Monte Carlo Probability Boosting

Min-Cut

Outline

- Monte-Carlo Algorithms
  - Probability Amplification/Boosting
- Min-Cut
  - Edge contraction
  - Contraction algorithm
  - Amplification

Outline

- Monte-Carlo Algorithms
  - Probability Amplification/Boosting
- Min-Cut
  - Edge contraction
  - Contraction algorithm
  - Amplification

Learning objectives:

You are able to

- apply probability amplification to Monte Carlo algorithms with one-sided error
- state the definition of a minimum-cut
- state the definition of edge-contraction
- analyze the error probability of the contraction algorithm

Recap – Monte Carlo Algorithms

Monte-Carlo:
The algorithm gives a correct output with probability p, for some $0 < p \leq 1$.

Amplification

Monte-Carlo:
The algorithm gives a correct output with probability p, for some $0 < p \leq 1$.

The output is sometimes correct!

Amplification

One-sided error:
Consider a decision problem. If the algorithm say “no”, it is allowed to be wrong (w.p. $1-p$). If it says “yes”, then the answer must be correct.

Monte-Carlo:
The algorithm gives a correct output with probability p, for some $0 < p \leq 1$.

The output is sometimes correct!

Amplification

Monte-Carlo:
The algorithm gives a correct output with probability p, for some $0 < p \leq 1$.

The output is sometimes correct!

One-sided error:
Consider a decision problem. If the algorithm say "no", it is allowed to be wrong (w.p. $1 - p$). If it says "yes", then the answer must be correct.

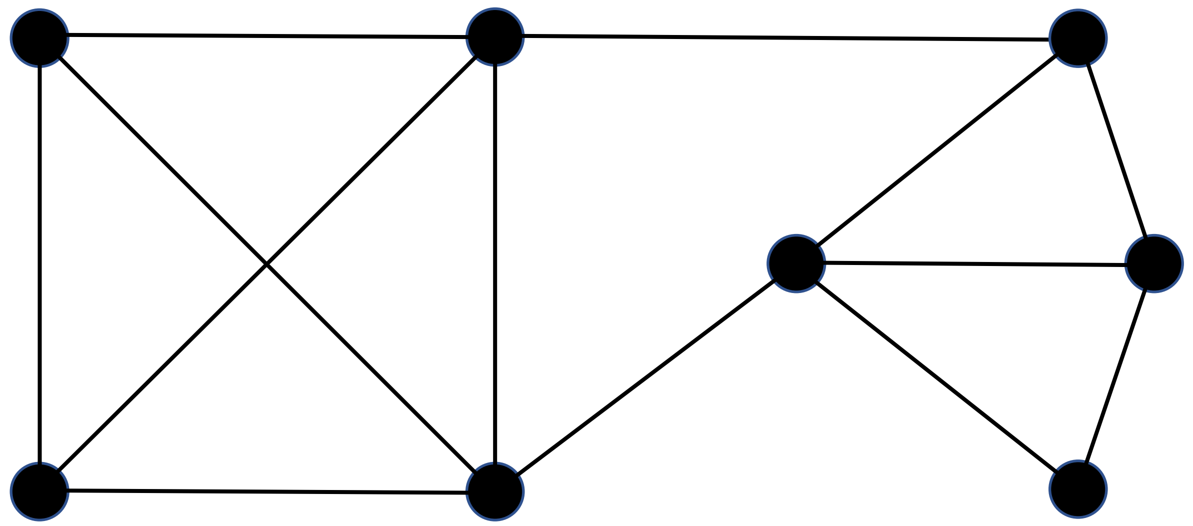
Run the algorithm x times, output "yes" if any run results in a "yes" answer.

Output is correct w.p. $(1 - p)^x$

Outline

- Monte-Carlo Algorithms
  - Probability Amplification/Boosting
- Min-Cut
  - Edge contraction
  - Contraction algorithm
  - Amplification

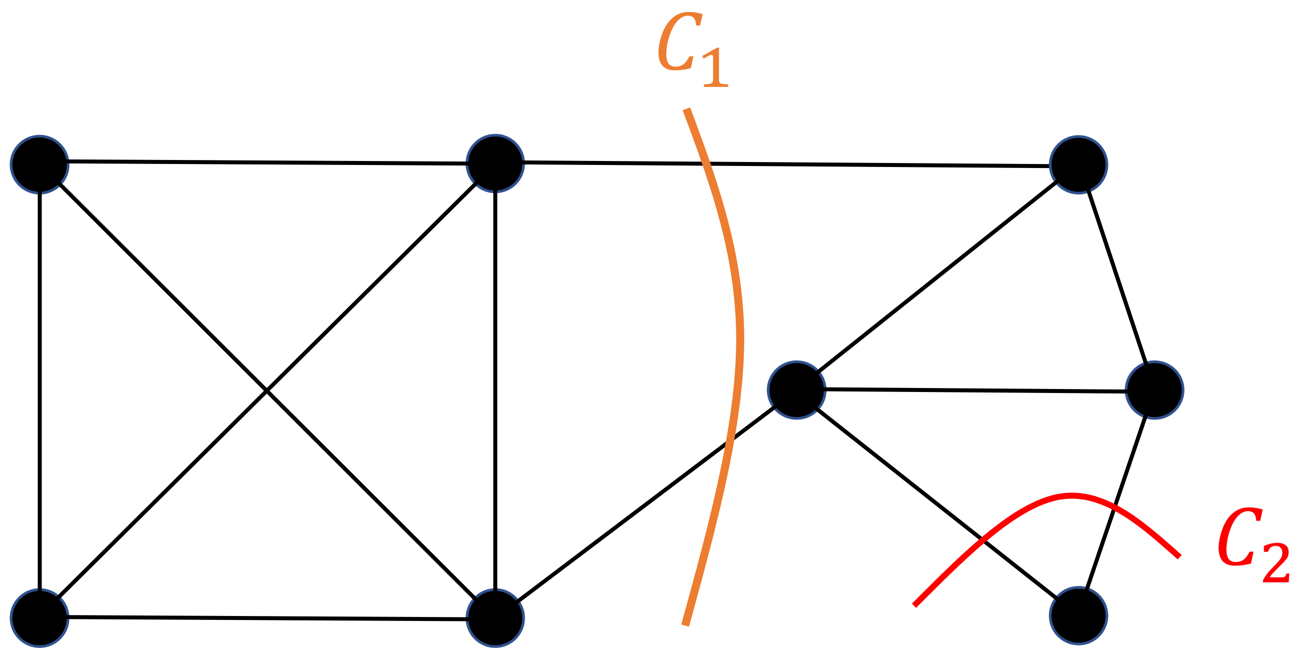
Graph Cut



Cut $C \subseteq E$ of graph $G = (V, E)$:
Divides the graph into two parts S and $V \setminus S$, s.t., if for $\{u, v\} \in E$ $u \in S$ and $v \in V \setminus S$ then $e \in C$.

The set C is called a *cut-set.*

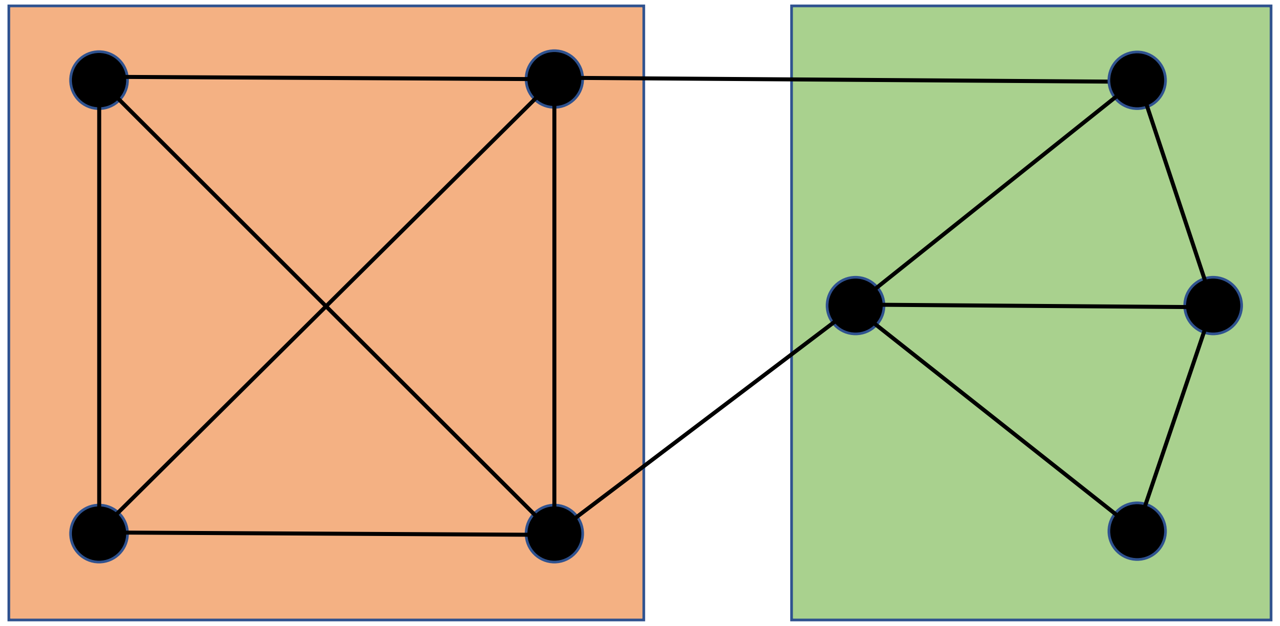
Graph Cut



Cut $C \subseteq E$ of graph $G = (V, E)$:
Divides the graph into two parts S and $V \setminus S$, s.t., if for $\{u, v\} \in E$ $u \in S$ and $v \in V \setminus S$ then $e \in C$.

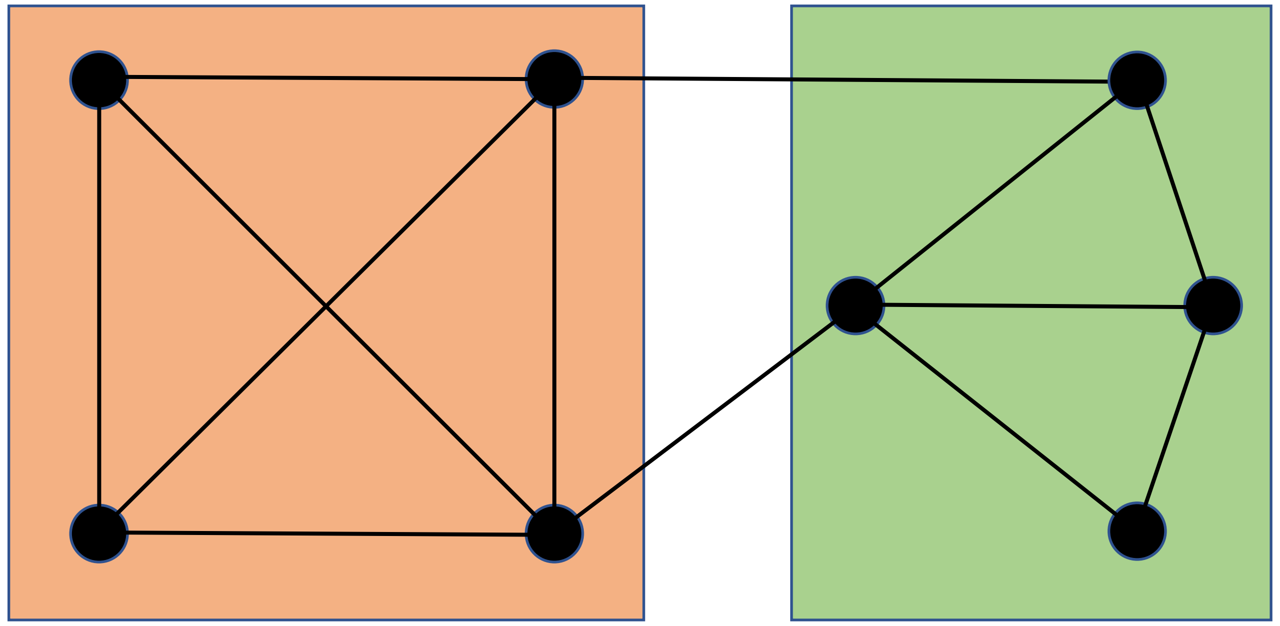
The set C is called a *cut-set.*

Graph Cut



Few edges across the well-connected components.

