Graph Streaming

Lecture 22

Nov 15, 2022

Graphs

- $G = (V, E)$ is an *undirected* graph
- $n = |V|$ and $m = |E|$
- Edges $e_1, e_2, \ldots, e_m$ seen as a stream, n known

Graphs

- $G = (V, E)$ is an *undirected* graph
- $n = |V|$ and $m = |E|$
- Edges $e_1, e_2, \ldots, e_m$ seen as a stream, n known

Questions:

- What graph problems can be solve with small space?
- Can we handle edge deletions?

Focus is on undirected graphs partly because directed graphs are hard to work with.

Semi-streaming Model

Most problems require us to compute a structure of size $\Theta(n)$. Lower bounds show that we require $\Omega(n)$ memory for even estimation problems

Assume we have $\Theta(n \text{polylog}(n)$ memory. About polylog per vertex of the graph

Can solve several interesting problems. Essentially reduce dense graphs to sparse graphs.

Connectivity

- Is G connected? Output a spanning tree if it is.
- Output an MST of G in the weighted case.
- Is G k-edge connected?

Basic Connectivity

- Maintain spanning forest: need only $O(n)$ edges
- When edge $e_i = (u, v)$ arrives. If u and v are in different components add e_i to spanning forest. Otherwise discard e_i.

MST

- Maintain spanning forest: need only $O(n)$ edges
- When edge $e_i = (u, v)$ arrives. If u and v are in different components add e_i to spanning forest.
- What if u and v are in same connected component?

MST

- Maintain spanning forest: need only $O(n)$ edges
- When edge $e_i = (u, v)$ arrives. If u and v are in different components add e_i to spanning forest.
- What if u and v are in same connected component? Check cycle formed by adding e_i and discard heaviest edge in cycle.

MST

- Maintain spanning forest: need only $O(n)$ edges
- When edge $e_i = (u, v)$ arrives. If u and v are in different components add e_i to spanning forest.
- What if u and v are in same connected component? Check cycle formed by adding e_i and discard heaviest edge in cycle.

Exercise: Prove that algorithm outputs an MST if G is connected.

MST

- Maintain spanning forest: need only $O(n)$ edges
- When edge $e_i = (u, v)$ arrives. If u and v are in different components add e_i to spanning forest.
- What if u and v are in same connected component? Check cycle formed by adding e_i and discard heaviest edge in cycle.

Exercise: Prove that algorithm outputs an MST if G is connected.

Note: we did not focus on time to process each edge in stream. Can use data structures to implement in $O(\log n)$ time per operation.

k-edge-connectivity

Definition

A graph $G = (V, E)$ is k-edge-connected if deleting any $k - 1$ edges still leaves a connected graph.

k-edge-connectivity

Definition

A graph $G = (V, E)$ is k-edge-connected if deleting any $k-1$ edges still leaves a connected graph.

Definition

Given a graph $G = (V, E)$ and $S \subset V$, $\delta(S)$ is the set of edges with exactly one end point in S.

k-edge-connectivity

Definition

A graph $G = (V, E)$ is k-edge-connected if deleting any $k - 1$ edges still leaves a connected graph.

Definition

Given a graph $G = (V, E)$ and $S \subset V$, $\delta(S)$ is the set of edges with exactly one end point in S.

Lemma

A graph G is k-edge connected iff $|\delta(S)| \geq k$ for all $S \subset V$.

Sparse certificates for k-edge connectivity

Observation: If G is k-edge-connected than $m \geq kn/2$. Why?

Sparse certificates for k-edge connectivity

Observation: If G is k-edge-connected than $m \geq kn/2$. Why?

Question: Suppose G is *edge-minimal* k-edge-connected graph on n nodes. What is an upper bound on the number of edges?

Sparse certificates for k-edge connectivity

Observation: If G is k-edge-connected than $m \geq kn/2$. Why?

Question: Suppose G is *edge-minimal* k-edge-connected graph on n nodes. What is an upper bound on the number of edges?

Theorem

An edge-minimal k-edge-connected graph on n nodes has at most $k(n-1)$ edges.

Sparse certificates for k-edge connectivity

Observation: If G is k-edge-connected than $m \geq kn/2$. Why?

Question: Suppose G is *edge-minimal* k-edge-connected graph on n nodes. What is an upper bound on the number of edges?

Theorem

An edge-minimal k-edge-connected graph on n nodes has at most $k(n-1)$ edges.

Theorem

Given a graph G finding the smallest 2-edge-connected subgraph is NP-Hard.

