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 $d_G : V \times V \rightarrow \mathbb{R}^+$ 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 $v$. Similarly, we define $d_T : V \times V \rightarrow \mathbb{R}^+$ to be the distance with respect to $T$.

### 1 Motivation

The study of low stretch spanning trees is guided by two high level hopes:

**Hope (1)**. Graphs have spanning trees that preserve their metrics, i.e.,

$$d_G(u,v) \approx d_T(u,v) \quad \text{for all } u,v \in V.$$

**Hope (2)**. Many NP-hard problems are much easier to solve on trees.

Supposing these are true, we have a natural recipe for designing algorithms to solve problems relating to metrics: (1) find a spanning tree preserving the graph metric, (2) solve the problem on the spanning tree, (3) return the solution and hope it is a good solution for the original graph.

### 2 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 $d_G(u,v) = 1$.

### 3 A First Relaxation: Low-Stretch Spanning Trees

To remedy the snag above, let us not require distances in $T$ be equal to those in $G$, but instead be within a constant multiplicative factor $\alpha$ of those in $G$.

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

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

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

Let us try our meta-algorithm out on 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 $\alpha$-stretch spanning tree of $G$, we can compute an $\alpha$-approximate solution for $G$. The algorithm is simple:

1. Find an $\alpha$-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 $\alpha$-approximate solution to the $k$-median problem for $G$.

**Proof.** Suppose that $C_G$ is an optimal solution for $G$. Then we have

$$\sum_{v \in V} d_T(v, C_G) \leq \alpha \sum_{v \in V} d_G(v, C_G),$$

because distances in the tree are stretched by only a factor of $\alpha$. Next,

$$\sum_{v \in V} d_T(v, C_T) \leq \sum_{v \in V} d_T(v, C_G),$$

because $C_G$ is some solution, and $C_T$ is the optimal solution on the tree $T$. Finally,

$$\sum_{v \in V} d_G(v, C_T) \leq \sum_{v \in V} d_T(v, C_T),$$

because distances in the graph are at most those in the tree. Therefore we have

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

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 $P^{-1} A \vec{x} = P^{-1} \vec{b}$ 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.

Naturally, we should wonder if there exists a “small” value for $\alpha$ such that for any graph $G$ we can always find an $\alpha$-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 $\alpha < (n - 1)$, there exist graphs containing no $\alpha$-stretch spanning tree.
4 A Second Relaxation: Randomization to the rescue

How do we avoid this problem? If we cannot get trees with small stretch deterministically, let us try to get trees with small stretch “on average”. We amend our definition as follows:

**Definition 5.4.** A (randomized) low-stretch spanning tree of stretch \( \alpha \) for a graph \( G = (V, E) \) is a probability distribution \( D \) over spanning trees of \( G \) such that for all \( u, v \in V \), we have

\[
d_G(u, v) \leq d_T(u, v) \quad \text{for all } T \text{ in the support of } D, \quad \text{and} \quad \mathbb{E}_{T \sim D}[d_T(u, v)] \leq \alpha d_G(u, v)
\]

(5.2)

Observe that the first property is with probability 1 (i.e., holds for all trees in the support of the distribution), whereas the second property holds only on average. While this is weaker than what we hoped for, the definition is still interesting. For example:

**Remark 5.5.** 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

\[
\mathbb{E}_{T \sim D} \left[ \sum_{v \in V} d_T(v, C) \right] \leq \alpha \sum_{v \in V} d_G(v, C)
\]

and thus we obtain

\[
\mathbb{E}_{T \sim D} \left[ \sum_{v \in V} d_G(v, C_T) \right] \leq \alpha \sum_{v \in V} d_G(v, C)
\]

In general, 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 \( d_{T_i}(u, v) \leq \alpha d_G(u, v) \). To see this in action consider the next example.

**Remark 5.6.** Given a randomized \( \alpha \)-stretch tree distribution on \( G \), one can compute APSP up to a factor of \( 2\alpha \) in \( \tilde{O}(n^2) \) time “w.h.p.” (with probability \( 1 - n^{-5} \)).

