

# Sampling edges III

Kent Quanrud

September 17, 2020

## 1 Sampling Graphs

Heretofore we have largely applied randomization to numerical settings (for lack of a better word) such as to approximate counters, or to preserve Euclidean distances between vectors. We now turn to applying sampling to approximate *combinatorial* structures, such as graphs.

Consider, for example, the  $(s, t)$ -flow problem. We have as input a graph  $G = (V, E)$ , positive edge capacities  $c : E \rightarrow \mathbb{R}_{>0}$ , and two vertices  $s$  and  $t$ . We want to route the maximum amount of flow from  $s$  to  $t$ . Can we sample a small subgraph of  $G$  while preserving the value of the maximum  $s$  to  $t$  flow? If so, then we could run a max flow algorithm on the sampled subgraph and get faster overall running times. Try to imagine sampling edges from a graph in the interest of flow. Flow has a combinatorial aspect that would appear much more delicate than, say, heavy hitter estimates. For example, missing a single important edge when sampling can disrupt a polynomial number of paths used by the maximum flow. Nonetheless, we will see that for *undirected* graphs, the surprising answer is yes: one can indeed subsample the important parts of an undirected graph and preserve the maximum flow.

## 2 Minimum cut

We first consider the *minimum cut* problem in undirected graphs. The input consists of a connected, undirected graph  $G = (V, E)$  with positive edge capacities  $c : E \rightarrow \mathbb{R}_{>0}$ . A **cut** is a set of edges  $C \subseteq E$  whose removal disconnects the graph. The goal is to

$$\text{minimize } \sum_{e \in C} c(e) \text{ over all cuts } C \subseteq E. \quad (1)$$

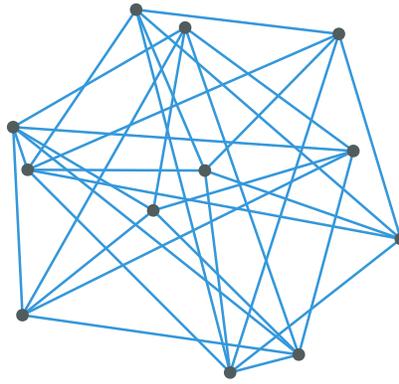


Figure 1: What is the minimum cut in this graph?

This problem is polynomial time solvable. Whatever the optimum cut is, it must be a minimum  $(s, t)$ -cut for some pair of vertices  $s$  and  $t$ . Thus to find the global minimum, one can guess  $s$  and  $t$  by looping over  $V$ , and compute the minimum  $\{s, t\}$ -cut for each choice of  $s$  and  $t$ . This takes a polynomial number of max flow operations (which can be reduced to linear; see Exercise 1).

For a set of vertices  $S$ , let  $\partial(S)$  denote the set of edges with exactly one endpoint in  $S$ .  $\partial(S)$  is called the cut **induced by  $S$** . The induced cuts are also the inclusionwise minimal cuts, and it suffices to consider only the induced cuts when solving (1).

We will study a stunning algorithm discovered by Karger (1993), that has had many implications beyond minimum cut. Consider the following description of Karger's algorithm.

*Repeatedly sample edges in proportion to their capacities until there is only one cut from which we have not yet sampled any edges. Return this cut.*

This algorithm is clearly ridiculous. For the unweighted setting, the above algorithm is equivalent to the following, equally absurd approach (see Exercise 2).

*Independently assign every edge  $e \in E$  a weight  $w_e \in [0, 1]$  uniformly at random. Build the minimum weight spanning tree  $T$  w/r/t  $w$ . Let  $e$  be the heaviest edge in  $T$ . Return the cut induced by the two components of  $T - e$ .*

Compare the two approaches above. Of course we know how to compute the minimum spanning tree; among other approaches, we can repeatedly add the smallest weight edge to  $T$  that does not create a cycle. On the other hand, in the first approach, it might appear difficult to keep track of which cuts we have and have not sampled from, being that there are so many cuts. This can be addressed by *contracting the graph*. Suppose we sample an edge  $e = \{s, t\}$ . Then we know that any cut  $\partial(S)$ , where  $s \in S$  and  $t \notin S$ , has now been sampled

```
random-contractions( $G = (V, E), c$ )
```

```
1. while  $|E| > 1$   
    A. sample  $e \sim c$   
    B.  $G \leftarrow G/e, c \leftarrow c/e$   
2. let  $E = \{e\}$   
3. return edges in the original graph that contracted to  $e$ 
```

Figure 2: A randomized minimum cut algorithm due to Karger (1993).

from. Thus we can safely **contract**  $e$ ; replacing  $s$  and  $t$  with a single vertex  $u$  that has the sum<sup>1</sup> of edges incident to  $s$  and  $t$ . Note that contracting  $e$  will only effect cuts that contain  $e$ . Now imagine we contract edges as we sample them. Eventually there are only two vertices left in the contracted graph, which represent two connected components in the input graph. These components induce the only cut we have not yet sampled from, and this is the cut that we return.

For an edge  $e \in G$ , we let  $G/e = (V/e, E/e)$  denote the graph obtained by contracting  $e$ , and we let  $c/e : E/e \rightarrow \mathbb{R}_{>0}$  denote the corresponding capacities. Pseudocode for the contraction algorithm is given in Figure 2.

The intuition behind Figure 2 is as follows. Here we describe the intuition for unweighted graphs for the sake of concreteness. (The intuition is the same for weighted graphs, except replacing “many edges” with “large capacity”, etc.) Suppose we have an unweighted graph  $G = (V, E)$ , and let  $C \subset E$  be the minimum cut. Since  $C$  is the minimum cut – keyword minimum – there are presumably very few edges in  $C$ . If we randomly sample an edge  $e \in E$ , then hopefully  $e \notin C$ . If  $C$  “survives” this round, then we have all made some progress because there is one less vertex in the graph after contracting  $e$ . In the next round,  $C$  is still the minimum cut, so the high-level logic from the first round still holds. Thus we can repeatedly sample edges and preserve the hope that we avoid  $C$ .

The above argument hinges on *how much smaller*  $C$  is than  $E$ . If we can argue that  $C$  is always a miniscule fraction of  $E$ , then we might hope that  $C$  survives to the end, after all. On the other hand, if  $C$  is even a small constant fraction of  $G$ , we will probably sample from  $C$  after a constant number of rounds. Observe also that over time,  $C$  becomes a larger and

---

<sup>1</sup>More precisely, for every edge  $f$  of the form  $\{s, z\}$  or  $\{t, z\}$ , we create a new edge  $\{u, z\}$  with the same capacity. If  $s$  and  $t$  both have edges to the same vertex  $z$ , we can either create two edges from  $u$  to  $z$  with the appropriate capacities, or make a single edge from  $u$  to  $v$  with the same capacity. We remove  $s$  and  $t$  and its incident edges from the graph, replacing them with  $u$  and the newly created edges incident to  $u$ .

larger fraction of  $E$ , as we contract and remove edges outside of  $C$ .

The key observation is that *every vertex  $v$  induces a cut  $\partial(v)$ , which must have at least as many edges as  $C$ . Thus the minimum cut is at most the minimum degree in the graph.* In turn, since the number of edges in  $E$  is the sum of degrees (divided by 2), *the minimum cut  $C$  is at most a  $2/n$  fraction of the total number of edges!* This observation holds initially in the input graph and thereafter in the contracted graphs, although  $n$  decreases by 1 in each iteration.

On the first iteration,  $C$  has at most a  $2/n$  chance of being hit. On the second iteration, assuming  $C$  survived the first iteration,  $C$  has (at most) a  $2/(n-1)$  chance of being hit. Continuing in this fashion, assuming  $C$  survived the first  $i-1$  iterations,  $C$  has a  $2/(n-i+1)$  chance of being hit in the  $i$ th iteration. If one combines these problems, one discovers that  $C$  has a  $\geq 1/\binom{n}{2}$  chance of surviving all  $n-1$  rounds. We can repeat the experiment  $\binom{n}{2} = O(n^2)$  (a polynomial!) number of times to find the minimum cut with constant probability, and  $O(n^2 \log n)$  times to find the minimum cut with high probability.

