6.5240 Sublinear Time Algorithms November 16, 2022

Lecture 19

Lecturer: Ronitt Rubinfeld *Scribe: Stan Zhang*

1 Definitions

Definition 1 *Given graph $G = (V, E)$, a "k-spanner" is a subgraph $H = (V, E'), E' \subset E$ such that $\forall v, u$, $\text{dist}_H(u, v) \leq k \, \text{dist}_G(u, v)$. Equivalently, H is a k-spanner if for all $(u, v) \in G \setminus H$, exists a path from u to v in H with length $\leq k$.*

Theorem 2 *$\forall k$, there exists a $2k - 1$ spanner with at most $O(n^{1+1/k})$ edges. Furthermore, this bound is tight for $k = 2, 3, 5$. The Erdos-Girth conjecture states that this bound is tight for all k.*

2 Model

2.1 Access to G

Neighbor probes: Given (u, i) output ith neighbor of u
Adjacency probes: Given (u, v) output whether or not $(u, v) \in G$. If $(u, v) \in G$ then the probe outputs j, the index of the location of (u, v). The probe outputs "no" if $(u, v) \notin G$.
Degree probes: Given u output $\deg(u)$

2.2 Desired Algorithm Performance

We present a LCA for 3-spanner with $\tilde{O}(n^{3/2})$ edges (setting $k = 2$ in Theorem 1.1 shows that such a spanner exists). Our algorithm allows us to query whether or not a given edge (u, v) is in the spanner in $\tilde{O}(n^{3/4})$. The best known algorithm takes $\tilde{O}(n^{1/2})$ queries (Arviv and Levi).

The point of a spanner is to make a dense graph more sparse. Thus, spanners are most pertinent to high-degree graphs.

3 Global Algorithm

3.1 First Attempt

Each node in V decides to be a "center" with probability c, iid. If u, v such that $(u, v) \in G$ are both connected to the same center then can delete (u, v).
Problem 1: Can we delete enough edges this way?
Problem 2: Can we figure out fast enough if u, v are connected to the same center?
For the rest of the lecture, we assume that the max degree is $n^{3/4}$. This case is still nontrivial, and the general case builds on these ideas.

3.2 Global Construction of 3-Spanners [Baswana Sen 2007]

We present a global algorithm that constructs a spanner of $\tilde{O}(n^{3/2})$ edges. Pick $S \subseteq V$ such that $|S| = \Theta(\sqrt{n} \log n)$. We call S the cluster centers. Each node independently puts itself in S with probability $\Theta(\frac{\log n}{\sqrt{n}})$.

Useful Observation: $\forall u \in V$ such that $\deg(u) \geq \sqrt{n}$, there is $\geq 1 - \frac{1}{n}$ chance that u is adjacent to a node $v \in S$. Using union bound, there is high probability that $\forall u \in V$ with $\deg(u) \geq \sqrt{n}$, u is adjacent to some node $v \in S$.

3.2.1 Constructing H

1.If $\deg(u) < \sqrt{n}$, add all edges (u, v) where $v \in S$
2.If $\deg(u) \geq \sqrt{n}$, add edge from u to some node in S (such a node exists with high probability by the useful observation).
3.If $\deg(u) \geq \sqrt{n}$, add 1 edge from u to some node in every adjacent cluster.

1) adds at most $n\sqrt{n}$ edges, as every low degree node has at most $\sqrt{n}$ edges, meaning that we add at most $\sqrt{n}$ edges for each node.

2) adds at most n edges, as we add at most 1 edge for every node.

3) adds at most $n\sqrt{n}\log n$ clusters as $|S| = \Theta(\sqrt{n}\log n)$ and for every node we add at most 1 edge to every node in S.

In total, we add at most $\tilde{O}(n^{3/2})$ edges.

