PRINCETON UNIV. F'24 COS 597B: RECENT ADVANCES IN GRAPH ALGORITHMS

Lecture 5: (α, ϕ)-Boundary Linked Expander Decomposition

Lecturer: *Huacheng Yu* Scribe: *Baris Onat*

Notation

Let's first go over the notation used in this lecture note. Given graph $G = (V, E)$, let $n = |V|$ and $m = |E|$. For vertex sets $A, B \subseteq V(G)$, let $E(A, B)$ denote the edges in between the vertices of A and B, formally $E(A, B) := \{(u, v) \in E(G) \mid u \in A, v \in B\}$. Define the volume of a subset as the sum of degrees of each vertex in the subset. For $A \subseteq V(G)$, $vol(A) := \sum_{a \in A} deg(a)$. The O-Tilde notation $\tilde{O}(n)$ represents $O(polylog(n))$.

Recap of Last Lecture

In the last lecture, we defined the (α, ϕ)-boundary linked expander decomposition ((α, ϕ)-ED) as a vertex partition $U = \{U_i\}_{i=1}^{n}$ with corresponding $\phi_i \geq \phi$ such that the following holds:

- $\sum_i |E(U_i, V \setminus U_i)| \leq \tilde{O}(\phi \cdot m)$.
- $\forall i$, $G[U_i]^{\alpha/\phi}$ is a ϕ_i-expander. Remember that $G[U_i]^{\alpha/\phi}$ refers to the induced graph on the vertex set U_i plus some self-loops. More accurately, for edges going out of U_i, we add α/ϕ many self-loops to the endpoint in U_i. Observe that the addition of these self-loops increases the volume of U_i.
- $\forall i$, $|E(U_i, V(G) \setminus U_i)| \leq \tilde{O}(\phi_i \cdot vol_G[U_i])$.

Think of the constants as $\alpha = \frac{1}{polylog(n)}$ and $\phi = \frac{1}{n^{o(1)}} \ll \alpha$. Remember that "recourse" refers to the number of edge updates to $G_{\mathcal{U}}$, denoted by ρ. Last time, we considered the simpler case of batch updates, where we showed that this can be done in $\tilde{O}(1/\alpha)$ recourse per update. In this lecture, we drop this assumption but work with sequential updates.

This Lecture

In this lecture, we will show that the expander decomposition ED can be maintained in $n^{o(1)}$ time per edge insertion/deletion. The following is the main idea of the algorithm:

- If there are many updates to a set U, rebuild it.
- Otherwise, find small set P such that after removing P, $U \setminus P$ is a relatively good expander. Then, remove from P from U and build an (α, ϕ)-ED for P.

We will show that identifying P and maintaining it under edge insertions/deletions is doable with existing techniques. We will use the theorem below which we will give a sketch of its proof at the end of the lecture:

Theorem 1. *Let $w = \alpha/\phi$. Suppose $G[U]^w$ is a ϕ-expander. Then, for a sequence of online updates, a set P of "pruned" vertices can be maintained. Let P_i be the set P after the i-th update. Then, the following holds:*

- $P_0 = \emptyset$ *and* $P_i \subseteq P_{i+1}$.
- $vol_{G[U]^w}(P_i) \leq O(i/\phi)$ *and* $|E_G(P_i, \cup P_i)| \leq O(i)$.
- $|E_G(P_i, V \setminus U)| \leq O(i/\alpha)$.
- $G_i[U \setminus P_i]^w$ *is a* $\phi/38$*-expander.*
- *The total running time for k updates is $\tilde{O}(k/\phi^2)$.*

The proof of the above theorem will be done later. Now let's outline how the algorithm will work and use the pruned sets. Given k updates, we will have to periodically compute the $ED(P)$ from scratch. At all such points, we are guaranteed to have $G[U \setminus P]^w$ as a $\phi/38$-expander. Between all states in between the recomputations, we will use the pruning algorithm to prune $G[U \setminus P]^w$. Therefore, we will keep levels in which we will prune the pruned set and so on. The intuition for this subroutine is that as we go down the levels, the sets we have to maintain will get smaller and get recomputed more often.

Now let's describe the level structure in more detail. We will keep h levels for $h \leq O(\log_B n)$ and some constant B. At each level, we will maintain a pruned set P recursively. We will in fact show that $G[U \setminus \cup_{i=1}^{h} P_i]^w$ is a $\phi/(38^h)$-expander. To be more clear here, P_i is a prune set for $U \setminus \left(\bigcup_{n=1}^{i-1} P_n\right)$. At the sublevels, we work with $U \setminus P$, then do a subroutine of pruning from scratch. The goal is to maintain a $(\alpha/(38)^h, \phi/(38)^h)$-ED (taking into account the self loops from w into account) at the last level.

