Design and Analysis of Algorithms I

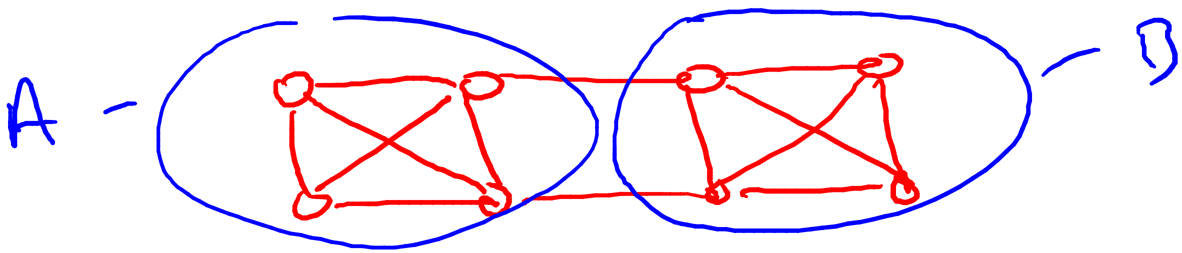
Contraction Algorithm

The Analysis

The Minimum Cut Problem

Input: An undirected graph G = (V, E).
[parallel edges  allowed]
[See other video for representation of input]

Goal: Compute a cut with fewest number of crossing edges.
(a min cut)



Random Contraction Algorithm

[due to Karger, early 90s]

While there are more than 2 vertices:

- pick a remaining edge (u,v) uniformly at random
- merge (or "contract") u and v into a single vertex
- remove self-loops

return cut represented by final 2 vertices.

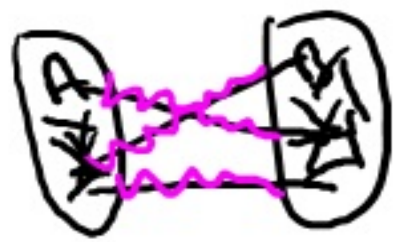
The Setup

Question: what is the probability of success?

Fix a graph G = (V, E) with n vertices, m edges.

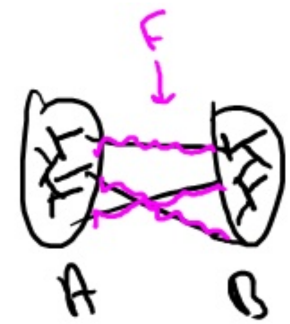
Fix a minimum cut (A, B).

Let k = # of edges crossing (A, B). (Call these edges F)



What Could Go Wrong?

1. Suppose an edge of F is contracted at some point $\Rightarrow$ algorithm will not output (A,B).



2. Suppose only edges inside A or inside B get contracted $\Rightarrow$ algorithm will output (A, B).

Thus: Pr [output is (A, B)] = Pr [never contracts an edge of F]

Let S_i = event that an edge of F contracted in iteration i.

Goal: Compute $\Pr[\neg S_1 \wedge \neg S_2 \wedge \neg S_3 \wedge \ldots\ldots \wedge \neg S_{n-2}]$

Ter
ver

What is the probability that an edge crossing the minimum cut (A, B) is chosen in the first iteration (as a function of the number of vertices n, the number of edges m, and the number k of crossing edges)?

- k/n
- k/m
- k/n^2
- n/m

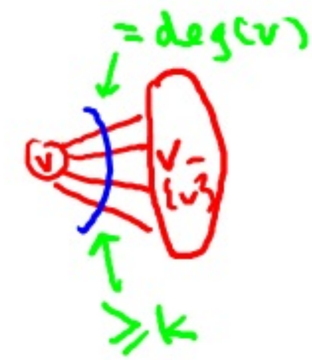
$$\Pr[S_1] = \frac{\#\text{ of crossing edges}}{\#\text{ of edges}} = \frac{k}{m}$$

each iteration, the decreased edge can be more than one, it is not stable,
so, we use vertices to represent the probability in the following slides….

