CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2025**

Lecture 22

November 25, 2025

Instructor: Sepehr Assadi

Topics of this Lecture

For the last part of the course, we switch back to the topic of graph algorithms, and see some strong primitives and (very) advanced algorithms on this topic.

1 Graph Simplification

The topic of today's lecture is *graph simplification* and *decomposition*. How we can we "simplify" a graph, say, reduce its edges and put more "structure" on it, while still preserving some of its main properties? In the next lectures, we see how this can play an important role in designing more efficient algorithms. In this lecture however, we consider a very simple variant of this problem: how to simplify a graph while preserving its shortest path structures approximately.

Given an undirected graph $G = (V, E)$, for any $u, v \in V$, the distance between u, v—denoted by$dist_G(u, v)$—is the length of the shortest path between u and v. When the graph G is weighted, $dist_G(u, v)$ is the sum of the edge weights on the minimum weight between u and v, which we continue to call the shortest path.

The goal of this lecture is to simplify G by reducing it to a *tree* T such that the shortest path distances in the tree will be approximately the same as the ones in G. This is generally quite useful as computation over trees is significantly simpler than over arbitrary graphs for most problems of interest (shortest path itself becomes trivial since there is only one path between each pair of vertices in a tree!).

Before delving into the actual result, let us first consider some naive attempts on approximating distances of a given graph with graphs with simpler structures. Given a graph G, consider "simplify" G by a complete graph K such that for all $u, v \in V$, $w_K(u, v) = dist_G(u, v)$, where $w_K(u, v)$ is the weight of the edge (u, v) in K. Then, to find the shortest path of any u, v, we simply look up the weight of this edge in K. This preserve the distances in G *exactly*, but can be very time consuming to construct, and take up a lot of space, so not clear how much "simpler" it is (although computing the distances from it is indeed simple).

Another approach would be to embed a graph G into a *single* tree such that $dist(u, v)$ is approximately preserved for all $u, v \in V$. A simple sanity check shows that such guarantee cannot always exist. Consider a cycle graph of n vertices: any way of turning the graph into a tree must remove at least one edge (u, v), which changes $dist_G(u, v)$ from 1 to $n - 1$, thus, nowhere close to a good approximation of $dist_G(u, v)$.

2 Probabilistic Tree embeddings

A workaround to the issue above—that a single tree can never preserve all distances in many graphs—was first introduced by a beautiful result of Bartal [Bar96], referred to as *probabilistic tree embedding*. The general goal is to consider a distribution of trees $\mathcal{D}$ such that the distance between any pair of vertices are approximately preserved in expectation, i.e. $\forall u, v \in V$,

$$\mathop{\mathbb{E}}_{T \sim \mathcal{D}} [dist_T(u, v)] \approx dist_G(u, v);$$

i.e., if we fix a pair of vertices and then sample a tree from this distribution, the vertices will have roughly the same distance in T and G in expectation. We should emphasize that here $V(T) \supseteq V(G)$, i.e., we allow additional vertices to appear in T as well.

Theorem 1 ([Bar96])**.** *For any undirected graph $G = (V, E)$, there exists a distribution $\mathcal{D}$ of weighted trees $T_1, \cdots, T_k$ such that for all $i \in [k]$, we have $V(G) \subseteq V(T_i)$ and that for any fixed pair of vertices $u, v \in V(G)$,*

$$dist_G(u, v) \leqslant \mathop{\mathbb{E}}_{T \sim D} [dist_T(u, v)] \leqslant O(\log^2 n) \cdot dist_G(u, v).$$

We call the $O(\log^2 n)$ factor above the **stretch** *of the tree embedding.*

It is worth pointing out that the stretch of tree embeddings were subsequently improved to $O(\log n)$ in [FRT04] which is the *optimal* bound for this problem. However, we stick with the slightly sub-optimal approach of [Bar96] which is easier for our purpose.

This result relies on another important topic in algorithm design: *Low Diameter Decomposition (LDD)*. We first review this topic and then show how to use it to prove Theorem 1.

3 Low Diameter Decomposition (LDD)

We start with some definitions. Given a graph $G = (V, E)$, the **diameter** of G is defined as

$$D(G) := \max_{u,v \in V} dist_G(u, v),$$

namely, the distance between the "furthest" pairs of vertices. For any subset $S \subseteq V$, the **weak diameter** of S is defined as

$$D(S) := \max_{u,v \in S} dist_G(u, v).$$

Note that the weak diameter of S is *not* the diameter of the induced subgraph $G[S]$, as we still calculate the distances in the entire graph G and not just $G[S]$; this is the reason this is called the *weak* diameter. Finally, for any integer $r \geqslant 1$ and any vertex $v \in V$, we define

$$Ball(v, r) := \{u \in V \mid dist_G(u, v) \leqslant r\}$$

and refer to it as the **ball of radius** r centered at v. We can now define LDDs as follows.

