In this lecture we will study the problem of finding low-stretch spanning trees in general graphs. A low-stretch spanning tree T in a graph G is a spanning tree for G with the additional constraint that the distance between any two vertices in T is at most a small constant factor times the distance between the two same vertices in G. In other words, the tree does not "stretch" distances too much.

Throughout this lecture, we will be considering a graph $G = (V, E)$ with edge lengths $(l_e)_{e \in E}$. As is convention, we let $n = |V|$ and $m = |E|$ unless otherwise noted. We define $d_G : V \times V \to \mathbb{R}^+$ to be the distance function in G. That is, for all u, v in V, $d_G(u, v)$ is the length of the shortest path in G from u to v.

1 Shortest Path Trees

At best, we could hope to find exact distance preserving trees, i.e.:

Definition 5.1. Let T be a spanning tree of G. We call T an all-pairs shortest path tree in G if for all u, v in V, $d_T(u, v) = d_G(u, v)$.

For any single source u in V, it is easy to compute a single-source shortest path tree T such that for all v in V, $d_T(u, v) = d_G(u, v)$. This can be obtained by running Dijkstra's algorithm or Bellman-Ford as applicable.

Unfortunately, an all-pairs shortest path tree usually does not exist. Consider the clique of n nodes, K_n, with unit edge lengths. Any spanning tree T for K_n will be missing most of the edges, and thus there must be nodes u, v in V such that $d_T(u, v) \geq 2$ even though $d_G(u, v) = 1$.

2 Low-Stretch Spanning Trees

This leads us to consider a relaxed definition. Say we do not require distances in T to be equal to those in G, but to be at most multiplied by a constant factor α.

Definition 5.2. Let T be a spanning tree of G, and let $\alpha \geq 1$. We call T a (deterministic) α-stretch spanning tree of G if

$$d_G(u, v) \leq d_T(u, v) \leq \alpha \, d_G(u, v).$$

holds for all $u, v \in V$.

Naturally, we can wonder if there exists a "small" value for α such that for any graph G we can always find an α-stretch spanning tree of G. The answer to that question is also unfortunately no.

For any n, consider the cycle of n nodes, C_n. Any spanning tree T in C_n is a path of $n - 1$ edges, built by simply removing one edge $\{u, v\}$ from C_n. The distance between the two endpoints of that removed edge in T is $d_T(u, v) = n - 1 = (n - 1)d_G(u, v)$. Therefore, for all α, there exist graphs containing no α-stretch spanning tree.

3 Motivation

So far, it seems as though we cannot give a good solution to this problem in general. But why is this problem even interesting in the first place? The motivation is that some problems are much

easier to solve on trees than on general graphs, and using low-stretch spanning trees enables us to give approximate solutions in a reasonable amount of time to such problems.

One such example is the k-median problem. Given $k \geq 1$, we want to find a subset $C \subseteq V$ of k vertices that minimizes $\sum_{v \in V} d_G(v, C)$.

If G is a tree, then, through dynamic programming, we can compute an exact solution in time polynomial in n and k. On the other hand, if G is a general graph, then the problem is NP-hard. However, using an α-stretch spanning tree of G, we can compute an α-approximate solution for G. The algorithm is simple:

1. Find an α-stretch spanning tree T of G.
2. Solve the k-median problem on T to get C_T.
3. Return C_T.

Claim 5.3. *C_T is an α-approximate solution to the k-median problem for G.*

Proof. Suppose that C_G is an optimal solution for G. Then the following inequalites hold:

$$\begin{aligned} \sum_{v \in V} d_T(v, C_G) &\leq \alpha \sum_{v \in V} d_G(v, C_G), \qquad (5.1) \\ \sum_{v \in V} d_T(v, C_T) &\leq \sum_{v \in V} d_T(v, C_G), \\ \sum_{v \in V} d_G(v, C_T) &\leq \sum_{v \in V} d_T(v, C_T). \end{aligned}$$

Therefore we have

$$\sum_{v \in V} d_G(v, C_T) \leq \alpha \sum_{v \in V} d_G(v, C_G). \qquad \square$$

Low-stretch spanning trees can also be used to more efficiently solve linear systems of the form $A\vec{x} = \vec{b}$ where A is the Laplacian matrix of some graph G. To do this, we let P be the Laplacian matrix of a low-stretch spanning tree of G, and then we solve the system $P^{-1}A\vec{x} = P^{-1}\vec{x}$ instead. This is called *preconditioning* with P. It turns out that this preconditioning allows certain algorithms for solving linear systems to converge faster to a solution. This technique also lets us solve linear systems where A is a symmetric, diagonally dominant matrix more efficiently.

