Relaxed Voronoi: A Simple Framework for Terminal-Clustering Problems*

Arnold Filtser[1]
Ben-Gurion University of the Negev, Be'er Sheva, Israel
arnoldf@cs.bgu.ac.il

Robert Krauthgamer[2]
Weizmann Institute of Science, Rehovot, Israel
robert.krauthgamer@weizmann.ac.il

Ohad Trabelsi[3]
Weizmann Institute of Science, Rehovot, Israel
ohad.trabelsi@weizmann.ac.il

Abstract

We reprove three known algorithmic bounds for terminal-clustering problems, using a single framework that leads to simpler proofs. In this genre of problems, the input is a metric space (X, d) (possibly arising from a graph) and a subset of terminals $K \subset X$, and the goal is to partition the points X such that each part, called a cluster, contains exactly one terminal (possibly with connectivity requirements) so as to minimize some objective. The three bounds we reprove are for Steiner Point Removal on trees [Gupta, SODA 2001], for Metric 0-Extension in bounded doubling dimension [Lee and Naor, unpublished 2003], and for Connected Metric 0-Extension [Englert et al., SICOMP 2014].

A natural approach is to cluster each point with its closest terminal, which would partition X into so-called Voronoi cells, but this approach can fail miserably due to its stringent cluster boundaries. A now-standard fix, which we call the `Relaxed-Voronoi` framework, is to use enlarged Voronoi cells, but to obtain disjoint clusters, the cells are computed greedily according to some order. This method, first proposed by Calinescu, Karloff and Rabani [SICOMP 2004], was employed successfully to provide state-of-the-art results for terminal-clustering problems on general metrics. However, for restricted families of metrics, e.g., trees and doubling metrics, only more complicated, ad-hoc algorithms are known. Our main contribution is to demonstrate that the `Relaxed-Voronoi` algorithm is applicable to restricted metrics, and actually leads to relatively simple algorithms and analyses.

2012 ACM Subject Classification Theory of computation → Random projections and metric embeddings, Theory of computation → Graph algorithms analysis

Keywords and phrases Clustering, Steiner point removal, Zero extension, Doubling dimension, Relaxed voronoi

Digital Object Identifier 10.4230/OASIcs.SOSA.2019.10

Related Version Full version at arXiv:1809.00942

* In earlier versions this algorithm was called "Noisy Voronoi".

[1] Work partially supported by the Lynn and William Frankel Center for Computer Sciences, ISF grant 1817/17, and by BSF Grant 2015813.
[2] Work partially supported by ONR Award N00014-18-1-2364, the Israel Science Foundation grant #1086/18, a Minerva Foundation grant, and a Google Faculty Research Award.
[3] Work partly done at IBM Almaden.

1 Introduction

We consider *terminal clustering* problems, where the input is a metric space (X, d) with k terminals $K \subseteq X$, and the goal is to partition the points (vertices) into k clusters, each containing exactly one terminal, so as to minimize some objective. In the *graphical version* of this problem, the input is a weighted graph $G = (V, E, w)$ with terminals $K \subset V$ and the metric d is derived as the shortest-path metric on $X = V$ with respect to the non-negative edge weights w, and every output cluster should be connected (as an induced subgraph of G).

We present for these problems a simple algorithmic framework that generalizes two different known algorithms, from [3, 10]. Using this framework, we obtain simple algorithms for two specific metric/graph classes, and recover their known bounds from [13, 19, 7] in a unified manner that is arguably simpler and more insightful than previous work. In our case, even the analysis is short and simple. Thus, our main contribution is to identify and present the framework, and to (non-trivially) apply it to specific metric/graph classes, and we hope it will lead to new results in the future. We proceed to define the two specific problems that we investigate, and briefly survey their known bounds.

Metric 0-Extension (M0E). In this problem, the input is a metric space (X, d) and a set of k terminals $K \subset X$, and the goal is to find a distribution $\mathcal{D}$ over retractions f (i.e., functions $f : X \to K$ that satisfy $f(x) = x$ for all $x \in K$), such that

$$\forall x, y \in X, \qquad \mathop{\mathbb{E}}_{f \sim \mathcal{D}} [d(f(x), f(y))] \leq \alpha \cdot d(x, y),$$

where $\alpha \geq 1$, called the *expected distortion*, is as small as possible. Throughout, we seek the smallest α that holds for a class of metric spaces, for example all metrics with k terminals, and then $\alpha = \alpha(k)$.

The above is closely related to the well-known 0-*Extension* problem, in which the input is a set X, a terminal set $K \subseteq X$, a metric d_K over the terminals and a cost function $c : \binom{X}{2} \to \mathbb{R}_+$, and the goal is to find a retraction $f : X \to K$ that minimizes $\sum_{\{x,y\} \in \binom{X}{2}} c(x, y) \cdot d_K(f(x), f(y))$. The 0-Extension problem, first proposed by Karzanov [17], generalizes the Multiway Cut problem [6] by allowing d_K to be any discrete metric (instead of a uniform metric) and it is also a special case of the Metric Labeling problem [18], whose objective function has additional terms that represent assignment costs. Karzanov introduced a linear programming (LP) relaxation for 0-Extension, which can be described as finding a (semi-)metric d_X over X that agrees with d_K on K, and minimizes $\sum_{\{x,y\} \in \binom{X}{2}} c(x, y) \cdot d_X(x, y)$. Rounding this LP relaxation is equivalent to the M0E problem (by the minimax theorem). Consequently, most previous work on 0-Extension has actually focused on solving M0E, and so does our work.

A well-known open problem is to determine the smallest distortion $\alpha(k)$ that suffices for all metric spaces with k terminals. The currently known bounds are $O(\log k / \log \log k)$ due to Fakcharoenphol, Harrelson, Rao, and Talwar [8] (improving over [3]), and $\Omega(\sqrt{\log k})$ due to Calinescu, Karloff and Rabani [3]. Improved upper bounds are known for special classes of metric spaces X, for example $O(1)$ for the case where X is the shortest-paths metric of a graph excluding a fixed minor [3]. Another example is when the submetric on the terminals (i.e., the restriction of d to K) is β-decomposable, which admits an $O(\beta)$ upper bound [19] (a somewhat similar bound was obtained in [1]). This implies an $O(\mathrm{ddim}(K))$ upper bound, where $\mathrm{ddim}(K)$ denotes the doubling dimension of the terminals' submetric (see Section 2 for definition), and our results reproduce the latter bound.

