"algorithms for Big Data"

Lecture 3: Streaming and Sketching

Slides at http://grigory.us/big-data-csclub.html

Grigory Yaroslavtsev
(Indiana University, Bloomington)
http://grigory.us

Count-Min Sketch

- https://sites.google.com/site/countminsketch/
- Stream: $\boldsymbol{m}$ elements from universe $[\boldsymbol{n}] = \{1, 2, \dots, \boldsymbol{n}\}$, e.g.

  $\langle x_1, x_2, \dots, x_{\boldsymbol{m}} \rangle = \langle 5, 8, 1, 1, 1, 4, 3, 5, \dots, 10 \rangle$
- f_i = frequency of i in the stream = # of occurrences of value i, $f = \langle f_1, \dots, f_{\boldsymbol{n}} \rangle$
- Problems:
  - Point Query: For $i \in [n]$ estimate f_i
  - Range Query: For $i, j \in [n]$ estimate $f_i + \dots + f_j$
  - Quantile Query: For $\phi \in [0,1]$ find j with $f_1 + \dots + f_j \approx \phi m$
  - Heavy Hitters: For $\phi \in [0,1]$ find all i with $f_i \geq \phi m$

Count-Min Sketch: Construction

- Let $H_1, \dots, H_d : [n] \to [w]$ be 2-wise independent hash functions
- We maintain $d \cdot w$ counters with values:

  $c_{i,j} =$ # elements e in the stream with $H_i(e) = j$
- For every x the value $c_{i,H_i(x)} \geq f_x$ and so:

$$f_x \leq \widetilde{f_x} = \min(c_{1,H_1(x)}, \dots, c_{d,H_d(x)})$$

- If $w = \frac{2}{\epsilon}$ and $d = \log_2 \frac{1}{\delta}$ then:

$$\Pr\left[f_x \leq \widetilde{f_x} \leq f_x + \epsilon m\right] \geq 1 - \delta.$$

Count-Min Sketch: Analysis

- Define random variables $\boldsymbol{Z}_1 \dots, \boldsymbol{Z}_d$ such that $c_{i,H_i(x)} = f_x + \boldsymbol{Z}_i$

$$\boldsymbol{Z}_i = \sum_{y \neq x, H_i(y) = H_i(x)} f_y$$

- Define $\boldsymbol{X}_{i,y} = 1$ if $H_i(y) = H_i(x)$ and 0 otherwise:

$$\boldsymbol{Z}_i = \sum_{y \neq x} f_y \boldsymbol{X}_{i,y}$$

- By 2-wise independence:

$$\mathbb{E}[\boldsymbol{Z}_i] = \sum_{y \neq x} f_y \, \mathbb{E}\left[\boldsymbol{X}_{i,y}\right] = \sum_{y \neq x} f_y \Pr[H_i(y) = H_i(x)] \leq \frac{m}{w}$$

- By Markov inequality,

$$\Pr[\boldsymbol{Z}_i \geq \epsilon m] \leq \frac{1}{w \, \epsilon} = \frac{1}{2}$$

Count-Min Sketch: Analysis

- All Z_i are independent

$$\Pr[Z_i \geq \epsilon m \quad for\ all \quad 1 \leq i \leq d\,] \leq \left(\frac{1}{2}\right)^d = \delta$$

- With prob. $1\ -\delta$ there exists j such that $Z_j \leq \epsilon m$

$$\widetilde{f_x} = \min\left(c_{1,H_1(x)}, \dots, c_{d,H_d(x)}\right) =$$
$$= \min(f_x, +Z_1\ \dots, f_x + Z_d) \leq f_x + \epsilon m$$

- CountMin estimates values f_x up to $\pm\epsilon m$ with total memory $O\left(\frac{\log m \log\frac{1}{\delta}}{\epsilon}\right)$`

Dyadic Intervals

- Define $\log n$ partitions of $[n]$:

