Karlsruhe Institute of Technology

Probability & Computing

Probability Amplification

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

Probability Amplification

Karlsruhe Institute of Technology

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*

✗ answers are always correct
✓ answers may be incorrect

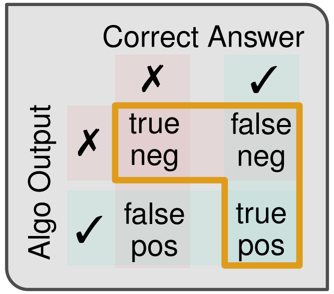
Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*

✓ answers are always correct
✗ answers may be incorrect



Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*
  - **Two-sided error**: *no bias*

✗ answers may be incorrect
✓ answers may be incorrect

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*
  - **Two-sided error**: *no bias*
- In optimization problems p is the probability of finding the optimum

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*
  - **Two-sided error**: *no bias*
- In optimization problems p is the probability of finding the optimum

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Definition: **Probability amplification** is the process of increasing the success probability of a Monte Carlo algorithm by using multiple runs.

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*
  - **Two-sided error**: *no bias*
- In optimization problems p is the probability of finding the optimum

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Definition: **Probability amplification** is the process of increasing the success probability of a Monte Carlo algorithm by using multiple runs.

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*
    - ✓ answers are always correct
    - ✗ answers may be incorrect
  - **Two-sided error**: *no bias*
- In optimization problems p is the probability of finding the optimum

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Definition: **Probability amplification** is the process of increasing the success probability of a Monte Carlo algorithm by using multiple runs.

Probability Amplification for true-biased algorithms

- Execute independently t times.
  - If ✓ at least once: Return ✓. (surely correct)
  - Otherwise: Return ✗. $\Pr[\text{“correct”}] \geq 1 - (1-p)^t \geq 1 - e^{-pt}$

$1 + x \leq e^x$ for $x \in \mathbb{R}$

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*
  - **Two-sided error**: *no bias*
- In optimization problems p is the probability of finding the optimum

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Definition: **Probability amplification** is the process of increasing the success probability of a Monte Carlo algorithm by using multiple runs.

Probability Amplification for true-biased algorithms

Exercise: For two-sided error.

- Execute independently t times.
  - If ✓ at least once: Return ✓. (surely correct)
  - Otherwise: Return ✗.

$$\Pr[\text{“correct”}] \geq 1 - (1-p)^t \geq 1 - e^{-pt}$$

$1 + x \leq e^x$ for $x \in \mathbb{R}$

Probability Amplification

Definition: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least $p \in (0, 1)$.

- In decision problems p is the probability of giving the correct answer
  - **One-sided error**: either *false-biased* or *true-biased*
  - **Two-sided error**: *no bias*
- In optimization problems p is the probability of finding the optimum

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Definition: **Probability amplification** is the process of increasing the success probability of a Monte Carlo algorithm by using multiple runs.

Probability Amplification for optimization algorithms

- Execute independently t times.
  - output best result

$$\Pr[\text{“optimal”}] \geq 1 - (1-p)^t \geq 1 - e^{-pt}$$

The Segmentation Problem

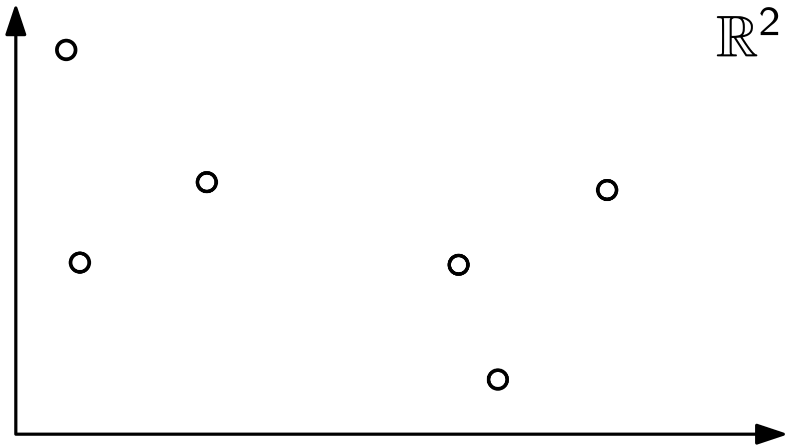
Input

- Set P of points in a feature space (e.g., $\mathbb{R}^d$)
- Similarity measure $\sigma\colon P \times P \mapsto \mathbb{R}_+$

The Segmentation Problem

Input

- Set P of points in a feature space (e.g., $\mathbb{R}^d$)
- Similarity measure $\sigma\colon P \times P \mapsto \mathbb{R}_+$



Example

- six points in $\mathbb{R}^2$
- σ is the inversed Euclidean distance

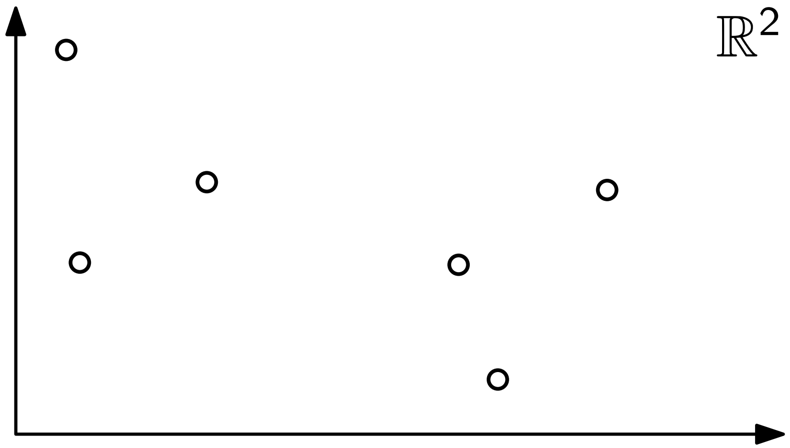
The Segmentation Problem

Input

- Set P of points in a feature space (e.g., $\mathbb{R}^d$)
- Similarity measure $\sigma : P \times P \mapsto \mathbb{R}_+$

Output: $P_1, \ldots, P_k$ such that

- Points within a P_i have high similarity
- Points in distinct P_i, P_j have low similarity



Example

- six points in $\mathbb{R}^2$
- σ is the inversed Euclidean distance
- segment into two sets

The Segmentation Problem

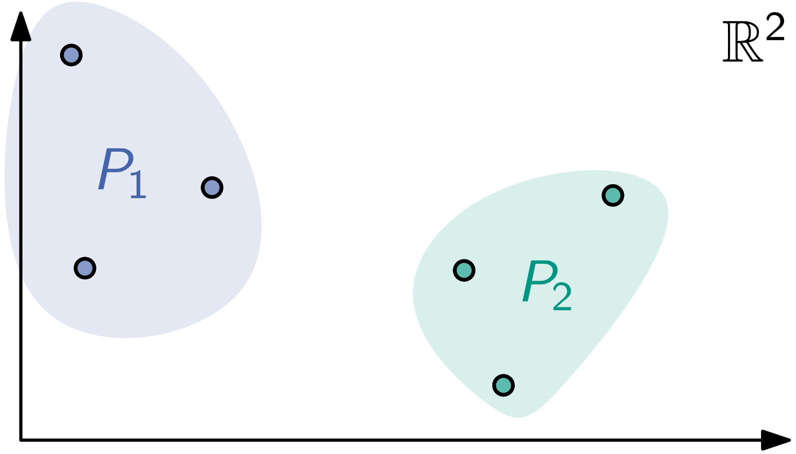
Karlsruhe Institute of Technology

Input

- Set P of points in a feature space (e.g., $\mathbb{R}^d$)
- Similarity measure $\sigma : P \times P \mapsto \mathbb{R}_+$

Output: $P_1, \ldots, P_k$ such that

- Points within a P_i have high similarity
- Points in distinct P_i, P_j have low similarity



Example

- six points in $\mathbb{R}^2$
- σ is the inversed Euclidean distance
- segment into two sets

The Segmentation Problem

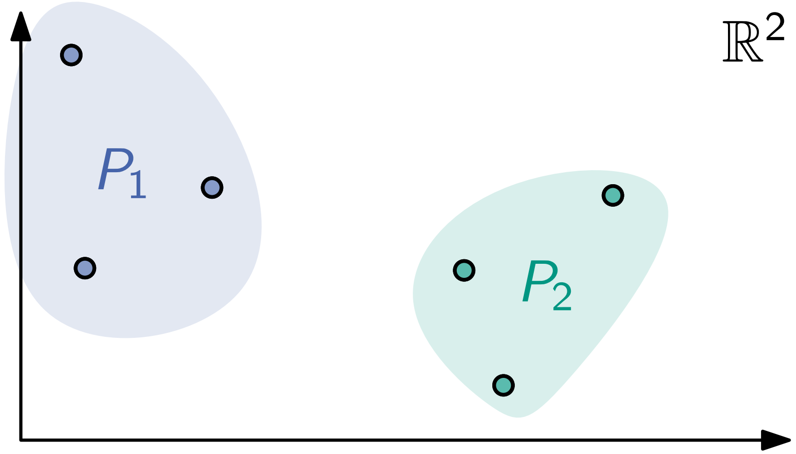
Input

- Set P of points in a feature space (e.g., $\mathbb{R}^d$)
- Similarity measure $\sigma : P \times P \mapsto \mathbb{R}_+$

Output: $P_1, \ldots, P_k$ such that

- Points within a P_i have high similarity
- Points in distinct P_i, P_j have low similarity

Applications: Compression, medical diagnosis, etc.



Example

- six points in $\mathbb{R}^2$
- σ is the inversed Euclidean distance
- segment into two sets

The Segmentation Problem

Input

- Set P of points in a feature space (e.g., $\mathbb{R}^d$)
- Similarity measure $\sigma\colon P \times P \mapsto \mathbb{R}_+$

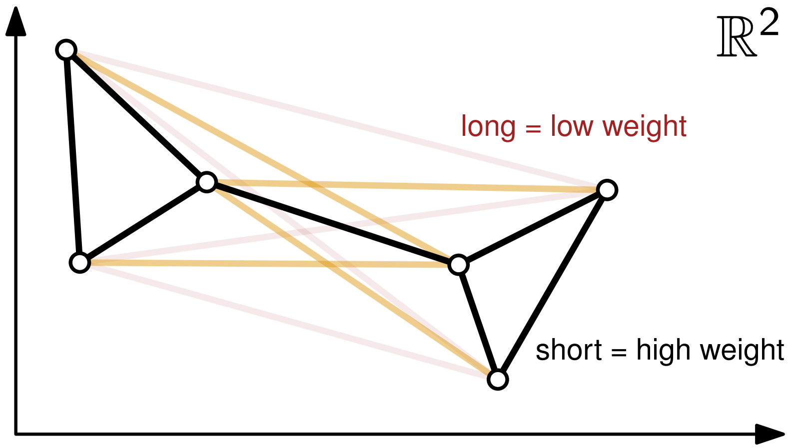
Output: $P_1, \ldots, P_k$ such that

- Points within a P_i have high similarity
- Points in distinct P_i, P_j have low similarity

Applications: Compression, medical diagnosis, etc.

