_j \ge 2-\epsilon\right] \ge \Pr\left[\left|\mathcal{R}_j - \mathbb{E}\left[\mathcal{R}_j\right]\right| < \epsilon\right] \ge 1 - \frac{\mathbb{V}\left[\mathcal{R}\right]}{\epsilon^2}\,,$$

$$\Pr\left[\mathcal{R}_{j'} \ge d(t_{j'}, v_j)\right] \le \Pr\left[\left|\mathcal{R}_{j'} - \mathbb{E}\left[\mathcal{R}_{j'}\right]\right| \ge \left(2\left|j-j'\right|-1\right)\epsilon\right] \le \frac{\mathbb{V}\left[\mathcal{R}\right]}{\left(2\left|j-j'\right|-1\right)^2 \cdot \epsilon^2}\,.$$

By the union bound, the probability that for some $j' \ne j$, $\mathcal{R}_{j'} \ge d(t_{j'}, v_j)$ is bounded by

$$\sum_{j \ne j'} \Pr\left[\mathcal{R}_{j'} \ge d(t_{j'}, v_j)\right] < \frac{\mathbb{V}\left[\mathcal{R}\right]}{\epsilon^2} \cdot 2 \cdot \sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\mathbb{V}\left[\mathcal{R}\right]}{\epsilon^2} \cdot \frac{\pi^2}{3}\,.$$

We conclude

$$\Pr\left[X_j\right] \ge \Pr\left[\mathcal{R}_{j'} \ge d(t_{j'}, v_j)\right] \cdot \left(1 - \sum_{j \ne j'} \Pr\left[\mathcal{R}_{j'} \ge d(t_{j'}, v_j)\right]\right)$$
$$\ge \left(1 - \frac{\mathbb{V}\left[\mathcal{R}\right]}{\epsilon^2}\right)\left(1 - \frac{\mathbb{V}\left[\mathcal{R}\right]}{\epsilon^2} \cdot \frac{\pi^2}{3}\right) = 1 - O\left(\frac{1}{\epsilon^2 \ln k}\right) = \Omega(1)$$

for $\epsilon = \Theta(\frac{1}{\sqrt{\log k}})$. The theorem now follows. ∎

Following the lines of the proof of Corollary 6.2, we conclude as follows.

COROLLARY 6.4. *Fix some $k \in \mathbb{N}$. There is a graph $G = (V, E, w)$ with terminal set K of size k such that with constant probability, the distortion of the minor returned by the* `Ball-Growing` *algorithm is $\Omega(\sqrt{\log k})$*

Remark 6.5. Theorem 6.3 can also be proved using concentration bounds. However, the lower bound remains $\Omega(\sqrt{\log k})$ so we provided the more basic proof using the Chebyshev inequality. Nevertheless, the curious reader can find the required concentration bounds for such a proof in Appendix A.

7. Discussion. In this paper we proved an $O(\log k)$ upper bound for the SPR problem, improving the previous $O(\log^2 k)$ upper bound by [11]. The lower bound is still only 8 [9]. Closing this gap remains an intriguing open problem. Both the `Relaxed-Voronoi` and `Ball-growing` algorithms proceed by creating random terminal partitions. These partitions are determined using random parameters, which are chosen with no consideration whatsoever of the input graph G. In contrast, the optimal tree algorithm of [24] is a deterministic recursive algorithm which make decisions

after considering the tree structure at hand. It seems that the input-oblivious approach of the `Relaxed-Voronoi` and `Ball-growing` algorithms is doomed for failure, and in fact, both these algorithms already fail to achieve constant distortion on a simple tree example. As a conclusion, input-sensitive approaches seem to be more promising for future attempts to resolve the SPR problem.

In a follow-up paper with Krauthgamer and Trabelsi [21], we used the `Relaxed-Voronoi` algorithm in order to re-prove Gupta's [24] upper bound of 8. Formally, let $r \in V$ be an arbitrary vertex and order the terminals w.r.t. their distances from r (that is, $d(t_1, r) \leq d(t_2, r) \leq \ldots d(t_k, r)$). Surprisingly, given a tree, if we run the `Relaxed-Voronoi` algorithm w.r.t. the order specified above (instead of an arbitrary order), and all magnitudes R_j are exactly 3, we will get a tree minor with distortion at most 8. This example demonstrates that one can use the `Relaxed-Voronoi` algorithm also in an input-sensitive manner in order to achieve optimal results.

