CS 598CSC: Topics in Graph Algorithms	Lecture date: 09/12/2024
Instructor: Chandra Chekuri	Scribe: CC

1 Low Diameter Decompositions

Let $G = (V, E)$ be a graph with non-negative edge lengths $\ell : E \to \mathbb{R}_+$. The shortest path distances induced by shortest paths in G define a metric space (V, d) where $d(u, v)$ is the shortest path distances between u and v in G. There are several different ways to decompose a graph based on the application in question. In the context of distances a useful notion is to decompose or partition the graph into subgraphs/clusters of small radius/diameter. More precisely, given a graph $G = (V, E)$ we would like to partition V into clusters with vertex sets $V_1, V_2, \ldots, V_h$ such that each V_i has diameter at most some given parameter δ. This is of course trivial by choosing the partition to consist of singletons. However, the goal in partitioning is to ensure that two vertices u, v that are close to each other, say $d(u, v) < \delta$, should ideally not be split apart into different clusters. It is clear that this is impossible to do this deterministically since the graph/metric space is connected. Consider the case of a line for instance. What we can hope for is a randomized partition that probabilistically ensures that any two points u, v are separated only with probability proportional to $d(u, v)/\delta$. The following is a formalization of this notion. We let $\mathcal{P}$ denote the set of all partitions of V and for a partition $P \in \mathcal{P}$ we let E_P be the set of edges/pairs that are separatd by P.

Definition 1. *Let $G = (V, E)$ be a graph with edge lengths $\ell : E \to \mathbb{R}_+$ inducing a distance $d : V \times V \to \mathbb{R}_+$ and let Δ be the diameter of the metric space (V, d). For a given $\delta \in [0, \Delta]$, a low-diameter decomposition with cutting probability parameter α is a probability distribution $\mathcal{D}$ over $\mathcal{P}$ (partitions of V) such that*

- *For any partition $P = (V_1, V_2, \ldots, V_h)$ in the support of the distribution, the diameter of each (V_i, d) is at most δ.*
- *For any given u, v, the probability that u and v are in different parts is at most $\alpha \cdot \frac{d(u,v)}{\delta}$. That is $\mathbf{P}[uv \in E_P] \leq \alpha \cdot \frac{d(u,v)}{\delta}$*

A low-diameter decomposition scheme with parameter α is a family of algorithms that given any $\delta \in [0, \Delta)$ generates a low-diameter decomposition with cutting/separation probability parameter at most α.

Strong vs weak diameter guarantee: A low-diameter decomposition is said to have the *strong diameter* guarantee if the diameter of each cluster V_i in the induced graphs $G[V_i]$ is at most δ. Note that the preceding definition does not require that because it is based on the metric clousure (V, d) of the given graph. The standard definition is called the *weak diameter* guarantee. Some applications require the strong diameter guarantee. We will, by default, work with weak diameter guarantee and mention strong diameter guarantee when needed.

Padded decomposition: The basic definition of low-diameter decomposition is often strengthened to require more. Given a point u let $B_d(u,r) = \{v \in V \mid d(u,v) \leq r\}$ be the ball of radius r around u. In padded decomposition we require that for each u, and for each $r \leq \delta$, the probability of $B(u,r)$ being contained in the same part is at least $e^{-\beta r}$.

Sparse covers: A sparse cover consists of several clusters $V_1, V_2, \ldots, V_h$. Each cluster should have weak/strong diameter at most δ. For each u, v with distance at most δ, there must be some cluster V_i that contains both u and v. And no vertex u must be in more than some number s of clusters. The techniques underlying sparse covers and low-diameter decompositions are related though we will mostly work only with low-diameter decompositions.

Main results: The main questions regarding low-diameter decomposition is the smallest α that one can obtain. It turns out that for general metric spaces $\alpha = O(\log n)$ is a tight bound for both strong and weak diameter guarantee; this was formally shown by Bartal [Bar96]. For planar graph metrics $\alpha = O(1)$ is achievable; weak diameter guarantee was first shown by Klein, Plotkin and Rao [KPR93] and the strong diameter guarantee was more difficult and was shown later. These are important results that have many applications. There is substantial work on extending the planar graph results to graphs that exclude a fixed minor - we will not dwell into these more advanced issues in this course. See [Fil24] for some recent work and pointers to literature on these topics.

2 CKR algorithm for weak diameter guarantee

Let (V,d) be a metric space with $|V| = n$. We had already seen the CKR-RandomPartition algorithm and analysis previously in the context of the MultiCut algorithm. Implicit in the algorithm is a metric partitioning scheme. We modify the algorithm to ensure that the weak diameter of each cluster is at most a given parameter δ. We will assume that $V = \{v_1, \ldots, v_n\}$.

