Randomized Algorithms
Lecture 3

Kousha Etessami

Min-Cut: Karger's randomized contraction algorithm

For an undirected graph $G = (V, E)$, a *cut* is a partition of the vertices V into two non-empty sets S and $V \setminus S$. A (*global*) *minimum cut* is a cut such that the total number of edges crossing the cut is minimized.
The **Min-Cut problem**: given G, find a minimum cut.
We assume w.l.o.g. that the graph G is connected (otherwise its "min-cut" has size 0, and we can determine whether G is connected in linear time $O(|V| + |E|)$).

Karger's (first) randomized algorithm (1993): D. Karger, "Global min-cuts in RNC and other ramifications of a simple mincut algorithm", SODA 1993.

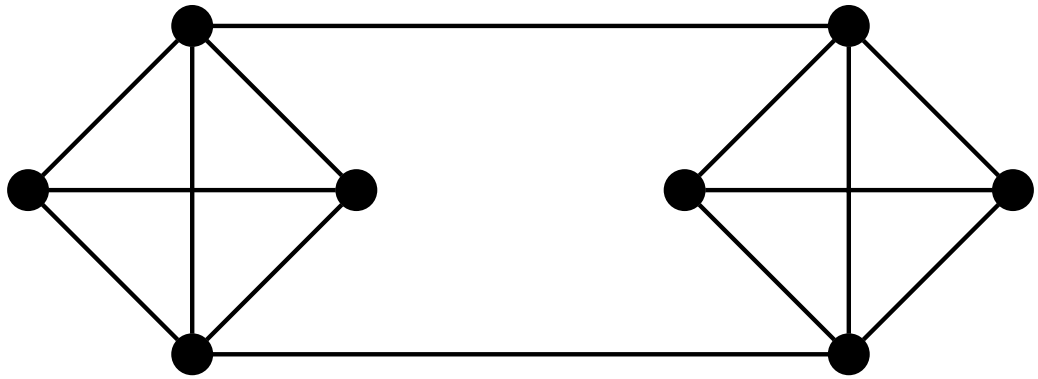
> *Repeatedly choose an edge uniformly at random (from the not-yet contracted edges) and* **contract** *its endpoints.*
> *When there are just two "vertices" left, return that cut.*

We will show this simple algorithm finds a minimum cut with high probability in time $O(m \cdot n^2 \log n)$, where $n = |V|$ and $m = |E|$.

History of min-cut algorithms

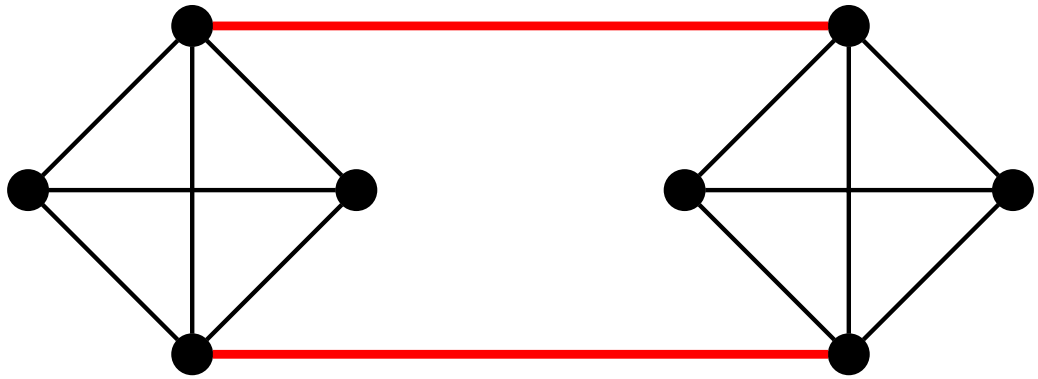
- A $\tilde{O}(mn^2)$ deterministic min-cut algorithm already follows from (directed) *maximum flow* algorithms, and the max-flow min-cut theorem ([Ford-Fulkerson'62] , [Gomory-Hu'61]).
- Later, better deterministic $O(mn \log n)$ time algorithms were given: [Nagamochi-Ibaraki,1992], [Hao-Orlin,1994], [Stoer-Wagner,1997].
- Meanwhile Karger (1993) gave his first, very simple, *randomized* algorithm which we will analyze, with running time $O(mn^2 \log n)$.
- Later, Karger and Stein (1996) improved on this randomized algorithm to give running time $O(n^2 \cdot \log^3 n)$.
- Finally, Karger (1996,2000), gave an improved randomized algorithm with "near-linear" running time: $O(m \log^3 n)$.
- Very recently, *deterministic* "near-linear" time algorithms obtained: $O(m \log^{12} n)$ [Kawarabayashi-Thorup,2015]; $O(m \log^2 n \log\log n)$ [Henzinger-Rao-Wang,2017]; $m^{1+o(1)}$ for *weighted* graphs [Li,2021].
- Experimental comparison by [Chekuri et. al., 1997] suggested the best performing algorithms in practice, up to 1996, were those of [Hao-Orlin,1994] and [Nagamochi-Ibaraki,1992].