In the sequel, we formalize the the above argument, as well as extend it to positive capacities. For ease of notation, for a set of edges  $C \subset E$ , we denote the sum of capacities over  $C$  by

$$\sum_{e \in C} c(e) \stackrel{\text{def}}{=} \sum_{e \in C} c(e).$$

**Lemma 1.** *Let  $C^*$  be the minimum cut in  $(G, c)$ , and suppose  $e \notin C^*$ . Then  $C^*$  is (or maps to) the minimum cut in the contracted graph  $(G/e, c/e)$ .*

*Proof sketch.* Direct inspection. ■

**Lemma 2.**  $\sum_{e \in E} c(e) \geq \frac{\lambda n}{2}$ .

*Proof.* Every vertex  $v$  has weighted degree  $\sum_{e \in \partial(v)} c(e) \geq \lambda$  since  $\partial(v)$  is a cut. Thus

$$\sum_{e \in E} c(e) = \frac{\sum_v \sum_{e \in \partial(v)} c(e)}{2} \geq \frac{\lambda n}{2}.$$
■

**Lemma 3.** *Let  $e \sim c$ . Then  $P[e \in C^*] \leq \frac{2}{n}$ .*

*Proof.* We have  $P[e \in C^*] = \frac{\sum_{e \in C^*} c(e)}{\sum_{e \in E} c(e)} \stackrel{\text{(a)}}{\leq} \frac{2}{n}$  by (a) Lemma 2. ■

**Lemma 4.** Let  $C^*$  be a minimum cut. With probability  $\geq 1/\binom{n}{2}$ , random-contractions returns  $C^*$ .

*Proof.* For  $k \in \mathbb{Z}_{\geq 0}$ , let  $E_k$  be the event that we have not sampled  $C^*$  after  $k$  iterations. Initially,  $P[E_0] = 1$ , and we want to show that  $P[E_{n-2}] \geq 1/\binom{n}{2}$ . By Lemma 3, we have

$$P[E_k | E_{k-1}] \geq 1 - \frac{2}{n-(k-1)} \text{ for each } k \in [n].$$

The probability of succeeding (event  $E_{n-2}$ ) is at least

1 trial.  $P[\text{Bad}] = (1 - \frac{1}{\binom{n}{2}})$

$$P[E_{n-2}] = \prod_{k=1}^{n-2} P[E_k | E_{k-1}] \geq \prod_{i=3}^n \left(1 - \frac{2}{i}\right)$$

$k$  trials independent  $P[\text{all bad}]$

$$= \prod_{i=3}^n \frac{i-2}{i} = \frac{(n-2)!2}{n!} = \frac{1}{\binom{n}{2}}$$

$1-x \leq e^{-x}$

$$= \left(1 - \frac{1}{\binom{n}{2}}\right)^k \leq e^{-k/\binom{n}{2}} = e^{-5 \log(n)} = \frac{1}{n^5}$$

Thus with probability about  $1/n^2$ , the random contraction algorithm returns the minimum cut. To find the minimum cut with constant probability, we rerun the algorithm  $O(n^2)$  times and return the best cut. To find the minimum cut with *high probability*, we rerun the algorithm  $O(n^2 \log n)$  times.

The nice thing about repetition is that we can run the randomized trials *in parallel*. Moreover, a single instance of the contraction algorithm (via its connection to minimum spanning trees) can be made to run in  $\text{polylog}(n)$  time with polynomially many processors. Thus one obtains a randomized parallel algorithm for minimum cut.

**Corollary 5.** A randomized minimum cut can be computed in parallel in polylogarithmic time with a polynomial number of processors.

That said, random-contractions is not just an algorithm. It is also a profound structural observation about the number of minimum cuts in an undirected graph. In the above algorithm, any fixed minimum cut is returned with probability  $1/\binom{n}{2}$ . This implies that there are at most  $\binom{n}{2}$  minimum cuts in the graph!

**Corollary 6.** There are at most  $\binom{n}{2}$  minimum cuts in a graph.

## 2.1 Approximate minimum cuts

Above we showed that the total number of exactly minimum cuts is at most  $\binom{n}{2}$ . Here we extend the approach to count the number of approximate minimum cuts. In particular, we analyze the number of  $\alpha$ -approximate minimum cuts for any given  $\alpha \geq 1$ .

apx-random-contractions( $G = (V, E), c, \epsilon$ )

1. let  $k = \lceil 2\alpha \rceil$
2. while  $|V| > \lceil \alpha 2 \rceil$ 
  - A. sample  $e \in c$
  - B.  $G \leftarrow G/e, c \leftarrow c/e$
3. return all  $2^{k-1} - 1$  cuts in the original graph that are preserved in the contracted graph  $G$

To analyze the number of approximate minimum cuts, we analyze a slightly different algorithm called `apx-random-contractions`. This algorithm is similar to `random-contractions`, excepts it stops contracting when there are  $\lceil \alpha 2 \rceil$  edges left in the graph, and returns all of the cuts in the contracted graph. We analyze and obtain a lower bound on the probability that a fixed approximate minimum cut is one of these surviving cuts. The total number of approximate minimum cuts is at most the reciprocal of this probability.

**Lemma 7.** *Let  $C$  be a  $\alpha$ -approximate minimum cut. For  $e \sim c$ ,  $P[e \in C] \leq \alpha \frac{2}{n}$ .*

*Proof.* We have  $P[e \in C^*] = \frac{\sum_{e \in C^*} c(e)}{\sum_{e \in E} c(e)} \stackrel{(a)}{\leq} \frac{\alpha 2}{n}$  by (a) Lemma 2. ■

**Theorem 8.** *Let  $C$  be a  $\alpha$ -approximate minimum cut. `random-contractions` returns  $C$  with probability  $\geq 1/\binom{n}{\lceil 2\alpha \rceil}$ .*

*Proof.* Let  $k = \lceil 2\alpha \rceil$ . We have

$$\prod_{i=k+1}^n \left(1 - \frac{2\alpha}{i}\right) \geq \prod_{i=k+1}^n \left(1 - \frac{k}{i}\right) = \binom{n-k}{n} \binom{n-k-1}{n-1} \cdots \binom{1}{k+1} = \frac{(n-k)!k!}{n!} = \binom{n}{k}^{-1}$$
■

**Corollary 9.** *Let  $\alpha \geq 1$ , and let  $k = \lceil 2\alpha \rceil$ . The number of  $\alpha$ -approximate minimum cuts is at most*

$$2^k \binom{n}{k} \leq n^k$$

Corollary 9 has been sharpened to  $O(n^{\lceil 2\alpha \rceil})$  for any constant  $\epsilon > 0$  (Karger, 2000).

## 3 Sparsification

**Sparsification** refers to the general method of taking a large, dense graph and producing a smaller, sparse graph (over the same vertex set) that preserves some desired structure of the original graph.

We start with a very familiar example. Recall that two vertices  $s$  and  $t$  are **connected** in an undirected graph if there is a path from  $s$  to  $t$ . Given an undirected graph  $G = (V, E)$ , suppose we wanted a sparse subgraph  $G' = (V, E')$  such that any two vertices  $s, t \in V$  are connected in  $G$  iff they are connected in  $G'$ . Here there is a solution  $G'$  with (slightly less than)  $n$  edges. The reader probably knows the answer and should pause to realize it.

### 3.1 Preserving small connectivities

**Definition 10.** *Let  $G = (V, E)$  be an undirected graph. The **connectivity between two vertices  $s, t \in V$**  is the size of the minimum  $\{s, t\}$ -cut. When  $G$  is unweighted, this is also the maximum number of disjoint paths from  $s$  to  $t$ . The **connectivity of an edge  $e \in E$**  is defined as the connectivity of its endpoints.*

Above, we discussed sparsifiers that maintain all pairwise connectivities in an undirected graph up to connectivity 1. Here we will analyze a generalization by Nagamochi and Ibaraki (1992) for maintaining connectivities up to a fixed cardinality  $k \in \mathbb{N}$ . That is, given an undirected graph  $G = (V, E)$  and a parameter  $k$ , we will compute a set  $F \subseteq E$  with less than  $kn$  edges with the following guarantees.

1. If  $s, t \in V$  have connectivity  $\ell \leq k$  in  $G$ , then  $s$  and  $t$  have connectivity  $\ell$  in  $F$ .
2. If  $s, t \in V$  have connectivity  $\ell \geq k$  in  $G$ , then  $s$  and  $t$  have connectivity  $\geq k$  in  $G$ .

Nagamochi and Ibaraki (1992) proposed the following simple greedy algorithm. Consider the partition of  $E$  into forests  $F_1, F_2, F_3, \dots$  constructed as follows.

*For each edge  $e \in E$  (in any order), find the first forest  $F_i$  such that  $F_i + e$  is also a forest, and add  $e$  to  $F_i$ .*

For each  $k \in \mathbb{N}$ , let  $\overline{F}_k = F_1 \cup \dots \cup F_k$  denote the union of the first  $k$  forests, as a subgraph of  $G$ .

**Theorem 11** (Nagamochi and Ibaraki (1992)). *Let  $s, t \in V$  have connectivity  $\ell$  in  $G$ . Then for all  $k \leq \ell$ ,  $s$  and  $t$  have connectivity  $\geq k$  in  $\overline{F}_k$ .*

The key observation to proving Theorem 11 is the following.