CKR-RandomPartition:
Pick θ uniformly at random from $[0, \delta/2)$
Pick a random permutation σ of $\{1, 2, ..., n\}$
for $i = 1$ to n
$V_{\sigma(i)} = B_d(v_{\sigma(i)}, \theta) \setminus \bigcup_{j<i} V_{\sigma(j)}$
Output the partition formed by $V_1, V_2, \ldots, V_n$ (some of them will be empty)

The analysis is essentially the same as we did for Multicut so we simply state the lemmas and encourage the reader to see how to prove them via the previous analysis.

Lemma 1. *CKR-RandomPartition correctly outputs a partition of V into clusters, each of which has weak diameter at most δ.*

Lemma 2. *The probability that u and v are in different clusters is at most $2H_n \frac{d(u,v)}{\delta}$.*

2.1 A subtle modification in the algorithm and guarantee

We indicate a small but subtle change in the algorithm and the guarantee it provides which will be used later in an surprising way in an optimal algorithm for low-stretch spanning trees. The only change in the algorithm is in choosing the radius in the range $[\delta/4, \delta/2)$ rather than from $[0, \delta/2)$.

CKR-RandomPartition:

Pick θ uniformly at random from $[\delta/4, \delta/2)$
Pick a random permutation σ of $\{1, 2, ..., n\}$
for $i = 1$ to n
$V_{\sigma(i)} = B_d(v_{\sigma(i)}, \theta) \setminus \bigcup_{j<i} V_{\sigma(j)}$

Output the partition formed by $V_1, V_2, \ldots, V_n$ (some of them will be empty)

The guarantee about the radius/diameter remain the same.

Lemma 3. *CKR-RandomPartition correctly outputs a partition of V into clusters, each of which has weak diameter at most δ.*

The main difference is in the probability guarantee which is refinement of the previous bound.

Lemma 4. *The probability that u and v are in different clusters is at most $\frac{4d(u,v)}{\delta} \log(\frac{|B(u,\delta)|}{|B(u,\delta/8)|})$.*

Proof. We sketch the proof assuming familiarity with the argument from the previous lecture for Multicut. Fix u, v and think of V as $v_1, v_2, \ldots, v_n$. Let L_i, R_i be defined as before; distance of u, v from v_i with one of them at L_i and the other at R_i. We have $R_i - L_i \leq d(u, v)$. If $d(u, v) \geq \delta/8$ then the edge is going to get cut with constant probability and the bound is not giving anything interesting so we are primarily interested in $d(u, v) < \delta/8$. As before we assume that we have sorted the vertices that $L_1 \leq L_2 \ldots, L_n$.

We consider the event A_i which is that v_i is the *first* vertex to separate the pair u, v. We can argue as before that $\mathbf{P}[A_i] \leq \frac{1}{i}\,\mathbf{P}[\theta \in [L_i, R_i]$ and this is at most $\frac{1}{i})4d(u, v)/\delta$ since we are choosing the radius from $[\delta/4, \delta/2)$. This is because A_i happens only if i comes first among the first i vertices according the sorting. The new twist is that since we chose $\theta \in [\delta/4, \delta/2)$ and $d(u, v) < \delta/8$, no vertex $v_j \in B(u, \delta/8)$ can separate u, v because $L_j \leq d(v_i, u) < \delta/8$ and $R_j \leq L_j + d(u, v) \leq \delta/8 + \delta/8 \leq \delta/4$. Any such vertex will capture both u, v if they are not already separated. Similarly, any vertex $v_j \notin B(u, \delta)$ can cut the pair because $L_j \geq \delta - d(u, v) \geq \delta - \delta/8 \geq \delta/2$. Therefore, if A is the probability of cutting then

$$\mathbf{P}[A] \leq \sum_{j \in B(u,\delta) \setminus B(u,\delta/8)} \mathbf{P}[A_j] \leq \frac{4d(u, v)}{\delta} \sum_{j > |B(u,\delta/8)|, j \leq |B(u,\delta)|} \frac{1}{j} \leq \frac{4d(u, v)}{\delta} \log \frac{|B(u, \delta)|}{B(u, \delta/8)|}.$$

■

3 Randomized region growing for a strong diameter guarantee

We now describe an alternate scheme that also gives a strong diameter guarantee. This is by Bartal [Bar96]. For this we need some properties of the geometric distribution.