$I_0 = \{1,2,3, \dots n\}$
$I_1 = \left\{\{1,2\}, \{3,4\}, \dots, \{n-1, n\}\right\}$
$I_2 = \{\{1,2,3,4\}, \{5,6,7,8\}, \dots, \{n-3, n-2, n-1, n\}\}$
…
$\mathrm{I}_{\mathrm{log\ n}} = \{\{1, 2,3, \dots, n\}\}$

- Exercise: Any interval (i, j) can be written as a disjoint union of at most $2 \log n$ such intervals.
- Example: For $n = 256$: $[48,107] = [48,48] \cup [49,64] \cup [65,96] \cup [97,104] \cup [105,106] \cup [107,107]$

Count-Min: Range Queries and Quantiles

- Range Query: For $i, j \in [n]$ estimate $f_i + \cdots f_j$
- Approximate median: Find j such that:

$$f_1 + \cdots + f_j \geq \frac{m}{2} + \epsilon m \text{ and}$$

$$f_1 + \cdots + f_{j-1} \leq \frac{m}{2} - \epsilon m$$

Count-Min: Range Queries and Quantiles

- Algorithm: Construct $\log n$ Count-Min sketches, one for each I_i such that for any $I \in I_i$ we have an estimate $\tilde{f}_l$ for f_l such that:

$$\Pr\left[f_l \leq \widetilde{f_l} \leq f_l + \epsilon m\right] \geq 1 - \delta$$

- To estimate $[i, j]$, let $I_1 \ldots, I_k$ be decomposition:

$$\widetilde{f_{[i,j]}} = \widetilde{f_{I_1}} + \cdots + \widetilde{f_{I_k}}$$

- Hence,

$$\Pr\left[f_{[i,j]} \leq \widetilde{f_{[i,j]}} \leq 2\,\epsilon m \log n\right] \geq 1 \; - 2\delta \log n$$

Count-Min: Heavy Hitters

- Heavy Hitters: For $\phi \in [0,1]$ find all i with $f_i \geq \phi m$ but no elements with $f_i \leq (\phi - \epsilon)m$
- Algorithm:
  - Consider binary tree whose leaves are [n] and associate internal nodes with intervals corresponding to descendant leaves
  - Compute Count-Min sketches for each I_i
  - Level-by-level from root, mark children I of marked nodes if $\widetilde{f_I} \geq \phi m$
  - Return all marked leaves
- Finds heavy-hitters in $O(\phi^{-1} \log n)$ steps

More about Count-Min

- Authors: Graham Cormode, S. Muthukrishnan [LATIN'04]
- Count-Min is linear:

  Count-Min(S1 + S2) = Count-Min(S1) + Count-Min(S2)

- Deterministic version: CR-Precis
- Count-Min vs. Bloom filters
  - Allows to approximate values, not just 0/1 (set membership)
  - Doesn't require mutual independence (only 2-wise)
- FAQ and Applications:
  - https://sites.google.com/site/countminsketch/home/
  - https://sites.google.com/site/countminsketch/home/faq

Fully Dynamic Streams

- Stream: $\boldsymbol{m}$ updates $(x_i, \Delta_i) \in [n] \times \mathbb{R}$ that define vector f where $f_j = \sum_{i:x_i=j} \Delta_i$.
- Example: For $n = 4$

$$\langle (1,3), (3, 0.5), (1,2), (2, -2), (2,1), (1, -1), (4,1) \rangle$$
$$f = (4, -1, 0.5, 1)$$

- Count-Min Sketch:

$$\Pr\left[|\widetilde{f_x} - f_x| + \epsilon\left|\left|f\right|\right|_1\right] \geq 1 - \delta$$

- Count Sketch: Count-Min with **random signs** and **median** instead of min:

$$\Pr\left[|\widetilde{f_x} - f_x| + \epsilon\left|\left|f\right|\right|_2\right] \geq 1 - \delta$$

