Lecture 6: Graph Sparsification

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

May 21, 2025

Overview

- Introduction
  - Why Sparsify?
  - Warm-up Problem
- Main Problem
  - Graph Sparsification
- Acknowledgements

Why do we sparsify?

Often times graph algorithms for graphs $G(V, E)$ have runtimes which depend on $|E|$. If the graph is dense, i.e. $|E| = \omega(n^{1+\gamma})$, for $\gamma > 0$, then this may be *too slow*.

We want graph that has *nearly-linear* number of edges $O(n \cdot \text{poly}\log n)$

- Settle for *approximate answers*

Why do we sparsify?

Often times graph algorithms for graphs $G(V, E)$ have runtimes which depend on $|E|$. If the graph is dense, i.e. $|E| = \omega(n^{1+\gamma})$, for $\gamma > 0$, then this may be *too slow*.

We want graph that has *nearly-linear* number of edges $O(n \cdot \text{poly} \log n)$

- Settle for *approximate answers*
- Used as primitives in many other algorithms (for instance, max-flow, sparsest cut, etc.)

Why do we sparsify?

Often times graph algorithms for graphs $G(V, E)$ have runtimes which depend on $|E|$. If the graph is dense, i.e. $|E| = \omega(n^{1+\gamma})$, for $\gamma > 0$, then this may be *too slow*.

We want graph that has *nearly-linear* number of edges $O(n \cdot \text{poly}\log n)$

- Settle for *approximate answers*
- Used as primitives in many other algorithms (for instance, max-flow, sparsest cut, etc.)
- Applications in network connectivity

Graph Cuts

Definition (Graph Cut)

If $G(V, E, w)$ is a weighted graph, a *cut* is a partition of the vertices into two non-empty sets $V = S \sqcup \overline{S}$. The *value* of a cut is the quantity

$$w(S, \overline{S}) := \sum_{e \in E(S, \overline{S})} w_e.$$

Contraction of Edges

Definition (Edge Contraction)

Let $G(V, E)$ be a graph. If $e = \{u, v\} \in E$ is an edge of G, then the *contraction* of e is a new graph $H(V \cup \{z\} \setminus \{u, v\}, F)$ where we replace the vertices u, v by *one* vertex z, and any edge $\{u, x\} =: f \in E \setminus \{e\}$ is replaced by $\{z, x\} \in F$.

Randomized Minimum Cut

- **Input:** undirected unweighted graph $G(V, E)$
- **Output:** minimum cut $(S, \overline{S})$, with high probability

Randomized Minimum Cut

- **Input:** undirected unweighted graph $G(V, E)$
- **Output:** minimum cut $(S, \overline{S})$, with high probability
- While there are more than 2 vertices in the graph:
  - Pick uniformly random edge and contract it

Randomized Minimum Cut

- **Input:** undirected unweighted graph $G(V, E)$
- **Output:** minimum cut $(S, \overline{S})$, with high probability
- While there are more than 2 vertices in the graph:
  - Pick uniformly random edge and contract it
- Output the two subsets encoded by the two remaining vertices.

Analysis

Why does this work?

Intuition: picking a random edge uniformly at random "favours" *small cuts* (i.e. preserves them) with higher probability.

Analysis

Why does this work?

Intuition: picking a random edge uniformly at random "favours" *small cuts* (i.e. preserves them) with higher probability.

Remark

The value of the minimum cut does note decrease after contraction.

Analysis

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$, where $n = |V|$.

Analysis

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$*, where* $n = |V|$.

- Let $(S, \overline{S})$ be a minimum cut, and $c := |E(S, \overline{S})|$. If we never contract an edge from $E(S, \overline{S})$, the algorithm succeeds.

Analysis

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$, where $n = |V|$.