Geometric distribution: Recall that Geom(p) for a parameter $p \in (0,1]$ is the number of independent Bernoulli trials of a coin with bias p to see the first heads (here p is the probability of the coin coming up heads). If $X \simeq \text{Geom}(p)$ then $\mathbf{P}[X = k] = (1-p)^{k-1}p$. The geometric distribution is a memory less discrete distribution (only one). That is, $\mathbf{P}[X > m+n \mid X > n] = \mathbf{P}[X > m]$; this is intuitive from the definition but can be verified formally. The corresponding continuous time distribution is the exponential distribution. The expectation of a random variable X distributed according to Geom(p) is $1/p$ and the variances is $(1-p)/p^2$.

Given parameter δ, we can shrink all edges with length at most δ/n^2 and discretize edge lengths to integer values. This does not affect the analysis by more than constant factors. Since we want strong diameter guarantee we will work with the underlying graph and not the metric completion. However, we will assume that for each edge $e = uv$ that $\ell(e)$ is equal to the shortest path distance between u and v in G.

RandomCarving:(G, δ)
  Let $p = \min\{1, \frac{4 \ln n}{\delta}\}$
  While $G \neq$ do
    Let v be an arbitrary vertex
    $R_v = \text{Geom}(p)$.
    Let $C_v = B_G(v, R_v)$ be the ball of radius R_v in current graph
    Add C_v to clusters
    $G \leftarrow G - C_v$
  EndWhile
  Output the clusters found during the algorithm

Note that $G - C_v$ is the graph obtained by removing all the vertices in C_v. This may increase the shortest path distances in the remaining graph. We also remark that the choice of p depends on n, the original number of vertices in the graph and does not change while the graph is undergoing changes.

Lemma 5. *With probability at least $(1 - 1/n)$, the strong diameter of each cluster output by the algorithm is at most δ.*

Lemma 6. *For any $u \neq v$, probability that u and v are in different clusters is $O(\log n)d(uv)/\delta$.*

We refer the reader to the proofs in Gupta's lecture notes.

4 Deterministic region growing and duality

The first $O(\log k)$-approximation algorithm for Multicut was due to Garg, Vazirani and Yannakakis [GVY93] via a deterministic region growing argument. The basic argument is simple and shows up in several places (the use of the argument in bounding flow-cut gaps is credited to Leighton and Rao [LR99]). Our goal here is to make a connection to low-diameter network decomposition that we have defined probabilistically. One of the consequence of the probabilistic statement is the following. Suppose we are given a graph $G = (V, E)$, edge lengths $\ell : E \to \mathbb{R}_+$ and edge weights $w : E \to \mathbb{R}_+$. Then G can be partitioned into clusters of diameter at most δ such that the total weight of edges between the clusters is $O(\log n)\frac{1}{\delta}\sum_e w_e \ell_e$. To see this we simply apply

the probabilistic low diameter decomposition to the metric induced by the edge lengths and use linearity of expectation. Recall that this is how we proved the approximation bound for Multicut where we obtained a slightly refined bound of $O(\log k)$. The deterministic region growing proof for Multicut can be found in the standard approximation textbooks [Vaz13, WS11]. The proof is typically explained via a continuous time argument and can be somewhat mysterious at first glance. Moreover the proof is given only for Multicut and not for low-diameter decomposition though one can easily adapt it. For this reason we give a proof here while also giving some pedagogical perspective that may be helpful for some readers.

We will define the notion of volume. For a set of edges $A \subseteq V$ we let $\mathrm{vol(A)} = \sum_{\mathrm{e\in A}} \mathrm{w_e}\ell_\mathrm{e}$. For a set of vertices $S \subseteq V$ we overload notation and define $\mathrm{vol(S)} = \mathrm{vol(E(S) \cup \delta(S))}$ (that is, the volume counts all edges that have an end point in S). Fix a vertex v and let R be the radius from v, that is $R = \max_u d(v,u)$ where d is based on the shortest path distances induced by ℓ. When considering the s-t mincut rounding we argued that if we picked a random $r \in (0,R)$ then the $E[w(\delta(S_r)] \leq \mathrm{vol(V)}$ where S_r is the set of vertices in the ball of radius r around v; technically we had $R = 1$ but the same proof holds by scaling by R. Note that we are bounding the weight of the cut around S_r by the *total volume* of the graph.

Instead we want a lemma that bounds the weight of the cut only with respect to the volume of S_r. This *local guarantee* will allow the algorithm to be used repeatedly to cut the graph into many clusters while still being able to charge to the total volume but now with an approximation factor.