4 Randomization to the rescue

Now that we know that this problem is an interesting one, how do we approach it without running into the issues outlined above? If we cannot get a small deterministic value for α, let us try to get a small value for α *in expectation*.

First, we will need to amend our definition to accomodate the fact that we are no longer looking for trees in a deterministic fashion.

Definition 5.4. A (randomized) low-stretch spanning tree of stretch α for a graph $G = (V, E)$ is specified as a probability distribution $\mathcal{D}$ over spanning trees of G such that

$$\begin{aligned} d_G(u, v) &\leq d_T(u, v) \qquad \text{for all } T \text{ in the support of } \mathcal{D}, \\ \mathop{\mathbf{E}}_{T \sim \mathcal{D}}[d_T(u, v)] &\leq \alpha \, d_G(u, v) \qquad (5.2) \end{aligned}$$

for all $u, v \in V$.

This definition is interesting because a randomized low-stretch spanning tree preserves distances "on average". For example, if one samples $k = c \log n$ trees $T_1, \ldots, T_k$ from such a distribution for some well-chosen c, then for all u, v in V, with high probability, there exists $i \in \{1, \ldots, k\}$ such that $d_{T_i}(u, v) \leq \alpha \, d_G(u, v)$.

Exercise 5.5. Prove that if inequality (5.2) holds only for all $\{u, v\} \in E$, then the inequality also holds for all $(u, v) \in V \times V$. (Hint: use the triangle inequality.)

Remark 5.6. With this definition, we can get a small α for C_n for all n. Let $\mathcal{D}$ be the uniform distribution over spanning trees of C_n. Picking a tree from $\mathcal{D}$ is equivalent to picking an edge uniformly at random from C_n and deleting it. For all $\{u, v\} \in E$, there is only a 1 in n chance of deleting the edge from u to v. Thus we now have

$$\begin{aligned} \mathop{\mathbf{E}}_{T \sim \mathcal{D}}[d_T(u, v)] &= \frac{n-1}{n} \cdot 1 + \frac{1}{n}(n-1) \\ &= 2\,\frac{n-1}{n} < 2. \end{aligned}$$

Then by exercise 5.5, $\mathcal{D}$ produces spanning trees of stretch 2 for C_n.

Remark 5.7. The previous approximation algorithm for k-median still works, only now it is a randomized algorithm, and the bound only holds in expectation. Inequality (5.1) becomes

$$\mathop{\mathbf{E}}_{T \sim \mathcal{D}} \left[\sum_{v \in V} d_T(v, C_G) \right] \leq \alpha \sum_{v \in V} d_G(v, C_G),$$

and thus we obtain

$$\mathop{\mathbf{E}}_{T \sim \mathcal{D}} \left[\sum_{v \in V} d_G(v, C_T) \right] \leq \alpha \sum_{v \in V} d_G(v, C_G).$$

We can now prove interesting results using this notion of embeddings into trees.

Theorem 5.8. *[AN12] For any graph G, there exists a spanning tree distribution $\mathcal{D}_{AN}$ with stretch factor $\alpha_{AN} = O(\log n \log \log n)$ and such that we can sample trees from $\mathcal{D}_N$ in $O(m \log n \log \log n)$ time.*

Theorem 5.9. *[AKPW95] For infinitely many n, there exist graphs G on n vertices such that any α-stretch spanning tree distribution $\mathcal{D}$ on G must have $\alpha = \Omega(\log n)$. In fact, G can be taken to be the n-vertex square grid or the n-vertex hypercube.*

In this lecture, we will restrict ourselves to metric graphs and prove a looser upper bound. The restriction to metric graphs is for simplicity; the following results can be extended to graphs in general.

Definition 5.10. A metric graph G is a complete graph such that edge lengths satisfy the triangle inequality, i.e. for all u, v, w in V, we have $l_{(u,v)} \leq l_{(u,w)} + l_{(w,v)}$.

We will give a proof of the following theorem.

Theorem 5.11. *[Bar96] For any metric graph G, there exists an efficiently samplable α_B-stretch spanning tree distribution $\mathcal{D}_B$ such that $\alpha_B = O(\log n \log \Delta)$, where*

$$\Delta = \frac{\max\{d_G(u, v) \mid u, v \in V\}}{\min\{d_G(u, v) \mid u, v \in V,\ u \neq v\}} = \frac{d_{\max}(G)}{d_{\min}(G)}$$

is called the aspect ratio of G.