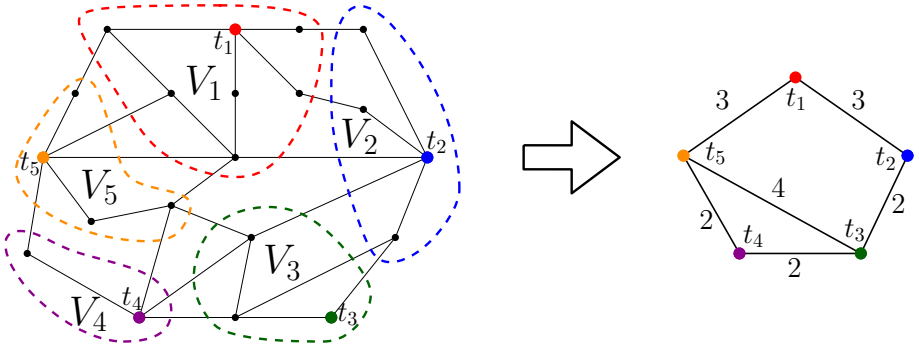

Figure 1 Example how a terminal partition of graph G (on left) induces a minor M (on right). The graph shown has unit weight edges and 5 terminals, and the terminal partition is shown using dashed curves. The distortion is $\frac{d_M(t_2,t_5)}{d_G(t_2,t_5)} = \frac{6}{2} = 3$.

Steiner Point Removal (SPR). In this problem, given a weighted graph $G = (V, E, w)$ and a set of terminals $K \subseteq V$, the goal is to find a minor $M = (K, E')$ of G (note its vertex set is exactly the set of terminals), that approximately preserves the distances between terminals, which means (using d_H to denote the shortest-path metric in H) that

$$\forall t, t' \in K, \qquad d_G(t, t') \le d_M(t, t') \le \alpha \cdot d_G(t, t'),$$

where $\alpha \ge 1$, called the *distortion*, is as small as possible. Again, we seek the best α that holds for a class of graphs, say all graphs with $k = |K|$ terminals.

Let us denote $K = \{t_1, \ldots, t_k\}$. A partition $\{V_1, \ldots, V_k\}$ of V is called a *terminal partition* (with respect to K) if for all $i = 1, \ldots, k$, the induced subgraph $G[V_i]$ is connected and contains t_i. The *induced minor* M of such a terminal partition is the minor obtained by contracting each V_i into a single vertex called (abusing notation) t_i. Thus, M has an edge between t_i and t_j iff G has an edge between V_i and V_j. The weight of this edge (if exists) is simply $d_G(t_i, t_j)$, which represents the shortest-path in G; see Figure 1 for an example. Most of the work on SPR so far used terminal partitions to obtain a minor, and so does our work.

For the case where the graph G is a tree, the smallest distortion possible for SPR is known to be 8. Gupta [13] constructed a tree achieving distortion 8; in fact, he was only interested in constructing a tree with vertex set K, and later Chan, Xia, Konjevod, and Richa [4] observed that Gupta's tree is actually a minor of the given tree G. Surprisingly, they further showed that 8 is the best possible distortion for the family of trees, as (unweighted) complete binary trees require distortion $8 - \epsilon$. Our results reproduce this upper bound of 8.

For SPR in general graphs there is currently a huge gap. The best lower bound known is just 8, known for trees, and recently Filtser [9] showed an $O(\log k)$ upper bound (improving over [16, 5]). No better upper bound is known even for seemingly much simpler cases such as planar graphs, and the only other bound known is $\alpha = O(1)$ for outerplanar graphs [2].

1.1 Algorithmic Framework

A natural and straightforward algorithm for terminal clustering is to simply partition the metric (or graph) into Voronoi cells, i.e., map each point (or vertex) to its closest terminal, to obtain a partition of X (or V) with one cluster for each terminal. However, there are easy examples where this algorithm fails miserably, because of the stringent cluster boundaries. A now-standard fix is to build around each terminal (iteratively) a cluster that is an enlarged Voronoi cell in the remaining metric (or graph).

Algorithm 1 `Metric-Relaxed-Voronoi`.

input : metric $M = (X, d)$, terminals K, ordering $\pi = (t_1, \ldots, t_k)$, magnitudes $R_1, \ldots, R_k \geq 1$
output : retraction $f : X \to K$ (i.e., $\forall x \in K,\ f(x) = x$)

```
1 for j = 1, ..., k do
2     for all unmapped points x such that d(t_j, x) ≤ R_j · D(x) do
3         set f(x) = t_j
4 return f
```

This approach was first used by Calinescu, Karloff and Rabani [3]. We generalize their method, so that all previous uses of this approach can be seen as instantiations of our algorithm with specific parameters. Our algorithm, called `Relaxed-Voronoi`, is formally described in Algorithm 1 where throughout we define

$$D(x) = d(x, K) = \min_{t \in K} d(x, t)$$

to be the distance from $x \in X$ to its closest terminal. The algorithm's parameters, formally presented as part of the input, are an ordering $\pi = (t_1, \ldots, t_k)$ of the terminals and corresponding magnitudes $R_1, \ldots, R_k \geq 1$ (one for each terminal). The algorithm is rather simple; each terminal t_j, in turn according to the ordering, creates a cluster $V_j = f^{-1}(t_j)$ containing all yet-unclustered points x at distance $d(x, t_j) \leq R_j \cdot D(v)$. That is, the cluster of t_j is a Voronoi cell "enlarged" by factor R_j in the remaining metric. Setting $R_1 = \cdots = R_k = 1$ recovers the partition into Voronoi cells.

The above algorithm cannot be used as is for the SPR problem, because a terminal partition has an additional connectivity requirement. Therefore, in the graphical case, instead of taking all remaining vertices x that satisfy $d_G(x, t_j) \leq R_j \cdot D(v)$, we create V_j in a Dijkstra-like iterative fashion, as follows. Initially $V_j = \{t_j\}$, and we repeatedly add to V_j any unclustered vertex that has a neighbor in V_j and is at distance $d_G(v, t_j) \leq R_j \cdot D(v)$. See Algorithms 2 and 3 for a formal description. This version of the `Relaxed-Voronoi` algorithm was first proposed by Filtser [10] for the SPR problem in general graphs. It is simpler to describe and to analyze than the `Ball-Growing` algorithm of previous work [16, 5, 9].[4] Filtser also showed that the `Relaxed-Voronoi` algorithm can be implemented in time $O(|E| \log |V|)$.[5]

1.2 Our Contribution

All previous uses of the `Relaxed-Voronoi` algorithm were on general metrics or graphs. Specifically, Calinescu et al. [3] and Fakcharoenphol et al. [8], used a uniformly random ordering π and a single random magnitude R (same for all terminals), and Filtser [10] used an arbitrary ordering π and magnitudes that are independently and identically distributed (i.i.d.) drawn from an exponential-like distribution. However, for special families of metrics or graphs,

[4] The `Ball-Growing` algorithm creates clusters in rounds, where each round iteratively enlarges every cluster, by increasing its radius around each terminal (in the remaining graph) by a value sampled from an exponential distribution.

