

# **Probability & Computing**

### **Probability Amplification**





**22 Maximilian Katzmann, Stefan Walzer – Probability a Computing Institute of Theoretical Information Maximilian Katzmann, Stefan Walzer – Probability a computing**  $\frac{1}{2}$ **<br>
<b>2** Maximilian Katzmann, Stefan Walzer – Probab **Definition**: A **Monte Carlo Algorithm** is a randomized algorithm with bounded running time that, for each input, answers correctly with probability at least  $p \in (0,1)$ .



**22 Maximilian Katzmann, Stefan Walzer – Probability actomputing and Theoretical Information Algorithms institute of Theoretical Information Computing a Direct Amplification problems**  $p$  **is the probability of giving the c 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<br> **One-sided error**: either *false-biased* or *true-biased*<br>  $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$ 





**22** Maximilian Katzmann, Stefan Walzer – Probability at omputing  $\frac{1}{2}$  Maximilian Katzmann, Stefan Walzer – Probability of Theoretical Information Algorithms **Probability** of Theoretical Information **Probability Pr 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 **Correct Answer <b>Correct** Answer **Correct** Answer **Correct Correct Correct Correct Correct Correct Correct Correct Correct Correct** 

 $\begin{array}{c|c|c|c|c|c} \hline \text{X} & \text{as } \text{X} \text{ is a positive number of vertices.} \ \hline \text{X} & \text{as } \text{X} \text{ is a positive number of vertices.} \ \hline \text{X} & \text{as } \text{X} \text{ is a positive number of vertices.} \ \hline \text{X} & \text{as } \text{X} \text{ is a positive number of vertices.} \ \hline \text{X} & \text{as } \text{X} \text{ is a positive number of vertices.} \ \hline \text{X} & \text{S} & \text{S} \ \hline \text{X} & \text{S} & \text{S} \ \hline \text{$ ✓ answers may be incorrect





**22** Maximilian Katzmann, Stefan Walzer – Probability accomputing and Theoretical Information Algorithms **Probability of Theoretical Information Computing amplification**<br> **2 Direction Probability of Theoretical Inform 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<br> **One-sided error**: either *false-biased* or *true-biased*<br> *X* answers are always correct<br> *X* answers may be incorrect<br> *X* answers may be incorre

version of the correct version of the correct version of the set of the version of the **x** answers may be incorrect





**22** Maximilian Katzmann, Stefan Walzer – Probability at computing a computing Institute of Theoretical Information Computing **ample of Theoretical Information Computing a Computing a computing a computing institute of 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<br> **Correct Answer <b>Two-sided error**: *ino bias* Algo Output Correct Answer and the correct Answer and the section of the correct answer and the secti
	- **One-sided error**: either *false-biased* or *true-biased*
	-

**x** answers may be incorrect ✓ answers may be incorrect





**22** Maximilian Katzmann, Stefan Walzer – Probability accomputing  $\frac{1}{2}$  Maximilian Katzmann, Stefan Walzer – Probability of Theoretical Information Computing **Probability of Theoretical Information** Probability **Prope 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<br> **One-sided error**: either *false-biased* or *true-biased*<br>
In optimization problems *p* is the probability of finding the optimum<br>  $\frac{a}{2} \times \frac{a}{2$ 
	- **One-sided error**: either *false-biased* or *true-biased*
	- **Two-sided error**: *no bias*
- 



false pos

true X<br>true fals<br>neg neg

✗

✓

true pos

false neg

**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

**22** Maximilian Katzmann, Stefan Walzer – Probability at computing a computing inter that is the probability of giving the correct answer<br> **2 Computing Information Probability of Theoretical Information Probability of T Definition**: **Probability amplification** is the process of increasing the success probability **I** In decision problems *p* is the probability of giving the correct answer<br> **One-sided error**: either *false-biased* or *true-biased*<br> **I No-sided error**: *no bias*<br> **I** In optimization problems *p* is the probability



**22** Maximilian Katzmann, Stefan Walzer – Probability at computing a computing inter that is the probability of giving the correct answer<br> **2 Computing Information Probability of Theoretical Information Probability of T 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



**Definition**: **Probability amplification** is the process of increasing the success probability **I** In decision problems *p* is the probability of giving the correct answer<br> **One-sided error**: either *false-biased* or *true-biased*<br> **I No-sided error**: *no bias*<br> **I** In optimization problems *p* is the probability



**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 √ answers are always corre<br>★ answers may be incorrect

Correct Answer<br>  $\frac{x}{\frac{1}{2}}$ <br>  $\frac{x}{\frac{1}{2}}$ <br>  $\frac{1}{2}$ <br>  $\frac{x}{\frac{1}{2}}$ <br>  $\frac{1}{2}$ <br>  $\frac{1}{2$ false pos true pos false neg true  $\begin{array}{c|c}\n \hline\n \text{true}-\text{DIASEO} & \text{true} \\
 \hline\n \text{X} & \text{answer} & \text{are always correct} \\
 \hline\n \text{X} & \text{answer} & \text{may be incorrect} \\
 \hline\n \text{X} & \text{answer} & \text{may be incorrect} \\
 \hline\n \end{array}$ ✗ ✓ ✗ ✓