Graph Cut

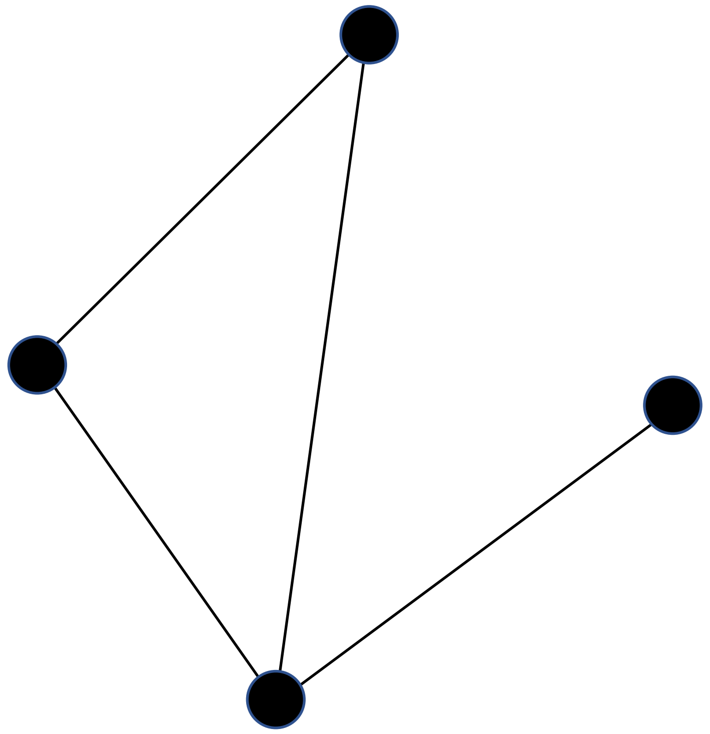


Few edges across the well-connected components.

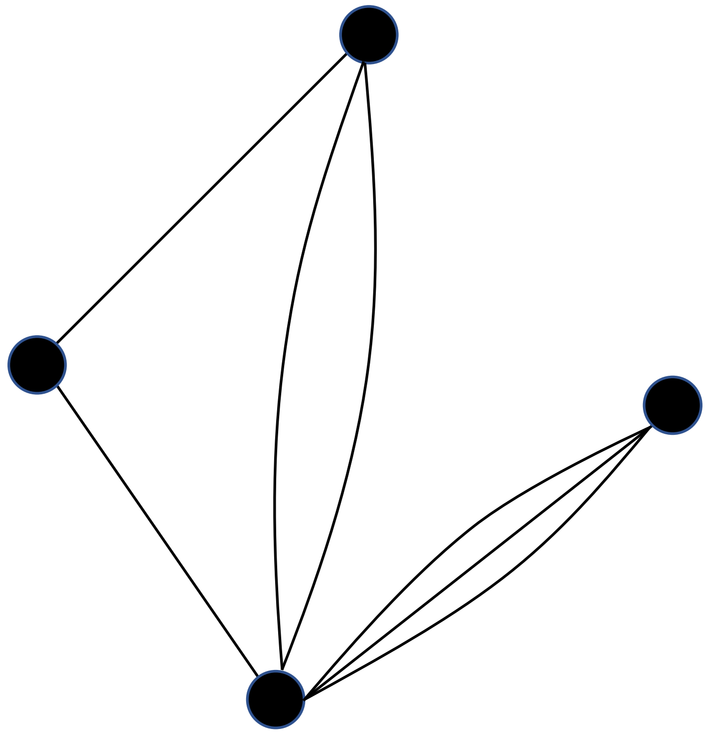
Community Detection:
Clusters with few edges between then are dissimilar.

Edge Contraction

A Simple Graph:
At most one edge between a pair of nodes.



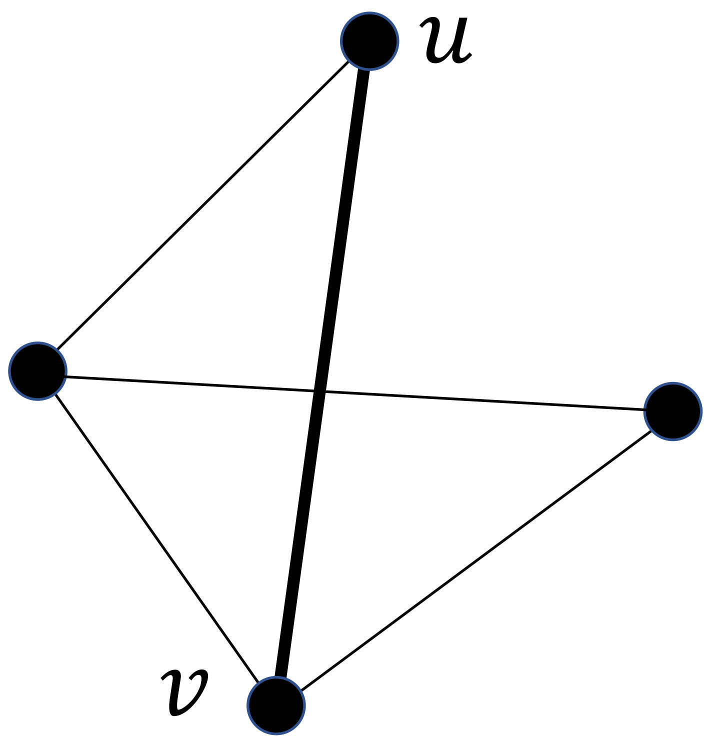
A Multi-Graph:
Many edges between pairs of nodes



Edge Contraction

Edge contraction of $e = \{u, v\}$:
Consider graph $G = (V, E)$.
We obtain a new graph $G/e = (V', E')$ such that

1. nodes u and v are merged into a new node w
2. if $\{u', u\} \in E$, then we add $\{u', w\}$ to E'
3. if $\{v', v\} \in E$, then we add $\{v', w\}$ to E'
4. all edges not involving u or v are added to E'

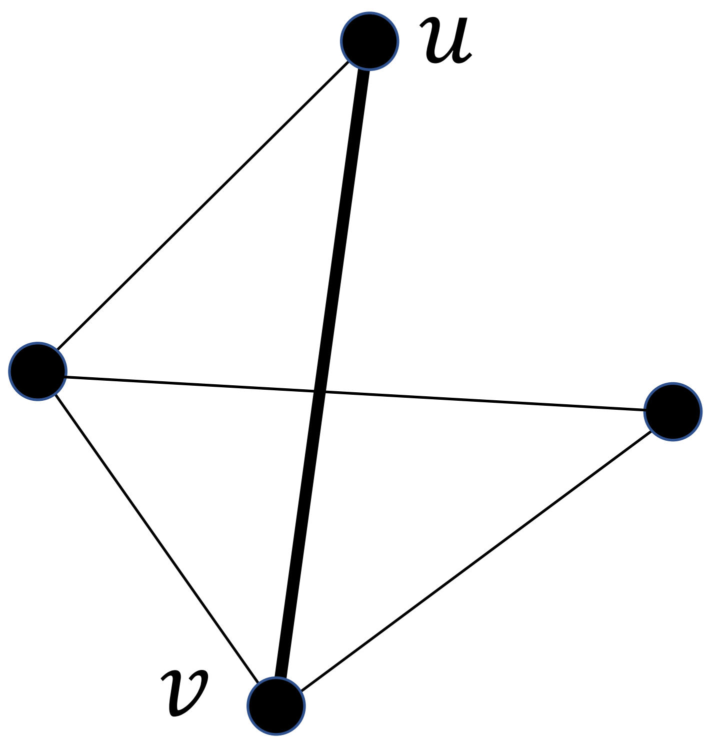


Edge Contraction

Edge contraction of $e = \{u, v\}$:
Consider graph $G = (V, E)$.
We obtain a new graph
$G/e = (V', E')$ such that

1. nodes u and v are merged into a new node w
2. if $\{u', u\} \in E$, then we add $\{u', w\}$ to E'
3. if $\{v', v\} \in E$, then we add $\{v', w\}$ to E'
4. all edges not involving u or v are added to E'

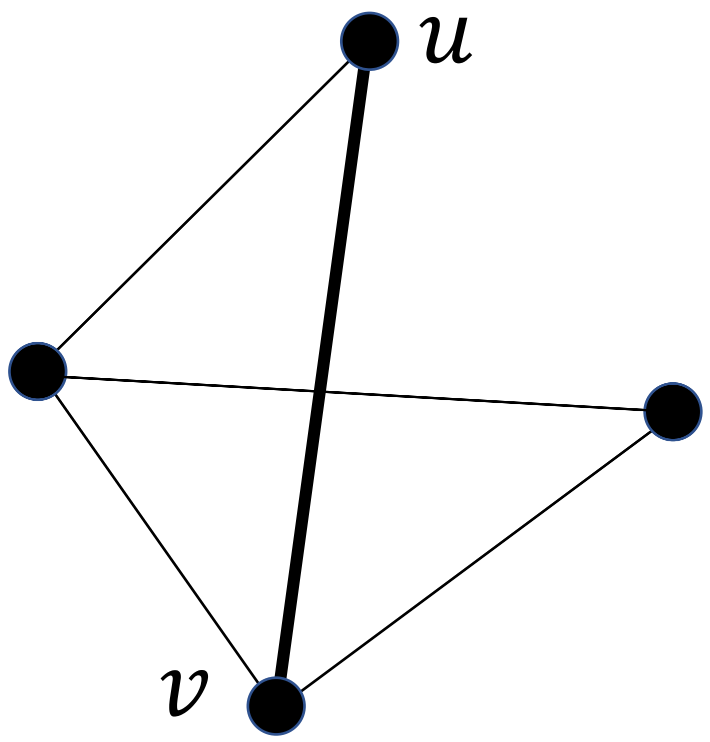
Standard notation

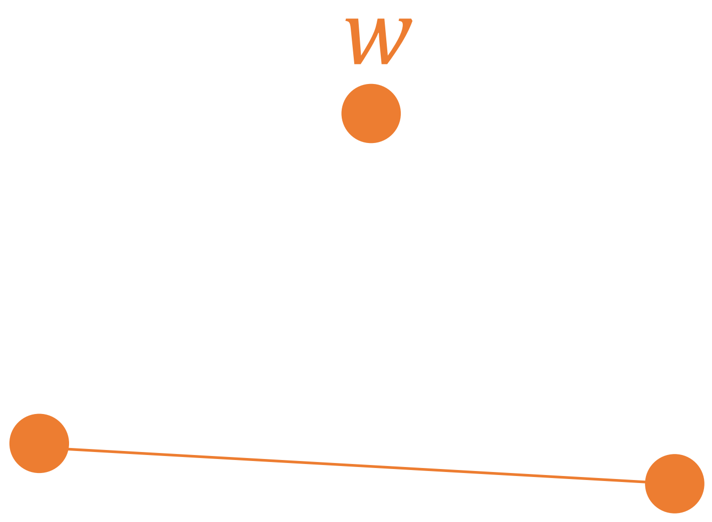


Edge Contraction

Edge contraction of $e = \{u, v\}$:
Consider graph $G = (V, E)$.
We obtain a new graph
$G/e = (V', E')$ such that

1. nodes u and v are merged into a new node w
2. if $\{u', u\} \in E$, then we add $\{u', w\}$ to E'
3. if $\{v', v\} \in E$, then we add $\{v', w\}$ to E'
4. all edges not involving u or v are added to E'

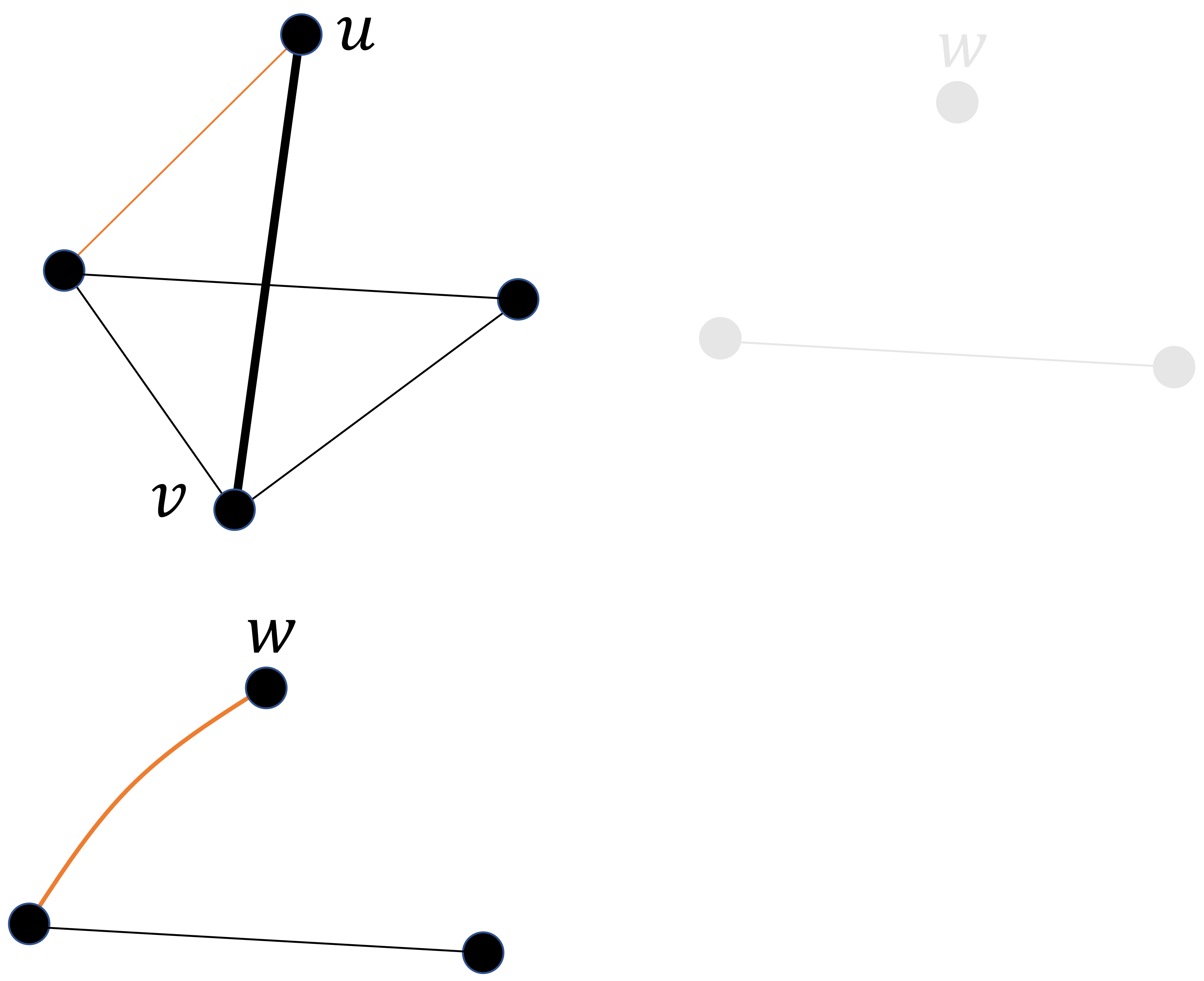




Edge Contraction

Edge contraction of $e = \{u, v\}$:
Consider graph $G = (V, E)$.
We obtain a new graph $G/e = (V', E')$ such that