Example



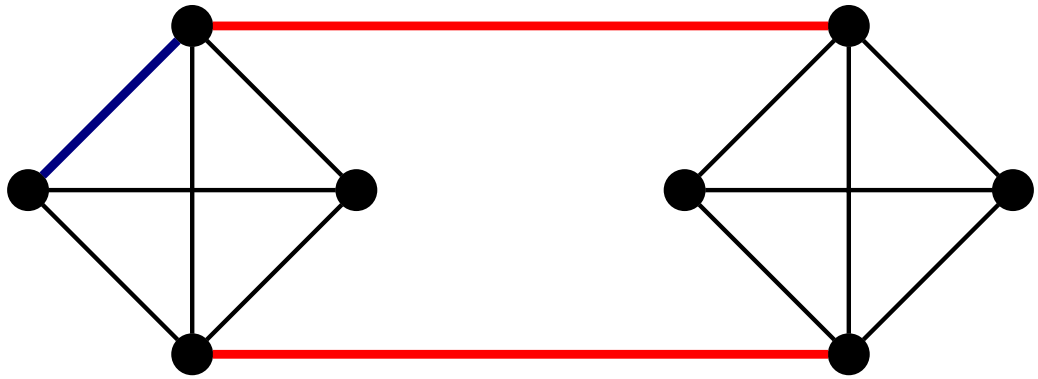
The min cut has size 2.

Example



The min cut has size 2.

Example

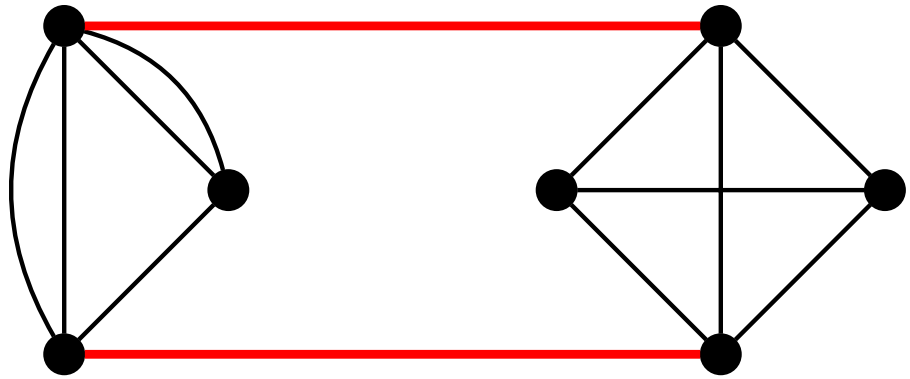


The algorithm randomly picks one edge out of 14.

We hope to avoid the min cut.

In this case the "bad" thing happens with probability $\frac{2}{14}$.

Example

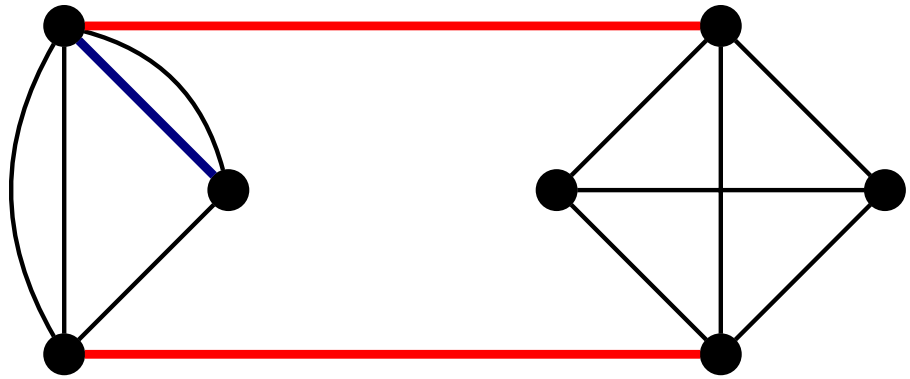


Contraction:

merge the endpoints of an edge into one.