[5] The $O(|E| \log |V|)$-time in [10] actually implements a slightly different algorithm, where the test $d_G(v, t_j) \leq R_j \cdot D(v)$ (line 4) is replaced by $d_{G[V_j \cup \{v\}]}(v, t_j) \leq R_j \cdot D(v)$. The distortion bound holds for this algorithm too.

Algorithm 2 `Graphic-Relaxed-Voronoi`.

input : weighted graph $G = (V, E, w)$, terminals K, ordering $\pi = (t_1, \ldots, t_k)$, magnitudes $R_1, \ldots, R_k \geq 1$
output : Minor M

$V_\perp \leftarrow V \setminus K$ // $V_\perp$ *is the currently unclustered vertices.*
for $j = 1, \ldots, k$ **do**
  $V_j \leftarrow$ `Create-Cluster`$(G, V_\perp, t_j, R_j)$
  $V_\perp \leftarrow V_\perp \setminus V_j$
return the terminal-centered minor M of G induced by $V_1, \ldots, V_k$

Algorithm 3 `Create-Cluster`.

input : weighted graph $G = (V, E, w)$, unclustered vertices $V_\perp$, terminal t_j, magnitude R_j
output : cluster V_j

$V_j \leftarrow \{t_j\}$, $U \leftarrow \emptyset$, $N \leftarrow \{$all neighbors of t_j in $V_\perp\}$
while $N \neq \emptyset$ **do**
  pick an arbitrary vertex $v \in N$ and remove it from N
  if $d_G(v, t_j) \leq R_j \cdot D(v)$ **then**
    add v to V_j
    add all the neighbors of v in $V_\perp \setminus (U \cup V_j)$ to N
  else
    add v to U
return V_j

this type of algorithm was never used; instead, ad-hoc algorithms were developed, leading to more involved algorithms and analyses. Our contribution is to tailor the `Relaxed-Voronoi` algorithm to special input families by choosing the ordering π deterministically but depending on the input at hand (rather than a random or arbitrary ordering). As a result, we reprove three known results using simpler algorithms and analyses. We believe that this approach will lead to additional and new results.

SPR on Trees. Gupta's algorithm [13], which achieves distortion 8, is designed specifically for trees and it is unclear how to generalize it. Its recursive definition makes it arguably difficult to understand intuitively how its output on a given tree would look like. For example, the fact that the algorithm is tight and produces a minor [4] was non-trivial and even surprising. This result has proved useful in the past, yet it is a bit mysterious why 8 is the optimal bound, i.e., what tradeoff does it optimize.

We use the `Relaxed-Voronoi` algorithm to construct a tree with optimal distortion 8. The choice of parameters in the algorithm is very simple – the magnitudes are all set to $R_j = 3$, and the ordering π is defined by listing the terminals in order of increasing distance from an arbitrary "root" vertex v (breaking ties arbitrarily). Our algorithm's description is simple and intuitive, its distortion bound 8 is explained by the analysis, and it is straightforward that the output tree is a minor of the input tree. Perhaps surprisingly, our algorithm outputs the same tree as Gupta's algorithm. Overall, our algorithm provides a better understanding of Gupta's celebrated result. We believe that this approach can be generalized to additional graph families, and hopefully achieve a constant distortion for SPR on (say) planar graphs (where the current bound is only $O(\log k)$, which holds for general graphs).

M0E on Doubling Metrics. Lee and Naor's [19] algorithm achieves $O(\text{ddim})$ when the submetric on the terminals (i.e., the metric's restriction to points in K) has doubling dimension at most ddim. Their algorithm is based on stochastic decompositions, specifically converting padded decompositions into separating decompositions, then defining (new) partial decompositions, and finally using these decompositions in all the possible distance scales.

We use the `Relaxed-Voronoi` algorithm to achieve the same $O(\text{ddim})$ upper bound, by setting the parameters as follows. The magnitudes R_j are i.i.d., each distributed like $2 \cdot e^Z$ where Z is drawn from an exponential distribution with parameter $\Theta(\text{ddim})$. We set π to be the Gonzalez order [12], where t_1 is an arbitrary terminal, and each successive t_i is the terminal farthest from $\{t_1, \ldots, t_{i-1}\}$, breaking ties arbitrarily. Our algorithm is much simpler, more elegant, and its straightforward implementation takes only $O(nk)$ time (assuming the input is given as a matrix of pairwise distances). We hope that our ideas could lead to a better upper bound for the SPR problem in the case where the metric restricted to the terminals has a bounded doubling dimension.

Connected M0E. This is a graphic version of the M0E problem. The input metric is the shortest-path metric of an edge-weighted graph $G = (V, E, w)$, and similarly to the M0E problem, the goal is to find a distribution over retractions $f : V \to K$, but with an additional requirement: each cluster $f^{-1}(t_j)$ must be connected (as a subgraph of G). Englert et al. [7] achieved for this problem expected distortion $\alpha = O(\log k)$ using an algorithm that partitions the graph vertices into clusters using stochastic decompositions in all possible distance scales, and then merging some clusters to enforce connectivity. We use a graphic version of the `Relaxed-Voronoi` algorithm (which guarantees connectivity) to achieve the same expected distortion $O(\log k)$. When describing this algorithm, we abuse notation and identify $f(v) = t_j$ with $v \in V_j$, i.e., when the algorithm adds a vertex v to cluster V_j, it should be understood as also assigning $f(v) = t_j$. The graphic `Relaxed-Voronoi` algorithm is much simpler than the previous algorithm of [7], and we set its parameters as follows. The ordering π is arbitrary, and the magnitudes R_j are i.i.d., each distributed like e^Z where Z is drawn from an exponential distribution with parameter $\Theta(\log k)$. Even though this problem is concerned with general graphs and there is nothing clever about the ordering, we still chose to present this result, as it gives further evidence to the strength and broad applicability of the `Relaxed-Voronoi` algorithm. Another advantage is that it can be implemented in $O(|E| \log |V|)$ time, while the algorithm of [7] requires more time (an unspecified polynomial). See Footnote 5 for additional details.

1.3 Related Work

The Voronoi-like approach was used also in other recent algorithms. Gupta and Talwar [15] introduced the `Random-Rates` algorithm, in which each terminal t_j samples a rate $\rho_j \geq 1$, and then every point x is clustered with the terminal t_j that minimizes the ratio $\frac{d(x,t_j)}{\rho_j}$. The main difference from the `Relaxed-Voronoi` algorithm is that in their algorithm, the terminals create their clusters simultaneously (rather than sequentially), which does not guarantee that the clusters are connected. Gupta and Talwar [15] proved an $O(\log k)$ expected distortion for this algorithm on the M0E problem. It seems unlikely that their algorithm can provide $O(\text{ddim}(K))$ upper bound, which usually follows by bounding the number of clusters relevant to any "separation event" by $2^{O(\text{ddim}(K))}$. We achieve this using the sequential ordering, but in their algorithm too many clusters can be relevant.