We would like to emphasize two additional open problems:

- Expected distortion: Currently the state of the art for usual (worst-case) distortion and expected distortion for the SPR problem is the same. Both have $O(\log k)$ upper bound and $\Omega(1)$ lower bound. There are cases where much better results can be achieved for expected distortion (e.g., embedding a graph into a tree must incur distortion $\Omega(n)$, while a distribution over embeddings into trees can have expected distortion $O(\log n)$ [19]). What are the right bounds for expected distortion in the SPR problem?
- Special graph families: Basu and Gupta [5] showed that constant distortion for the SPR problem can be achieved on outer-planar graphs. It will be very interesting to achieve better upper bounds for planar graphs, and more generally for minor-free graphs, bounded treewidth graphs, etc. In the expected distortion regime, an $O(1)$ upper bound is already known [17] for minor-free graphs. Possibly one can use the `Relaxed-Voronoi` algorithm with a clever choice of order and magnitudes in order to achieve such results.

Appendix A. Concentration bounds for sum of exponential distributions.

LEMMA A.1. *Suppose $X_1, \ldots, X_n$'s are independent random variables, where each X_i is distributed according to $\mathsf{Exp}(\lambda_i)$. Let $X = \sum_i X_i$ and $\lambda_M = \max_i \lambda_i$. Set $\mu = \mathbb{E}[X] = \sum_i \lambda_i$.*
For $0 < t \leq \frac{1}{2\lambda_M}$, and $\alpha \geq 2t\lambda_M$,

$$\Pr[X \geq (1+\alpha)\mu] \leq \exp\left(-t\mu \cdot (\alpha - 2t\lambda_M)\right),$$
$$\Pr[X \leq (1-\alpha)\mu] \leq \exp\left(-t\mu\,(\alpha - t\lambda_M)\right) \ .$$

Proof. For each X_i, the moment generating function w.r.t. t equals

$$\mathbb{E}\left[e^{tX_i}\right] = \frac{1}{1-t\lambda_i} = 1 + t\lambda_i \left(\sum_{\ell \geq 0} (t\lambda_i)^\ell\right) \leq 1 + t\lambda_i\,(1 + 2t\lambda_i) \leq e^{t\lambda_i(1+2t\lambda_i)}.$$

Using the Markov inequality,

$$\begin{aligned}\Pr\left[X \geq (1+\alpha)\mu\right] &= \Pr\left[e^{tX} \geq e^{t(1+\alpha)\mu}\right] \\ &\leq \mathbb{E}\left[e^{tX}\right] \cdot e^{-t(1+\alpha)\mu} \\ &= e^{-t(1+\alpha)\sum_\ell \lambda_\ell} \cdot \prod_\ell \mathbb{E}\left[e^{tX_\ell}\right] \\ &\leq e^{-(1+\alpha)\sum_\ell t\lambda_\ell} \cdot e^{\sum_\ell t\lambda_\ell(1+2t\lambda_\ell)} \\ &= e^{\sum_\ell (t\lambda_\ell \cdot (2t\lambda_\ell - \alpha))} \\ &\leq e^{\left(\sum_\ell t\lambda_\ell\right)\cdot(2t\lambda_M-\alpha)} = e^{-t\mu\cdot(\alpha-2t\lambda_M)},\end{aligned}$$

where in the second equality we use the fact that $\{X_i\}_i$ are independent.

For the second inequality, it holds that

$$\mathbb{E}\left[e^{-tX_i}\right] = \frac{1}{1+t\lambda_i} = \sum_{\ell\geq 0}(-1)^\ell (t\lambda_i)^\ell \leq 1 - t\lambda_i(1-t\lambda_i) \leq e^{-t\lambda_i(1-t\lambda_i)} .$$

Therefore,

$$\begin{aligned}\Pr\left[X \leq (1-\alpha)\mu\right] &= \Pr\left[e^{-tX} \geq e^{-t(1-\alpha)\mu}\right] \\ &\leq \mathbb{E}\left[e^{-tX}\right] / e^{-t(1-\alpha)\mu} \\ &= e^{t(1-\alpha)\mu} \cdot \Pi_\ell \mathbb{E}\left[e^{-tX_\ell}\right] \\ &\leq e^{(1-\alpha)\sum_\ell t\lambda_\ell} \cdot e^{-\sum_\ell t\lambda_\ell(1-t\lambda_\ell)} \\ &= e^{-\sum_\ell t\lambda_\ell(\alpha - t\lambda_\ell)} \\ &\leq e^{-t\mu(\alpha-t\lambda_M)} .\end{aligned}$$

∎

We derive the following corollary.