Parallel edges are preserved, and self-loops removed.

Example

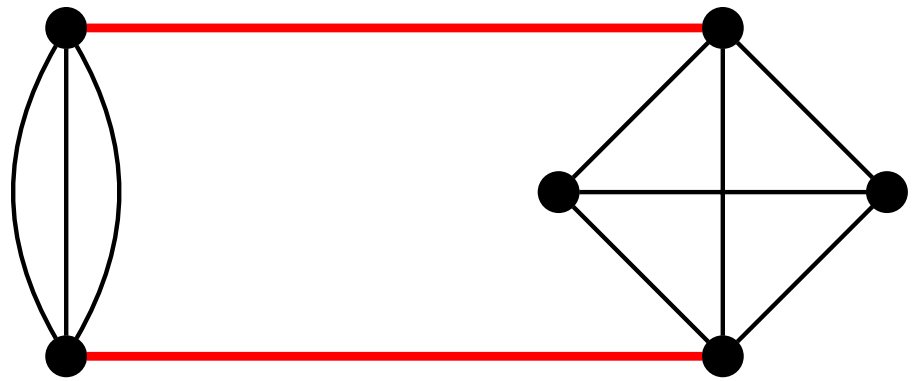


Contraction:

merge the endpoints of an edge into one.

Parallel edges are preserved, and self-loops removed.

Example

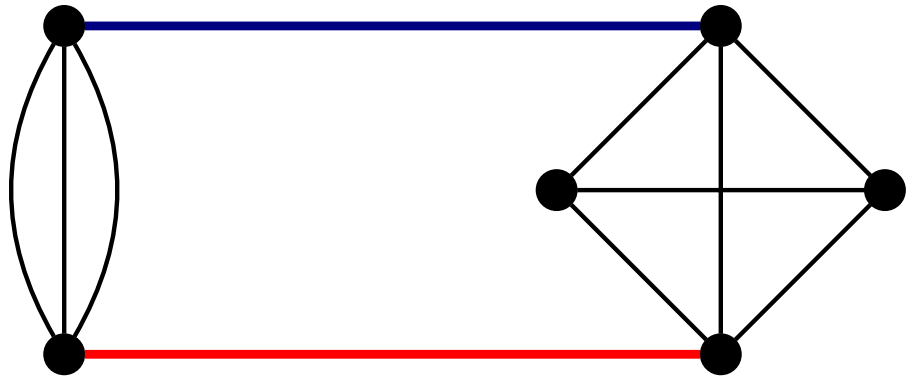


Contraction:

merge the endpoints of an edge into one.

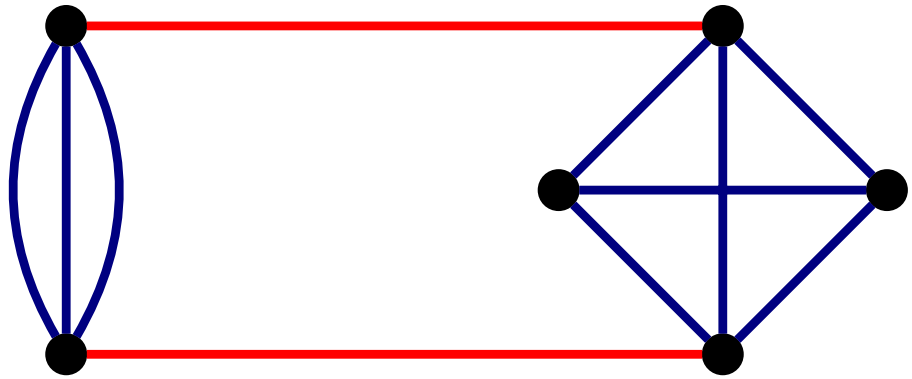
Parallel edges are preserved, and self-loops removed.