We have a spanner with $\tilde{O}(n^{3/2})$, but what is the stretch? Suppose $(u, v) \in G \setminus H$. If u, v are in the same cluster center, then they both keep edge to center 3, and $\text{dist}_H(u, v) = 2$. If (u, v) are in different clusters, then v must have kept to some z in u's cluster by step 3 of the construction. Then, if $z \neq c_u$, then there is a path of length 3 from u to v: $u \to c_u \to z \to v$, where c_u the center of the cluster u is in, and if $z = c_u$, then there is a path of length 2 from u to v: $u \to c_u \to v$. In any case, $\text{dist}_H(u, v) \leq 3$, as desired.

4 Converting Global Algorithm to LCA

Question: Given $(u, v) \in G$, is it in H?
Rule 1: If (u, v) are low-degree, then the answer is "yes". We can query $\deg(u), \deg(v)$, and if either are less than $\sqrt{n}$, then we're done.

Rule 2: If (u, v) have high-degree. If v is u's center or vice versa, then say Yes.
Rule 3: If (u, v) is the chosen edge from u to v's cluster, or from v to u's cluster, then say Yes.
If none of the rules are satisfied, then say "No".

4.1 First Center Attempt

For each high-degree node u, we choose the first center in the list of u's neighbors to be u's cluster. Furthermore, for each u, v such that v is a center and u is adjacent to v's cluster, chosen edge from u to v's cluster is the first incident node w in v's cluster.

4.1.1 Rule 2

We can figure out if v the cluster center of u in $O(\sqrt{n})$ queries because with we hit a cluster center in the first $O(\sqrt{n})$ elements with high probability due to the fact that all of them are independent coin tosses (this follows from the same analysis as the useful observation).

4.1.2 Rule 3

This is problematic. On querying (u, v), we want to keep the edge if v introduces u to another cluster (eg v is the first node incident to u that is in the same cluster as v). The problem is that the algorithm is $\deg(u) \times \sqrt{n}$, because for each incident node we need to check if it is in the same cluster as v. Alternatively, we can look at all the nodes connected to c_v, and see which ones are connected to u that come before v. Either way, it takes $\deg(u) \times \sqrt{n}$ queries, which is too much.

4.2 Multiple Center Attempt

For each high-degree node u, we connect u to all centers in the first $\sqrt{n}$ locations. In other words, $S_u = \{v | v \text{is in the first } \sqrt{n} \text{ locations of u and v is a center}\}$. For rule 3, we keep edge (u, v) if v introduces u to any cluster in S_v.

Observation: For all u such that $\deg(u) \geq \sqrt{n}$, $1 \leq |S_u|$ from the useful obesrvation, but it is also unlikely that $|S_u| > \log^2 n$ by Chernoff.

4.2.1 Rule 2

At most $O(\log^2 n)$ edges are kept per node, or at most $O(n \log^2 n)$ total. To verify if v is in S_u, just see if both v is a center and v is in the first $\sqrt{n}$ elements in u's adjacency list, using adjacency probe. This takes $O(1)$ time. We can compute S_u in $\sqrt{n}$ steps – go down the first $\sqrt{n}$ locations of u, and see which of these decided to be centers.

4.2.2 Rule 3

First, we find S_v in $O(\sqrt{n})$ steps. Then, for each $w \in S_v$, check if there exists x that comes before v in u's adjacency list such that w is the center of x (we for each x we can check in 1 query by checking the index of w in x)- if so, then cross w off. If there exists a w that is not crossed off, then v introduces u to w, and thus we keep (u, v). Otherwise, v does not introduce u to any w, and thus we discard (u, v).

This takes $\tilde{O}(\deg(u))$ queries - there are at most $\log^2 n$ cluster centers in S_v, and for each of these cluster centers $w \in S_v$ we check at most the entire adjacency list of u up to v (meaning at most $\deg(u)$ elements), making 1 query for each possible x that comes before v in the adjacency list to see if S_v is the cluster center of x. Since we assumed that $\deg(u) = O(n^{3/4})$, this algorithm has the desired performance of $\tilde{O}(n^{3/4})$ queries.