**Probability Amplification**<br> **22 Maximilian Katzmann Stefan Walzer – Probability of Theoretical Information Computing Institute of Theoretical Information Computing <b>and Computing Information** Probability of Theoretical I **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  $X$ . Pr["correct"]  $\geq 1-(1-p)^t$   $\geq 1-e^{-pt}$

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



Correct Answer<br>  $\begin{array}{ccc}\nX & \swarrow \\
\frac{1}{2} & \swarrow \\
\frac{1}{2} & \searrow \\
\frac{1}{2} & \searrow \\
\frac{1}{2} & \searrow\n\end{array}$  true<br>
true<br>
true<br>
false true<br>
pos pos

true X<br>true fals<br>neg neg

✗

✓

false pos

true pos

false neg

**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

**22** Maximilian Katzmann, Stefan Walzer-Probability at Computing Algorithm is a randomized algorithm with bounded running<br> **20 Maximilian Katzmann** Path Batzmann Path Batzmann Path Batzmann Path Batzmann Path Batzmann Pat **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  $X$ . Pr["correct"]  $\geq 1-(1-p)^t$   $\geq 1-e^{-pt}$

Exercise: For two-sided error.

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



**22** Maximilian Katzmann, Stefan Walzer – Probability accomputing accomputing Theoretical Information Computing interesting of Theoretical Information Computing **Probability** of The probability of thrue-biased<br> **2 Compu 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

false pos true pos false neg true X<br>true fals<br>neg neg ✗ ✓

**Definition**: **Probability amplification** is the process of increasing the success probability **I** In decision problems *p* is the probability of giving the correct answer<br> **One-sided error**: either *false-biased* or *true-biased*<br> **I No-sided error**: *no bias*<br> **I** In optimization problems *p* is the probability

**Probability Amplification for optimization algorithms**<br>■ Execute independently *t* times.<br>■ output best result

- -

$$
\mathsf{Pr}[\text{``optimal''}] \ge 1 - \left(1 - p\right)^t \ \ge 1 - e^{-pt}
$$

- The Segmentation Problem<br>
Input<br>
Set P of points in a feature space (e.g.,  $\mathbb{R}^d$ )<br>
Similarity measure  $\sigma: P \times P \to \mathbb{R}_+$ <br>  $\blacksquare$ <br>
Similarity according to the problem of the set of the set of the set of the set of the *d* )
	- Similarity measure  $\sigma$ :  $P \times P \mapsto \mathbb{R}_+$



- *d* )
- Similarity measure  $\sigma$ :  $P \times P \mapsto \mathbb{R}_+$



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

- *d* )
- **Similarity measure**  $\sigma$ **:**  $P \times P \mapsto \mathbb{R}_+$
- **Output**:  $P_1, \ldots, P_k$  such that
- Points within a *P<sup>i</sup>* have high similarity
- Points in distinct *P<sup>i</sup>* , *P<sup>j</sup>*



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

- *d* )
- **Similarity measure**  $\sigma$ **:**  $P \times P \mapsto \mathbb{R}_+$
- **Output**:  $P_1, \ldots, P_k$  such that
- Points within a *P<sup>i</sup>* have high similarity
- Points in distinct *P<sup>i</sup>* , *P<sup>j</sup>*



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

**• Similarity measure**  $\sigma$ :  $P \times P \mapsto \mathbb{R}_+$ 

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

- Points within a *P<sup>i</sup>* have high similarity
- Points in distinct *P<sup>i</sup>* , *P<sup>j</sup>*



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

**• Similarity measure**  $\sigma$ :  $P \times P \mapsto \mathbb{R}_+$ 

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

- Points within a *P<sup>i</sup>* have high similarity
- Points in distinct *P<sup>i</sup>* , *P<sup>j</sup>*

- 
- **Approach**: Model as graph **Each point is a node**<br> **Each point is a node**<br> **Edges between all node pairs, with the weight given by esegment into two sets**



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

**• Similarity measure**  $\sigma$ :  $P \times P \mapsto \mathbb{R}_+$ 

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

- Points within a *P<sup>i</sup>* have high similarity
- Points in distinct *P<sup>i</sup>* , *P<sup>j</sup>*

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



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



**• Similarity measure**  $\sigma$ :  $P \times P \mapsto \mathbb{R}_+$ 

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

- Points within a *P<sup>i</sup>* have high similarity
- Points in distinct *P<sup>i</sup>* , *P<sup>j</sup>*

- 
- **Approach**: Model as graph **Each point is a node**<br> **Each point is a node**<br> **Edges between all node pairs, with the weight given by esegment into two sets** the similarity of the two nodes **Applications**: Compression, medical diagnosis, etc. <br> **Approach**: Model as graph **a** *k* **c** is the inversed Euclidear distance<br>
■ Edges between all node pairs, with the weight given by ■ segment into two sets<br>
the simi
- Find *cut-set* (edges to remove) of minimal weight such that the graph decomposes into *k* components.



