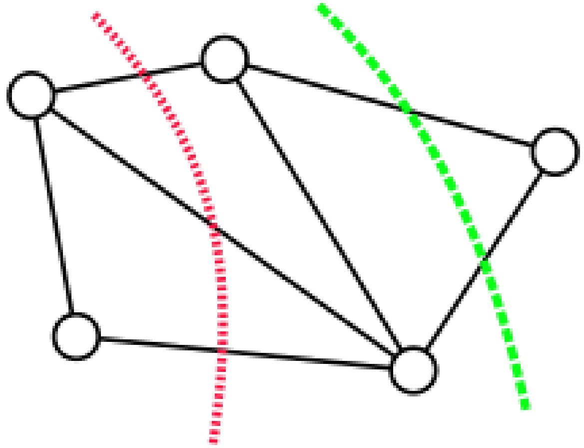
Lecture 8 (Adv): Karger's algorithms



THE UNIVERSITY OF SYDNEY

Randomization

– Algorithmic design patterns.
  – Greed.
  – Divide-and-conquer.
  – Dynamic programming.
  – Network flow.
  – Randomization.

in practice, access to a pseudo-random number generator

– Randomization: Allow fair coin flip in unit time.

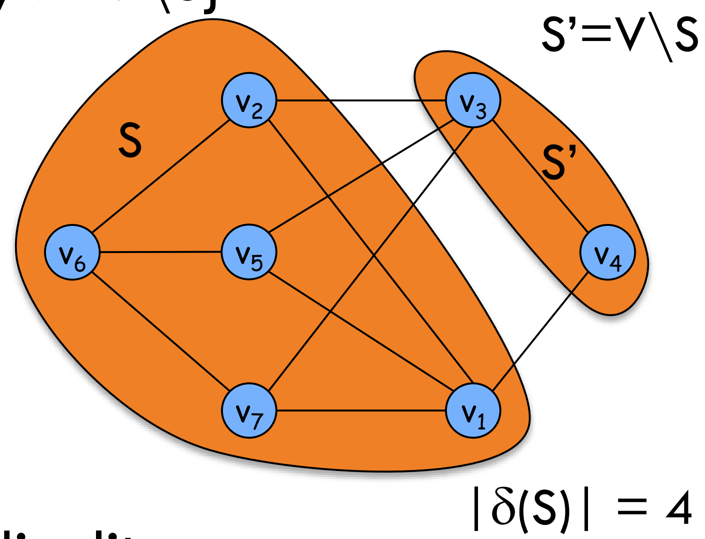
– Why randomize? Can lead to simplest, fastest, or only known algorithm for a particular problem.

– Examples: Symmetry breaking protocols, graph algorithms, quicksort, hashing, load balancing, Monte Carlo integration, cryptography.

13.2 Global Minimum Cut

Input: A connected, undirected graph $G = (V, E)$.

For a set $S \subset V$ let $\delta(S) = \{(u,v) \in E : u \in S, v \in V \setminus S\}$.

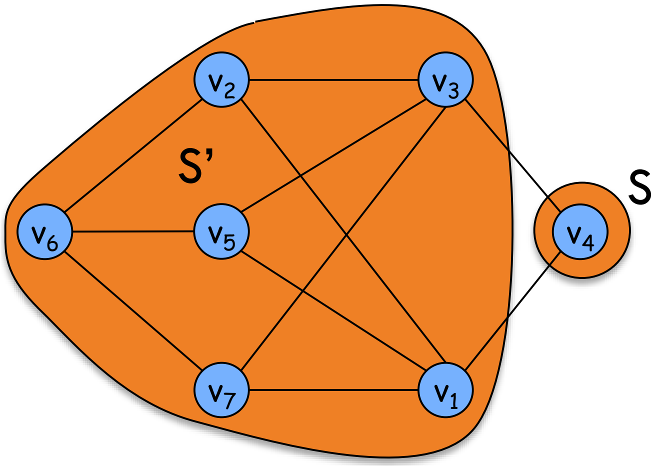


Aim: Find a cut (S, S') of minimum cardinality.

13.2 Global Minimum Cut

Input: A connected, undirected graph $G = (V, E)$.

For a set $S \subset V$ let $\delta(S) = \{(u,v) \in E : u \in S, v \in V \setminus S\}$.



$|\delta(S)| = 2$

Aim: Find a cut (S, S') of minimum cardinality.

