Partitioning Metric Spaces

Computational and Metric Geometry

Instructor: Yury Makarychev

1 Minimum Multiway Cut Problem

1.1 Preliminaries

Definition 1.1. *We are given a graph $G = (V, E)$ and a set of terminals $T = \{s_1, \dots, s_k\} \subset V$. We need to partition the graph into k pieces $P_1, \dots, P_k$ such that $s_i \in P_i$. Our goal is to minimize the number of cut edges.*

When $k = 2$, Multiway Cut is just the minimum s—t cut problem and can be solved in polynomial-time. However, when $k \geq 3$ the problem is NP-hard.

Exercise 1. *Design a combinatorial 2-approximation algorithm for Minimum Multiway Cut.*

1. *Consider $i \in \{1, \dots, k\}$. Show how to find a minimum cut $(S_i, \bar{S}_i)$ that separates s_i and $T \setminus \{s_i\}$ (that is, a cut $(S, \bar{S})$ such that $s_i \in S_i$ and $s_j \in \bar{S}_i$ for $j \neq i$).*

2. *Consider the following algorithm for the Minimum Multiway Cut problem. Find sets S_1, …, S_k as described above. Remove all edges cut by cuts $(S_1, \bar{S}_1)$, …, $(S_k, \bar{S}_k)$. We get a partition of G into a number of connected components. Let P_i be the connected component that contains s_i. If a connected component does not contain any terminals, merge it with an arbitrary set P_i. Show that this algorithm gives 2 approximation for the problem.*

We will see how to get a 3/2 approximation using linear programming. Consider the following relaxation for the problem. For every vertex u, we introduce k LP variables $\bar{u}_1, \dots, \bar{u}_k$ and let $\bar{u} = (\bar{u}_1, \dots, \bar{u}_k)$. Let $\Delta = \{x \in \mathbb{R}^d : x_1 + \dots x_k = 1, x_1 \geq 0, \dots, x_k \geq 0\}$; Δ is a simplex with vertices $e_1 = (1, 0, \dots, 0)$, $e_2 = (0, 1, 0, \dots, 0)$, …, $e_k = (0, \dots, 0, 1)$ (see Figure 1). We write the following linear program.

$$\begin{aligned} &\text{minimize } \frac{1}{2} \sum_{(u,v)\in E} \|\bar{u} - \bar{v}\|_1 \\ &\text{subject to} \\ &\qquad \bar{s}_i = e_i && \text{for every } i, \\ &\qquad \bar{u} \in \Delta && \text{for every } i. \end{aligned}$$

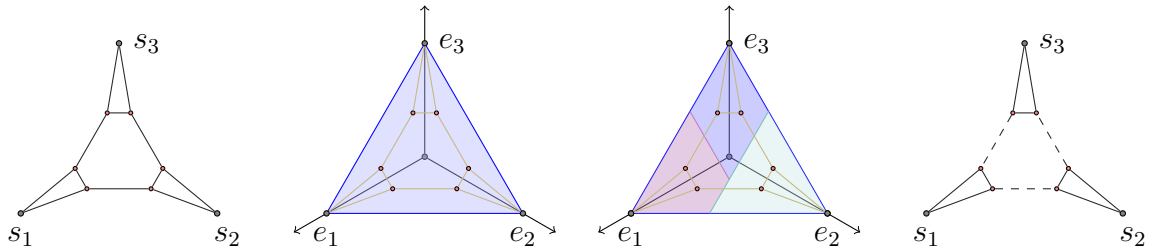

Figure 1: The figure shows (1) an input graph, (2) a feasible LP solution, (3) a random partitioning, and (4) the corresponding cut in the graph.

Exercise 2. *Show that this program is indeed a linear program. Hint: Introduce additional LP variables y_{uvi}. Write $y_{uvi} \geq \bar{u}_i - \bar{v}_i$ and $y_{uvi} \geq \bar{v}_i - \bar{u}_i$. Replace each term $\|\bar{u} - \bar{v}\|_1$ in the objective function with $\sum_{i=1}^{k} y_{uvi}$. Show that the obtained program is equivalent to our original program.*

We denote the value of the optimal solution by OPT, and the value of LP by LP.

Claim 1.2. *The LP is a relaxation of the problem. That is, $\mathsf{LP} \leq \mathsf{OPT}$.*

Proof. Consider the optimal solution $P_1, \ldots, P_k$. Let $\bar{u} = e_i$ for $u \in P_i$. Clearly, $\bar{u}$ is a feasible LP solution. We compute the value of this solution. Consider an edge $e = (u, v)$. Suppose that $u \in P_i$ and $v \in P_j$. The contribution of e to the objective function is

$$\frac{\|u - v\|_1}{2} = \frac{\|e_i - e_j\|_1}{2} = \begin{cases} 1, & \text{if } i \neq j, \\ 0, & \text{if } i = j. \end{cases}$$

That is, the contribution of e is 1 if e is cut, and 0, otherwise. Thus the total contribution of all edges equals the number of cut edges. We get that the value of the LP solution $\bar{u}$ is OPT. Therefore, $\mathsf{LP} \leq \mathsf{OPT}$. □

1.2 Rounding Scheme for $k = 2$

It is instructive to first consider the case $k = 2$. Define $d(u, v) = \|\bar{u} - \bar{v}\|_1/2$. Let us choose r uniformly at random from $(0, 1)$. Let $P_1 = B_r(s_1)$ and $P_2 = V \setminus P_1$. Note that (P_1, P_2) is a feasible solution to Multiway Cut:

- Since $d(s_1, s_1) = 0 < r$, we have $s_1 \in B_r(s_1) = P_1$.
- Since $d(s_1, s_2) = 1 > r$, we have $s_2 \notin B_r(s_1) = P_1$ and thus $s_2 \in P_2$.

