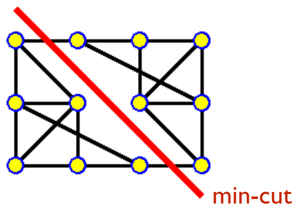
15-251
Great Ideas in Theoretical Computer Science

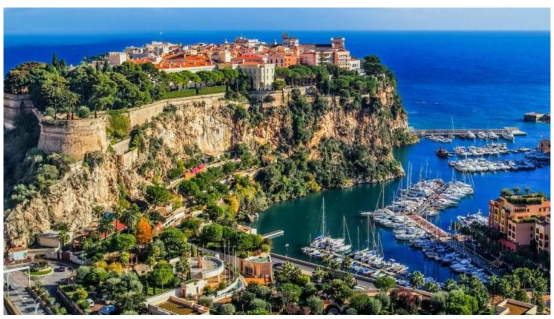
Lecture 24:
Randomized Algorithms 2

November 16th, 2017



CASE STUDY

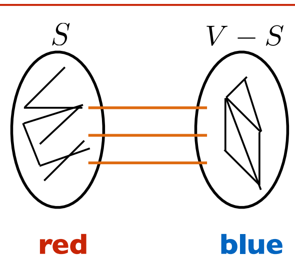
Monte Carlo Algorithm for Min Cut



Gambles with **correctness**.
Doesn't gamble with **run-time**.

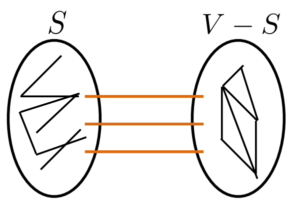
Cut Problems

Max Cut Problem (Ryan O'Donnell's favorite problem):
Given a connected graph $G = (V, E)$,
color the vertices **red** and **blue** so that the number of edges with two colors (e = {**u**,**v**}) is maximized.



Cut Problems

Max Cut Problem (Ryan O'Donnell's favorite problem):
Given a connected graph $G = (V, E)$,
find a non-empty subset $S \subset V$ such that
number of edges from S to $V - S$ is maximized.



size of the cut = # edges from S to $V - S$.

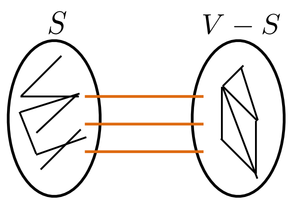
Max Cut Problem is **NP**-hard!

Cut Problems

Randomized Approximation for Max Cut

Cut Problems

Min Cut Problem (my favorite problem):
Given a connected graph $G = (V, E)$,
find a non-empty subset $S \subset V$ such that
number of edges from S to $V - S$ is ***minimized***.



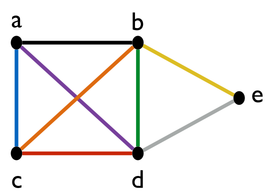
size of the cut = # edges from S to $V - S$.

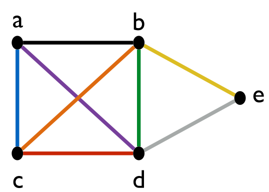
(how many possible "cuts" are there?)

Randomized Algorithm for Min Cut
(contraction algorithm)

Contraction algorithm for min cut

Example run 1





Select an edge randomly:

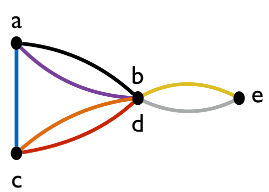
{b,d} selected

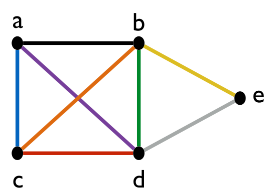
Contract that edge.

Size of min-cut: 2

Contraction algorithm for min cut

Example run 1





Select an edge randomly:

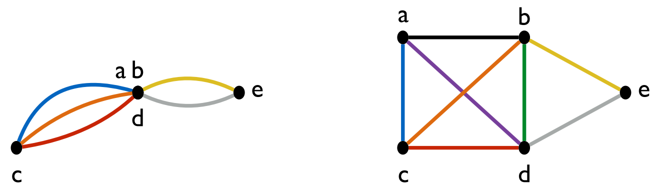
{a, d} selected

Contract that edge. (delete self loops)

Size of min-cut: 2

Contraction algorithm for min cut

Example run 1



Size of min-cut: 2

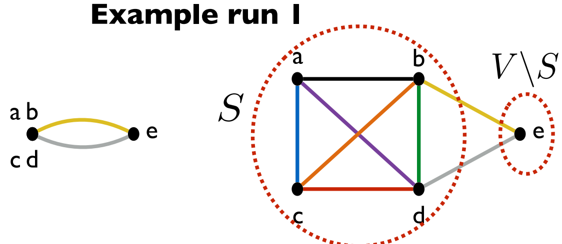
Select an edge randomly:

{c, a} selected

Contract that edge. (delete self loops)

Contraction algorithm for min cut

Example run 1



Size of min-cut: 2

When two vertices remain, you have your cut:

S = {a, b, c, d} V\S = {e} *size:* 2

Contraction algorithm for min cut

$$G = G_0 \xrightarrow{\text{contract}} G_1 \xrightarrow{\text{contract}} G_2 \xrightarrow{\text{contract}} \cdots \xrightarrow{\text{contract}} G_{n-2}$$

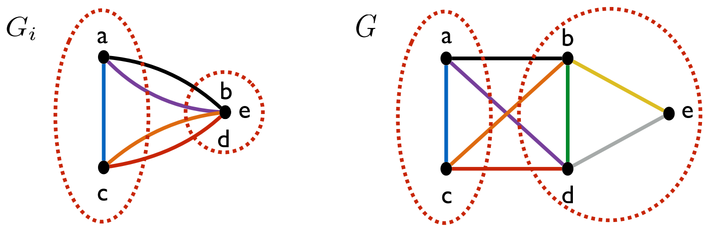
n vertices 2 vertices

$n-2$ iterations

Contraction algorithm for min cut

Observation:

For any i: A cut in G_i of size k corresponds exactly to a cut in G of size k.



