CMPSCI 711: More Advanced Algorithms

Section 2-1: Graph Streams

Andrew McGregor

Last Compiled: April 29, 2012

Graph Streams

- Consider a stream of m edges

$$\langle e_1, e_2, \ldots \quad \ldots, e_m \rangle$$

  defining a graph G with nodes $V = [n]$ and $E = \{e_1, \ldots, e_m\}$
- Massive graphs include social networks, web graph, call graphs, etc.
- What can we compute about G in $o(m)$ space?
- Focus on *semi-streaming* space restriction of $O(n \cdot \text{polylog}\, n)$ bits.

Warm-Up: Connectivity

- *Goal:* Compute the number of connected components.
- *Algorithm:* Maintain a spanning forest F
  - $F \leftarrow \emptyset$
  - For each edge (u, v), if u and v aren't connected in F,
  $$F \leftarrow F \cup \{(u, v)\}$$
- *Analysis:*
  - F has the same number of connected components as G
  - F has at most $n - 1$ edges.
- *Thm:* Can count connected components in $O(n \log n)$ space.

Extension: k-Edge Connectivity

- *Goal:* Check if all cuts are of size at least k.
- *Algorithm:* Maintain k forests $F_1, \ldots, F_k$
  - $F_1, \ldots, F_k \leftarrow \emptyset$
  - For each edge (u, v), find smallest $i \leq k$ such that u and v aren't connected in F_i,

    $$F_i \leftarrow F_i \cup \{(u, v)\}$$

    If no such i exists, ignore edge.
- *Analysis:*
  - Each F_i has at most $n - 1$ edges so total edges is $O(nk)$
  - *Lemma:* Min-Cut$(V, E) < k$ iff Min-Cut$(V, F_1 \cup \ldots \cup F_k) < k$
- *Thm:* Can check k-connectivity in $O(kn \log n)$ space.

Proof of Lemma

- Let $H = (V, F_1 \cup \ldots \cup F_k)$ and let $(S, V \setminus S)$ be an arbitrary cut.
- Since H is a subgraph:

$$|E_G(S)| \geq |E_H(S)|$$

  where $E_H(S)$ and $E_G(S)$ are the edges across the cut in H and G
- Suppose there exists $(u, v) \in E_G(S)$ but $(u, v) \notin F_1 \cup \ldots \cup F_k$. Then (u, v) must be connected in each F_i. Since F_i are disjoint,

$$|E_H(S)| \geq \min(|E_G(S)|, k)$$

Spanners

Definition

An α-spanner of graph G is a subgraph H such that for any nodes u, v,

$$d_G(u,v) \leq d_H(u,v) \leq \alpha d_G(u,v) \ .$$

where d_G and d_H are the shortest path distances in G and H respectively.

- *Algorithm:*
  - $H \leftarrow \emptyset$.
  - For each edge (u,v), if $d_H(u,v) \geq 2t$, $H \leftarrow H \cup \{(u,v)\}$
- *Analysis:*
  - Distances increase by at most a factor $2t-1$ since an edge (u,v) is only forgotten if there's already a detour of length at most $2t-1$.
  - *Lemma:* H has $O(n^{1+1/t})$ edges since all cycles have length $\geq 2t+1$.

Theorem

Can $(2t-1)$-approximate all distances using only $O(n^{1+1/t})$ space.

Proof of Lemma

Lemma

A graph H on n nodes with no cycles of length $\leq 2t$ has $O(n^{1+1/t})$ edges.

- Let $d = 2m/n$ be the average degree of H.
- Let J be the graph formed by removing nodes with degree less than $d/2$ until no such nodes remain.
- J is not empty because $< m/(d/2) = n$ nodes can be removed.
- Grow a BFS of depth t from an arbitrary node in J.
- Because a) no cycles of length less than $2t + 1$ and b) all degrees in J are at least $d/2$, number of nodes at t-th level of BFS is at least

$$(d/2 - 1)^t = (m/n - 1)^t$$

- But $(m/n - 1)^t \leq |J| \leq n$ and therefore,

$$m \leq n + n^{1+1/t} \ .$$

Sparsifier

Definition

An α-sparsifier of graph G is a weighted subgraph H such that for any cut $(S, V \setminus S)$,

$$C_G(S) \leq C_H(S) \leq \alpha C_G(S) \ .$$

where C_G and C_H is the capacity of the cut in G and H respectively.

Theorem (Batson, Spielman, Srivastava)

There exists a (non-streaming) algorithm $\mathcal{A}$ that constructs a $(1+\epsilon)$-sparsifier with only $O(n\epsilon^{-2})$ edges.

Idea for stream algorithm is to use $\mathcal{A}$ as a black box to "recursively" sparsify the graph stream.

Basic Properties of Sparsifiers

Lemma

Suppose H_1 and H_2 are α-sparsifiers of G_1 and G_2. Then $H_1 \cup H_2$ is an α-sparsifier of $G_1 \cup G_2$.

Lemma

Suppose J is an α-sparsifiers of H and H is an α-sparsifier of G. Then J is an α^2-sparsifier of G.

Stream Sparsification

- Divide length m stream into segments of length $t = O(n\epsilon^{-2})$
- Let $G_0, G_1, \ldots, G_{m/t-1}$ be graphs defined by each segment and let

$$G_0^1 = G_0 \cup G_1 \ , \ G_2^1 = G_2 \cup G_3 \ , \ \ldots \ , \ G_{m/t-2}^1 = G_{m/t-2} \cup G_{m/t-1}$$

  and for $i > 1$,

$$G_{j2^i}^i = G_{j2^i} \cup G_{j2^i+1} \cup \ldots \cup G_{j2^i+2^i-1}$$

  and note that $G_0^{\log m} = G$.
- Let $\tilde{G}_{j2^i}^i$ be a $(1+\gamma)$-sparsifier of $\tilde{G}_{j2^i}^{i-1} \cup \tilde{G}_{j2^i+2^{i-1}}^{i-1}$ and $\tilde{G}_j = G_j$.
- Hence, $\tilde{G}_0^{\log n}$ is a $(1+\gamma)^{\log m}$-sparsifier of G.
- Can compute $\tilde{G}_0^{\log n}$ in $O(n\gamma^{-2} \log m)$ space.
- Setting $\gamma = \frac{\epsilon}{\log m}$ gives $(1+\epsilon)$-sparsifier in $O(n\epsilon^{-2} \log^3 m)$ space.

Spectral Sparsification

- Given a graph G, the Laplacian matrix $L_G \in \mathbb{R}^{n \times n}$ has entries:

$$L_{ij} = \begin{cases} \deg(i) & \text{if } i = j \\ -1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

- H is an $(1+\epsilon)$ spectral sparsifier if for all

$$\forall x \in \mathbb{R}^n, \quad (1-\epsilon)x^T L_G x \leq x^T L_H x \leq (1+\epsilon)x^T L_G x$$

- Note that $x^T L_G x = \sum_{(i,j)\in E}(x_i - x_j)^2$ and hence H is a $(1+\epsilon)$ sparsifier if

$$\forall x \in \{0,1\}^n, \quad (1-\epsilon)x^T L_G x \leq x^T L_H x \leq (1+\epsilon)x^T L_G x$$

  and therefore spectral sparsification is a generalization of ("cut" or "combinatorial") sparsification.
- Spectral sparsifiers also approximate eigenvalues. These relate to expansion properties, random walks, mixing times etc.