Let us compute the expected number of cut edges. Consider an edge (u, v). Note that (u, v) is cut if and only if either $u \in B_r(s_1)$ and $v \notin B_r(s_1)$ or $u \notin B_r(s_1)$ and $v \in B_r(s_1)$. That is, if and only if $d(u, s_1) \leq r < d(v, s_1)$ or $d(v, s_1) \leq r < d(u, s_1)$, or equivalently

$$\min(d(u, s_1), d(v, s_1)) \leq r < \max(d(u, s_1), d(v, s_1)).$$

Since we sample r from a segment of length 1, the probability that

$$r \in [\min(d(u,s_1), d(v,s_1)), \max(d(u,s_1), d(v,s_1)))$$

is

$$\max(d(u,s_1), d(v,s_1))) - \min(d(u,s_1), d(v,s_1)) = |d(u,s_1) - d(v,s_1)| \leq d(u,v).$$

By the linearity of expectation, we get that the expected number of cut edges is at most

$$\sum_{e \in E} \Pr(e \text{ is cut}) \leq \sum_{e \in E} d(u,v) = \mathsf{LP}.$$

We showed that we get a solution of value at most OPT in expectation[1].

1.3 Rounding Scheme

Now we consider the general case when $k \geq 3$. We need to partition the graph into k pieces. Let us try to generalize the approach we used for $k = 2$. We again choose a random radius, and consider balls $B_r(s_1), \ldots, B_r(s_k)$. Note that now these balls are not necessarily disjoint: the distance between every s_i and s_j is 1 and so balls $B_r(s_i)$ and $B_r(s_j)$ might intersect if $r > 1/2$. In general, $\{B_r(s_1), \ldots, B_r(s_k)\}$ is not a partition of V.

Exercise 3. *One way to deal with this problem is to sample r uniformly at random from $(0, 1/2)$. Then sets $P_1 = B_r(s_1)$, ..., $P_{k-1} = B_r(s_{k-1})$, and $P_k = V \setminus (P_1 \cup \ldots P_{k-1})$ are disjoint and define a valid partition of V. Show that the expected cost of this partition is at most $2\mathsf{LP}$.*

We resolve this problem as follows. We choose a random permutation π of $\{1, \ldots, k\}$ and process terminals according to the order defined by π: $s_{\pi(1)}, s_{\pi(2)}, \ldots, s_{\pi(k-1)}$. When we process terminal s_i, we add to P_i those vertices in $B_r(s_i)$ that were not added previously to any set P_j. We present the algorithm in Figure 2.

Claim 1.3. *The algorithm returns a feasible solution.*

Proof. The claim follows from the fact that $s_i \in B_r(s_i)$ and $s_i \notin B_r(s_j)$ for every $j \neq i$. □

Now we compute the expected cost of the solution. Consider an edge $e = (u, v)$. Note that

$$d(u, s_i) = \frac{\|\bar{u} - e_i\|}{2} = \frac{(1 - \bar{u}_i) + \sum_{j \neq i} \bar{u}_j}{2} = \frac{(1 - \bar{u}_i) + (1 - \bar{u}_i)}{2} = 1 - \bar{u}_i.$$

Let

$$A_i = \min(d(u, s_i), d(v, s_i)) \quad \text{and} \quad B_i = \max(d(u, s_i), d(v, s_i)).$$

[1] In fact, we always get an optimal solution, since we never get a solution better than an optimal solution (there is no such solution), and thus we cannot get a solution which is worse than an optimal — if we did the expected value would be worse than OPT.

Approximation Algorithm for Minimum Multiway Cut

Input: graph $G = (V, E)$ and a set of terminals $T = \{s_1, \ldots, s_k\} \subset V$.
Output: a partition $P_1, \ldots, P_k$ of V such that $s_i \in P_i$.

solve the LP relaxation for Multiway Cut. Define $d(u, v) = \frac{1}{2}\|\bar{u} - \bar{v}\|_1$.
choose a random permutation π of $\{1, \ldots, k\}$
choose r uniformly at random from $(0, 1)$
let $A = \varnothing$
for $i = \pi(1), \pi(2), \ldots, \pi(k-1)$ do
  let $P_i = B_r(s_i) \setminus A$
  let $A = A \cup P_i$
let $P_{\pi(k)} = V \setminus A$
return partition $P_1, \ldots, P_k$