Sparse certificates for k-edge connectivity

Theorem

An edge-minimal k-edge-connected graph on n nodes has at most $k(n-1)$ edges.

Constructive proof via algorithm.

```
For i = 1 to k do
    Let F_i be a spanning forest in (V, E \ ∪_{j=1}^{i-1} F_j)

Output H = (V, F_1 ∪ F_2 ... ∪ F_k)
```

For $i = 1$ to k do
  Let F_i be a spanning forest in $(V, E \setminus \cup_{j=1}^{i-1} F_j)$

Output $H = (V, F_1 \cup F_2 \ldots \cup F_k)$

Sparse certificates for k-edge connectivity

Theorem

An edge-minimal $\boldsymbol{k}$-edge-connected graph on $\boldsymbol{n}$ nodes has at most $\boldsymbol{k}(\boldsymbol{n}-1)$ edges.

Constructive proof via algorithm.

```
For i = 1 to k do
    Let F_i be a spanning forest in (V, E \ ∪_{j=1}^{i-1} F_j)

Output H = (V, F_1 ∪ F_2 ... ∪ F_k)
```

Easy to see that $\boldsymbol{H}$ as at most $\boldsymbol{k}(\boldsymbol{n}-1)$ edges.

Lemma

$\boldsymbol{H}$ is $\boldsymbol{k}$-edge-connected if $\boldsymbol{G}$ is.

Streaming setting

```
For i = 1 to k do
    Let F_i be a spanning forest in (V, E \ ∪_{j=1}^{i-1} F_j)

Output H = (V, F_1 ∪ F_2 ... ∪ F_k)
```

For $i = 1$ to k do

  Let F_i be a spanning forest in $(V, E \setminus \cup_{j=1}^{i-1} F_j)$

Output $H = (V, F_1 \cup F_2 \ldots \cup F_k)$

Algorithm can be implemented in streaming setting. How?

k-node-connectivity

Definition

A graph $G = (V, E)$ is k-node-connected (or k-vertex-connected) if deleting any $k - 1$ nodes leaves a connected graph.

k-node-connectivity

Definition

A graph $G = (V, E)$ is k-node-connected (or k-vertex-connected) if deleting any $k - 1$ nodes leaves a connected graph.

Theorem

An edge-minimal k-edge-connected graph on n nodes has at most kn edges.

Above theorem is much more tricky than for the edge case.

See [Zelke] for references and streaming algorithm.

Part I

Cut Sparsifiers

Graph Sparsification

$G = (V, E)$ input graph and could be dense

- n is reasonable to store
- n^2 may be unreasonable to store
- edges are some times implicit and may be generated on the fly

Sparsification: Given $G = (V, E)$ create a *sparse* graph $H = (V, F)$ such that H mimics G for some property of interest

Graph Sparsification

$G = (V, E)$ input graph and could be dense

- n is reasonable to store
- n^2 may be unreasonable to store
- edges are some times implicit and may be generated on the fly

Sparsification: Given $G = (V, E)$ create a *sparse* graph $H = (V, F)$ such that H mimics G for some property of interest

- Connectivity
- Distances (spanners and variants)
- Cuts (cut sparsifiers)
- ...

Cut Sparsifier

Definition

Given an edge weighted graph $G = (V, E)$ with $w : E \to \mathbb{R}_+$ an edge weighted graph $H = (V, F)$ with $w' : F \to \mathbb{R}_+$ is an ϵ-approximate cut sparsifier if for all $S \subset V$,

$$w(\delta_G(S)) \leq w'(\delta_H(S)) \leq (1 + \epsilon) w(\delta_G(S))$$

.

Cut Sparsifier

Definition

Given an edge weighted graph $G = (V, E)$ with $w : E \to \mathbb{R}_+$ an edge weighted graph $H = (V, F)$ with $w' : F \to \mathbb{R}_+$ is an ϵ-approximate cut sparsifier if for all $S \subset V$,

$$w(\delta_G(S)) \leq w'(\delta_H(S)) \leq (1+\epsilon)w(\delta_G(S))$$

.

Very important concept and many powerful applications in graph algorithms and beyond

Fundamental results

Theorem (Benczur-Karger'00)

Given a graph $G = (V, E)$ on m edges and n nodes and any $\epsilon > 0$, one can construct in randomized $O(m \log^3 n)$ time a cut-sparsifier with $O(\frac{1}{\epsilon^2} n \log n)$ edges.