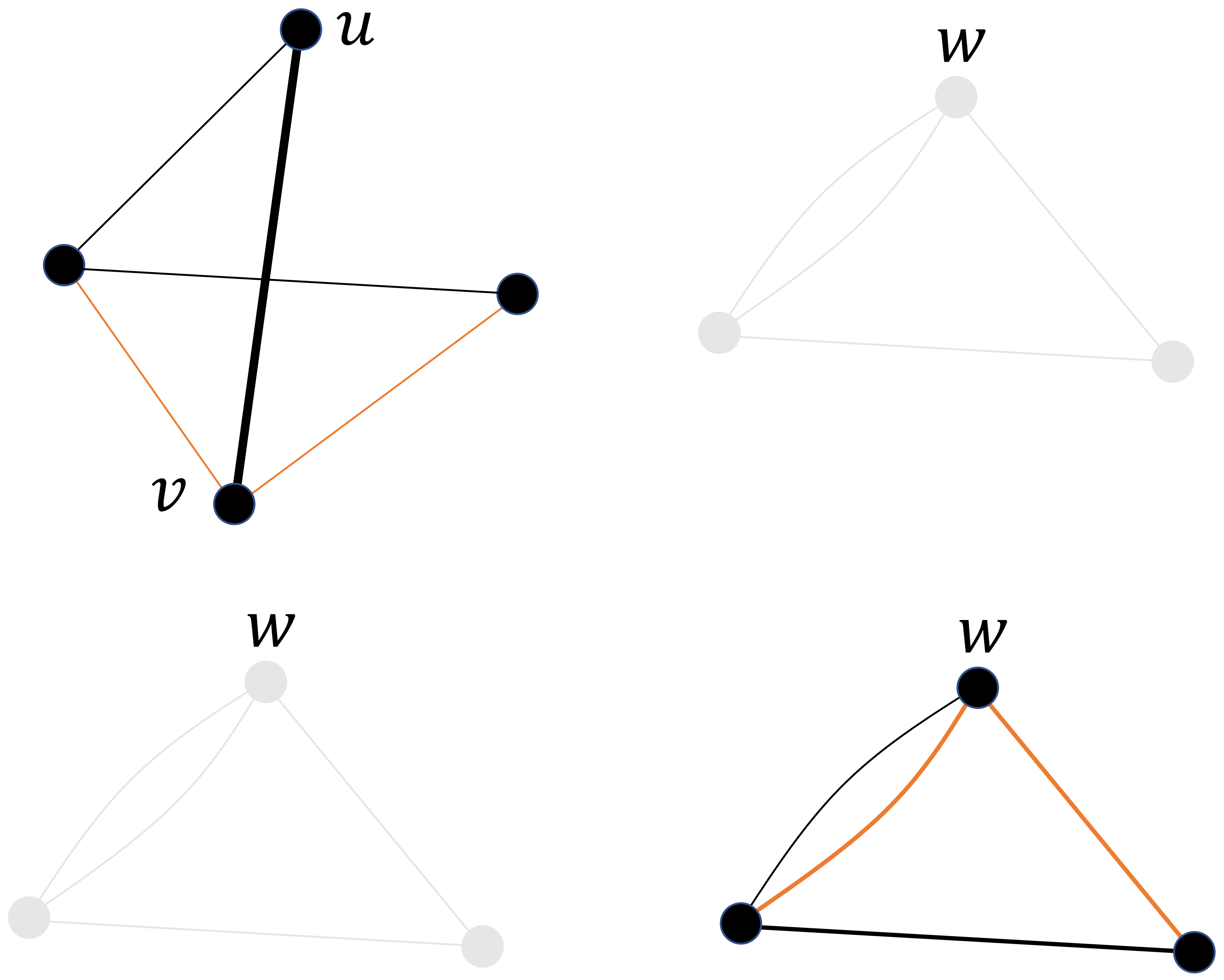
1. nodes u and v are merged into a new node w
2. if $\{u', u\} \in E$, then we add $\{u', w\}$ to E'
3. if $\{v', v\} \in E$, then we add $\{v', w\}$ to E'
4. all edges not involving u or v are added to E'



Edge Contraction

Edge contraction of $e = \{u, v\}$:
Consider graph $G = (V, E)$.
We obtain a new graph
$G/e = (V', E')$ such that

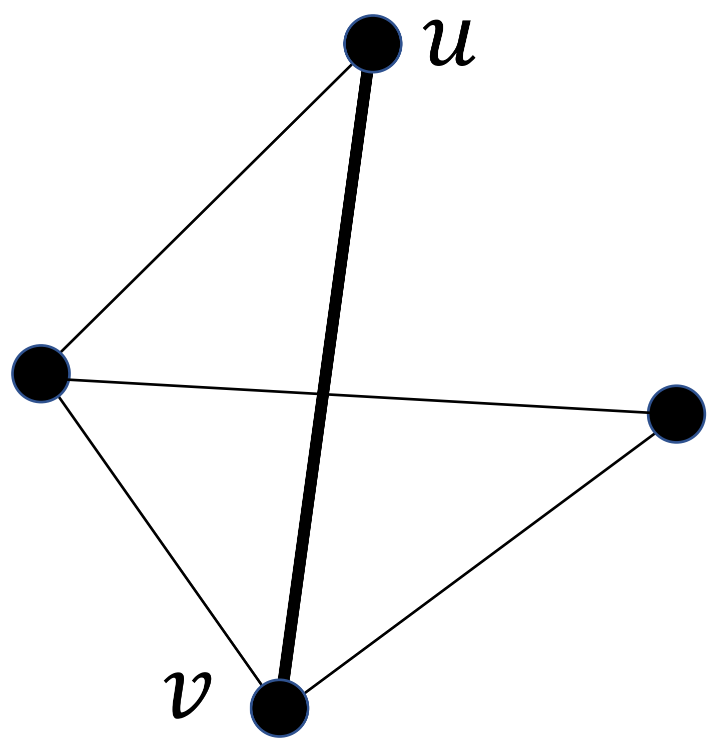
1. nodes u and v are merged into a new node w
2. if $\{u', u\} \in E$, then we add $\{u', w\}$ to E'
3. if $\{v', v\} \in E$, then we add $\{v', w\}$ to E'
4. all edges not involving u or v are added to E'



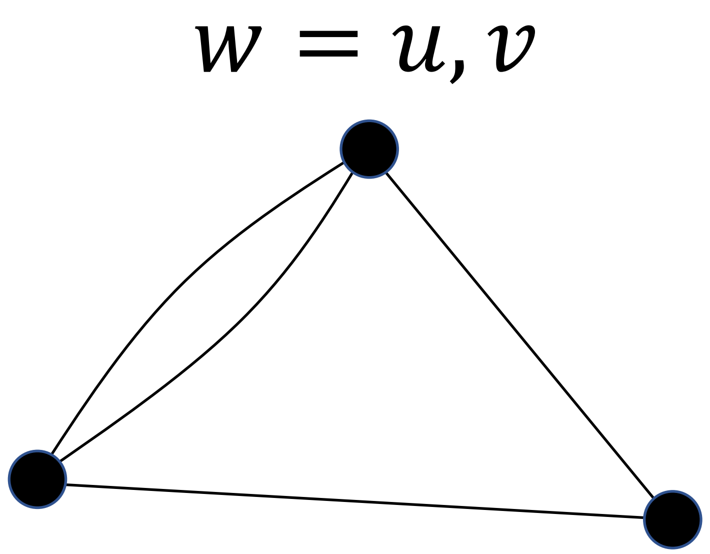
Edge Contraction

Edge contraction of $e = \{u, v\}$:
Consider graph $G = (V, E)$.
We obtain a new graph $G/e = (V', E')$ such that

1. nodes u and v are merged into a new node w
2. if $\{u', u\} \in E$, then we add $\{u', w\}$ to E'
3. if $\{v', v\} \in E$, then we add $\{v', w\}$ to E'
4. all edges not involving u or v are added to E'



G



G'

Outline

- Monte-Carlo Algorithms
  - Probability Amplification/Boosting
- Min-Cut
  - Edge contraction
  - Contraction algorithm
  - Amplification

Theorem:
There is a polynomial time Monte Carlo algorithm to find a Minimum Cut.

Corollary:
There is a polynomial time algorithm to find a Minimum Cut with high probability.

Minimum Cut – The Contraction Algorithm

The Algorithm (Karger'93):
Let $G = (V, E)$ be a connected graph

While $|V| > 2$:
  Pick an edge e uniformly at random.
  $G \coloneqq G/e$

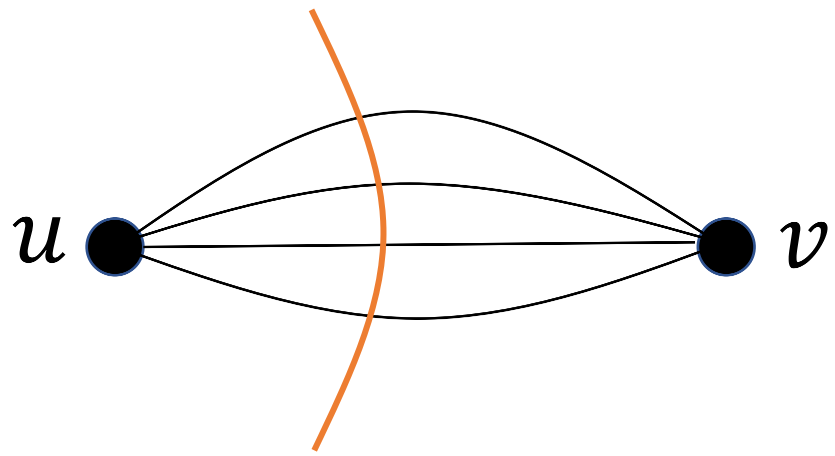
Return the unique cut of contracted graph G

Minimum Cut – The Contraction Algorithm

The Algorithm (Karger'93):
Let $G = (V, E)$ be a connected graph

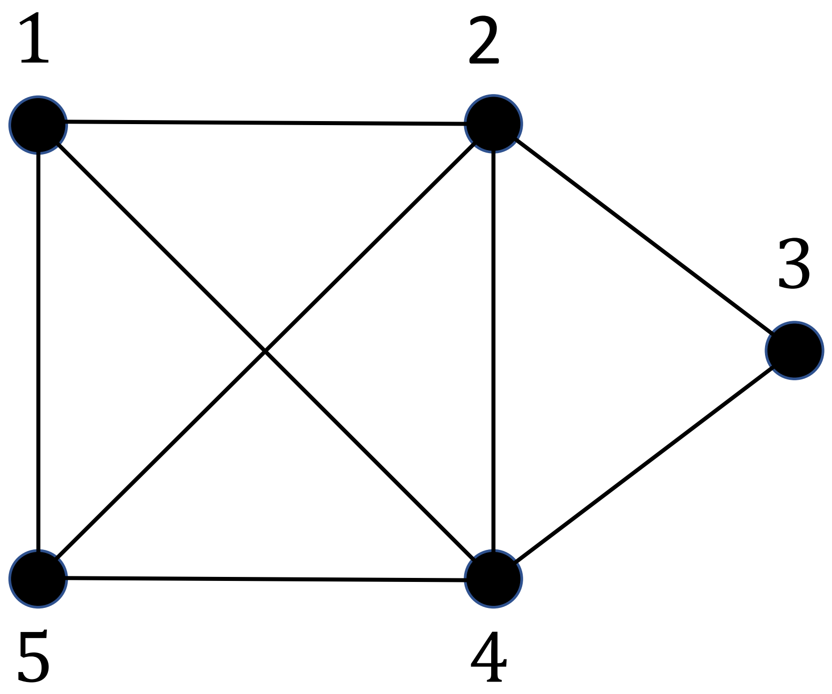
While $|V| > 2$:
 Pick an edge e uniformly at random.
 $G := G/e$