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

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



- 
- *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*: 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*: 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*: 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  $V_2$





- 
- *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  $V_2$
- **In Weight of a cut: size of the cut-set** (or sum of weights in a weighted 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  $V_2$
- **In 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*

*V*1

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





**55 Maximilian Katzmann, Stefan Walzer – Probability & Computing Institute of Theoretical Informatics, Algorithm Engineering & Scalable Algorithms<br>
<b>5** Maximilian Katzmann, Stefan Walzer – Probability & Computing<br> **5** Maxi  $n-1$  − 1 cuts in a graph with *n* nodes.



■ Number of possible assignments of *n* nodes to 2 parts<sup>1</sup>



*n*



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

■ Number of possible assignments of *n* nodes to 2 parts<sup>1</sup>



*n*



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

Number of possible assignments of *n* nodes to 2 parts Partitions with empty parts that do not represent cuts 2  $2^n - 2$ <br>Partitions with empty parts that do not represent cuts —



 $2^n - 2$ 



Number of possible assignments of *n* nodes to 2 parts Partitions with empty parts that do not represent cuts 2  $2^n - 2$ <br>Partitions with empty parts that do not represent cuts —



 $2^n - 2$ 



**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<sup>n</sup> −2)<br>
Number of possible assignments of *n* nodes to 2 parts<br>
Partitions with empty parts that do not represent cuts<br>
Swapping parts does not yield a new partition –
- 





- Number of possible assignments of *n* nodes to 2 parts
- (2<sup>n</sup> −2)<br>
Number of possible assignments of *n* nodes to 2 parts<br>
Partitions with empty parts that do not represent cuts<br>
Swapping parts does not yield a new partition –
- 





- Number of possible assignments of *n* nodes to 2 parts
- (2<sup>n</sup> −2)<br>
Number of possible assignments of *n* nodes to 2 parts<br>
Partitions with empty parts that do not represent cuts<br>
Swapping parts does not yield a new partition –
- **Algorithm: Random Cut**

**Return a uniformly random cut.** 





- Number of possible assignments of *n* nodes to 2 parts
- (2<sup>n</sup> −2)<br>
Number of possible assignments of *n* nodes to 2 parts<br>
Partitions with empty parts that do not represent cuts<br>
Swapping parts does not yield a new partition –
- **Algorithm: Random Cut**
- Return a uniformly random cut.
- **Minor challenge: How to uniformly sample cuts?**





- Number of possible assignments of *n* nodes to 2 parts
- (2<sup>n</sup> −2)<br>
Number of possible assignments of *n* nodes to 2 parts<br>
Partitions with empty parts that do not represent cuts<br>
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**




 $n-1$  − 1 cuts in a graph with *n* nodes.

- Number of possible assignments of *n* nodes to 2 parts
- (2<sup>n</sup> −2)<br>
Number of possible assignments of *n* nodes to 2 parts<br>
Partitions with empty parts that do not represent cuts<br>
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?



 $(2^n - 2)/2$ 



 $n-1$  − 1 cuts in a graph with *n* nodes.

- Number of possible assignments of *n* nodes to 2 parts
- (2<sup>n</sup> −2)<br>
Number of possible assignments of *n* nodes to 2 parts<br>
Partitions with empty parts that do not represent cuts<br>
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  $U(\{0,1\}^n)$  and use rejection sampling



 $(2^n - 2)/2$ 



 $n-1$  − 1 cuts in a graph with *n* nodes.

- Number of possible assignments of *n* nodes to 2 parts
- (2<sup>n</sup> −2)<br>
Number of possible assignments of *n* nodes to 2 parts<br>
Partitions with empty parts that do not represent cuts<br>
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  $U(\{0,1\}^n)$  and use rejection sampling
		- **actually for bounded running time: declare failure rather than sampling again**
		- samples each cut with probability 1/2<sup>n-1</sup>



 $(2^n - 2)/2$ 



# Running time:  $O(n)$  much better than the  $\Omega(n^3)$  in the deterministic setting , but...<br>
Running time:  $O(n)$  much better than the  $\Omega(n^3)$  in the deterministic setting , but...<br>
Shadimilan Katanam. Sietin Waze – Probabill  $3)$  in the deterministic setting, but...



**6** Maximilian Katzmann, Stefan Walzer – Probability:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut: Analysis  $\frac{1}{2}$ <br> **6** Maximilian Katzmann, Stefan Walzer – Probability a Computing  $\frac{1}{2}$  method of Theoretical I **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut. much better than the  $\Omega(n^3)$  in the deterministic setting, but...