13.2 Global Minimum Cut

Applications: Partitioning items in a database, identify clusters of related documents, network reliability, network design, circuit design, TSP solvers.

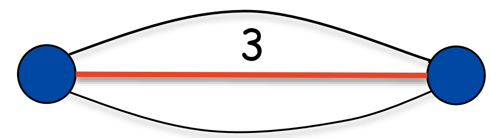
Network flow solution.

- Replace every edge (u, v) with two directed edges (u, v) and (v, u).
- Pick some vertex s and compute min s-v cut separating s from each other vertex $v \in V$.

Running time: $O((n-1) \cdot \text{MaxFlows})$

Karger's Contraction Algorithm

Definition: A multigraph is a graph that allows multiple edges between a pair of vertices.



Karger’s Contraction Algorithm

Definition: A multigraph is a graph that allows multiple edges between a pair of vertices.

Algorithm:

1. Start with the input graph G=(V,E).
2. While |V|>2 do

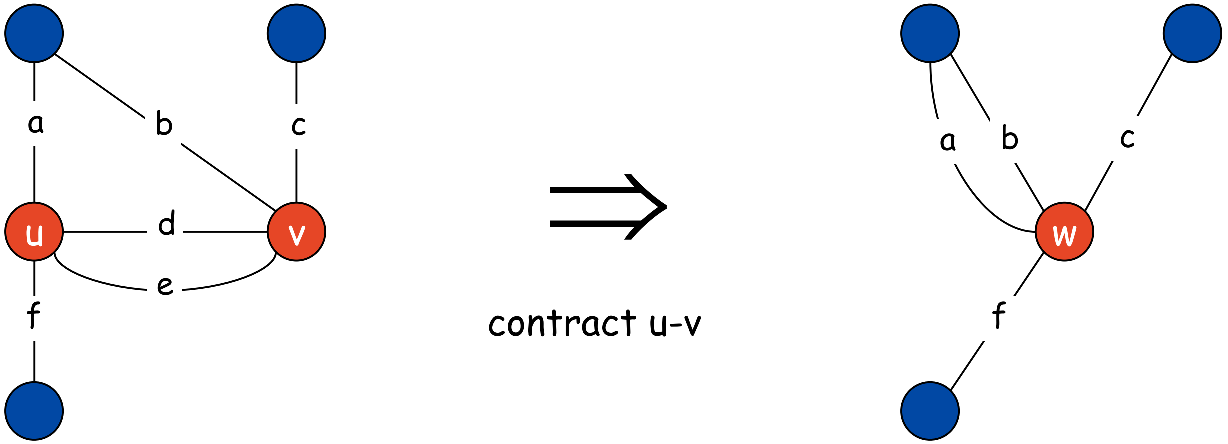
   Contract an arbitrary edge (u,v) in G.
3. Return the cut (only one possible cut).

Karger's Contraction Algorithm

Let G=(V,E) be a multigraph (without self-loops).

Contract an edge $e=(u,v) \in E \quad \Rightarrow \quad G \backslash e$

- Replace u and v by single new super-node w
- Replace all edges (u,x) or (v,x) with an edge (w,x)
- Remove self-loops to w.



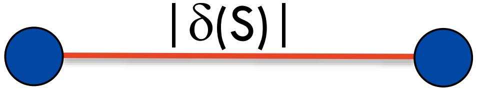
Karger's Contraction Algorithm

Definition: A multigraph is a graph that allows multiple edges between a pair of vertices.