Approach: Model as graph

- Each point is a node
- Edges between all node pairs, with the weight given by the similarity of the two nodes



Example

- six points in $\mathbb{R}^2$
- σ is the inversed Euclidean distance
- segment into two sets

The Segmentation Problem

Input

- Set P of points in a feature space (e.g., $\mathbb{R}^d$)
- Similarity measure $\sigma : P \times P \mapsto \mathbb{R}_+$

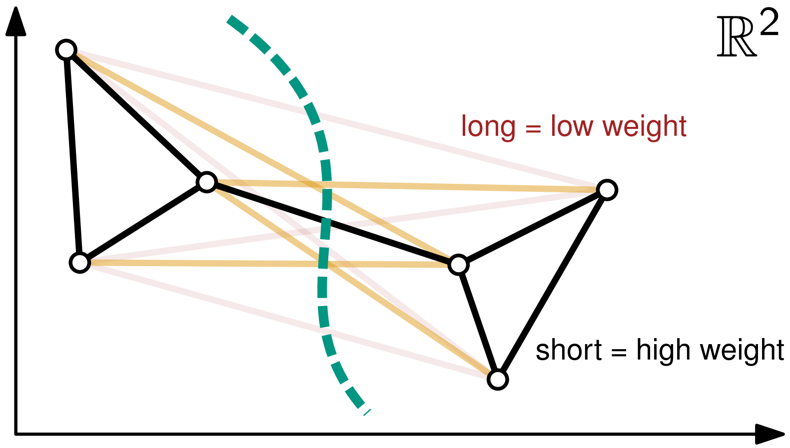
Output: $P_1, \ldots, P_k$ such that

- Points within a P_i have high similarity
- Points in distinct P_i, P_j have low similarity

Applications: Compression, medical diagnosis, etc.

Approach: Model as graph

- Each point is a node
- Edges between all node pairs, with the weight given by the similarity of the two nodes
- Find *cut-set* (edges to remove) of minimal weight such that the graph decomposes into k components.



Example

- six points in $\mathbb{R}^2$
- σ is the inversed Euclidean distance
- segment into two sets

The Segmentation Problem

Input

- Set P of points in a feature space (e.g., $\mathbb{R}^d$)
- Similarity measure $\sigma\colon P \times P \mapsto \mathbb{R}_+$

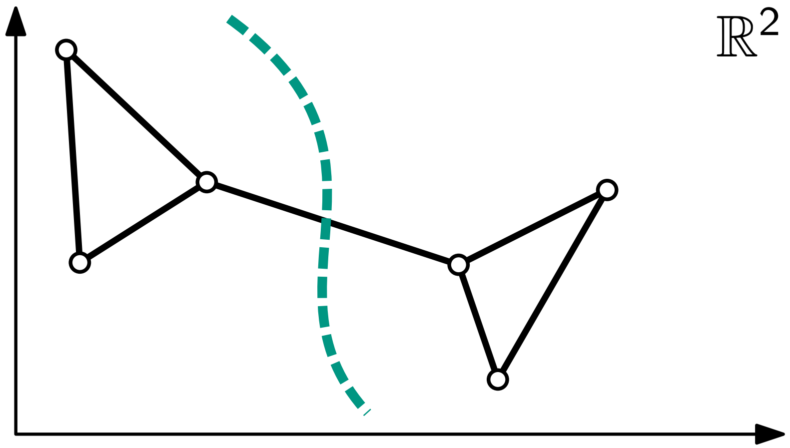
Output: $P_1, \ldots, P_k$ such that

- Points within a P_i have high similarity
- Points in distinct P_i, P_j have low similarity

Applications: Compression, medical diagnosis, etc.

Approach: Model as graph

- Each point is a node
- Edges between all node pairs, with the weight given by the similarity of the two nodes
- Find *cut-set* (edges to remove) of minimal weight such that the graph decomposes into k components.



Example

- six points in $\mathbb{R}^2$
- σ is the inversed Euclidean distance
- segment into two sets

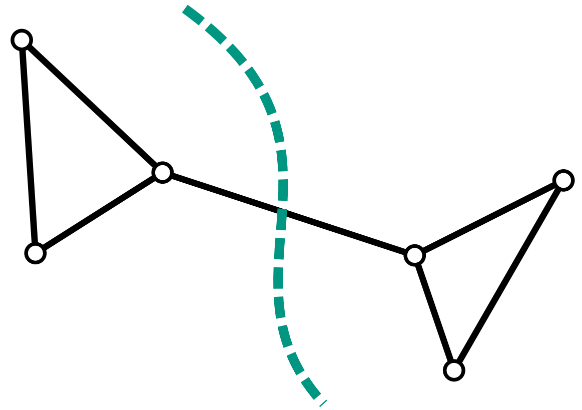
Today

$k = 2$ and $\sigma\colon P \times P \mapsto \{0, 1\}$

Computing Min Cuts

Cuts

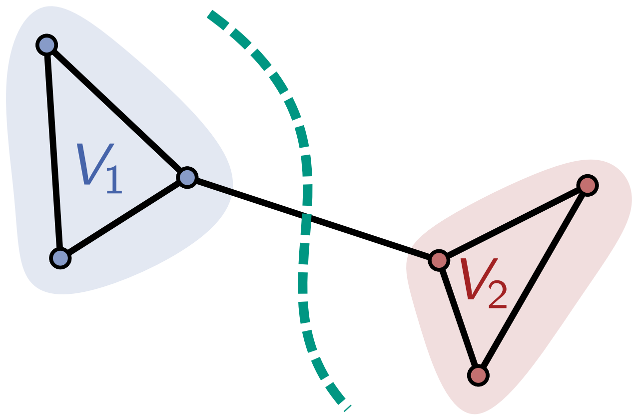
- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of V into non-empty parts V_1, V_2 such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$.



Computing Min Cuts

Cuts

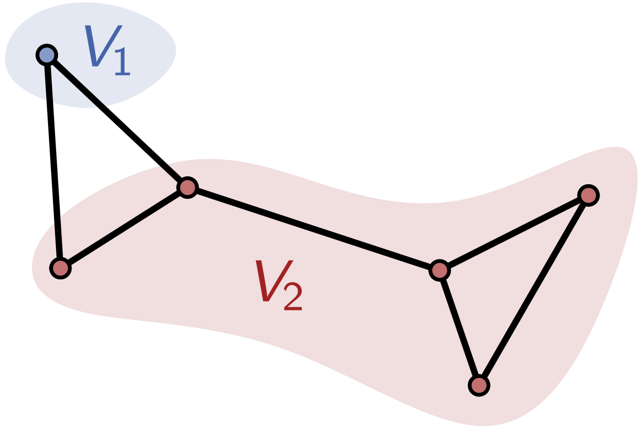
- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of V into non-empty parts V_1, V_2 such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$.



Computing Min Cuts

Cuts

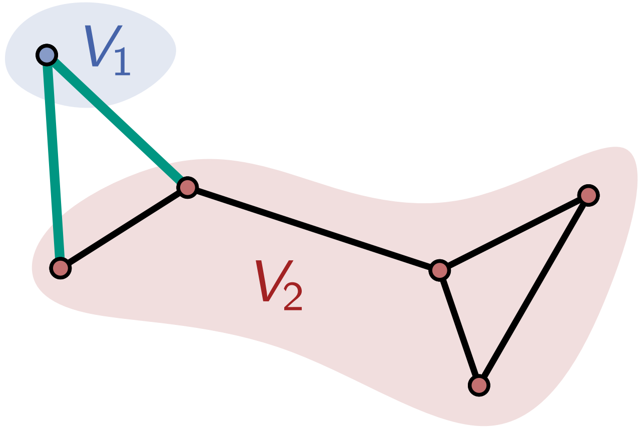
- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of V into non-empty parts V_1, V_2 such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$.



Computing Min Cuts

Cuts

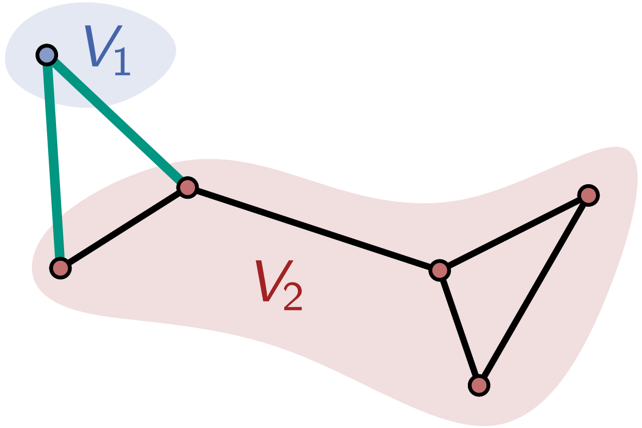
- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of V into non-empty parts V_1, V_2 such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$.
- *Cut-set*: set of edges with an endpoints in V_1 and V_2



Computing Min Cuts

Cuts

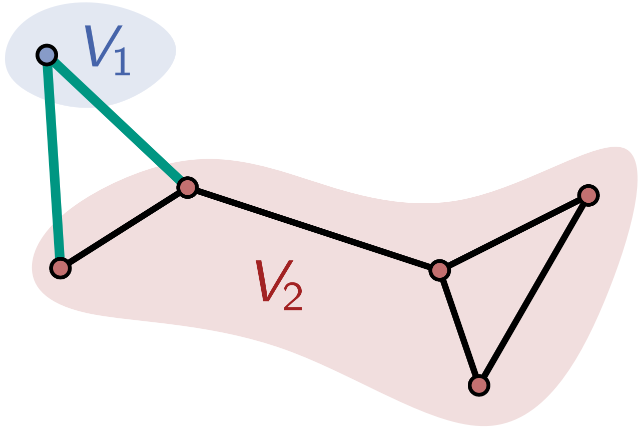
- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of V into non-empty parts V_1, V_2 such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$.
- *Cut-set*: set of edges with an endpoints in V_1 and V_2
- *Weight of a cut*: size of the cut-set (or sum of weights in a weighted graph)



Computing Min Cuts

Cuts

- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of V into non-empty parts V_1, V_2 such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$.
- *Cut-set*: set of edges with an endpoints in V_1 and V_2
- *Weight of a cut*: size of the cut-set (or sum of weights in a weighted graph)



Today Goal: Compute a Min-Cut

i.e. a cut of minimum weight or cut-set of minimum size
the weight of the min-cut is known as the edge-connectivity of G

- Known deterministic strategies have worst case running time $\Omega(n^3)$.
- We'll see randomised algorithm with running time $O(n^2 \cdot \log^3(n))$.