Theorem 7. *Let $G = (V,E)$ be a graph with edge lengths $\ell : E \to \mathbb{R}_+$ and edge weights $w : E \to \mathbb{R}_+$. Then G can be partitioned into clusters $C_1, C_2, \ldots, C_h$ of strong diameter at most δ such that $\sum_{i=1}^{h} \sum_{e \in \delta(C_i)} w_e \leq O(\log n) \frac{1}{\delta} \sum_{e \in E} w_e \ell_e$.*

Remark 8. *In the context of Multicut we were interested in $\delta = 1/2$ and $\Delta(G) = 1$. The $1/\delta$ term is necessary when considering smaller diameter clusters.*

Note that we are trying to remove as few edges as possible while making the diameter small. First, we will remove all edges e with $\ell_e > \frac{\delta}{c \log n}$ for some sufficiently large constant c and the total weight of these edges is clearly $O(\log n)\frac{1}{\delta}\sum_{e\in E} w_e\ell_e$. Thus, we can assume that $\ell_e < \frac{\delta}{c\log n}$. This is a simple preprocessing step to simplify the analysis. We will work with the remaining graph and overload notation and assume that it is G.

Fix a vertex v and consider $r_i = i \cdot \delta/(c \log n)$ for $i = 0, 1, 2 \ldots$, and let $S_i = B(v, r_i)$ be the set of vertices in the ball of radius r_i. We have $S_0 = \{v\}$ and $S_{i+1} \supset S_i$ for $i \geq 0$.

The claim below follows from the fact that edge lengths were assumed to be at most $\delta/(c \log n)$ and $r_{i+1} = r_i + \delta/(c \log n)$.

Claim 9. *If $e = uv \in \delta(S_i)$ then $u, v \in S_{i+1}$.*

For each i we define $\mathrm{vol'(S_i)} = \mathrm{vol(S_i)} + \mathrm{vol(V)/n}$. Note that this means that $\mathrm{vol'(S_0)} = \mathrm{vol(V)/n}$. This is a technical scaling trick.

Claim 10. *There is some $0 \leq i < 1 + \lceil \log n \rceil$ such $\mathrm{vol'(S_{i+1})} \leq 2\mathrm{vol'(S_i)}$.*

Proof. Suppose not. Then $\mathrm{vol'(S_j)} \geq 2^{\mathrm{j}}\mathrm{vol(S_0)}$ for $j = 1 + \lceil \log n \rceil$ but that would imply that $\mathrm{vol'(S_j)} > 2\mathrm{n} \cdot \mathrm{vol'(V_0)} = 2\mathrm{vol(V)}$ which is a contradiction. ■

Lemma 11. *There is a radius $r \in [r_i, r_{i+1})$ such that $w(\delta(S_r)) \leq O(\frac{\log n}{\delta})(\mathrm{vol(S_r)} + \mathrm{vol(V)/n})$.*

Proof. Pick r at random from $[r_i, r_{i+1})$ and consider S_r. The only edges that can be cut are those whose end points are in S_{i+1} and hence they are counted in $\mathrm{vol}'(\mathrm{S_{i+1}})$. Let e be such an edge. We have $\mathbf{P}[e \in \delta(S_r)] \leq \frac{\ell(e)}{r_{i+1}-r_i} \leq \frac{c \log n}{\delta}\ell(e)$. Thus the expected cost of $w(S_r)$ is at most

$$\frac{c \log n}{\delta}\mathrm{vol}'(\mathrm{S_{i+1}}) \leq 2\frac{\mathrm{c\,log\,n}}{\delta}\mathrm{vol}'(\mathrm{S_i}) \leq 2\frac{\mathrm{c\,log\,n}}{\delta}(\mathrm{vol(S_i)} + \mathrm{vol(V)/n}) \leq 2\frac{\mathrm{c\,log\,n}}{\delta}(\mathrm{vol(S_r)} + \mathrm{vol(V)/n}).$$

■

Remark 12. *The analysis applies to directed graphs where we obtain a guarantee about $w(\delta^+(S_r))$. Note that S_r has the property that every vertex is reachable from v within a distance of $r \leq \delta/2$, however, removing $\delta^+(S_r)$ may still leave incoming arcs.*

