DISCLAIMER: These notes are not necessarily an accurate representation of what I said during the class. They are mostly what I intend to say, and have not been carefully edited.

Recall the problem of finding low diameter decompositions.

**Definition 1.** A $(\beta, \Delta)$ probabilistic low diameter decomposition is a distribution over partition of vertices such that

1. Each partition has diameter $\Delta$.
2. Each edge is between pieces with probability at most $\beta$.

Specifically, we wanted to find $(\beta, O(\beta^{-1} \log n))$ probabilistic low diameter decompositions. Which is a distribution over partition of the graph into vertex subsets, so that each partition has diameter $O(\beta^{-1} \log n)$, and an edge is cut with probability at most $\beta$.

In Lecture 7 (where we introduced tree embeddings), we gave an algorithm (based on ball growing) that found small diameter pieces with few edges between them. But this guarantee was only in the total number of edges. Bartal trees, which we introduced in Lecture 8, instead require bounds on the expected stretch of an edge.

The focus of this lecture is an algorithm based on a more global/ symmetric view of ball growing [MPX13]. It has in turn been used in a variety of algorithms related to spanners and hopsets [MPVX15 EN16 EN17]. It can be viewed as ‘reversing’ the algorithmic and theoretical components of the previous ball growing algorithm:

1. Ball growing has its algorithm driven by certifying number of edges between pieces, and spends the proof on bounding diameter.
2. This routine has its algorithm driving by producing small diameter pieces, and spends the proof bounding the probability of edges being cut.

The starting point is the need to have a shortest path tree contained in each piece, known as strong diameter decomposition. This suggests the approach of doing tie breaks based on distances. Specifically, we can ensure the shortest path from $u$ to the center of the cluster, $v$ is in the piece corresponding to $u$ by associating a starting time $\delta_v$ for all vertices, and assigning $u$ to

$$\arg \min_v \delta_v + d(u, v)$$

If $u$ assigned to $v$, the entire shortest path from $u$ to $v$ assigned to $v$ as well.
Such a routine also has a simple parallel implementation: BFS to distance about \( \max(\delta_u) \), parallel if all \( \delta_u \) are small.

So the only question left is how to pick the \( \delta_v \)'s. Here the intuition is to think about running shortest path as setting the graph on fire, and the \( \delta_v \)'s serve as the starting time for such fires:

- Compete graph: 1 vertex burn everything
- Line: about \( \beta n \) fires.

So we want to interpolate between these: we do so by gradually setting the graph on fire instead.

The choice that works is:

\[
\delta_u \leftarrow -\text{Exp}(\beta)
\]

where \( \text{Exp}(\beta) \) is the exponential distribution with parameter \( \beta \):

- Density at \( x \): \( \beta \exp(-\beta x) \).
- Cumulative at \( x \): \( 1 - \exp(-\beta x) \).

A more discrete interpretation of this schema is that we should expect one vertex to catch fire at time \( -\beta \log n \), two more at \( -\beta(\log n - 1) \), four more at \( -\beta(\log n - 2) \), and so on.

Implementing this Decomposition

- Breath first search (BFS) / Dijkstra’s algorithm:
  - Finds shortest paths from a single vertex to all vertices.
  - Visit vertices in increasing order of distance: can because edge weights are non-negative
  - Each edge visited once, \( O(m \log n) \) time with priority queue (can optimize, see PS1 problem 7).
  - Parallelization: process one layer (nodes with same distance) at a time.

- Create super source, \( s^* \).
- Distance from \( s^* \) to \( u \): \( \delta_u \).
- Negative lengths! Fix by setting length from \( s \) to \( u \) to \( \delta_u - \min_v \delta_v \).
- Fractional distances: all other edges are integral, fractional part only serves as tie-breaking.
To analyze this scheme, we first bound the maximum diameter of a piece. Note that \( u \) is a candidate cluster center for itself. So we have the diameter of any cluster is at most

\[
\max_u - \delta_u.
\]

By looking at the CDF of the exponential and applying union bound, we get that with high probability, \( \delta_u \leq O(\log n \beta^{-1}) \) for all \( u \).

Next, we bound the probability of each edge being cut: Note that if \( e \) is cut, then there are two fires that reach its midpoint within 1 time unit of each other.

We can view this in the reverse direction from the midpoint. The times at which the fires reach it are exponentials adjusted by the distance values:

\[
dist(m,u) - \delta_v.
\]

From this view, the question becomes: if we have a (large) collection of shifted exponentials, what is the probability that the minimum among them are within \( O(1) \) of each other.

To bound this, we use the memoryless property of exponential distributions:

\[
\Pr_X [X \geq n + m | x \geq m] = \Pr_X [X \geq n].
\]

In particular, this implies that the difference between largest and second largest still follows \( \text{Exp}(x, -\beta) \). This implies that the probability of fire in 1 unit of time is at most \( 1 - \exp(-\beta) \approx \beta \).

References