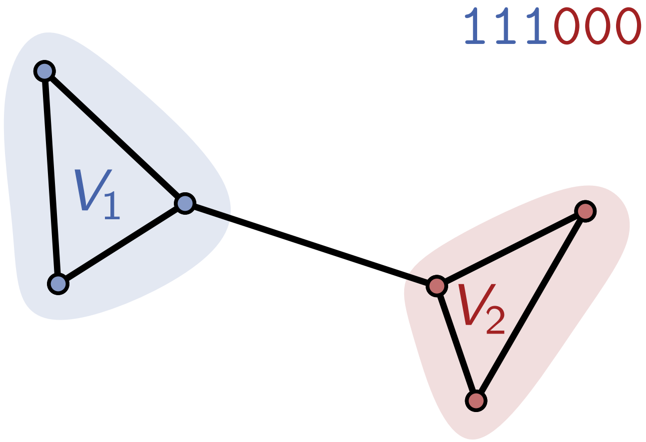
A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

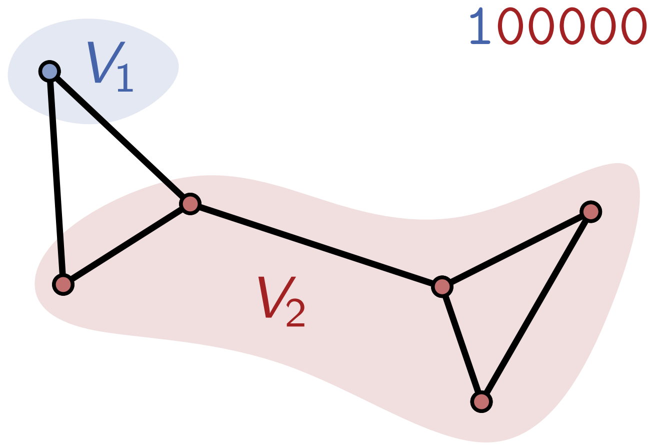
- Number of possible assignments of n nodes to 2 parts 2^n



A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

- Number of possible assignments of n nodes to 2 parts 2^n

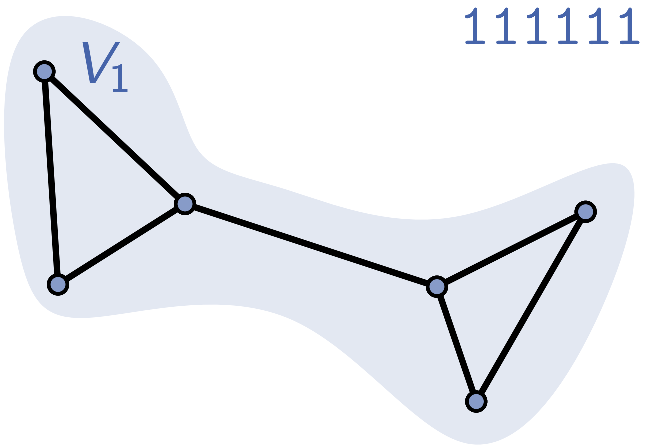


A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$2^n - 2$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts

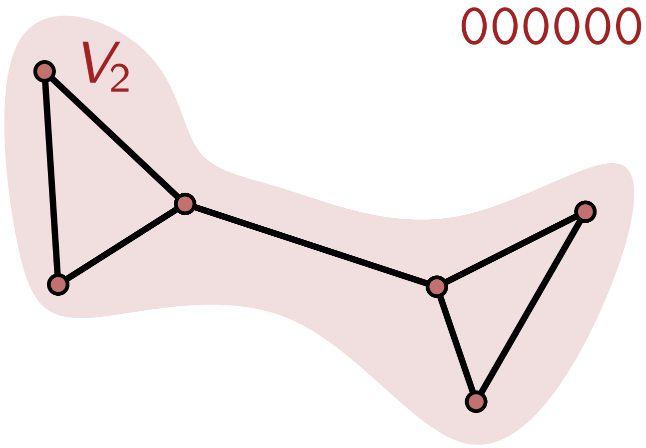


A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$2^n - 2$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts

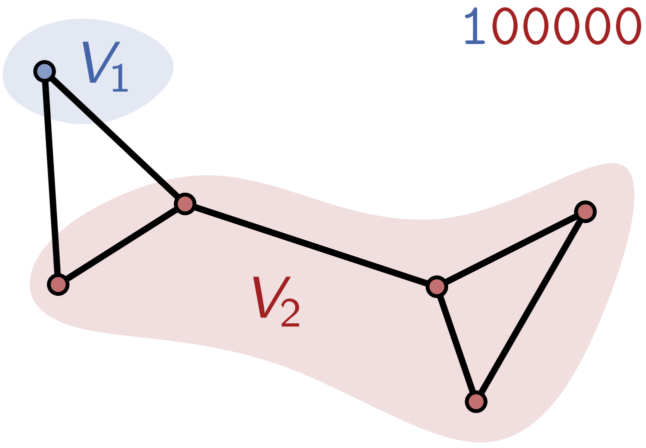


A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$(2^n - 2)/2$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts
- Swapping parts does not yield a new partition

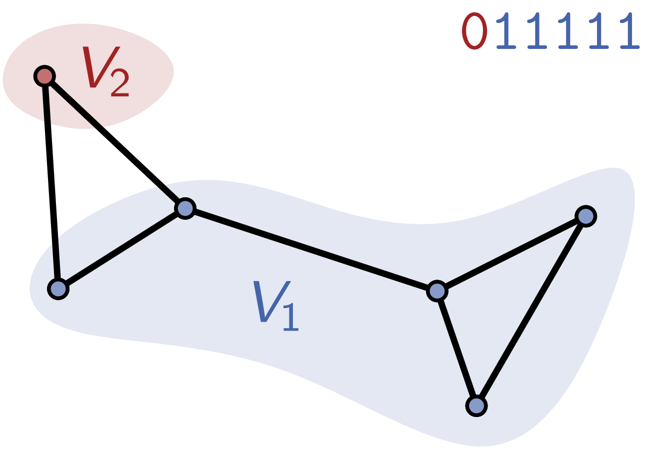


A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$$\left(2^n - 2\right)/2$$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts
- Swapping parts does not yield a new partition

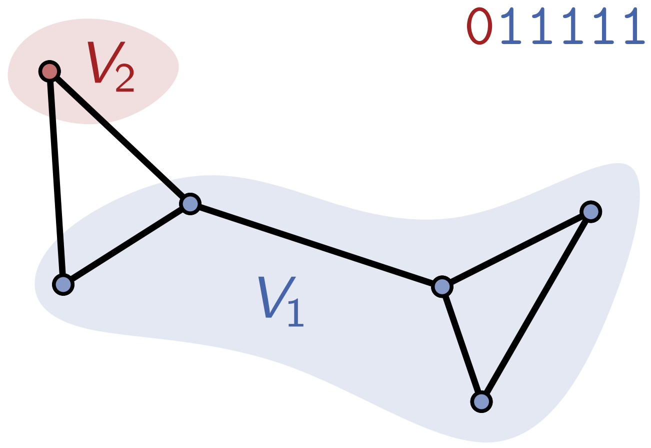


A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$\left(2^n - 2\right) / 2$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts
- Swapping parts does not yield a new partition



Algorithm: Random Cut

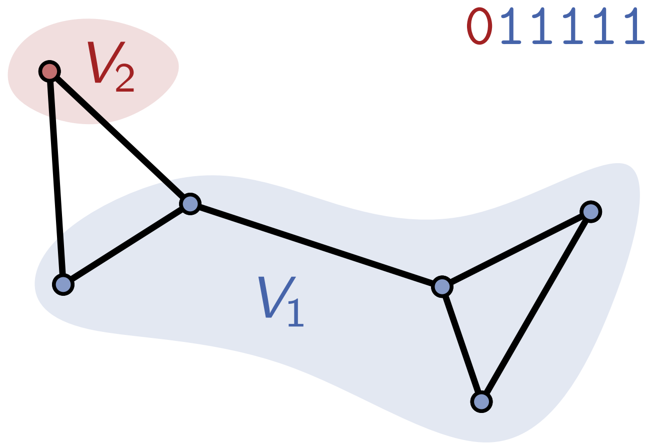
- Return a uniformly random cut.

A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$(2^n - 2)/2$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts
- Swapping parts does not yield a new partition



Algorithm: Random Cut

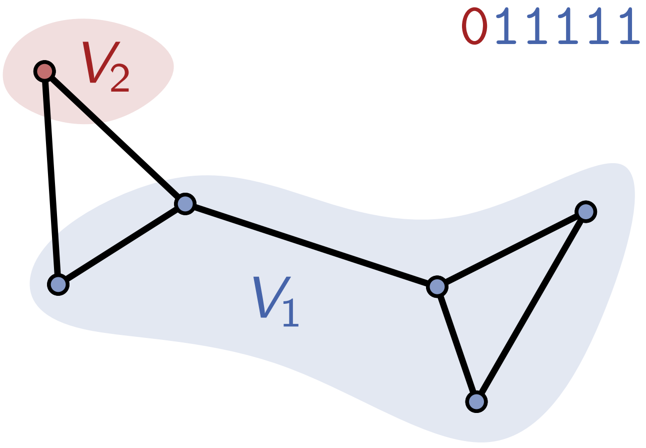
- Return a uniformly random cut.
- Minor challenge: How to uniformly sample cuts?

A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$(2^n - 2)/2$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts
- Swapping parts does not yield a new partition



Algorithm: Random Cut

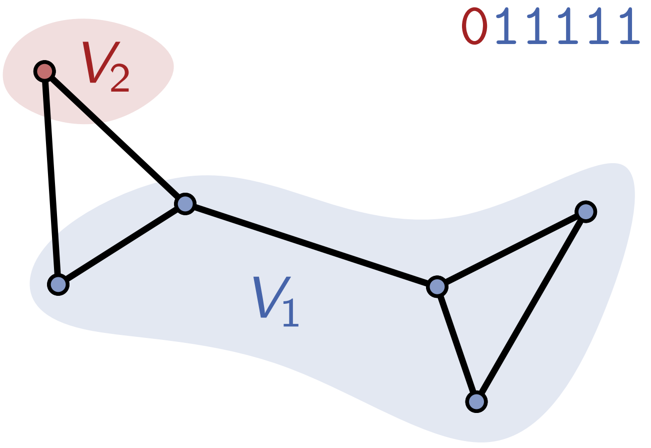
- Return a uniformly random cut.
- Minor challenge: How to uniformly sample cuts?
  - Represent cut using bit-string

A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$(2^n - 2)/2$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts
- Swapping parts does not yield a new partition



Algorithm: Random Cut

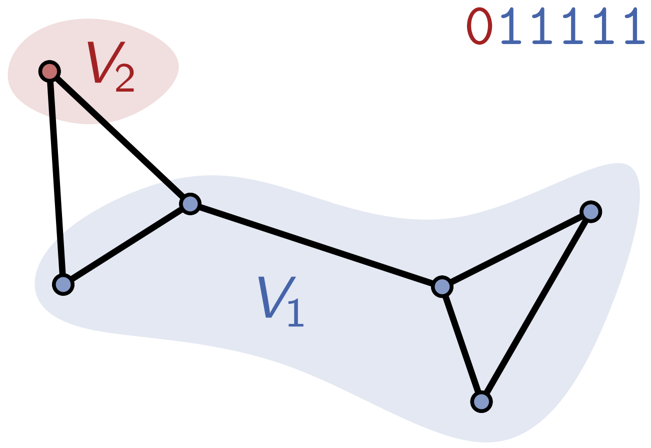
- Return a uniformly random cut.
- Minor challenge: How to uniformly sample cuts?
  - Represent cut using bit-string
  - Have to uniformly sample bit-string *while avoiding* `11...1` and `00...0`?

A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$(2^n - 2) / 2$

- Number of possible assignments of n nodes to 2 parts
- Partitions with empty parts that do not represent cuts
- Swapping parts does not yield a new partition



Algorithm: Random Cut

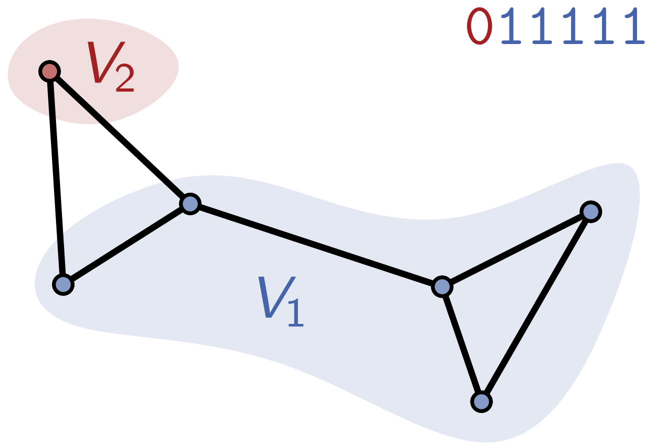
- Return a uniformly random cut.
- Minor challenge: How to uniformly sample cuts?
  - Represent cut using bit-string
  - Have to uniformly sample bit-string *while avoiding* `11...1` and `00...0`?
    - intution: sample from $\mathcal{U}(\{0,1\}^n)$ and use rejection sampling

A Trivial Algorithm: Random Cut

Observation: There are $2^{n-1} - 1$ cuts in a graph with n nodes.