**Lemma 12.** *Let  $C = \partial(S)$  be an induced cut with  $\ell$  edges. Then for all  $k \leq \ell$ ,  $|C \cap \overline{F}_i| \geq i$ .*

*Proof.* If not, there is an edge  $e \in C \setminus \overline{F}_k$ . For each  $i \in [k]$ , if  $e \notin F_i$ , then there must be another edge in  $C = \partial(S)$  in  $F_i$ . Applied to each  $i$ , we see that  $C$  has at least one edge in each  $F_i$ , hence at least  $k$  total in  $\overline{F}_k$  - a contradiction. ■

The proof of Theorem 11 is now immediate: by Lemma 12, every  $\{s, t\}$ -cut in  $\overline{F}_k$  has at least  $k$  edges.

Theorem 11 gives us our first nontrivial sparsifier. With  $< kn$  edges, we can preserve all connectivities up to size  $k$ . Don't be fooled by its simplicity! This is a landmark algorithm that has inspired many more ideas.

In the sequel, we will be interested in sparsifiers whose size is *independent of  $k$* . To this end, we will have to introduce both approximations and andomization. Before continuing on, however, we require the following observation about the Nagamochi-Ibaraki sparsifier: Nagamochi-Ibaraki sparsifier encodes, for every edge  $e$ , a lower bound on its connectivity, as follows.

**Lemma 13.** *For each edge  $e$ , if  $e \in F_k$ , then  $e$  has connectivity  $\geq k$ .*

*Proof.* Let  $e = \{s, t\}$ , let  $C$  be an  $\{s, t\}$ -cut, and let  $\ell = |C|$ . If  $\ell < k$ , then by Lemma 12, all  $\ell$  edges of  $C$  would be in  $\overline{F}_\ell$ , a contraction to  $e \in F_k$ . ■

## 3.2 Randomized sparsification

Let  $G = (V, E)$  be an unweighted and undirected graph, and let  $F_1, F_2, \dots$  partition  $E$  into forests per the greedy process in Section 3.1. Let  $\epsilon > 0$  be fixed.

Consider the randomized and reweighted subgraph  $G' = (V, E', c')$  constructed as follows. For each edge  $e \in E$ , let  $i$  be the index of the forest  $F_i$  such that  $e \in F_i$ , and let

$$p_e = \min \left\{ \left( \frac{c \log^2(n)}{\epsilon^2} \right) \frac{1}{i}, 1 \right\}$$

for a universal constant  $c \geq 1$  TBD. The random graph  $G' = (V, E', w)$  is now generated as follows.

*Independently for each edge  $e \in E$ , with probability  $p_e$ , add  $e$  to  $E'$  with weight  $w(e) = 1/p_e$ .*

We say that  $(G' = (V, E'), w)$  **preserves the weight of a cut**  $C = \partial(S)$  up to a  $(1 \pm \epsilon)$ -multiplicative factor if

$$(1 - \epsilon)|C| \leq \sum_{e \in C \cap E'} w(e) \leq (1 + \epsilon)|C|.$$

**Theorem 14.** *With high probability,  $(G', w)$  has  $O(n \log^3(n)/\epsilon^2)$  edges, and preserves the weight of every (induced) cut up to an  $(1 \pm \epsilon)$ -multiplicative factor.*

We can interpret Theorem 14 as a randomized approximation of the deterministic Nagamochi-Ibaraki sparsifiers from Section 3.1. Recall that the Nagamochi-Ibaraki forests preserved all cuts up to a fixed cardinality  $k$  exactly, with at most  $k(n - 1)$  edges total. This bound is ideal for constant  $k$ , but less appealing for larger values of  $k$ . Theorem 14 preserves all cuts of *all sizes* up to an  $(1 \pm \epsilon)$ -multiplicative. The output graph is almost linear in the number of vertices. A natural application of Theorem 14 is as a preprocessing step to maximum flow. By Theorem 14, we can reduce the number of edges in the graph to  $O(n \log^2(n)/\epsilon^2)$  edges while preserving the value of the maximum flow up to an  $(1 \pm \epsilon)$ -multiplicative factor. Running Ford-Fulkerson (or whatever other flow algorithm) in  $G'$  is faster because  $G'$  is smaller, and will still give an accurate estimate of the flow.

We note that there are also algorithms for weighted graphs that are almost based on random sampling in proportion to some measure of connectivity; see Section 3.3 below.

### 3.2.1 First observations

It is not at all obvious that  $G'$  should approximately preserve the value of every cut. However, it is easier to see that  $G$  is sparse.

**Lemma 15.** *With high probability,  $G'$  has  $O(n \log^3(n)/\epsilon^2)$  edges.*

*Proof.* Each edge in  $F_i$  is added with probability  $\leq O(\log^2(n)/i\epsilon^2)$ . In particular, there are at most  $n - 1$  edges with  $p_e = c \log^2(n)/\epsilon^2$ ,  $n - 1$  edges with  $p_e = c \log^2(n)/2\epsilon^2$ , and so forth. We conclude that

$$\mathbb{E}[|E'|] = \sum_{e \in E} p_e \leq \frac{c \log^2(n)}{\epsilon^2} \sum_i \frac{1}{i} |F_i| \stackrel{(a)}{\leq} O\left(\frac{n \log^2(n)}{\epsilon^2}\right).$$

For (a), we note that an easier upper bound is obtained by

$$\sum_i \frac{1}{i} |F_i| \leq n \sum_i \frac{1}{i} = O(n \log(n)).$$

To instead upper bound by  $O(n)$ , we observe that subject to  $\sum_i |F_i| \leq m$ , the LHS is maximized by have the first  $O(m/n)$  have  $n - 1$  edges. ■

Let us now return to cuts. We first verify that the random graph preserves the value of every cut in expectation.

**Lemma 16.** *Let  $C = \partial(S)$  be a cut. Then*

$$\mathbb{E} \left[ \sum_{e \in C \cap E'} w(e) \right] = |C|.$$

*Proof.* The key is that whenever we sample an edge  $p_e$  with probability  $p_e$ , we scale up the capacity by  $1/p_e$ . These two factors cancel each other out. Indeed,

$$\mathbb{E} \left[ \sum_{e \in C \cap E'} w(e) \right] \stackrel{(a)}{=} \sum_{e \in C} \mathbb{P}[e \in E'] \mathbb{E}[w(e) | e \in E'] = \sum_{e \in C} p_e \cdot \frac{1}{p_e} = |C|.$$

Here (a) is by linearity of expectation. ■

We can also show that any particular cut is preserved with high probability.

**Lemma 17.** *Let  $C = \partial(S)$  be a cut. Then for any  $\beta > 0$ ,*

$$\mathbb{P} \left[ \sum_{e \in S \cap E'} w(e) \geq (1 + \epsilon) |S| + \beta \right] \leq e^{-c \log^2(n) \beta / \epsilon |S|}$$

and

$$\mathbb{P} \left[ \sum_{e \in S \cap E'} w(e) \leq (1 - \epsilon) |S| - \beta \right] \leq e^{-c \log(n) \beta / \epsilon |S|}.$$

*Proof.* Let  $k = |C|$ . Then  $C \subseteq F_1 \cup \dots \cup F_k$ . For every edge  $e \in C$ , we either have  $p_e = 1$  and  $e \in C$  deterministically, or with probability  $p_e$ ,  $e \in E'$  and

$$w(e) = \frac{1}{p_e} \leq \frac{\epsilon^2 ck}{\log^2(n)} = \left( \frac{\epsilon^2}{c \log^2(n)} \right) |C|.$$

Taking a step back, our goal is to estimate  $|C|$  by random sampling. As observed in Lemma 16, the expected value of the random sampling is correct. Each edge  $e \in C$  represents an independent random quantity that contributes at most a  $(\epsilon^2/c \log^2(n))$ -fraction of  $C$ . When many small and independent parts add up to a (relatively) large expected sum, then the random sum will be strongly concentrated around its concentration. Applying Chernoff bounds gives the desired result. ■

Note that for  $\beta = \epsilon|S|$ , Lemma 17 says that any particular cut is preserved up to an  $(1 \pm 2\epsilon)$ -multiplicative factor with probability of error  $\leq n^{-c \log(n)}$ . This is very encouraging, as the error probability is small enough to take the union bound over polynomially many cuts. However, it is not enough for Theorem 14, which says that *all cuts* are preserved with high probability – and there are, alas,  $2^n$  many different cuts.

### 3.2.2 Overview of the proof

We have already collected some observations that suggests Theorem 14 is not unreasonable. Every cut is preserved in expectation, and up to an  $(1 \pm \epsilon)$ -approximation with very high probability. The problem is that we want to preserve *all* cuts, there are too many cuts to simply apply a union bound.

We take inspiration from Section 2, where we analyzed a simple randomized algorithm for minimum cut. Intuitively, when sparsifying a graph, the minimum cuts should be the hardest cuts to preserve as they have the smallest margin for error. A remarkable consequence of the contraction algorithm is that there are only  $\binom{n}{2}$  minimum cuts in a graph. An extension of the same argument shows that, for any approximation factor  $\alpha > 1$ , there are at most  $n^{O(\alpha)}$   $\alpha$ -approximate minimum cuts. Thus there are only  $\text{poly}(n)$  cuts within a factor of, say, 2 of the minimum cut. This structure paints a more optimistic picture than assuming that there are  $2^n$  cuts we have to preserve and that they are all equally sensitive to random sampling.