Theorem (Batson-Spielman-Srivastava'08)

Given a graph $G = (V, E)$ on m edges and n nodes and any $\epsilon > 0$, one can construct in deterministic polynomial time a cut-sparsifier with $O(\frac{1}{\epsilon^2} n)$ edges.

The preceding theorem is stronger. Gives a *spectral sparsifier*.

Fundamental results

Theorem (Benczur-Karger'00)

Given a graph $G = (V, E)$ on m edges and n nodes and any $\epsilon > 0$, one can construct in randomized $O(m \log^3 n)$ time a cut-sparsifier with $O(\frac{1}{\epsilon^2} n \log n)$ edges.

Theorem (Batson-Spielman-Srivastava'08)

Given a graph $G = (V, E)$ on m edges and n nodes and any $\epsilon > 0$, one can construct in deterministic polynomial time a cut-sparsifier with $O(\frac{1}{\epsilon^2} n)$ edges.

The preceding theorem is stronger. Gives a *spectral sparsifier*.

What is a cut-sparsifier of a complete graph K_n?

Fundamental results

Theorem (Benczur-Karger'00)

Given a graph $G = (V, E)$ on m edges and n nodes and any $\epsilon > 0$, one can construct in randomized $O(m \log^3 n)$ time a cut-sparsifier with $O(\frac{1}{\epsilon^2} n \log n)$ edges.

Theorem (Batson-Spielman-Srivastava'08)

Given a graph $G = (V, E)$ on m edges and n nodes and any $\epsilon > 0$, one can construct in deterministic polynomial time a cut-sparsifier with $O(\frac{1}{\epsilon^2} n)$ edges.

The preceding theorem is stronger. Gives a *spectral sparsifier*.

What is a cut-sparsifier of a complete graph K_n? An expander graph!

Cut sparsifiers in streaming

Question: Can we create a cut-sparsifier on the fly in roughly $O(n\,\text{polylog}(n))$ space as edges come by?

Can use cut-sparsifier algorithms as a black box.

Merge and Reduce

Observation (Merge): If $H_1 = (V, F_1)$ is a α-approximate sparsifier for $G_1 = (V, E_1)$ and $H_2 = (V, F_2)$ is a α-approximate cut-sparsifier for $G_2 = (V, E_2)$ then $H_1 \cup H_2 = (V, F_1 \cup F_2)$ is a α-approximate cut-sparsifier for $G_1 \cup G_2 = (V, E_1 \cup E_2)$.

Merge and Reduce

Observation (Merge): If $H_1 = (V, F_1)$ is a α-approximate sparsifier for $G_1 = (V, E_1)$ and $H_2 = (V, F_2)$ is a α-approximate cut-sparsifier for $G_2 = (V, E_2)$ then $H_1 \cup H_2 = (V, F_1 \cup F_2)$ is a α-approximate cut-sparsifier for $G_1 \cup G_2 = (V, E_1 \cup E_2)$.

Observation (Reduce): If $H = (V, F)$ is a α-approximate cut sparsifier for $G = (V, E_1)$ and $H' = (V, F')$ is a β-approximate cut-sparsifier for H then H' is a $(\alpha\beta)$-approximate cut-sparsifier for G.

Cut sparsifiers in streaming

Question: Can we create a cut-sparsifier on the fly in roughly $O(n\text{polylog}(n))$ space as edges come by?

Can use cut-sparsifier algorithms as a black box.

Merge and Reduce via a binary tree approach over the m edges in the stream. Seen this approach twice already: range queries in CountMin sketch and quantile summaries.

Cut sparsifiers in streaming

- Split stream of m edges into k graphs of m/k edges each. Let $G_1, G_2, \ldots, G_k$ be the k graphs. Assume for simplicity that k is a power of 2.
- Imagine a binary tree with $G_1, \ldots, G_k$ as leaves
- Build a sparsifier bottom up. At each internal node merge the sparisfiers and reduce with approximation α

Cut sparsifiers in streaming

- Split stream of m edges into k graphs of m/k edges each. Let $G_1, G_2, \ldots, G_k$ be the k graphs. Assume for simplicity that k is a power of 2.
- Imagine a binary tree with $G_1, \ldots, G_k$ as leaves
- Build a sparsifier bottom up. At each internal node merge the sparisfiers and reduce with approximation α

Questions:

- What is α to ensure that final sparsifier is ϵ-approximate?
- How much space needed in streaming setting?

Cut sparsifiers in streaming

- What is α to ensure that final sparsifier is ϵ-approximate?
- How much space needed in streaming setting?