Miller, Peng and Xu [20] introduced the `Parallel-Partition` algorithm to partition a graph into low-diameter clusters (without a given set of terminals). In this algorithm,

each vertex u samples a random shift $s_u \geq 0$, and then every vertex x joins the cluster of u with minimum $d(x,u) - s_u$. This algorithm produces connected clusters, however, it gets as an input a target diameter $\Delta > 0$, and its guarantees are proportional to this parameter. In contrast, the `Relaxed-Voronoi` algorithm is scale-free and handles all distances scales simultaneously (similar to the above `Random-Rates` algorithm), and therefore it is more natural for terminal-partitioning problems.

2 Preliminaries

Consider an undirected graph $G = (V, E)$ with non-negative edge weights $w : E \to \mathbb{R}_{\geq 0}$ and 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 the given *terminals*. As mentioned earlier, for a vertex $v \in V$ we define $D(v) = \min_{t \in K} d_G(v,t)$ to be the distance from v to its closest terminal.

A graph H is a *minor* of a graph G if it can be obtained from G by edge deletions, edge contractions, and vertex deletions. As defined earlier, a partition $\{V_1, \ldots, V_k\}$ of V is called a *terminal partition* (with respect to K) if for all $i = 1, \ldots, k$, the induced subgraph $G[V_i]$ is connected and contains t_i. The *minor induced* by a terminal partition $\{V_1, \ldots, V_k\}$ is the minor M obtained by contracting each set V_i into a single vertex called (abusing notation) t_i. Notice that M has an edge between t_i and t_j iff there are vertices $v_i \in V_i$ and $v_j \in V_j$ such that $\{v_i, v_j\} \in E$. The weight of this edge (if exists) is simply $d_G(t_i, t_j)$, which represents the shortest-path in G. It is easily verified that by the triangle inequality, for every pair of (not necessarily adjacent) terminals t_i, t_j, we have $d_M(t_i, t_j) \geq d_G(t_i, t_j)$. The *distortion* of the induced minor is $\max_{i \neq j} \frac{d_M(t_i,t_j)}{d_G(t_i,t_j)}$. It was proved in [10] that the `Relaxed-Voronoi` algorithm always returns a terminal partition.

▶ **Lemma 2.1** (Lemma 2 in [10]). *The sets $V_1, \ldots, V_k$ constructed by Algorithm 2 constitute a terminal partition.*

We say that a metric (X, d) has *doubling dimension* ddim if every ball of radius $r > 0$ can be covered by at most 2^{ddim} balls of radius $r/2$. We will use the following *packing property* of doubling spaces [14]: Consider a set N such that for every $x \neq y \in N$ it holds that $d(x,y) \geq \delta$. Then every ball of radius $\Delta \geq \delta$ contains at most $\left(\frac{4\Delta}{\delta}\right)^{O(\text{ddim})} = 2^{O\left(\text{ddim} \cdot \log \frac{\Delta}{\delta}\right)}$ points from N.

We denote by $\text{EXP}(\lambda)$ the *exponential distribution* with mean $\lambda > 0$, which has density function $f(x) = \frac{1}{\lambda} e^{-\frac{x}{\lambda}}$ for $x \geq 0$. This distribution is *memoryless*: if $X \sim \text{EXP}(\lambda)$, then for all $a, b \geq 0$ we have $\Pr[X \geq a + b \mid X \geq a] = \Pr[X \geq b]$. In other words, conditioned on $X \geq a$, it holds that $X \sim a + \text{EXP}(\lambda)$.

3 SPR on trees

In this section we analyze the `Relaxed-Voronoi` algorithm (Algorithm 2) on trees.

▶ **Theorem 3.1.** *Let T be a tree and r be an arbitrary vertex. Let π be an ordering of the terminals according to an increasing distance from r. Then the tree T_K returned by the `Relaxed-Voronoi` algorithm on input $(T, K, \pi, \{3, 3, \ldots, 3\})$ has distortion at most 8. Moreover, the algorithm can be implemented in linear time.*

In Section 3.1 we bound the distortion produced by our algorithm, and in Section 6 we describe its linear time implementation. See Figure 2 for an example execution of the algorithm on a complete unweighted binary tree (the lower bound example used by [4]).

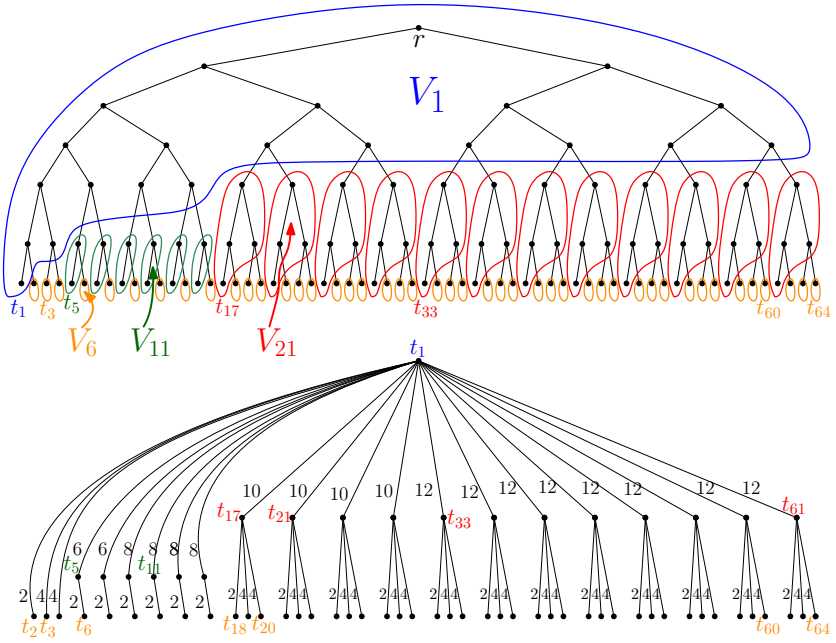

Figure 2 An example execution of the `Relaxed-Voronoi` algorithm. The top graph is the input, a complete binary tree of height 6 with all the leaves as terminals. Choosing r to be the root of the tree, the terminal ordering π can be arbitrary, and we choose a left-to-right ordering. The resulting clusters are shown using colored curves. The bottom graph shows the minor induced by the terminal partition above, representing every cluster by its top-most vertex (rather than the terminal). The distortion is $\frac{d_{T_k}(t_{64},t_{60})}{d_T(t_{64},t_{60})} = \frac{32}{6} = 5\frac{2}{3}$.

