COL751 - Lecture 6

1 A linear time 3-spanner construction

The spanner constructions we have seen till now relied on computing distances from either a set of sources or between multiple source-destination pairs. In fact, though spanners were introduced in early 1990s, no linear time constructions were known for more than a decade. In 2003, Baswana and Sen (from IIT Delhi) published a novel approach for computing $2k-1$ spanners which demonstrated that spanners are inherently local structures that do not require distance computations. Further, they showed that their algorithms are easy to parallelize and implement in dynamic / streaming settings. We will see in this section the Baswana and Sen's construction for $k = 3$.

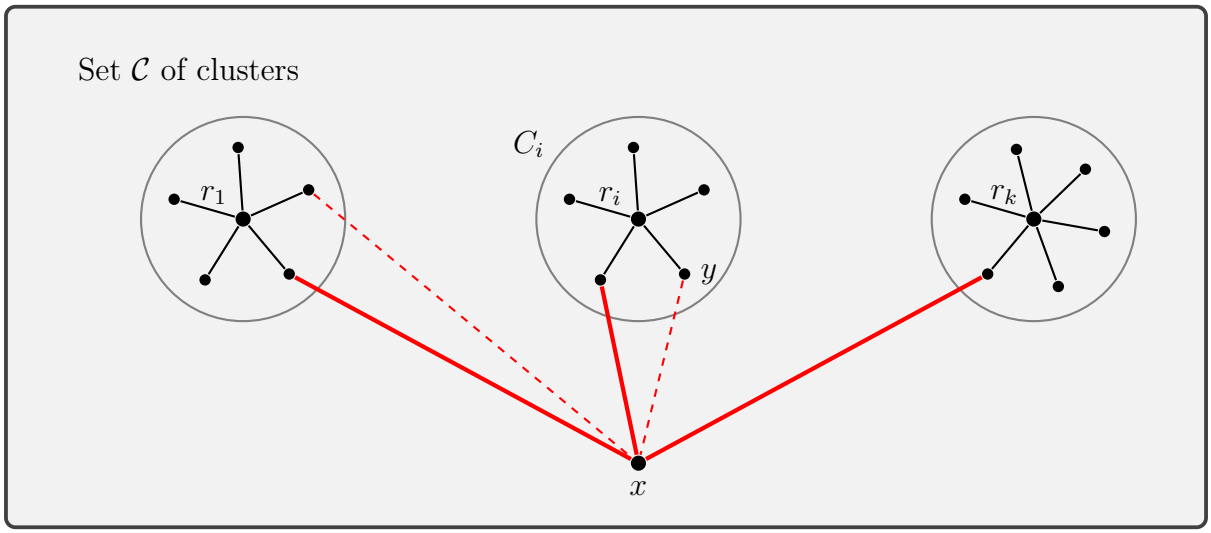
Algorithm

1. We initialize H as $(V, \emptyset)$, and compute a random set R of $k = \Theta(\sqrt{n})$ vertices.

2. Next we add to H, all edges incident to vertices of degree at most $\sqrt{n}\log n$.

3. For each $x \in V$, we add to H an edge from the set $x \times (N(x) \cap R)$, if it exists. This step creates $|R|$ clusters (or stars) around vertices in set R, which we henceforth refer by notation $\mathcal{C}$.

4. Finally, for each $x \in V$ and each cluster $C \in \mathcal{C}$ satisfying x has a neighbor in C, we add to H exactly one edge from the set $x \times V(C)$.



Lemma 1 *The number of edges in H is at most $O(n\sqrt{n}\log n)$.*

Lemma 2 *The time to compute graph H is at most $O(m + n\sqrt{n}\log n)$, which is $O(m)$ for $m = \Omega(n^{1.5}\log n)$.*

Lemma 3 *With probability at least $1 - 1/n^2$, each vertex of degree at least $\sqrt{n}\log n$ has a neighbor in R.*

Lemma 4 *For each edge (x, y) not in H, $dist(x, y, H) \leqslant 3$.*

Proof: Consider an edge (x, y) not present in H. Note that degree of y must be at least $\sqrt{n}\log n$ and hence y would have a neighbor in R, say r_i. Let us suppose vertex y belongs to cluster C_i. (See figure in previous page). As there is at least one edge between x and C_i in H, we have $dist(x, y, H) \leqslant 3$. This proves the claim. □

Theorem 5 (Baswana, Sen (2003)) *For any n vertex, m edges undirected graph G we can compute a 3-spanner of $O(n^{1.5}\log n)$ size in $O(m + n)$ time.*

Homework How can you extend above construction to weighted graphs? Also how can you parallelize the construction of 3-spanners?

2 Construction of 3-spanner in streaming model

Consider a scenario where graph $G = (V, E)$ is stored in some server, where accessing data is a costly step. The question in streaming model is - *whether it is possible to compute a $(2k-1)$-spanner of optimal size by doing only one single pass over the list $E = (e_1, \ldots, e_m)$ of edges.* The answer is yes, and here is a simple algorithm for the same.

Algorithm

1. We initialize H as $(V, \emptyset)$, and compute a random set R of $k = \Theta(\sqrt{n})$ vertices.

2. For each vertex $v \in V$, we keep the following information:

   - $deg(v)$ − stores degree of v in partially scanned G.
   - $cluster(v)$ − stores mapping from v to cluster of v, if it exists.
   - $flag(v, C_i)$ − stores whether a neighbor between v and C_i is added to H, for each $i \in [1, \sqrt{n}\log n]$.

3. On scanning an edge $e = (x, y)$, we

   - The first step is to increment degree of x and y by one. Next if degree of x or y in current H is at most $\sqrt{n}\log n$, then we add e to H.

- If $cluster(y) = NULL$ and $x \in R$, then add edge e to forest $\mathcal{C}$ and set $cluster(y) = x$. Similarly, if $cluster(x) = NULL$ and $y \in R$, then add edge e to forest $\mathcal{C}$ and set $cluster(x) = y$.
- If x has no neighbor in $cluster(y)$, then connect x to $cluster(y)$ using edge e. Similarly, if y has no neighbor in $cluster(x)$, then connect y to $cluster(x)$ using edge e.

The correctness of theorem follows from analysis in Section 1. Note that an edge once added to spanner is never removed, also each edge in the input stream is processed in worst case $O(1)$ time.

Theorem 6 *For any unweighted graph G a 3-spanner of size $O(\min(m, n^{1.5}\log n))$ can be computed in streaming model in one pass with $O(1)$ processing time per edge. The working memory required to compute spanner is $O(\min(m, n^{1.5}\log n))$, and we do not need to know m (length of input stream) a priori.*

Challenge Problem Can you extend the algorithm to fully-dynamic setting, where each entry of stream contains pair $(e, update)$, where *update* is either an *insertion* or a *deletion*.