Figure 2: Approximation algorithm for Multiway Cut

We have,

$$B_i - A_i = |d(u, s_i) - d(v, s_i)| = |\bar{u}_i - \bar{v}_i|.$$

We may assume without loss of generality that

$$A_1 \leq A_2 \leq \cdots \leq A_k. \tag{1}$$

Let us write $i \prec j$ if $\pi^{-1}(i) < \pi^{-1}(j)$ (i precedes j in the order defined by π). We say that an index i settles the edge (u, v) if i is the first index w.r.t. π such that $u \in B_r(s_i)$ or $v \in B_r(s_i)$ (or both). In other words, index i settles edge e if

$$A_i \leq r \text{ and } A_j > r \text{ for all } j \prec i. \tag{2}$$

Let $\mathcal{E}_i$ be the event that i settles (u, v). Note that at most one of the events $\mathcal{E}_i$ happens. (If no event $\mathcal{E}_i$ happens than u and v belong to $P_{\pi(k)}$, and the edge (u, v) is not cut.)

When the event $\mathcal{E}_i$ happens, we add either one or both of the vertices u and v to P_i. Note that in the former case, the edge (u, v) is cut since $u \in P_i$ and $v \notin P_i$; in the latter case, the edge (u, v) is not cut since $u, v \in P_i$. We conclude that

$$\Pr(e \text{ is cut}) = \sum_{i=1}^{k} \Pr(\mathcal{E}_i \text{ and } |\{u, v\} \cap B_r(s_i)| = 1) = \sum_{i=1}^{k} \Pr(\mathcal{E}_i \text{ and } A_i \leq r < B_i).$$

We are going to show now that $\Pr(\mathcal{E}_i | r) \leq 1/2$ for every $i > 1$. Consider the event $\mathcal{L}_i$ that $i \succ 1$. Event $\mathcal{L}_i$ happens with probability $1/2$. We claim that when $\mathcal{L}_i$ happens $\mathcal{E}_i$ does not

happen and therefore

$$\Pr(\mathcal{E}_i|r) \leq 1 - \Pr(\mathcal{L}_i|r) = \frac{1}{2}.$$

Assume to the contrary that both events happen. Then

- $r \geq A_i$ and $r < A_j$ for every $j \prec i$ (see (2)),
- and $1 \prec i$,

therefore, $r \geq A_i$ and $r < A_1$, and $A_i < A_1$, which contradicts to our assumption (1). We have,

$$\begin{aligned}
\Pr(e \text{ is cut}) &= \sum_{i=1}^{k} \Pr(\mathcal{E}_i \text{ and } A_i \leq r < B_i) = \sum_{i=1}^{k} \mathbb{E}_r\left[\Pr(\mathcal{E}_i|r) \Pr(A_i \leq r < B_i|r)\right] \\
&\leq (B_1 - A_1) + \sum_{i=2}^{k} \frac{B_i - A_i}{2} = \frac{B_1 - A_1}{2} + \sum_{i=1}^{k} \frac{B_i - A_i}{2} \\
&= \frac{|\bar{u}_1 - \bar{v}_1|}{2} + \sum_{i=1}^{k} \frac{|\bar{u}_i - \bar{v}_i|}{2} = \frac{|\bar{u}_1 - \bar{v}_1| + \|\bar{u} - \bar{v}\|_1}{2}.
\end{aligned}$$

Observe that

$$\|u - v\|_1 \geq |u_1 - v_1| + \left|\sum_{i=2}^{k} u_i - \sum_{i=2}^{k} v_i\right| = |u_1 - v_1| + |(1 - u_1) - (1 - v_1)| = 2|u_1 - v_1|.$$

Thus $\Pr(e \text{ is cut}) \leq 3\|\bar{u} - \bar{v}\|_1/4$. By the linearity of expectation, the expected number of cut edges is at most

$$\sum_{(u,v)\in E} \frac{3}{4}\|\bar{u} - \bar{v}\|_1 = \frac{3\,\mathsf{LP}}{2} \leq \frac{3\,\mathsf{OPT}}{2}.$$

We proved that our algorithm gives a 3/2 approximation.

Remark 1.4. *Our analysis showed that we find a solution of expected cost* $\frac{3\mathsf{OPT}}{2}$*. Note that this implies that for every* $\varepsilon > 0$ *with probability at least* ε *we find a solution of cost at most* $\frac{3\mathsf{OPT}}{2(1-\varepsilon)}$ *(by Markov's inequality). Hence by running the algorithm polynomially many times and returning the best solution, we can find a solution of cost at most*

$$\left(\frac{3}{2} + \frac{1}{p(n)}\right) \cdot \mathsf{OPT}$$

with high probability for every fixed polynomial $p(n)$*.*

The results presented in this section are due to Calinescu, Karloff, and Rabani.