$\left(2^n - 2\right)/2$

- Number of possible assignments of n nodes to 2 parts (2^n)
- Partitions with empty parts that do not represent cuts (-2)
- Swapping parts does not yield a new partition ($/2$)



Algorithm: Random Cut

- Return a uniformly random cut.
- Minor challenge: How to uniformly sample cuts?
  - Represent cut using bit-string
  - Have to uniformly sample bit-string *while avoiding* `11...1` and `00...0`?
    - intution: sample from $\mathcal{U}(\{0,1\}^n)$ and use rejection sampling
    - actually for bounded running time: declare failure rather than sampling again
    - samples each cut with probability $1/2^{n-1}$

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "$=$" if there is only one min-cut.

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "$=$" if there is only one min-cut.

$\rightarrow$ exponentially small!

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "$=$" if there is only one min-cut.

$\rightarrow$ exponentially small!

Amplification

- Repeat the algorithm to obtain t independent random cuts, return the smallest

$$\Pr[\text{"min cut found"}] \geq 1 - \left(1 - 1/2^{n-1}\right)^t$$

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "$=$" if there is only one min-cut.

$\rightarrow$ exponentially small!

Amplification

- Repeat the algorithm to obtain t independent random cuts, return the smallest

$$\Pr[\text{“min cut found”}] \geq 1 - \left(1 - \underbrace{1/2^{n-1}}_{\text{minimum}}\right)^t$$

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "$=$" if there is only one min-cut.

$\rightarrow$ exponentially small!

Amplification

- Repeat the algorithm to obtain t independent random cuts, return the smallest

$$\Pr[\text{"min cut found"}] \geq \underbrace{1 -}_{\text{not}} \left(1 - \underbrace{1/2^{n-1}}_{\text{minimum}}\right)^t$$

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "$=$" if there is only one min-cut.

$\rightarrow$ exponentially small!

Amplification

- Repeat the algorithm to obtain t independent random cuts, return the smallest

$$\Pr[\text{“min cut found”}] \geq \underbrace{1 -}_{\text{not}} \left(1 - \underbrace{1/2^{n-1}}_{\text{minimum}}\right)^{\overset{t}{\downarrow\, t \text{ times}}}$$

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "$=$" if there is only one min-cut.

$\rightarrow$ exponentially small!

Amplification

- Repeat the algorithm to obtain t independent random cuts, return the smallest

$$\Pr[\text{"min cut found"}] \geq 1 - \left(1 - 1/2^{n-1}\right)^t \geq 1 - e^{-t/2^{n-1}}$$

$1 + x \leq e^x$ for $x \in \mathbb{R}$

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "$=$" if there is only one min-cut.

$\rightarrow$ exponentially small!

Amplification

- Repeat the algorithm to obtain t independent random cuts, return the smallest

$$\Pr[\text{"min cut found"}] \geq 1 - \left(1 - 1/2^{n-1}\right)^t \geq 1 - e^{-t/2^{n-1}}$$

$1 + x \leq e^x$ for $x \in \mathbb{R}$

- For $t = 2^{n-1}$ min cut found with constant probability $1 - 1/e \approx 0.63$

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "=" if there is only one min-cut.

$\rightarrow$ exponentially small!

Amplification

- Repeat the algorithm to obtain t independent random cuts, return the smallest

$$\Pr[\text{“min cut found”}] \geq 1 - \left(1 - 1/2^{n-1}\right)^t \geq 1 - e^{-t/2^{n-1}}$$

$1 + x \leq e^x$ for $x \in \mathbb{R}$

- For $t = 2^{n-1}$ min cut found with constant probability $1 - 1/e \approx 0.63$
- For $t = 2^{n-1} \cdot \ln(n)$ min cut found with high probability $1 - 1/n$

Random Cut: Analysis

Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

Success probability: $\geq 1/2^{n-1}$ "=" if there is only one min-cut.

$\rightarrow$ exponentially small!

Amplification

- Repeat the algorithm to obtain t independent random cuts, return the smallest

$$\Pr[\text{“min cut found”}] \geq 1 - \left(1 - 1/2^{n-1}\right)^t \geq 1 - e^{-t/2^{n-1}}$$

$1 + x \leq e^x$ for $x \in \mathbb{R}$

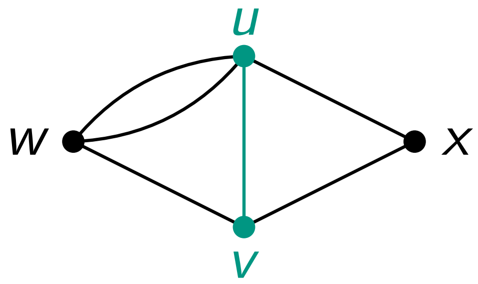
- For $t = 2^{n-1}$ min cut found with constant probability $1 - 1/e \approx 0.63$
- For $t = 2^{n-1} \cdot \ln(n)$ min cut found with high probability $1 - 1/n$

this is terrible so far...

Karger's Algorithm

Edge Contraction

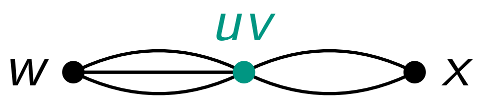
- Merge two adjacent nodes in a multigraph without self-loops



Karger's Algorithm

Edge Contraction

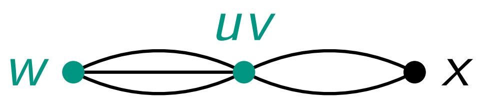
- Merge two adjacent nodes in a multigraph without self-loops



Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops



Karger’s Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set

Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

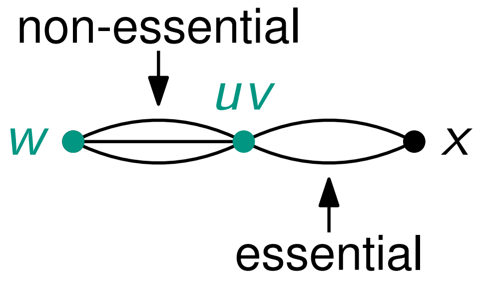
Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set

Contraction Algorithm

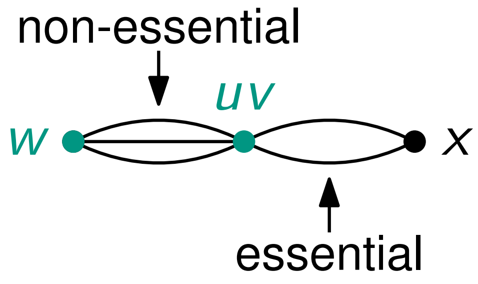
- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones



Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

for $i = 1$ to $n - 2$ **do**

  $e := \mathcal{U}(E_{i-1})$

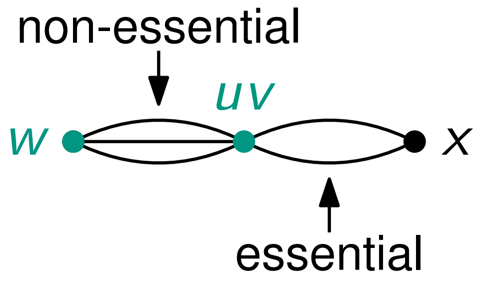
  $G_i = G_{i-1}$.**contract**(e)

return unique cut-set in G_{n-2}

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

  for $i = 1$ to $n - 2$ **do** // $O(n)$

    $e := \mathcal{U}(E_{i-1})$ // $O(1)$

    $G_i = G_{i-1}.$**contract**(e) // $O(n)$

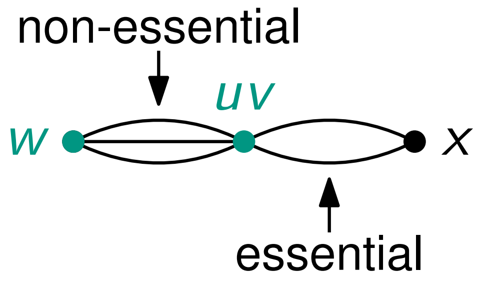
  return unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n - 2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

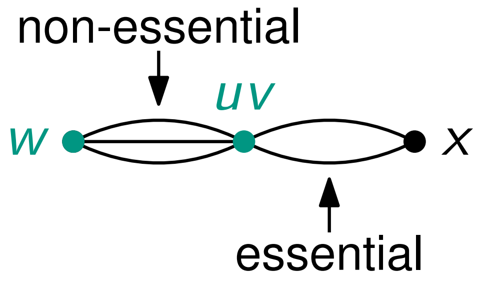
(the converse does not hold)

- Let C be a cut-set in G_i.
  - $G_i \setminus C$ is disconnected
- Assume C is not a cut-set in G_0.
  - $G_0 \setminus C$ is connected.
- $G_i \setminus C$ arises from $G_0 \setminus C$ by i edge contractions.
- $\lightning$ contractions cannot disconnect a graph

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n - 2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

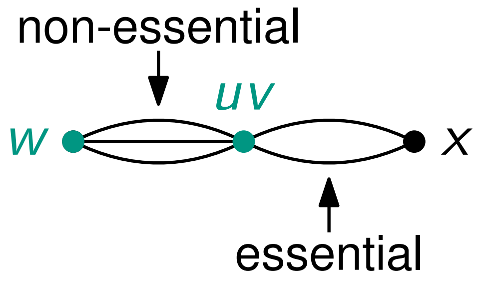
Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n - 2$ **do** — // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ — // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) — // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

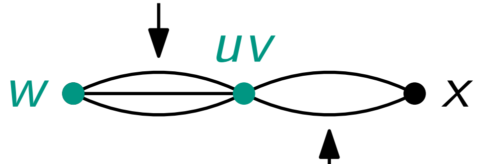
- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

$\Pr[\mathcal{E}_1] = 1 - \frac{k}{m}$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n-2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0. ⚡

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i$ = "C in G_i"

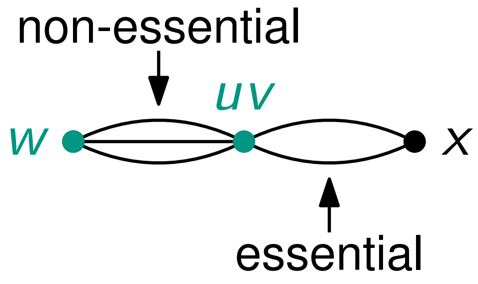
Observation: min-degree $\geq k$

$\Pr[\mathcal{E}_1] = 1 - \frac{k}{m}$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n-2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0. ↯

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

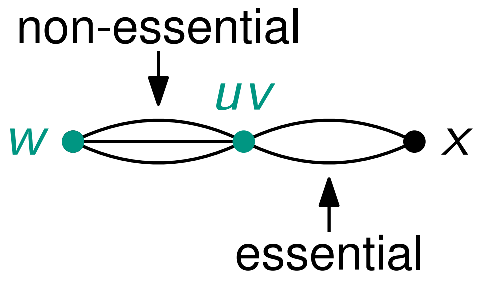
(holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_1] = 1 - \frac{k}{m}$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

for $i = 1$ to $n-2$ **do** // $O(n)$

  $e := \mathcal{U}(E_{i-1})$ // $O(1)$

  $G_i = G_{i-1}$.**contract**(e) // $O(n)$

return unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0. ↯

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

(holds for all G_i due to 1st observation)