Count Sketch

- In addition to $H_i: [n] \rightarrow [w]$ use random signs $r[i] \rightarrow \{-1,1\}$

$$c_{i,j} = \sum_{x:H_i(x)=j} r_i(x) f_x$$

- Estimate:
  $\hat{f}_x = median(r_1(x)c_{1,H_1(x)}, \dots, r_d(x)c_{d,H_d(x)})$
- Parameters: $d = O\left(\log \frac{1}{\delta}\right), w = \frac{3}{\epsilon^{\mathbf{2}}}$

$$\Pr\left[|\widetilde{f_x} - f_x| + \epsilon ||f||_2\right] \geq 1 - \delta$$

ℓ_p-Sampling

- Stream: $\boldsymbol{m}$ updates $(x_i, \Delta_i) \in [n] \times \mathbb{R}$ that define vector f where $f_j = \sum_{i:x_i=j} \Delta_i$.
- ℓ_p-Sampling: Return random $I \in [n]$ and $R \in \mathbb{R}$:

$$\Pr[I = i] = (1 \pm \epsilon) \frac{|f_i|^p}{||f||_p^p} + n^{-c}$$

$$R = (1 \pm \epsilon) f_I$$

ℓ_0-sampling

- Maintain $\widetilde{F_0}$, and (1 ± 0.1)-approximation to F_0.
- Hash items using $h_j: [n] \to \left[0, 2^j - 1\right]$ for $j \in [\log n]$
- For each j, maintain:

$$D_j = (1 \pm 0.1)|\{t | h_j(t) = 0\}|$$

$$S_j = \sum_{t, h_{j(t)} = 0} f_t i_t$$

$$C_j = \sum_{t, h_j(t) = 0} f_t$$

- Lemma: At level $j = 2 + \lceil \log \widetilde{F_0} \rceil$ there is a unique element in the streams that maps to 0 (with constant probability)
- Uniqueness is verified if $D_j = 1 \pm 0.1$. If so, then output S_j / C_j as the index and C_j as the count.

Proof of Lemma

- Let $j = \lceil \log \widetilde{F_0} \rceil$ and note that $2F_0 < 2^j < 12\ F_0$
- For any i, $\Pr\left[h_j(i) = 0\right] = \frac{1}{2^j}$
- Probability there exists a unique i such that $h_j(i) = 0$,

$$\sum_i \Pr\left[h_j(i) = 0 \ and\ \forall k \neq i, h_j(k) \neq 0\right]$$

$$= \sum_i \Pr\left[h_j(i) = 0\right] \Pr\left[\forall k \neq i, h_j(k) \neq 0 \middle| \, h_{j(i)} = 0\right]$$

$$\geq \sum_i \Pr\left[h_j(i) = 0\right] \left(1 - \sum_{k \neq i} \Pr\left[h_j(k) = 0 \middle| h_j(i) = 0\right]\right)$$

$$= \sum_i \Pr\left[h_j(i) = 0\right] \left(1 - \sum_{k \neq i} \Pr\left[h_j(k) = 0\right]\right) \geq \sum_i \frac{1}{2^j} \left(1 - \frac{F_0}{2^j}\right) \geq \frac{1}{24}$$

- Holds even if h_j are only 2-wise independent

Application: Social Networks

- Each of n people in a social network is friends with some arbitrary set of other $n-1$ people
- Each person knows only about their friends
- With no communication in the network, each person sends a postcard to Mark Z.
- If Mark wants to know if the graph is connected, how long should the postcards be?

Sketching Graphs?

- We know how to sketch vectors: $v \to Mv$
- How about sketching graphs?
- $G(V, E) \equiv A_G$ (adjacency matrix): $A_G \to MA_G$
- Sketch columns of A_G
- $n = |V|, m = |E|$
- $O(poly(\log n))$ sketch per vertex / $\tilde{O}(n)$ total
  - Check connectivity
  - Check bipartiteness
- As always, space rather than dimension. Why?