Example



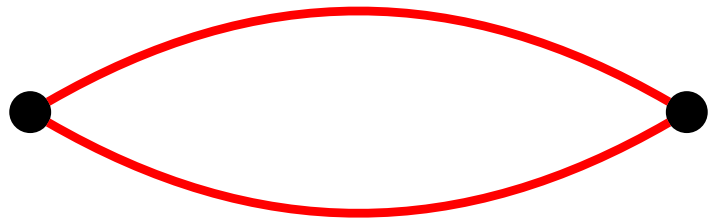
If we contract a cut edge, then we will not find that cut.

Example



Ideally, we should contract all edges except the min cut.

Example



Ideally, we should contract all edges except the min cut.

Implementation of the algorithm

Naive implementation of contractions would require complicated data structures to keep track of everything.

An equivalent way of looking at the algorithm is to pick a random permutation of all edges first, and then contracting edges from the first to the last.

What we need to find is the shortest prefix of the permutation such that those edges induce exactly two connected components.

Finding the connected components induced by a prefix with m' edges is known to take linear time $O(m' + n)$. Thus, by a binary search, we can find the shortest prefix that induces two components in $O(m \log m)$ time. By being a bit more clever during binary search, we can do it in time $O(m) + O(m/2) + O(m/4) + \cdots = O(m)$.

Denote by KargerOneTrial one iteration of Karger's algorithm.

Karger's contraction algorithm - analysis

Let k be the size of a min cut of G, let $S \subset V$ be a *specific* partition where C_S, the set of edges between S and $V \setminus S$, is of cardinality k.

Karger's contraction algorithm - analysis

Let k be the size of a min cut of G, let $S \subset V$ be a *specific* partition where C_S, the set of edges between S and $V \setminus S$, is of cardinality k.

We must have $deg(v) \geq k$ for every $v \in V$. (Why?)

Karger's contraction algorithm - analysis

Let k be the size of a min cut of G, let $S \subset V$ be a *specific* partition where C_S, the set of edges between S and $V \setminus S$, is of cardinality k.

We must have $deg(v) \geq k$ for every $v \in V$. (Why?)

The algorithm chooses a sequence of random edges $e_1, e_2, \ldots$

Let E_j be the event that $e_j \notin C_S$. ("Good" event.)

Karger's contraction algorithm - analysis

Let k be the size of a min cut of G, let $S \subset V$ be a *specific* partition where C_S, the set of edges between S and $V \setminus S$, is of cardinality k.

We must have $deg(v) \geq k$ for every $v \in V$. (Why?)

The algorithm chooses a sequence of random edges $e_1, e_2, \ldots$

Let E_j be the event that $e_j \notin C_S$. ("Good" event.)

Calculating $\Pr[E_1]$, there are k "cut-edges" (from C_S), and at least $k \cdot n/2$ edges overall. Hence

$$\Pr[E_1] \geq 1 - \frac{2k}{kn} \geq 1 - \frac{2}{n}.$$

We next calculate $\Pr[E_2 \mid E_1]$, the probability that the *2nd* edge avoids C_S, *conditional that the first edge was outside* C_S.

Karger's contraction algorithm - analysis

$\Pr[E_2 \mid E_1]$:

- Still have all k C_S edges (since we assumed E_1).
- Graph now has $(n-1)$ "vertices", each having degree $\geq k$ (why?); hence the graph now has at least $k \cdot (n-1)/2$ edges overall.

Hence

$$\Pr[E_2 \mid E_1] \geq 1 - \frac{2k}{(n-1)k} = 1 - \frac{2}{n-1}.$$

Next we will generalise this bound, namely, for any initial sequence of j edge-choices satisfying $\cap_{i=1}^{j} E_i$, we give a lower bound on

$$\Pr[E_{j+1} \mid E_1 \cap \ldots \cap E_j].$$

Karger's contraction Algorithm - Analysis

For any $j = 1, \ldots, n-3$, we analyse the *conditional* probability $\Pr[E_{j+1} \mid E_1 \cap \ldots \cap E_j]$:

- All k C_S edges still remain (since we assume $E_1 \cap \ldots \cap E_j$).
- How many edges have been removed? At least j

  Not necessarily exactly j, as we might have contracted "parallel edges" created by earlier contractions, which has the effect of removing more than one edge at a time when we contract a new edge.
- How many vertices have been removed? Exactly j

  The graph now has $(n-j)$ "vertices", and each must have degree $\geq k$ (why?); hence the graph now has at least $k \cdot (n-j)/2$ edges overall.