Collect $N = \Theta(n \log^c n)$ edges before processing since we can afford roughly that much space. So each leaf corresponds to a graph with N edges

Depth of tree is $\leq \log(m/N) \leq \log n$. Due to reduce operations final approximation is $(1+\alpha)^d$. Choose α such that $(1+\alpha)^d \leq (1+\epsilon)$ which implies $\alpha \simeq \epsilon/(ed) \simeq \epsilon/(e \log n)$

Cut sparsifiers in streaming

p Collect $N = \Theta(n \log^c n)$ edges before processing since we can afford roughly that much space. So each leaf corresponds to a graph with N edges

Depth of tree is $\leq \log(m/N) \leq \log n$. Due to reduce operations final approximation is $(1+\alpha)^d$. Choose α such that $(1+\alpha)^d \leq (1+\epsilon)$ which implies $\alpha \simeq \epsilon/(ed) \simeq \epsilon/(e \log n)$

Cut sparsifiers in streaming

p Collect $N = \Theta(n \log^c n)$ edges before processing since we can afford roughly that much space. So each leaf corresponds to a graph with N edges

Depth of tree is $\leq \log(m/N) \leq \log n$. Due to reduce operations final approximation is $(1+\alpha)^d$. Choose α such that $(1+\alpha)^d \leq (1+\epsilon)$ which implies $\alpha \simeq \epsilon/(ed) \simeq \epsilon/(e \log n)$

Space analysis: Sparsifier size with $\alpha = \epsilon/\log n$ is $O(n \log^2 n/\epsilon^2)$ (if one uses BSS sparsifier, otherwise another log factor for Benczur-Karger sparsifier).

Need another $\log n$ factor to store sparsfiers at $\log n$ levels for streaming. So total space is $O(N + n \log^3 n/\epsilon^2)$. Hence choose $N = O(n \log^3 n)$.

Spectral Sparsifier

Spectral sparisifer is a stronger notion than cut sparsifier. Comes from linear algebraic view of graphs.

Definition

The Laplacian L_G of a n-vertex undirected graph $G = (V, E)$ with non-negative edge-weights $w : E \to \mathbb{R}_+$ is a $n \times n$ symmetric diagonally dominant matrix where (i) $L_G(ii) = deg(i)$ for each $i \in [n]$ and $L_G(ij) = L_G(ij) = -w(ij)$ if $ij \in E$ and 0 otherwise.

- L_G is a positive semi-definite matrix and has rank $< n$
- Since L_G is psd it has non-negative real eigenvalues and $x^T L_G x \geq 0$ for all $x \in R^n$
- $x^T L_G x = \sum_{ij \in E} w(ij)(x_i - x_j)^2$
- Suppose $x = 1_S$ the indicator of a set $S \subseteq V$ then $x^T L_G x = w(\delta(S))$

Spectral Sparsifier

Spectral sparisifer is a stronger notion than cut sparsifier. Comes from linear algebraic view of graphs.

Definition

Given $G = (V, E)$ with edge weights $w : E \to \mathbb{R}_+$ a weighted graph $H = (V, E_H)$ with $w' : E_H \to \mathbb{R}_+$ is a $(1+\epsilon)$-spectral sparsifier for G if

$$x^T L_G x \leq x^T L_H x \leq (1+\epsilon) x^T L_G x$$

for all $x \in R^n$. Equivalently, $L_G \preceq L_H \preceq (1+\epsilon) L_G$.

Spectral Sparsifier

Spectral sparisifer is a stronger notion than cut sparsifier. Comes from linear algebraic view of graphs.

Definition

Given $G = (V, E)$ with edge weights $w : E \rightarrow \mathbb{R}_+$ a weighted graph $H = (V, E_H)$ with $w' : E_H \rightarrow \mathbb{R}_+$ is a $(1+\epsilon)$-spectral sparsifier for G if

$$x^T L_G x \leq x^T L_H x \leq (1+\epsilon) x^T L_G x$$

for all $x \in R^n$. Equivalently, $L_G \preceq L_H \preceq (1+\epsilon) L_G$.

Observation: An α-approximate spectral sparisfier is an α-approximate cut sparsifier but converse is not necessarily true.

Spectral Sparsifier

Theorem (Batson-Spielman-Srivastava'08)

Given a graph $G = (V, E)$ on m edges and n nodes and any $\epsilon > 0$, one can construct in deterministic polynomial time a spectral-sparsifier with $O(\frac{1}{\epsilon^2} n)$ edges.

Reduce and Merge framework extends easily for spectral sparsifiers as well so one can compute spectral sparisfiers in $O(n \text{poly}(\log n))$ space in the streaming setting.