CS 594: Representations in Algorithm Design **Spring 2022**

Lecture 12: 2/17/2022

Lecturer: Xiaorui Sun *Scribe: Jake Maranzatto*

1 Last time and today

- Dynamic Connectivity, Expanders, Expander Pruning

2 Dynamic Connectivity in $n^{o(1)}$ time deterministically.

In many cases we would prefer a deterministic algorithm over randomized algorithms(e.g. databases). Oftentimes if there exists a randomized algorithm we can find a deterministic algorithm with the same runtime guarantees. As an example for this lecture we can use the expander decomposition (which we can find in $m^{o(1)}$-time a $(\phi, \phi \cdot n^{o(1)})$-expander decomposition) to solve the dynamic connectivity problem.

Recall expander pruning: If G is a ϕ-expander, and we remove k edges from G to produce G'. Then there is an algorithm that runs in $O(k/\phi)$ to find a subset $S \subseteq V$ such that:

- $Vol(S) \leq \frac{8k}{\phi}$
- $E(S, \bar{S}) \leq 4k$
- $G'[\bar{S}]$ is a $\phi/6$ expander

Observe that we cannot hope that the whole graph G' is a good expander, as removing k edges may disconnect the graph.

2.1 Using expander pruning to solve dynamic connectivity

2.1.1 The basic idea

- run an expander decomposition on each connected component of G

- Use a single vertex to represent each component of the expander decomposition, in a new 'reduced graph' $G' = (V', E')$
- There is set of parallel edges $F \subset E'$ connecting $a', b' \in V'$ if and only if there are $|F|$ edges between $a, b \subset G$ in the expander decomposition of G. If the expander decomposition is 'good', then $|E'|$ is small
- To test connectivity in G, we need a map $P : V(G) \to V(G')$ such that $P(v)$ maps v to its corresponding expander in the decomposition. Therefore v is connected to $u \iff P(v)$ is connected to $P(u)$
- This handles queries, how do we handle updates to G?

2.1.2

Without loss of generality, assume G has bounded degree 3. If G does not have this property, for each vertex v with $d(v) > 3$, 'blow up' v into a path with length $d(v)$, and for each vertex i in the path connect i to exactly one neighbor of v.

Given G, produce an expander decomposition $(\frac{1}{\sqrt{m}}, \frac{n^{o(1)}}{\sqrt{m}})$, and as described above produce a reduced graph with $m^{1/2+o(1)}$ vertices and edges.

Consider the single updates for inserting and deleting an edge. Our goal is to handle these updates in time $O(m^{1/2+o(1)})$.

1. Case 1: Delete edge (x, y)
   (a) If (x, y) is a crossing edge in the expander decomposition, make no change to the expander decomposition, no change to the mapping function P, and remove a single parallel edge from G' corresponding to the expanders containing x, y. This takes $O(1)$-time.
   (b) If (x, y) is an edge contained in some expander, use the expander pruning algorithm. After removing (x, y) from the expander $H \subset G$, we can find a partition $S, \bar{S} \subset H$ with the properties outlined above. $\bar{S}$ is a $\phi/6$ expander, but no guarantee on S. So for the reduced graph G', the vertex corresponding to H now corresponds to $\bar{S}$. The remaining vertices in S are each added individually to G', that is $v \in S$ act as single-vertex expanders in G', so add all corresponding edges to $\bar{S}$ and $v \in S$. Further, update the mapping P as $P(x) = x$, $\forall x \in S$

   The running-time for this change is the time to add vertices to G', add edges incident to S, and to remove edges incident to $\bar{S}$. These can all be bounded as $O(Vol(S)) = O(\sqrt(m)$, as we assume the maximum degree of G is 3. Further we know updating the mapping P takes time $O(\sqrt{m})$

2. Case 2: Insert edge (x, y)

   (a) If (x, y) is a crossing edge, do not change the expander decomposition or map P, but insert (x, y) as a parallel edge in G' This takes $O(1) - time$

   (b) If (x, y) is a not a crossing edge, reduce this case to a crossing edge deletion. Suppose $x, y \in H$ for expander $H \subset G$. For every edge incident to $x, y \in H$, remove these edges from H. This then ensures that x, y are isolated vertices in both G, and G'. Adding back all these edges, plus (x, y) can be handled by case 1. The cost then is $O(\sqrt{m})$

Therefore, both updates can be handled in time $O(\sqrt{m})$, however the expansion of the decomposition is decreased by a constant factor, so after many single updates G' will have $O(n)$ vertices and $O(m)$ edges in the worst-case. However the following lemma proves useful.

Lemma 1 *One can handle a sequence of t updates in time $O(t\sqrt{m})$, and preserve a $\phi/6$ expander decomposition.*

Batch all edge deletions and edge insertions together. Add all edges first, then apply the expander pruning algorithm for the This idea, along with the preceding analysis gives the following Theorem.

Theorem 2 (Nanoyka,Saranurak,Wulff,Nelson 2017) *There is a deterministic, dynamic algorithm for the dynamic connectivity problem that runs in time $O(m^{1/2+o(1)})$ per update.*