Therefore

$$\Pr[E_{j+1} \mid E_1 \cap \ldots \cap E_j] \;\geq\; 1 - \frac{2k}{(n-j)k} \;=\; 1 - \frac{2}{n-j}.$$

Karger's contraction Algorithm - Analysis

We hope that our contraction of random edges will lead us to a scenario where we are left with two "vertices" without contracting any of the C_S edges (min-cut) on the way.

If we achieve this, then one "vertex" will contain all of S, the other "vertex" all of $V \setminus S$, and the parallel edges between them are exactly the edges in the min-cut C_S.

The probability we get to this nice scenario is the probability that E_1 holds, *and* E_2 holds, *and* E_3 holds, and ...
But we can rewrite this probability using the definition of conditional probability. Formally,

$$\begin{aligned}
\Pr[\cap_{j=1}^{n-2} E_j] &= \Pr[E_1] \cdot \Pr[E_2 \mid E_1] \cdot \ldots \cdot \Pr[E_{n-2} \mid \cap_{i=1}^{n-3} E_i] \\
&= \prod_{j=1}^{n-2} \Pr[E_j \mid \cap_{i=1}^{j-1} E_i] \\
&\geq \prod_{j=1}^{n-2} \left(1 - \frac{2}{n-(j-1)}\right) = \prod_{j=3}^{n} \left(1 - \frac{2}{j}\right)
\end{aligned}$$

Karger's contraction Algorithm - Analysis

Expanding $\prod_{j=3}^{n}\left(1-\frac{2}{j}\right)$, we have

$$\begin{aligned}\prod_{j=3}^{n}\left(1-\frac{2}{j}\right) &= \prod_{j=3}^{n}\frac{j-2}{j}\\ &= \left(\frac{1}{3}\right)\left(\frac{2}{4}\right)\left(\frac{3}{5}\right)\left(\frac{4}{6}\right)\cdots\left(\frac{n-4}{n-2}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-2}{n}\right)\end{aligned}$$

Karger's contraction Algorithm - Analysis

Expanding $\prod_{j=3}^{n}\left(1-\frac{2}{j}\right)$, we have

$$\begin{aligned}\prod_{j=3}^{n}\left(1-\frac{2}{j}\right) &= \prod_{j=3}^{n}\frac{j-2}{j}\\ &= \left(\frac{1}{3}\right)\left(\frac{2}{4}\right)\left(\frac{3}{5}\right)\left(\frac{4}{6}\right)\cdots\left(\frac{n-4}{n-2}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-2}{n}\right)\\ &= \frac{2}{n(n-1)}\end{aligned}$$

So the probability that a single "run" of KargerOneTrial generates a particular minimum cut of the original graph is *at least* $\frac{2}{n(n-1)}$.

Hence the probability that it generates some minimum cut could be even more in practice. *(Why?)*

Karger's contraction Algorithm - Repeated iterations

We can improve our success probability by running KargerOneTrial many times, and returning the minimum of all the different cuts it produces.

If we do k trials, the probability that *none* is a min cut is *at most*

$$\left(1 - \frac{2}{n(n-1)}\right)^k.$$

[1] To see why this is true, first notice that for all $x \in \mathbb{R}$, $e^x \geq 1 + x$. Now let $x = -1/n$, and raise both sides to the power $-(1/x) = n$.

Karger's contraction Algorithm - Repeated iterations

We can improve our success probability by running KargerOneTrial many times, and returning the minimum of all the different cuts it produces.

If we do k trials, the probability that *none* is a min cut is *at most*

$$\left(1 - \frac{2}{n(n-1)}\right)^k.$$

We can relate this to $e \sim 2.71828\ldots$ (the base of the natural logarithm) using the following fact[1]: For all $n \geq 1$, $(1 - \frac{1}{n})^n \leq \frac{1}{e}$.

$$\Rightarrow \left(1 - \frac{2}{n(n-1)}\right)^{\frac{n(n-1)}{2}} \leq e^{-1},$$

and taking $k = c \cdot \frac{n(n-1)}{2} \cdot \ln(n)$, we get

$$\left(1 - \frac{2}{n(n-1)}\right)^k = \left(\left(1 - \frac{2}{n(n-1)}\right)^{\frac{n(n-1)}{2}}\right)^{c\ln(n)} \leq (e^{-1})^{c\ln(n)} = \frac{1}{n^c}.$$

[1] To see why this is true, first notice that for all $x \in \mathbb{R}$, $e^x \geq 1 + x$. Now let $x = -1/n$, and raise both sides to the power $-(1/x) = n$.