2 Multicut Problem

Definition 2.1. *We are given a graph $G = (V, E)$ and a set of source-terminal pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$. We need to cut the graph into pieces such that each (s_i, t_i) pair is separated. Our objective is to minimize the number of cut edges.*

We use the following LP relaxation for Multicut.

$$\begin{aligned} \text{minimize} \quad & \sum_{(u,v)\in E} d(u,v) \\ \text{subject to} \quad & \\ & d(s_i, t_i) = 1 \quad \text{for every } i \\ & d(\cdot,\cdot) \text{ is a semi-metric on } V. \end{aligned}$$

Question 1. *Why do not we write an ℓ_1-relaxation similar to the relaxation for Multiway Cut?*

Answer: We do not know which terminals are separated and which terminals are not separated. Say, we cannot require that $\bar{s}_i = e_i$ and $\bar{s}_j = e_j$ since s_i and s_j possibly lie in one piece of the optimal cut; similarly, we cannot require that $\bar{s}_i = e_k$ and $\bar{s}_j = e_k$ since s_i and s_j possibly lie in different pieces of the cut. Finally, we cannot just require that $\|\bar{s}_i - \bar{t}_i\|_1 = 1$ since this constraint is not a linear constraint.

Let $\mathcal{P}$ be the optimal solution. Denote the piece that u belongs to by $\mathcal{P}(u)$. The intended LP solution corresponding to $\mathcal{P}$ is

$$d(u,v) = \begin{cases} 1, & \text{if } \mathcal{P}(u) \neq \mathcal{P}(v), \\ 0, & \text{otherwise.} \end{cases}$$

Solution $d(\cdot,\cdot)$ is a feasible LP solution since (1) $\mathcal{P}(s_i) \neq \mathcal{P}(t_i)$ and so $d(s_i, t_i) = 1$ and (2) $d(\cdot,\cdot)$ is a semi-metric. Therefore, $\mathsf{LP} \leq \mathsf{OPT}$. Consider the following algorithm for the problem.

Approximation Algorithm for Multicut
Input: graph $G = (V, E)$ and a set of source-terminal pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$.
Output: a partition P of V such that each pair (s_i, t_i) is separated by P.

solve the LP relaxation for Multicut
choose a random permutation π of $\{1, \ldots, k\}$
choose r uniformly at random from $(0, 1/2)$
let $A = \varnothing$
for $i = \pi(1), \pi(2), \ldots, \pi(k)$ do
  let $P_i = B_r(s_i) \setminus A$
  let $A = A \cup P_i$
let $P_{k+1} = V \setminus A$
return partition $\{P_i\}$ *//some sets in the partition may be empty*

The algorithm is very similar to the algorithm for Minimum Multiway Cut. Essentially, the only difference is that we choose r uniformly at random from $(0, 1/2)$ rather than from $(0, 1)$. Note that vertex s_i does not necessarily lies in P_i; it might lie in some P_j for $j \preceq i$.

Lemma 2.2. *The algorithm always finds a feasible solution.*

Proof. We need to prove that s_i and t_i belong to different pieces of the cut for every i. Let $P_j = \mathcal{P}(s_i)$ be the piece s_i belongs to. Note that either $j = i$ or $j \prec i$. In particular, $j \neq k + 1$. For every $u, v \in P_j$,

$$d(u, v) \leq d(u, s_j) + d(s_j, v) \leq 2r < 1.$$

Since $d(s_i, t_i) = 1$, terminal t_i cannot belong to P_j. □

We now prove that the algorithm gives $O(\log k)$ approximation.

Lemma 2.3. *Partition P cuts at most $O(\log k\, \mathsf{OPT})$ edges in expectation.*

Proof. Consider an edge $e = (u, v)$. Define

$$A_i = \min(d(u, s_i), d(v, s_i)) \quad \text{and} \quad B_i = \max(d(u, s_i), d(v, s_i)).$$

Assume without loss of generality that $A_1 \leq A_2 \leq \cdots \leq A_k$. We say that an index i settles (u, v) if i is the first index w.r.t. π such that $u \in B_r(s_i)$ or $v \in B_r(s_i)$ or both; i.e., index i settles edge e if $A_i \leq r$ and $A_j > r$ for all $j \prec i$. Let $\mathcal{E}_i$ be the event that i settles (u, v). Note that at most one of the events $\mathcal{E}_i$ happens. We have,

$$\Pr(e \text{ is cut}) = \sum_{i=1}^{k} \Pr(\mathcal{E}_i \text{ and } A_i \leq r < B_i).$$

We prove that $\Pr(\mathcal{E}_i | r) \leq 1/i$. Consider the set $I = \{i : A_i \leq r\}$. This set depends only on r and not on the permutation π. Note that the first index in I w.r.t. π settles (u, v). Each element of I may be the first index w.r.t. π with the same probability $1/|I|$. If $i \in I$ then also $1, \ldots, i - 1 \in I$ since $A_1 \leq \cdots \leq A_{i-1} \leq A_i$. Therefore, $\Pr(\mathcal{E}_i | r) \leq \frac{1}{i} \cdot \Pr(i \in I | r) + 0 \cdot \Pr(i \notin I | r) \leq 1/i$. We have,

$$\Pr(e \text{ is cut}) \leq \sum_{i=1}^{k} \frac{\Pr(A_i \leq r < B_i)}{i} \leq 2 \sum_{i=1}^{k} \frac{d(u, v)}{i} = O(\log k)\, d(u, v).$$

We conclude that the expected number of cut edges is

$$\sum_{(u,v) \in E} O(\log k)\, d(u, v) \leq O(\log k)\, \mathsf{LP} \leq O(\log k)\, \mathsf{OPT}.$$ □

The first $O(\log n)$ approximation algorithm for Multicut was designed by Garg, Vazirani, and Yannakakis. The algorithm presented in this section is due to Calinescu, Karloff, and Rabani.