Graph Streams

- Semi-streaming model: [Muthukrishnan ’05; Feigenbaum, Kannan, McGregor, Suri, Zhang’05]
  - Graph defined by the stream of edges $e_1, \dots, e_m$
  - Space $\tilde{O}(n)$, edges processed in order
  - Connectivity is easy on $\tilde{O}(n)$ space for insertion-only
- Dynamic graphs:
  - Stream of insertion/deletion updates $+e_{i_1}, -e_{i_2}, \dots, -e_{i_t}$ (assume sequence is correct)
  - Resulting graph has edge e_i if it wasn’t deleted after the last insertion
- Linear sketching dynamic graphs:

$$MA_{G \setminus e} = MA_G - \mathrm{MA_e}$$

Distributed Computing

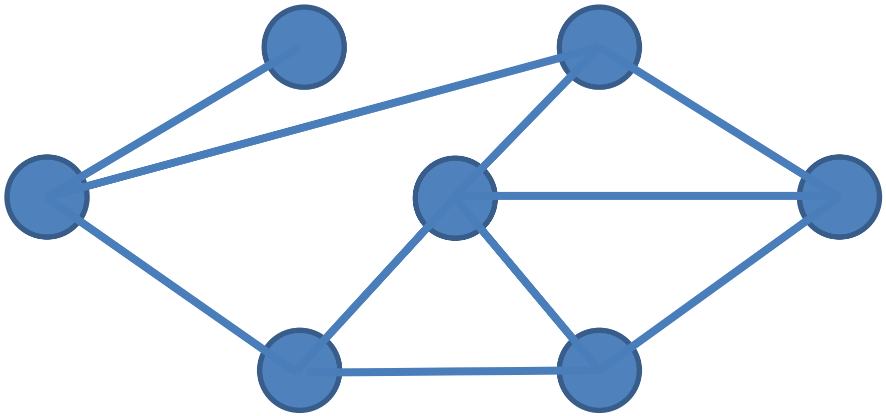
- Linear sketches for distributed processing
- S servers with o(m) memory:
  - Send m/S edges $(E_1, \ldots, E_s)$ to each server
  - Compute sketches $ME_1, \ldots, ME_s$ locally
  - Send sketches to a central server
  - Compute $MA_G = \sum_i^s ME_i$
- M has to have a small representation (same issue as in streaming)

Connectivity

- Thm. Connectivity is sketchable in $\tilde{O}(n)$ space
- Framework:
  - Take existing connectivity algorithm (Boruvka)
  - Sketch $A_G \to MA_G$
  - Run Boruvka on MA_G
- Important that the sketch is homomorphic w.r.t the algorithm

Part 1: Parallel Connectivity (Boruvka)

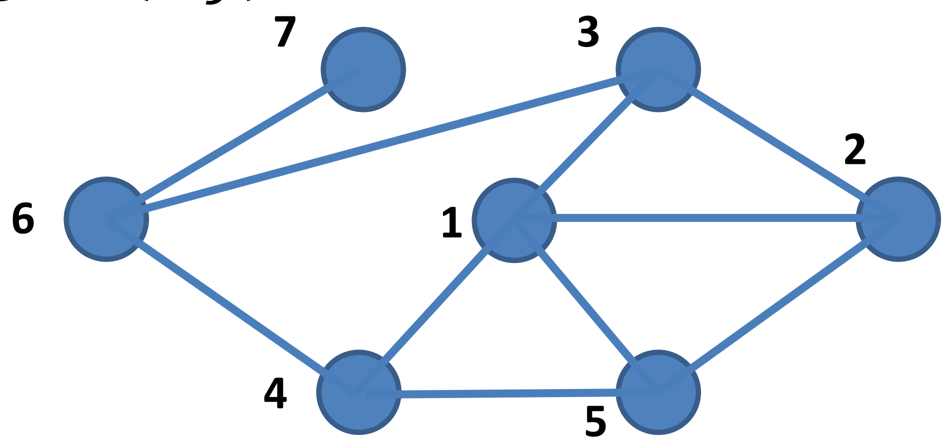
- Repeat until no edges left:
  - For each vertex, select any incident edge
  - Contract selected edges