- Let $(S, \overline{S})$ be a minimum cut, and $c := |E(S, \overline{S})|$. If we never contract an edge from $E(S, \overline{S})$, the algorithm succeeds.
- Probability that an edge from $E(S, \overline{S})$ is contracted in the i^{th} iteration (conditioned on cut still alive)

Analysis

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$, where $n = |V|$.

- Let $(S, \overline{S})$ be a minimum cut, and $c := |E(S, \overline{S})|$. If we never contract an edge from $E(S, \overline{S})$, the algorithm succeeds.
- Probability that an edge from $E(S, \overline{S})$ is contracted in the i^{th} iteration (conditioned on cut still alive)
  - Each vertex is a cut, so each vertex has degree $\geq c \Rightarrow$

$$\geq \frac{(n-i+1)\cdot c}{2} \quad \text{edges remain.}$$

Analysis

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$, where $n = |V|$.

- Let $(S, \overline{S})$ be a minimum cut, and $c := |E(S, \overline{S})|$. If we never contract an edge from $E(S, \overline{S})$, the algorithm succeeds.
- Probability that an edge from $E(S, \overline{S})$ is contracted in the i^{th} iteration (conditioned on cut still alive)
  - Each vertex is a cut, so each vertex has degree $\geq c \Rightarrow$

$$\geq \frac{(n-i+1) \cdot c}{2} \quad \text{edges remain.}$$

  - Contracting random edge, probability we kill cut $(S, \overline{S})$ is

$$= |E(S, \overline{S})| \cdot \frac{1}{(\#\text{ edges})} \leq c \cdot \frac{2}{(n-i+1) \cdot c} = \frac{2}{n-i+1}$$

Analysis

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$, where $n = |V|$.

- Let $(S, \overline{S})$ be a minimum cut, and $c := |E(S, \overline{S})|$. If we never contract an edge from $E(S, \overline{S})$, the algorithm succeeds.
- Probability that an edge from $E(S, \overline{S})$ is contracted in the i^{th} iteration (conditioned on cut still alive)
  - Each vertex is a cut, so each vertex has degree $\geq c \Rightarrow$

$$\geq \frac{(n-i+1)\cdot c}{2} \quad \text{edges remain.}$$

  - Contracting random edge, probability we kill cut $(S, \overline{S})$ is

$$= |E(S, \overline{S})| \cdot \frac{1}{(\#\text{ edges})} \leq c \cdot \frac{2}{(n-i+1)\cdot c} = \frac{2}{n-i+1}$$

  - $\Pr[(S, \overline{S}) \text{ survives}] \geq (1-2/n)\cdot(1-3/n)\cdots(1-2/3) = 2/n(n-1)$

Hmmmmm, this is not with high probability...

- To improve success probability, repeat this randomized procedure t times (for which t?)
- If we repeat for t times, failure probability is

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^t$$

Hmmmmm, this is not with high probability...

- To improve success probability, repeat this randomized procedure t times (for which t?)
- If we repeat for t times, failure probability is

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^t$$

- setting $t = 2n(n-1)$ then

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^t \leq \exp\left(-\frac{2t}{n(n-1)}\right) = e^{-4}$$

Hmmmmm, this is not with high probability...

- To improve success probability, repeat this randomized procedure t times (for which t?)
- If we repeat for t times, failure probability is

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^t$$

- setting $t = 2n(n-1)$ then

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^t \leq \exp\left(-\frac{2t}{n(n-1)}\right) = e^{-4}$$

- **Running time:** One execution implemented in $O(n^2)$, so t executions in time $O(n^2 t) = O(n^4)$.

Hmmmmm, this is not with high probability...

- To improve success probability, repeat this randomized procedure t times (for which t?)
- If we repeat for t times, failure probability is

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^t$$

- setting $t = 2n(n-1)$ then

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^t \leq \exp\left(-\frac{2t}{n(n-1)}\right) = e^{-4}$$

- **Running time:** One execution implemented in $O(n^2)$, so t executions in time $O(n^2 t) = O(n^4)$.
- You will work on some running time improvements in your homework!