Theorem 2 (Low Diameter Decomposition)**.** *Given any undirected graph $G = (V, E)$ and parameter $D \geqslant 0$, there exists a random partition $V = C_1 \sqcup C_2 \sqcup \cdots \sqcup C_k$ such that:*

1. *$D(C_i) \leqslant D$ for all $i \in [k]$, i.e., weak diameter of each "cluster" C_i is at most D (deterministically);*
2. *For all $u, v \in V$,*

$$\Pr(C(u) \neq C(v)) = \frac{O(\log n)}{D} \cdot dist_G(u, v),$$

where $C(u)$ denotes the partition containing vertex u.

This theorem states that we can always partition a graph into disjoint components such that vertices inside each component have short distances with each other and vertices that are close in the original graphs are unlikely to be assigned to different components.

Proof of Theorem **??**. We show the following procedure produces a partition $C_1 \sqcup \cdots \sqcup C_k$ of V that satisfies the two properties in LDD with high probability.

Algorithm 1. An algorithm for constructing an LDD.

Start with all vertices in V being unmarked. Repeat until there are no unmarked vertices:

1. Randomly pick an unmarked vertex v
2. Sample $R_v \sim G(p)$, where $G(p)$ is the geometric distribution of success probability $p := \dfrac{10 \log n}{D}$ on each trial[a].
3. Put all *unmarked vertices* in $B(v, R_v)$ into a cluster. Mark all of them.

[a] That is R_v is distributed as the number of trials we need to run before getting the first success.

Let us first show that the weak diameter of each cluster will be at most D with high probability.

Claim 3. *Let C be any cluster output by Algorithm 1. Then, $D(C) \leqslant D$ with high probability.*

Proof. Let v be the center of the cluster C, namely, the vertex picked in Algorithm 1 that led to C being $B(v, R_v)$ intersection with unmarked vertices. Any pair of vertices u, w in C have

$$dist_G(u, w) \leqslant dist_G(u, v) + dist_G(v, w) \leqslant R_v + R_v,$$

since both $u, w \in B(v, R_v)$. We argue that with high probability, $R_v \leqslant D/2$ which happens because

$$\Pr\left(R_v > \frac{D}{2}\right) \leqslant (1-p)^{D/2} \leqslant \exp\left(-\frac{10 \log n}{D} \cdot \frac{D}{2}\right) \leqslant n^{-5};$$

the rest follows from a union bound over the at most n clusters. □

We now prove the second property. The main part is to prove it for adjacent vertices, which is done by the following claim – we will then show how this easily implies the property for all pairs of vertices.

Claim 4. *For any edge (u, v),*

$$\Pr\left(C(u) \neq C(v)\right) = \frac{O(\log n)}{D}.$$

Proof. Let $(u, v) \in E$. We have

$$\begin{aligned}\Pr\left(C(u) \neq C(v)\right) = \Pr u \text{ was clustered first} \cdot \Pr\left(C(u) \neq C(v) \mid u \text{ was clustered first}\right) \\ + \Pr v \text{ was clustered first} \cdot \Pr\left(C(u) \neq C(v) \mid v \text{ was clustered first}\right).\end{aligned}$$

Let us consider the term

$$\Pr\left(C(u) \neq C(v) \mid u \text{ was clustered first}\right).$$

Let w be the vertex which clustered u. By conditioning on u being clustered first, we have that

$$R_w \geqslant dist_G(w, u).$$

At the same time, for $C(u) \neq C(v)$, we should have

$$R_w < dist_G(w,v) \leqslant dist_G(w,u) + 1,$$

given that there is an edge (u,v) in the graph. Thus, we have,

$$\Pr\left(C(u) \neq C(v) \mid u \text{ was clustered first}\right) = \Pr\left(R_w < dist_G(w,u) + 1 \mid R_w \geqslant dist_G(w,u)\right) = p = \frac{O(\log n)}{D}.$$

Since we can bound the term when v is clustered first exactly the same way also, we can conclude the proof of the claim. □

Finally, we extend the property of Claim 4 to all pairs of vertices (u,v). Consider a shortest path $P_{uv} = w_1, \ldots, w_{d+1}$ between u and v where $w_1 = u$, $w_{d+1} = v$ and $d = dist_G(u,v)$. By union bound, we have,

$$\Pr\left(C(u) \neq C(v)\right) = \Pr\left(\vee_{i=1}^{d} C(w_i) \neq C(w_{i+1})\right) \leqslant \sum_{i=1}^{d} \Pr\left(C(w_i) \neq C(w_{i+1})\right) \leqslant dist_G(u,v) \cdot \frac{O(\log n)}{D},$$

by applying Claim 4 to each edge (w_i, w_{i+1}).