COROLLARY A.2. *Suppose $X_1, \ldots, X_n$ are independent random variables, where $X_i \sim \mathsf{Exp}(\lambda_i)$. Let $X = \sum_i X_i$ and $\lambda_M = \max_i \lambda_i$. Set $\mu = \mathbb{E}[X] = \sum_i \lambda_i$. Then,*

$$\text{For } \alpha \leq 2: \ \Pr\left[X \geq (1+\alpha)\mu\right] \leq \exp\left(-\frac{\alpha^2\mu}{8\lambda_M}\right) ,$$

$$\text{For } \alpha \leq 1: \ \Pr\left[X \leq (1-\alpha)\mu\right] \leq \exp\left(-\frac{\alpha^2\mu}{4\lambda_M}\right) .$$

For the first inequality we choose the parameter $t = \frac{\alpha}{2} \cdot \frac{1}{2\lambda_M}$, while for the second inequality we choose the parameter $t = \alpha \cdot \frac{1}{2\lambda_M}$.

Appendix B. The `Ball-Growing` algorithm. The `Ball-Growing` algorithm assumes w.l.o.g. that the minimal distance between a terminal to a Steiner vertex in the input graph is exactly 1. Throughout the execution of the algorithm each terminal t_j is associated with a radius R_j and cluster $V_j \subset V$. The algorithm iteratively grows clusters $V_1, \ldots, V_k$ around the terminals. Once some vertex v joins some cluster V_j, it will stay there. When all the vertices are clustered, the algorithm terminates. Initially the cluster V_j contains only the terminal t_j, while R_j equals 0. The algorithm will have rounds, where each round consist of k steps. In step j of round ℓ, the algorithm samples a number q_j^ℓ according to distribution $\mathsf{Exp}(D \cdot r^\ell)$ (note that the mean of the distribution grows by a factor of r in each round). The radius R_j grows by q_j^ℓ. We consider the graph induced by the unclustered vertices $V_\perp$ union V_j. Every unclustered vertex of distance at most R_j from t_j in $G[V_\perp \cup V_j]$ joins V_j.

Algorithm B.1. $M = \texttt{Ball-Growing}(G = (V, E), w, K = \{t_1, \dots, t_k\})$.

Set $r \leftarrow 1 + \delta / \ln k$, where $\delta = 1/20$.
Set $D \leftarrow \frac{\delta}{\ln k}$.
For each $j \in [k]$, set $V_j \leftarrow \{t_j\}$, and set $R_j \leftarrow 0$.
Set $V_\perp \leftarrow V \setminus \left(\cup_{j=1}^k V_j\right)$.
Set $\ell \leftarrow 0$.
while $\left(\cup_{j=1}^k V_j\right) \neq V$ **do**
  for j from 1 to k **do**
    Choose independently at random q_j^ℓ distributed according to $\mathsf{Exp}(D \cdot r^\ell)$.
    Set $R_j \leftarrow R_j + q_j^\ell$.
    Set $V_j \leftarrow B_{G[V_\perp \cup V_j]}(t_j, R_j)$. . *// This is the same as* $V_j \leftarrow V_j \cup B_{G[V_\perp \cup V_j]}(t_j, R_j)$.
    Set $V_\perp \leftarrow V \setminus \left(\cup_{j=1}^k V_j\right)$.
  end for
  $\ell \leftarrow \ell + 1$.
end while
return the terminal-centered minor M of G induced by $V_1, \dots, V_k$.

Appendix C. Index.

Preliminaries.

d_G: shortest path metric in G.
$G[A]$: graph induced by A.
$K = \{t_1, \dots, t_k\}$: set of terminals.
$D(v) = \min_{t \in K} d_G(v, t)$.
Terminal partition: partition $\{V_1, \dots, V_k\}$ of V, s.t. for every i, $t_i \in V_i$ and V_i is connected.
Induced minor: given terminal partition $\{V_1, \dots, V_k\}$, the induced minor obtained by contracting each V_i into the super vertex t_i. The weight of the edge $\{t_i, t_j\}$ (if it exists) set to be $d_G(t_i, t_j)$.
Distortion of induced minor: $\max_{i,j} \frac{d_M(t_i, t_j)}{d_G(t_i, t_j)}$.
$\mathsf{Geo}(p)$: geometric distribution with parameter λ.
$\mathsf{Exp}(\lambda)$: exponential distribution with parameter p.