**6** Maximilian Katzmann, Stefan Walzer – Probability  $\geq 1/2^{n-1}$  and  $\equiv$  if there is only one min-cut:<br>
→ exponentially small!<br>  $\rightarrow$  exponentially small!<br>  $\rightarrow$  exponentially small!<br> **Engineering Accomputing Accomputi Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut. much better than the  $\Omega(n^3)$  in the deterministic setting, but...

 $\rightarrow$  exponentially small!



**Running time:**  $O(n)$  much better than the  $\Omega(n^3)$  in the deterministic setting , but...<br>
Success probability:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut.<br>
→ exponentially small!<br> **Amplification**<br>
■ Repeat the algo **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut. much better than the  $\Omega(n^3)$  in the deterministic setting, but...

 $\rightarrow$  exponentially small!

**• Repeat the algorithm to obtain t independent random cuts, return the smallest**  $\Pr[\text{``min cut found''}] \geq 1 - \left(1 - 1/2^{n-1}\right)^t$ 



**Running time:**  $O(n)$  much better than the  $\Omega(n^3)$  in the deterministic setting , but...<br> **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut.<br>  $\rightarrow$  exponentially small!<br> **Amplification**<br> **Computer** Re **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut. much better than the Ω(*n*<br>  $\rightarrow$  1/2<sup>*n*-1</sup> "=" if there is o<br>  $\rightarrow$  exponentially small!<br>
to obtain *t* independent r<br>
"]  $\geq$  1 - (1 - 1/2<sup>*n*-1</sup>)<sup>*t*</sup><br>
minimum  $3)$  in the deterministic setting, but...

 $\rightarrow$  exponentially small!

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

$$
\mathsf{Pr}[\text{``min cut found''}] \ge 1 - \big(1 - \underbrace{1/2^{n-1}}_{\text{minimum}}\big)^t
$$



**Running time:**  $O(n)$  much better than the  $\Omega(n^3)$  in the deterministic setting , but...<br> **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut.<br>  $\rightarrow$  exponentially small!<br> **Amplification**<br> **Computer** Re **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut. much better than the Ω(*n*<br>  $\rightarrow$  1/2<sup>*n*-1</sup> "=" if there is o<br>  $\rightarrow$  exponentially small!<br>
to obtain *t* independent r<br>
"]  $\geq$  1 - (1 - 1/2<sup>*n*-1)<sup>*t*</sup><br>
not minimum</sup>  $3)$  in the deterministic setting, but...

 $\rightarrow$  exponentially small!

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

$$
\mathsf{Pr}[\text{``min cut found''}] \geq \underbrace{1 - (1 - 1/2^{n-1})}_{\text{not}}{}^t
$$



**Running time:**  $O(n)$  much better than the  $\Omega(n^3)$  in the deterministic setting , but...<br> **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut.<br>  $\rightarrow$  exponentially small!<br> **Amplification**<br> **Computer** Re **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut. much better than the Ω(*n*<sup>3</sup>)<br>  $\geq 1/2^{n-1}$  "=" if there is onl<br>
→ exponentially small!<br>
to obtain *t* independent rar<br>
"]  $\geq 1 - (1 - 1/2^{n-1})^t$ <br>
not minimum *t* times  $3)$  in the deterministic setting, but...

 $\rightarrow$  exponentially small!

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

$$
\Pr[\text{``min cut found''}] \ge \underbrace{1 - \left(1 - \underbrace{1/2^{n-1}}_{\text{not}}\right)^t}_\text{not}\biguparrow^t_\text{time}
$$



**Running time:**  $O(n)$  much better than the  $\Omega(n^3)$  in the deterministic setting , but...<br>
Success probability:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut.<br>
→ exponentially small!<br> **Amplification**<br>
■ **Repeat the alg Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut. much better than the  $\Omega(n^3)$  in the deterministic setting, but...

 $\rightarrow$  exponentially small!

**• Repeat the algorithm to obtain t independent random cuts, return the smallest**  $\Pr[\text{``min cut found''}] \ge 1 - \left(1 - 1/2^{n-1}\right)^t \ge 1 - e^{-t/2^{n-1}}$  $1 + x \le e^x$  for  $x \in \mathbb{R}$ 



**Running time:**  $O(n)$  much better than the  $\Omega(n^3)$  in the deterministic setting, but...<br> **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut.<br>  $\rightarrow$  exponentially small!<br> **Amplification**<br> **Computer** Rep **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut. : *O*(*n*) much better than the Ω(*n*<sup>3</sup>) in the determinis<br> **ability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut.<br>
→ exponentially small!<br>
Ilgorithm to obtain *t* independent random cuts, return<br>
cut found"]  $\geq$  $3)$  in the deterministic setting, but...