Combinatorial Application

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$, *where* $n = |V|$.

Combinatorial Application

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$*, where* $n = |V|$.

Corollary

There are at most $O(n^2)$ *minimum cuts in an undirected graph.*

Combinatorial Application

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$, *where* $n = |V|$.

Corollary

There are at most $O(n^2)$ *minimum cuts in an undirected graph.*

- Each minimum cut survives with probability $\Omega(1/n^2)$
- Events that two different cuts survive are disjoint

Combinatorial Application

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$, where $n = |V|$.

Corollary

There are at most $O(n^2)$ minimum cuts in an undirected graph.

- Each minimum cut survives with probability $\Omega(1/n^2)$
- Events that two different cuts survive are disjoint
- Non-trivial statement to prove using other arguments!

Combinatorial Application

Theorem (Karger)

The probability that the algorithm outputs a minimum cut is at least $2/n(n-1)$, where $n = |V|$.

Corollary

There are at most $O(n^2)$ minimum cuts in an undirected graph.

- Each minimum cut survives with probability $\Omega(1/n^2)$
- Events that two different cuts survive are disjoint
- Non-trivial statement to prove using other arguments!

This is all good, but we haven't "sparsified" anything so far!

- Introduction
  - Why Sparsify?
  - Warm-up Problem
- Main Problem
  - Graph Sparsification
- Acknowledgements

Graph Sparsification

Definition (Weight of a cut)

Let $G(V, E, w)$ be undirected weighted graph. For any cut $(S, \overline{S})$, let the weight of $(S, \overline{S})$ be

$$w(S, \overline{S}) := \sum_{e \in E(S, \overline{S})} w(e).$$

Graph Sparsification

Definition (Weight of a cut)

Let $G(V, E, w)$ be undirected weighted graph. For any cut $(S, \overline{S})$, let the weight of $(S, \overline{S})$ be

$$w(S, \overline{S}) := \sum_{e \in E(S, \overline{S})} w(e).$$

Definition (Sparse Graph)

We say that a graph $G(V, E)$ is *sparse* if $|E| = \tilde{O}(|V|)$.

Graph Sparsification

Definition (Weight of a cut)

Let $G(V, E, w)$ be undirected weighted graph. For any cut $(S, \overline{S})$, let the weight of $(S, \overline{S})$ be

$$w(S, \overline{S}) := \sum_{e \in E(S, \overline{S})} w(e).$$

Definition (Sparse Graph)

We say that a graph $G(V, E)$ is *sparse* if $|E| = \tilde{O}(|V|)$.

Question

How to make a graph sparse (nearly linear # edges) while approximating the value of every cut of a graph?

Graph Sparsification

- **Input:** graph $G(V, E, w_G)$, $\varepsilon > 0$.

$$n = |V|, \;\; m = |E|.$$

- **Output:** graph $H(V, F, w_H)$ such that *for every cut* $(S, \overline{S})$, we have

$$(1 - \varepsilon) \cdot w_G(S, \overline{S}) \leq w_H(S, \overline{S}) \leq (1 + \varepsilon) \cdot w_G(S, \overline{S})$$

Graph Sparsification

- **Input:** graph $G(V, E, w_G)$, $\varepsilon > 0$.

$$n = |V|, \;\; m = |E|.$$

- **Output:** graph $H(V, F, w_H)$ such that *for every cut $(S, \overline{S})$*, we have

$$(1-\varepsilon) \cdot w_G(S, \overline{S}) \leq w_H(S, \overline{S}) \leq (1+\varepsilon) \cdot w_G(S, \overline{S})$$

- *Assumption (for this class):* the input graph $G(V, E)$ is
  1. *unweighted*
  2. has *minimum cut value* $\Omega(\log n)$ — i.e., a large-ish cut