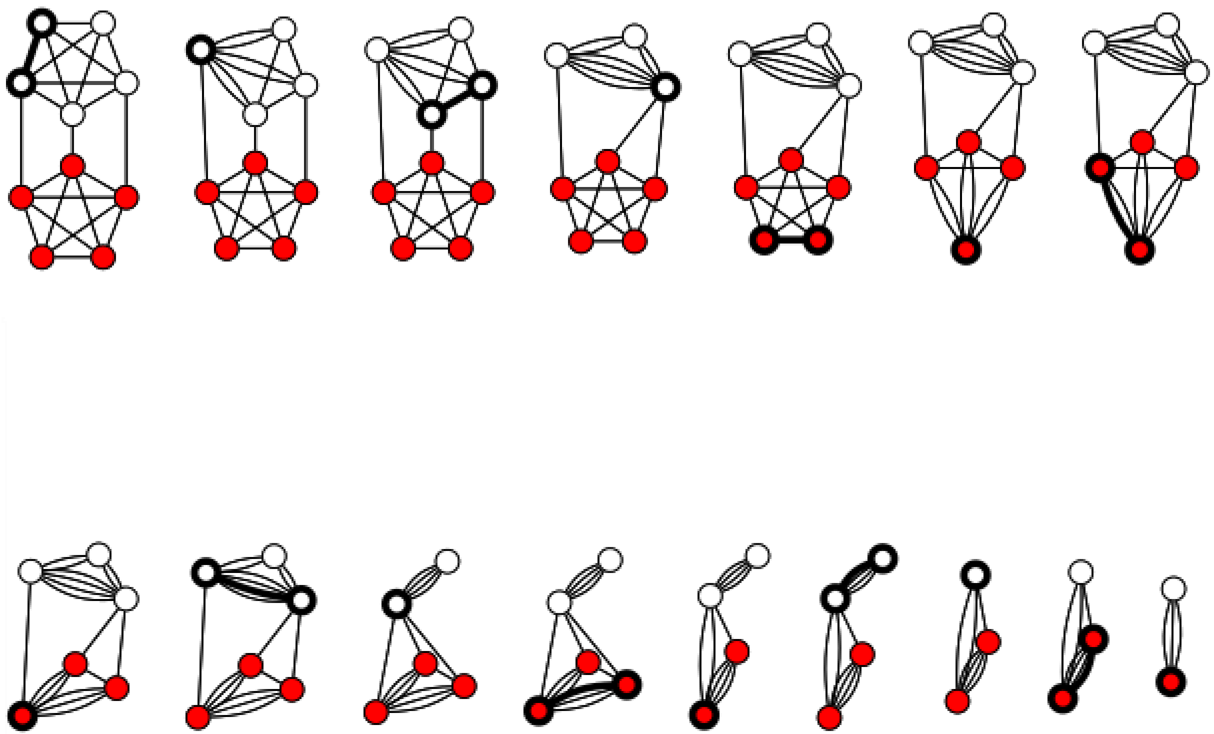
Algorithm:

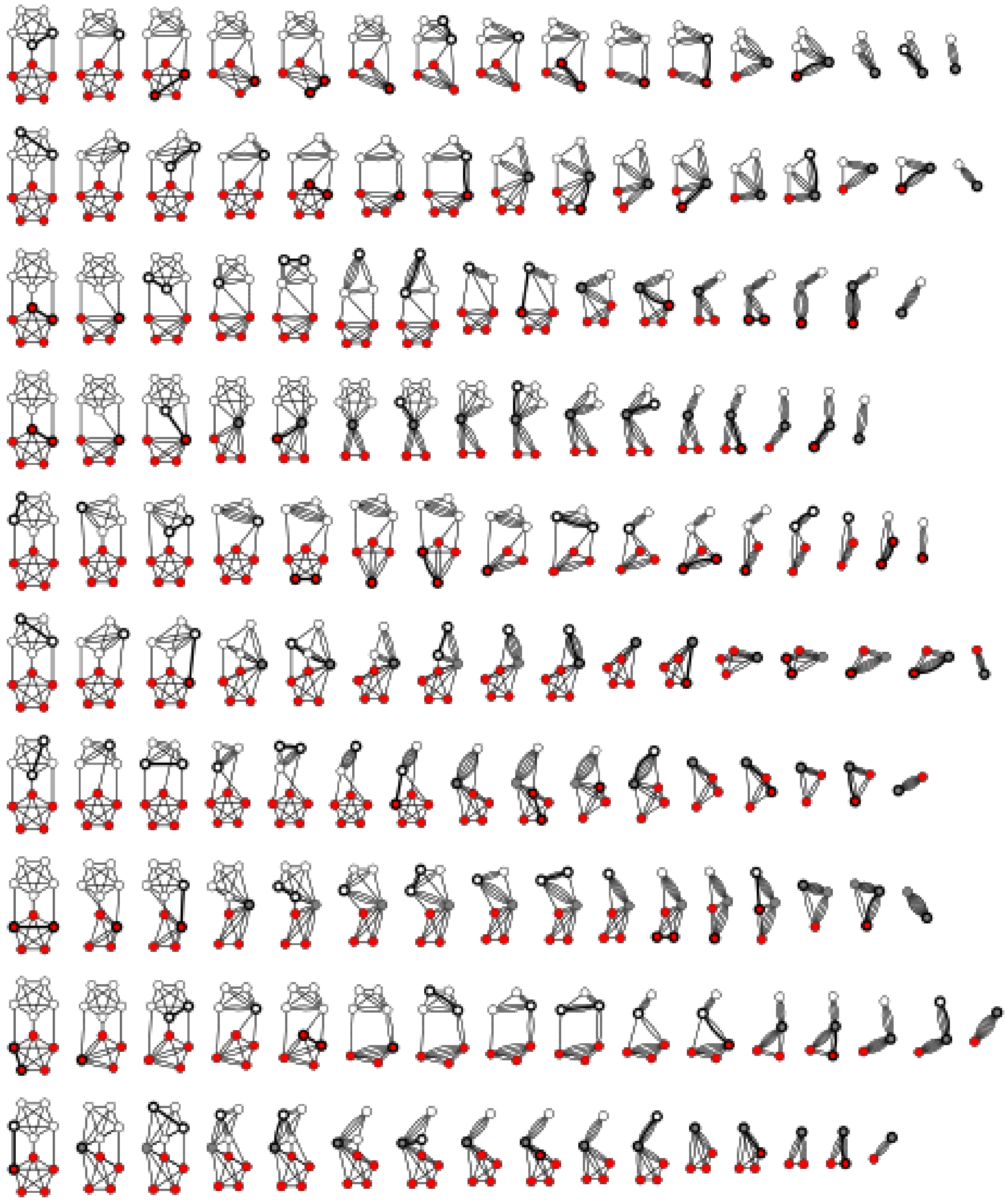
1. Start with the input graph G=(V,E).
2. While |V|>2 do