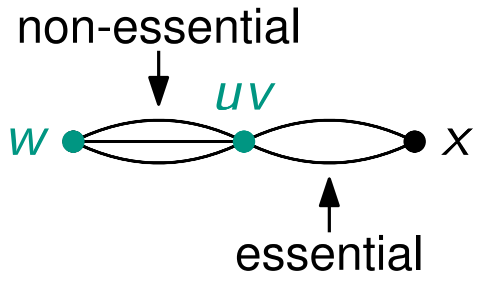
$\Pr[\mathcal{E}_1] = 1 - \frac{k}{m}$

$$m = \frac{1}{2}\sum_{v \in V} \deg(v) \geq \frac{1}{2}\sum_{v \in V} k \geq \frac{1}{2}nk$$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n-2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}.$**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

(holds for all G_i due to 1st observation)

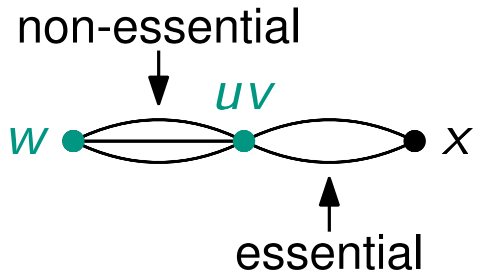
$$\Pr[\mathcal{E}_1] = 1 - \frac{k}{m} \geq 1 - \frac{k}{nk/2} = 1 - \frac{2}{n}$$

$$m = \frac{1}{2}\sum_{v \in V} \deg(v) \geq \frac{1}{2}\sum_{v \in V} k \geq \frac{1}{2} nk$$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n-2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

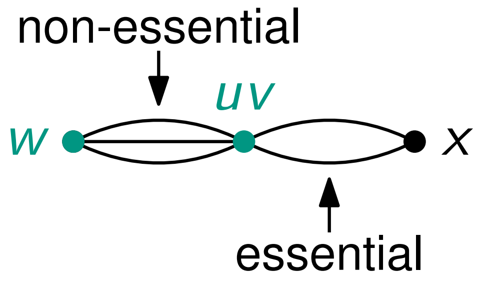
(holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

```
for i = 1 to n − 2 do            // O(n)
    e := U(E_{i−1})              // O(1)
    G_i = G_{i−1}.contract(e)    // O(n)
return unique cut-set in G_{n−2}
```

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

(holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$

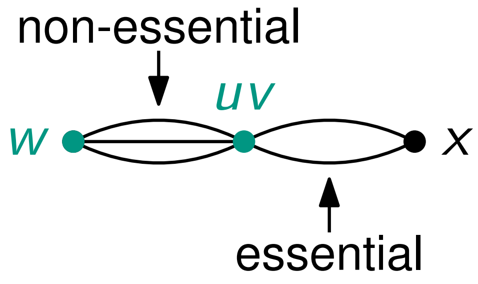
$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1}$

- none of the k edges of C contracted
- do not contract k edges in an $n-1$-node graph

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n - 2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

(holds for all G_i due to 1st observation)

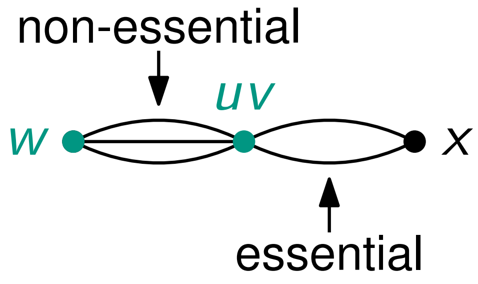
$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n-2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

(holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

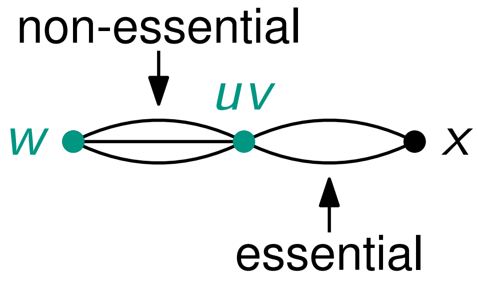
$$\underbrace{\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]}_{\text{chain rule of probability}}$$

Karger's Algorithm

Karlsruhe Institute of Technology

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n-2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

(holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

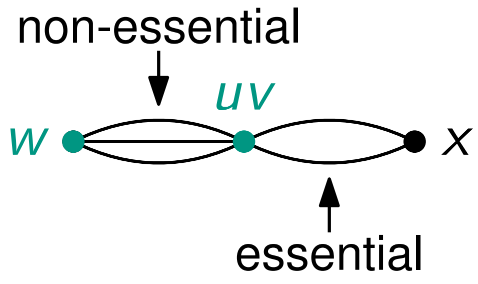
$$\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right)\cdots\left(1 - \frac{2}{4}\right)\left(1 - \frac{2}{3}\right)$$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

```
for i = 1 to n − 2 do               // O(n)
    e := U(E_{i−1})                 // O(1)
    G_i = G_{i−1}.contract(e)       // O(n)
return unique cut-set in G_{n−2}
```

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

(holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

$$\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$$

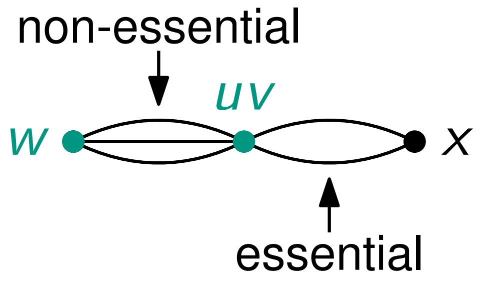
$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right)\cdots\left(1 - \frac{2}{4}\right)\left(1 - \frac{2}{3}\right)$$

$1 = \frac{n-i}{n-i}$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

```
for i = 1 to n − 2 do            // O(n)
    e := U(E_{i−1})              // O(1)
    G_i = G_{i−1}.contract(e)    // O(n)
return unique cut-set in G_{n−2}
```

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

(holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

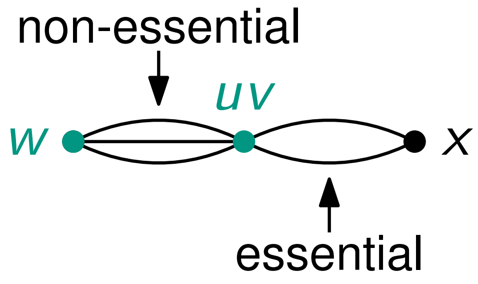
$$\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$$

$$\geq \left(\frac{n}{n} - \frac{2}{n}\right)\left(\frac{n-1}{n-1} - \frac{2}{n-1}\right)\left(\frac{n-2}{n-2} - \frac{2}{n-2}\right) \cdots \left(\frac{4}{4} - \frac{2}{4}\right)\left(\frac{3}{3} - \frac{2}{3}\right)$$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

```
for i = 1 to n − 2 do              // O(n)
    e := U(E_{i−1})                 // O(1)
    G_i = G_{i−1}.contract(e)       // O(n)
return unique cut-set in G_{n−2}
```

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$ (holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

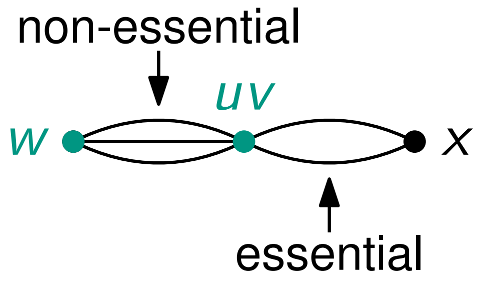
$$\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$$

$$\geq \left(\frac{n-2}{n}\right)\left(\frac{n-1-2}{n-1}\right)\left(\frac{n-2-2}{n-2}\right)\cdots\left(\frac{4-2}{4}\right)\left(\frac{3-2}{3}\right)$$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

```
for i = 1 to n − 2 do            // O(n)
    e := U(E_{i−1})              // O(1)
    G_i = G_{i−1}.contract(e)    // O(n)
return unique cut-set in G_{n−2}
```

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$ (holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

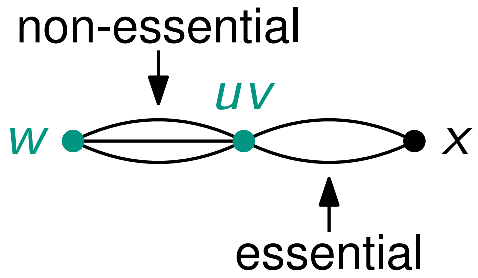
$$\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$$

$$\geq \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{2}{4}\right)\left(\frac{1}{3}\right)$$

Karger's Algorithm

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n-2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}.$**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

(holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$

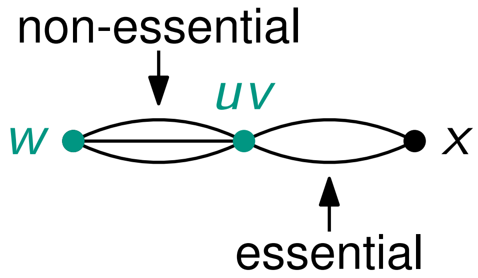
$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

$$\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$$

$$\geq \left(\frac{\cancel{n-2}}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{\cancel{n-2}}\right)\cdots\left(\frac{2}{4}\right)\left(\frac{1}{3}\right)$$

Karger's Algorithm

Edge Contraction
- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm
- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n-2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$ (holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

$$\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$$

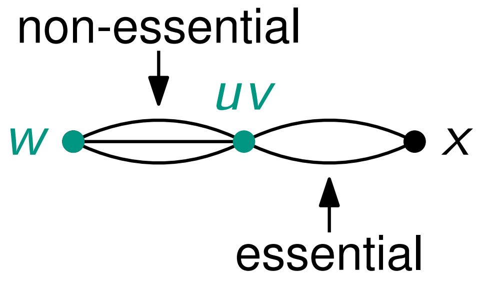
$$\geq \left(\frac{\cancel{n-2}}{n}\right)\left(\frac{\cancel{n-3}}{n-1}\right)\left(\frac{\cancel{n-4}}{\cancel{n-2}}\right)\cdots\left(\frac{2}{\cancel{4}}\right)\left(\frac{1}{\cancel{3}}\right)$$

Karger's Algorithm

Karlsruhe Institute of Technology

Edge Contraction

- Merge two adjacent nodes in a multigraph without self-loops
- A (multi) graph with two nodes has a unique cut-set



Contraction Algorithm

- Motivation: distinguish *non-essential* (not part of a min-cut) as well as *essential* edges (part of a min-cut) & hope there are few essential ones

Karger$(G_0 = (V_0, E_0))$

- **for** $i = 1$ to $n - 2$ **do** // $O(n)$
  - $e := \mathcal{U}(E_{i-1})$ // $O(1)$
  - $G_i = G_{i-1}$.**contract**(e) // $O(n)$
- **return** unique cut-set in G_{n-2}

- Running time in $O(n^2)$
- Can be implemented to run in $O(m)$

Success Probability

Observation: A cut-set in G_i is a cut-set in G_0.