3.1 Distortion Analysis

To better understand the final distortion bound 8, we analyze the `Relaxed-Voronoi` algorithm for a general $R > 1$, and we optimize it only at the very end, setting $R = 3$ to obtain distortion 8.

Denote by T_K the tree minor returned by the algorithm, and call the vertex r (used to determine π) the root. Let t_1 be the first terminal w.r.t π, and let V_1 be the cluster that the `Relaxed-Voronoi` algorithm constructs for t_1. This terminal t_1 is the closest terminal to the root r, and actually also to every vertex on the shortest path from t_1 to r. Therefore r joins the cluster V_1. Let $C_1, \ldots, C_s$ be the connected components of the remaining graph $G \setminus V_1 = G[V \setminus V_1]$, and let $K_i = C_i \cap K$ be the subset of terminals in component C_i. We claim that for every vertex $v \in C_i$ (for every i), its closest terminal t_v satisfies $t_v \in K_i$. Indeed, assume towards contradiction that some vertex u on the path between v to $t_v \in K_i$ joined V_1. Consider then an arbitrary vertex u' on the path from v to u, and note that t_v is also the closest terminal to both u, u'. By the triangle inequality, $d_T(t_1, u') \leq d_T(t_1, u) + d_T(u, u') \leq R \cdot \big(d_T(t_v, u) + d_T(u, u')\big) = R \cdot d_T(t_v, u')$. This implies that every vertex on the path from u to v will join V_1 (recall we assumed u joins V_1, and the algorithm iteratively adds neighbor of vertices already in V_1), in contradiction with $v \in C_i$. See Figure 3 for illustration.

By construction, there is only a single edge e_i that connects C_i and V_1, and denote its two endpoints by $r_i \in C_i$ and $s_i \in V_1$. Let $\tilde{t}_i \in K$ be the closest terminal to r_i, thus $\tilde{t}_i \in C_i$. Observe that r_i is the closest vertex to r among all vertices in C_i, and in particular every path from a terminal $t \in C_i$ to r goes through r_i. Let π_i be an ordering of the terminals in K_i according to increasing distance from r_i. Note that π_i is just the order π restricted to K_i.

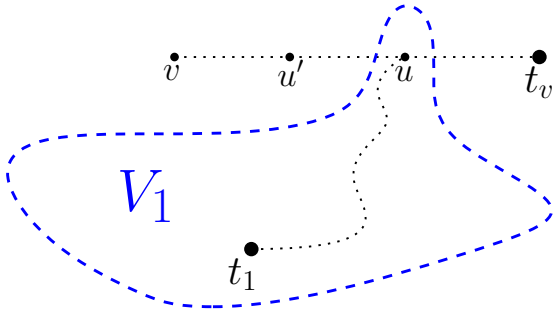

Figure 3 Illustrating the argument that for every $v \in C_i$, also its closest terminal $t_v \in C_i$. Assuming some u on the path between them joined V_1, we conclude the entire path from u to v joins V_1.

Since all the clusters created by the `Relaxed-Voronoi` algorithm are connected, no vertex in C_i can join a cluster associated with a terminal outside K_i. In particular, for every $v \in C_i$ the distance $D(v)$ to the closest terminal in the restricted tree $G[C_i]$ remains the same (as $t_v \in C_i$). Therefore, if we execute the `Relaxed-Voronoi` algorithm on C_i with terminal set K_i and order π_i, the partition of C_i to clusters will be identical to the partition of C_i induced by the original algorithm (on T with the order π). Accordingly, if we combine all the clusters created by such executions with V_1, we get the same terminal partition as produced by the `Relaxed-Voronoi` algorithm on the original graph.

Next, we argue by induction on the number of terminals that for every terminal t, $d_{T_K}(t_1, t) \leq \frac{R+1}{R-1} \cdot d_T(t_1, t)$. In a tree with a single or two terminals this claim is trivial. We now prove the induction step. Let t_i be some terminal which belongs to the connected component C_i (in $T \setminus V_1$). By applying the induction hypothesis to the tree C_i with order π_i, it holds that $d_{T_k}(t_i, \tilde{t}_i) \leq \frac{R+1}{R-1} \cdot d_T(t_i, \tilde{t}_i)$ as $\tilde{t}_i$ is the first terminal in the order π_i. Note that r_i will necessarily join the cluster of $\tilde{t}_i$, therefore the edge $(e_i = \{s_i, r_i\})$ crosses the clusters of t_1 and $\tilde{t}_i$, which implies that there is an edge between t_1 to $\tilde{t}_i$ in T_K. See Figure 4 for illustration.

As r_i has a neighbor in V_1 but did not join V_1, necessarily $d_T(t_1, r_i) > R \cdot D(r_i) = R \cdot d_T(r_i, \tilde{t}_i)$. We conclude,

$$\begin{aligned}
d_{T_K}(t_1, t_i) &\leq d_{T_K}(t_1, \tilde{t}_i) + d_{T_K}(\tilde{t}_i, t_i) \\
&\leq d_T(t_1, \tilde{t}_i) + \frac{R+1}{R-1} \cdot d_T(\tilde{t}_i, t_i) \\
&\leq d_T(t_1, r_i) + D(r_i) + \frac{R+1}{R-1} \cdot \big(D(r_i) + d_T(r_i, t_i)\big) \\
&< d_T(t_1, r_i) + \big(1 + \frac{R+1}{R-1}\big) \cdot \frac{d_T(t_1, r_i)}{R} + \frac{R+1}{R-1} \cdot d_T(r_i, t_i) \\
&= \frac{R+1}{R-1} \cdot \big(d_T(t_1, r_i) + d_T(r_i, t_i)\big) = \frac{R+1}{R-1} \cdot d_T(t_1, t_i) \ .
\end{aligned}$$

Finally, we show by induction that for every pair of terminals $t_i, t_j \in K \setminus \{t_1\}$, $d_{T_k}(t_i, t_j) < \frac{(R+1)^2}{R-1} \cdot d_T(t_i, t_j)$. If t_i, t_j belong to the same connected component of $T \setminus V_1$ then the argument follows by the induction hypothesis. Otherwise, $t_i \in C_i$ and $t_j \in C_j$ for $i \neq j$. Recall that there is a single edge $e_i = \{s_i, r_i\}$ from C_i to V_1. Clearly, the unique path in T from t_i to t_j goes through V_1 and in particular through s_i and s_j (note that it is possible that $s_i = s_j$). Therefore, $d_T(t_i, t_j) \geq d_T(t_i, s_i) + d_T(s_j, t_j)$. As $s_i \in V_1$, it holds that $d_T(t_1, s_i) \leq R \cdot D(s_i) \leq R \cdot d_T(t_i, s_i)$. Therefore,

$$d_T(t_1, t_i) \leq d_T(t_1, s_i) + d_T(s_i, t_i) \leq (R+1) \cdot d_T(s_i, t_i) \,. \tag{1}$$