Wrapping up

- Probability tools used in our analysis were simple: we have used conditional probability iteratively:

$$\begin{aligned}\Pr[\cap_{j=1}^{n-2} E_j] &= \Pr[\cap_{j=2}^{n-2} E_j \mid E_1] \cdot \Pr[E_1] \\ &= \Pr[\cap_{j=3}^{n-2} E_j \mid E_1 \cap E_2] \cdot \Pr[E_1 \cap E_2 \mid E_1] \cdot \Pr[E_1] \\ &= \ldots\end{aligned}$$

  (also used simple inequalities relating $(1 - \frac{1}{n})^n$ and e)

- No approximation guarantee - analysis does not address the quality of C_S when it fails to be optimum.

#Min-Cut

We have shown that for a particular min-cut, C, the probability of finding C is at least $\frac{2}{n(n-1)}$. This implies that $|\mathcal{C}| \leq \frac{n(n-1)}{2}$, where $\mathcal{C}$ is the set of all Min-Cuts.

#Min-Cut

We have shown that for a particular min-cut, C, the probability of finding C is at least $\frac{2}{n(n-1)}$. This implies that $|\mathcal{C}| \leq \frac{n(n-1)}{2}$, where $\mathcal{C}$ is the set of all Min-Cuts.

Let F_C be the event of finding C. For $C' \neq C$, $\Pr[F_C \cap F_{C'}] = 0$.

Thus, $\Pr\left[\cup_{C \in \mathcal{C}} F_C\right] = \sum_{C \in \mathcal{C}} \Pr[F_C] \leq 1$.

#Min-Cut

We have shown that for a particular min-cut, C, the probability of finding C is at least $\frac{2}{n(n-1)}$. This implies that $|\mathcal{C}| \leq \frac{n(n-1)}{2}$, where $\mathcal{C}$ is the set of all Min-Cuts.

Let F_C be the event of finding C. For $C' \neq C$, $\Pr[F_C \cap F_{C'}] = 0$.

Thus, $\Pr\left[\cup_{C \in \mathcal{C}} F_C\right] = \sum_{C \in \mathcal{C}} \Pr[F_C] \leq 1$.

On the other hand, $\sum_{C \in \mathcal{C}} \Pr[F_C] \geq \sum_{C \in \mathcal{C}} \frac{2}{n(n-1)} = \frac{2|\mathcal{C}|}{n(n-1)}$.

#Min-Cut

We have shown that for a particular min-cut, C, the probability of finding C is at least $\frac{2}{n(n-1)}$. This implies that $|\mathcal{C}| \leq \frac{n(n-1)}{2}$, where $\mathcal{C}$ is the set of all Min-Cuts.

Let F_C be the event of finding C. For $C' \neq C$, $\Pr[F_C \cap F_{C'}] = 0$.

Thus, $\Pr\left[\cup_{C \in \mathcal{C}} F_C\right] = \sum_{C \in \mathcal{C}} \Pr[F_C] \leq 1$.

On the other hand, $\sum_{C \in \mathcal{C}} \Pr[F_C] \geq \sum_{C \in \mathcal{C}} \frac{2}{n(n-1)} = \frac{2|\mathcal{C}|}{n(n-1)}$.

Thus, $|\mathcal{C}| \leq \frac{n(n-1)}{2}$.

A u.a.r. cut is minimum with probability $\frac{|\mathcal{C}|}{2^n-1} \leq \frac{n(n-1)}{2(2^n-1)}$. Hence Karger's algorithm succeeds with probability much higher than a random cut.

#Min-Cut

We have shown that for a particular min-cut, C, the probability of finding C is at least $\frac{2}{n(n-1)}$. This implies that $|\mathcal{C}| \leq \frac{n(n-1)}{2}$, where $\mathcal{C}$ is the set of all Min-Cuts.