- Lemma: process converges in $O(\log n)$ steps

Part 2: Graph Representation

- For a vertex i let a_i be a vector in $\mathbb{R}^{\binom{n}{2}}$
- Non-zero entries for edges (i, j)
  - $a_i[i, j] = 1$ if $j > i$
  - $a_i[i, j] = -1$ if j $< i$
- Example:



$$a_1 = (\quad 1, 1, 1, 1, 0, \dots, 0)$$
$$a_2 = (-1, 0, 0, 0, 0, 0, 1, 0, 1, \dots, 0)$$

{1,2}, {1,3}, {1,4}, {1,5}, {1,6}, {1,7}, {2,3}, {2.4}. {2,5}, …

- Lem: For any $S \subseteq V$ supp$(\sum_{i \in S} a_i) = E(S, V \setminus S)$

Part 3: L_0-Sampling

- There is a distribution over $M \in \mathbb{R}^{d \times m}$ with $d = O(\log^2 m)$ such w.p. 9/10 that $\forall a \in \mathbb{R}^m$:

$$Ca \to e \in supp(a)$$

[Cormode, Muthukrishnan, Rozenbaum'05; Jowhari, Saglam, Tardos '11]

- Constant probability suffices $-$ still $O(\log n)$ Boruvka iterations

Final Algorithm

- Construct $\log n$ ℓ_0-samplers for each a_i
- Run Boruvka on sketches:
  - Use $C_1 a_j$ to get an edge incident on a node j
  - For $i = 2$ to t:
    - To get incident edge on a component $S \subseteq V$ use:

$$\sum_{j \in S} C_i a_j = C_i \left(\sum_{j \in S} a_j \right) \rightarrow$$

$$\rightarrow e \in supp \left(\sum_{j \in S} a_j \right) = E(S, V \setminus S)$$

K-Connectivity

- Graph is k-connected is every cut has size $\geq k$
- Thm: There is a $O(nk \log^3 n)$-size linear sketch for k-connectivity
- Generalization: There is an $O(n \log^5 n / \epsilon^2)$-size linear sketch which allows to approximate all cuts in a graph up to error $(1 \pm \epsilon)$

K-connectivity Algorithm

- Algorithm for k-connectivity:
  - Let F_1 be a spanning forest of $G(V, E)$
  - For $i = 2, \ldots, k$
    - Let F_i be a spanning forest of $G(V, E \setminus F_1 \setminus \cdots \setminus F_{i-1})$
- Lem: $G(V, F_1 + \cdots + F_k)$ is k-connected iff G(V,E) is.
- $\Rightarrow$ Trivial
- $\Leftarrow$ Consider a cut in $G(V, \sum_{i=1}^{k} F_i)$ of size $< k$