Graph Sparsification

- **Input:** graph $G(V, E, w_G)$, $\varepsilon > 0$.

$$n = |V|, \;\; m = |E|.$$

- **Output:** graph $H(V, F, w_H)$ such that *for every cut* $(S, \overline{S})$, we have

$$(1-\varepsilon) \cdot w_G(S, \overline{S}) \leq w_H(S, \overline{S}) \leq (1+\varepsilon) \cdot w_G(S, \overline{S})$$

- *Assumption (for this class):* the input graph $G(V, E)$ is
  1. *unweighted*
  2. has *minimum cut value* $\Omega(\log n)$ i.e., a large-ish cut

Algorithm:

- Let $p \in (0, 1)$ be a parameter.
- For each edge $e \in E(G)$, with probability p, make e an edge of H with weight $w_H(e) = 1/p$.

Graph Sparsification

Idea:

- Set p to get correct expected value for both $\#$ edges in H and the value of each cut $(S, \overline{S})$ in H.

Graph Sparsification

Idea:

- Set p to get correct expected value for both $\#$ edges in H and the value of each cut $(S, \overline{S})$ in H.
- After that, need to prove concentration around expected values *for all cuts simultaneously*!

Graph Sparsification

Idea:

- Set p to get correct expected value for both $\#$ edges in H and the value of each cut $(S, \overline{S})$ in H.
- After that, need to prove concentration around expected values *for all cuts simultaneously*!
- Use Chernoff-Hoeffding and assumption that min-cut value is large.

Graph Sparsification

Idea:

- Set p to get correct expected value for both # edges in H and the value of each cut $(S, \overline{S})$ in H.
- After that, need to prove concentration around expected values *for all cuts simultaneously*!
- Use Chernoff-Hoeffding and assumption that min-cut value is large.

Theorem ([Karger, 1993])

Let c be the value of the min-cut of G. Set

$$p = \frac{15 \ln n}{\varepsilon^2 \cdot c}.$$

Graph H given by algorithm from previous slide **approximates all cuts of** *G and has $O(p \cdot |E|)$ edges with probability $\geq 1 - 4/n$.*

Graph Sparsification

- Take a cut $(S, \overline{S})$. Suppose $k := w_G(S, \overline{S})$. Let

  $$X_e = \begin{cases} 1, & \text{if edge } e \textbf{ included in } H \\ 0, & \text{otherwise} \end{cases}$$

Graph Sparsification

- Take a cut $(S, \overline{S})$. Suppose $k := w_G(S, \overline{S})$. Let
$$X_e = \begin{cases} 1, & \text{if edge } e \textbf{ included in } H \\ 0, & \text{otherwise} \end{cases}$$
-
$$\mathbb{E}[|F|] = \sum_{e \in E} \mathbb{E}[X_e] = \sum_{e \in E} (p \cdot 1 + (1-p) \cdot 0) = p \cdot |E|$$

Graph Sparsification

- Take a cut $(S, \overline{S})$. Suppose $k := w_G(S, \overline{S})$. Let
$$X_e = \begin{cases} 1, & \text{if edge } e \textbf{ included in } H \\ 0, & \text{otherwise} \end{cases}$$
- $$\mathbb{E}[|F|] = \sum_{e \in E} \mathbb{E}[X_e] = \sum_{e \in E} (p \cdot 1 + (1-p) \cdot 0) = p \cdot |E|$$
- $$\mathbb{E}[w_H(S, \overline{S})] = \sum_{e \in E(S,\overline{S})} \mathbb{E}[w_H(e)] = \sum_{e \in E(S,\overline{S})} (p \cdot \frac{1}{p} + (1-p) \cdot 0)$$
$$= |E(S, \overline{S})| = k = w_G(S, \overline{S})$$

Graph Sparsification - Concentration

- Take a cut $(S, \overline{S})$. Suppose $k := w_G(S, \overline{S})$. Let $w_e = \begin{cases} 1/p, & \text{if edge } e \textbf{ included in } H \\ 0, & \text{otherwise} \end{cases}$