Return the unique cut of contracted graph G



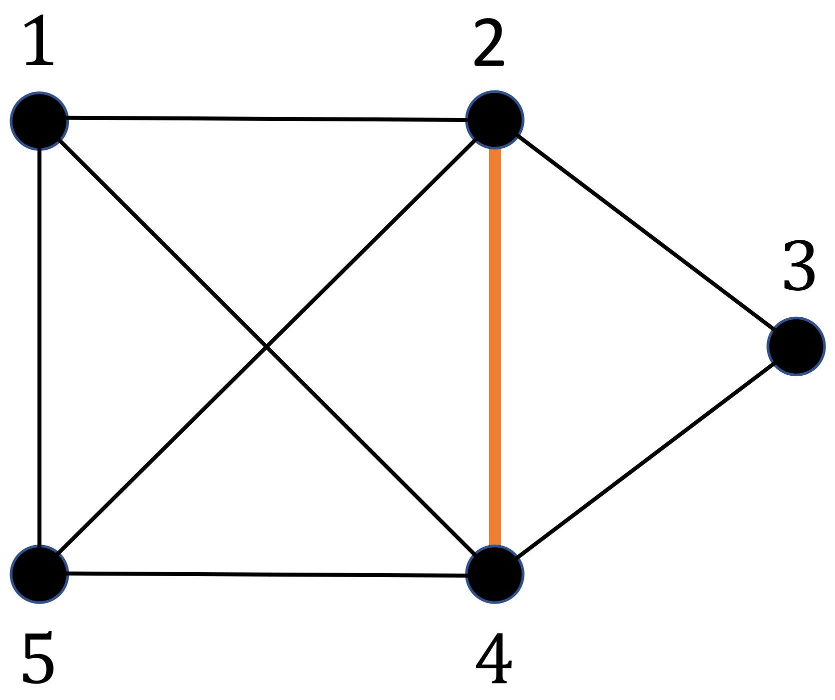
Minimum Cut – The Contraction Algorithm

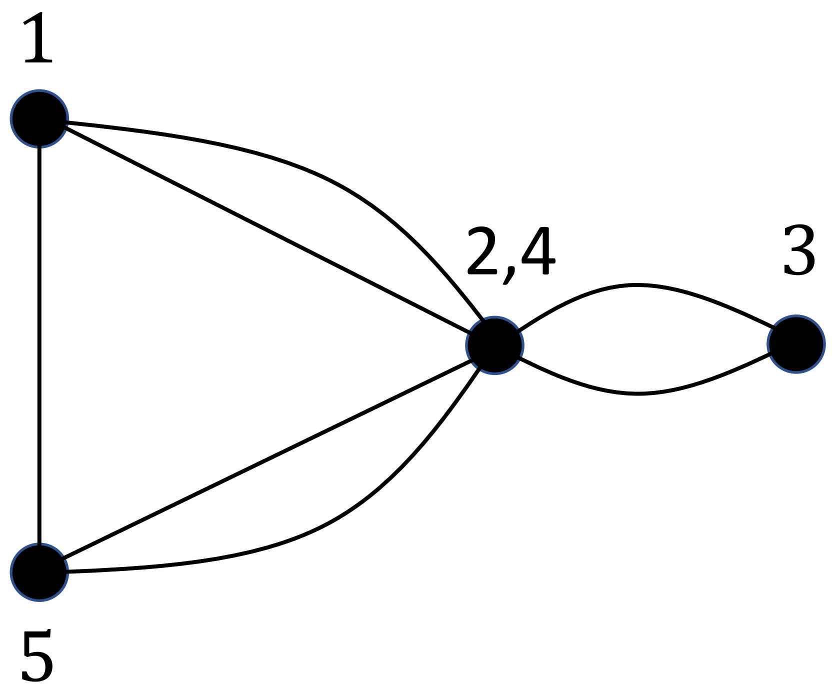
While $|V| > 2$:
  Pick an edge e uniformly at random.
  $G \coloneqq G/e$



Minimum Cut – The Contraction Algorithm

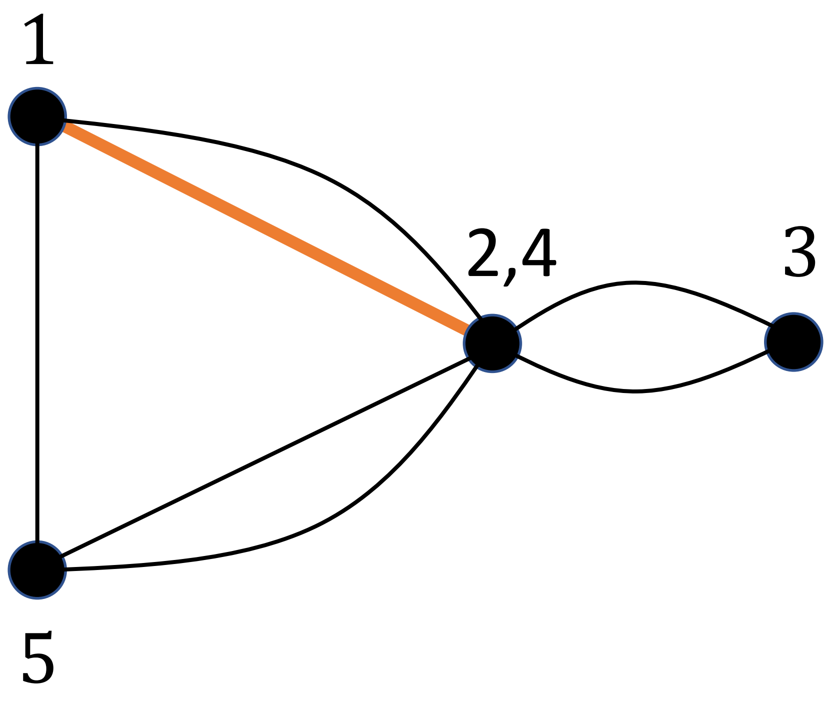
While $|V| > 2$:
Pick an edge e
uniformly at random.
$G \coloneqq G/e$

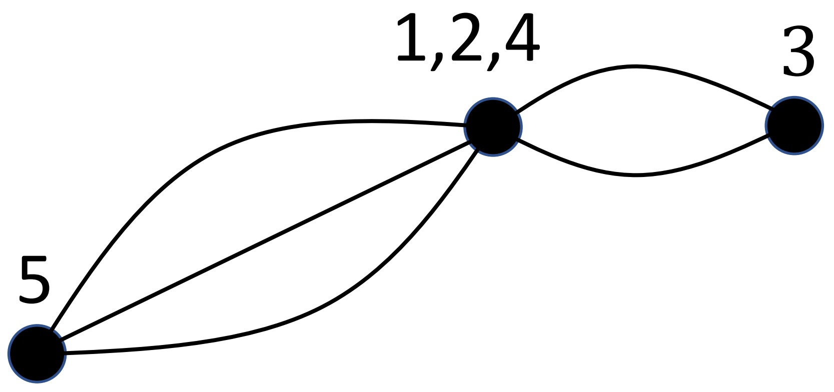




Minimum Cut – The Contraction Algorithm

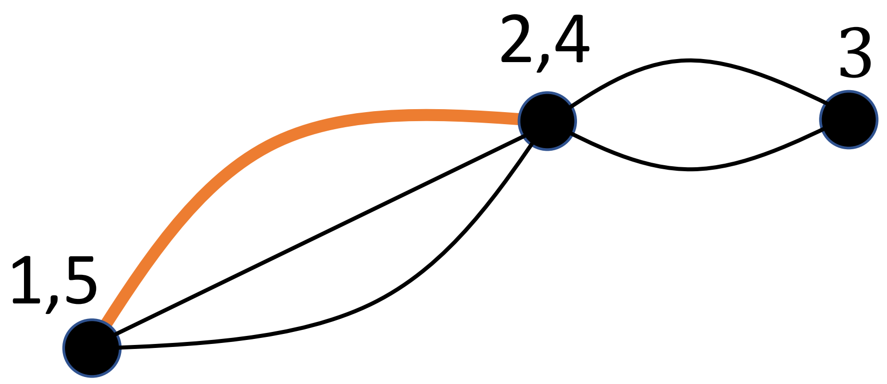
While $|V| > 2$:
  Pick an edge e
  uniformly at random.
  $G \coloneqq G/e$

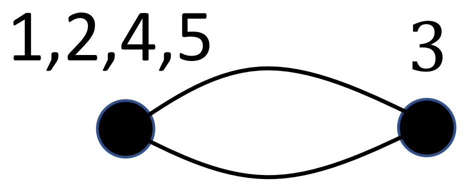




Minimum Cut – The Contraction Algorithm

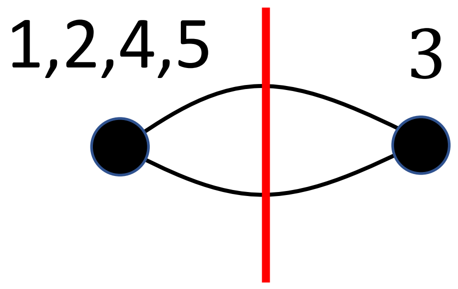
While $|V| > 2$:
  Pick an edge e
  uniformly at random.
  $G \coloneqq G/e$

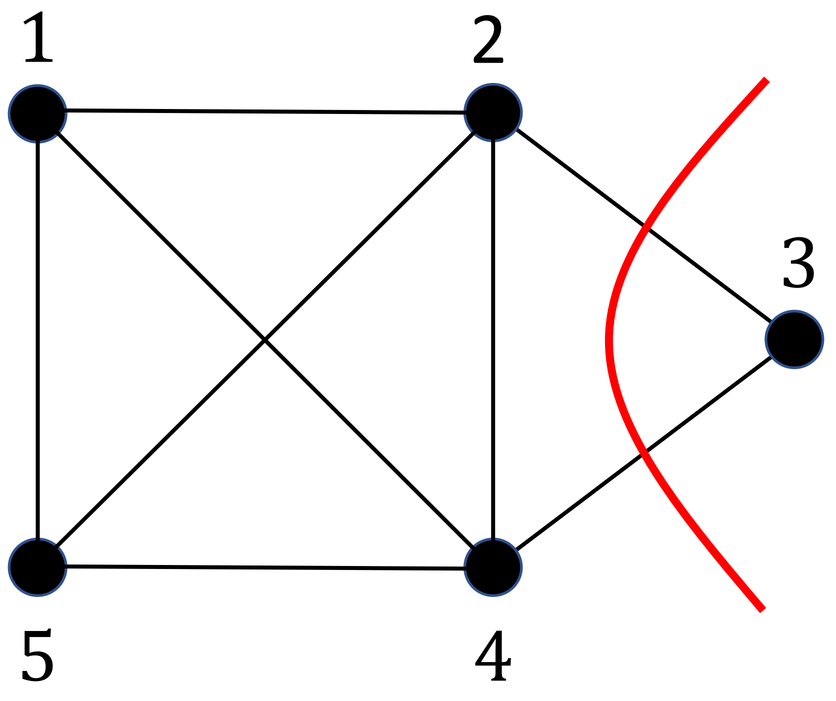




Minimum Cut – The Contraction Algorithm

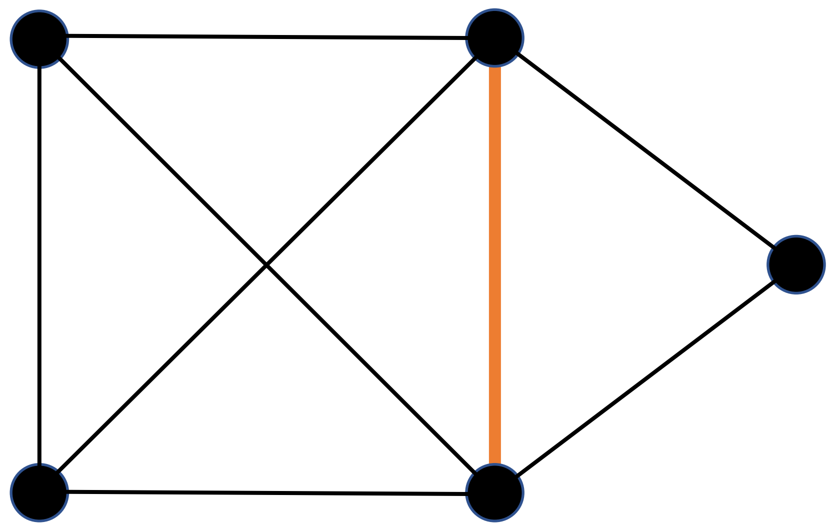
While $|V| > 2$:
  Pick an edge e uniformly at random.
  $G := G/e$