- Consider min-cut in G_0 with cut-set C and $|C| = k$
- $\mathcal{E}_i =$ "C in G_i"

Observation: min-degree $\geq k$

$\Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$ (holds for all G_i due to 1st observation)

$\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1 - \frac{2}{n-1} \longrightarrow \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}] \geq 1 - \frac{2}{n-i+1}$

$$\Pr[\mathcal{E}_{n-2}] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_{n-2} \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$$

$$\geq \left(\frac{\cancel{n-2}}{n}\right)\left(\frac{\cancel{n-3}}{n-1}\right)\left(\frac{\cancel{n-4}}{\cancel{n-2}}\right)\cdots\left(\frac{2}{\cancel{4}}\right)\left(\frac{1}{\cancel{3}}\right)$$

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

Karger's Algorithm Amplified

Karlsruhe Institute of Technology

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

Karger's Algorithm Amplified

Karlsruhe Institute of Technology

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

Success probability $\geq p$

Number of repetitions t

Amplified prob. $\geq 1 - e^{-pt}$

Karger's Algorithm Amplified

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp(-\tfrac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$$

for $t = \frac{n^2}{2} \ln(n)$

Success probability $\geq p$

Number of repetitions t

Amplified prob. $\geq 1 - e^{-pt}$

Karger's Algorithm Amplified

Karlsruhe Institute of Technology

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$$

for $t = \frac{n^2}{2} \ln(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(n^2 \log(n))$ Karger repetitions run in $O(n^4 \log(n))$ total time and return a min-cut with high probability.

Karger's Algorithm Amplified

Karlsruhe Institute of Technology

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp(-\tfrac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$$

for $t = \frac{n^2}{2} \ln(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(n^2 \log(n))$ Karger repetitions run in $O(n^4 \log(n))$ total time and return a min-cut with high probability.

Much better than exp. time of Randomized Cut!

Karger's Algorithm Amplified

Karlsruhe Institute of Technology

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp(-\tfrac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$$

for $t = \frac{n^2}{2} \ln(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(n^2 \log(n))$ Karger repetitions run in $O(n^4 \log(n))$ total time and return a min-cut with high probability.

Much better than exp. time of Randomized Cut!

Sidenote: Number of minimum cuts

- Let $C_1, \ldots, C_\ell$ be all the min-cuts in G and $\mathcal{E}^i_{n-2}$ for $i \in [\ell]$ be the event that C_i is returned by Karger's algorithm

Karger's Algorithm Amplified

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp(-\tfrac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$$

for $t = \frac{n^2}{2} \ln(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(n^2 \log(n))$ Karger repetitions run in $O(n^4 \log(n))$ total time and return a min-cut with high probability. Much better than exp. time of Randomized Cut!

Sidenote: Number of minimum cuts

- Let $C_1, \ldots, C_\ell$ be all the min-cuts in G and $\mathcal{E}^i_{n-2}$ for $i \in [\ell]$ be the event that C_i is returned by Karger's algorithm
- Just seen: $\Pr[\mathcal{E}^i_{n-2}] \geq \frac{2}{n^2}$

Karger's Algorithm Amplified

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp(-\tfrac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$$

for $t = \frac{n^2}{2} \ln(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(n^2 \log(n))$ Karger repetitions run in $O(n^4 \log(n))$ total time and return a min-cut with high probability.

Much better than exp. time of Randomized Cut!

Sidenote: Number of minimum cuts

- Let $C_1, \ldots, C_\ell$ be all the min-cuts in G and $\underbrace{\mathcal{E}^i_{n-2}}$ for $i \in [\ell]$ be the event that C_i is returned by Karger's algorithm

  disjoint, since the algorithm returns only one cut
- Just seen: $\Pr[\mathcal{E}^i_{n-2}] \geq \frac{2}{n^2}$

Karger's Algorithm Amplified

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp(-\tfrac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$$

for $t = \frac{n^2}{2} \ln(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(n^2 \log(n))$ Karger repetitions run in $O(n^4 \log(n))$ total time and return a min-cut with high probability. Much better than exp. time of Randomized Cut!

Sidenote: Number of minimum cuts

- Let $C_1, \ldots, C_\ell$ be all the min-cuts in G and $\mathcal{E}^i_{n-2}$ for $i \in [\ell]$ be the event that C_i is returned by Karger's algorithm (disjoint, since the algorithm returns only one cut)
- Just seen: $\Pr[\mathcal{E}^i_{n-2}] \geq \frac{2}{n^2}$

$$\Pr\left[\bigcup_{i\in[\ell]} \mathcal{E}^i_{n-2}\right] = \sum_{i\in[\ell]} \Pr[\mathcal{E}^i_{n-2}] \geq \frac{2\cdot\ell}{n^2}$$

Karger's Algorithm Amplified

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp(-\tfrac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$$

for $t = \frac{n^2}{2} \ln(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(n^2 \log(n))$ Karger repetitions run in $O(n^4 \log(n))$ total time and return a min-cut with high probability. Much better than exp. time of Randomized Cut!

Sidenote: Number of minimum cuts

- Let $C_1, \ldots, C_\ell$ be all the min-cuts in G and $\mathcal{E}^i_{n-2}$ for $i \in [\ell]$ be the event that C_i is returned by Karger's algorithm
  (disjoint, since the algorithm returns only one cut)
- Just seen: $\Pr[\mathcal{E}^i_{n-2}] \geq \frac{2}{n^2}$

$$1 \geq \Pr\left[\bigcup_{i\in[\ell]} \mathcal{E}^i_{n-2}\right] = \sum_{i\in[\ell]} \Pr[\mathcal{E}^i_{n-2}] \geq \frac{2\cdot\ell}{n^2}$$

Karger's Algorithm Amplified

Theorem: On a graph with n nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$.

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp(-\tfrac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$$

for $t = \frac{n^2}{2} \ln(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(n^2 \log(n))$ Karger repetitions run in $O(n^4 \log(n))$ total time and return a min-cut with high probability. Much better than exp. time of Randomized Cut!

Sidenote: Number of minimum cuts

- Let $C_1, \ldots, C_\ell$ be all the min-cuts in G and $\mathcal{E}^i_{n-2}$ for $i \in [\ell]$ be the event that C_i is returned by Karger's algorithm
  (disjoint, since the algorithm returns only one cut)
- Just seen: $\Pr[\mathcal{E}^i_{n-2}] \geq \frac{2}{n^2}$

$$1 \geq \Pr\left[\bigcup_{i\in[\ell]} \mathcal{E}^i_{n-2}\right] = \sum_{i\in[\ell]} \Pr[\mathcal{E}^i_{n-2}] \geq \frac{2\cdot\ell}{n^2}$$

Observation: $\ell \leq \frac{n^2}{2}$.

More Amplification: Karger-Stein

Karlsruhe Institute of Technology

Motivation

- Probability that a min-cut survives i contractions

$$\Pr[\mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}]$$

$$\geq \left(1 - \tfrac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right)\cdots\left(1 - \frac{2}{n-i+2}\right)\left(1 - \frac{2}{n-i+1}\right)$$

More Amplification: Karger-Stein

Motivation

- Probability that a min-cut survives i contractions

$$\Pr[\mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}]$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{n-i+2}\right)\left(1 - \frac{2}{n-i+1}\right)$$

$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right) \cdots \left(\frac{n-i+2-2}{n-i+2}\right)\left(\frac{n-i+1-2}{n-i+1}\right)$$

More Amplification: Karger-Stein

Karlsruhe Institute of Technology

Motivation

- Probability that a min-cut survives i contractions

$$\Pr[\mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}]$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right)\cdots\left(1 - \frac{2}{n-i+2}\right)\left(1 - \frac{2}{n-i+1}\right)$$

$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{n-i}{n-i+2}\right)\left(\frac{n-i-1}{n-i+1}\right)$$

More Amplification: Karger-Stein

Motivation

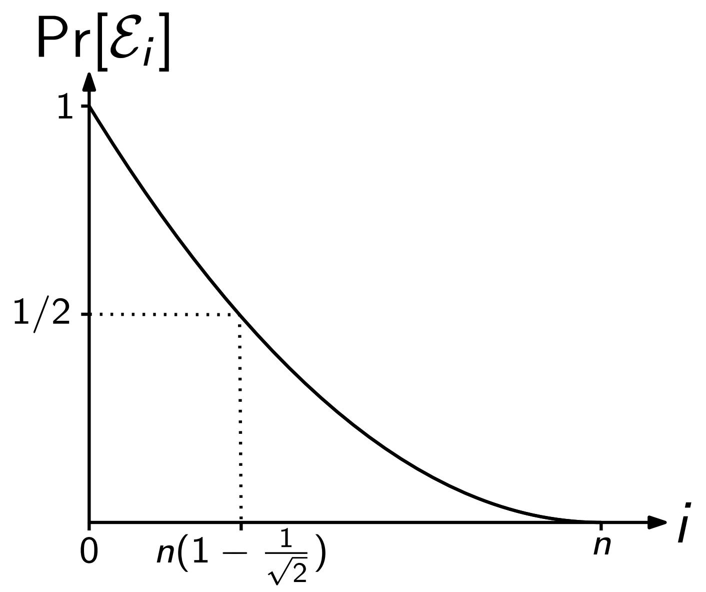
- Probability that a min-cut survives i contractions

$$\Pr[\mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}]$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{n-i+2}\right)\left(1 - \frac{2}{n-i+1}\right)$$

$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right) \cdots \left(\frac{n-i}{n-i+2}\right)\left(\frac{n-i-1}{n-i+1}\right)$$

$$= \frac{(n-i)(n-i-1)}{n(n-1)} \geq \frac{(n-i-1)(n-i-1)}{n \cdot n} = \left(1 - \frac{i+1}{n}\right)^2.$$



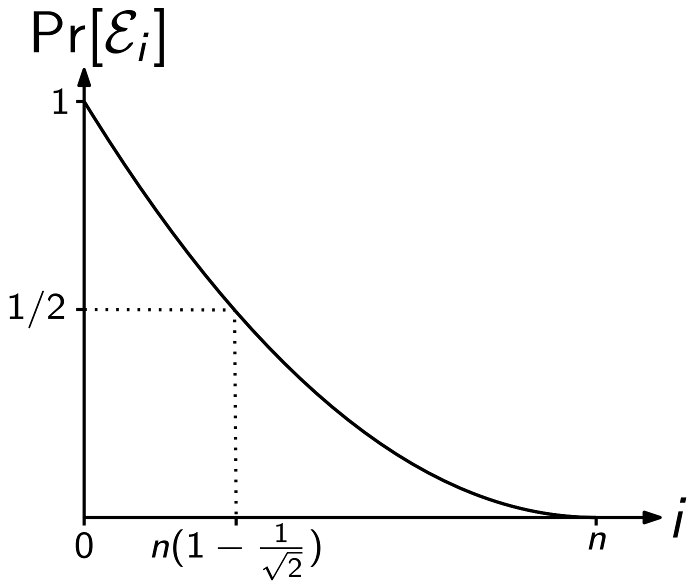
More Amplification: Karger-Stein

Motivation

- Probability that a min-cut survives i contractions

$$\Pr[\mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}]$$
$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right)\cdots\left(1 - \frac{2}{n-i+2}\right)\left(1 - \frac{2}{n-i+1}\right)$$
$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{n-i}{n-i+2}\right)\left(\frac{n-i-1}{n-i+1}\right)$$
$$= \frac{(n-i)(n-i-1)}{n(n-1)} \geq \frac{(n-i-1)(n-i-1)}{n \cdot n} = \left(1 - \frac{i+1}{n}\right)^2.$$

- Probability becomes very small only towards the very end.