Before proving the theorem, we need to define an additional notion of graph decomposition.

Definition 5.12. Given a metric graph $G = (V, E)$ and parameters $D > 0$ and $\beta > 0$, a low-diameter decomposition scheme (or LDD scheme[1]) is a randomized algorithm that partitions V into $V_1, \ldots, V_t$ such that

- for all $i \in \{1, \ldots, t\}$ and for all u, v in V_i, we have $d_G(u, v) \leq D$.
- for all $u, v \in V$ such that $u \neq v$, we have $\mathbf{Pr}[u, v \text{ in different clusters}] \leq \frac{d_G(u,v)}{D}\beta$.

We will assume the following lemma and prove it later.

Lemma 5.13. *For any $D > 0$, there exists an LDD scheme with $\beta = O(\log n)$.*

Using this lemma, we can define the following algorithm to sample from Bartal's α_B-spanning tree distribution:

Algorithm: Bartal(G, i): ▷ Assumption: $d_{max}(G) \leq 2^i$

- As a base case, return G if G is a single vertex.
- Use lemma 5.13 with $D = 2^{i-1}$ to get a vertex partition $V_1, \ldots, V_t$ with induced subgraphs $G_1, \ldots, G_t$.
- For every j in $\{1, \ldots, t\}$, recursively apply the algorithm: $T_j \leftarrow \text{Bartal}(G_j, i-1)$.
- Add edges of length 2^i from the root r_1 of T_1 to the roots of $T_2, \ldots, T_t$.
- Return the resulting tree rooted at r_1.

Now we can prove theorem 5.11.

Proof of theorem 5.11. By scaling the edge lengths appropriately, we may assume that $d_{min}(G) = 1$ and $d_{max}(G) = \Delta$ without loss of generality.

We define the distribution $\mathcal{D}_B$ by sampling $T \sim \mathcal{D}_B$ via $T = \text{Bartal}(G, \lfloor \log \Delta \rfloor + 1)$. We want to show $\alpha = O(\log n \log \Delta)$.

Claim: For a graph $G' = (V', E')$ and $i \in \mathbb{N}$, let $T' = \text{Bartal}(G', i)$. Then $\mathbf{E}[d_{T'}(u, v)] \leq 8i\beta d_{G'}(u, v)$ for all u, v in V'.

To prove the claim, we proceed by induction on i. Let $T_1, \ldots, T_t$ be the result of running the algorithm recursively on $G_1, \ldots, G_t$, a β-LDD of G'. Let u, v be two vertices in V', let a, b be indices such that u is in T_a and v is in T_b, and let r_a, r_b be the roots of T_a, T_b. Then we have

$$\mathbf{E}[d_{T'}(u, v)] = \mathbf{E}[d_{T'}(u, v) \mid a \neq b]\ \mathbf{Pr}[a \neq b] + \mathbf{E}[d_{T'}(u, v) \mid a = b]\ \mathbf{Pr}[a = b].$$

We also know

$$\begin{aligned}
\mathbf{E}[d_{T'}(u, v) \mid a \neq b] &= \mathbf{E}[d_{T_a}(u, r_a) + d_{T'}(r_a, r_1) + d_{T'}(r_1, r_b) + d_{T_b}(r_b, v)], \\
\mathbf{Pr}[a \neq b] &\leq \frac{\beta\, d_{G'}(u, v)}{2^{i-1}}, \\
\mathbf{E}[d_{T'}(u, v) \mid a = b] &\leq 8(i-1)\beta d_{G'}(u, v) \qquad \text{(by induction hypothesis)}, \\
\mathbf{Pr}[a = b] &\leq 1.
\end{aligned}$$

[1] This is not standard notation.