3 Padded Decomposition I

We have successfully used essentially the same algorithm to solve two different problems. In this section, we again use this algorithm; this time we apply it to an arbitrary metric space (X, d) on n points and get *a padded bounded stochastic decomposition* of X.

Definition 3.1. *Let (X, d) be a metric space on n points. An (α, Δ)–padded decomposition of X is a probabilistic distribution of partitions $\mathcal{P}$ of X such that*

- *The diameter of each cluster $C \in \mathcal{P}$ is at most Δ, $d(u, v) \leq \Delta$ for every $u, v \in C$.*
- *The probability that u and v are separated is at most $\alpha \cdot d(u, v)/\Delta$.*

Remark 3.2. *Some authors call this decomposition an α–separating Δ-bounded stochastic decomposition of X and call the decomposition that we define later in Theorem 5.1 a padded Δ-bounded stochastic decomposition.*

Padded Decomposition

Input: a finite metric space (X, d) and a parameter Δ.
Output: a $(O(\log |X|), \Delta)$–padded decomposition $\mathcal{P}$ of X.

choose a random ordering $\pi : \{1, \ldots, n\} \to X$ of X
choose r uniformly at random from $(0, \Delta/2)$
let $A = \varnothing$
for $i = 1, \ldots, n$ do
  let $C_i = B_r(\pi(i)) \setminus A$
  let $A = A \cup C_i$
return partition $\mathcal{P} = \{C_i\}$ *//some clusters in the partition may be empty*

Alternatively, we can formulate the algorithm as follows. We choose a random threshold $r \in (0, \Delta/2)$ and random ordering π on X. For each point u, consider the ball $B_r(u)$, find the first vertex in $B_r(u)$ w.r.t. to the order defined by π,

$$i = \min\{j : \pi(j) \in B_r(u)\},$$

and assign u to C_i (see the right diagram in Figure 3).

It is immediate that every set in partition has diameter at most Δ. The same argument as we used to prove that our algorithm for Multicut gives an $O(\log k)$ approximation shows that the probability that u and v are separated is at most $\alpha \cdot d(u, v)/\Delta$ where $\alpha = O(\log n)$. We proved he following theorem.

Theorem 3.3. *For every metric space (X, d) and every $\Delta > 0$, there is an $(O(\log n), \Delta)$–padded decomposition of X (where $n = |X|$). Moreover, there is a polynomial–time randomized algorithm that finds a $(O(\log n), \Delta)$–padded decomposition.*

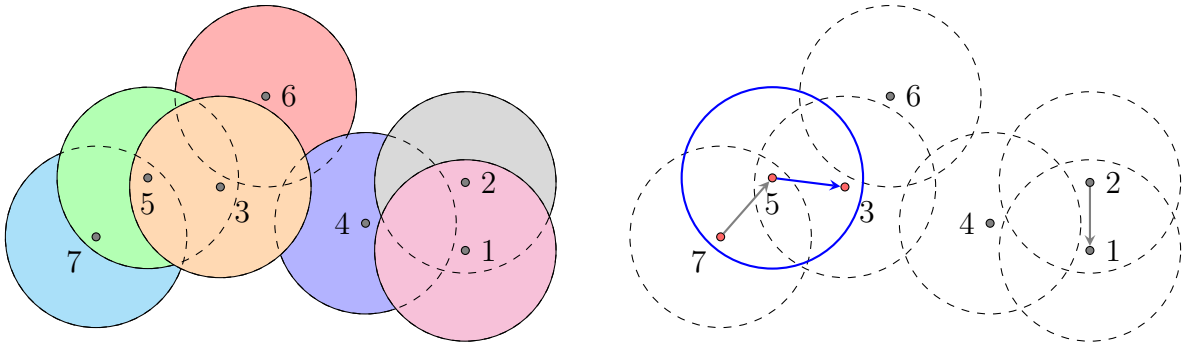

Figure 3: Padded Decomposition. Vertices are listed w.r.t. random ordering π. The left diagram shows how the algorithm works. The right diagram shows that each vertex u is assigned to the cluster C_i with $i = \min\{j : \pi(j) \in B_r(u)\}$. For example, the ball around vertex 5 consists of vertices 3, 5 and 7; thus vertex 5 is assigned to the cluster C_3.

4 Hierarchically Well-Separated Trees

Let T be a tree on V with positive edge lengths. Recall that the tree metric $d_T(\cdot, \cdot)$ is the shortest metric on T. Tree metrics are much simpler than general metrics, and many problems that cannot be solved on arbitrary metrics can be solved on tree metrics. This observation leads to a natural question, can an arbitrary metric be "well-approximated" with a tree metric? An arbitrary metric may be very far from being a tree metric, but as we will show every metric can be approximated with a distribution of tree metrics.