Finally, we note that technically, we only showed $D(C_i) \leqslant D$ holds with high probability. To make this a deterministic statement, we can modify Algorithm 1 such that when there is a cluster that violates the diameter constraint, we output the trivial output where every vertex in that cluster is now its own cluster. Since the probability of this occurring is $\leqslant \frac{1}{n^4}$, the increase in the error probability of $\Pr(C(u) \neq C(v))$ is negligible, so $\Pr(C(u) \neq C(v)) \lesssim \frac{\log n}{D} \cdot dist_G(u,v)$ still holds but now we have a deterministic upper bound on the weak diameter of each cluster. □

4 Back to Tree Embeddings: Proof of Theorem 1

We now go back to probabilistic tree embeddings and prove Theorem 1.

Proof of Theorem 1. We construct a tree embedding of G by recursively finding LDDs in G for geometrically decreasing values of the weak diameter. The algorithm is formally as follows.

Algorithm 2. A probabilistic tree embedding algorithm given $G = (V,E)$, a set $U \subseteq V$ of vertices to be clustered, and a parameter D as an upper bound on the weak diameter of U in G. To obtain a tree embedding of a given graph G, we run this algorithm for (G, V, n).

1. If G only has a single vertex, return that single vertex as the tree embedding.
2. Let $C_1 \sqcup \cdots \sqcup C_k$ be an LDD of U in G with parameter $D/2$ using Theorem 2.
3. For each $i \in [k]$, recursively compute a tree embedding of $(G, C_i, D/2)$ called T_i.
4. Create a new tree T using a new root r which is connected to the roots of each of the sub-trees T_i's for $i \in [k]$, using an edge of weight $D/2$. Return T as the tree embedding of (G, U, D).

First observe that the vertices in G are the leaves in the outputted tree T. Roughly speaking, the procedure recursively split vertices into different groups at each level based on their distance. For any $u, v \in V$, if $dist_G(u,v)$ is small, then they are likely to be split towards the end of the algorithm, hence the smallest subtree in T that contains u, v is expected to be small and subsequently $dist_T(u,v)$ is small. If $dist_G(u,v)$ is large, then they are expected to be separated early on, hence $dist_T(u,v)$ is large.

By construction, for all possible tree T output by the algorithm we have $V(G) \subseteq V(T)$, which gives the first property of the embedding.

For the second property, we first claim that $dist_T(u,v) \geqslant dist_G(u,v)$ for all possible output tree T. Consider the recursive call that separate u and v and let D be the weak diameter parameter of that call. Since u and v are not separated yet, we have $dist_G(u,v) \leqslant D$ since weak diameter of a recursive call with value D is at most D. On the other hand, since u and v are now separated, we have, $dist_T(u,v) \geqslant D$ as we need to traverse two edges of weight $D/2$ to go from u to the root of T and then to v, hence, we always have

$$dist_T(u,v) \geqslant dist_G(u,v),$$

deterministically (and thus certainly in expectation).

Finally, we argue that expected stretch of T cannot be too large either. We have

$$\begin{aligned}
\mathop{\mathbb{E}}_{T}[dist_T(u,v)] &\leqslant \sum_{i=0}^{\log(n/D)} \Pr\left(u,v \text{ separated in call } D \text{ where } 2^i \cdot dist_G(u,v) \leqslant D < 2^{i+1} \cdot dist_G(u,v)\right) \cdot D \\
&\qquad\qquad \text{(by the construction of the algorithm)} \\
&= \sum_{i=0}^{\log(n/D)} \underbrace{\frac{O(\log n)}{2^i \cdot dist_G(u,v)} \cdot dist_G(u,v)}_{\text{by Theorem 2}} \quad \cdot \quad \underbrace{\left(2^{i+1} \cdot dist_G(u,v)\right)}_{\text{upper bound on } D} \\
&= O(\log^2 n).
\end{aligned}$$

This concludes the proof. □

References

[Bar96] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 184–193, 1996. 2

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004. Special Issue on STOC 2003. 2