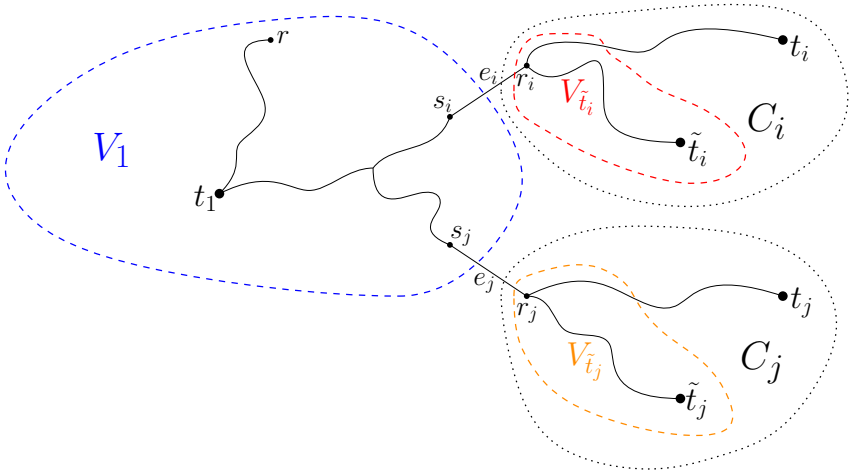

Figure 4 Illustrating the bound on $d_{T_k}(t_i, t_j)$. Initially $d_{T_k}(t_1, t_i)$ is bounded. Notice that $\tilde{t}_i$ is the closest terminal to r_i. Using the induction hypothesis we have that $d_{T_k}(\tilde{t}_i, t_i) \leq \frac{R+1}{R-1} \cdot d_T(\tilde{t}_i, t_i)$. As $\{t_1, \tilde{t}_i\}$ is an edge in T_k, the bound follows.
Next, the bound on $d_{T_k}(t_i, t_j)$. Notice that $d_T(t_i, t_j) \geq d_T(t_i, s_i) + d_T(t_j, s_j)$. $d_{T_k}(t_i, t_j)$ is upper bounded by going through t_1, using the assertion above.

Similarly $d_T(t_1, t_j) \leq (R+1) \cdot d_T(s_j, t_j)$. Using our claim above about t_1, we conclude (see Figure 4 for illustration)

$$\begin{aligned} d_{T_k}(t_i, t_j) &\leq d_{T_k}(t_i, t_1) + d_{T_k}(t_1, t_j) \\ &\leq \frac{R+1}{R-1} \cdot (d_T(t_i, t_1) + d_T(t_1, t_j)) \\ &\overset{(1)}{\leq} \frac{(R+1)^2}{R-1} \cdot (d_T(t_i, s_i) + d_T(s_j, t_j)) \\ &\leq \frac{(R+1)^2}{R-1} \cdot d_T(t_i, t_j). \end{aligned}$$

The expression $\frac{(R+1)^2}{R-1}$ is minimized by choosing $R = 3$, which proves the upper bound 8.

4 M0E for Doubling Metrics

In this section we analyze the `Relaxed-Voronoi` algorithm (Algorithm 1) for the M0E problem, in the case where the metric spaces restricted on the terminals has doubling dimension ddim. Given a metric space (X, d), Gonzalez's order [12] is defined as follows. x_1 is an arbitrary point, x_2 is the farthest point from x_1, and in general x_i is the farthest point from $\{x_1, \ldots, x_{i-1}\}$. In other words, x_i is the point maximizing $d(x_i, \{x_1, \ldots, x_{i-1}\})$.

▶ **Theorem 4.1.** *Let (X, d) be a metric space with a set of terminals $K \subseteq X$ such that the metric space restricted to the terminals has doubling dimension* ddim*. Let π be Gonzalez's order. Let $R_j = 2 \cdot e^{Z_j}$, where $Z_1, \ldots, Z_k$ are i.i.d. variables sampled according to the distribution* $\mathrm{EXP}(c \cdot \mathrm{ddim})$ *for large enough constant c. Then the expected distortion returned by the `Relaxed-Voronoi` algorithm for the M0E problem is $O(\mathrm{ddim})$.*

Proof. Consider a point $x \in X$, and let i_x be the minimal index such that $d(t_x, t_{i_x}) \leq D(x)$. Set $K_x = \{t_1, \ldots, t_{i_x}\}$. As $R_{i_x} \geq 2$, if x is unassigned until the i_x round, then $f(x) = t_{i_x}$.

Therefore, $f(x) \in K_x$. For every $t, t' \in K_x \setminus \{t_x\}$, $d(t,t') \geq D(x)$. Using the packing property, for $i \geq 1$, $\left|B\left(v, 2^i \cdot D(v)\right) \cap K_x\right| \leq \left|B\left(t_x, \left(2^i+1\right) \cdot D(v)\right) \cap K_x\right| = 2^{O(i \cdot \mathrm{ddim})}$.

▶ **Lemma 4.2.** *For every $x \in X$, $\mathbb{E}\left[d(x, f(x))\right] = O\left(1\right) \cdot D(x)$.*

Proof. For $i \geq 3$, let $K_i \subseteq K_x$ be the set of terminals at distance $[2^{i-1}, 2^i) \cdot D(v)$ from x. In order for the terminal $t_j \in K_i$ to cover x, it must be that $R_j \geq 2^{i-1}$, where a terminal t covers a point z if $f(z) = t$. This happens with probability at most

$$\Pr\left[R_j \geq 2^{i-1}\right] = \Pr\left[Z_j \geq (i-2) \cdot \ln 2\right] = e^{-c \cdot \mathrm{ddim} \cdot (i-2) \cdot \ln 2} \leq e^{-\frac{c}{5} \cdot \mathrm{ddim} \cdot i} .$$

By the union bound, the probability that some terminal from K_i covers x is bounded by $|K_i| \cdot e^{-\frac{c}{5} \cdot \mathrm{ddim} \cdot i}$. We conclude that for large enough constant c,

$$\begin{aligned}\mathbb{E}\left[d(x, f(x))\right] &\leq 2^2 \cdot D(x) + \sum_{i=3}^{\infty} \Pr\left[f(x) \in K_i\right] \cdot 2^i \cdot D(x) \\ &= 4 \cdot D(x) + D(x) \cdot \sum_{i=3}^{\infty} 2^{O(i \cdot \mathrm{ddim})} \cdot e^{-\frac{c}{5} \cdot \mathrm{ddim} \cdot i} \cdot 2^i = O(D(x)) .\end{aligned}$$