We first bucket the edges by their sampling probabilities. For  $k = 1, 2, \dots$ , let

$$H_k = \bigcup_{2^{k-1} \leq i \leq 2^k} F_i.$$

Note that all edges in a bucket  $H_k$  are sampled with roughly the same probability, varying by at most a multiplicative factor of 2. This generates at most  $2 \log(n)$  buckets total.

Fix a bucket  $H_k$ . In the foregoing, a **section** of a bucket  $H_k$  is an edge set of the form  $A = H_k \cap C$ , where  $C = \partial(S)$  is an induced cut. We want to show that each section  $A = H_k \cap C$  is approximated fairly well by the sampled edges in  $H_k$ . To this end, one first

observes that if  $C \cap H_k \neq \emptyset$ , then  $C$  must have at least  $2^{k-1}$  edges. Each edge  $e \in H_k$  is given weight (roughly)  $\epsilon^2 2^k / \log^2(n)$ , which in particular is at most a  $\epsilon^2 / \log^2(n)$  fraction of  $|C|$  (up to constants). That is, each edge  $e \in H_k \cap C$  represents a small and independently random contribution to the final estimate of  $|C|$ . One can apply concentration bounds and obtain the following.

**Lemma 18.** *Let  $k \in \mathbb{N}$ . For every section of the form  $A = H_k \cap C$ , where  $C = \partial(S)$  is a cut, we have*

$$\mathbb{P} \left[ \sum_{e \in A \cap E'} w(e) \geq (1 + \epsilon)|A| + \left( \frac{\epsilon}{\log n} \right) |C| \right] \leq n^{-c_0 |C| / 2^{k+1}}$$

and

$$\mathbb{P} \left[ \sum_{e \in A \cap E'} w(e) \leq (1 - \epsilon)|A| - \left( \frac{\epsilon}{\log n} \right) |C| \right] \leq n^{-c_0 |C| / 2^{k+1}}.$$

We leave Lemma 18 as an exercise, although really all of the ideas of the proof are given in the description above.

Lemma 18 is helpful, but we find ourselves in a similar predicament as before, where now the number of sections of cuts may still be exponentially large. We want to argue that there is a nicer structure for which Lemma 18 will suffice.

Intuitively speaking, we should worry most about the sections  $A = H_k \cap C$  where  $|C|$  is small, because these give us the smallest margin of error. We will argue, by simulating the random contraction algorithm with some modifications based on *Mader's splitting lemma* (introduced later), that the number of sections of the form  $H_k \cap C$  where  $|C| \leq 2^k$  is a polynomial. In fact, there is a fairly smooth increase in the number of distinct sections  $H_k \cap C$ , where  $|C| \leq \alpha 2^k$  for  $\alpha > 1$ , as follows.

**Lemma 19.** *Let  $H$  be a set of edges where every edge has connectivity  $\geq K$ . Consider all sections of the form  $A = C \cap H$ , where  $C = \partial(S)$  is a cut. For all  $\alpha \geq 1$ , there are at most distinct  $n^{2\alpha}$  nonempty sections of the form  $A = H \cap C$ , where  $C$  is a cut with  $\leq \alpha K$ .*

We will prove Lemma 19 in Section 3.2.3 below.

Lemma 19 implies a structure to the sections of a bucket  $H_k$  that is much more forgiving than simply assuming there are  $2^n$  sections with cuts and every cut has size roughly  $2^k$ . Lemma 19 says that there are only polynomially many sections with cuts with  $\leq 2^k$  that we really need to be worried about. It also says that there are only polynomially many more sections

of cuts with  $\leq 2^{k+1}$  edges. While there are some more sections in this category, we also have twice as much room for error.

**Lemma 20.** *Let  $k \in \mathbb{N}$ . With probability  $\geq 1 - 1/\text{poly}(n)$ , for all sections of the form  $A = H_k \cap C$ , we have*

$$(1 - \epsilon)|A| - \frac{\epsilon}{\log n}|C| \leq \sum_{e \in A \cap E'} w(e) \leq (1 + \epsilon)|A| + \frac{\epsilon}{\log n}|C|.$$

Let us prove Lemma 20 now under the assumption that Lemma 18 and Lemma 19 hold true.

*Proof of Lemma 20.* Recall that every edge  $e \in H_k = F_{2^{k-1}} \cup F_{2^{k-1}+1} \cup \dots \cup F_{2^k-1}$  has connectivity  $\geq 2^{k-1}$ . Thus any cut  $C = \partial(S)$  intersecting  $H_k$  has at least  $2^{k-1}$  edges.

We first group the number of distinct sections  $H_k \cap C$  by  $|C|$ . For  $\ell \in \mathbb{N}$ , let

$$\mathcal{A}_\ell = \{H_k \cap C : 2^{k+\ell-1} \leq |C| < 2^{k+\ell}\}.$$

Fix  $\ell \in \mathbb{N}$ . By Lemma 19, we have

$$|\mathcal{A}_\ell| \leq n^{c_0 2^\ell} \tag{2}$$

for some constant  $c_0 > 0$ . For each section  $A = H_k \cap C$  where  $C \in \mathcal{A}_\ell$ , we have

$$(1 - \epsilon)|A| - \frac{\epsilon}{\log n}|C| \leq \sum_{e \in A \cap E'} w(e) \leq (1 + \epsilon)|A| + \frac{\epsilon}{\log n}|C| \tag{3}$$

with probability of error

$$\leq e^{-c \log(n)|C|} = n^{-c_1 |C|/2^k} \leq n^{-c_1 2^\ell/2}$$

for some constant  $c_1 > 0$  that can be increased by increasing the constant  $c$  in our sampling probabilities  $p_e$ . In particular, we can make (say)  $c_1 \geq 2c_0 + 4$ . In that case that probability of error is

$$\leq n^{-(c_0+2)2^\ell}. \tag{4}$$

Comparing (2) and (4), we see that we *can take the union bound over  $\mathcal{A}_\ell$ !* Indeed, by the

union bound, the error probability of failing (3) for at least one  $A \in \mathcal{A}_\ell$  is

$$|\mathcal{A}_\ell| \cdot n^{-(c_0+2)2^\ell} \leq n^{-2^{\ell+1}}.$$

We can now take the union bound over all  $(O(\log \log n))$  choices of  $\ell$ . ■

Theorem 14 now follows from Lemma 20. We first restate Theorem 14 for the reader's convenience.

**Theorem 14.** *With high probability,  $(G', w)$  has  $O(n \log^3(n)/\epsilon^2)$  edges, and preserves the weight of every (induced) cut up to an  $(1 \pm \epsilon)$ -multiplicative factor.*

*Proof.* We have already verified the number of edges in Lemma 15. It remains to verify each cut. With high probability, Lemma 20 holds for every bucket  $H_k$ . For each cut  $C = \partial(S)$ , we have

$$\sum_{e \in C \cap E'} w(e) = \sum_k \sum_{e \in C \cap H_k \cap E'} w(e) \leq \sum_k \left( (1 + \epsilon) |C \cap H_k| + \frac{\epsilon}{\log n} |S| \right) = (1 + 2\epsilon) |C|.$$

Likewise, we have

$$\sum_{e \in C \cap E'} w(e) = \sum_k \sum_{e \in C \cap H_k \cap E'} w(e) \geq \sum_k \left( (1 - \epsilon) |C \cap H_k| - \frac{\epsilon}{\log n} |S| \right) = (1 - 2\epsilon) |C|.$$

This completes the proof of Theorem 14. ■

### 3.2.3 Counting sections

It remains to prove Lemma 19, regarding the number of distinct sections induced by cuts of various sizes. To prove Lemma 19, we require the following beautiful graph theoretic fact due to Mader and Lovasz. We will not prove this fact, and instead refer the reader to the textbook Frank, 2011.

**Fact 1** (Splitting off lemma). *Let  $G = (V, E)$  be an undirected graph, and let  $v$  be a fixed vertex with even degree  $k$ . Then one “split off” the edges at  $v$  while preserving the connectivity of all pairs of vertices (not including  $v$ ) in the graph.*

Here “splitting off” means that we can pair up the edges incident to  $v$ , and replace them with the  $k/2$  edges formed by concatenating them and removing  $v$  from the middle. (e.g., two edges  $\{a, v\}$  and  $\{b, v\}$  are “split off” and replaced by a new edge  $\{a, b\}$ ).

To get some sense of our use of Fact 1, fix  $k \in \mathbb{N}$ , and consider a section of the form  $A = C \cap H_k$  where  $C = \partial(S)$ . Suppose we apply the splitting off lemma to split off a vertex  $v$  with degree  $< 2^{k-1}$ . First, note such a vertex  $v$  cannot be incident to any edge in  $H_k$  because every edge in  $H_k$  still has connectivity  $\geq 2^{k-1}$ . Second, when splitting off  $v$ , we note that the cut  $C$  might decrease in size, but cannot increase. Third, the connectivity of every edge in  $H_k$  is preserved, and in particular stays  $\geq 2^{k-1}$ . Thus when splitting off  $v$ , the section  $A = C \cap H_k$  maps directly to the same section  $A$ , now as an intersection  $w \cap H_k$  where  $|w| \leq |C|$ .

