1 More on Randomization – Another Min-Cut Problem

- Previously, we saw an algorithm for the *minimum s-t cut problem*.
- Given a directed, weighted graph $G$ with vertices $s$ and $t$, find a partition of the vertices into sets $S$ and $V - S$, with $s \in S$ and $t \in V - S$, such that the total weight of all edges pointing from $s$ to $t$ is minimized.
- We’re now going to consider a related problem, *globally minimum cut*.
- This time, $G$ is an *undirected, unweighted* graph.
- We want to find a partition of $G$’s vertices into two nonempty sets $A$ and $B$, such that the number of edges that go between $A$ and $B$ is minimized.
- **Example:**

  - This problem is useful in *network reliability*: given a telecom network of nodes connected by bidirectional links, how many links can go down before the network is partitioned?

Can we reduce this problem to something we know how to solve already?

- First, convert $G$ to a weighted, directed graph $G'$.
- In particular, replace each undirected edge in $G$ by a pair of oppositely directed edges of weight 1.
- Now fix one vertex $s$ of $G'$.
- For each other vertex $t \neq s$, find a minimum $s$-$t$ cut in $G'$.
- Finally, return the minimum of all the cuts found, replacing each directed edge by the undirected edge that produced it.
- **Claim**: this algorithm returns a globally minimum cut of $G$. 
• (Proof is left as an exercise.)

This is a correct algorithm, but it’s ugly and quite slow – you have to do \( n - 1 \) rounds of a max-flow algorithm, which itself is expensive (at least, Ford-Fulkerson is). Is there a faster way?

## 2 Karger’s Algorithm

Here’s a faster, simpler algorithm:

• Karger (1993) came up with the following randomized algorithm, which is similar in spirit to Kruskal’s MST algorithm.

• Start with your undirected graph \( G \).

• Pick an edge \((u, v) \in G\) at random.

• Now contract the two endpoints of this edge into a single “supernode” \( w \).

• For each edge incident on either \( u \) or \( v \) (but not both), make it incident on \( w \) instead.

• If a given vertex \( x \) has edges to both \( u \) and \( v \), keep them both – the contracted graph is actually a multigraph.

• Keep contracting until you have exactly two vertices \( w_A \) and \( w_B \) left, such that \( A \) is the set of vertices contracted into \( w_A \), and \( B \) is the set contracted into \( w_B \).

• Finally, return the cut \( A, B \).

Unfortunately, this algorithm is usually wrong... but not always!

• **Thm**: For a graph \( G \) with \( n \) vertices, Karger’s algorithm returns a globally minimum cut of \( G \) with probability at least \( 1/n^2 \).

• **Pf**: Let \( A, B \) be a globally minimum cut of \( G \), and let \( F \) be the set of \( k \) edges crossing between \( A \) and \( B \).

• (Remember, \( |F| \) is how we define the size of the cut.)

• Karger’s algorithm fails to return \( A, B \) iff it ever contracts together a vertex in \( A \) (or a supernode containing one) with a vertex in \( B \) (ditto).
• This happens iff it ever randomly chooses an edge from $F$, since otherwise, nodes from $A$ and $B$ never get merged.

We claim that there are “plenty” of other edges in $G$ that are not in $F$, so that there’s a chance that the algorithm will leave $F$ alone.

• Because the minimum cut for $G$ has size $k$, every other cut of $G$ has size $\geq k$.

• In particular, every vertex $v \in V$ has degree at least $k$, because we can partition $V$ into $\{v\}$ and $V - \{v\}$, which is a cut.

• Conclude that $G$ has at least $nk/2$ edges ($k$ per vertex, but we counted each edge once for each of its two endpoints).

• Hence, our first edge choice does not fail with probability at least

$$\frac{nk/2 - k}{nk/2} = 1 - \frac{2}{n}.$$

• What about the situation in the multigraph $G'$ after the first contraction?

• Every cut in $G'$ corresponds to a cut in $G$ (just expand the supernode back to its vertices), so contraction cannot create a smaller cut than $F$ in $G'$.

• Hence, by the same argument as before, every vertex in $G'$ still has degree at least $k$.

• In fact, we can argue inductively that every vertex has degree at least $k$ in the multigraph remaining after any number of contractions.

• Conclude that after the $i$th contraction, the multigraph has at least $(n - i)k/2$ edges.

