EECS 495: Randomized Algorithms	**Lecture 4**
Min Cut	

Reading: Text: Chapter 1, Chapter 10

Take-Aways

Concepts/Techniques:

- Las Vegas vs Monte Carlo
  - Las Vegas, last week: RandQS – *always* correct, running time variable
  - Monte Carlo, this week: Min-Cut – *always* fast, solution may be incorrect
- Boosting correctness with repetition
- Reusing computation to reduce running time

Application:

- Global min-cut

Simple Min-Cut Alg

Given an (undirected) graph $G = (V, E)$,

Def: A *multi-graph* is a graph with multiple edges between pairs of vertices.

Def: A *cut* in a graph is a set of edges whose removal results in G being broken into two or more connected components.

Def: A *minimum cut* is a cut of minimum cardinality.

Question: How can we find a min-cut?

Algorithm: Use LP-duality: min-s-t-cut equals max-s-t-flow, check all pairs of vertices.

Question: Can you describe the implementation fo this algorithm?

It's complicated!

Goal: Simple algorithm for min-cut.

Idea: Min cuts have few edges → contract random edges and hope to avoid hitting cut.

Algorithm: Randomized Global Min-Cut
Repeat

- Pick edge (u, v) uniformly at random
- Merge (*contract* endpoints (u, v) to get new meta-vertex $z_{u,v}$
- Delete self-loop but keep multi-edges

Until only 2 meta-vertices remain.
Output remaining edges.

Example: K_4 with one edge missing (square plus diagonal)

Claim: Alg outputs global min-cut with probability at least $1/\binom{n}{2}$.

Proof: Consider min-cut $F \subseteq E$ with $|F| = k$.

Fact: Degree of any vertex is at least k.

$\left[\left[\textit{Else cut out vertex of smaller degree to get smaller cut.}\right]\right]$

Let $\mathcal{E}_i$ be event that edge contracted in step i is *not* in min-cut.

[[*So these are the GOOD events for us.*]]

Question: $\Pr[\neg\mathcal{E}_1] = ?$

- by randomization, $k/|E|$
- twice # edges is sum of degrees, so by fact $|E| \geq nk/2$

Therefore, $\Pr[\neg\mathcal{E}_1] \leq k/(nk/2) = 2/n$.

Question: $\Pr[\neg\mathcal{E}_i|\mathcal{E}_1,\ldots,\mathcal{E}_{i-1}] = ?$

Idea: So long as we don't contract an edge from F, F is a min-cut in contracted graph

$\left[\left[\textit{Because all cuts in contracted graph are cuts in original graph.}\right]\right]$

Let G_i be contracted graph in i'th step given $\mathcal{E}_1,\ldots,\mathcal{E}_{i-1}$:

- number vertices is $(n-i+1)$
- min-cut in G_i has size $= k$ (assuming $\mathcal{E}_1,\ldots,\mathcal{E}_{i-1}$)
- number of edges is $\geq (n-i+1)k/2$

Therefore, $\Pr[\neg\mathcal{E}_i|\mathcal{E}_1,\ldots,\mathcal{E}_{i-1}] \leq 2/(n-i+1)$.

Recall conditional probability:

Def: The *conditional probability* of $\mathcal{E}_1$ given $\mathcal{E}_2$ is

$$\Pr[\mathcal{E}_1|\mathcal{E}_2] = \Pr[\mathcal{E}_1 \wedge \mathcal{E}_2]/\Pr[\mathcal{E}_2]$$

Fact: For any set of (possibly dependent) events, the above gives

$$\Pr[\wedge_{i=1}^n \mathcal{E}_i] = \Pr[\mathcal{E}_1]\Pr[\mathcal{E}_2|\mathcal{E}_1]\ldots\Pr[\mathcal{E}_n|\wedge_{i=1}^{n-1}\mathcal{E}_i]$$

Question: What's probability all good contractions?

$$\begin{aligned}
\Pr[\text{GOOD}] &= \Pr[\mathcal{E}_1 \wedge \ldots \wedge \mathcal{E}_{n-2}] \\
&= (1-\Pr[\neg\mathcal{E}_1])(1-\Pr[\neg\mathcal{E}_2|\mathcal{E}_1)\ldots \\
&\geq (1-\frac{2}{n})(1-\frac{2}{n-1})(1-\frac{2}{n-2})\ldots(1-\frac{2}{3}) \\
&= (\frac{n-2}{n})(\frac{n-3}{n-1})(\frac{n-4}{n-2})\ldots(\frac{1}{3}) \\
&= \frac{2\cdot 1}{n\cdot(n-1)} \\
&= \frac{1}{\binom{n}{2}}
\end{aligned}$$

Question: How to improve correctness?