We now restate and prove Lemma 19.

**Lemma 19.** *Let  $H$  be a set of edges where every edge has connectivity  $\geq K$ . Consider all sections of the form  $A = C \cap H$ , where  $C = \partial(S)$  is a cut. For all  $\alpha \geq 1$ , there are at most distinct  $n^{2\alpha}$  nonempty sections of the form  $A = H \cap C$ , where  $C$  is a cut with  $\leq \alpha K$ .*

*Proof.* We will prove the bound indirectly by directly proving the following: the distinct number of sections of the desired form in an  $n$ -vertex graph is at most the maximum number of  $\alpha$ -approximate minimum cuts in an  $n$ -vertex graph.

Suppose we run the contraction algorithm with the following two modifications.

1. We first note that by doubling all the edges initially, we can assume the graph is even degree. The evenness of the degree is also preserved under edge contractions. Now, whenever there is a vertex  $v$  with degree  $< K$ , we split off  $v$ .
2. We stop the algorithm when there are  $\lceil 2\alpha \rceil$  vertices left in the graph, and return all sections induced by subsets of these vertices. (Thus  $< 2^{\lceil 2\alpha \rceil}$  sections are returned.)

Fix any section of the form  $A = H \cap C$ , where  $|C| < \alpha K$ . Note that  $A$  is preserved under splitting off, as well as the fact that  $A$  can be represented as an intersection  $H \cap C$  with a cut  $C$  such that  $|C| < \alpha 2^K$ . We return  $A$  iff (some) cut  $C$  such that  $A = H \cap C$  survives to the end of the algorithm.

Recall that the splitting off ensures that the minimum degree is  $\geq K$  at all times. If there are  $n$  vertices left and the minimum degree is  $K$ , then the probability that the next edge is sampled from  $C$  is

$$\leq \frac{2|C|}{Kn} < \frac{2\alpha}{n}.$$

This is the exact same as if counting the number of  $\alpha$ -approximate minimum cuts in Section 2.1. Thus we obtain the same bound. ■

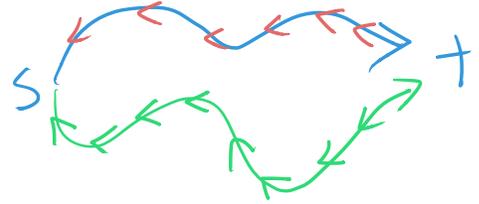
### 3.3 Weighted, Sparser, Stronger... and Deterministic

Benczúr and Karger (2015) showed how to compute cut sparsifiers for *weighted* graphs with only  $O(n \log(n)/\epsilon^2)$  edges. This approach samples edges in proportion to (underestimates of) their *edge strengths*, a different notion of connectivity. Benczúr and Karger, 2015 was actually the first such sparsification result, and precedes the simpler algorithm described above, which is from Fung, Hariharan, Harvey, and Panigrahi (2019). We point out two generalizations of cut sparsifiers, which we plan to return to later. Spielman and Srivastava (2011) showed how to randomly sample  $O(n \log(n)/\epsilon^2)$  edges as to preserve the *Laplacian quadratic form* of an undirected graph up to an  $(1 \pm \epsilon)$ -multiplicative factor - which is a stronger approximation guarantee that approximately preserves all cuts as well as many other properties of interest. Batson, Spielman, and Srivastava (2012) then showed how to sparsify the Laplacian (and preserve all cuts) with only  $O(n/\epsilon^2)$  edges *deterministically*. Removing the  $\log n$  factor is extremely surprising and consequential. Previously, the  $\log n$  factor arises for both graph and Laplacian sparsifiers from (essentially) a Chernoff-type union bound. Removing the  $\log n$  factor required a fundamentally new approach based on carefully studying the eigenvalues associated with a graph. This result has also had many implications outside of computer science including the resolution of the longstanding *Kadison-Singer problem* from functional analysis Marcus, Spielman, and Srivastava, 2013.

## 4 Max flow

**Input:**  $G = (V, E)$  simple undirected graph,  $s, t \in V$

**Output:** max flow of value  $\lambda$



### Ford-Fulkerson

*Repeatedly find a path from  $s$  to  $t$  in the residual graph, and route one unit of flow.*

- $O(m)$  time per unit flow. **BFS/DFS**
- $\lambda$  iterations.
- $O(m\lambda)$  time.
- $\lambda \leq n \implies O(mn)$  time.

Faster?

Faster approximation?

From last time:

For  $e \in E$ , let  $i$  be index of forest  $F_i$  s.t.  $e \in F_i$ , and let

$$p_e = \min \left\{ \frac{c \log^2(n)}{i \epsilon^2}, 1 \right\}$$

for constant  $c \geq 1$  TBD. The random graph  $G' = (V, E', w)$  is generated as:

*Independently for each edge  $e \in E$ , with probability  $p_e$ , add  $e$  to  $E'$  with weight  $w(e) = 1/p_e$ .*

**Theorem 14.** *With high probability,  $(G', w)$  has  $O(n \log^3(n)/\epsilon^2)$  edges, and preserves the weight of every (induced) cut up to an  $(1 \pm \epsilon)$ -multiplicative factor.*

$$\tilde{\lambda} \in (1 \pm \epsilon) \lambda$$

$$m \rightarrow n \log^3(n)/\epsilon^2$$

$$O\left(\left(m + \frac{n\lambda}{\epsilon^2}\right) \log^4(n)\right)$$

For  $e \in E$ , let  $i$  be index of forest  $F_i$  s.t.  $e \in F_i$ , and let

$$p_e = \min \left\{ \frac{c \log^2(n)}{i \epsilon^2}, 1 \right\}$$

for constant  $c \geq 1$  TBD. The random graph  $G' = (V, E', w)$  is generated as:

*Independently for each edge  $e \in E$ , with probability  $p_e$ , add  $e$  to  $E'$  with weight  $w(e) = 1/p_e$ .*

**Theorem 14.** *With high probability,  $(G', w)$  has  $O(n \log^3(n)/\epsilon^2)$  edges, and preserves the weight of every (induced) cut up to an  $(1 \pm \epsilon)$ -multiplicative factor.*

- sparsify to  $(G', w)$  with  $m = O(n \text{polylog}(n)/\epsilon^2)$
- run Ford-Fulkerson on  $G'$
- $\implies (1 \pm \epsilon)$ -approximation to  $\lambda$  in  $O((m + n\lambda/\epsilon^2) \text{polylog}(n))$  randomized time

$$\left( m + \frac{n\lambda}{\epsilon^2} \right) \log^{O(1)}(n)$$

**Faster?**

**Faster exact?!?!**

Karger and Levine (2015)

**Theorem 21.** Max flow in simple undirected graph in  $O((m + \lambda n) \text{polylog}(n))$  randomized time.

FF goal: 1  $s \rightarrow t$  path

Iteration 1 in  $O(n \text{polylog}(n))$  time? (instead of  $m$ )  $O(n^3)$



sample  $n \text{polylog}(n)$

hit every  $s \rightarrow t$  cut w/ sparsifier

Iteration 2 in  $O(n \text{polylog}(n))$  time?

$s \rightarrow t$  cuts little smaller



Iteration  $\lambda/2$  in  $O(n \text{polylog}(n))$  time?

## Karger and Levine (2015)

**Theorem 22.** Max flow in simple undirected graph in  $O((m + \lambda n) \text{polylog}(n))$  randomized time.

In  $i$ th iteration, sample

$$O\left(\frac{\lambda n}{\lambda + i - 1} \text{polylog}(n)\right) \quad O\left(\frac{\lambda n}{\lambda + i - 1}\right) \text{polylog}(n)$$

edges proportional to  $p_e$ 's from sparsifiers. Look for  $s \rightarrow t$  path in sampled edges.

$$\frac{\lambda}{\lambda + i - 1} \quad n \text{polylog}(n)$$

## Karger and Levine (2015)

**Theorem 23.** Max flow in simple undirected graph in  $O((m + \lambda n) \text{polylog}(n))$  randomized time.

In  $i$ th iteration, sample

$$O\left(\frac{\lambda}{\lambda+i} n\right) \quad O\left(\frac{n}{\lambda+i-1} \text{polylog}(n)\right)$$

edges proportional to  $p_e$ 's from sparsifiers. Look for  $s \rightarrow t$  path in sampled edges.