Modification. Every edge on $P_{t,t'}$ has weight at most $c_w \cdot d_G(t, t')$.

Constants.

$p = \frac{1}{5}$: parameter of the geometric distribution.
$\delta = \frac{1}{20 \cdot \ln k}$: jumps in R_j are of magnitude $1 + \delta$.
$c_w = \frac{\delta}{24}$.
$c_{\mathbf{int}} = \frac{1}{6}$: governs the size of interval in the partition $\mathcal{Q}$ of $P_{t,t'}$.
$c_{\mathbf{con}} = \frac{1}{2}$: used to bound the variation of the charge function from its expectation.
$c_d = e^2$: bound on the maximal size of R_j.

Events.

$\mathcal{E}^{\mathrm{fBig}}$: denotes that for some pair of terminals t, t', $f(\{X(Q)\}_{Q \in \mathcal{Q}} \geq 43 \cdot d_G(t, t')$.
$\mathcal{E}^{\mathrm{B}}$: denotes that there exist j such that $R_j > c_d$.

Notation.

V_j: cluster of t_j.
R_j: magnitude of the cluster of t_j.
$V_\perp$: set of unclustered (uncovered) vertices.
$P_{t,t'} = \{t = v_0, \dots, v_\gamma = t'\}$: shortest path from t to t'.
$L(\{v_a, v_{a+1}, \dots, v_b\}) = d_G(v_a, v_b)$: internal length.
$L^+(\{v_a, v_{a+1}, \dots, v_b\}) = d_G(v_{a-1}, v_{b+1})$: external length.
$\mathcal{Q}$: partition of $P_{t,t'}$ into intervals Q.
a_j: the leftmost active vertex covered by t_j.
b_j: the rightmost active vertex covered by t_j.
$\mathcal{D}_j = \{a_j, \dots, b_j\}$: detour created by terminal t_j.
Slice maximal subinterval (of some Q) of active vertices.
r_v: minimal choice of R_j such that v joins V_j.
v^j: vertex with the minimal r_v (among active vertices).
Q_j: interval containing v_j.
S_j: slice containing v_j.
$f(\{x_Q\}_{Q \in \mathcal{Q}}) = \sum_{Q \in \mathcal{Q}} x_Q \cdot L^+(Q)$, charge function.
B_Q: a coin box which resembles the interval Q.
$d_{G, V_i + V_j}(t_i, t_j)$: the weight of the shortest path in G between t_1 and t_2 that uses only vertices from $V_i \cup V_j$ and only a single crossing edge between V_i to V_j.

Counters.

$\mathcal{S}(Q)$: (current) number of slices in interval Q.
$X(Q)$: number of detours the interval Q is (currently) charged for.
$\tilde{X}(Q)$: number of detours the interval Q is charged for by the end of Algorithm 3.1.
$Z(Q)$: number of active coins in B_Q. Each coin is active when added to the box.
$Y(Q)$: number of inactive coins in B_Q. A coin becomes inactive after tossing.
$\tilde{Y}(Q)$: number of inactive coins in B_Q by the end of the process.

Acknowledgment. The author would like to thank his advisors Ofer Neiman, for fruitful discussions, and Robert Krauthgamer, for useful comments.

REFERENCES

[1] A. V. Aho and J. E. Hopcroft, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Boston, MA, 1974.

[2] A. Andoni, A. Gupta, and R. Krauthgamer, *Towards* $(1+\epsilon)$*-approximate flow sparsifiers*, in Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, Portland, OR, 2014, pp. 279–293, https://doi.org/10.1137/1.9781611973402.20.

[3] Y. Bartal, *Probabilistic approximations of metric spaces and its algorithmic applications*, in Proceedings of the 37th Annual Symposium on Foundations of Computer Science, Burlington, VT, 1996, pp. 184–193, https://doi.org/10.1109/SFCS.1996.548477.

