Relaxed Voronoi: A Simple Framework for Terminal-Clustering Problems

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

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

Ohad Trabelsi
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Steiner Point Removal on trees [Gupta, SODA 2001], for Metric 0-Extension in bounded doubling dimension [Lee and Naor, unpublished 2003], and for Connected Metric 0-Extension [Englert et al., SICOMP 2014].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
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 \(K \subseteq V\) 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 \(D\) over retractions \(f\) (i.e., functions \(f : X \to K\) that satisfy \(f(x) = x\) for all \(x \in K\)), such that

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

where \(\alpha \geq 1\), called the expected distortion, is as small as possible. Throughout, we seek the smallest \(\alpha\) 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 \(K \subset X\), 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 \(\beta\)-decomposable, which admits an \(O(\sqrt{\beta})\) upper bound [19] (a somewhat similar bound was obtained in [1]). This implies an \(O(ddim(K))\) upper bound, where \(ddim(K)\) denotes the doubling dimension of the terminals’ submetric (see Section 2 for definition), and our results reproduce the latter bound.
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

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

where $\alpha \geq 1$, called the distortion, is as small as possible. Again, we seek the best $\alpha$ 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.

\textbf{input}: metric \( M = (X, d) \), terminals \( K \), ordering \( \pi = (t_1, \ldots, t_k) \), magnitudes \( R_1, \ldots, R_k \geq 1 \)

\textbf{output}: retraction \( f : X \rightarrow K \) (i.e., \( \forall x \in K \), \( f(x) = x \))

\begin{algorithmic}
  \FOR {\( j = 1, \ldots, k \)}
    \FOR {all unmapped points \( x \) such that \( d(t_j, x) \leq R_j \cdot D(x) \)}
      \STATE set \( f(x) = t_j \)
    \ENDFOR
  \ENDFOR
  \RETURN \( f \)
\end{algorithmic}

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 \( d(x, t_j) \leq R_j \cdot D(x) \). 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]. Filtser also showed that the Relaxed-Voronoi algorithm can be implemented in time \( O(|E| \log |V|) \).

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 \( \pi \) and a single random magnitude \( R \) (same for all terminals), and Filtser [10] used an arbitrary ordering \( \pi \) 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 \( d_{G[V \cup \{v\}]}(v, t_j) \leq R_j \cdot D(v) \). 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 $M$

1. $V_{\perp} \leftarrow V \setminus K$ \hspace{1cm} // $V_{\perp}$ is the currently unclustered vertices.
2. for $j = 1, \ldots, k$ do
3.   $V_j \leftarrow \text{Create-Cluster}(G, V_{\perp}, t_j, R_j)$
4.   $V_{\perp} \leftarrow V_{\perp} \setminus V_j$
5. 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 $V_j$

1. $V_j \leftarrow \{t_j\}$, $U \leftarrow \emptyset$, $N \leftarrow \{\text{all neighbors of } t_j \text{ in } V_{\perp}\}$
2. while $N \neq \emptyset$ do
3.   pick an arbitrary vertex $v \in N$ and remove it from $N$
4.   if $d_G(v, t_j) \leq R_j \cdot D(v)$ then
5.     add $v$ to $V_j$
6.     add all the neighbors of $v$ in $V_{\perp} \setminus (U \cup V_j)$ to $N$
7.   else
8.     add $v$ to $U$
9. 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 $\pi$ 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 $\pi$ 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 $\text{ddim}$. 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 $\pi$ 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 $f : V \rightarrow K$, 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 $\pi$ 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 $t_j$ if 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.

\begin{lemma} [Lemma 2 in [10]] \label{lemma:terminal-partition}
The sets $V_1, \ldots, V_k$ constructed by Algorithm 2 constitute a terminal partition.
\end{lemma}

We say that a metric $(X, d)$ has doubling dimension $\dim$ if every ball of radius $r > 0$ can be covered by at most $2^{\dim}$ 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 $\left(\frac{4\Delta}{\delta}\right)^{\dim} = 2^{O(\dim \cdot \log \#)}$ 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.

\begin{theorem} \label{theorem:spr-on-trees}
Let $T$ be a tree and $r$ be an arbitrary vertex. Let $\pi$ 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.
\end{theorem}

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 \( \pi \) 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 \( \frac{\delta_{2r}(t_{60},t_{64})}{\delta_{2r}(t_{60},t_{64})} = \frac{52}{2} = 26 \).

