Karger's Minimum Cut Algorithm[1]

1 Randomized Minimum Cut

- We are given an ***undirected*** *multigraph* $G = (V, E)$. We will use n to denote the number of vertices and m to denote the number of edges. Note that we are working with multigraphs, and so there can be multiple parallel edges between the same two vertices.

  The goal is to find a minimum cut in the graph $G = (V, E)$. To remind every one, a cut is a collection of edges whose removal separates the graph into two non-trivial connected components. Non-trivial implies each component has at least one vertex. Alternately, the objective is to find $S \subseteq V$ such that $|\partial S|$ is minimized, where $\partial S := \{(u, v) \in E : \quad |\{u, v\} \cap S| = 1\}$ is the edges with exactly one endpoint in S. We will denote C to be the value of the minimum cut.

- The naive algorithm for the problem is as follows: select an arbitrary vertex $s \in V$, go over all vertices $t \in V \setminus \{s\}$ and find the minimum s, t-cut. Return the smallest such cut. Do you see why this would return the minimum cut in $G = (V, E)$? Finding the minimum s, t-cut can be done in many ways, but none of them are "trivial". In this lecture, we will see a really simple, indeed at first glance seemingly outright dumb, algorithm to solve the minimum cut problem. The algorithm can be described as follows:

  > *Select a random edge in G. Contract the end points of this vertex to get one supervertex. Keep repeating till you have only two. Return the remaining edges.*

  The inventor of this algorithm[2] is David Karger who published this in 1993 as a graduate student.

- To describe details, we need to specify what we mean by "contract the end points" of an edge. This is a graph operation which takes input two vertices x and y, and returns a graph where both these vertices are replaced by a "super vertex" (xy), and for every edge of the form (x, v) (similarly (y, v)) where $v \notin \{x, y\}$, we replace it with $((xy), v)$. See Figure 1 for an illustration.

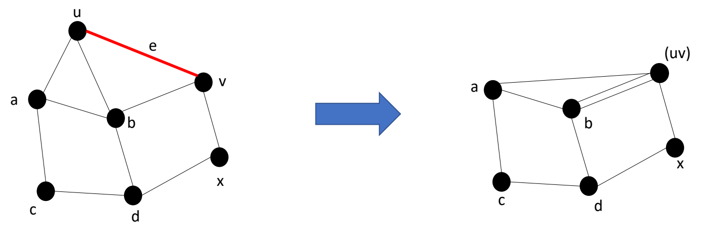

Figure 1: *The vertices u, v are being contracted. Note this leads to parallel edges. The original edge (u, v) is lost.*

[1]*Lecture notes by Deeparnab Chakrabarty. Last modified : 26th March, 2021*
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

[2]"*Global Min-Cuts in RNC and Other Ramifications of a Simple Mincut Algorithm.*", D. Karger. Proc. 4th Annual ACM-SIAM Symp. on Disc. Alg. (SODA)

More formally, one can think of the following algorithm as a *naive* implementation of the contract procedure.

```
1:  procedure CONTRACT($G, x, y$):
2:      ▷ Return graph where $x$ and $y$ are contracted
3:      Initialize empty graph $H$.
4:      $V(H) = V(G) \setminus \{x, y\} \ \cup \ \{(xy)\}$
5:      for $(u, v) \in E(G)$ do:
6:          If $u = x$ and $v = y$, ignore $(u, v)$.
7:          If $u = x$ or $u = y$, then add $((xy), v)$ to $E(H)$.
8:          If $v = x$ or $v = y$, then add $((xy), u)$ to $E(H)$.
9:          If all above fails, add $(u, v)$ to $E(H)$.
10:     ▷ Observe that the contract process decreases the number of vertices exactly by 1.
```

- Formally Karger's Random Contraction Algorithm can be described as

```
1:  procedure RANDMC($G$):▷ $G$ is undirected
2:      ▷ Return a cut $S$ in $G$
3:      $H \leftarrow G$ ▷ $H$ will keep contracting
4:      while $|V(H)| > 2$: do
5:          Choose an edge $(x, y) \in E(H)$ uniformly at random.
6:          $H \leftarrow$CONTRACT($H, x, y$)
7:      ▷ At this point $|V(H)| = 2$
8:      Return the edges in $E(H)$ as the cut. Alternately, the two vertices of $H$ correspond to
    subsets of vertices in $G$.
```

- ***Analysis.*** We will prove the following lemma.

Lemma 1. The probability that RANDMC returns a minimum cut is at least $\frac{2}{n(n-1)}$.

Proof. Let S^* be ***any*** minimum cut in G with $|\partial S^*| = C$. We show that RANDMC returns this particular cut with probability $\geq \frac{2}{n(n-1)}$ which will prove the claim. The proof hinges on two observations.

