6.889 Sublinear Time Algorithms April 22, 2019

Lecture 19

Lecturer: Ronitt Rubinfeld *Scribe: William Loucks*

1 Graph Spanners

Definition 1 *Given undirected graph $G = (V, E)$, a **k-spanner** is a subgraph $H = (V, E')$ such that $E' \subseteq E$ and $\forall u, v \in V$, $dist_H(u, v) \leq k \cdot dist_G(u, v)$*

Equivalently, one could state that $\forall (u, v) \notin H$ (edges therein), there exists a path from u to v in H of length less than k, also known as the *stretch*. We know that optimal $(2k-1)$-spanners obtain $O(n^{1+1/k})$ edges for $k = 2, 3$, and 5. Further, assuming the Erdos Girth Conjecture, for all k, there exists G for which any $(2k-1)$-spanner must have $O(n^{1+1/k})$ edges.

2 LCA for 3-Spanner

Today, we show a 3-spanner with $\widetilde{O}(n^{3/2})$ edges and a probe complexity of $\widetilde{O}(n^{3/4})$. Note, the complexity varies based on whether or not one wishes to minimize the query complexity or the number of edges.
We assume the following probes with complexity $O(1)$ to graph G:

1. *Neighbor:* Given u, i, output the ith neighbor of u
2. *Adjacency:* Given (u, v), output j if $(u, v) \in G$ and where v is the jth neighbor of of u; output "no" otherwise
3. *Degree:* Given u, output $deg(u)$

First, consider a thought: pick a random center and if u and v are connected to the same center, delete the edge (u, v). Then $dist_G(u, v) = 1$ and $dist_H(u, v) = 2$, permissible for a 3-spanner.

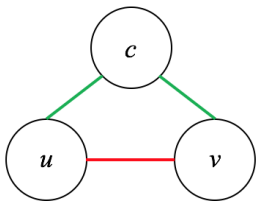

Figure 1: u and v connected to the same center, can delete (u, v)

However, we may not delete *enough* edges this way, and it may be difficult to determine – in sublinear time – that u and v are connected to the same center.

2.1 Global Construction of 3-Spanners [Baswana Sen '07]

Construction of H:

- Pick $S \subseteq V$ such that $|S| = \Theta(\sqrt{n} \cdot \log(n))$. In other words, each node tosses a coin with bias $\Theta(\frac{\log(n)}{\sqrt{n}})$. These form the *cluster centers*.
- With high probability, $\forall u \in V$ such that u has degree greater than $\sqrt{n}$, u is adjacent to at least one center $v \in S$. Ultimately, u chooses only one $v \in S$ to be its cluster center.

Further, to construct H we comply with the following rules:

1. If u has low degree ($< \sqrt{n}$), then add all edges (u, v)
2. If u has high degree ($\geq \sqrt{n}$), then add edge (u, v) to cluster center
3. If u has high degree ($\geq \sqrt{n}$), add **one** edge to **every** adjacent cluster

Given the above, step 1 results in at most $n \cdot \sqrt{n}$ edges; step 2 adds at most n edges; and step 3 results in at most $n \cdot \sqrt{n} \cdot \log(n)$ edges. Thus, the overall edge count obtains $\widetilde{O}(n^{3/2})$.

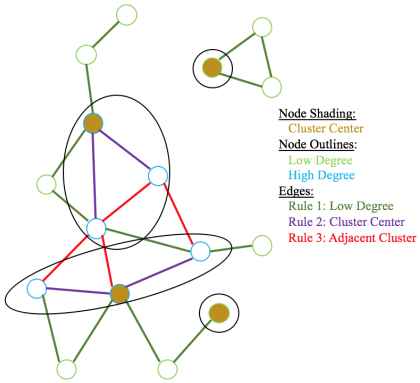

Figure 2: Example constructing H given the above rules. Clusters circled for illustration.

Note, for u and v in the same cluster, both nodes kept their edges to the cluster center. Thus, $dist_H(u, v) = 2$. Moreover, for u and v in different clusters: if $(u, v) \notin H$, the nodes must have kept some other edge (u, w) such that w is in v's cluster. Thus, either w is v's center, or $(u, w), (w, c_v), (c_v, v) \in H$ yielding $dist_H(u, v) = 3$.

2.2 Local Algorithms for Constructing H

Given $(u, v) \in G$, is $(u, v) \in H$?

1. If u or v have low degree, then yes.
2. If u is v's center, then yes. Equivalently, if v is u's center, then yes.
3. If (u, v) is a chosen edge from u to v's cluster or from v to u's cluster, then yes.

Considering the above, for rule 1, we just use two *degree* probes. However, implementing rules 2 and 3 require additional effort. We first illustrate a naive approach and then depict a less complex alternative.

A naive first attempt may be to choose the first node, w, which connects to each center in u's incidence list. Implementing rules 2 and 3 could then be as follows:

Rule 2:

- Check if v is a center
- Check if any node preceding v on u's incidence list is a center

This results in overall complexity $O(\text{max degree})$, but we can do better. Namely, $O(\sqrt{n})$ by our observation earlier that if u has degree greater than $\sqrt{n}$, then u is adjacent to at least one center with high probability.

Rule 3:

- Locate v's cluster center, c_v
- Check if any neighbors of c_v come earlier in u's incidence list.

The resultant complexity is $O(n)$, since we take $O(\sqrt{n})$ to ensure c_v is the selected neighbor's center and $\Omega(\sqrt{n})$ to determine if the selected neighbor is earlier in u's incidence list.

The following are improved plans to test rules 2 and 3:

***Improved* Rule 2:** u chooses all centers in the first $\sqrt{n}$ locations of the incidence list:

$$c_u = \{v|\ v \text{ is in the first } \sqrt{n} \text{ locations of } u\text{'s incidence list and } v \text{ is a center}\}$$

Note, with high probability, $1 \leq |c_u| \leq \log(n)$ if $deg(u) \geq \sqrt{n}$

Now,

- Check if v is a center
  1. Invoke an *adjacency* query to determine v's location in u's incidence list
  2. Check if v is a center
- Check the first $\sqrt{n}$ locations in u's list to determine which are centers.

Now, we can use one *adjacency* probe to determine if u is v's center and, if needed, $\sqrt{n}$ *neighbor* probes to find another center.

***Improved* Rule 3:** u chooses the first edge which introduces u to v's cluster.

- Locate the cluster centers, C_v
- For each neighbor x of u up to v:
  - For each $w \in C_v$:
    If w is a center of x, then cross off w
  - If any $w \in C_v$ not crossed off, then keep (u, v) in H
  - Else, discard.

In the end, we used $\sqrt{n}$ neighbor probes to compute the set of cluster centers. Further, for each neighbor of u, $deg(u)$, we iterated over the number of cluster centers, $O(\log(n))$, and invoked one *adjacency* query, $O(1)$, to determine if any cluster center was the center for a given neighbor. Assuming the max degree to be $O(n^{3/4})$, the overall complexity of this implementation to check rule 3 obtains $\widetilde{O}(n^{3/4})$.