Idea: Run n^2 times:

$$\Pr[\text{MEGA GOOD}] = 1 - \Pr[\text{MEGA BAD}]$$

$$\Pr[\text{MEGA BAD}] \leq (1-\frac{1}{n^2})^{n^2} = \frac{1}{e}$$

(by independence)

[[*Run $n^2 \log n$ times to get w.h.p.*]]

Question: Running time?

- n loops
- n time per loop
- run alg $n^2 \log n$ times

so $O(n^4 \log n)$.

Improving Running Time

Recall **Algorithm:** Use LP-duality: min-s-t-cut equals max-s-t-flow, check all pairs of vertices.

Question: Running time?

- one max-flow: $O(mn\log(n^2/m))$
- naive implementation: $O(mn^3\log(n^2/m))$
- better, can use just $(n-1)$ flows: $O(mn^2\log(n^2/m))$

- better yet, one max flow suffices: $O(mn \log(n^2/m))$

Note: Max flow yields min cut, but not clear how to go the other way.

Question: Can we solve min-s-t-cut faster than max-s-t-flow?

Don't know, but dropping s-t requirement,

Claim: Careful modification of global min-cut runs in time $O(n^2 \log^{O(1)} n)$, much faster than max flow for dense graphs.

Idea: Alg unlikely to mess up at the beginning:

- $\Pr[\neg \mathcal{E}_1] \leq 2/n$

so repeating first step n^2 times is wasteful!

Claim: Suppose terminate alg when t vertices remain (as opposed to 2). Then min-cut survives with prob. $\geq$

$$\binom{t}{2} / \binom{n}{2} = \Omega((t/n)^2)$$

[[*Just modify previous calculation.*]]

Idea: Contract until t vertices remain, then use deterministic alg.
Problem: Deterministic alg complicated and too slow.

Idea: Use *two* invocations of randomized alg!

Algorithm: FastCut$(G(V, E))$

- $n \leftarrow |V|$
- if $n \leq 6$, compute min-cut of G by brute-force
- else
  - $t \leftarrow \lceil (1 + n/\sqrt{2}) \rceil$
  - perform two independent contraction-sequences to get two graphs H_1 and H_2 with t vertices each
  - recursively compute min-cuts in H_1 and H_2
  - return smaller cut

Intuition: binary computation tree.

Draw tree, root is G, for node H children are H_1 and H_2.

Question: How many levels? about $2 \log n$

Question: How many leaves? about n^2

[[*Note computation tree of simple alg is like a star with n^2 leaves; hence speedup not from solving fewer problems, but from sharing work.*]]

Claim: Alg runs in time $O(n^2 \log n)$. **Proof:** Recurrence is:

$$T(n) = 2T(\lceil (1 + n/\sqrt{2}) \rceil) + O(n^2)$$

and soln is as above.

Claim: Alg is correct with probability $\Omega(1/\log n)$.

[[*Hence can repeat $\log n$ times to get constant probability of success at expense of factor $\log n$ in running time.*]]

Proof: Suppose min-cut of G has size k.

Recursive call on H returns correct answer if

- cut of size k survives to H_1 (or H_2)
- AND recursive call on H_1 (or H_2) is correct

Question: What's prob. cut survives to H_1?

$$\lceil (1 + t/\sqrt{2}) \rceil^2 / t^2 \geq 1/2$$

by claim.

Let $P(t)$ denote prob. alg succeeds for graph with t vertices:

$$P(t) \geq 1 - (1 - \frac{1}{2}P(\lceil(1 + t/\sqrt{2})\rceil))^2$$

Change of variables: Let $k = \Theta(\log t)$ be depth of recursion, $p(k)$ be lower-bound on success prob.

- $p(0) = 1$
- $p(k+1) = p(k) - p(k)^2/4$ (subbing in to above eqn)

Further change of variables: $q(k) = 4/p(k) - 1$ or $p(k) = 4/(q(k) + 1)$

$$q(k+1) = q(k) + 1 + \frac{1}{q(k)}$$

Soln (prove by induction):

$$k < q(k) < k + H_{k-1} + 3$$

where H_i is i'th harmonic number.

Therefore:

$$q(k) = k + \Theta(\log k) \rightarrow p(k) = \Theta(1/k) \rightarrow P(t) = \Theta(1/\log t)$$

and results follows using $t = n$.

$\left[\left[\right.\right.$ *Like branching processes, trying to see prob. min-cut dies out in this birth process.* $\left.\left.\right]\right]$