  - RANDMC performs $(n-2)$ iterations of the while loop since in each loop the number of vertices drops by exactly 1. The cut ∂S^* is returned by RANDMC if and only if in none of these iterations an edge from ∂S^* is picked in Line 5. To this end, let us define $\mathcal{E}_t$ to be the event that in iteration t, an edge from ∂S^* is not picked. Therefore, we get the probability RANDMC returns S^* is

$$\mathbf{Pr}[\text{RANDMC returns the cut } S^*] \quad = \quad \mathbf{Pr}\left[\bigwedge_{t=1}^{n-2} \mathcal{E}_t\right]$$

  - Conditioned on the event that none of the edges from ∂S^* have been contracted, the cut ∂S^* is a minimum cut in H as well: if there is a smaller cut in H, then there would be a corresponding smaller cut in G as well. And therefore, since the minimum cut is ***atmost*** the minimum degree and the sum of all the degrees in a graph is twice the number of edges, we get that for any iteration t if H_t is the graph H, then

$$\begin{aligned}\text{Conditioned on } \mathcal{E}_1 \wedge \cdots \wedge \mathcal{E}_{t-1},\ \ 2|E(H_t)| &\geq |V(H_t)| \cdot \min_{v \in V(H_t)} \deg_{H_t}(v) \\ &\geq (n-t+1) \cdot C\end{aligned}$$

The probability that in the tth loop, Line 5 picks an edge from ∂S^* is at most $|C|/|E(H_t)|$. Which implies, $\mathbf{Pr}[\mathcal{E}_t] \geq 1 - \frac{|C|}{E(H_t)}$, and thus from the above observation we get

$$\mathbf{Pr}[\mathcal{E}_t \mid \mathcal{E}_1 \wedge \cdots \wedge \mathcal{E}_{t-1}] \geq 1 - \frac{2}{n-t+1} = \frac{n-t-1}{n-t+1} \tag{1}$$

The proof can now be completed as follows.

$$\begin{aligned}\mathbf{Pr}[\text{RANDMC returns the cut } S^*] &= \mathbf{Pr}\left[\bigwedge_{t=1}^{n-2} \mathcal{E}_t\right] \\ &= \prod_{t=1}^{n-2} \mathbf{Pr}[\mathcal{E}_t \mid \mathcal{E}_1 \wedge \cdots \wedge \mathcal{E}_{t-1}] \\ &\underbrace{\geq}_{(1)} \prod_{t=1}^{n-2} \frac{n-t-1}{n-t+1} = \frac{2}{n(n-1)} \quad \square\end{aligned}$$

The success probability of $\Omega(1/n^2)$ may not sound impressive. However, to obtain a minimum cut with high probability, we simply repeat the experiment many times and take the smallest cut returned. Thus, we get the following theorem.

Theorem 1. Running RANDMC $O(n^2 \lg(1/\delta))$ times and taking the smallest cut thus obtained returns a minimum cut with probability $\geq 1 - \delta$.

- *Number of Minimum Cuts.* There is one other fascinating consequence of Lemma 1. Note what is saying : it is telling us that *any* given minimum cut S^* is returned with probability at least $\frac{2}{n(n-1)}$. This proves that in any undirected multigraph, the number of minimum cuts is *at most* $\frac{n(n-1)}{2}$. Why? Suppose not, and let $\mathcal{S}$ be the collection of all minimum cuts with $|\mathcal{S}| > \frac{n(n-1)}{2}$. For any $S \in \mathcal{S}$, let $\mathcal{E}_S$ be the event S is returned by RANDMC. Clearly, $\mathcal{E}_S$'s are mutually exclusive. And thus,

$$1 \geq \mathbf{Pr}[\bigvee_{S \in \mathcal{S}} \mathcal{E}_S] = \sum_{S \in \mathcal{S}} \mathbf{Pr}[\mathcal{E}_S]$$

Lemma 1 tells us that $\mathbf{Pr}[\mathcal{E}_S] \geq \frac{2}{n(n-1)}$ for every $S \in \mathcal{S}$, and thus $|\mathcal{S}| \leq \frac{n(n-1)}{2}$.

Exercise: *Come up with an undirected graph which has exactly* $\frac{n(n-1)}{2}$ *minimum cuts. Hint:* $\frac{n(n-1)}{2} = \binom{n}{2}$.

- *Remark on Running Time.* The algorithm, arguably, is much simpler than applying s, t-flows. But is it really faster? What is the running time of the algorithm? Well, the killer is that the probability of success is quite low, and we had to run it $O(n^2)$ times. However, how fast is RANDMC? Naively, it has $O(n)$ loops, and each loop contracts an edge. If we naively implement CONTRACT then that takes $O(m)$ time. So the naive implementation of RANDMC takes $O(nm)$ time. Giving an overall runtime of $O(n^3 m)$ for obtaining minimum cut. Not great.

  A few things. Firstly, the naive implementation of contract is naive – most of the graph is untouched, why are we spending $O(m)$ time? Secondly, one doesn't really *need* to contract G and keep an H as long as we can simulate that behavior. A careful analysis can lead you to an $O(n^2)$ implementation of RANDMC. Still, this gives an $O(n^4)$ time algorithm. But the idea is so simple and robust, that we can actually get better. And this leads to an algorithm, called the Karger-Stein algorithm (along with Cliff Stein who was at Dartmouth at the time), giving an $\widetilde{O}(n^2)$ time algorithm for minimum cut.

  Ponder This: *Does this algorithm work when the graph is directed? Or if we want an s, t-cut? Can you think of how to modify algorithms to solve these problems?*

Learning Tidbits:

- Algorithm Design: *Sometimes, the simplest sounding ideas shouldn't be discarded.*
- Analysis: *Breaking event into sequence of conditional events.*
- Structural: *Undirected graphs have very few (polynomially many) minimum cuts. Problem 5 in the Problem Set explores a robust version of this. I, personally, have used this fact many times in my papers.*