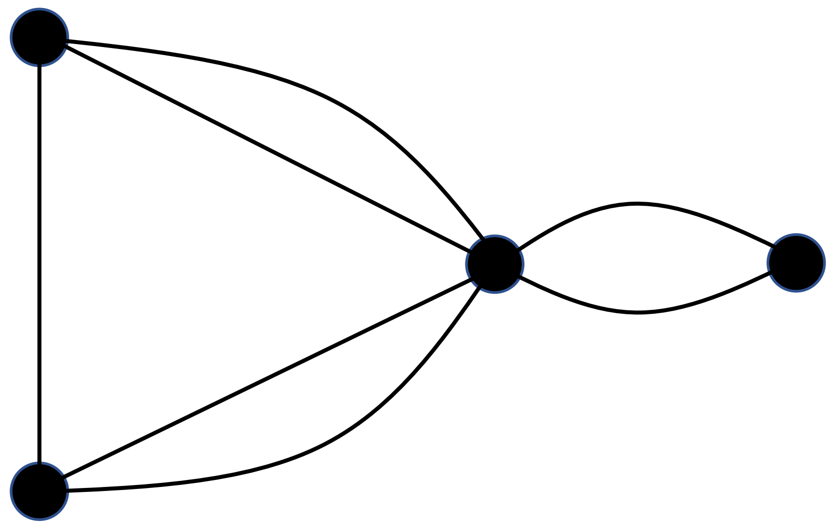




Minimum Cut – The Contraction Algorithm

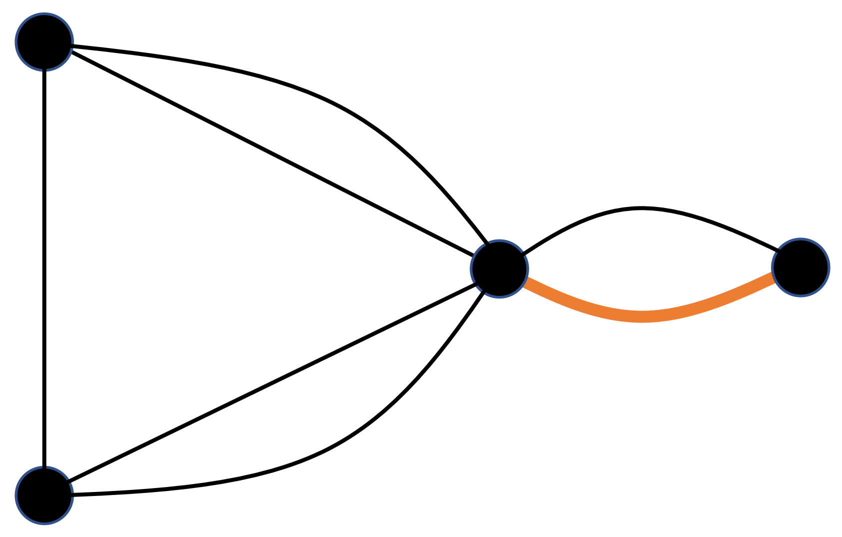
While $|V| > 2$:
  Pick an edge e uniformly at random.
  $G \coloneqq G/e$

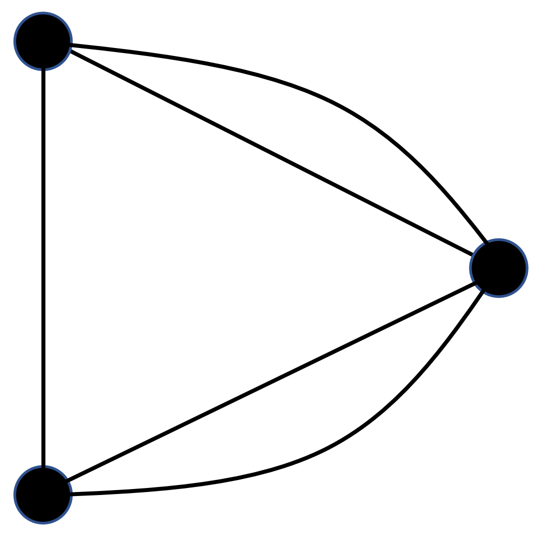




Minimum Cut – The Contraction Algorithm

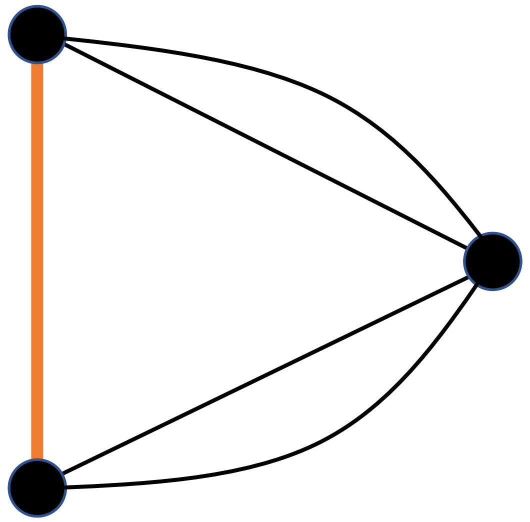
While $|V| > 2$:
  Pick an edge e
  uniformly at random.
  $G \coloneqq G/e$

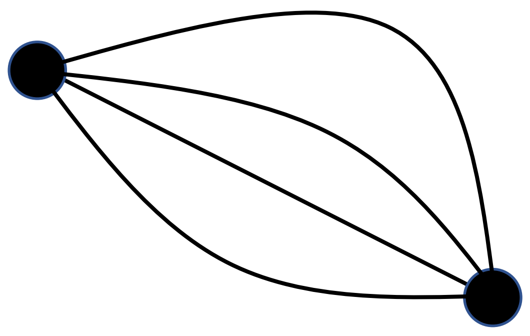




Minimum Cut – The Contraction Algorithm

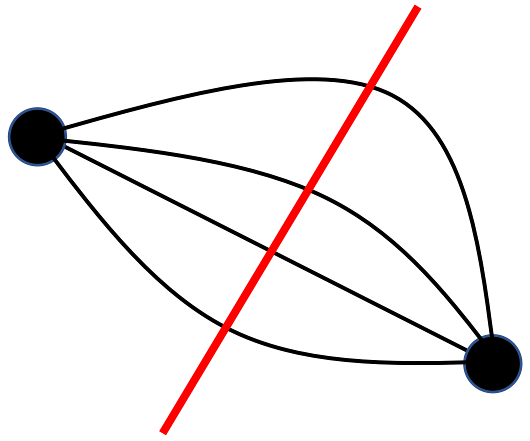
While $|V| > 2$:
  Pick an edge e
  uniformly at random.
  $G \coloneqq G/e$

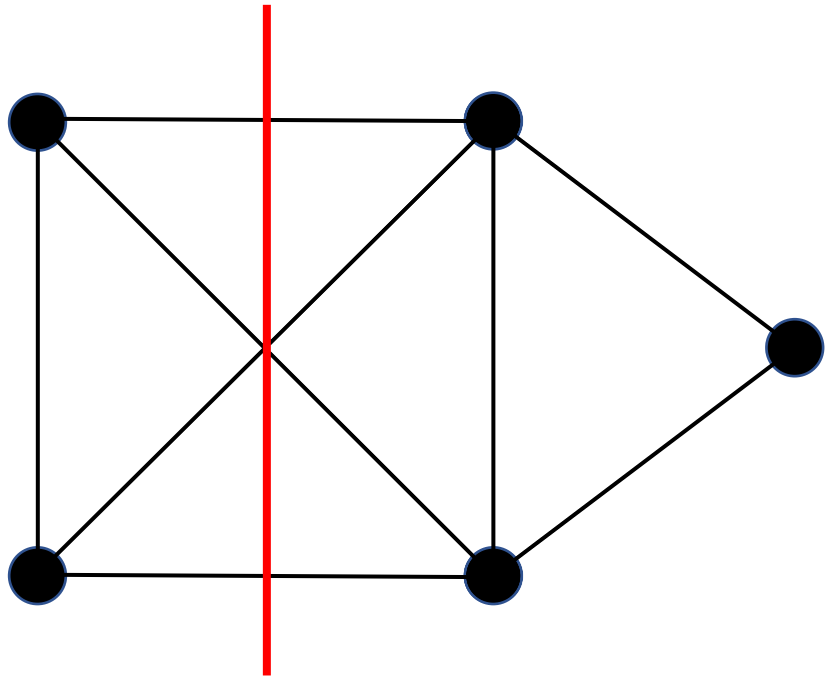




Minimum Cut – The Contraction Algorithm

While $|V| > 2$:
  Pick an edge e uniformly at random.
  $G \coloneqq G/e$





Outline

- Monte-Carlo Algorithms
  - Probability Amplification/Boosting
- Min-Cut
  - Edge contraction
  - Contraction algorithm
  - Amplification

Theorem:
There is a polynomial time Monte Carlo algorithm to find a Minimum Cut.

Corollary:
There is a polynomial time algorithm to find a Minimum Cut with high probability.

The Contraction Algorithm - Runtime

While $|V| > 2$:
  Pick an edge e uniformly at random.
  $G := G/e$

Sloppy Analysis:
The algorithm runs in polynomial time.

Outline

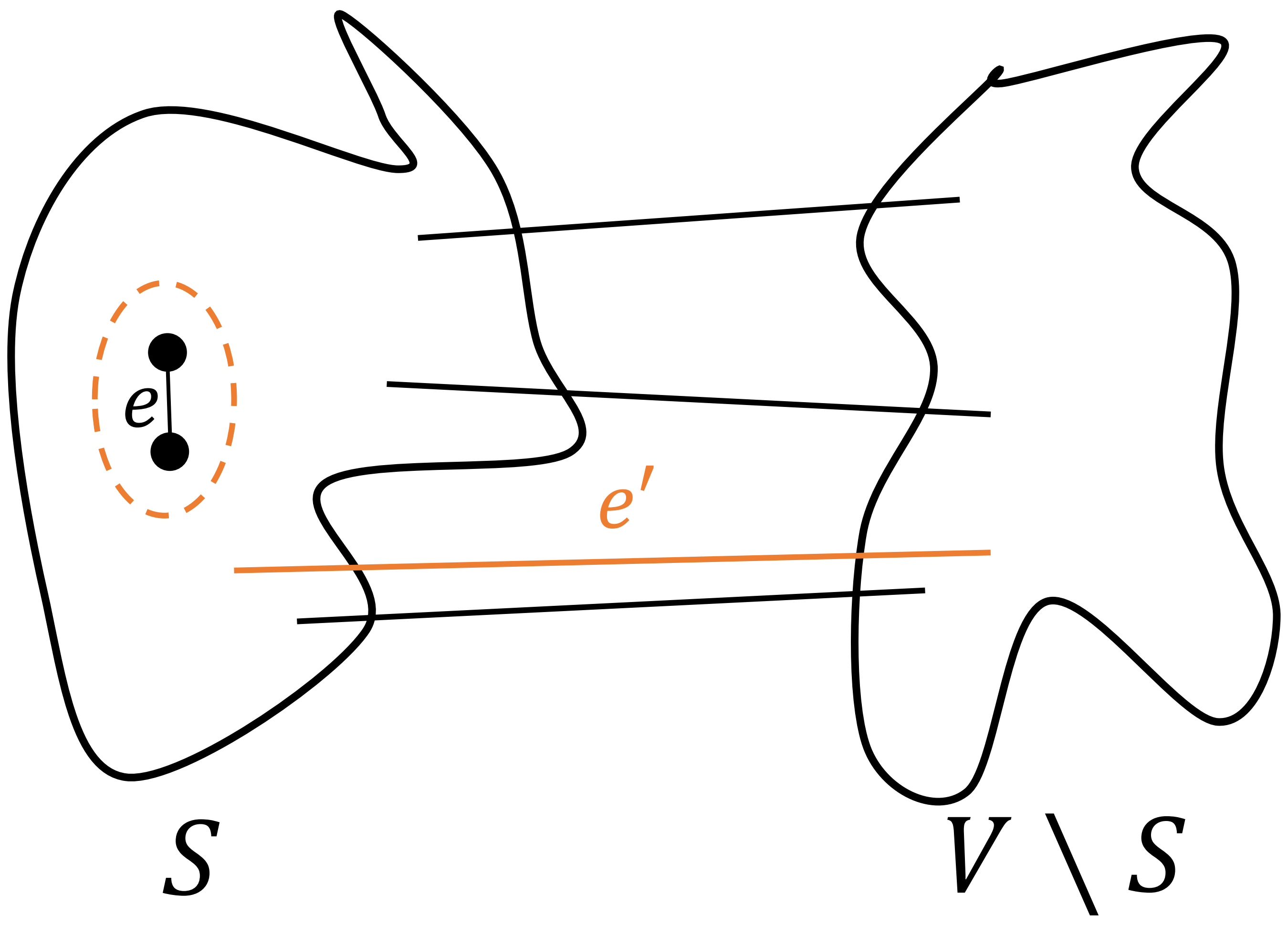
- Monte-Carlo Algorithms
  - Probability Amplification/Boosting
- Min-Cut
  - Edge contraction
  - Contraction algorithm
  - Amplification

Theorem:
There is a polynomial time Monte Carlo algorithm to find a Minimum Cut.

Corollary:
There is a polynomial time algorithm to find a Minimum Cut with high probability.

The Contraction Algorithm - Correctness

Observation:
A cut in G/e is also a cut in G.