Now we can explain the algorithm:

- The main goal is to maintain the (α, ϕ)-ED, which we will recompute every $\tilde{O}(\phi \cdot vol(G))$ many updates. This gives us a vertex partition U_i, where each is a ϕ_i expander. This corresponds to the top level.
- Per U_i, we will maintain a pruned set P. We will recompute the (α, ϕ_i)-ED of U_i every $\tilde{O}(\phi_i \cdot vol_G(U_i))$ relevant updates.
- To maintain P efficiently, we will maintain a (α, ϕ_i)-ED on P, which will be recomputed for every $\tilde{O}(\frac{1}{B}\phi_i \cdot vol_G(U_i))$ many relevant updates.
- Remember that $G[U_i \setminus P]$ is a $\phi/38$-expander. When necessary, we will remove P from U. In this case, we will need to maintain the prune set $P' \subseteq U_i \setminus P$, which we will do by similarly maintaining an ED of P', recomputed periodically.
- We will maintain h many such levels. At the h-th level, we have a $\phi/38^h$ expander to maintain.

The runtime and recourse calculation of the algorithm is quite technical. We will instead focus on the amortized time analysis.

Amortized Time Analysis

Let's outline the time and recourse per operation for each level of the algorithm:

- In the top level where we maintain a (α, ϕ)-ED, we spend $1/\phi^{o(1)}$ time per operation and $\tilde{O}(1)$ recourse per operation.
- In the second level, where we maintain $P \subseteq U_i$, we will similarly have $1/\phi^{o(1)}$ time per operation and $\tilde{O}(1)$ recourse per operation.
- In the next level where we maintain $P' \subseteq (U_i \setminus P)$, we will spend $B(1/\phi)^{o(1)}$ time per operation and $\tilde{O}(B/\alpha)$ recourse per operation.
- In the last level, we will have $B(1/\phi)^{o(1)}38^{o(h)}$ time per operation and $\tilde{O}((B/\alpha)\cdot 38^{o(h)})$ recourse per operation.

This analysis shows that the last level is the bottleneck of the algorithm because it is recomputed the most often. As we go down the levels of the algorithm, we will recompute the ED more frequently. This outlines a method for maintaining the (α, ϕ)-ED by dividing the task into subroutines.

Maintaining the Pruned Set P

Initially, $\forall S \subseteq U$, $vol_{G[U]^w}(S) \leq \frac{1}{2} vol_{G[U]^w}(U)$ and $|E(S, U \setminus S)| \geq \phi \cdot vol(S)$. We want to mainly a slightly lower expansion throughout the edge updates, mainly $|E(S, U \setminus S)| \geq (\phi/38)\cdot vol(S)$. Observe that this bound breaks if $|E(S, U \setminus S)|$ decreases by $\Omega(1)$, a constant factor, or $vol(S)$ increases by a constant factor. The following are bad conditions that could break the inequality:

- Delete a cut edge: the LHS goes down by 1.
- Insert an edge inside S: the RHS goes up by $O(\phi)$.
- Insert an outgoing edge from S, the RHS goes up by $O(\alpha)$.

We will solve this issue by solving the following max-flow problem:

- Each edge of G has capacity $1/\phi$, each vertex v connects to the sink with an edge with capacity equal to $vol_{G[U]^w}(v)$.
- Whenever there is an update to the edge $(u, v) \in E(G)$, send $10/\phi$ flow to u and v.

It might be the case that the max flow is equal to the total capacity from S. Then, we will have for all S, (number of updates relevant to $S)\frac{10}{\phi} \leq vol(S) + |E(S, U \setminus S)|\frac{1}{\phi}$. This implies that G is still a $\Omega(\phi)$-expander.

However, it might be the case that the max flow is no longer equal to the total capacity from S. Then, we can use the push relabel algorithm to find some S such that (number of updates relevant to $S)\frac{10}{\phi} \geq \Omega(vol(S)) + 1$. Therefore, we can identify and maintain the prune set P by solving the appropriate max-flow problem.

References

1. Gramoz Goranci et al. “The expander hierarchy and its applications to dynamic graph algorithms”. In: *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’21. Virtual Event, Virginia: Society for Industrial and Applied Mathematics, 2021, pp. 2212–2228. ISBN: 9781611976465.