Definition 4.1. *Let us say that a metric space (X, d) embeds into a distribution of dominating trees with distortion α if there is a probabilistic distribution of tree metrics d_T on $X' \supset X$ such that for every $u, v \in X$:*

$$d(u, v) \leq d_T(u, v) \text{ and } \mathbb{E}_T [d_T(u, v)] \leq \alpha d(u, v).$$

Note that T is a tree on a superset X' of X. We call vertices in $X' \setminus X$ *Steiner* vertices.

Question 2. *We showed that every metric space embeds into a distribution of cut metrics with distortion $O(\log n)$. Every cut metric trivially is a tree metric. Is an embedding in a distribution of cuts an embedding into a distribution of dominating trees? Why?*

We show that every metric space embeds into a distribution of trees.

Theorem 4.2. *Let (X, d) be a metric space on n points. Assume that $d(u, v) \geq 1$ for every $u, v \in V$. Let $\Delta_0 = \text{diam}(X) = \max_{u,v \in X} d(u, v)$ be the diameter of X. Then there is an embedding of X into a distribution of dominating trees with distortion $O(\log n (\log \Delta_0 + 1))$*

Proof. We find a padded decomposition $\mathcal{P}_1$ of X with $\Delta = \Delta_0/2$, then find a subpartition of each cluster of $\mathcal{P}_1$ with $\Delta = \Delta_0/4$ and so on. At level $i + 1$ of recursion, we partition

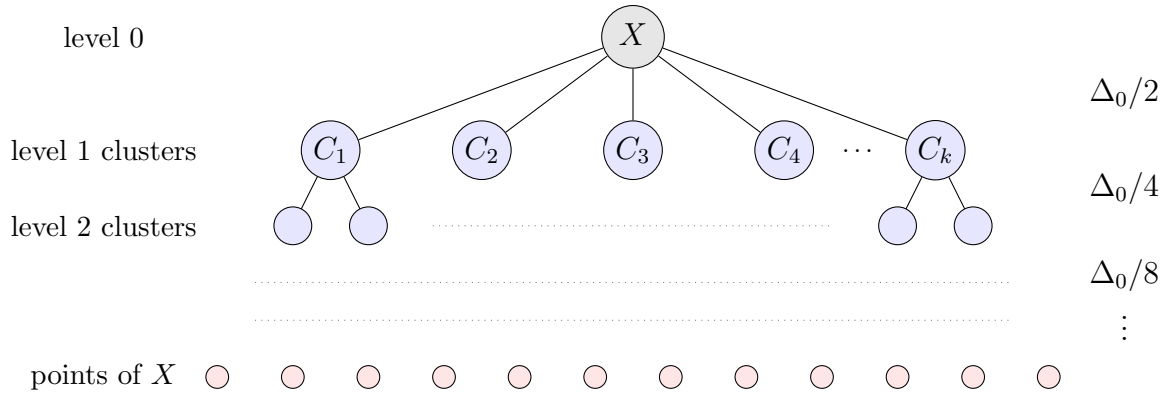

Figure 4: Hierarchically Well-Separated Tree

each cluster obtained at level i with paramter $\Delta = \Delta_0/2^{i+1}$. We recurse until $\Delta/2^i < 1$. We obtain *a hierarchical decomposition* of X. Consider the corresponding decomposition tree T:

- There is a vertex in T for each cluster in the hierarchical decomposition (including the top level cluster X); we denote the vertex for C by v_C.
- The tree is rooted at v_X.
- v_A is a child of v_B if A is a cluster in the partition of B.
- The level $level(v_C)$ of a vertex v_C is the depth of v_C in the tree (the number of edges on the path from the root to v_C); the level of cluster C is $level(v_C)$.
- All edges between vertices at levels i and $i+1$ have length $\Delta_0/2^{i+1}$.
- We identify leaf vertices with elements of X. Specifically, for each leaf vertex v_C, we have $\mathrm{diam}(C) < 1$ and thus C contains a single point. We identify v_C with the single element of C.
- It may happen that a partitioning of some cluster C is trivial: the only sub-cluster of C is C. In this case, we keep two vertices v_C (one is at level i; the other is at level $i+1$).

Formally, we use procedure FindHST$(X, \Delta_0/2)$, presented in Figure 4, to find T. Let us say that two vertices u and v are separated at level i, if u and v lie in the same cluster at level $i-1$ but in different clusters at level $i+1$.