Graph Sparsification - Concentration

- Take a cut $(S, \overline{S})$. Suppose $k := w_G(S, \overline{S})$. Let
  $$w_e = \begin{cases} 1/p, & \text{if edge } e \textbf{ included in } H \\ 0, & \text{otherwise} \end{cases}$$
- $w_H(S, \overline{S})$ is a sum of independent random variables w_e

Graph Sparsification - Concentration

- Take a cut $(S, \overline{S})$. Suppose $k := w_G(S, \overline{S})$. Let
$$w_e = \begin{cases} 1/p, & \text{if edge } e \textbf{ included in } H \\ 0, & \text{otherwise} \end{cases}$$
- $w_H(S, \overline{S})$ is a sum of independent random variables w_e
- Chernoff Bound:

$$\Pr[|w_H(S, \overline{S}) - k| \geq \varepsilon \cdot k] \leq 2 \exp\left(-\frac{\varepsilon^2 kp}{3}\right) = 2n^{-5k/c}$$

Graph Sparsification - Concentration

- Take a cut $(S, \overline{S})$. Suppose $k := w_G(S, \overline{S})$. Let
  $$w_e = \begin{cases} 1/p, & \text{if edge } e \textbf{ included in } H \\ 0, & \text{otherwise} \end{cases}$$
- $w_H(S, \overline{S})$ is a sum of independent random variables w_e
- Chernoff Bound:
  $$\Pr[|w_H(S, \overline{S}) - k| \geq \varepsilon \cdot k] \leq 2 \exp\left(-\frac{\varepsilon^2 kp}{3}\right) = 2n^{-5k/c}$$
- Note that $k \geq c$, as c is the weight of the minimum cut

Graph Sparsification - Concentration

- Take a cut $(S, \overline{S})$. Suppose $k := w_G(S, \overline{S})$. Let
$$w_e = \begin{cases} 1/p, & \text{if edge } e \textbf{ included in } H \\ 0, & \text{otherwise} \end{cases}$$
- $w_H(S, \overline{S})$ is a sum of independent random variables w_e
- Chernoff Bound:
$$\Pr[|w_H(S, \overline{S}) - k| \geq \varepsilon \cdot k] \leq 2 \exp\left(-\frac{\varepsilon^2 kp}{3}\right) = 2n^{-5k/c}$$
- Note that $k \geq c$, as c is the weight of the minimum cut
- This is probability of *single cut* deviating from its mean... How can we handle the *exponentially many* cuts in the graph?

Graph Sparsification - Concentration

- Take a cut $(S, \overline{S})$. Suppose $k := w_G(S, \overline{S})$. Let
  $$w_e = \begin{cases} 1/p, & \text{if edge } e \textbf{ included in } H \\ 0, & \text{otherwise} \end{cases}$$
- $w_H(S, \overline{S})$ is a sum of independent random variables w_e
- Chernoff Bound:

$$\Pr[|w_H(S, \overline{S}) - k| \geq \varepsilon \cdot k] \leq 2 \exp\left(-\frac{\varepsilon^2 kp}{3}\right) = 2n^{-5k/c}$$

- Note that $k \geq c$, as c is the weight of the minimum cut
- This is probability of *single cut* deviating from its mean... How can we handle the *exponentially many* cuts in the graph?
- **Observation:** probability that large cut violated is *much smaller*, and there are *not many small cuts*!