We can find the radius r deterministically by trying all possible values of r in $[r_i, r_{i+1})$ since there are at most n interesting values when S_r changes. Thus, we have shown that there is a radius $r \leq (1 + 2\log n)\delta/(c \log n)$ such that $w(\delta(B(v,r)) \leq 2\frac{c \log n}{\delta}(\mathrm{vol(B(v,r))} + \mathrm{vol(V)/n})$. Note that the radius of S_r is at most $\delta/2$ if we choose c large enough. Thus the strong diameter of S_r is at most δ. This is the first cluster. We remove $B(v,r)$ from G and repeat the process with a remaining vertex to find the next cluster. The process stops when there is no more vertices left in the graph. Let $v_1, v_2, \ldots, v_h$ be the centers of the vertices chosen in the clustering process and let $C_1, C_2, \ldots, C_h$ be the clusters found. The construction ensures that the strong diameter of each cluster is at most δ. For each C_i we have seen that $w(\delta(C_i)) \leq O(\frac{\log n}{\delta})(\mathrm{vol(C_i)} + \mathrm{vol(V)/n})$. Note that an edge e contributes to the volume of at most two clusters. Thus,

$$\sum_{i=1}^{h} w(\delta(C_i)) \leq O(\frac{\log n}{\delta})\mathrm{vol(V)}.$$

4.1 Using LP Duality to Obtain an LDD

We will now show how one can use Theorem 7 to obtain a probabilistic low diameter decomposition as we initially defined it. For this we rely on a useful trick that is used often but also not known as widely as it should be. Recall that an LDD is a distributions over partitions P such that clusters in P have diameter at most δ. Given a graph $G = (V, E)$ with edge lengths $\ell : E \to \mathbb{R}_+$ we define $\mathcal{P}$ to be the set of all partitions of V which are *valid* — that is, $P \in \mathcal{P}$ if each cluster of P has diameter at most δ. This is an exponentially large set and we are just defining it implicitly. Let E_P be the set of edges crossing the partition P (in other words cut by P). Any probability distribution $\mathcal{D}$ of $\mathcal{P}$ can be defined by numbers $y_P, P \in \mathcal{P}$ such that $\sum_P y_P = 1$ and $\bar{y} \geq 0$. The probability that an edge $e = uv$ is cut by the distribution y is $\sum_{P: e \in E_P} y_P$. Let this be α_e. We want to find the distribution $\alpha_e/\ell(e)$ since we are interested in the scaled probability. We write this as a large implicit LP problem.

$$\begin{aligned} & \min \alpha \\ & \text{s.t} \\ \sum_{P \in \mathcal{P}} y_P &= 1 \\ y_P &\geq 0 \quad P \in \mathcal{P} \\ \sum_{P: e \in E_P} y_P &\leq \alpha \ell(e) \quad e \in E \end{aligned}$$

We write the dual LP

$$\begin{aligned} & \max \beta \\ & \text{s.t} \\ \sum_{e \in E} \ell(e) x_e &= 1 \\ \beta &\geq \sum_{e \in E_P} x_e \quad P \in \mathcal{P} \\ x_e &\geq 0 \quad e \in E \end{aligned}$$

It may be a bit hard to interpret the dual at first glance but it is worth rewriting it as the following non-linear problem:

$$\beta = \max_{x \in \mathbb{R}^m_+} \min_{P \in \mathcal{P}} \frac{\sum_{e \in E_P} x_e}{\sum_{e \in E} \ell(e) x_e}$$

Exercise 1. *Verify that the two optimization problems are the same.*

We claim that the following upper bound on β.

Claim 13. $\beta = O(\frac{\log n}{\delta})$.

Proof. This follows from Suppose x^* is an optimum solution to the dual. Think of x^*_e as the weight w_e in the setting of Theorem 7. Then we know that there is a partition $P \in \mathcal{P}$ such that $\sum_{e \in E_P} x^*_e \leq O(\frac{\log n}{\delta}) \sum_{e \in E} x^*_e \ell(e)$. Hence $\beta = O(\frac{\log n}{\delta})$. ■

Thus we have obtained an existential proof that a good probabilistic LDD exists. But this seems not very useful since we want an efficient algorithm to sample from this distribution. It turns out that one can use an algorithmic version of Theorem 7 to obtain an efficient algorithm for creating and sampling the desired distribution. This is based on the multiplicative weight update (MWU) method that we will see later in the course.

5 LDDs and Probabilistic Embedding of Metrics into Dominating Tree Metrics

Using tree representations of graphs is a powerful tool in algorithm design. Here we will be interested in representing the distances in an undirected graph via distances in a tree. Let $G = (V, E)$ be a

graph with non-negative edge length $\ell : E \to \mathbb{R}_+$. This naturally defines a metric (V, d) via shortest path distances.

Can we approximate distances in G by a spanning tree $T = (V, E_T)$ of G? Clearly $d_T(u, v) \geq d_G(u, v)$ for every pair u, v where $d_T(u, v)$ is the distance in the tree and $d_G(u, v)$ is the distance in the graph. An extreme example is the cycle C on n vertices with unit edge lengths. A spanning tree corresponds to removing one edge (u, v) from C and in the resulting tree T, $d_T(u, v) = n - 1$ while $d_G(u, v) = 1$.