[4] Y. Bartal, A. Filtser, and O. Neiman, *On notions of distortion and an almost minimum spanning tree with constant average distortion*, in Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, VA, 2016, pp. 873–882, https://doi.org/10.1137/1.9781611974331.ch62.

[5] A. Basu and A. Gupta, *Steiner Point Removal in Graph Metrics*, manuscript, http://www.math.ucdavis.edu/~abasu/papers/SPR.pdf (2008).

[6] J. D. Batson, D. A. Spielman, and N. Srivastava, *Twice-Ramanujan sparsifiers*, SIAM J. Comput., 41 (2012), pp. 1704–1721, https://doi.org/10.1137/090772873.

[7] A. A. Benczúr and D. R. Karger, *Approximating* s-t *minimum cuts in* $\tilde{O}(n^2)$ *time*, in Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, Philadelphia, PA, 1996, pp. 47–55, https://doi.org/10.1145/237814.237827.

[8] G. Călinescu, H. J. Karloff, and Y. Rabani, *Approximation algorithms for the* 0*-extension problem*, SIAM J. Comput., 34 (2004), pp. 358–372.

[9] T.-H. Chan, D. Xia, G. Konjevod, and A. Richa, *A tight lower bound for the Steiner point removal problem on trees*, in Proceedings of the 9th International Conference on Approximation Algorithms for Combinatorial Optimization Problems, and 10th International Conference on Randomization and Computation, Springer-Verlag, Berlin, 2006, pp. 70–81, https://doi.org/10.1007/11830924_9.

[10] M. Charikar, T. Leighton, S. Li, and A. Moitra, *Vertex sparsifiers and abstract rounding algorithms*, in Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, Las Vegas, NV, 2010, pp. 265–274, https://doi.org/10.1109/FOCS.2010.32.

[11] Y. K. Cheung, *Steiner point removal—distant terminals don't (really) bother*, in Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 2018, pp. 1353–1360.

[12] Y. K. Cheung, G. Goranci, and M. Henzinger, *Graph minors for preserving terminal distances approximately—lower and upper bounds*, in 43rd International Colloquium on Automata, Languages, and Programming, Rome, Italy, 2016, pp. 131:1–131:14, https://doi.org/10.4230/LIPIcs.ICALP.2016.131.

[13] J. Chuzhoy, *On vertex sparsifiers with Steiner nodes*, in Proceedings of the 44th Symposium on Theory of Computing Conference, New York, 2012, pp. 673–688, https://doi.org/10.1145/2213977.2214039.

[14] D. Coppersmith and M. Elkin, *Sparse source-wise and pair-wise distance preservers*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, PA, 2005, pp. 660–669, http://dl.acm.org/citation.cfm?id=1070432.1070524.

[15] M. Elkin, A. Filtser, and O. Neiman, *Prioritized metric structures and embedding*, in Proceedings of the 47th Annual ACM on Symposium on Theory of Computing, Portland, OR, 2015, pp. 489–498, https://doi.org/10.1145/2746539.2746623.

[16] M. Elkin, A. Filtser, and O. Neiman, *Terminal embeddings*, Theoret. Comput. Sci., 697 (2017), pp. 1–36, https://doi.org/10.1016/j.tcs.2017.06.021.

[17] M. Englert, A. Gupta, R. Krauthgamer, H. Räcke, I. Talgam-Cohen, and K. Talwar, *Vertex sparsifiers: New results from old techniques*, SIAM J. Comput., 43 (2014), pp. 1239–1262, https://doi.org/10.1137/130908440.

[18] J. Fakcharoenphol, C. Harrelson, S. Rao, and K. Talwar, *An improved approximation algorithm for the* 0*-extension problem*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, 2003, pp. 257–265, http://dl.acm.org/citation.cfm?id=644108.644153.

[19] J. Fakcharoenphol, S. Rao, and K. Talwar, *A tight bound on approximating arbitrary metrics by tree metrics*, J. Comput. System Sci., 69 (2004), pp. 485–497, https://doi.org/10.1016/j.jcss.2004.04.011.

[20] A. Filtser, *Steiner point removal with distortion* O*(log* k*)*, in Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 2018, pp. 1361–1373, https://doi.org/10.1137/1.9781611975031.90.