Claim 4.3. *T is a dominating tree; that is, $d_T(u,v) \geq d(u,v)$ for every $u, v \in X$.*

Proof. Let $u, v \in X$. Suppose that u and v are separated at level i. Then both u and v lie in one cluster C at level $i-1$, but lie in different clusters, A and B, at level i. Then

$$d_T(u,v) \geq d_T(v_A, v_B) = d_T(v_A, v_C) + d_T(v_B, v_C) = 2 \cdot \Delta_0/2^i \geq \mathrm{diam}(C) \geq d(u,v). \qquad \square$$

procedure FindHST(Y, Δ)

Input: a finite metric space (Y, d) and a parameter Δ.
Output: a rooted dominating tree T on $Y' \supset Y$; points of Y are leaves of T.

if $\Delta < 1$ return a tree on the single vertex of Y
find a random (α, Δ)–padded decomposition $\mathcal{P}$ of Y.
for each cluster C in $\mathcal{P}$
  $T_C = \text{FindHST}(C, \Delta/2)$
create a new vertex v_Y
let v_Y be the root of T
connect v_Y with roots v_C of trees T_C by edges of length Δ
return the obtained tree T

Figure 5: Procedure FindHST(Y, Δ) generates a random dominating tree for Y.

Claim 4.4. $\mathbb{E}_T[d_T(u,v)] \leq O(\alpha \cdot (h+1))$, *where* $\alpha = O(\log n)$ *and* $h = \lfloor \log_2 \Delta_0 \rfloor$ *is the depth of* T.

Proof. The probability that u and v are separated at level i is at most $\alpha \cdot \frac{d(u,v)}{\Delta_0/2^i} = 2^i\alpha/\Delta_0$. If u and v are separated at level i, the distance between them is

$$2\sum_{j=i}^{h} \Delta_0/2^j = O(\Delta_0/2^i).$$

Thus the expected distance between u and v in T is at most

$$\sum_{i=0}^{h} \left((\Delta_0/2^i) \cdot (\alpha \cdot 2^i \cdot d(u,v)/\Delta_0)\right) = O\left(\sum_{i=0}^{h} \alpha d(u,v)\right) = O\left(\alpha \cdot (h+1) \cdot d(u,v)\right). \qquad \square$$

This concludes the proof of Theorem 4.2. $\square$

We obtained an embedding into distribution of dominating trees with distortion

$$O(\log n(\log \Delta_0 + 1)).$$

Note that every tree in the distribution satisfies the following properties: points of X are leaves of the tree, all leaves are at the same level, all edges between levels i and $i+1$ have the same length (for a given i). We call a tree that satisfies these properties a hierarchically well-separated tree (HST).

Results presented in this section are due to Bartal.

5 Padded Decomposition II

Our bound on the distortion depends on the depth of the decomposition tree. That happens because we first show that each level contributes at most $\alpha d(u,v)$ and then add up contributions of all $h+1$ levels. We now show how to improve the distortion to $O(\log n)$. To this end, we get a padded decomposition with stronger guarantees.

Theorem 5.1. *Let (X,d) be a metric space on n points, and $\Delta > 0$. There is a probabilistic distribution of partitions $\mathcal{P}$ of X that satisfies the following properties.*

- *Each cluster C in $\mathcal{P}$ has diameter at most Δ.*
- *For every $x \in X$ and $\rho \in (0, \Delta/8)$,*

$$\Pr\left(B_\rho(x) \not\subset \mathcal{P}(x)\right) \leq \alpha(x)\frac{\rho}{\Delta},$$

where $\alpha(x) = O(\log \frac{|B_\Delta(x)|}{|B_{\Delta/8}(x)|})$.

Remark 5.2. *The decomposition $\mathcal{P}$ is an (α,Δ)–padded decomposition. Indeed for every x and y,*

$$\Pr\left(\mathcal{P}(x) \neq \mathcal{P}(y)\right) = \Pr\left(y \notin \mathcal{P}(x)\right) \leq \Pr\left(B_{d(x,y)(x)} \not\subset \mathcal{P}\right)(x) \leq \alpha(x) d(x,y)/\Delta,$$

where $\alpha(x) = O(\log \frac{|B(x,\Delta)|}{|B(x,\Delta/8)|}) = O(\log n)$.

Proof. We slightly change the algorithm that computes the padded decomposition.

Padded Decomposition II

Input: a finite metric space (X,d) and a parameter Δ.
Output: a padded decomposition $\mathcal{P}$ of X.

choose a random ordering $\pi : \{1,\ldots,n\} \to X$ of X
choose r uniformly at random from $(\Delta/4, \Delta/2)$
let $A = \varnothing$
for $i = 1,\ldots,n$ do
  let $C_i = B_r(\pi(i)) \setminus A$
  let $A = A \cup C_i$
return partition $\mathcal{P} = \{C_i\}$ *//some clusters in the partition may be empty*

The analysis of this algorithm is very similar to the one in Section 3. Consider a point x. Sort vertices of X by their distance to x; denote them by $u_1,\ldots,u_n$. Let $A_i = d(u_i,x) - \rho$ and $B_i = d(u_i,x) + \rho$ for every $i \in \{1,\ldots,n\}$. Then $A_1 \leq A_2 \leq \cdots \leq A_n$ and $B_1 \leq B_2 \leq \cdots \leq B_n$.