 $\rightarrow$  exponentially small!

**• Repeat the algorithm to obtain t independent random cuts, return the smallest**  $\Pr[\text{``min cut found''}] \ge 1 - \left(1 - 1/2^{n-1}\right)^t \ge 1 - e^{-t/2^{n-1}}$  $1 + x \le e^x$  for  $x \in \mathbb{R}$ 

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



**Running time:**  $O(n)$  much better than the  $\Omega(n^3)$  in the deterministic setting , but...<br>
Success probability:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut.<br>
→ exponentially small!<br> **Amplification**<br>
■ Repeat the algo **Success probability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut. **:**  $O(n)$  much better than the Ω( $n^3$ ) in the determ<br> **ability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cu<br>
→ exponentially small!<br>
algorithm to obtain *t* independent random cuts, re<br>
cut found"]  $\geq 1 - (1 - 1/2^{n$  $3)$  in the deterministic setting, but...

 $\rightarrow$  exponentially small!

**• Repeat the algorithm to obtain t independent random cuts, return the smallest**  $\Pr[\text{``min cut found''}] \ge 1 - \left(1 - 1/2^{n-1}\right)^t \ge 1 - e^{-t/2^{n-1}}$  $1 + x \le e^x$  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  $1 - 1/n$ 



**Running time:**  $O(n)$  much better than the  $\Omega(n^3)$  in the deterministic setting , but...<br>
Success probability:  $\geq 1/2^{n-1}$  "=" if there is only one min-cut.<br>
→ exponentially small!<br> **Amplification**<br>
■ Repeat the algo **Success probability**: ≥  $1/2^{n-1}$ **:**  $O(n)$  much better than the Ω( $n^3$ ) in the determ<br> **ability**:  $\geq 1/2^{n-1}$  "=" if there is only one min-cu<br>
→ exponentially small!<br>
algorithm to obtain *t* independent random cuts, re<br>
cut found"]  $\geq 1 - (1 - 1/2^{n$  $3)$  in the deterministic setting, but...

**• Repeat the algorithm to obtain t independent random cuts, return the smallest**  $\Pr[\text{``min cut found''}] \ge 1 - \left(1 - 1/2^{n-1}\right)^t \ge 1 - e^{-t/2^{n-1}}$  $\geq 1/2^{n-1}$  "=" if there is only one min-cut.<br>
→ exponentially small!<br>
to obtain t independent random cuts, return the smallest<br>  $\binom{3^n}{2} \geq 1 - \left(1 - 1/2^{n-1}\right)^t \geq 1 - e^{-t/2^{n-1}}$ <br>  $\frac{1 + x \leq e^x$  for  $\frac{1}{2}$ <br>
cut fo

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  $1 - 1/n$ 

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

this is terrible





**Karger's Algorithm**<br> **Edge Contraction**<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• Merge two adjacent nodes in a multigraph without se







# **Karger's Algorithm**<br> **Edge Contraction**<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• Merge two adjacent nodes in a multigraph without se





# Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br>
• A (multi) graph with two nodes has a unique cut-set<br>
• A the contraction of the con ■ A (multi) graph with two nodes has a unique cut-set





A (multi) graph with two nodes has a unique cut-set

# **Contraction Algorithm**

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br>
• Contraction Algorithm<br>
• Movivalion: clistinguish *Toon-essen* Motivation: distinguish *non-essential* as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones





- 
- A (multi) graph with two nodes has a unique cut-set

# **Contraction Algorithm**

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br>
• Contraction Algorithm<br>
• Monte Margingthen *Tonnessential*<br>
• **• Motivation: distinguish** *non-essential* as well as *essential* edges dge Contraction<br>
Merge two adjacent nodes in a multigraph without self-loops<br>
A (multi) graph with two nodes has a unique cut-set<br>
ontraction Algorithm<br>
Motivation: distinguish *non-essential*<br>
as well as *essential* edges





- 
- A (multi) graph with two nodes has a unique cut-set

# **Contraction Algorithm**

**Motivation: distinguish** *non-essential* as well as *essential* edges dge Contraction<br>
Merge two adjacent nodes in a multigraph without self-loops<br>
A (multi) graph with two nodes has a unique cut-set<br>
ontraction Algorithm<br>
Motivation: distinguish *non-essential*<br>
as well as *essential* edges

```
Karger's Algorithm<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br>
Contraction Algorithm<br>
• Motivation: distinguish fron-essentia
   Karger(G_0 = (V_0, E_0))for i = 1 to n - 2 do
          e \coloneqq \mathcal{U}(E_{i-1})G_i = G_{i-1}. contract(e)
      return unique cut-set in Gn−2
```