### 3.1 Distortion Analysis

To better understand the final distortion bound 8, we analyze the Relaxed-Voronoi algorithm for a general \( R \geq 1 \), 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 \( \pi \) the root. Let \( t_1 \) be the first terminal w.r.t \( \pi \), 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, \( d_T(t_1, u') \leq d_T(t_1, u) + d_T(u, u') \leq R \cdot (d_T(t_v, u) + d_T(u, u')) = R \cdot d_T(t_v, u') \). 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 \( \ell_i \in K \) be the closest terminal to \( r_i \), thus \( \ell_i \in C_i \). 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 \( \pi_i \) be an ordering of the terminals in \( K_i \) according to increasing distance from \( r_i \). Note that \( \pi_i \) is just the order \( \pi \) restricted to \( K_i \).
We now prove the induction step. Let \( V \) be some terminal that belongs to the connected component \( C_i \) (in \( T \setminus V \)). By applying the induction hypothesis to the tree \( C_i \) with order \( \pi_i \), it holds that \( d_{T'}(t_i, t_i) \leq \frac{R+1}{R-1} \cdot d_T(t_i, t_i) \) as \( t_i \) is the first terminal in the order \( \pi_i \). Note that \( t_i \) will necessarily join the cluster of \( i \), therefore the edge \((e_i = \{s_i, r_i\})\) crosses the clusters of \( t_i \) and \( i \), which implies that there is an edge between \( t_i \) and \( 1 \) in \( T' \). See Figure 4 for illustration.

Next, we argue by induction on the number of terminals that for every terminal \( t \),
\[
d_{T'}(t, t) \leq \frac{R+1}{R-1} \cdot d_T(t, t).
\]
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 \( T \setminus V \)). By applying the induction hypothesis to the tree \( C_i \) with order \( \pi_i \), it holds that \( d_{T'}(t_i, t_i) \leq \frac{R+1}{R-1} \cdot d_T(t_i, t_i) \) as \( t_i \) is the first terminal in the order \( \pi_i \). Note that \( r_i \) will necessarily join the cluster of \( i \), therefore the edge \((e_i = \{s_i, r_i\})\) crosses the clusters of \( t_i \) and \( i \), which implies that there is an edge between \( t_i \) and \( 1 \) in \( T' \). See Figure 4 for illustration.

As \( r_i \) has a neighbor in \( V \) but did not join \( V \), necessarily \( d_T(t_i, r_i) > R \cdot D(r_i) = R \cdot d_T(r_i, 1) \). We conclude,
\[
d_{T'}(t_i, r_i) \leq d_{T'}(t_i, t_i) + d_{T'}(t_i, r_i) \\
\leq d_T(t_i, t_i) + \frac{R+1}{R-1} \cdot d_T(t_i, r_i) \\
\leq d_T(t_i, r_i) + \frac{R+1}{R-1} \cdot (D(r_i) + d_T(r_i, t_i)) \\
< d_T(t_i, r_i) + \left(1 + \frac{R+1}{R-1}\right) \cdot \frac{d_T(t_i, r_i)}{R} + \frac{R+1}{R-1} \cdot d_T(r_i, t_i) \\
= \frac{R+1}{R-1} \cdot (d_T(t_i, r_i) + d_T(r_i, t_i)) = \frac{R+1}{R-1} \cdot d_T(t_i, t_i).
\]

Finally, we show by induction that for every pair of terminals \( t_i, t_j \in K \setminus \{t_1\}, d_{T'}(t_i, t_j) < \left(\frac{R+1}{R-1}\right) \cdot d_T(t_i, t_j) \). If \( t_i, t_j \) belong to the same connected component of \( T \setminus V \) 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 \( V \). Clearly, the unique path in \( T \) from \( t_i \) to \( t_j \) goes through \( V \) and in particular through \( s_i \) and \( s_j \) (note that it is possible that \( s_i = s_j \)). Therefore, \( d_T(t_i, t_j) \geq d_T(t_i, s_i) + d_T(s_i, t_j) \). As \( s_i \in V \), it holds that \( d_T(t_i, s_i) \leq R \cdot D(s_i) \leq R \cdot d_T(t_i, s_i) \). Therefore,
\[
d_T(t_i, t_j) \leq d_T(t_i, s_i) + d_T(s_i, t_j) \leq (R+1) \cdot d_T(s_i, t_j).
\]
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 $d_{T_k}(t_i, t_i) \leq \frac{R+1}{R-1} \cdot d_T(t_i, t_i)$. As $\{t_i, \tilde{t}_i\}$ is an edge in $T_k$, the bound follows. Next, the bound on $d_{T_k}(t_j, t_i)$. 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)

$$d_{T_k}(t_i, t_j) \leq d_{T_k}(t_1, t_i) + d_{T_k}(t_1, t_j)$$

$$\leq \frac{R+1}{R-1} \cdot (d_T(t_i, s_i) + d_T(t_j, s_j))$$

$$\leq \frac{(R+1)^2}{R-1} \cdot d_T(t_i, t_j).$$

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 $\pi$ be Gonzalez’s order. Let $R_i = 2 \cdot 2^Z$, where $Z_1, \ldots, Z_k$ are i.i.d. variables sampled according to the distribution $\text{EXP}(c \cdot \ddim)$ for large enough constant $c$. Then the expected distortion returned by the Relaxed-Voronoi algorithm for the M0E problem is $O(\ddim)$.