More Amplification: Karger-Stein

Motivation

- Probability that a min-cut survives i contractions

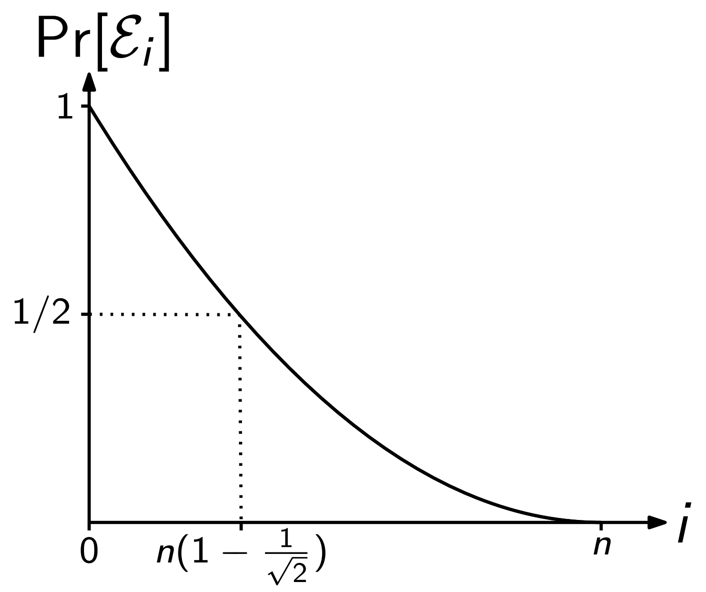
$$\Pr[\mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}]$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{n-i+2}\right)\left(1 - \frac{2}{n-i+1}\right)$$

$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right) \cdots \left(\frac{n-i}{n-i+2}\right)\left(\frac{n-i-1}{n-i+1}\right)$$

$$= \frac{(n-i)(n-i-1)}{n(n-1)} \geq \frac{(n-i-1)(n-i-1)}{n \cdot n} = \left(1 - \frac{i+1}{n}\right)^2.$$

- Probability becomes very small only towards the very end.
- Idea: stop when a min-cut is still likely to exist and recurse



More Amplification: Karger-Stein

Motivation

- Probability that a min-cut survives i contractions

$$\Pr[\mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}]$$

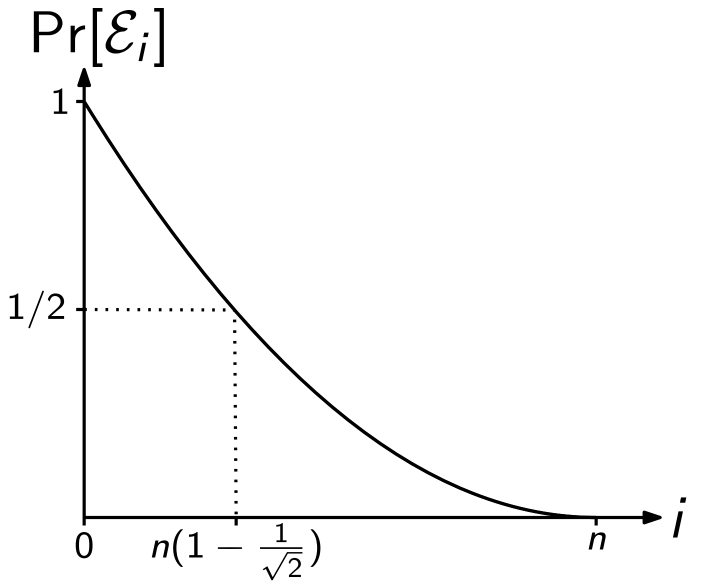
$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right)\cdots\left(1 - \frac{2}{n-i+2}\right)\left(1 - \frac{2}{n-i+1}\right)$$

$$= \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{n-i}{n-i+2}\right)\left(\frac{n-i-1}{n-i+1}\right)$$

$$= \frac{(n-i)(n-i-1)}{n(n-1)} \geq \frac{(n-i-1)(n-i-1)}{n \cdot n} = \left(1 - \frac{i+1}{n}\right)^2.$$

- Probability becomes very small only towards the very end.
- Idea: stop when a min-cut is still likely to exist and recurse
- After $s = n - n/\sqrt{2} - 1$ steps we have

$$\Pr[\mathcal{E}_s] \geq \left(1 - \frac{n - n/\sqrt{2}}{n}\right) = \left(1 - (1 - 1/\sqrt{2})\right)^2 = (1/\sqrt{2})^2 = \frac{1}{2}$$



More Amplification: Karger-Stein

Motivation

- Probability that a min-cut survives i contractions

$$\Pr[\mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot \Pr[\mathcal{E}_i \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}]$$

$$\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{n-i+2}\right)\left(1 - \frac{2}{n-i+1}\right)$$

$$= \left(\frac{\cancel{n-2}}{n}\right)\left(\frac{\cancel{n-3}}{n-1}\right)\left(\frac{\cancel{n-4}}{\cancel{n-2}}\right) \cdots \left(\frac{n-i}{\cancel{n-i+2}}\right)\left(\frac{n-i-1}{\cancel{n-i+1}}\right)$$

$$= \frac{(n-i)(n-i-1)}{n(n-1)} \geq \frac{(n-i-1)(n-i-1)}{n \cdot n} = \left(1 - \frac{i+1}{n}\right)^2.$$

- Probability becomes very small only towards the very end.

KargerStein$(G_0 = (V_0, E_0))$
- **if** $|V_0| = 2$ **then** return unique cut-set
- **for** $i = 1$ to $s = |V_0| - \frac{|V_0|}{\sqrt{2}} - 1$ **do**
  - $e := \mathcal{U}(E_{i-1})$
  - $G_i = G_{i-1}$.**contract**(e)
- $C_1 :=$ **KargerStein**(G_s) // inde-
- $C_2 :=$ **KargerStein**(G_s) // pendent // runs
- **return** smaller of C_1, C_2

- Idea: stop when a min-cut is still likely to exist and recurse
- After $s = n - n/\sqrt{2} - 1$ steps we have

$$\Pr[\mathcal{E}_s] \geq \left(1 - \frac{n - n/\sqrt{2}}{n}\right) = \left(1 - (1 - 1/\sqrt{2})\right)^2 = (1/\sqrt{2})^2 = \frac{1}{2}$$

Karger-Stein: Running Time

```
          KargerStein(G_0 = (V_0, E_0))
// O(1)     if |V_0| = 2 then return unique cut-set
// O(n)     for i = 1 to s = |V_0| - |V_0|/√2 - 1 do
// O(1)       e := U(E_{i-1})
// O(n)       G_i = G_{i-1}.contract(e)
            C_1 := KargerStein(G_s)   // inde-
                                      // pendent
            C_2 := KargerStein(G_s)   // runs
            return smaller of C_1, C_2
```

Karger-Stein: Running Time

Karlsruhe Institute of Technology

Recursion

- After $t = n - n/\sqrt{2} - 1$ steps the number of nodes is $n/\sqrt{2} + 1$

$$T(n) = 2T\left(\frac{n}{\sqrt{2}} + 1\right) + O(n^2)$$

```
KargerStein(G_0 = (V_0, E_0))
    if |V_0| = 2 then return unique cut-set            // O(1)
    for i = 1 to s = |V_0| - |V_0|/√2 - 1 do           // O(n)
        e := U(E_{i-1})                                 // O(1)
        G_i = G_{i-1}.contract(e)                       // O(n)
    C_1 := KargerStein(G_s)    // inde-
    C_2 := KargerStein(G_s)    // pendent
                               // runs
    return smaller of C_1, C_2
```

Karger-Stein: Running Time

Recursion

- After $t = n - n/\sqrt{2} - 1$ steps the number of nodes is $n/\sqrt{2} + 1$

$$T(n) = 2T\left(\frac{n}{\sqrt{2}} + 1\right) + O(n^2)$$

Solution (essentially by Master Theorem)

$$T(n) = O(n^2 \log n)$$

KargerStein$(G_0 = (V_0, E_0))$
- // $O(1)$ **if** $|V_0| = 2$ **then** return unique cut-set
- // $O(n)$ **for** $i = 1$ to $s = |V_0| - \frac{|V_0|}{\sqrt{2}} - 1$ **do**
  - // $O(1)$ $e := \mathcal{U}(E_{i-1})$
  - // $O(n)$ $G_i = G_{i-1}$.**contract**(e)
- $C_1 :=$ **KargerStein**(G_s) // inde-
- // pendent
- $C_2 :=$ **KargerStein**(G_s) // runs
- **return** smaller of C_1, C_2

Karger-Stein: Success Probability

Know: Each call to Karger-Stein breaks the min-cut with probability at most $\frac{1}{2}$.

↑ before calling itself recursively

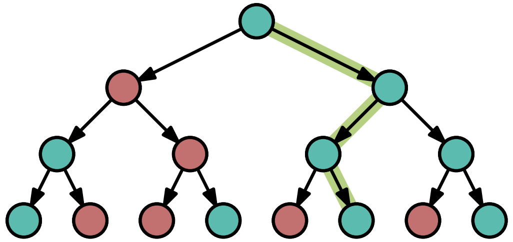
Karger-Stein: Success Probability

Know: Each call to Karger-Stein breaks the min-cut with probability at most $\frac{1}{2}$.

before calling itself recursively

Auxiliary Problem

Given complete binary tree of height d where each node is randomly coloured red or green (with probability $\frac{1}{2}$ each).
What is the probability p_d that a green root-to-leaf path exists?



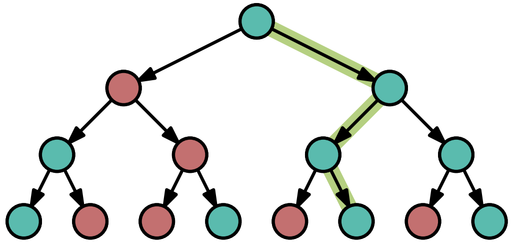
Karger-Stein: Success Probability

Know: Each call to Karger-Stein breaks the min-cut with probability at most $\frac{1}{2}$.

↑ before calling itself recursively

Auxiliary Problem

Given complete binary tree of height d where each node is randomly coloured red or green (with probability $\frac{1}{2}$ each).
What is the probability p_d that a green root-to-leaf path exists?



$p_0 = 1/2$ // root green $\qquad p_d = \frac{1}{2}(1 - (1 - p_{d-1})^2)$ // root green, **not** no path in both left and right subtree

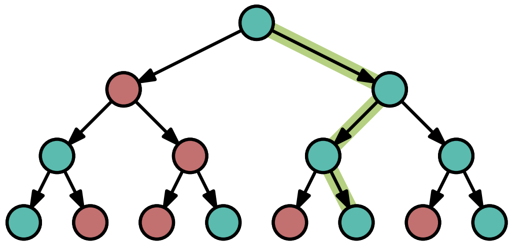
Karger-Stein: Success Probability

Know: Each call to Karger-Stein breaks the min-cut with probability at most $\frac{1}{2}$.

before calling itself recursively

Auxiliary Problem

Given complete binary tree of height d where each node is randomly coloured red or green (with probability $\frac{1}{2}$ each).
What is the probability p_d that a green root-to-leaf path exists?



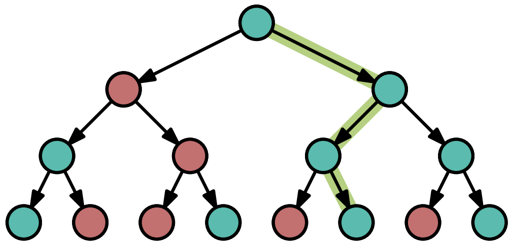
$p_0 = 1/2$ // root green $\quad p_d = \frac{1}{2}(1 - (1 - p_{d-1})^2)$ // root green, **not** no path in both left and right subtree

Claim: $p_d \geq \frac{1}{d+2}$. Proof by induction.