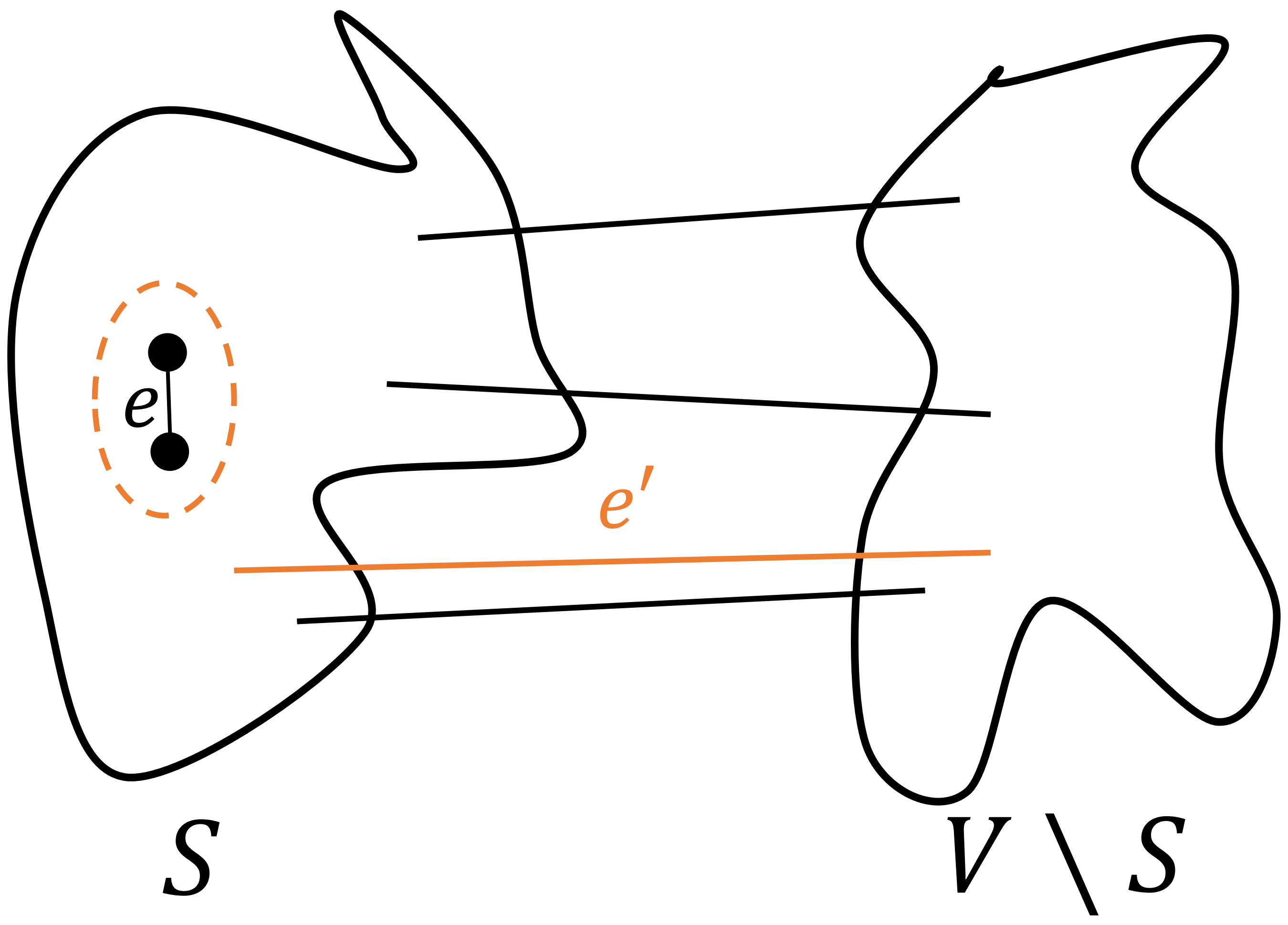


Proof:
Let C be the set of edges that define a cut $(S, V \setminus S)$ in G/e and let $e = \{u, v\}$ (in G).

Suppose that there is an edge e' between S and $V \setminus S$ in G.

The Contraction Algorithm - Correctness

Observation:
A cut in G/e is also a cut in G.

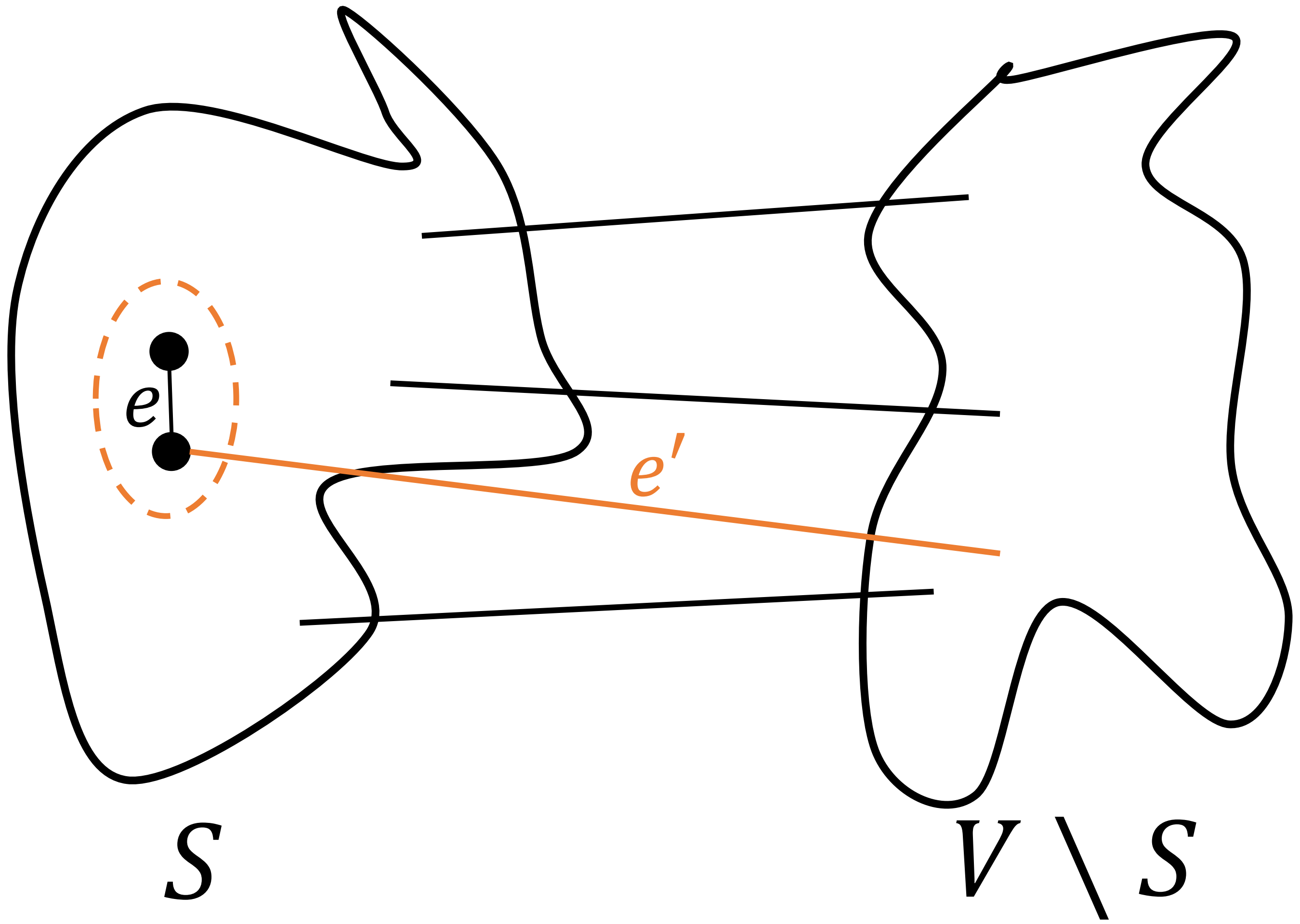


Proof:
Let C be the set of edges that define a cut $(S, V \setminus S)$ in G/e and let $e = \{u, v\}$ (in G).

Suppose that there is an edge e' between S and $V \setminus S$ in G. If $e' \cap e = \emptyset$, then e' also crosses $(S, V \setminus S)$ in G/e and hence, must be in C.

The Contraction Algorithm - Correctness

Observation:
A cut in G/e is also a cut in G.



Proof:
Let C be the set of edges that define a cut $(S, V \setminus S)$ in G/e and let $e = \{u, v\}$ (in G).

Suppose that there is an edge e' between S and $V \setminus S$ in G and that $u \in e'$. Since both endpoints of e are in S and $e' \neq e$, e' must still cross the cut in G/e.
Therefore, $e' \in C$.

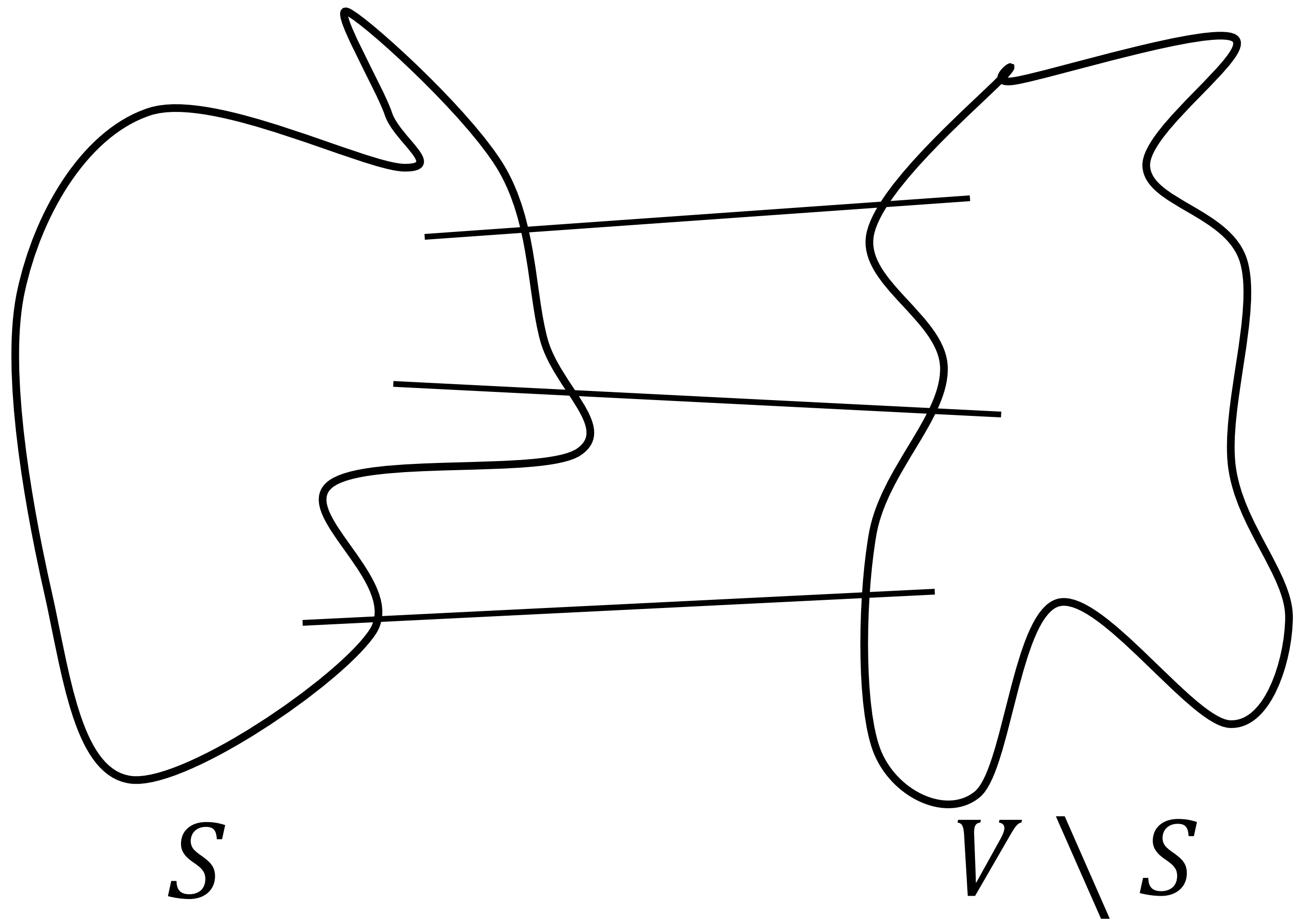
Minimum Cut

Theorem:
The contraction algorithm outputs a minimum cut with probability at least $2/n(n-1)$.