■ A (multi) graph with two nodes has a unique cut-set

# **Contraction Algorithm**

**Karger's Algorithm**<br> **Edge Contraction**<br> **Edge Contraction**<br> **Edge Contraction**<br> **Edge two adjacent nodes in a multigraph without self-loops**<br> **Example that Algorithm**<br> **Example in the multiply of the contraction Algorit** Motivation: distinguish *non-essential* as well as *essential* edges  $\frac{1}{2}$  part of a min-cut & hope there are few essential ones **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

**Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\qquad$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*)

**return** unique cut-set in *Gn*−<sup>2</sup>

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

*w x* non-essential essential





- 
- A (multi) graph with two nodes has a unique cut-set

# **Contraction Algorithm • Motivation: distinguish** *non-essential*

as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

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

for  $i = 1$  to  $n - 2$  do  $\qquad$  //  $O(n)$  $e \coloneqq \mathcal{U}(E_{i-1})$  $\frac{1}{0}$   $(0, 1)$ 

*G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*)

**return** unique cut-set in *Gn*−<sup>2</sup>

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

is a cut-set in  $\mathit{G}_{0}.$ 

Let *C* be a cut-set in *G<sup>i</sup>* .

**Success Probability**

- $G_i \setminus C$  is disconnected
- Assume *C* is not a cut-set in *G*<sub>0</sub>.
	- $G_0 \setminus C$  is connected.
- $G_i \setminus C$  arises from  $G_0 \setminus C$  by *i* edge contractions.
- **E** *f* contractions cannot disconnect a graph





- 
- A (multi) graph with two nodes has a unique cut-set

// *O*(*n*)

# **Contraction Algorithm**

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br>
Contraction Algorithm<br>
• Motivation: clistinguish *inner* essen **• Motivation: distinguish** *non-essential* as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

$$
Karger(G_0 = (V_0, E_0))
$$
  
**for**  $i = 1$  to  $n - 2$  **do**

$$
e := \mathcal{U}(E_{i-1}) \qquad \qquad \textcolor{blue}{\mathcal{U}(0(1))}
$$

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

**return** unique cut-set in *Gn*−<sup>2</sup>

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

**Success Probability**

essential

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

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





- 
- A (multi) graph with two nodes has a unique cut-set

# **Contraction Algorithm**

**• Motivation: distinguish** *non-essential* as well as *essential* edges **ontraction Algorithm** not part of a min-cut<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges part of a min-cut<br>& hope there are few essential ones

# **Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\qquad$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$

*G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*)

**return** unique cut-set in *Gn*−<sup>2</sup>

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

**Success Probability**

essential

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

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br>
Contraction Algorithm<br>
• Motivation: clistinguish *inner* essen **n** Consider min-cut in  $G_0$  with cut-set C and  $|C| = k$  $\mathcal{E}_i$  = "C in  $G_i$ "  $Pr[\mathcal{E}_1]=1-\frac{k}{n}$ *m*





- 
- A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm • Motivation: distinguish** *non-essential* as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

for  $i = 1$  to  $n - 2$  do  $\qquad$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*) **return** unique cut-set in *Gn*−<sup>2</sup>

Running time in  $O(n^2)$ 

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

■ Can be implemented to run in  $O(m)$ 

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br> **Contraction Algorithm**<br>
• Motivation: clistinguish *inner* exs *w x* **Success Probability**  $C$ onsider min-cut in  $G_0$  with cut-set  $C$  and  $|C|$  $\mathcal{E}_i$  = "C in  $G_i$ "  $Pr[\mathcal{E}_1]=1-\frac{k}{n}$ **Observation**: A cut-set in *G<sub>i</sub>* is a cut-set in *G*<sub>0</sub>.  $\Big\{$ **Observation**: min-degree >  $k$ essential





**1** Consider **m**  
**n** 
$$
\mathcal{E}_i
$$
 = "C in G

*m*

■ A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm • Motivation: distinguish** *non-essential* **ontraction Algorithm** not part of a min-cut<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges part of a min-cut<br>& hope there are few essential ones

as well as *essential* edges

**Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\qquad$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*)

**return** unique cut-set in *Gn*−<sup>2</sup>

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





- 
- A (multi) graph with two nodes has a unique cut-set
- **Contraction Algorithm** Motivation: distinguish *non-essential* as well as *essential* edges  $\frac{1}{2}$  part of a min-cut & hope there are few essential ones **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

**Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\qquad$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*) **return** unique cut-set in *Gn*−<sup>2</sup>