$\Rightarrow \exists i^*$: this cut didn't grow in step i^*

$\Rightarrow$ there is a cut in $G(V, E)$ of size $< k$

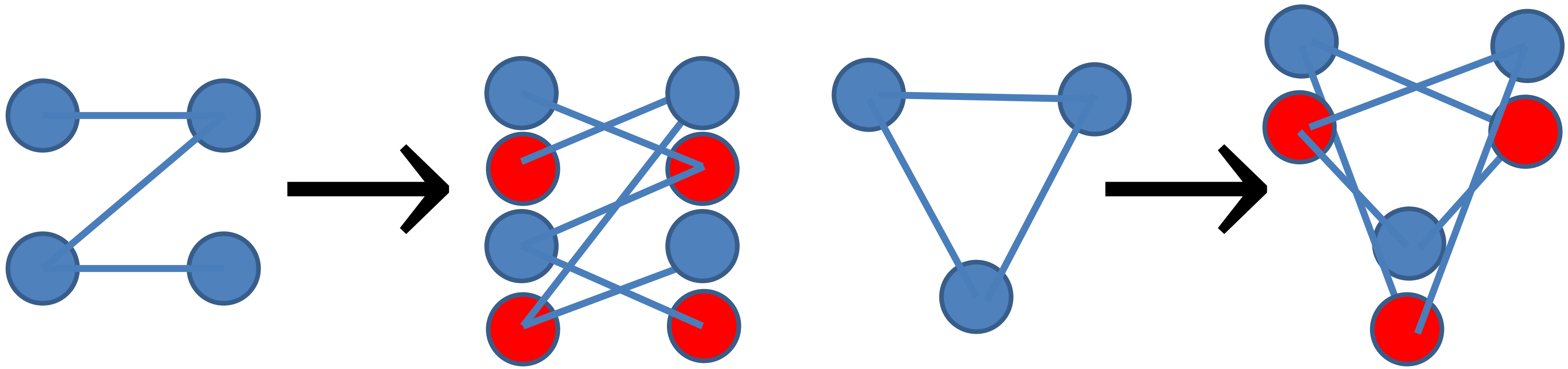
$\Rightarrow$ contradiction

K-connectivity Algorithm

- Construct k independent linear sketches $\{M_1 A_G, M_2 A_G \ldots, M_k A_G\}$ for connectivity
- Run k-connectivity algorithm on sketches:
  - Use $M_1 A_G$ to get a spanning forest F_1 of G
  - Use $M_2 A_G - M_2 A_{F_1} = M_2(A_{G-F_1})$ to find F_2
  - Use $M_3 A_G - M_3 A_{F_1} - M_3 A_{F_2} = M_3(A_{G-F_1-F_2})$ to find F_3
  - …

Bipartiteness

- Reduction: Given G define G' where vertices $v \to (v_1, v_2)$; edges $(u, v) \to (u_1, v_2)$ & (u_2, v_1)



- Lem: # connected components doubles iff the graph is bipartite.
- Thm: $O(n \log^3 n)$-size linear sketch for k-connectivity (sketch G' (implicitly).)

Minimum Spanning Tree

- If $n_i = \#$ connected components in a subgraph induced by edges of weight $\leq (1+\epsilon)^i$:

$$w(MST) \leq n - (1+\epsilon)^r + \sum_{i=0\ldots r-1} \lambda_i n_i \leq (1+\epsilon)w(MST)$$

where $\lambda_i = ((1+\epsilon)^{i+1} - (1+\epsilon)^i$

- cc(G) = #connected components of G
- Round weights up to the nearest power of $1+\epsilon$
- $G_i \equiv$ subgraph with edges of weight $\leq (1+\epsilon)^i$
- Edges taken by the Kruskal's algorithm:
  - n – cc(G_0) edges of weight 1
  - $cc(G_0) - cc(G_1)$ edges of weight $(1+\epsilon)$
  - …
  - cc(G_{i-1}) – cc(G_i) edges of weight $(1+\epsilon)^i$

Minimum Spanning Tree

- Let $r = \log_{1+\epsilon} W$ where W = max edge weight
- Overall weight:

$$n - cc(G_0) + \sum_1^r (1+\epsilon)^i \left(cc(G_{i-1}) - cc(G_i)\right)$$

$$= n - (1+\epsilon)^r + \sum_0^{r-1} \left((1+\epsilon)^{i+1} - (1+\epsilon)^i\right) cc(G_i)$$

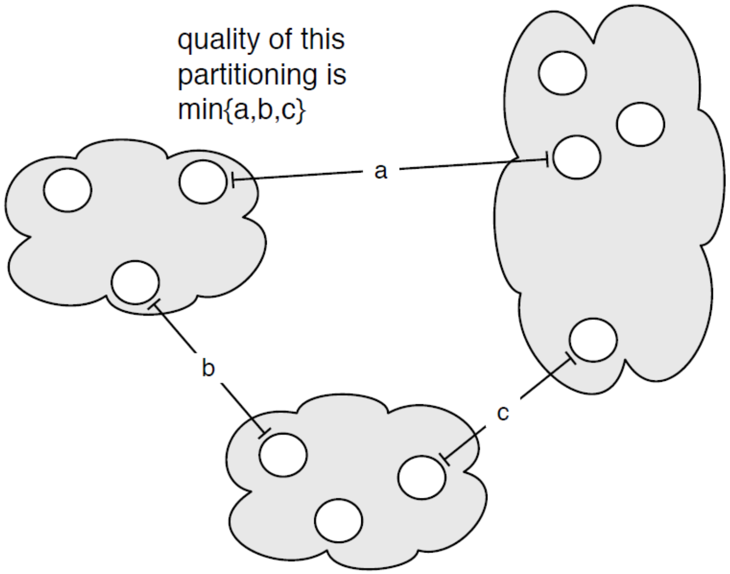
- Thm: $(1+\epsilon)$-approx. MST weight can be computed with $\tilde{O}(n)$ linear sketch for $W = poly(n)$