The algorithm is again simple. Sample \( 10 \log n \) trees from the distribution, and for each pair \( u, v \in V \), set \( d(u, v) \) to be the smallest distance found over all iterations. It follows immediately from the definition of low stretch spanning trees that \( d(u, v) \geq d_G(u, v) \). To see that \( d(u, v) \leq 2\alpha d_G(u, v) \), observe that by Markov’s inequality:

\[
\Pr[d_T(u, v) \geq 2\mathbb{E}[d_T(u, v)]] = \Pr[d_T(u, v) \geq 2\alpha d_G(u, v)] \leq 1/2
\]

The probability that this relation holds for all sampled trees is less than \( (1/2)^{10 \log n} \leq n^{-10} \). By a union bound over all \( O(n^2) \) choices for \( u, v \), we have that for all \( u, v \in V \), \( d(u, v) \) is within a \( 2\alpha \) factor of \( d_G(u, v) \) with probability at least \( 1 - n^{-8} \).

**Exercise 5.7.** 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.)

5 Low-Stretch Spanning Tree Construction

Of course, now comes the important question: given a graph \( G \) can we always find a randomized low-stretch spanning tree for \( G \) with a reasonable value of \( \alpha \)? As a sanity check, note the following.
Remark 5.8. We can get a small $\alpha$ for $C_n$ for all $n$. Let $D$ be the uniform distribution over spanning trees of $C_n$. Picking a tree from $D$ 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

$$E_{T \sim D}[d_T(u, v)] = \frac{n - 1}{n} \cdot 1 + \frac{1}{n} (n - 1) = 2 \frac{n - 1}{n} < 2.$$ 

Then by exercise 5.7, $D$ produces spanning trees of stretch 2 for $C_n$.

Finally, we can answer this question affirmatively.

**Theorem 5.9 ([AN12]).** For any graph $G$, there exists a distribution $D_{AN}$ over spanning trees of $G$ with stretch factor $\alpha_{AN} = O(\log n \log \log n)$ and such that we can sample trees from $D_N$ in $O(m \log n \log \log n)$ time.

Moreover, the stretch bound $\alpha_{AN}$ is almost optimal, up to the $O(\log \log n)$ factor, as the following lower bound shows.

**Theorem 5.10 ([AKPW95]).** For infinitely many $n$, there exist graphs $G$ on $n$ vertices such that any $\alpha$-stretch spanning tree distribution $D$ 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; see, e.g., the proof of Theorem 5.9 which extends the results to general graphs.

**Definition 5.11.** 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 $l_{(u,v)} \leq l_{(u,w)} + l_{(w,v)}$.

### 5.1 Bartal’s Construction

In this section, we prove the following theorem.

**Theorem 5.12 ([Bar96]).** For any metric graph $G$, there exists an efficiently samplable $\alpha_B$-stretch spanning tree distribution $D_B$ 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 $G$.

In the case that $d_{\min} \geq 1$, and $d_{\max} \leq \text{poly}(n)$, this guarantees an approximation ratio of $O(\log^2 n)$. To prove the theorem, we need to define an additional notion of graph decomposition.

**Definition 5.13.** Given a metric graph $G = (V, E)$ and parameters $D > 0$ and $\beta > 0$, a low-diameter decomposition scheme (or LDD scheme) 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 $\Pr[u, v \text{ in different clusters}] \leq \frac{d_G(u, v)}{D} \beta$. 

4
We will assume the following lemma and prove it later.

**Lemma 5.14.** For any $D > 0$, there exists an LDD scheme with $\beta = O(\log n)$.

Using this lemma, we can define the following algorithm to construct random low-stretch spanning trees:

**Algorithm 1:** Bartal’s Low-Stretch Tree Algorithm

**Input:** Graph $G = (V, E)$ and parameter $i$, such that $d_{\text{max}}(G) \leq 2^i$

**Output:** Random low-stretch spanning tree for $G$

1. As a base case, return $G$ if $G$ is a single vertex;
2. Use Lemma 5.14 with $D = 2^{i+1}$ to get partition $V_1, \ldots, V_t$ with induced subgraphs $G_1, \ldots, G_t$;
3. for $j$ in $\{1, \ldots, t\}$ do
   4. $T_j \leftarrow \text{Bartal}(G_j, i - 1)$, with $r_j$ being the root of tree $T_j$;
5. end
6. Add edges of length $2^i$ from the root $r_1$ of $T_1$ to the roots of $T_2, \ldots, T_t$;
7. Return the resulting tree rooted at $r_1$.

Now we can prove theorem 5.12.