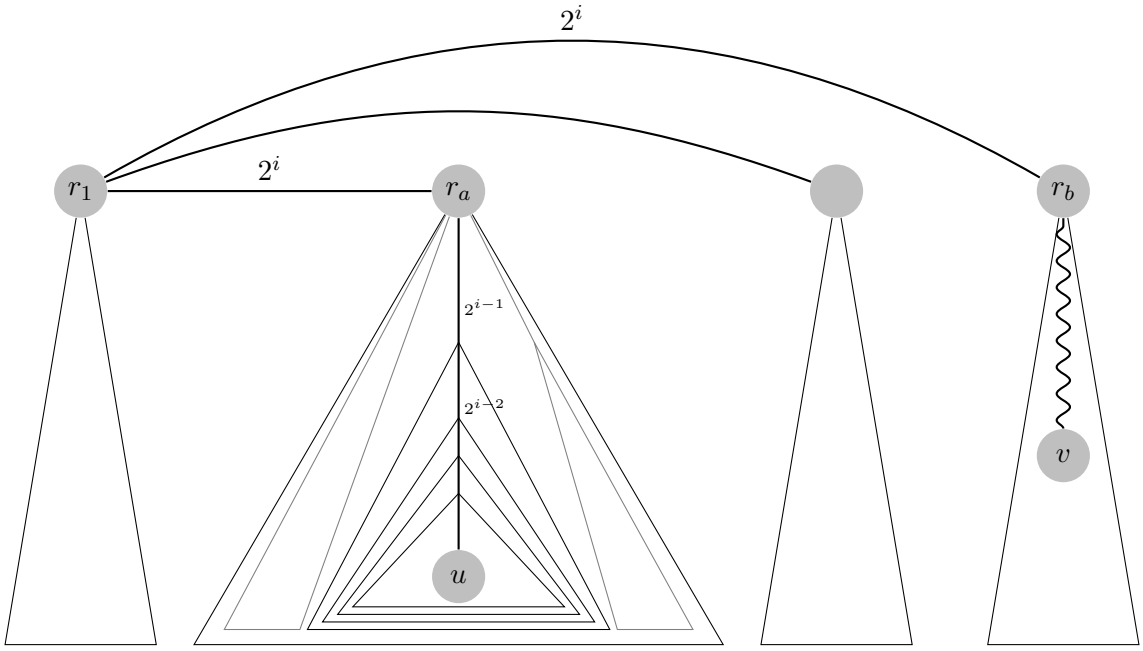

Figure 5.1: Diagram of the decomposition formed by Bartal's algorithm

Recursively, since the path from u to r_a is made of edges from roots to roots,

$$d_{T_a}(u, r_a) \leq 1 + 2 + 4 + \cdots + 2^{i-1} < 2^i.$$

Identically, $d_{T_b}(r_b, v) < 2^i$. Furthermore, we know that $d_{T'}(r_a, r_1) \leq 2^i$ and $d_{T'}(r_1, r_b) \leq 2^i$. This gives

$$\mathbf{E}[d_{T'}(u, v) \mid a \neq b] < 2^i + 2^i + 2^i + 2^i = 2^{i+2},$$

which finally gives us

$$\begin{aligned}
\mathbf{E}[d_{T'}(u, v)] &< 2^{i+2} \frac{\beta\, d_{G'}(u, v)}{2^{i-1}} + 8(i-1)\beta\, d_{G'}(u, v) \\
&= 8(1 + i - 1)\beta\, d_{G'}(u, v) \\
&= 8i\, \beta\, d_{G'}(u, v).
\end{aligned}$$

This proves the claim. Then we set $G' = G$ and $i = \lfloor \log \Delta \rfloor + 1$ to get that $\alpha_B = O(i\beta) = O(\log \Delta \log n)$. □

Finally, we give a LDD scheme that proves lemma 5.13.

Algorithm: LDD(G, D):

- Pick any unmarked vertex v.
- Sample R_v from the geometric distribution $Geom(p = \min(1, \frac{4 \log n}{D}))$.
- Mark all unmarked vertices w such that $d_G(v, w) \leq R_v$ as belonging to v's cluster G_v.
- If there exists an unmarked vertex, repeat.