Minimum Cut

Consider some minimum cut of size k

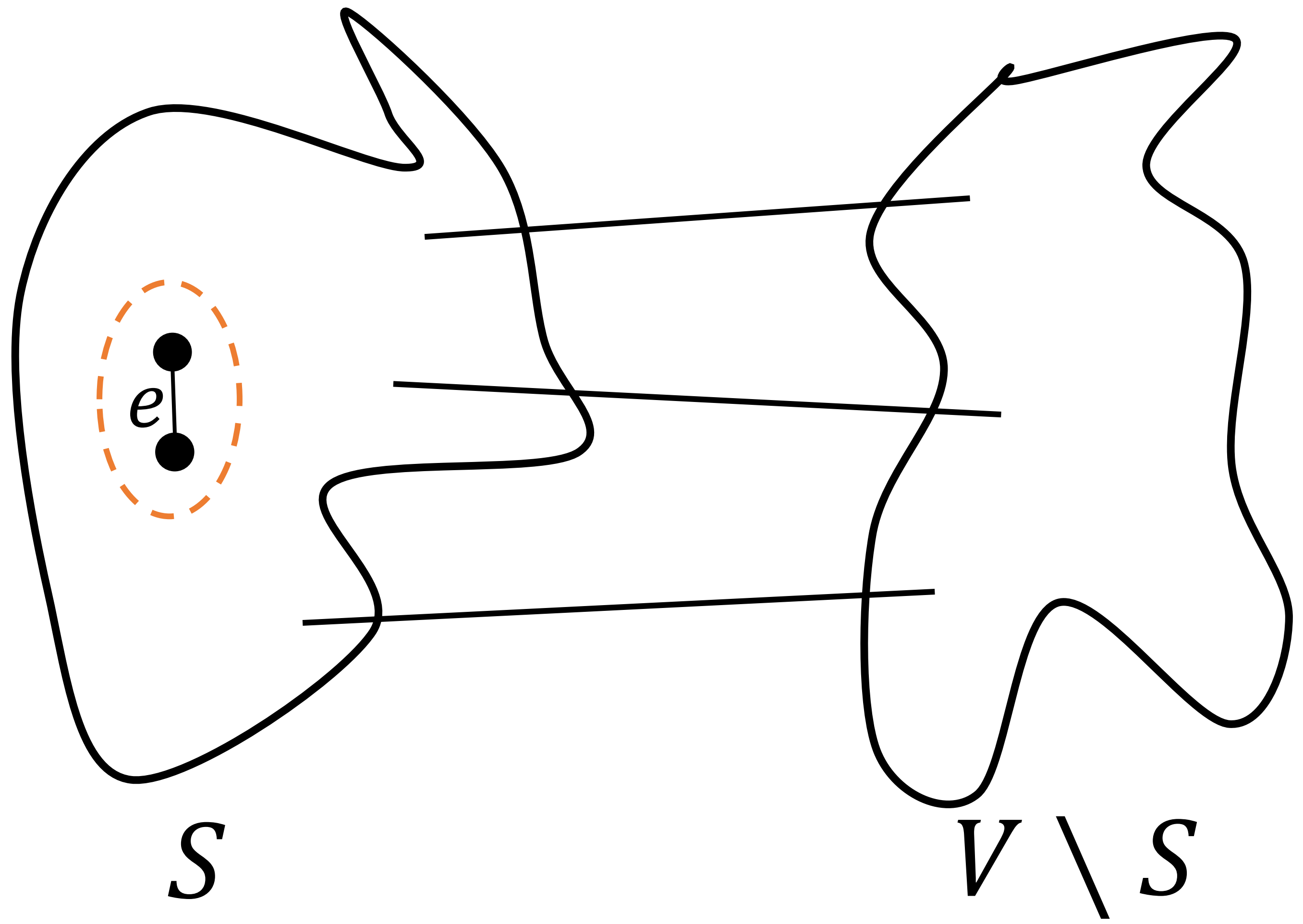
We obtain this cut if we always contract in S or in $V \setminus S$.



Minimum Cut

Consider some minimum cut of size k

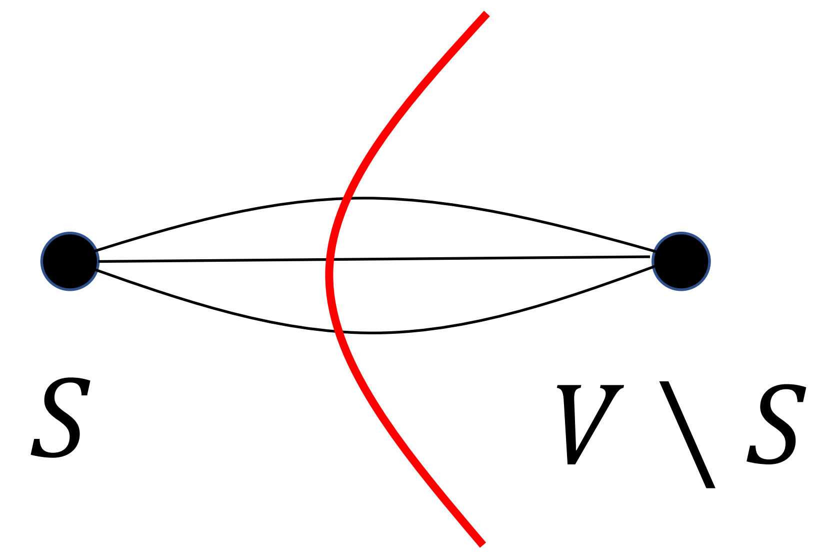
We obtain this cut if we always contract in S or in $V \setminus S$.



Minimum Cut

Consider some minimum cut of size k

We obtain this cut if we always contract in S or in $V \setminus S$.

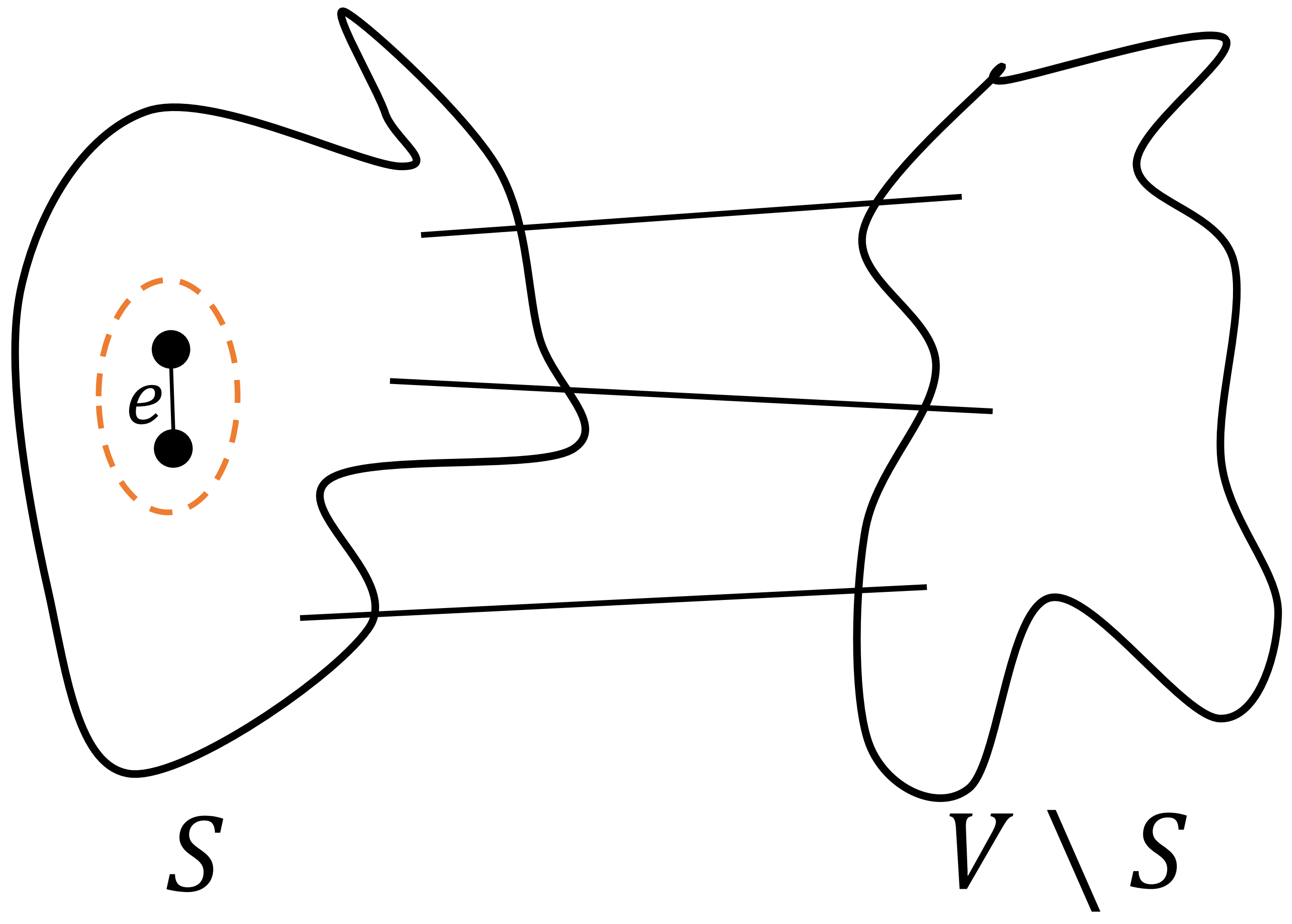


Minimum Cut

Observation:
The algorithm runs for $n - 2$ iterations.

Let F_i be the event that no edges of the minimum cut C was contracted in the first i iterations.

Task:
Bound $P(F_{n-2})$.

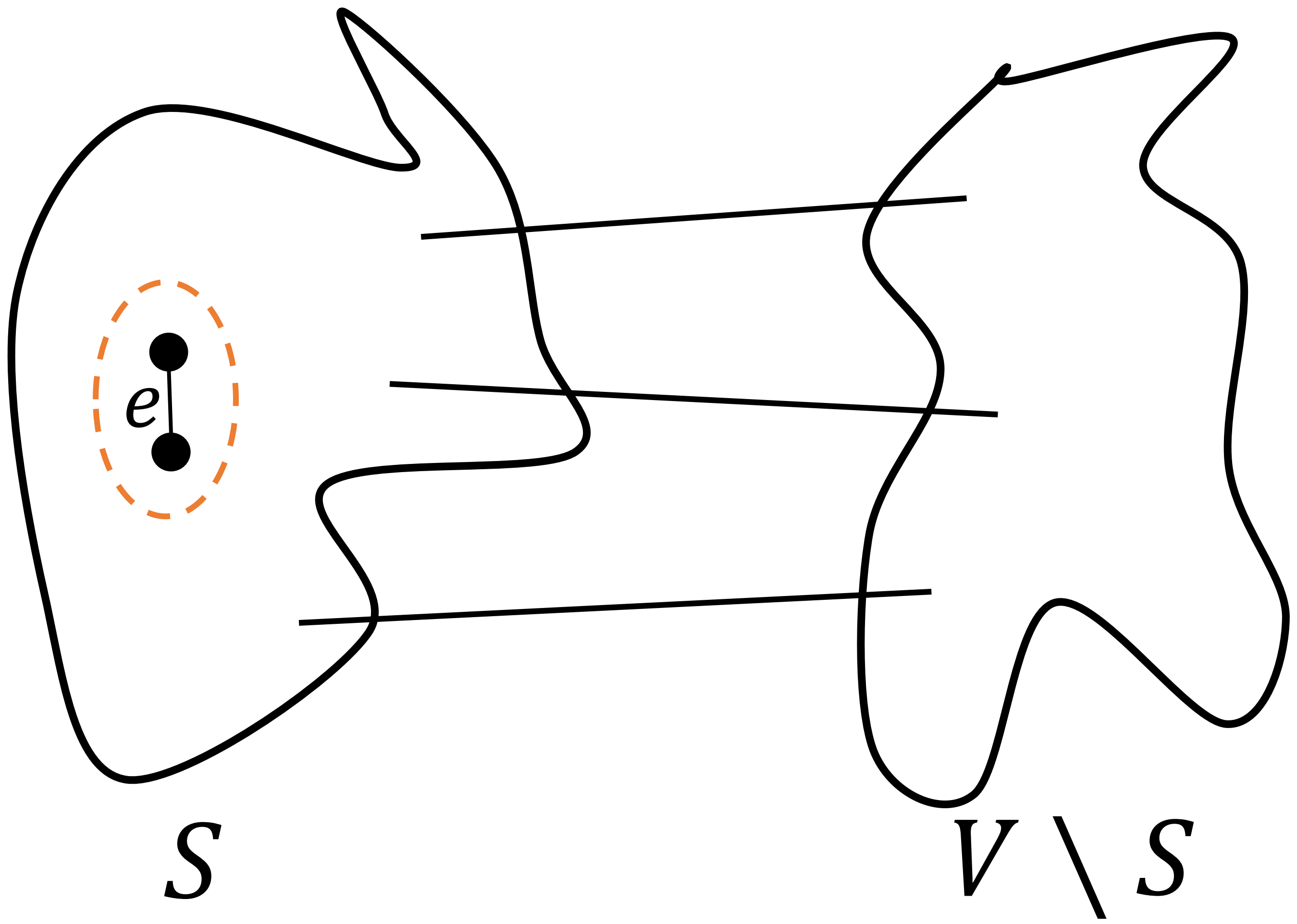


Minimum Cut

Let F_i be the event that no edges of the minimum cut C was contracted in the first i iterations.

Tutorial Session:

$$P(F_{n-2}) \geq \frac{2}{n(n-1)}$$



Minimum Cut

We obtain a cut $C = (S, V \setminus S)$ if we always contract in S or in $V \setminus S$.

The probability that we never contract C is

$$P(F_{n-2}) \geq \frac{2}{n(n-1)}$$

Theorem:
The contraction algorithm outputs a minimum cut with probability at least $2/n(n-1)$.