   Contract an arbitrary edge (u,v) in G.
3. Return the cut (only one possible cut).



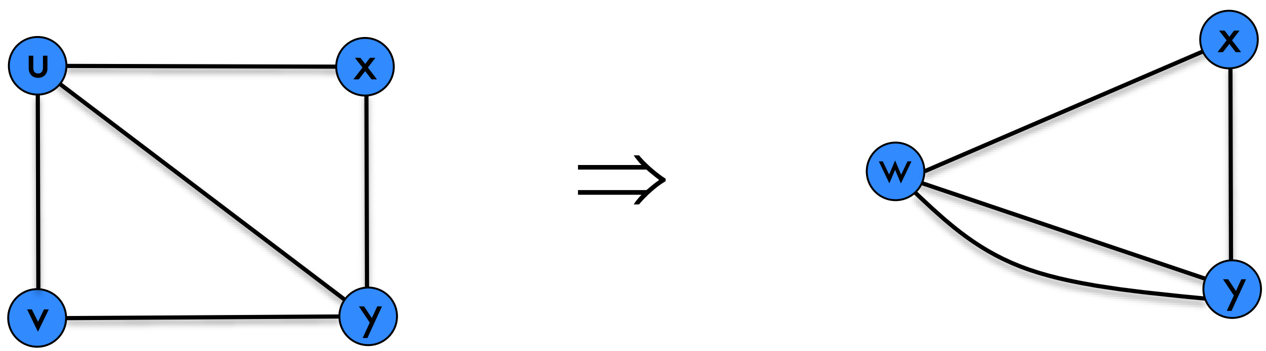
Karger's contraction algorithm





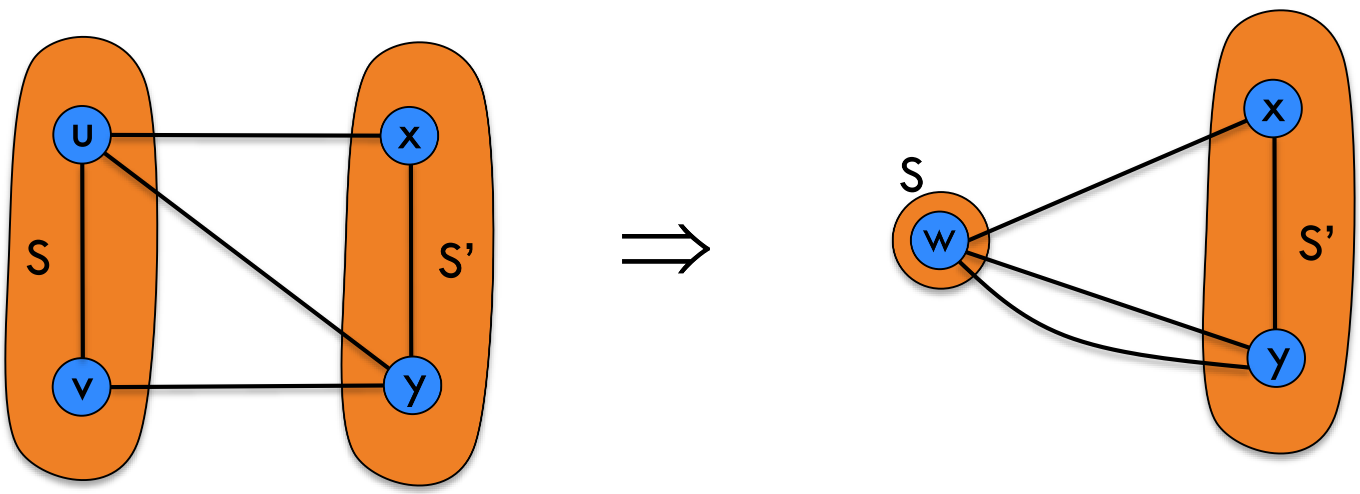
Karger's Contraction Algorithm

Observation: An edge (u,v) contraction preserves those cuts where u and v are both in S or in S'.



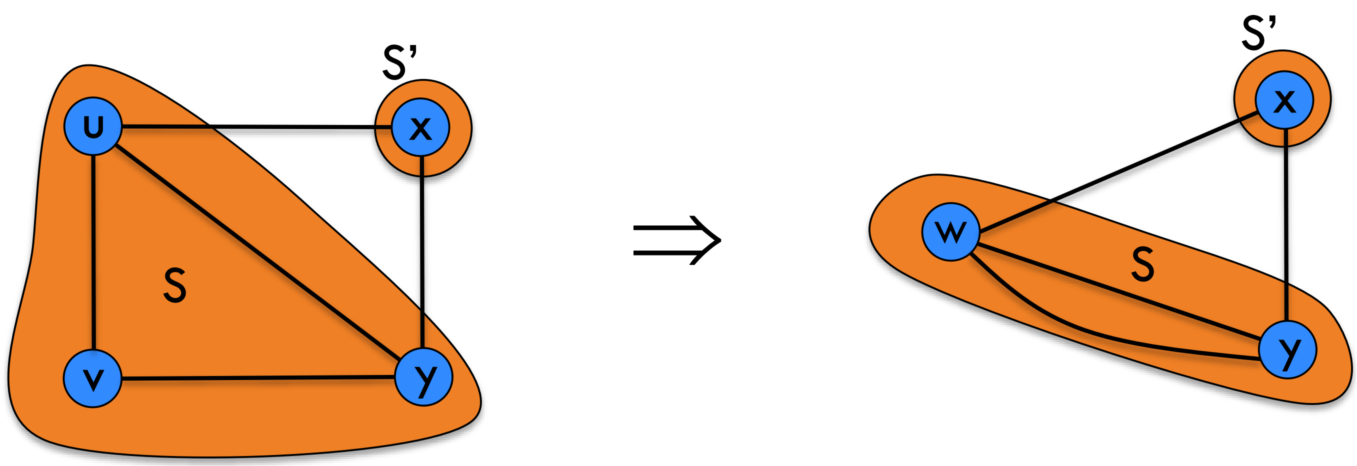
Karger's Contraction Algorithm

Observation: An edge (u,v) contraction preserves those cuts where u and v are both in S or in S'.



Karger's Contraction Algorithm

Observation: An edge (u,v) contraction preserves those cuts where u and v are both in S or in S'.



If u,v$\in$S then $\delta_G(S) = \delta_{G\backslash e}(S)$.
(with u and v replaced with w)

Algorithm: General idea

– Contract n-2 edges $\Rightarrow$ two vertices remain in G'

Algorithm: General idea

– Contract n-2 edges $\Rightarrow$ two vertices remain in G’

– The two vertices in G’ correspond to a partition (S,S’) in G.

Algorithm: General idea

– Contract n-2 edges $\Rightarrow$ two vertices remain in G'

– The two vertices in G' correspond to a partition (S,S') in G.

– The edges remaining in G' corresponds to $\delta_G(S)$.

– Output $\delta_G(S)$.

If we never contract edges from a minimal cut $\delta(S^*)$ then the algorithm will report $\delta(S^*)$.