**Key Lemma.** Sample always contains an  $s \rightarrow t$  path.

**Proof of theorem.**

$$\sum_{i=1}^{\lambda} \left(\frac{\lambda}{\lambda+i}\right)n$$

$$= \lambda n \left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{\lambda}\right)$$

$m$   
build forests

$$\lambda n \text{polylog}(n)$$

## 5 Key Lemma

For  $e \in E$ , let  $i$  be index of forest  $F_i$  s.t.  $e \in F_i$ , and let

$$p_e = \min\left\{\frac{c \log^2(n)}{i \epsilon^2}, 1\right\} = \frac{1}{i}$$

for constant  $c \geq 1$  TBD. The random graph  $G' = (V, E', w)$  is generated as:

*Independently for each edge  $e \in E$ , with probability  $p_e$ , add  $e$  to  $E'$  with weight  $w(e) = 1/p_e$ .*

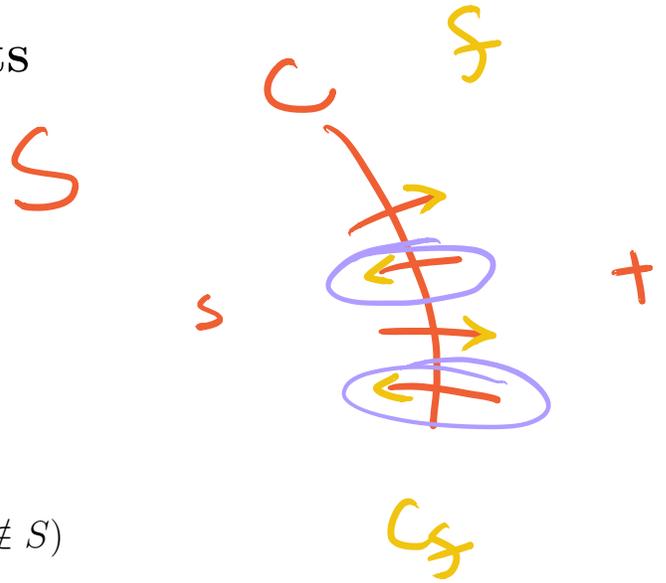
**Lemma 24.** *Let  $\lambda$  be the value of the max flow. Suppose we have routed  $(1 - \alpha)\lambda$  units of flow and  $\alpha\lambda$  units of flow remain to be routed, where  $\alpha > 0$ . Let  $E' \subset E$  sample (at least)  $O\left(\frac{1}{\alpha} n \log^3 n\right)$  edges from the same distribution as above. Then with high probability,  $E'$  contains an  $s \rightarrow t$  path.*

## 6 Undirected vs directed cuts

- Annoying - directed cuts
- Directed cuts generally a mess

### 6.1 Notation

- $f$  fixed flow
- $C = \partial(S)$  undirected  $\{s, t\}$ -cut ( $s \in S, t \notin S$ )
- define  $C_f$  = edges in  $C$  directed from  $s$  to  $t$  in residual graph



**Goal:** Sample one edge from  $C_f$  for all  $\{s, t\}$ -cuts  $C$ .

## 7 Last time: counting sections

For  $e \in E$ , let  $i$  be index of forest  $F_i$  s.t.  $e \in F_i$ , and let

$$p_e = \min \left\{ \frac{c \log^2(n)}{i \epsilon^2}, 1 \right\}$$

for constant  $c \geq 1$  TBD. The random graph  $G' = (V, E', w)$  is generated as:

*Independently for each edge  $e \in E$ , with probability  $p_e$ , add  $e$  to  $E'$  with weight  $w(e) = 1/p_e$ .*

**Lemma 18.** *Let  $k \in \mathbb{N}$ . For every section of the form  $A = H_k \cap C$ , where  $C = \partial(S)$  is a cut, we have*

$$\mathbb{P} \left[ \sum_{e \in A \cap E'} w(e) \geq (1 + \epsilon) |A| + \left( \frac{\epsilon}{\log n} \right) |C| \right] \leq n^{-c_0 |C| / 2^{k+1}}$$

and

$$\mathbb{P} \left[ \sum_{e \in A \cap E'} w(e) \leq (1 - \epsilon) |A| - \left( \frac{\epsilon}{\log n} \right) |C| \right] \leq n^{-c_0 |C| / 2^{k+1}}.$$

**Lemma 19.** *Let  $H$  be a set of edges where every edge has connectivity  $\geq K$ . Consider all sections of the form  $A = C \cap H$ , where  $C = \partial(S)$  is a cut. For all  $\alpha \geq 1$ , there are at most distinct  $n^{2\alpha}$  nonempty sections of the form  $A = H \cap C$ , where  $C$  is a cut with  $\leq \alpha K$ .*

**Lemma 20.** *Let  $k \in \mathbb{N}$ . With probability  $\geq 1 - 1/\text{poly}(n)$ , for all sections of the form  $A = H_k \cap C$ , we have*

$$(1 - \epsilon) |A| - \frac{\epsilon}{\log n} |C| \leq \sum_{e \in A \cap E'} w(e) \leq (1 + \epsilon) |A| + \frac{\epsilon}{\log n} |C|.$$

Then everything adds up: e.g.,

$$\sum_{e \in C \cap E'} w(e) = \sum_k \sum_{e \in C \cap H_k \cap E'} w(e) \geq \sum_k \left( (1 - \epsilon) |C \cap H_k| - \frac{\epsilon}{\log n} |S| \right) = (1 - 2\epsilon) |C|.$$

## 8 Something similar for directed cuts?

### 8.1 Key lemma last time

**Lemma 19.** *Let  $H$  be a set of edges where every edge has connectivity  $\geq K$ . Consider all sections of the form  $A = C \cap H$ , where  $C = \partial(S)$  is a cut. For all  $\alpha \geq 1$ , there are at most distinct  $n^{2\alpha}$  nonempty sections of the form  $A = H \cap C$ , where  $C$  is a cut with  $\leq \alpha K$ .*

*is  $\exists$ -cut*

### 8.2 Proof ideas

- Split off vertices w/ degree  $< K$
- Stop with  $2\alpha$  vertices left, return all remaining cuts

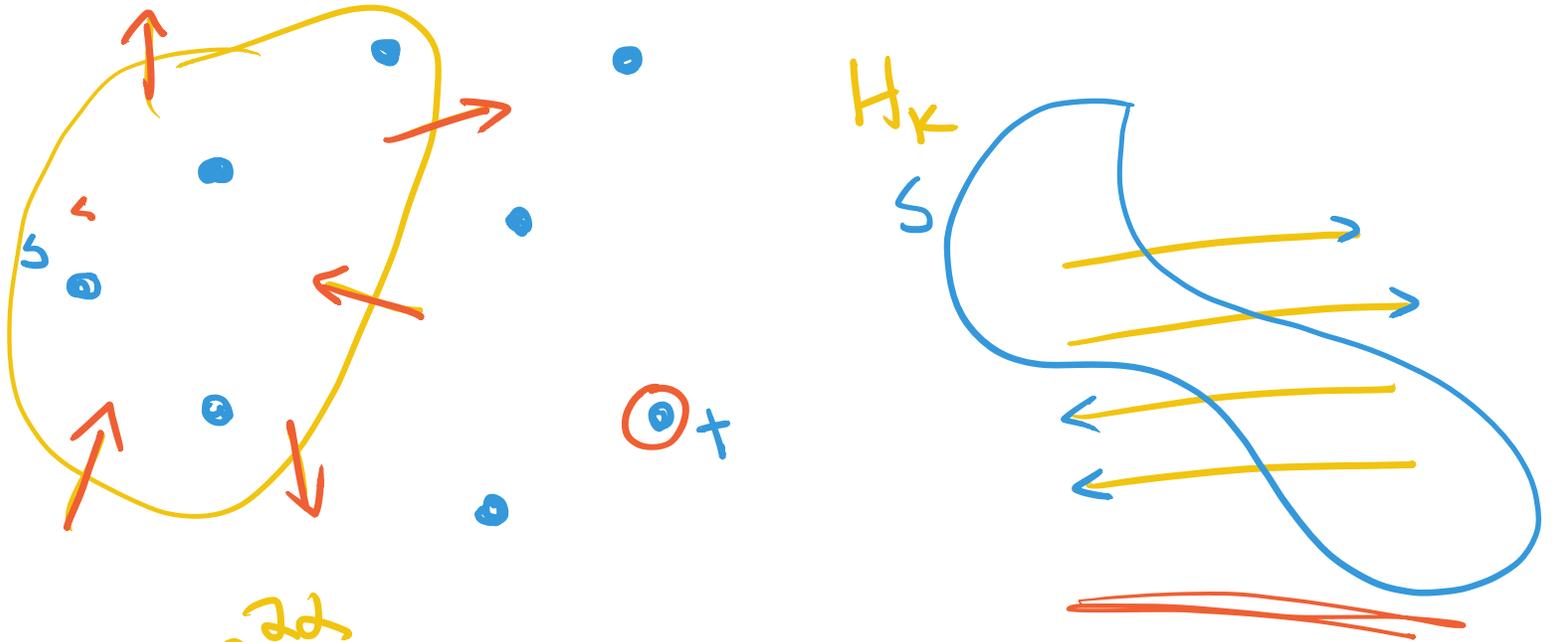
## 9 Something similar for directed cuts?