Graph Sparsification - Concentration

- Take a cut $(S, \overline{S})$. Suppose $k := w_G(S, \overline{S})$. Let
  $w_e = \begin{cases} 1/p, & \text{if edge } e \textbf{ included in } H \\ 0, & \text{otherwise} \end{cases}$
- $w_H(S, \overline{S})$ is a sum of independent random variables w_e
- Chernoff Bound:

$$\Pr[|w_H(S, \overline{S}) - k| \geq \varepsilon \cdot k] \leq 2 \exp\left(-\frac{\varepsilon^2 kp}{3}\right) = 2n^{-5k/c}$$

- Note that $k \geq c$, as c is the weight of the minimum cut
- This is probability of *single cut* deviating from its mean... How can we handle the *exponentially many* cuts in the graph?
- **Observation:** probability that large cut violated is *much smaller*, and there are *not many small cuts*!
- So we can do a clever union bound!

Number of Cuts Lemma

Lemma (Number of small cuts)

If c is the size of the minimum cut in our graph, then the number of cuts with at most $\alpha \cdot c$ edges for $\alpha \geq 1$ is at most $n^{2\alpha}$.

Number of Cuts Lemma

Lemma (Number of small cuts)

If c is the size of the minimum cut in our graph, then the number of cuts with at most $\alpha \cdot c$ edges for $\alpha \geq 1$ is at most $n^{2\alpha}$.

Practice problem: generalize our earlier proof on the # minimum cuts to this case.

Union Bound on # Cuts

$$\Pr[\text{some cut is violated}] \leq \sum_{S \subseteq V} \Pr[(S, \overline{S}) \text{ is violated}]$$

Union Bound on # Cuts

$$\Pr[\text{some cut is violated}] \leq \sum_{S \subseteq V} \Pr[(S, \overline{S}) \text{ is violated}]$$

$$\leq \sum_{\alpha = 1,2,4,8,\ldots} \sum_{\substack{S \subseteq V \\ \alpha c \leq |w_G(S, \overline{S})| \leq 2 \cdot \alpha c}} \Pr[(S, \overline{S}) \text{ is violated}]$$

Union Bound on # Cuts

$$\Pr[\text{some cut is violated}] \leq \sum_{S \subseteq V} \Pr[(S, \overline{S}) \text{ is violated}]$$

$$\leq \sum_{\alpha=1,2,4,8,\ldots} \quad \sum_{\substack{S \subseteq V \\ \alpha c \leq |w_G(S,\overline{S})| \leq 2 \cdot \alpha c}} \Pr[(S, \overline{S}) \text{ is violated}]$$

$$\leq \sum_{\alpha=1,2,4,8,\ldots} n^{4\alpha} \cdot \Pr[(S, \overline{S}) \text{ is violated} \mid \alpha c \leq |w_G(S, \overline{S})| \leq 2 \cdot \alpha c]$$

Union Bound on # Cuts

$$
\begin{aligned}
\Pr[\text{some cut is violated}] &\le \sum_{S \subseteq V} \Pr[(S, \overline{S}) \text{ is violated}] \\
&\le \sum_{\alpha = 1,2,4,8,\ldots} \sum_{\substack{S \subseteq V \\ \alpha c \le |w_G(S,\overline{S})| \le 2\cdot\alpha c}} \Pr[(S, \overline{S}) \text{ is violated}] \\
&\le \sum_{\alpha = 1,2,4,8,\ldots} n^{4\alpha} \cdot \Pr[(S, \overline{S}) \text{ is violated} \mid \alpha c \le |w_G(S,\overline{S})| \le 2 \cdot \alpha c] \\
&\le \sum_{\alpha = 1,2,4,8,\ldots} n^{4\alpha} \cdot 2n^{-5\alpha c/c} \\
&= \sum_{\alpha = 1,2,4,8,\ldots} n^{-\alpha} \le 4/n
\end{aligned}
$$