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

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br> **Contraction Algorithm**<br>
• Motivation: clistinguish *inner* exp *w x* **Success Probability n** Consider min-cut in  $G_0$  with cut-set C and  $|C| = k$  $\dot{m} =$ 1 2  $\sum$ *v*∈*V* deg(*v*) ≥ 1 2  $\sum$ *v*∈*V*  $k \geq$ 1 2 *nk*  $\mathcal{E}_i$  = "C in  $G_i$ "  $Pr[\mathcal{E}_1]=1-\frac{k}{n}$ *m*  $\underline{\mathsf{U}_0.}$ **Observation**: min-degree >  $k$ (holds for all *G<sup>i</sup>* on-essential<br>
ique cut-set<br> **Success Probability**<br> **Observation**: A cut-set in *G*<sub>*i*</sub> is a cut-set in *G*<sub>0</sub><br> **e**<br> **Observation**: A cut-set *C* and  $|C|$ <br> **e**<br> **E**<sub>*i*</sub> = "*C* in *G*<sub>*i*</sub>" **Observation**: min-degree ≥ is a cut-set in  $\mathit{G}_{0}.$ 



# **Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br> **Contraction Algorithm**<br>
• Motivation: distinguish *inner* exec **Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\qquad$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*) **return** unique cut-set in *Gn*−<sup>2</sup> Running time in  $O(n^2)$ **• Motivation: distinguish** *non-essential* as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones ■ Can be implemented to run in  $O(m)$  $\dot{m} =$ 1 2  $\mathcal{E}_i$  = "C in  $G_i$ "  $Pr[\mathcal{E}_1]=1-\frac{k}{n}$ *m*  $\geq 1 - \frac{k}{nk}$ *nk=*2  $= 1 - \frac{2}{n}$ *n*

- 
- A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm**





- 
- A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm • Motivation: distinguish** *non-essential* 

as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

**Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\blacksquare$  //  $O(n)$ 

 $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*)

**return** unique cut-set in *Gn*−<sup>2</sup>

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



2

*n*

(holds for all *G<sup>i</sup>*



**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br> **Contraction Algorithm**<br>
• Motivation: clistinguish *inno-essen* for  $i = 1$  to  $n - 2$  do  $\blacksquare$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*) **return** unique cut-set in *Gn*−<sup>2</sup> *w x* Running time in  $O(n^2)$ as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones **Success Probability n** Consider min-cut in  $G_0$  with cut-set C and  $|C| = k$ none of the *k* edges of *C* contracted  $\mathcal{E}_i = \text{``C in } G_i$ " **Observation**: min-degree ≥ *k*<br>  $[\mathcal{E}_1] \ge 1 - \frac{2}{n}$  (holds for all *G<sub>i</sub>* due to 1st observation<br>  $[\mathcal{E}_2 | \mathcal{E}_1] \ge 1 - \frac{2}{n-1}$ <br>
→ none of the *k* edges of *C* contracted<br>
— do not contract *k* (holds for all *G<sup>i</sup>*  $\begin{CD} \mathsf{non}\text{-essential}\\ \mathsf{w}\text{-}\mathsf{essential} \ \overbrace{\mathsf{G}_i \text{ is a cut-set in } G_0.} \ \mathsf{th}\text{ cut-set } C \text{ and } |C| = \ \mathsf{tion}\text{: min-degree} \geq k \ \mathsf{is}\text{ for all } G_i \text{ due to 1st observation)} \end{CD}$  $Pr[\mathcal{E}_2 | \mathcal{E}_1] \geq 1 - \frac{2}{n-1}$ *n*−1  $Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$ *n* **Observation**: A cut-set in  $G_i$  is a cut-set in  $G_0$ .

■ A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm**

**• Motivation: distinguish** *non-essential* 

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

- 
- Can be implemented to run in  $O(m)$



- 
- A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm** Motivation: distinguish *non-essential* as well as *essential* edges  $\frac{1}{2}$  part of a min-cut & hope there are few essential ones **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

**Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\blacksquare$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ 

*G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*)

**return** unique cut-set in *Gn*−<sup>2</sup>

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

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br> **Contraction Algorithm**<br>
• Motivation: clistinguish *inner* exs *w x* **Success Probability Consider min-cut in**  $G_0$  **with cut-set** *C* **and**  $|C| = k$  $\mathcal{E}_i = "C$  in  $G_i" \quad |$  **Observation**: min-degree  $\geq k$  $\frac{2}{n}$  (holds for all  $G_i$  $\begin{CD} \mathsf{non}\text{-essential}\\ \mathsf{w}\text{-}\mathsf{essential} \ \overbrace{\mathsf{G}_i \text{ is a cut-set in } G_0.} \ \mathsf{th}\text{ cut-set } C \text{ and } |C| = \ \mathsf{tion}\text{: min-degree} \geq k \ \mathsf{is}\text{ for all } G_i \text{ due to 1st observation)} \end{CD}$  $Pr[\mathcal{E}_2 | \mathcal{E}_1] \geq 1 - \frac{2}{n-1}$  $\frac{2}{n-1}$  → Pr $[\mathcal{E}_i \mid \mathcal{E}_1 \cap ... \cap \mathcal{E}_{i-1}] \ge 1 - \frac{2}{n-i}$  $\begin{array}{l} \Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n} \qquad \qquad \hbox{(holds for all $G_i$ due to 1st observation)} \[5pt] \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} \end{array}$ *n* **Observation**: A cut-set in  $G_i$  is a cut-set in  $G_0$ .