How do we select the edges?

Karger's Contraction Algorithm

Algorithm:

1. Start with the input graph G=(V,E).
2. While |V|>2 do

   Contract an arbitrary edge (u,v) in G.
3. Return the cut S (only one possible cut).

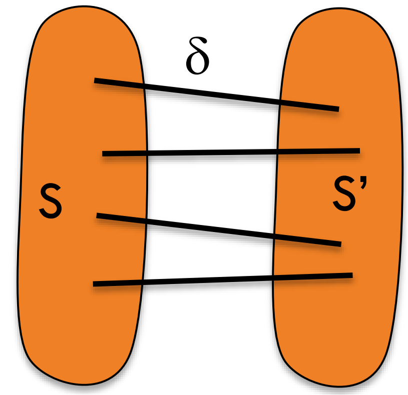
Algorithm: Since S* is a minimum cut it has few edges!

Claim: This algorithm has a reasonable chance of finding a minimal cut.

Prove the claim

Claim: The algorithm returns a minimal cut with probability $\geq 2/n^2$.

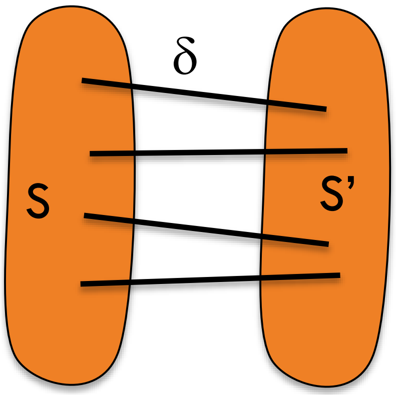
Proof: Consider a global min cut (S,S') of G. Let δ be edges with one endpoint in S and the other in S'.
Let $k = |\delta|$ = size of the min cut.



Prove the claim

Claim: The algorithm returns a minimal cut with probability $\geq 2/n^2$.

Proof: Consider a global min cut (S,S') of G. Let δ be edges with one endpoint in S and the other in S'.
Let $k = |\delta|$ = size of the min cut.



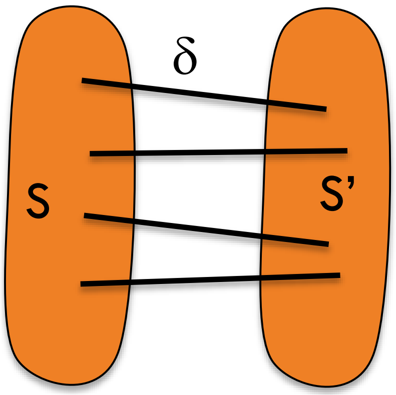
Step 1: contract an edge in δ with probability $k/|E|$.

Size of E?

Prove the claim

Claim: The algorithm returns a minimal cut with probability $\geq 2/n^2$.

Proof: Consider a global min cut (S,S') of G. Let δ be edges with one endpoint in S and the other in S'.
Let $k = |\delta|$ = size of the min cut.



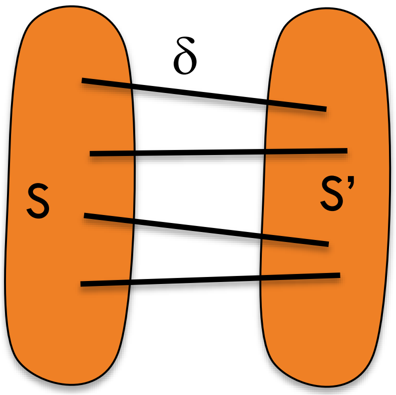
Step 1: contract an edge in δ with probability $k/|E|$.

Every node has degree $\geq k$
otherwise (S,S') would not be min-cut.
$\Rightarrow \ |E| \geq \frac{1}{2}kn$.

Prove the claim

Claim: The algorithm returns a minimal cut with probability $\geq 2/n^2$.

Proof: Consider a global min cut (S,S') of G. Let δ be edges with one endpoint in S and the other in S'.
Let $k = |\delta|$ = size of the min cut.