Outline

- Monte-Carlo Algorithms
  - Probability Amplification/Boosting
- Min-Cut
  - Edge contraction
  - Contraction algorithm
  - Amplification

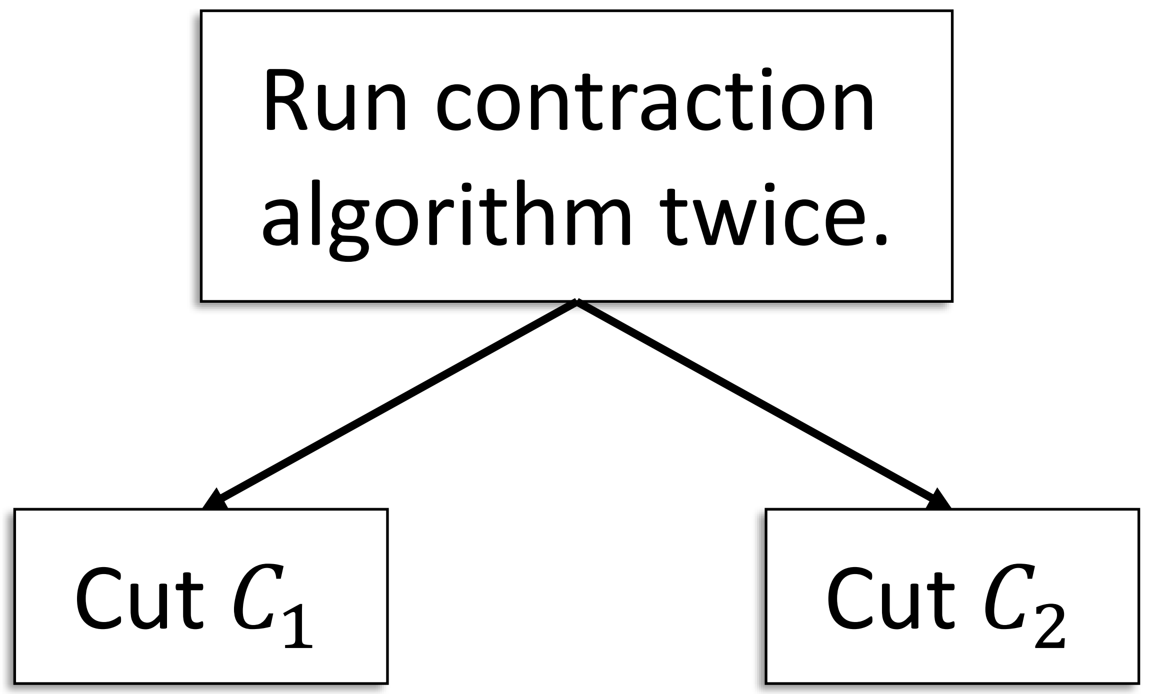
Theorem:
There is a polynomial time Monte Carlo algorithm to find a Minimum Cut.

Corollary:
There is a polynomial time algorithm to find a Minimum Cut with high probability.

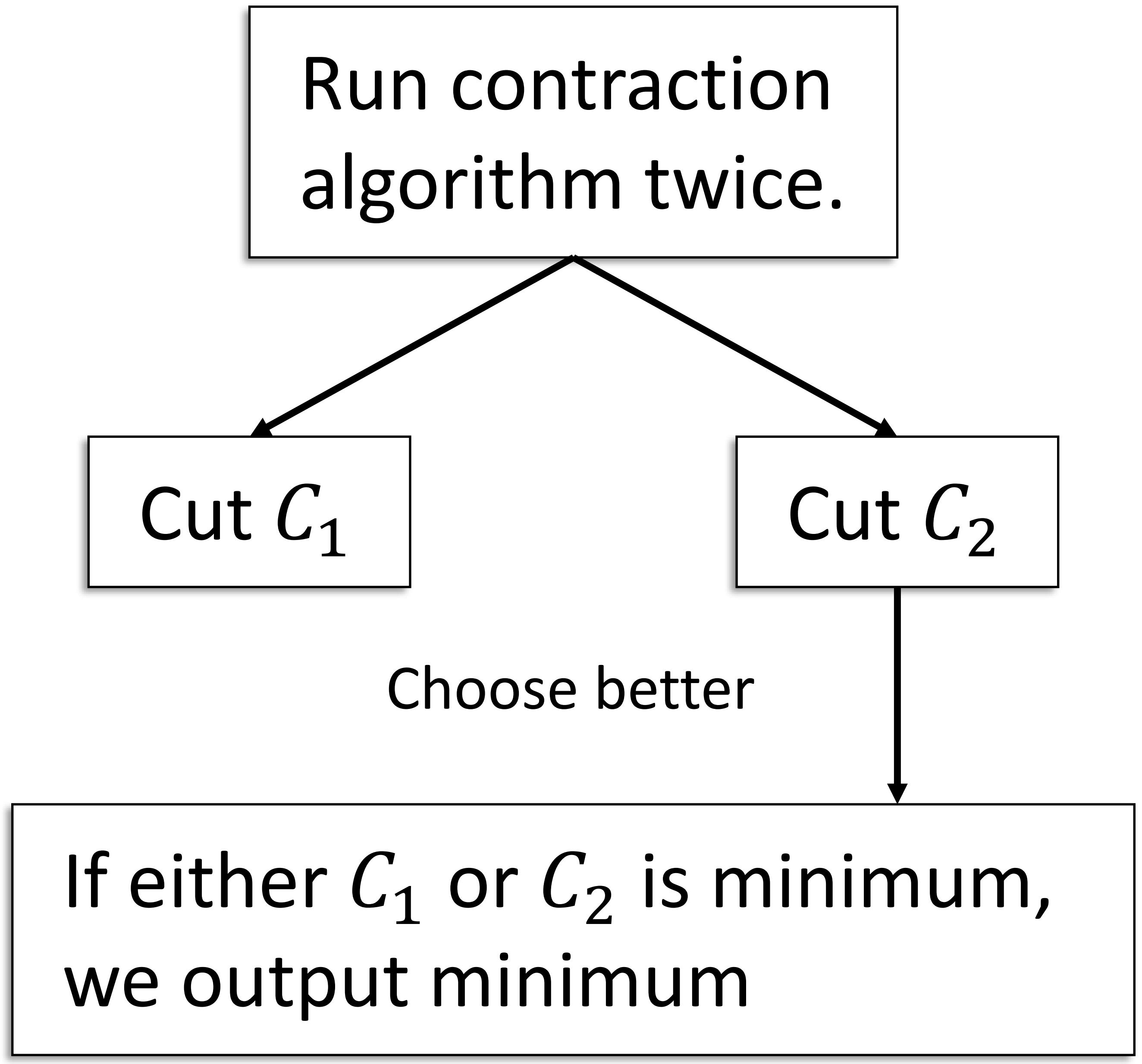
Minimum Cut

Probability boosting:
The algorithm outputs a minimum cut with high probability.

Minimum Cut



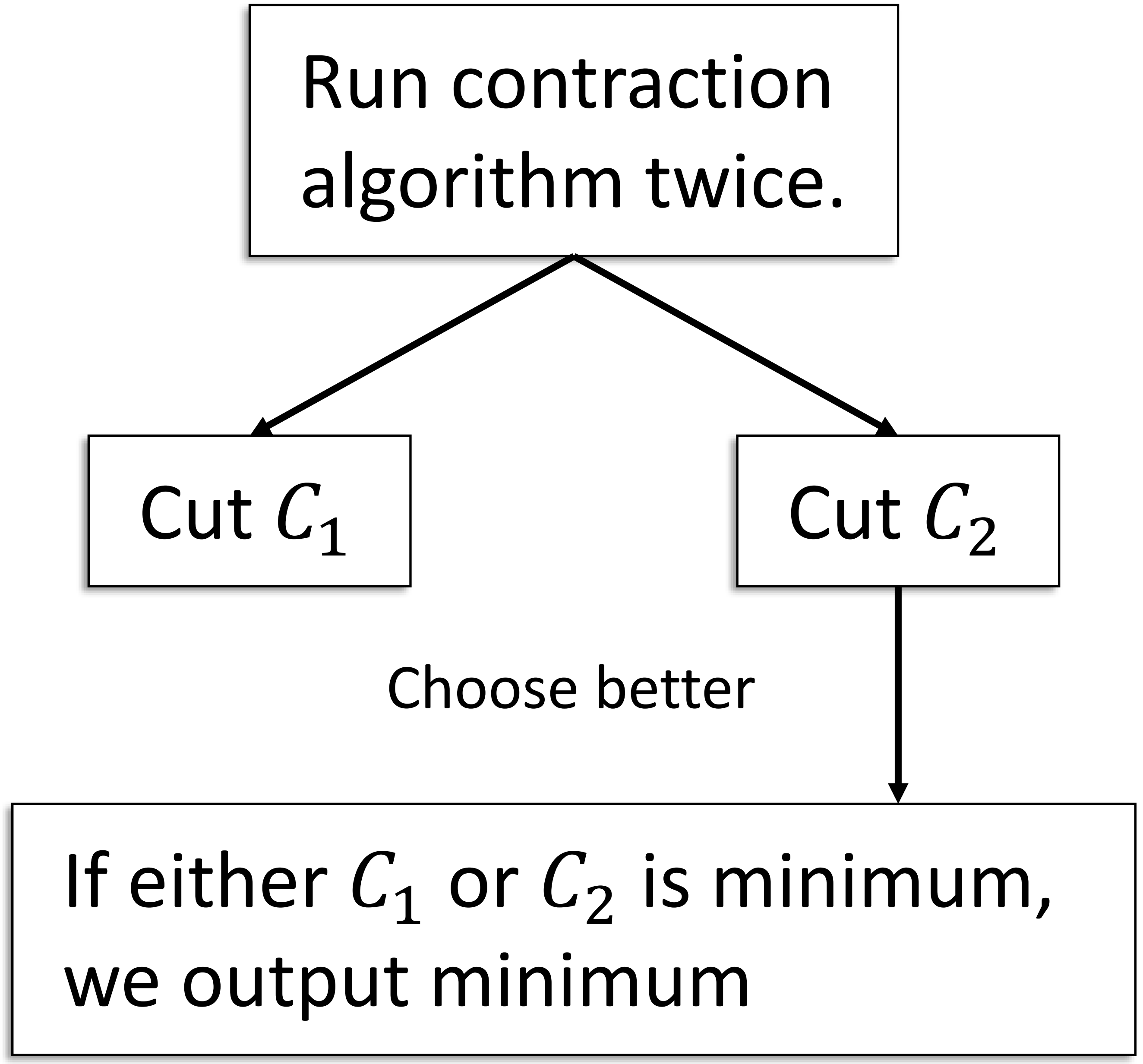
Minimum Cut


Run contraction algorithm twice.
Cut C_1
Cut C_2
Choose better
If either C_1 or C_2 is minimum, we output minimum

Minimum Cut

Theorem:
The contraction algorithm outputs a minimum cut with probability $\frac{2}{n(n-1)} = x$.

$$P(C_1 \text{ and } C_2 \text{ not optimum}) \leq (1-x)^2$$



Minimum Cut

Theorem:
The contraction algorithm outputs a minimum cut with probability $\frac{2}{n(n-1)} = x$.

Run contraction algorithm $O((1/x)\log n)$ times.

Probability that the output is not optimum is at most

$$(1-x)^{\left(\frac{c \log n}{x}\right)}$$

Minimum Cut

Theorem:
The contraction algorithm outputs a minimum cut with probability $\frac{2}{n(n-1)} = x$.

Run contraction algorithm $O((1/x)\log n)$ times.

Probability that the output is not optimum is at most

$$(1-x)^{\left(\frac{c\log n}{x}\right)} \leq (2^{-x})^{\left(\frac{c\log n}{x}\right)}$$

A well-known estimate:

$$1 - x \leq e^{-x} < 2^{-x}$$

Minimum Cut

Run contraction algorithm $O((1/x)\log n)$ times.

Theorem:
The contraction algorithm outputs a minimum cut with probability $\frac{2}{n(n-1)} = x.$

Probability that the output is not optimum is at most
$(1-x)^{\left(\frac{c\log n}{x}\right)} \leq (2^{-x})^{\left(\frac{c\log n}{x}\right)}$
$= 2^{-c\log n} = n^{-c}$

A well-known estimate:
$1 - x \leq e^{-x} < 2^{-x}$

Contraction Algorithm - Analysis

Theorem:
The contraction algorithm outputs a minimum cut with probability $\frac{2}{n(n-1)} = x$.

Contraction Algorithm - Analysis

Theorem:
The contraction algorithm outputs a minimum cut with probability $\frac{2}{n(n-1)} = x$.

Run contraction algorithm $O((1/x)\log n)$ times to obtain w.h.p.

Contraction Algorithm - Analysis

Theorem:
The contraction algorithm outputs a minimum cut with probability $\frac{2}{n(n-1)} = x$.

Observation:
The contraction algorithm runs in polynomial time.

Run contraction algorithm $O((1/x)\log n)$ times to obtain w.h.p.

Contraction Algorithm - Analysis

Theorem:
The contraction algorithm outputs a minimum cut with probability $\frac{2}{n(n-1)} = x$.

Observation:
The contraction algorithm runs in polynomial time.

Run contraction algorithm $O((1/x)\log n)$ times to obtain w.h.p.

Corollary:
There is an algorithm to find a minimum cut in polynomial time w.h.p.

Wrap-Up

Karger's Algorithm

$+$

Probability Boosting

Edge contraction