Let F_C be the event of finding C. For $C' \neq C$, $\Pr[F_C \cap F_{C'}] = 0$.

Thus, $\Pr\left[\cup_{C \in \mathcal{C}} F_C\right] = \sum_{C \in \mathcal{C}} \Pr[F_C] \leq 1$.

On the other hand, $\sum_{C \in \mathcal{C}} \Pr[F_C] \geq \sum_{C \in \mathcal{C}} \frac{2}{n(n-1)} = \frac{2|\mathcal{C}|}{n(n-1)}$.

Thus, $|\mathcal{C}| \leq \frac{n(n-1)}{2}$.

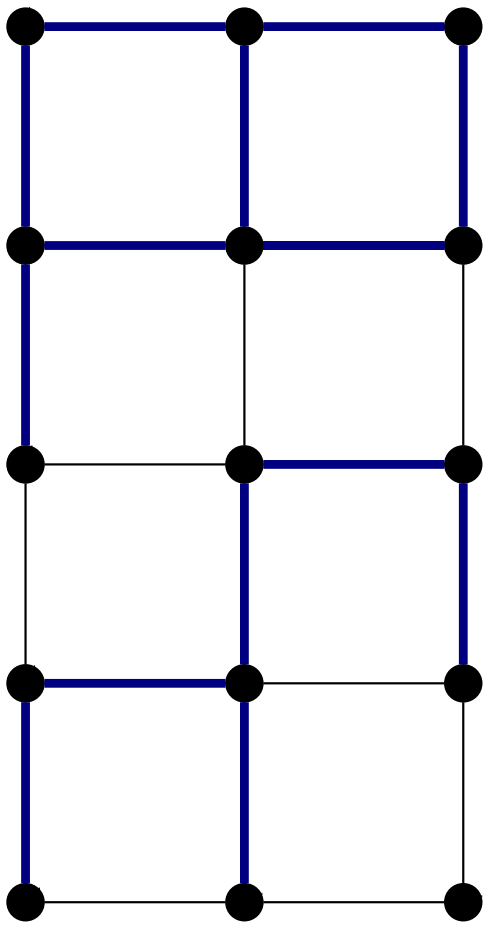
A u.a.r. cut is minimum with probability $\frac{|\mathcal{C}|}{2^n-1} \leq \frac{n(n-1)}{2(2^n-1)}$. Hence Karger's algorithm succeeds with probability much higher than a random cut.

Other use of random contraction

Suppose each edge of a graph $G = (V, E)$ "fails" with prob p, independently. Let G_p denote the randomly generated graph that results from the failures. Can we compute or approximate $\Pr[G_p \text{ is connected}]$ efficiently? Exact computation is #**P**-complete (Valiant, 1979), so a polynomial-time algorithm is unlikely.

Based on random contractions, Karger (1999) gave the first polynomial-time randomised approximation algorithm ("FPRAS") for Unreliability, namely for $1 - \Pr[G_p \text{ is connected}]$.

The first poly-time randomized approximation algorithm ("FPRAS") for Reliability, i.e., an algorithm that computes w.h.p. an approximation of $\Pr[G_p \text{ is connected}]$ within multiplicative error factor $(1 \pm \epsilon)$ in time polynomial in the encoding size of input G and in $\frac{1}{\epsilon}$, was found by Guo and Jerrum (2018). However it is based on a variant of the constructive version of the Lovász Local Lemma (which we cover later in the course).



Expectation vs. Whp

There are two typical kinds of guarantees we will work with.

- Expectation. This includes the expected running time, expected output, etc.
- With high probability (whp or w.h.p.). The meaning of this can vary. Sometimes it means probability $1-o(1)$, which for example would include $1-\frac{1}{\log n}$. Sometimes it is stronger, namely $1-\frac{1}{n^c}$.

Reading

Our next topic will be a review of some discrete probability, and the "Coupon Collector" problem.

- Some of you may have seen the "Coupon Collector" problem in lower level classes.
- We will re-visit it, but as well as deriving the expected value, we will also bound the variance (2nd moment), and look at the implications of that.
- You might want to read sections 2.3, 2.4 and 3.3 of [MU] in advance (if your probability is rusty, also read 2.1, 2.2, 3.1 and 3.2)