◀

Consider a pair of points $x, y \in X$ such that $d(x,y) = \epsilon \cdot \min\{D(x), D(y)\}$. If $\epsilon = \Omega(1)$, assume w.l.o.g that $D(x) \leq D(y)$, then $D(y) \leq D(x) + d(x,y) = O(1) \cdot d(x,y)$. Using Theorem 4.2 we conclude

$$\begin{aligned}\mathbb{E}\left[d(f(x), f(y))\right] &\leq \mathbb{E}\left[d(f(x), x)\right] + d(x,y) + \mathbb{E}\left[d(y, f(y))\right] \\ &= O\left(D(x) + D(y)\right) + d(x,y) = O\left(1\right) \cdot d(x,y) .\end{aligned} \tag{2}$$

Thus from now on we can assume that ϵ is upper bounded by small enough constant, and we also drop the assumption that $D(x) \leq D(y)$. We say that a terminal t_j *settles* the pair $\{x,y\}$ if it is the first terminal to cover at least one point among $\{x,y\}$, and denote this event by $\mathcal{S}_j$. We say that t_j *cuts* $\{x,y\}$ if t_j settles $\{x,y\}$ but covers only one of x, y, and denote this event by $\mathcal{C}_j$. Set $R_x = \frac{d(x,t_j)}{D(x)}$, $R_y = \frac{d(y,t_j)}{D(y)}$. Assuming w.l.o.g that $R_x \leq R_y$, we get

$$R_y = \frac{d(t_j, y)}{D(y)} \leq \frac{d(t_j, x) + d(v,u)}{D(x) - d(v,u)} \leq \frac{R_x \cdot D(x) + \epsilon \cdot D(x)}{D(x) - \epsilon \cdot D(x)} \leq \frac{1+\epsilon}{1-\epsilon} \cdot R_x < (1+3\epsilon) \cdot R_x . \tag{3}$$

Assuming that t_j settles $\{x,y\}$, using the memoryless property we can bound the probability that t_j cuts $\{x,y\}$.

$$\begin{aligned}\Pr\left[\mathcal{C}_j \mid \mathcal{S}_j\right] &= \Pr\left[R_j < R_y \mid R_j \geq R_x\right] \overset{(3)}{<} \Pr\left[2 \cdot e^{Z_j} < R_x \cdot (1+3\epsilon) \mid 2 \cdot e^{Z_j} < R_x\right] \\ &= \Pr\left[Z_j < \ln(1+3\epsilon)\right] < \Pr\left[Z_j < 3\epsilon\right] = 1 - e^{-3\epsilon \cdot c \cdot \mathrm{ddim}} \leq 6\epsilon \cdot c \cdot \mathrm{ddim} .\end{aligned} \tag{4}$$

Suppose that t_j indeed cuts $\{x,y\}$. Following the same arguments as Theorem 4.2, the expected distance between y to $f(y)$ still will be $O(D(y)) = O(\frac{1}{\epsilon}) \cdot d(x,y)$. Thus,

$$\begin{aligned}\mathbb{E}\left[d(f(x), f(y)) \mid \mathcal{C}_j\right] &\leq d(t_j, x) + d(x,y) + \mathbb{E}\left[d(y, f(y)) \mid \mathcal{C}_j\right] \\ &= d(t_j, \{x,y\}) + O\left(\tfrac{1}{\epsilon}\right) \cdot d(x,y) .\end{aligned} \tag{5}$$

For $i \geq 1$, denote by $\tilde{K}_i \subseteq K_x \cup K_y$ the set of terminals at distance $[2^{i-1}, 2^i) \cdot \min\{D(x), D(y)\}$ from $\{x,y\}$. By packing arguments, $|\tilde{K}_i| = 2^{O(i \cdot \mathrm{ddim})}$. By similar arguments to Theorem 4.2,

for $i \geq 3$, the probability that $\{x, y\}$ is settled by a terminal from $\tilde{K}_i$ is bounded by $2^{-\Omega(i \cdot \mathrm{ddim})}$. We conclude,

$$\begin{aligned}
&\mathbb{E}\left[d(f(x), f(y))\right] = \\
&= \sum_j \Pr\left[\mathcal{S}_j\right] \cdot \Pr\left[\mathcal{C}_j \mid \mathcal{S}_j\right] \cdot \mathbb{E}\left[d(f(x), f(y)) \mid \mathcal{C}_j\right] \\
&\overset{(4,5)}{\leq} 6\epsilon \cdot c \cdot \mathrm{ddim} \cdot \sum_j \Pr\left[\mathcal{S}_j\right] \cdot \left(d(t_j, \{x, y\}) + O\left(\tfrac{1}{\epsilon}\right) \cdot d(x, y)\right) \\
&= O\left(\mathrm{ddim}\right) \cdot d(x, y) + O\left(\epsilon \cdot \mathrm{ddim}\right) \cdot \left(4 + \sum_{i \geq 3} 2^{-\Omega(i \cdot \mathrm{ddim})} \cdot 2^i\right) \cdot \min\{D(x), D(y)\} \\
&= O\left(\mathrm{ddim}\right) \cdot d(x, y).
\end{aligned}$$

◀

5 Connected M0E

In this section we apply the (Graphic) `Relaxed-Voronoi` algorithm (Algorithm 2) to the connected-M0E problem.

▶ **Theorem 5.1.** *Let $G = (V, E, w)$ be a weighted graph and $K \subseteq X$ a set of terminals of size k. Let π be arbitrary, and let $R_j = e^{Z_j}$, where $Z_1, \ldots, Z_k$ are i.i.d. variables sampled according to distribution $\mathrm{EXP}(c \cdot \ln k)$ for large enough constant c. Then the expected distortion returned by the `Relaxed-Voronoi` algorithm for the connected M0E problem is $O(\log k)$.*

By the triangle inequality, it is enough to prove that for every edge $\{u, v\} \in E$ (where $d_G(v, u) = w(v, u)$) it holds that $\mathbb{E}_{f \sim \mathcal{D}}\left[d(f(u), f(v)\right] \leq \alpha \cdot d_G(v, u)$. The proof itself follows almost the same lines as the proof of Theorem 4.1. With high probability, $R_j \leq 2$ for every terminal t_j. Therefore, for every vertex v, $d(v, f(v)) \leq 2 \cdot D(v)$. Once a vertex v joins the cluster V_j, the probability that its unclustered neighbor vertex u, at distance $\epsilon \cdot D(v)$, does not join V_j is bounded by $O(\epsilon \cdot \log k)$ (similarly to Equation (4)). Using these two facts we can bound the expected distortion by $O(\log k)$. We skip the exact details.