Exercise 2. *Prove that for every weighted graph G there is a spanning tree T such that $d_T(u, v) \leq (n-1)d_G(u, v)$.*

Alon et al [AKPW95], motivated by applications of spanning tree based metric approximations, noticed that if we are allowed to pick a probability distribution over spanning trees then the expected distance for any pair of vertices can be much better than the above worst-case example shows. For instance, in the cycle, if we pick a spanning tree at random from the n trees of the cycle then for any given pair (u, v) the *expected* distance is only 2! This kind of guarantee suffices for many applications. [AKPW95] showed that for any weighted graph $G = (V, E)$ there is a distribution $\mathcal{D}$ over spanning trees of G such that for any u, v,

$$E[d_T(u, v)] \leq \exp(\sqrt{\log n \log \log n}) \cdot d_G(u, v)$$

We think of this as probabilistic approximation of a graph metric by tree metrics. This can be viewed also as a metric *embedding* result. Note that the distances only increase so we are interested in the worst-case guarantee of how much the expected distance for any pair increases? In keeping with metric embedding terminology this quantity is often called *distortion* or *stretch*.

Alon et al proved a lower bound of $\Omega(\log n)$ on the distortion required for probabilistic tree approximation and conjectured that this is the tight bound. It turned out to be quite difficult to obtain even a poly-logarithmic bound. It was finally shown by Elkin et al. [EEST05], and currently the best known bound is $O(\log n \log \log n)$ [ABN08]. Removing the $\log \log n$ factor is an interesting and non-trivial open problem (or proving a stronger lower bound than $\Omega(\log n)$).

Bartal [Bar96] overcame the difficulty of spanning tree embeddings by considering metric embeddings. What does this mean? Instead of starting with the topology of the graph G we work with the metric completion (V, d) and view it as a complete graph on V. Any spanning tree of the complete graph is now allowed. Note that such a tree may not correspond to a spanning tree of the original graph G; for instance consider the cycle C and its metric completion. As Bartal noted, it can have additional vertices.

Definition 2. *A tree (V_T, E_T) with edge lengths $\ell_T : E \to \mathbb{R}_+$ is a* dominating *tree metric for a finite metric (V, d) if $V \subseteq V_T$ and for all $u, v \in V$, $d_T(u, v) \geq d(u, v)$ where d_T is the metric induced on V_T by the edge lengths of T.*

Bartal considered approximating metrics probabilistically by dominating tree metrics.

Definition 3. *Let (V, d) be a metric space. A probabilistic approximation of (V, d) by dominating tree metrics is a probability distribution $\mathcal{D}$ over a collection of trees $T_1, T_2, \ldots, T_h$ if each T_i is a dominating tree metric for (V, d). The distribution has stretch/distortion α if for all $u, v \in V$,*

$$E_{\mathcal{D}}[d_T(u, v)] \leq \alpha \cdot d(u, v).$$

Bartal developed LDDs and showed that one can use them to efficiently sample from a distribution that has stretch $O(\log^2 n)$ [Bar96]. He subsequently improved the stretch bound to $O(\log n \log\log n)$ [Bar98] and this was finally improved to the optimal $O(\log n)$ bound by Fakcharaenphol, Talwar and Rao [FRT03] using the CKR-RnadomPartition procedure. The construction of the trees from LDDs is quite simple in retrospect and we do the basic construction and then show how the subtle properties of the CKR procedure yields the optimal bound.

5.1 Low stretch trees via LDDs

Bartal's construction is based a recursive use of LDDs. The high-level idea is quite simple. Let (V, d) be a metric space with diameter Δ. We use a LDD with parameter $\delta = \Delta/2$ to randomly partition V into clusters $V_1, V_2, \ldots, V_h$ of diameter at most $\Delta/2$. We then recursively find a tree for each of the V_i, say with root r_i. We create a new dummy root r and connect each r_i to r with an edge of length Δ. The base case is when $|V| = 1$ when we simply return a tree with the root as the lone vertex. That's it! We can easily prove that this yields a stretch of $O(\log n \log \Delta)$ (here we assume that the minimum distance is at least 1) which depends on the aspect ratio but this can be easily altered to get stretch $O(\log^2 n)$. To be more formal, especially for the next section, we use the notion of levels and use $\delta = \Delta/2^i$ at level i to make the analysis cleaner.

TreeEmbed:$((V, d), D)$
  if $|V| = 1$ then
    return tree with single vertex $v \in V$ as root
  Create tree T with root r
  Use LDD with on (V, d) with $\delta = D/2$ to obtain clusters $V_1, V_2, \ldots, V_h$
  For $j = 1$ to h
    Let T_j be output of **TreeEmbed**$((V_1, d), D/2)$ with root r_j
    Connect T_j to T by adding edge (r, r_j) of length D
  return T with root r

We will assume that the minimum non-zero distance in the metric space is 1 by scaling. Let Δ be the diameter of the metric space with this assumption.