**Lemma 25.** Let  $H$  be a set of edges where every edge has connectivity  $K$  in the original, undirected graph. Consider all sections of the form

$$A = H_k \cap C_f,$$

$$2^{|A|}$$

where  $C = \partial(S)$  is an undirected  $\{s, t\}$ -cut, and  $C_f \subseteq C$  is the subset of undirected edges that have a directed edge in the directed  $s \rightarrow t$  cut from  $S$  to  $V \setminus S$ . For all  $\alpha \in \mathbb{N}$ , there are at most  $n^{O(\alpha)}$  distinct sections such that  $C$  has at most  $\alpha K$  edges.



$$\Rightarrow 2^{2\alpha}$$

$$2^{2\alpha} \cdot \frac{1}{\text{odds of surviving}}$$

$$2^{2\alpha} \cdot \frac{1}{\binom{n}{2\alpha}}$$

## 10 Something similar for directed cuts?

**Lemma 26.** *Let  $H$  be a set of edges where every edge has connectivity  $K$  in the original, undirected graph. Consider all sections of the form*

$$A = H_k \cap C_f,$$

*where  $C = \partial(S)$  is an undirected  $\{s, t\}$ -cut, and  $C_f \subseteq C$  is the subset of undirected edges that have a directed edge in the directed  $s \rightarrow t$  cut from  $S$  to  $V \setminus S$ . For all  $\alpha \in \mathbb{N}$ , there are at most  $n^{O(\alpha)}$  distinct sections such that  $C$  has at most  $\alpha K$  edges.*

**One change:**

- with  $2\alpha$  vertices remaining, return all remaining  $s \rightarrow t$  cuts

# 11 Something similar for directed cuts!

**Lemma 26.** Let  $H$  be a set of edges where every edge has connectivity  $K$  in the original, undirected graph. Consider all sections of the form

$$A = H_k \cap C_f,$$

where  $C = \partial(S)$  is an undirected  $\{s, t\}$ -cut, and  $C_f \subseteq C$  is the subset of undirected edges that have a directed edge in the directed  $s \rightarrow t$  cut from  $S$  to  $V \setminus S$ . For all  $\alpha \in \mathbb{N}$ , there are at most  $n^{O(\alpha)}$  distinct sections such that  $C$  has at most  $\alpha K$  edges.

**Lemma 27.** Suppose we sample  $O(\alpha n \log^3(n))$  edges proportional to  $p_e$ , as defined in Section 3.2. With probability  $1 - 1/\text{poly}(n)$ , for all sections of the form  $A = H_k \cap C_f$ , where  $C = \partial(S)$  is an undirected  $\{s, t\}$ -cut, we have

each section well concentrated

$$\sum_{e \in A \cap E'} w(e) \geq \frac{(1-\epsilon)}{4} |A| - \frac{1}{16\alpha \log n} |A|.$$

$\epsilon = O(1/n)$   
"2" |C|

**Lemma 28.** Let  $\beta > 1$ , and suppose we sample  $O(\beta n \log^3(n))$  edges proportional to  $p_e$ , as defined in Section 3.2. With probability  $1 - 1/\text{poly}(n)$ , for all  $\{s, t\}$ -cuts, we have

$X_1, \dots, X_n \in [0, 1]$

$$\sum_{e \in C_f \cap E'} w(e) \geq \frac{3}{4} |C_f| - \frac{1}{\beta} |C| > 0$$

$\beta = \frac{100}{\alpha}$   
 $\frac{3}{4} \frac{\alpha |C|}{2} - \frac{1}{\beta} |C| \Rightarrow \beta = \frac{100}{\alpha}$

$$\Pr [ X_1 + \dots + X_n \leq (1-\epsilon) E[X_1 + \dots + X_n] - \beta ] \leq e^{-\epsilon \beta}$$

## 12 Final touch

**Lemma 29.** Suppose the residual graph has maximum  $\{s, t\}$ -flow  $\alpha\lambda$ , where  $\lambda$  is the value of the maximum flow originally. For every undirected  $\{s, t\}$ -cut  $C = \partial(S)$ , where  $s \in S \subseteq V - t$ ,

$$|C_f| \geq \alpha|C|/2.$$

$(1-\alpha)\lambda$  units flow

$$\begin{aligned} |C| - (1-\alpha)\lambda \\ |C| \geq \lambda \end{aligned}$$

directed  $s \rightarrow t$  cut  $\geq \alpha\lambda$

$$\begin{aligned} |C_f| &\geq \frac{|C| - (1-\alpha)\lambda}{2} \\ &\geq \frac{\alpha|C|}{2} \end{aligned}$$

## 13 Proof of Key Lemma

**Lemma 24.** *Let  $\lambda$  be the value of the max flow. Suppose we have routed  $(1 - \alpha)\lambda$  units of flow and  $\alpha\lambda$  units of flow remain to be routed, where  $\alpha > 0$ . Let  $E' \subset E$  sample (at least)  $O(\frac{1}{\alpha}n \log^3 n)$  edges from the same distribution as above. Then with high probability,  $E'$  contains an  $s \rightarrow t$  path.*

## 14 Proof of Key Lemma

**Lemma 24.** *Let  $\lambda$  be the value of the max flow. Suppose we have routed  $(1 - \alpha)\lambda$  units of flow and  $\alpha\lambda$  units of flow remain to be routed, where  $\alpha > 0$ . Let  $E' \subset E$  sample (at least)  $O(\frac{1}{\alpha}n \log^3 n)$  edges from the same distribution as above. Then with high probability,  $E'$  contains an  $s \rightarrow t$  path.*

*Proof.* For all  $\{s, t\}$ -cuts  $C$ , we have

$$\sum_{e \in C_f \cap E'} w(e) \stackrel{(a)}{\geq} \frac{3}{4}|C_f| - \frac{\alpha}{4}|C| \stackrel{(b)}{>} 0$$

where (a) is by Lemma 28 and (b) is by Lemma 29. Thus we sample at least one edge from every directed  $\{s, t\}$ -cut in the residual graph, which implies that there is an  $s \rightarrow t$  path. ■

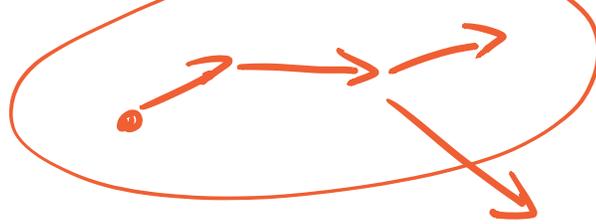
## 15 MST

MST over  $m$  edges vs sorting  $m$  numbers?

1.  $O(m \log n)$  – “Borůvka’s algorithm”

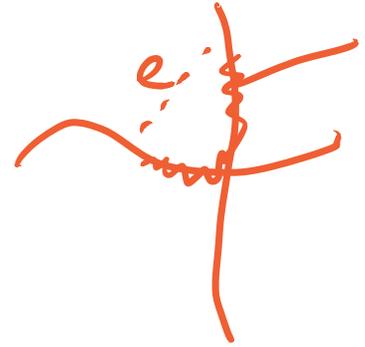
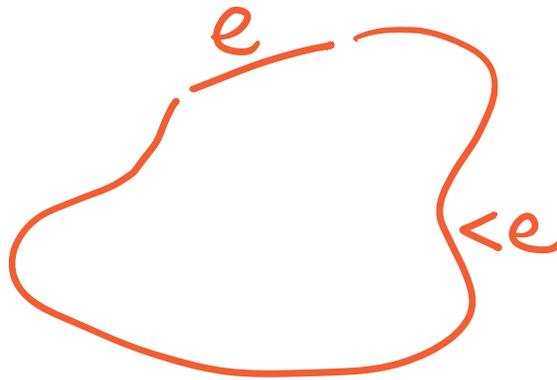
2.  $O(m + n \log n)$  – Fibonacci heaps

- edge “light” if in MST
- edge “heavy” if not.



**Lemma 30.** *If  $e$  is min weight edge in a cut  $C = \partial(S)$ , then  $e$  is light.*

**Lemma 31.** *If  $e$  is the max weight edge in cycle, then  $e$  is heavy.*



## 15.1 Verifying a spanning tree

Manika Henzinger

**Theorem 32** (Dixon, Rauch, and Tarjan (1992)). In  $O(m + n)$  time, one can verify if a set of edges  $F$  is a minimum spanning forest of  $G$ .