• Hence, the chance that the $i$th edge choice still does not fail, given that none of the previous ones did, is at least

$$\frac{(n - i)k/2 - k}{(n - i)k/2} = 1 - \frac{2}{n - i}.$$

• Hence, the chance that we select $n - 2$ edges (enough to contract down to 2 supernodes) without failing is at least

$$\prod_{i=0}^{n-3} \left(1 - \frac{2}{n - i}\right) = \prod_{i=0}^{n-3} \left(\frac{n - i - 2}{n - i}\right).$$

• Now if we write out this product, we notice that the numerator of each term cancels with the denominator two terms later:

$$\frac{n - 2}{n} \frac{n - 3}{n - 1} \frac{n - 4}{n - 2} \cdots \frac{3}{5} \frac{2}{4} \frac{1}{3}.$$

• After all these cancellations, the only terms remaining in the product are the leftmost two denominators and the rightmost two numerators, i.e.

$$\frac{2 \cdot 1}{n(n - 1)} = \frac{1}{\binom{n}{2}}.$$

QED
3 What Good is a Mostly Wrong Algorithm?

Karger’s algorithm is terrible – it only returns any particular globally minimum cut with probability $\epsilon = 1/O(n^2)$.

- **But** – it does have a nonzero chance of success!
- Moreover, this chance is nonzero for *any* graph, since it depends only on the algorithm’s random choices, not on the graph structure.
- So, what if we were to run the algorithm many times and return the best (i.e. smallest) answer?
- In particular, suppose we run it $\ell$ times.
- The chance that none of these $\ell$ trials returns a globally minimum cut is at most $(1 - \epsilon)^\ell$.
- Now suppose we are willing to accept a chance $\delta > 0$ that our answer is wrong, i.e. not optimal.
- Repeat the algorithm $\ell = \frac{1}{\epsilon} \ln \left( \frac{1}{\delta} \right)$ times.
- Then the chance of failure is at most

  $$(1 - \epsilon)^\ell \leq e^{-\epsilon \ell} = e^{-\ln(1/\delta)} = \delta.$$  

  where the first step uses the fact that $1 - x \leq e^{-x}$.

- Conclude that if we rerun the algorithm a *polynomial* number of times (that is, $O(\log 1/\delta)$), the chance of failure becomes exponentially small, i.e. $\delta$.
- We can never guarantee success, but we can make the chance of failure extremely remote – less than the chance that our computer spontaneously suffers hardware failure, or that a nearby supernova incinerates our planet, or...
- Hence, this algorithm is for all practical purposes “correct enough.”
- For any fixed $\delta$, we need run only $O(1/\epsilon) = O(n^2)$ times.

This is an example of a general strategy.

- If your randomized algorithm finds the optimum with probability $\epsilon > 0$, then repeating the algorithm $O(\log 1/\delta)$ times finds the optimum with probability at least $1 - \delta$.
- For fixed $\delta$, the number of trials is $O(1/\epsilon)$.
- Ok, but this was for optimization. What if you have a decision problem?
- Suppose you have an algorithm that returns the correct answer with probability at least $1/2 + \epsilon$. 


• (If it returns the correct answer less than half the time, then just say the opposite of what it tells you, and you’ll be right more than half the time.)

• Run the algorithm $\frac{4}{\epsilon^2} \ln \frac{1}{\delta}$ times, and take the majority vote among the answers.

• What’s the chance that you get the wrong answer at the end?

• Note that

$$\frac{\ell}{2} < \ell \left( \frac{1}{2} + \epsilon \left( \frac{1}{2} - \epsilon \right) \right)$$

$$= \ell(1 - \epsilon) \left( \frac{1}{2} + \epsilon \right).$$

• Let $t$ be the quantity on the last line. Applying the Chernoff bound, the chance that fewer than $t > \ell/2$ trials give the right answer is at most

$$e^{-\frac{1}{2}\epsilon^2 \left( \frac{1}{2} + \epsilon \right) \ell} = e^{-\frac{1}{2} \epsilon^2 \left( \frac{1}{2} + \epsilon \right) \frac{4}{\epsilon^2} \ln \frac{1}{\delta}}$$

$$= e^{(1 + 2\epsilon) \ln \delta}$$

$$= \delta^{1 + 2\epsilon}$$

$$< \delta.$$

• Once again, if we undertake $O(\log 1/\delta)$ trials, we get the right answer with probability at least $1 - \delta$. 