6 Linear-Time Implementation

Our algorithm often uses $D(v)$. The next lemma state that this values can be computed efficiently.

▶ **Lemma 6.1.** *There is a linear-time algorithm, that given as an input a weighted graph $G = (V, E, w)$ and $K \subseteq V$ a set of terminals, outputs for every vertex $v \in V$ its distance from K.*

Proof. We describe the algorithm. We root the tree in some arbitrary vertex $r \in V$. Thus each vertex (other then r) has a parent vertex. Our algorithm has two phases. In the first phase we sweep the tree upwards from the leafs to the root. For a vertex v, denote by $d(v)$ the distance from v to it's closest terminal among its descendants (∞ if it has no descendant terminal). The goal of the first phase is for each vertex to learn $d(v)$, and this is done in a dynamic programing fashion according to the order induced by the tree. At the beginning each leaf v know $d(v)$ (0 if terminal and ∞ otherwise). Then, iteratively each internal vertex v with children $\{v_1, \ldots, v_s\}$ computes $d(v) = \min_i \{d(v_i) + d(v_i, v)\}$ or $d(v) = 0$ if v itself is a terminal. It is straightforward by induction that by the end of the first phase each vertex

has the right value of $d(v)$. Moreover, for the root vertex r, $D(r) = d(r)$ (as all the terminals are the descendants of r).

In the second phase we sweep the tree downwards from the root to the leaves. In the first step, r informs all its children the value $D(r)$. Then, iteratively, each vertex v with parent v' computes $D(v) = \min\{d(v), D(v') + d(v', v)\}$. Again, by induction this is indeed the right value (as every path ending in v which starts at a non-descendant of v must go through v'). By the end of the second phase each vertex knows the correct value of $D(v)$. The linear time implementation follows as we traversed each edge exactly twice. ◀

The execution of the `Relaxed-Voronoi` algorithm starts by computing the $D(v)$ values in linear time according to Theorem 6.1. Next, in order to determine the permutation π, we choose an arbitrary vertex r and run Dijkstra from it. In a tree, one can run the classic Dijkstra algorithm (as in [11]) using a queue instead of a heap. As there is a unique path from r to any other vertex, the algorithm still works properly. Next, we cluster the vertices according to the permutation π. The set N from the `Create-Cluster` procedure can be implemented as a simple queue. As there is a unique path between every pair of vertices, once a vertex v joins N, we can update $d(v, t_j)$ to its correct value. Moreover, there is no reason to maintain U. As in all the executions of the `Create-Cluster` procedure for all terminals, each edge is traversed exactly once, the total linear time follows.

References

1. Aaron Archer, Jittat Fakcharoenphol, Chris Harrelson, Robert Krauthgamer, Kunal Talwar, and Éva Tardos. Approximate classification via earthmover metrics. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 1079–1087, 2004. URL: `http://dl.acm.org/citation.cfm?id=982792.982952`.
2. A. Basu and A. Gupta. Steiner Point Removal in Graph Metrics. Unpublished Manuscript, available from `http://www.math.ucdavis.edu/~abasu/papers/SPR.pdf`, 2008.
3. Gruia Călinescu, Howard J. Karloff, and Yuval Rabani. Approximation Algorithms for the 0-Extension Problem. *SIAM J. Comput.*, 34(2):358–372, 2004. `doi:10.1137/S0097539701395978`.
4. T.-H. Chan, Donglin Xia, Goran Konjevod, and Andrea 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*, APPROX'06/RANDOM'06, pages 70–81, 2006. `doi:10.1007/11830924_9`.
5. Yun Kuen Cheung. Steiner Point Removal - Distant Terminals Don't (Really) Bother. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1353–1360, 2018. `doi:10.1137/1.9781611975031.89`.
6. Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The Complexity of Multiway Cuts (Extended Abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, STOC 1992*, pages 241–251, 1992. `doi:10.1145/129712.129736`.
7. Matthias Englert, Anupam Gupta, Robert Krauthgamer, Harald Räcke, Inbal Talgam-Cohen, and Kunal Talwar. Vertex Sparsifiers: New Results from Old Techniques. *SIAM J. Comput.*, 43(4):1239–1262, 2014. `doi:10.1137/130908440`.
8. Jittat Fakcharoenphol, Chris Harrelson, Satish Rao, and Kunal Talwar. An improved approximation algorithm for the 0-extension problem. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 257–265, 2003. URL: `http://dl.acm.org/citation.cfm?id=644108.644153`.

9 Arnold Filtser. Steiner Point Removal with Distortion $O(\log k)$. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 1361–1373, 2018. doi:10.1137/1.9781611975031.90.

10 Arnold Filtser. Steiner Point Removal with distortion $O(\log k)$, using the Noisy-Voronoi algorithm. *CoRR*, abs/1808.02800, 2018. arXiv:1808.02800.

11 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.

12 Teofilo F. Gonzalez. Clustering to Minimize the Maximum Intercluster Distance. *Theor. Comput. Sci.*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

13 Anupam Gupta. Steiner Points in Tree Metrics Don'T (Really) Help. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 220–227, 2001. URL: http://dl.acm.org/citation.cfm?id=365411.365448.

14 Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded Geometries, Fractals, and Low-Distortion Embeddings. In *44th Symposium on Foundations of Computer Science (FOCS 2003)*, pages 534–543, 2003. doi:10.1109/SFCS.2003.1238226.

15 Anupam Gupta and Kunal Talwar. Random Rates for 0-Extension and Low-Diameter Decompositions. *CoRR*, abs/1307.5582, 2013. arXiv:1307.5582.

16 Lior Kamma, Robert Krauthgamer, and Huy L. Nguyen. Cutting Corners Cheaply, or How to Remove Steiner Points. *SIAM J. Comput.*, 44(4):975–995, 2015. doi:10.1137/140951382.

17 Alexander V. Karzanov. Minimum 0-Extensions of Graph Metrics. *Eur. J. Comb.*, 19(1):71–101, 1998. doi:10.1006/eujc.1997.0154.

18 Jon M. Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields. *J. ACM*, 49(5):616–639, 2002. doi:10.1145/585265.585268.

19 James R. Lee and Assaf Naor. Metric decomposition, smooth measures, and clustering. Unpublished Manuscript, available from https://www.math.nyu.edu/~naor/homepage%20files/cluster.pdf, 2003.

20 Gary L. Miller, Richard Peng, and Shen Chen Xu. Parallel graph decompositions using random shifts. In *25th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '13, pages 196–203, 2013. doi:10.1145/2486159.2486180.