The First Iteration

<u>Key Observation</u>: $\underbrace{\text{degree}}_{\text{\# of incident edges}}$ of each vertex is at least k

<u>Reason</u>: each vertex v defines a cut ({v}. V-{v}).



Since $\underbrace{\sum_v degree(v)}_{\geq kn} \overset{\text{for undirected graph}}{=} 2m,$ we have $m \geq \frac{kn}{2}$

Since $\Pr[S_1] = \frac{k}{m}, \Pr[S_1] \leq \frac{2}{n}$

The Second Iteration

<u>Recall:</u> $\Pr[\neg S_1 \wedge \neg S_2] = \underbrace{\Pr[\neg S_2 | \neg S_1]}_{= 1 - \frac{k}{\text{\# of remaining edge}}} . \Pr[\neg S_1]$ $\geq (1 - \frac{2}{n})$

what is this?

Note: all nodes in contracted graph define cuts in G (with at least k crossing edges).

➢ all degrees in contracted graph are at least k

So: # of remaining ec $\geq \frac{1}{2}k(n-1)$

So $\Pr[\neg S_2 | \neg S_1] \geq 1 - \frac{2}{(n-1)}$

All Iterations

In general:

$$\Pr[\neg S_1 \wedge \neg S_2 \wedge \neg S_3 \wedge .... \wedge \neg S_{n-2}]$$

$$= \Pr[\neg S_1]\, \Pr[\neg S_2 | \neg S_1]\, \Pr[\neg S_3 | \neg S_2 \wedge \neg S_1] ...... \Pr[\neg S_{n-2} | \neg S_1 \wedge ... \wedge \neg S_{n-3}]$$

$$\geq (1 - \tfrac{2}{n})(1 - \tfrac{2}{n-1})(1 - \tfrac{2}{n-2}) .....(1 - \tfrac{2}{n-(n-4)})(1 - \tfrac{2}{n-(n-3)})$$

$$= \frac{\cancel{n-2}}{n} \cdot \frac{\cancel{n-3}}{n-1} \cdot \frac{\cancel{n-4}}{\cancel{n-2}} ...... \frac{2}{\cancel{4}} \cdot \frac{1}{\cancel{3}} = \frac{2}{n(n-1)} \geq \frac{1}{n^2}$$

Problem: low success probability! (But: non trivial)

recall $\simeq 2^n$ cuts !

So, brute-force method will result in Pr = 1/(2^n). ref: overview, slide 5

Tim Roughgarden

Repeated Trials

Solution: run the basic algorithm a large number N times, remember the smallest cut found.

Question: how many trials needed?

Let T_i = event that the cut (A, B) is found on the i^{th} try.

➢ by definition, different T_i's are independent

So: $$\Pr[\text{all N trails fail}] = \Pr[\neg T_1 \wedge \neg T_2 \wedge \ldots \wedge \neg T_N]$$

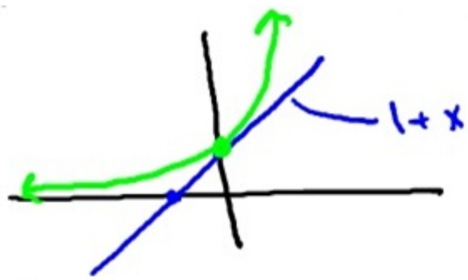
$$= \prod_{i=1}^{N} \Pr[\neg T_i] \leq \left(1 - \frac{1}{n^2}\right)^N$$

By independence !

Repeated Trials (con'd)

Pr[all trials fail] $\leq (1 - \frac{1}{n^2})^N$

Calculus fact: $\forall$real numbers x, 1+x $\leq e^x$



So: if we take $N = n^2, \Pr[\text{all fail}] \leq (e^{-\frac{1}{n^2}})^{n^2} = \frac{1}{e}$

If we take $N = n^2 \ln n, \Pr[\text{all fail}] \leq (\frac{1}{e})^{\ln n} = \frac{1}{n}$

n^2 is the number of times we execute,
m is the calculation we need each execution

Running time: polynomial in n and m but slow $(\Omega(n^2 m))$

But: can get big speed ups (to roughly $O(n^2)$) with more ideas.