If $B_\rho(x) \cap B_r(u_i) \neq \varnothing$ but $B_\rho(x) \not\subset B_r(u_i)$, then $r \in [A_i, B_i)$. Let $\mathcal{E}_i$ be the event that i is the first index w.r.t. π such that $r \geq A_i$. Note that if $\mathcal{E}_i$ and $r \geq B_i$ then $B_\rho(x) \subset \mathcal{P}(x)$. Indeed, on one hand no vertex of $B_\rho(x)$ belongs to $\mathcal{P}(u_j)$ for $j \prec i$ because $r < A_j$; on the other hand, $B_\rho(x) \subset B_r(u_i)$ since $B_i \leq r$. Thus

$$\Pr\left(B_\rho(u) \not\subset \mathcal{P}(x)\right) \leq \sum_{i=1}^{n} \Pr\left(\mathcal{E}_i \text{ and } A_i \leq r < B_i\right).$$

Note that $\Pr\left(\mathcal{E}_i | r\right) \leq 1/i$ and $\Pr\left(A_i \leq r < B_i\right) \leq \frac{B_i - A_i}{\Delta/4} = 8\rho/\Delta$. Moreover, if $\Delta/2 < A_i$ then $r < A_i$ and thus $\mathcal{E}_i$ cannot happen. Therefore, if $i > |B_\Delta(x)|$ then $i \notin B_\Delta(x)$ (since all vertices u_i are sorted according to their distance to x) and hence

$$A_i = d(u_i, x) - \rho \geq \Delta - \rho > \Delta/2,$$

and hence $\Pr\left(\mathcal{E}_i\right) = 0$. Additionally, if $\Delta/4 > B_i$ then $r > B_i$ and $\Pr\left(A_i \leq r < B_i\right) = 0$. Therefore, if $i \leq |B_{\Delta/8}(x)|$ then $i \in B_{\Delta/8}(x)$ and $B_i = d(x, u_i) + \rho \leq \Delta/8 + \rho < \Delta/4$ and hence $\Pr\left(A_i \leq r < B_i\right) = 0$. We conclude that

$$\Pr\left(B_\rho(u) \not\subset \mathcal{P}(x)\right) \leq \sum_{i=|B_{\Delta/8}(x)|}^{|B_\Delta(x)|} \frac{1}{i} \times \frac{8\rho}{\Delta} = O\left(\log \frac{|B_\Delta(x)|}{|B_{\Delta/8}(x)|} \times \frac{\rho}{\Delta}\right).$$

□

Proof sketch: We use Theorem 5.1 to construct a distribution of dominating HSTs with distortion $O(\log n)$. We simply plug in the new padded decomposition in the construction from Theorem 4.2. Now, the probability that u and v are separated at level i is at most

$$O\left(\log \frac{|B_\Delta(u)|}{|B_{\Delta/8}(u)|} \times 2^i \times \frac{d(u,v)}{\Delta_0}\right)$$

if $d(u,v) \leq \Delta_0/2^{i+3}$, and at most 1, otherwise. If u and v are separated at level i, the distance between them is $O(\Delta_0/2^i)$. Thus the expected distance between u and v is at most

$$O(1) \sum_{i=0}^{h} \log \frac{|B_\Delta(u)|}{|B_{\Delta/8}(u)|} \times \frac{d(u,v)}{\Delta_0} = O(\log n) \frac{d(u,v)}{\Delta_0},$$

where $h \approx \log_2 \frac{\Delta_0}{8d(u,v)}$.

Remark 5.3. *Give another proof of Bourgain's theorem using embeddings into distributions of dominating trees. Prove a lower bound of $\Omega(\log n)$ on the distortion of embeddings into distributions of trees.*

The results presented in this section are due to Fakcharoenphol, Rao, and Talwar.

6 k-Median

There are many applications of padded decompositions and HSTs in theoretical computer science. We will not have time to discuss these applications in detail. We show below only one application of HSTs to demonstrate the general approach.

Definition 6.1. *We are given a metric space (X, d). The k-Median problem is to find a set S of k points in X so as to minimize $\sum_{u \in X} d(u, S)$.*

The problem can be solved exactly on tree metrics using dynamic-programming.

Exercise 4. *Design a polynomial-time algorithm that solves the problem on trees.*

Consider an instance of the k-Median problem. Let S^* be its optimal solution Compute an embedding of X in a distribution of dominating trees T with distortion $\alpha = O(\log n)$.

Exercise 5. *Show that we may assume that T is a tree on X (that is, T has no Steiner vertices).*

Hint: Consider a tree T from the distribution. Change the length of every edge going from a vertex to its left child to 0, double the length of every other edge. Then contract all edges of length 0.

Compute an optimal solution S' for k-Median on (X, d_T). We always have,

$$\sum_{u \in X} d(u, S') \leq \sum_{u \in X} d_T(u, S') \leq \sum_{u \in X} d_T(u, S^*).$$

We also have,

$$\mathbb{E}_T \left[\sum_{u \in X} d_T(u, S^*) \right] \leq \alpha \sum_{u \in X} d(u, S^*) = \alpha \cdot \mathsf{OPT}.$$

Therefore, S' is an α–approximate solution in expectation.

Remark 6.2. *This algorithm is due to Bartal. It was the first approximation algorithm for k-Median. Later Charikar, Guha, Tardos, and Shmoys gave a constant factor approximation algorithm for the problem. The best currently known algorithm for the problem is due to Li and Svensson.*