**Proof.** Consider a point $x \in X$, and let $i_x$ be the minimal index such that $d(t_{i_x}, t_{i_x}) \leq D(x)$. 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 $f(x) = t_{i_x}$. 

$$f(x) = t_{i_x}.$$
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 $i \geq 1$, $|B(t, v, 2^i \cdot D(v)) \cap K_x| \leq |B(t_x, (2^i + 1) \cdot D(v)) \cap K_x| = 2^{O(i \cdot \ddim)}$.

\begin{lemma}
For every $x \in X$, $E[d(x, f(x))] = O(1) \cdot D(x)$.
\end{lemma}

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

By the union bound, the probability that some terminal from $K_i$ covers $x$ is bounded by $|K_i| \cdot e^{-\frac{i}{2} \cdot \ddim \cdot i}$. We conclude that for large enough constant $c$,
\[ E[d(x, f(x))] \leq 2^i \cdot D(x) + \sum_{i=3}^{\infty} \Pr[f(x) \in K_i] \cdot 2^i \cdot D(x) \]
\[ = 4 \cdot D(x) + D(x) \cdot \sum_{i=3}^{\infty} 2^{O(i \cdot \ddim) \cdot \frac{i}{2} \cdot \ddim \cdot i} \cdot 2^i = O(D(x)) . \]
\end{proof}

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
\[ E[d(f(x), f(y))] \leq E[d(f(x), x)] + d(x, y) + E[d(y, f(y))] \]
\[ = O(D(x) + D(y)) + d(x, y) = O(1) \cdot d(x, y) . \]

Thus from now on we can assume that $\epsilon$ 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 $S_j$. 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 $C_j$. Set $R_x = \frac{d(t_j, x)}{D(x)}$, $R_y = \frac{d(t_j, y)}{D(y)}$. Assuming w.l.o.g. that $R_x \leq R_y$, we get
\[ R_y = \frac{d(t_j, y)}{D(y)} \leq \frac{d(s, u) + d(v, u)}{D(x) - d(v, u)} \leq \frac{R_x \cdot D(x) + \epsilon \cdot D(x)}{D(x) - \epsilon \cdot D(x)} \leq 1 + \epsilon \leq \frac{1 + \epsilon}{1 - \epsilon} \cdot R_x < (1 + 3\epsilon) \cdot R_x . \]

Assuming that $t_j$ settles $\{x, y\}$, using the memoryless property we can bound the probability that $t_j$ cuts $\{x, y\}$.
\[ \Pr[C_j | S_j] = \Pr[R_j < R_y \mid R_j \geq R_x] \]
\[ = \Pr[Z_j < \ln(1 + 3\epsilon) \mid \Pr[Z_j < 3\epsilon] = 1 - e^{-3\epsilon \cdot c \cdot \ddim} \leq 6\epsilon \cdot c \cdot \ddim . \]

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,
\[ E[d(f(x), f(y)) \mid C_j] \leq d(t_j, x) + d(x, y) + E[d(y, f(y)) \mid C_j] \]
\[ = d(t_j, x, y) + O(\frac{1}{\epsilon}) \cdot d(x, y) . \]

For $i \geq 1$, denote by $K_i \subseteq K_x \cup K_y$ the set of terminals at distance $[2^{i-1}, 2^i) \cdot \min\{D(x), D(y)\}$ from $\{x, y\}$. By packing arguments, $|K_i| = 2^{O(i \cdot \ddim)}$. 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 $2^{-\Omega(i \cdot \ddim)}$. We conclude,

$$
\mathbb{E}[d(f(x), f(y))] = 
\sum_j \Pr[S_j] \cdot \Pr[C_j | S_j] \cdot \mathbb{E}[d(f(x), f(y)) | C_j]
$$

\begin{align}
&\leq 6\epsilon \cdot c \cdot \ddim \cdot \sum_j \Pr[S_j] \cdot (d(t_j, \{x, y\}) + O\left(\frac{1}{j}\right) \cdot d(x, y)) \\
&= O(\ddim) \cdot d(x, y) + O(\epsilon \cdot \ddim) \cdot \left(4 + \sum_{i \geq 3} 2^{-\Omega(i \cdot \ddim)} \cdot 2^i\right) \cdot \min\{D(x), D(y)\} \\
&= O(\ddim) \cdot d(x, y).
\end{align}

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 $\pi$ be arbitrary, and let $R_j = \epsilon^{z_j}$, where $Z_1, \ldots, Z_k$ are i.i.d. variables sampled according to distribution $\text{EXP}(c \cdot \ln k)$ 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 $\mathbb{E}_{f \sim D}[d(f(u), f(v))] \leq \alpha \cdot d_G(v, u)$. 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 than $r$) has a parent vertex. Our algorithm has two phases. In the first phase we sweep the tree upwards from the leaves to the root. For a vertex $v$, denote by $d(v)$ the distance from $v$ to it’s closest terminal among its descendants ($\infty$ 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 programming fashion according to the order induced by the tree. At the beginning each leaf $v$ know $d(v)$ (0 if terminal and $\infty$ otherwise). Then, iteratively each internal vertex $v$ computes $d(v) = \min\{d(v_i) + d(v_i, v)\}$ 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 $\pi$, 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 $\pi$. 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


Relaxed Voronoi: A Simple Framework for Terminal-Clustering Problems