Union Bound on # Cuts

$$\begin{aligned}
\Pr[\text{some cut is violated}] &\leq \sum_{S \subseteq V} \Pr[(S, \overline{S}) \text{ is violated}] \\
&\leq \sum_{\alpha=1,2,4,8,\ldots} \sum_{\substack{S \subseteq V \\ \alpha c \leq |w_G(S,\overline{S})| \leq 2 \cdot \alpha c}} \Pr[(S, \overline{S}) \text{ is violated}] \\
&\leq \sum_{\alpha=1,2,4,8,\ldots} n^{4\alpha} \cdot \Pr[(S, \overline{S}) \text{ is violated} \mid \alpha c \leq |w_G(S, \overline{S})| \leq 2 \cdot \alpha c] \\
&\leq \sum_{\alpha=1,2,4,8,\ldots} n^{4\alpha} \cdot 2n^{-5\alpha c/c} \\
&= \sum_{\alpha=1,2,4,8,\ldots} n^{-\alpha} \leq 4/n
\end{aligned}$$

Another application of Chernoff gives us that H has the right number of edges $|F| \approx p \cdot |E|$ (i.e., sparse)

How to remove the assumption?

- Assumed that the graph has large min-cut value ($c = \Omega(\log n)$).

How to remove the assumption?

- Assumed that the graph has large min-cut value ($c = \Omega(\log n)$).
- Without min-cut assumption, uniform sampling won't work

How to remove the assumption?

- Assumed that the graph has large min-cut value ($c = \Omega(\log n)$).
- Without min-cut assumption, uniform sampling won't work
- **[Benczur, Karger 1996]:** without minimum cut assumption, just sample non-uniformly in clever way!

How to remove the assumption?

- Assumed that the graph has large min-cut value ($c = \Omega(\log n)$).
- Without min-cut assumption, uniform sampling won't work
- **[Benczur, Karger 1996]:** without minimum cut assumption, just sample non-uniformly in clever way!
- Sample edge with probability proportional to "connectivity" of two endpoints (i.e., how relevant is the edge between them?)

How to remove the assumption?

- Assumed that the graph has large min-cut value ($c = \Omega(\log n)$).
- Without min-cut assumption, uniform sampling won't work
- **[Benczur, Karger 1996]:** without minimum cut assumption, just sample non-uniformly in clever way!
- Sample edge with probability proportional to "connectivity" of two endpoints (i.e., how relevant is the edge between them?)
- **Strong Connectivity:** a k-strong component is a maximal induced subgraph that is k-edge-connected. For each edge e, let s_e be the maximum value k such that there exists a k-strong component containing e.

How to remove the assumption?

- Assumed that the graph has large min-cut value ($c = \Omega(\log n)$).
- Without min-cut assumption, uniform sampling won't work
- **[Benczur, Karger 1996]:** without minimum cut assumption, just sample non-uniformly in clever way!
- Sample edge with probability proportional to "connectivity" of two endpoints (i.e., how relevant is the edge between them?)
- **Strong Connectivity:** a k-strong component is a maximal induced subgraph that is k-edge-connected. For each edge e, let s_e be the maximum value k such that there exists a k-strong component containing e.
- Sample edge e with probability $p_e = \Theta\left(\frac{\log n}{\varepsilon^2 \cdot s_e}\right)$ and weight $1/p_e$.

Acknowledgement

- Lecture based largely on Prof. Lau's notes.
- See Prof. Lau's Lecture 1 notes at
  `https://cs.uwaterloo.ca/~lapchi/cs466/notes/L01.pdf`
- See Prof. Lau's Lecture 3 notes at
  `https://cs.uwaterloo.ca/~lapchi/cs466/notes/L03.pdf`
- See Mohsen's notes (in the references section of the course webpage) for the general Benczur-Karger algorithm.

References I

Motwani, Rajeev and Raghavan, Prabhakar (2007)
Randomized Algorithms

Mitzenmacher, Michael, and Eli Upfal (2017)
Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis.
Cambridge university press, 2017.

Karger, David (1993)
Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm.
SODA 93, 21–30.

Benczur, Andras and Karger, David (1996)
Approximating st minimum cuts in $\tilde{O}(n^2)$ time.
Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, 47 – 55.