Contraction algorithm for min cut

Theorem:
Let $G = (V, E)$ be a graph with *n* vertices.
The probability that the contraction algorithm will output a min-cut is $\geq 1/n^2$.

Should we be impressed?

- The algorithm runs in polynomial time.
- There are exponentially many cuts. ($\approx 2^n$)
- There is a way to boost the probability of success to $1 - \frac{1}{e^n}$ (and still remain in polynomial time)

Proof of Theorem

Pre-proof Poll

Let k be the size of a minimum cut.

Which of the following are true (can select more than one):

For $G = G_0$, $k \leq \min_v \deg_G(v)$ $(\forall v,\ k \leq \deg_G(v))$

For $G = G_0$, $k \geq \min_v \deg_G(v)$

For every G_i, $k \leq \min_v \deg_{G_i}(v)$ $(\forall v,\ k \leq \deg_{G_i}(v))$

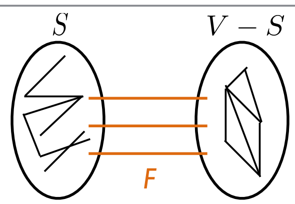
For every G_i, $k \geq \min_v \deg_{G_i}(v)$

Poll answer

Proof of theorem

Fix some minimum cut.

$|F| = k$
$|V| = n$
$|E| = m$



Will show $\Pr[\text{algorithm outputs } F] \geq 1/n^2$

(Note $\Pr[\text{success}] \geq \Pr[\text{algorithm outputs } F]$)

Proof of theorem

Proof of theorem

Proof of theorem

Proof of theorem

Proof of theorem

Contraction algorithm for min cut

Theorem:

Let $G = (V, E)$ be a graph with *n* vertices.
The probability that the contraction algorithm will output a min-cut is $\geq 1/n^2$.

Should we be impressed?

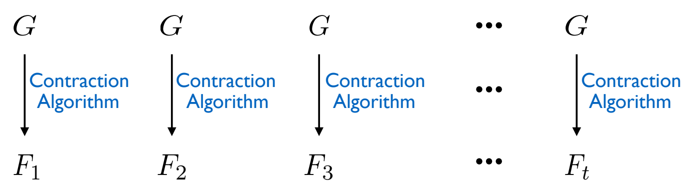
- The algorithm runs in polynomial time.
- There are exponentially many cuts. ($\approx 2^n$)
- There is a way to boost the probability of success to $1 - \frac{1}{e^n}$ (and still remain in polynomial time)

Boosting Phase

(and the world's greatest approximation!)

Boosting phase

Run the algorithm ***t*** times using fresh random bits.



Output the minimum among F_i's.

larger $t \implies$ better success probability

What is the relation between t and success probability?

Boosting phase

What is the relation between t and success probability?

Let A_i = "in the i'th repetition, we **don't** find a min cut."

$$\Pr[\text{error}] = \Pr[\text{don't find a min cut}]$$

$$= \Pr[A_1 \cap A_2 \cap \cdots \cap A_t]$$

ind. events

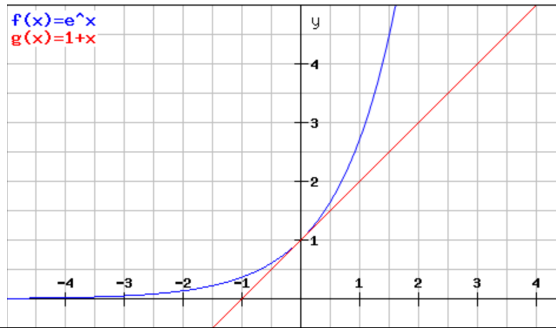
$$= \Pr[A_1] \Pr[A_2] \cdots \Pr[A_t]$$

$$= \Pr[A_1]^t \leq \left(1 - \frac{1}{n^2}\right)^t$$

Boosting phase

$$\Pr[\text{error}] \leq \left(1 - \frac{1}{n^2}\right)^t$$

World's most useful inequality: $\forall x \in \mathbb{R}: \ 1 + x \leq e^x$



Boosting phase

$$\Pr[\text{error}] \leq \left(1 - \frac{1}{n^2}\right)^t$$

World's most useful inequality: $\forall x \in \mathbb{R}: \ 1 + x \leq e^x$

Let $x = -1/n^2$

$$\Pr[\text{error}] \leq (1+x)^t \leq (e^x)^t = e^{xt} = e^{-t/n^2}$$

$$t = n^3 \implies \Pr[\text{error}] \leq e^{-n^3/n^2} = 1/e^n \implies$$

$$\Pr[\text{success}] \geq 1 - \frac{1}{e^n}$$

Conclusion for min cut

We have a polynomial-time algorithm that solves the min cut problem with probability $1 - 1/e^n$.

Theoretically, not equal to 1.
Practically, equal to 1.

Important Note

Boosting is not specific to Min-cut algorithm.

We can boost the success probability of Monte Carlo algorithms via repeated trials.

Final remarks

Randomness adds an interesting dimension to computation.

Randomized algorithms can be faster and more elegant than their deterministic counterparts.

There are some interesting problems for which:

- there is a poly-time randomized algorithm,
- we can't find a poly-time deterministic algorithm.