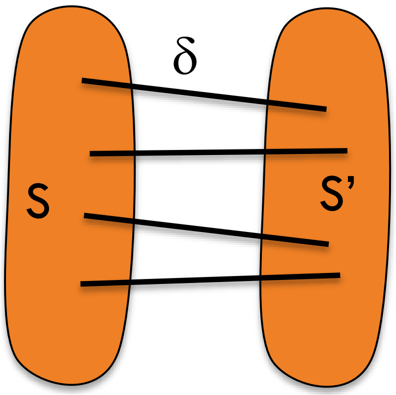
Step 1: contract an edge in δ with probability $k/|E|$.

with probability $\leq 2/n$.

Prove the claim

Claim: The algorithm returns a minimal cut with probability $\geq 2/n^2$.

Proof: Consider a global min cut (S,S') of G. Let δ be edges with one endpoint in S and the other in S'.
Let $k = |\delta|$ = size of the min cut.



Step 1: contract an edge in δ with probability $2/n$.

Observation:

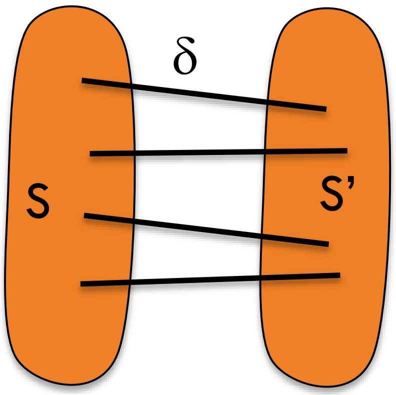
The minimum degree in any (intermediate) multigraph is at least k.
(Otherwise there would be a smaller cut)

Specifically this means that if an intermediate multigraph has n' vertices, it will have at least $n' \cdot k/2$ edges.

Prove the claim

Claim: The algorithm returns a minimal cut with probability $\geq 2/n^2$.

Proof: Consider a global min cut (S,S') of G. Let δ be edges with one endpoint in S and the other in S'.
Let $k = |\delta|$ = size of the min cut.



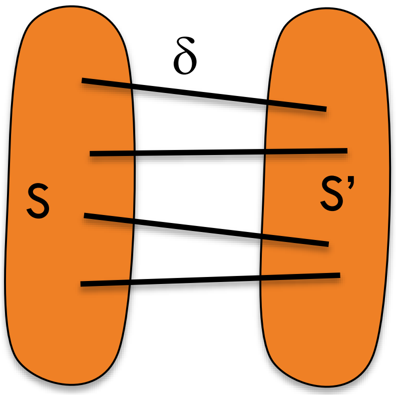
Step 1: contract an edge in δ with probability $2/n$.

After step i: The multigraph G_i has n-i vertices
and at least $(n-i)\cdot k/2$ edges.

Prove the claim

Claim: The algorithm returns a minimal cut with probability $\geq 2/n^2$.

Proof: Consider a global min cut (S,S') of G. Let δ be edges with one endpoint in S and the other in S'. Let $k = |\delta|$ = size of the min cut.



Step 1: contract an edge in δ with probability $2/n$.

After step i: The multigraph G_i has n-i vertices and at least $(n-i)\cdot k/2$ edges.

Probability that the algorithm finds minimum cut?

$$\Pr[\text{edges in the final graph is } \delta] = \Pr[e_1, e_2, \ldots, e_{n-2} \notin \delta]$$

Proof

Theorem: $\Pr[e_1, e_2, \ldots, e_{n-2} \notin \delta] > 2/n^2$

Proof:

$$\Pr[e_1, e_2, \ldots, e_{n-2} \notin \delta] =$$

$$= \Pr[e_1 \notin \delta] \prod \Pr[e_{i+1} \notin \delta : e_1, \ldots, e_i \notin \delta]$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \ldots \left(1 - \frac{2}{3}\right)$$

$$= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \frac{n-5}{n-3} \ldots \frac{2}{4} \cdot \frac{1}{3}$$

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

Proof

Theorem: $\Pr[e_1, e_2, \ldots, e_{n-2} \notin \delta] > 2/n^2$

Proof:

$$\Pr[e_1, e_2, \ldots, e_{n-2} \notin \delta] =$$

$$= \Pr[e_1 \notin \delta] \prod \Pr[e_{i+1} \notin \delta : e_1, \ldots, e_i \notin \delta]$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \ldots \left(1 - \frac{2}{3}\right)$$

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

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

Amplification

To amplify the probability of success, run the contraction algorithm many times.