**Question.**

*Can one compute an MST as fast as one can verify it?*

Henzinger

**Theorem 33.** Let  $G = (V, E)$  be a weighted, undirected graph and let  $F \subseteq E$  be a forest. In  $O(m + n)$  time, one can compute the subset of edges that are heavy w/r/t  $F$ .

edge  $e$  is "heavy w/r/t  $F$ "  
if  $e$  is not  $\text{MST}(F + e)$

## 16 Randomized heavy filtering

**Definition 34.** Let  $F \subset E$  be a subset of edges. We say that  $e$  is **heavy w/r/t  $F$**  if  $e$  is heavy in  $F + e$ .

1. Let  $E' \subset E$  randomly sample each edge with probability  $1/2$ .

$= O(m)$

2. Remove all edges that are heavy w/r/t  $E'$ .

Bottleneck?

recurse

## 17 Karger, Klein, and Tarjan, 1995

1. Run Borůvka's algorithm for two iterations. ←
2. Let  $E' \subset E$  randomly sample each edge with probability  $1/2$ . ←
3. Let  $F' \subseteq E'$  be the minimum spanning forest of  $E'$ , obtained recursively. ←
4. Remove all edge that are heavy w/r/t  $F'$ . ←

    
n vertices  $\rightarrow \frac{n}{2}$  

## 18 Karger, Klein, and Tarjan, 1995

1. Run Borůvka's algorithm for two iterations.

$n \rightarrow n/4$

2. Let  $E' \subset E$  randomly sample each edge with probability  $1/2$ .

← flip coins

3. Let  $F' \subseteq E'$  be the minimum spanning forest of  $E'$ , obtained recursively.

4. Remove all edge that are heavy w/r/t  $F'$ .

**Lemma 35.** The expected number of  $F'$ -light edges is  $2n/4$

↑  
not heavy w/r/t  $F'$

imagine building MST on  $E$

every time add edge, flip coin

## 19 Karger, Klein, and Tarjan, 1995

1. Run Borůvka's algorithm for two iterations.

$$n \rightarrow n/4 \quad "2n" = \frac{n}{2}$$

2. Let  $E' \subset E$  randomly sample each edge with probability  $1/2$ .

3. Let  $F' \subseteq E'$  be the minimum spanning forest of  $E'$ , obtained recursively.

4. Remove all edge that are heavy w/r/t  $F'$ .

**Lemma 36.** The expected number of  $F$ -light edges is  $2n$ .

$$n \approx n$$

*Proof.*

1. Run greedy except flip coin before adding to MST
2. The number of  $F$ -light edges is exactly the total number of coin flips.
3. The total number of heads is at most  $n - 1$ .
4. The expected number of fair coin tosses until  $n - 1$  heads is  $2n - 1$ .



## 19.1 Running time analysis

Consider recursion tree.

**Lemma 37.** *The running time is proportional to the number of edges over all the subproblems.*

**Lemma 38.** *If a problem has  $m$  edges, then the left subproblem has  $m/2$  edges in expectation.*

**Lemma 39.** *If a subproblem has  $m$  edges, then the total expected number of edges over itself and all recursively left subproblems is  $2m$ .*

**Lemma 40.** *The running time is proportional to the original number of edges plus the number of edges over all right subproblems.*

**Lemma 41.** *The expected number of edges in a right subproblem is twice the number of vertices.*

**Lemma 42.** *The running time is proportional to the original number of edges plus the number of vertices in all right subproblems.*

**Lemma 43.** *The total number of vertices in all right subproblems is  $n/2$ .*

## 20 Exercises

2

**Exercise 1.** Section 2 describes a reduction from minimum cut to maximum flow, that requires  $O(n^2)$  calls to a max flow subroutine. Design and analyze a reduction that uses only  $O(n)$  calls to a max flow subroutine.

**Exercise 2.** Consider the randomized algorithm for minimum cut based on building the minimum spanning tree w/r/t randomized weights, described in Section 2.

1. Prove that this algorithm is equivalent to the random contractions algorithm for unweighted graphs.
2. Adjust the randomized spanning tree algorithm to account for weights, and prove its correctness.

**Exercise 3.** Let  $G = (V, E)$  be an undirected graph. For  $k \in \mathbb{N}$  a  **$k$ -cut** is a set of edges whose removal disconnects the graph into at least  $k$  connected components. Note that for  $k \geq 3$ , the minimum  $k$ -cut problem cannot easily be reduced to  $(s, t)$ -flow. In fact, the problem is NP-Hard when  $k$  is part of the input. *Hochbaum 80's*

1. Design and analyze a randomized algorithm that runs in  $n^{O(k)}$  randomized time and returns the minimum  $k$ -cut with high probability.
2. How does your algorithm relate to the preceding statement that  $k$ -cut is NP-Hard?

**Exercise 4.** Let  $G = (V, E)$  be an undirected graph. A **singleton cut** is a cut induced by a single vertex; i.e.,  $\partial(s)$  where  $s \in V$ . A **non-singleton cut** is a cut that is not a singleton cut.

Consider the problem of finding the minimum non-singleton cut...

**Exercise 5.** Let  $G = (V, E)$  be a simple **Eulerian** directed graph<sup>3</sup>.

1. Design and analyze an algorithm that computes a reweighted subgraph  $(G' = (V, E'), w : E' \rightarrow \mathbb{R}_{>0})$  with  $O(n \text{ poly}(\log n, 1/\epsilon))$  edges that preserves the value of every directed cut up to an  $(1 \pm \epsilon)$ -multiplicative factor.

<sup>2</sup>An earlier version of this note had an exercise about graphs with large minimum degree, which I realized was wrong. (In the previous version, this was exercise 3.)

<sup>3</sup>A directed graph with Eulerian if every vertex has the same in-degree as out-degree

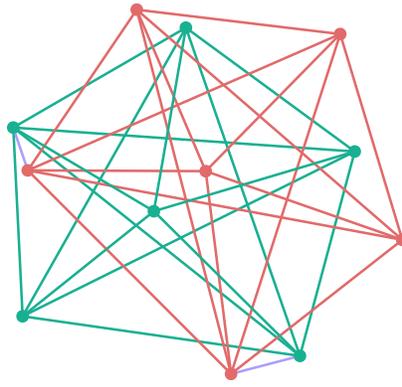


Figure 3: The minimum cut from Figure 1

2. Design and analyze a  $O((m + n^2) \text{polylog}(n))$  time algorithm for maximum flow on simple Eulerian graphs.
3. Suppose you were given a simple directed graph  $G$  that was “approximately Eulerian” up to an  $\alpha$ -multiplicative factor. (For example,  $\alpha = 2$ .) That is, suppose that the in-degree was always within a multiplicative factor of the out-degree. Can one design sparsifiers and faster max flow algorithms for such graphs? If no, what goes wrong? If yes, what is the dependency on  $\alpha$ ? Here a relatively brief answer will suffice.

## References

- Dixon, Brandon, Monika Rauch, and Robert Endre Tarjan (1992). “Verification and Sensitivity Analysis of Minimum Spanning Trees in Linear Time”. In: *SIAM J. Comput.* 21.6, pp. 1184–1192.
- Nagamochi, Hiroshi and Toshihide Ibaraki (1992). “A linear-time algorithm for finding a sparse  $k$ -connected subgraph of a  $k$ -connected graph.” In: *Algorithmica* 7.1, p. 583.
- Karger, David R. (1993). “Global Min-cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm”. In: *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*. Pp. 21–30.
- Karger, David R., Philip N. Klein, and Robert Endre Tarjan (1995). “A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees”. In: *J. ACM* 42.2, pp. 321–328.
- Karger, David R. (2000). “Minimum cuts in near-linear time”. In: *J. ACM* 47.1. Preliminary version in STOC, 1996, pp. 46–76.
- Frank, Andras (2011). *Connections in combinatorial optimization*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press.

- Spielman, Daniel A. and Nikhil Srivastava (2011). “Graph Sparsification by Effective Resistances”. In: *SIAM J. Comput.* 40.6, pp. 1913–1926. Preliminary version in STOC, 2008.
- Batson, Joshua D., Daniel A. Spielman, and Nikhil Srivastava (2012). “Twice-Ramanujan Sparsifiers”. In: *SIAM J. Comput.* 41.6, pp. 1704–1721. Preliminary version in STOC, 2009.
- Marcus, Adam, Daniel A Spielman, and Nikhil Srivastava (2013). *Interlacing Families II: Mixed Characteristic Polynomials and the Kadison-Singer Problem*. arXiv: 1306.3969 [math.CO].
- Benczúr, András A. and David R. Karger (2015). “Randomized Approximation Schemes for Cuts and Flows in Capacitated Graphs”. In: *SIAM J. Comput.* 44.2, pp. 290–319.
- Karger, David R. and Matthew S. Levine (2015). “Fast Augmenting Paths by Random Sampling from Residual Graphs”. In: *SIAM J. Comput.* 44.2, pp. 320–339.
- Fung, Wai Shing, Ramesh Hariharan, Nicholas J. A. Harvey, and Debmalya Panigrahi (2019). “A General Framework for Graph Sparsification”. In: *SIAM J. Comput.* 48.4, pp. 1196–1223. Preliminary version in STOC, 2011.