MST: Single Linkage Clustering

- [Zahn'71] **Clustering** via MST (Single-linkage):

k clusters: remove $\boldsymbol{k-1}$ longest edges from MST

- Maximizes **minimum** intercluster distance



[Kleinberg, Tardos]

Cut Sparsification

- Two problems:
  - Approximating Min-Cut in the graph (up to $1 \pm \epsilon$)
  - Preserving all cuts in the graph (up to $1 \pm \epsilon$)
- General cut sparsification framework:
  - Sample each edge e with probability p_e
  - Assign sampled edges weights $1/p_e$
- Expected weight of each cut is preserved, but too many cuts – can't take union bound

Cut Sparsification

- For an edge e let λ_e = weight of the minimum cut that contains e
- $\lambda =$ size of the Min-Cut in G
- Thm [Fung et al.]: If G is an undirected weighted graph then if $p_e \geq \min\left(\frac{C \log^2 n}{\lambda_e\, \epsilon^2}, 1\right)$ then the cut sparsification alg. preserves weights of all cuts up to $(1 \pm \epsilon)$
- Thm [Karger]: $p_e \geq \min\left(\frac{C \log n}{\lambda\, \epsilon^2}, 1\right)$ preserves Min-Cut up to $(1 \pm \epsilon)$

Minimum Cut

Algorithm:

- For $i = \{0,1,\ldots,2\log n\}$:
  - Let G_i be the subgraph of G where each edge is sampled with probability $1/2^i$
  - Let $H_i = \mathrm{F}_1,\ldots,F_k$ where $k = O\left(\frac{1}{\epsilon^2}\cdot\log n\right)$ and F_i are forests constructed by the k-connectivity alg.
- Return $2^j\lambda(H_j)$ where $j = \min\{i : \lambda(H_i) < k\}$

Space: $O\left(\frac{n\log^4 n}{\epsilon^2}\right)$, works for dynamic graph streams

Minimum Cut: Analysis

- Key property: If G_i has $\leq k$ edges across a cut then H_i contains all such edges
- $i^* = \left\lfloor \log \max\left\{1, \frac{\lambda\epsilon^2}{6 \log n}\right\} \right\rfloor$
- $i \leq i^* \Rightarrow p_e \geq \min\left(\frac{6 \log n}{\lambda\epsilon^2}, 1\right) \Rightarrow$ min cut in G_i is approximating min-cut in G up to $(1 \pm \epsilon)$
- $i = i^*$: By Chernoff bound # edges in G_{i^*} that crosses min-cut in G is $O\left(\frac{1}{\epsilon^2} \log n\right) \leq k$ w.h.p.

Cut Sparsification

Algorithm:

- For $i = \{0,1,\ldots,2\log n\}$:
  - Let G_i be the subgraph of G where each edge is sampled with probability $1/2^i$
  - Let $H_i = \mathrm{F}_1, \ldots, F_k$ where $k = O\left(\frac{1}{\epsilon^2} \cdot \log^2 n\right)$ and F_i are forests constructed by the k-connectivity alg.
- For each edge e let $j_e = \min\{\mathrm{i}: \lambda_e(H_i) < k\}$.
- If $e \in H_{j_e}$ then add e to the sparsifier with weight 2^{j_e}

- Space: $O\left(\frac{n\log^5 n}{\epsilon^2}\right)$, works for dynamic graph streams
- Analysis similar to the Min-Cut using [Fung et al.]