Karger-Stein: Success Probability

Know: Each call to Karger-Stein breaks the min-cut with probability at most $\frac{1}{2}$.
↳ before calling itself recursively

Auxiliary Problem
Given complete binary tree of height d where each node is randomly coloured red or green (with probability $\frac{1}{2}$ each).
What is the probability p_d that a green root-to-leaf path exists?



$p_0 = 1/2$ // root green $\quad p_d = \frac{1}{2}\left(1 - (1 - p_{d-1})^2\right)$ // root green, **not** no path in both left and right subtree

Claim: $p_d \geq \frac{1}{d+2}$. Proof by induction.

$$p_0 = \frac{1}{2} = \frac{1}{0+2} \quad \checkmark$$

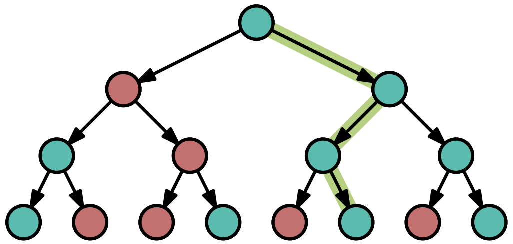
$$p_d = \frac{1}{2}\left(1 - (1 - p_{d-1})^2\right) \geq \frac{1}{2}\left(1 - (1 - \tfrac{1}{d+1})^2\right) = \frac{1}{2}\left(\frac{2}{d+1} - \frac{1}{(d+1)^2}\right)$$

$$= \frac{1}{2} \cdot \frac{2d+2-1}{(d+1)^2} = \frac{1}{2} \cdot \frac{2d+1}{d^2+2d+1} \geq \frac{1}{2} \cdot \frac{2d}{d^2+2d} = \frac{1}{d+2}$$ // for $1 \leq a \leq b$ we have $\frac{a}{b} \geq \frac{a-1}{b-1}$

Karger-Stein: Success Probability

Know: Each call to Karger-Stein breaks the min-cut with probability at most $\frac{1}{2}$.
↳ before calling itself recursively

Auxiliary Problem
Given complete binary tree of height d where each node is randomly coloured red or green (with probability $\frac{1}{2}$ each).
What is the probability p_d that a green root-to-leaf path exists?



$p_0 = 1/2$ // root green $\quad p_d = \frac{1}{2}(1-(1-p_{d-1})^2)$ // root green, **not** no path in both left and right subtree

Claim: $p_d \geq \frac{1}{d+2}$. Proof by induction.

$$p_0 = \frac{1}{2} = \frac{1}{0+2} \quad ✓$$

$$p_d = \frac{1}{2}(1-(1-p_{d-1})^2) \geq \frac{1}{2}\left(1-\left(1-\frac{1}{d+1}\right)^2\right) = \frac{1}{2}\left(\frac{2}{d+1} - \frac{1}{(d+1)^2}\right)$$

$$= \frac{1}{2}\cdot\frac{2d+2-1}{(d+1)^2} = \frac{1}{2}\cdot\frac{2d+1}{d^2+2d+1} \geq \frac{1}{2}\cdot\frac{2d}{d^2+2d} = \frac{1}{d+2}$$ // for $1 \leq a \leq b$ we have $\frac{a}{b} \geq \frac{a-1}{b-1}$

Corollary: Karger-Stein succeeds with probability at least $p_{\log_{\sqrt{2}}(n)} = \frac{1}{O(\log n)}$.

Karger-Stein Amplified

Theorem: On a graph with n nodes, Karger-Stein runs in $O(n^2 \log(n))$ time and returns a minimum cut with probability at least $1/O(\log(n))$.

Karger-Stein Amplified

Theorem: On a graph with n nodes, Karger-Stein runs in $O(n^2 \log(n))$ time and returns a minimum cut with probability at least $1/O(\log(n))$.

Amplification

Success probability $\geq p$

Number of repetitions t

Amplified prob. $\geq 1 - e^{-pt}$

Karger-Stein Amplified

Theorem: On a graph with n nodes, Karger-Stein runs in $O(n^2 \log(n))$ time and returns a minimum cut with probability at least $1/O(\log(n))$.

Amplification

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp\left(-\frac{t}{O(\log(n))}\right) = 1 - O\left(\frac{1}{n}\right)$$

for $t = \log^2(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Karger-Stein Amplified

Theorem: On a graph with n nodes, Karger-Stein runs in $O(n^2 \log(n))$ time and returns a minimum cut with probability at least $1/O(\log(n))$.

Amplification

$$\Pr[\text{“min-cut found”}] \geq 1 - \exp\left(-\frac{t}{O(\log(n))}\right) = 1 - O\left(\frac{1}{n}\right)$$

for $t = \log^2(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

Corollary: On a graph with n nodes, $O(\log^2(n))$ repetitions of Karger-Stein run in $O(n^2 \log^3(n))$ total time and return a minimum cut with high probability.

Karger-Stein Amplified

Karlsruhe Institute of Technology

Theorem: On a graph with n nodes, Karger-Stein runs in $O(n^2 \log(n))$ time and returns a minimum cut with probability at least $1/O(\log(n))$.

Amplification

$$\Pr[\text{"min-cut found"}] \geq 1 - \exp\left(-\frac{t}{O(\log(n))}\right) = 1 - O\left(\frac{1}{n}\right)$$

for $t = \log^2(n)$

Success probability $\geq p$
Number of repetitions t
Amplified prob. $\geq 1 - e^{-pt}$

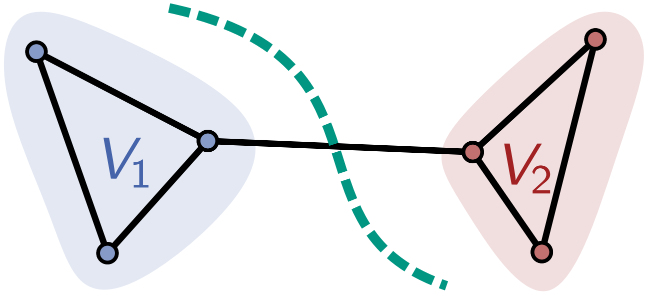
Corollary: On a graph with n nodes, $O(\log^2(n))$ repetitions of Karger-Stein run in $O(n^2 \log^3(n))$ total time and return a minimum cut with high probability.

- Compared to $O(n^4 \log(n))$ for Karger
- Compared to $\Omega(n^3)$ for deterministic approaches

Conclusion

Minimum Cut

- Fundamental graph problem
- Many deterministic flow-based algorithms ...
- . . . with worst-case running times in $\Omega(n^3)$



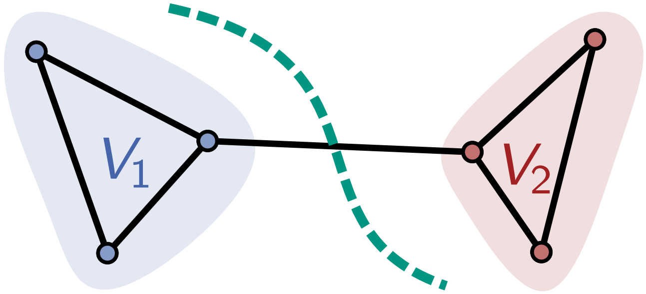
Conclusion

Minimum Cut

- Fundamental graph problem
- Many deterministic flow-based algorithms ...
- . . . with worst-case running times in $\Omega(n^3)$

Randomized Algorithms

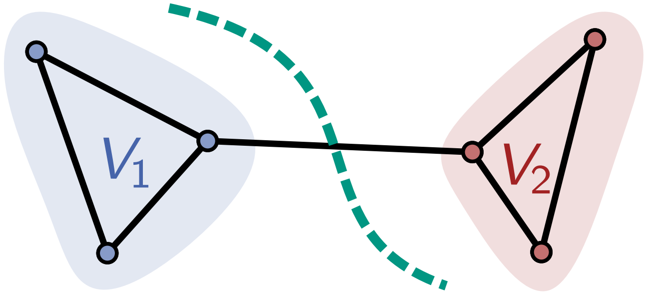
- Karger's edge-contraction algorithm



Conclusion

Minimum Cut

- Fundamental graph problem
- Many deterministic flow-based algorithms ...
- ... with worst-case running times in $\Omega(n^3)$



Randomized Algorithms

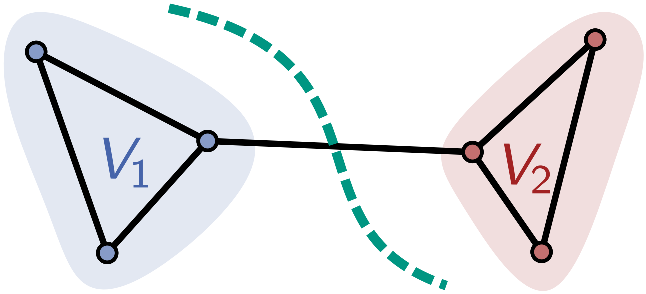
- Karger's edge-contraction algorithm

Probability Amplification

- Monte Carlo algorithms with and without biases
- Repetitions amplify success probability
- Karger-Stein: Amplify before failure probability gets large

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Conclusion



Minimum Cut

- Fundamental graph problem
- Many deterministic flow-based algorithms ...
- ... with worst-case running times in $\Omega(n^3)$

Randomized Algorithms

- Karger's edge-contraction algorithm

Probability Amplification

- Monte Carlo algorithms with and without biases
- Repetitions amplify success probability
- Karger-Stein: Amplify before failure probability gets large

Algo Output \ Correct Answer	✗	✓
✗	true neg	false neg
✓	false pos	true pos

Outlook

"Minimum cuts in near-linear time", Karger, J.Acm. '00

Success w.h.p. in time $O(m \log^3(n))$

"Faster algorithms for edge connectivity via random 2-out contractions", Ghaffari & Nowicki & Thorup, SODA'20

Success w.h.p. in time $O(m \log(n))$ and $O(m + n \log^3(n))$

Mögliche Prüfungsfragen

- Was ist ein Monte-Carlo-Algorithmus?
  - Welche Varianten gibt es?
- Was versteht man unter Probability Amplification?
- Wie funktioniert Probability Amplification...
  - ... bei einseitigem Fehler?
  - ... bei zweiseitigem Fehler?
  - ... bei Optimierungsproblemen?
  - Wie hängt die Fehlerwahrscheinlichkeit mit der Anzahl Wiederholungen zusammen?
- Was ist das Minimum Cut Problem?
  - Was leisten die besten bekannten deterministischen Algorithmen?
  - Was sind Erfolgswahrscheinlichkeit und Laufzeit des trivialen Random Cut Algorithmus?
  - Wie funktioniert der Algorithmus von Karger?
    - Was bedeutet $\Pr[\mathcal{E}_t]$ und wie haben wir diese Wahrscheinlichkeit abgeschätzt?
    - Was ergibt sich für die Laufzeit und die Erfolgswahrscheinlichkeit?
  - Wie ergibt sich der Algorithmus von Karger und Stein aus dem Algorithmus von Karger?
    - Wie haben wir die Erfolgswahrscheinlichkeit und Laufzeit abgeschätzt?
    - Wie erreiche ich eine Erfolgswahrscheinlichkeit von $1 - \frac{1}{n}$?