[21] A. Filtser, R. Krauthgamer, and O. Trabelsi, *Relaxed voronoi: A simple framework for terminal-clustering problems*, in Proceedings of the 2nd Symposium on Simplicity in Algorithms, San Diego, CA, 2019, pp. 10:1–10:14, https://doi.org/10.4230/OASIcs.SOSA.2019.10.

[22] M. L. Fredman and R. E. Tarjan, *Fibonacci heaps and their uses in improved network optimization algorithms*, J. ACM, 34 (1987), pp. 596–615, https://doi.org/10.1145/28869.28874.

[23] G. Goranci, M. Henzinger, and P. Peng, *Improved guarantees for vertex sparsification in planar graphs*, in Proceedings of the 25th Annual European Symposium on Algorithms, Vienna, Austria, 2017, pp. 44:1–44:14, https://doi.org/10.4230/LIPIcs.ESA.2017.44.

[24] A. Gupta, *Steiner points in tree metrics don't (really) help*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, PA, 2001, pp. 220–227, http://dl.acm.org/citation.cfm?id=365411.365448.

[25] A. Gupta, V. Nagarajan, and R. Ravi, *An improved approximation algorithm for requirement cut*, Oper. Res. Lett., 38 (2010), pp. 322–325.

[26] L. Kamma, R. Krauthgamer, and H. L. Nguyen, *Cutting corners cheaply, or how to remove Steiner points*, in Proceedings of SODA, 2014, pp. 1029–1040.

[27] L. Kamma, R. Krauthgamer, and H. L. Nguyen, *Cutting corners cheaply, or how to remove Steiner points*, SIAM J. Comput., 44 (2015), pp. 975–995, https://doi.org/10.1137/140951382.

[28] T. Kavitha and N. M. Varma, *Small stretch pairwise spanners*, in Automata, Languages, and Programming Part I, Lecture Notes in Comput. Sci. 7965, Springer-Verlag, Berlin, 2013, pp. 601–612, https://doi.org/10.1007/978-3-642-39206-1_51.

[29] R. Krauthgamer, H. L. Nguyen, and T. Zondiner, *Preserving terminal distances using minors*, SIAM J. Discrete Math., 28 (2014), pp. 127–141, https://doi.org/10.1137/120888843.

[30] R. Krauthgamer and I. Rika, *Mimicking networks and succinct representations of terminal cuts*, in Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 2013, pp. 1789–1799, https://doi.org/10.1137/1.9781611973105.128.

[31] R. Krauthgamer and I. Rika, *Refined Vertex Sparsifiers of Planar Graphs*, CoRR abs/1702.05951, 2017.

[32] F. T. Leighton and A. Moitra, *Extensions and limits to vertex sparsification*, in Proceedings of the 42nd ACM Symposium on Theory of Computing, Cambridge, MA, 2010, pp. 47–56, https://doi.org/10.1145/1806689.1806698.

[33] N. Linial and M. E. Saks, *Decomposing graphs into regions of small diameter*, in Proceedings of the 2nd Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, San Francisco, CA, 1991, pp. 320–330, http://dl.acm.org/citation.cfm?id=127787.127848.

[34] K. Makarychev and Y. Makarychev, *Metric extension operators, vertex sparsifiers and Lipschitz extendability*, in Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science, Las Vegas, NV, 2010, pp. 255–264, https://doi.org/10.1109/FOCS.2010.31.

[35] G. L. Miller, R. Peng, A. Vladu, and S. C. Xu, *Improved parallel algorithms for spanners and hopsets*, in Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, Portland, OR, 2015, pp. 192–201, https://doi.org/10.1145/2755573.2755574.

[36] A. Moitra, *Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size*, in Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, Atlanta, GA, 2009, pp. 3–12, https://doi.org/10.1109/FOCS.2009.28.

[37] D. Peleg and A. A. Schäffer, *Graph spanners*, J. Graph Theory, 13 (1989), pp. 99–116, https://doi.org/10.1002/jgt.3190130114.

[38] L. Roditty, M. Thorup, and U. Zwick, *Deterministic constructions of approximate distance oracles and spanners*, in Automata, Languages and Programming, Lecture Notes in Comput. Sci. 3580, Springer-Verlag, Berlin, 2005, pp. 261–272, https://doi.org/10.1007/11523468_22.

[39] M. Thorup and U. Zwick, *Approximate distance oracles*, J. ACM, 52 (2005), pp. 1–24, https://doi.org/10.1145/1044731.1044732.