**Proof of Theorem 5.12.** By scaling the edge lengths appropriately, we may assume that $d_{\text{min}}(G) = 1$ and $d_{\text{max}}(G) = \Delta$ without loss of generality. We define the distribution $\mathcal{D}_B$ by sampling $T \sim \mathcal{D}_B$ via $T \leftarrow \text{Bartal}(G, \lfloor \log \Delta \rfloor + 1)$. We want to show $\alpha = O(\log n \log \Delta)$.

**Claim 5.15.** For a graph $G' = (V', E')$ and $i \in \mathbb{N}$, let $T' \leftarrow \text{Bartal}(G', i)$. Then $\mathbb{E}[d_{T'}(u, v)] \leq 8i\beta d_{G'}(u, v)$ for all $u, v$ in $V'$.

**Proof.** To prove the claim, we proceed by induction on $i$. The reader can use Figure ?? as an illustration. Let $T_1, \ldots, T_t$ be the result of running the algorithm recursively on $G_1, \ldots, G_t$, a $\beta$-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

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

We also know

$$\mathbb{E}[d_{T'}(u, v) \mid a \neq b] = \mathbb{E}[d_{T_a}(u, r_a) + d_{T'}(r_a, r_1) + d_{T'}(r_1, r_b) + d_{T_b}(r_b, v)],$$

$$\mathbb{P}[a \neq b] \leq \frac{\beta d_{G'}(u, v)}{2^i},$$

$$\mathbb{E}[d_{T'}(u, v) \mid a = b] \leq 8(i - 1)\beta d_{G'}(u, v) \quad \text{(by induction hypothesis)},$$

$$\mathbb{P}[a = b] \leq 1.$$

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

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

$$\mathbb{E}[d_{T'}(u, v) \mid a = b] < 8(i - 1)\beta d_{G'}(u, v).$$
which finally gives us

\[
\mathbb{E}[d_T(u, v)] < 2^{i+2} \frac{\beta d_{G'}(u, v)}{2^{i-1}} + 8(i-1)\beta d_{G'}(u, v)
= 8(1 + i - 1)\beta d_{G'}(u, v)
= 8i\beta d_{G'}(u, v).
\]

This proves the claim. \(\square\)

Then we set \(G' = G\) and \(i = \lfloor \log \Delta \rfloor + 1\) to get that \(\alpha_B = O(i\beta) = O(\log \Delta \log n)\). \(\square\)

### 5.2 A Low-Diameter Decomposition Scheme

Finally, we give a LDD scheme that proves lemma 5.14.

**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.

**Proof of lemma 5.14.** First, we check that each cluster’s diameter will be at most \(D\) with high probability. It suffices to check that \(R_v \leq D/2\) for each cluster \(G_v\). This is because \(G\) is a metric graph. Thus, if \(R_v \leq D/2\), then for any \(x, y \in G_v\), we invoke the triangle inequality to get

\[
d_G(x, y) \leq d_G(x, v) + d_G(v, y) \leq D/2 + D/2 = D.
\]
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.

The probability that \( R_v > D/2 \) for one particular cluster is

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

Therefore, by union bound, the probability that all clusters \( G_v \) have \( R_v \leq D/2 \) is

\[
\Pr[\forall v \in V, R_v \leq D/2] = 1 - \Pr[\exists v \in V, R_v > D/2] \\
\geq 1 - \frac{n}{n^2} \\
= 1 - \frac{1}{n}.
\]

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 \( \Pr[x, y \text{ in different clusters}] \leq \frac{\beta d_G(u,v)}{D} \) 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

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

Therefore, this LDD scheme gives us \( \beta = O(\log n) \). \( \square \)
Remark 5.16. For general graphs, one may realize that Bartal’s construction does not produce true spanning trees. For many applications, it suffices that the trees have $\alpha$-stretch on the vertex even if they are not spanning or have Steiner vertices.

It turns out that while there are major obstacles to finding true low-stretch spanning trees, it is indeed possible. Elkin et al. [EST05] gave the first polylog-stretch spanning trees, which took eight years following Bartal’s construction. (The first low-stretch spanning trees had stretch $2^{O(\sqrt{\log n \log \log n})}$ by Alon et al. [AKPW95], which is smaller than $n^\epsilon$ for any $\epsilon > 0$ but larger than polylogarithmic, i.e., $(\log n)^C$ for any $C > 0$.)

6 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 distorted 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 $\ell_p$-space with $O((\log n)/p)$ distortion [Mat97].

Acknowledgments

These lecture notes were scribed by Roie Levin, based on previous scribe notes of Titouan Rigoudy, Tom Tseng.

References