Theorem 14. *Let Δ be the diameter of (V, d). The algorithm* **TreeEmbed**$((V, d), \Delta))$ *outputs a random dominating tree metric $T = (V_T, E_T, \ell_T)$ such that for each $u, v \in V$, $E[d_T(u, v)] \leq O(\alpha \log \Delta) d(u, v)$ where α is the cutting probability of the LDD procedure.*

The proof is by induction. The base case is easy since the tree has a singleton and there is nothing to prove. It is easy to see that if we start with $D = \Delta$ then at depth i of the recursion the parameter is $\Delta/2^{i-1}$ and it is an upper bound on the diameter of the metric space in that recursive call.

We omit the proof of the following easy claim.

Claim 15. *The length of the root to leaf path of a tree created at level i of the recursion is at most $\sum_{j \geq i} \Delta/2^{j-1} \leq 2\Delta/2^{i-1}$.*

The following should be easy to see from the construction.

Claim 16. *The output of the algorithm is a dominating metric for (V, d).*

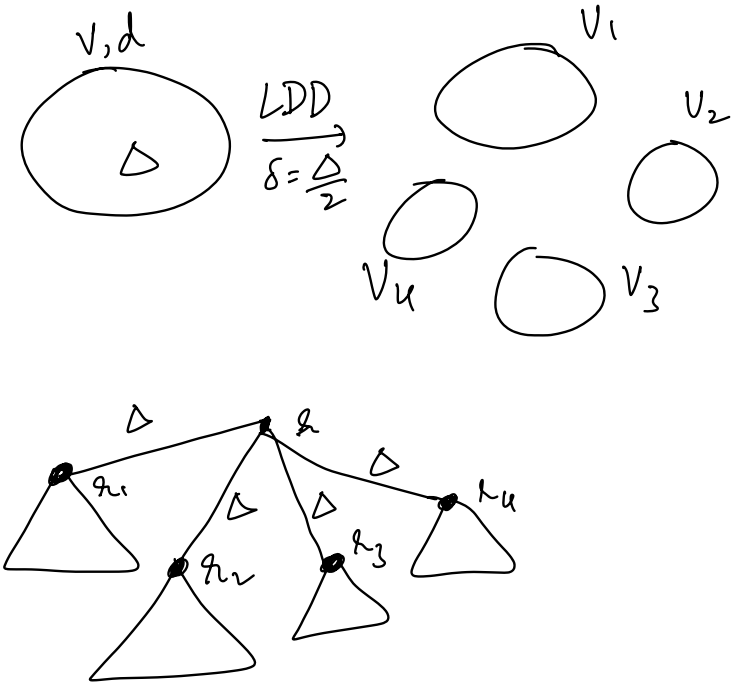

Figure 1: Illustration of recursive tree construction

Fix a pair of vertices u, v with distance $d(u, v)$.

Claim 17. *Suppose u and v are first separated at level i of the recursion. Then $d_T(u, v) \leq 4\Delta/2^{i-1}$.*

Proof. Follows from the previous claim. ■

The pair u, v are separated in the first level of the recursion due to the LDD with probability $\alpha d(u, v)/(\Delta/2) \leq 2\alpha d(u, v)/\Delta$ in which case their distance in the tree is at most 4Δ. Otherwise they are in the same cluster and we can apply induction. Note that u and v are definitely separated by level t where t is the smallest integer such that $\Delta/2^{t+1} < d(u, v)$. The depth of the recursion is at most $1 + \lceil \log \Delta \rceil \leq 2 \log \Delta$.

It is easy to unroll the induction and use the preceding claim to obtain:

$$E[d_T(u, v)] \leq \sum_{i=0}^{t+1} 2\alpha \frac{d(u, v)}{(\Delta/2^{i-1})} \cdot (4\frac{\Delta}{2^{i-1}}) \leq O(\alpha \log \Delta) d(u, v)$$

since the depth of the recursion is $O(\log \Delta)$.