Claim: If we repeat the contraction algorithm $r\binom{n}{2}$ times with independent random choices, the probability that all runs fail is at most

$$\left(1-\frac{1}{\binom{n}{2}}\right)^{r\binom{n}{2}} \le (1/e)^r$$

$$\boxed{\left(1-\frac{1}{x}\right)^x \le 1/e}$$

Amplification

To amplify the probability of success, run the contraction algorithm many times.

Claim: If we repeat the contraction algorithm $r\binom{n}{2}$ times with independent random choices, the probability that all runs fail is at most

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^{r\binom{n}{2}} \leq (1/e)^r$$

constant

Set $r = (c \ln n)$ then probability of failure is: $e^{-c \ln n} = n^{-c}$

and probability of success is: $1-1/n^c$

Karger's Contraction Algorithm

Algorithm:

1. Start with the input graph $G=(V,E)$.
2. While $|V|>2$ do

    Contract an arbitrary edge (u,v)
3. Return the cut S (only one possible cut).

Running time?

Karger's Contraction Algorithm

Algorithm:

1. Start with the input graph $G=(V,E)$.
2. While $|V|>2$ do

   Contract an arbitrary edge (u,v)
3. Return the cut S (only one possible cut).

Running time: n-2 iterations.

each iteration requires $O(n)$ time

$\Rightarrow O(n^2)$

The algorithm is iterated $O(n^2 \log n)$ times…total running time $O(n^4 \log n)$.

Improved algorithm

Improvement. [Karger-Stein 1996] $O(n^2 \log^3 n)$.

– Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n/\sqrt{2}$ nodes remain.
– Run contraction algorithm until $n/\sqrt{2}$ nodes remain.
– Run contraction algorithm twice on resulting graph, and return best of two cuts.

Running time?

Improved algorithm

Improvement. [Karger-Stein 1996] $O(n^2 \log^3 n)$.

- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n/\sqrt{2}$ nodes remain .
- Run contraction algorithm until $n/\sqrt{2}$ nodes remain.
- Run contraction algorithm twice on resulting graph, and return best of two cuts.

Running time: $T(n) = 2(n^2 + T(n/\sqrt{2}\,))$

$= O(n^2 \log n)$ [Master Thm]

Improved algorithm

Improvement. [Karger-Stein 1996] $O(n^2 \log^3 n)$.

- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n/\sqrt{2}$ nodes remain .
- Run contraction algorithm until $n/\sqrt{2}$ nodes remain.
- Run contraction algorithm twice on resulting graph, and return best of two cuts.

Running time: $T(n) = 2(n^2+T(n/\sqrt{2}\,))$

$= O(n^2 \log n)$ [Master Thm]

Probability of success?

Improved algorithm

Improvement. [Karger-Stein 1996] $O(n^2 \log^3 n)$.

- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n/\sqrt{2}$ nodes remain .
- Run contraction algorithm until $n/\sqrt{2}$ nodes remain.
- Run contraction algorithm twice on resulting graph, and return best of two cuts.

Running time: $T(n) = 2(n^2+T(n/\sqrt{2}\,))$
$= O(n^2 \log n)$ [Master Thm]

Probability of failure: $\Pr[n] \leq (1-\tfrac{1}{2}\cdot\Pr[n/\sqrt{2}\,])^2$
$= O(1/\log n)$

Improved algorithm

Improvement. [Karger-Stein 1996] $O(n^2 \log^3 n)$.

Run the algorithm $c \log^2 n$ times

- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n/\sqrt{2}$ nodes remain .
- Run contraction algorithm until $n/\sqrt{2}$ nodes remain.
- Run contraction algorithm twice on resulting graph, and return best of two cuts.

Running time: $T(n) = 2(n^2 + T(n/\sqrt{2}\,))$
$= O(n^2 \log n)$

Probability of failure: $\Pr[n] \leq (1 - ½ \cdot \Pr[n/\sqrt{2}\,])^2$
$= O(1/\log n)$

Improved algorithm

Improvement. [Karger-Stein 1996] $O(n^2 \log^3 n)$.

- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n/\sqrt{2}$ nodes remain.
- Run contraction algorithm until $n/\sqrt{2}$ nodes remain.
- Run contraction algorithm twice on resulting graph, and return best of two cuts.

Best known. [Karger 2000] $O(m \log^3 n)$.

faster than best known max flow algorithm or deterministic global min cut algorithm

Reading material

Eric Vigoda's lecture notes

http://www.cc.gatech.edu/~vigoda/7530-Spring10/Kargers-MinCut.pdf