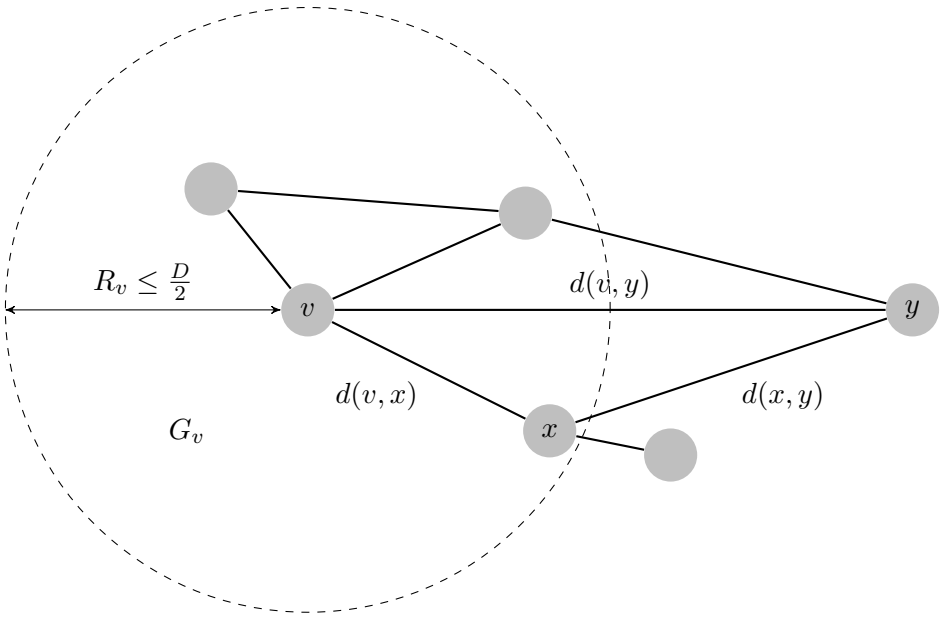

Figure 5.2: A cluster forming around v in the LDD process, separating x and y. The graph is complete, but many edges are omitted in this diagram to reduce clutter.

Proof of lemma 5.13. First, we check that each cluster's diameter will be at most D with high probability. It suffices to check that $R_v \le D/2$ for each cluster G_v. This is because G is a metric graph. Thus, if $R_v \le D/2$, then for any $x, y \in G_v$, we invoke the triangle inequality to get

$$d_G(x,y) \le d_G(x,v) + d_G(v,y) \le D/2 + D/2 = D.$$

The probability that $R_v > D/2$ for one particular cluster is

$$\mathbf{Pr}[R_v > D/2] = (1-p)^{D/2} \le e^{-pD/2} \le e^{-2\log n} = \frac{1}{n^2}.$$

Therefore, by union bound, the probability that all clusters G_v have $R_v \le D/2$ is

$$\begin{aligned}\mathbf{Pr}[\forall v \in V,\ R_v \le D/2] &= 1 - \mathbf{Pr}[\exists v \in V,\ R_v > D/2] \\ &\ge 1 - \frac{n}{n^2} \\ &= 1 - \frac{1}{n}.\end{aligned}$$

This proves that each cluster's diameter will be small with high probability. Next, we must show that for some $\beta = O(\log n)$, the inequality $\mathbf{Pr}[x, y \text{ in different clusters}] \le \frac{\beta\, d_G(u,v)}{D}$ holds for all distinct vertices $x, y \in V$.

Sampling from the geometric distribution is like repeatedly flipping a coin and counting the number of flips we get before the first heads. This process is memoryless, meaning that if we have already performed N flips, the probability that we will get a heads is still p.

Let x and y be distinct vertices, and consider the first time at which one of these vertices is inside the current ball centered at, say, vertex v. Without loss of generality, let the vertex inside the current ball be x. At this point, we have performed $d_G(v,x)$ flips. The probability that we separate

x and y is then the probability that we get a heads within $d_G(v, y)$ flips total, i.e. within the next $d_G(v, y) - d_G(v, x)$ flips. Then we can use union bound, and since G is a metric graph, we can use the triangle inequality as well to get

$$\mathbf{Pr}[x, y \text{ in different clusters}] \leq (d_G(v, y) - d_G(v, x))\, p \leq d_G(x, y)\, p \leq \frac{4 \log n\, d_G(x, y)}{D}.$$

Therefore, this LDD scheme gives us $\beta = O(\log n)$. □

5 Concluding Remarks

There is a natural correspondence between metric graphs and finite metric spaces. Thus, in this lecture, we saw a way to probabilistically approximate a finite metric space with a simpler metric space over a tree. This idea of approximating metric spaces has been extensively studied in various forms.

For example, the Johnson–Lindenstrauss lemma, which we will see in a future lecture, says that if we have n points in finite-dimensional Euclidean space, we can embed the points in $\mathbb{R}^{O(\log n/\epsilon^2)}$ such that distances between points are disorted by a factor of at most $1 \pm \epsilon$ [JL84].

Another result by Matoušek shows that a finite metric space on n points can be embedded into ℓ_p-space with $O((\log n)/p)$ distortion [Mat97].

References

[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. 5.9

[AN12] Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 395–406, New York, NY, USA, 2012. ACM. 5.8

[Bar96] Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 184–193. IEEE, 1996. 5.11

[JL84] William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984. 5

[Mat97] Jiří Matoušek. On embedding expanders into ℓ_p spaces. *Israel Journal of Mathematics*, 102(1):189–197, 1997. 5