Remark 18. *The tree may require depth $\log \Delta$ to provide a good approximation. Note that in general $\log \Delta$ can be as large as n and moreover one cannot expect the depth of the tree (in terms of number of edges) to be less than $\log \Delta$ if one wants reasonable distortion. Consider the metric induced by a path with n edges and edge lengths 2^i, $i = 1$ to n. In such cases the dependence of*

the stretch on $\log \Delta$ *is undesirable. One can alter the algorithm slightly to make the stretch bound* $O(\log^2 n)$ *as follows. In applying the LDD with parameter* δ*, we ensure that any pair* u, v *such that* $d(u,v) \leq \delta/n^2$ *is not cut during the procedure. We can do this by contracting all such pairs without changing the diameter of the resulting metric space too much. This will ensure that in the tree construction process a pair* u, v *participates in only* $O(\log n)$ *levels, and hence the expected stretch can be bounded by* $O(\alpha \log n)$*.*

Hierarchically well-separated trees (HSTs): The reader may have noticed that the trees constructed by the algorithm have an additional strong property: the edge lengths at each level are the same and the length from the root to the leaf go down by a factor of 2 at each level (we can adjust this factor which will of course impact the stretch factor correspondingly). A tree metric with such a property is called hierarchically well-separated tree metric and this additional property can be exploited in algorithms and comes for free in the construction.

5.2 An optimal construction

[FRT03] showed that if one used the LDD based on the CKR-RandomPartition in the TreeEmbed algorithm then the expected stretch is $O(\log n)$ which is optimal! We recall the guarantee of the LDD which is refined one.

Lemma 19. *The probability that* u *and* v *are in different clusters is at most* $\frac{4d(u,v)}{\delta} \log(\frac{|B(u,\delta)|}{|B(u,\delta/8)|})$.

Thus, the LDD guarantee is no longer uniform but depends on the scale δ. We plug this refined bound into the previous expression for the stretch.

$$\begin{aligned} E[d_T(u,v)] &\leq \sum_{i=0}^{t+1} 2\log(\frac{|B(u,\Delta/2^i)|}{|B(u,\Delta/2^{i+3})|})\frac{d(u,v)}{(\Delta/2^{i-1})} \cdot (4\frac{\Delta}{2^{i-1}}) \\ &\leq 8d(u,v)(\log|B(u,\Delta)| + \log|B(u,\Delta/2)| + \log|B(u,\Delta/4)|) \\ &\leq O(\log n)d(u,v). \end{aligned}$$

Lower bounds: How can we prove that the $O(\log n)$ bound for LDDs and tree embeddings are near optimal (modulo precise constant factors)? One can use existence of low-girth graphs which are closely tied to expanders in a direct fashion — see Gupta's notes. Another way is via indirection. In the previous lecture we saw that the integrality gap of the LP relaxation for Multicut is $\Omega(\log k)$. Note that we prove an upper bound of $O(\log k)$ essentially via an LDD. In fact if α is the factor for LDD then we get an $O(\alpha)$ approximation for Mulitcut via the LP. Thus, one sees that $\alpha = \Omega(\log n)$ for general metrics (in the integrality gap example for Multicut $k = \Omega(n^2)$). One can use similar approach to prove lower bound for tree embeddings.

Fast Algorithms for LDDs and Tree Embeddings: We focused on the quality of LDDs and tree embeddings but not so much on the running times. It is easy to see that the algorithms themselves can be implemented in polynomial-time. The main computation is about shortest paths and if one computes all-pairs-shortest paths (APSP) then the algorithms are pretty simple.

However APSP is slow; takes $O(mn)$ time. It is possible compute LDDs and metric tree embeddings in close to linear time on a weighted graph with m edges. This involves computing approximate shortest paths and several tricks and sometimes we give up on the quality of the approximation by logarithmic factors. See [BGS16] for an $O(m \log n)$ time algorithm for finding an $O(\log n)$ stretch (in expectation) tree.

References

[ABN08] Ittai Abraham, Yair Bartal, and Ofer Neiman. Nearly tight low stretch spanning trees. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 781–790. IEEE, 2008.

[AKPW95] Noga Alon, Richard M Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, 1995.

[Bar96] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 184–193. IEEE, 1996.

[Bar98] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 161–168, 1998.

[BGS16] Guy E Blelloch, Yan Gu, and Yihan Sun. Efficient construction of probabilistic tree embeddings. *arXiv preprint arXiv:1605.04651*, 2016.

[EEST05] Michael Elkin, Yuval Emek, Daniel A Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 494–503, 2005.

[Fil24] Arnold Filtser. On sparse covers of minor free graphs, low dimensional metric embeddings, and other applications, 2024.

[FRT03] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455, 2003.

[GVY93] N. Garg, V.V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi) cut theorems and their applications. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 698–707. ACM New York, NY, USA, 1993.

[KPR93] Philip Klein, Serge A Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 682–690, 1993.

[LR99] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6):787–832, 1999. Conference version is from 1988.

[Vaz13] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[WS11] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.