- 
- A (multi) graph with two nodes has a unique cut-set
- **Contraction Algorithm • Motivation: distinguish** *non-essential* as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

**Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\blacksquare$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*) **return** unique cut-set in *Gn*−<sup>2</sup> Running time in  $O(n^2)$ 

■ Can be implemented to run in  $O(m)$ 

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br> **Contraction Algorithm**<br>
• Motivation: clistinguish *inner* exs *w x* **Success Probability Consider min-cut in**  $G_0$  **with cut-set** *C* **and**  $|C| = k$  $\mathcal{E}_i = "C$  in  $G_i" \quad |$  **Observation**: min-degree  $\geq k$  $Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$  (holds for all  $G_i$  $\begin{CD} \mathsf{non}\text{-essential}\\ \mathsf{w}\text{-}\mathsf{essential} \ \overbrace{\mathsf{G}_i \text{ is a cut-set in } G_0.} \ \mathsf{th}\text{ cut-set } C \text{ and } |C| = \ \mathsf{tion}\text{: min-degree} \geq k \ \mathsf{is}\text{ for all } G_i \text{ due to 1st observation)} \end{CD}$  $Pr[\mathcal{E}_2 | \mathcal{E}_1] \geq 1 - \frac{2}{n-1}$  $\frac{2}{n-1}$  → Pr $[\mathcal{E}_i \mid \mathcal{E}_1 \cap ... \cap \mathcal{E}_{i-1}] \ge 1 - \frac{2}{n-i}$ *n*−*i*+1  $Pr[\mathcal{E}_{n-2}] = Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdot \ldots \cdot Pr[\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$ *n* **Observation:** A cut-set in *G<sub>i</sub>* is a<br> **Consider min-cut in** *G***<sub>0</sub> with cut-<br>
<b>P**<sub>r</sub> $[\mathcal{E}_1] \ge 1 - \frac{2}{n}$  (holds for all *G*)<br>
P<sub>r</sub> $[\mathcal{E}_2 | \mathcal{E}_1] \ge 1 - \frac{2}{n-1} \rightarrow Pr[\mathcal{E}_i | \mathcal{E}_1 \cap ... \cap$ <br>
Pr $[\mathcal{E}_{n-2}] = Pr[\mathcal{E}_1] \cdot Pr[\mathcal{$ **Observation**: A cut-set in  $G_i$  is a cut-set in  $G_0$ .



*n*−2

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br> **Contraction Algorithm**<br>
• Motivation: distinguish *inner* expe **Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\blacksquare$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*) **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones **Success Probability Consider min-cut in**  $G_0$  **with cut-set** *C* **and**  $|C| = k$  $\mathcal{E}_i = "C$  in  $G_i" \quad |$  **Observation**: min-degree  $\geq k$  $Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$  (holds for all  $G_i$  $\begin{CD} \mathsf{non}\text{-essential}\\ \mathsf{w}\text{-}\mathsf{essential} \ \overbrace{\mathsf{G}_i \text{ is a cut-set in } G_0.} \ \mathsf{th}\text{ cut-set } C \text{ and } |C| = \ \mathsf{tion}\text{: min-degree} \geq k \ \mathsf{is}\text{ for all } G_i \text{ due to 1st observation)} \end{CD}$  $Pr[\mathcal{E}_2 | \mathcal{E}_1] \geq 1 - \frac{2}{n-1}$  $\frac{2}{n-1}$  → Pr $[\mathcal{E}_i \mid \mathcal{E}_1 \cap ... \cap \mathcal{E}_{i-1}] \ge 1 - \frac{2}{n-i}$  $Pr[\mathcal{E}_{n-2}] = Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdot \ldots \cdot Pr[\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$ *n*  $\geq (1 - \frac{2}{n})$ 2 2 2  $\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{4}\right)\left(1-\frac{2}{3}\right)$ **Observation**: A cut-set in  $G_i$  is a cut-set in  $G_0$ .

*n*

*n*−1

■ A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm • Motivation: distinguish** *non-essential* 

as well as *essential* edges

**return** unique cut-set in *Gn*−<sup>2</sup>

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

*w x*



*n*−*i*+1

2 3

4

*w x*

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br>
• Contraction Algorithm<br>
• Motivation: clistinguish *inner* exs **Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\blacksquare$  //  $O(n)$ **Success Probability Consider min-cut in**  $G_0$  **with cut-set** *C* **and**  $|C| = k$  $\mathcal{E}_i = "C$  in  $G_i" \quad |$  **Observation**: min-degree  $\geq k$  $Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$  (holds for all  $G_i$  $\begin{CD} \mathsf{non}\text{-essential}\\ \mathsf{w}\text{-}\mathsf{essential} \ \overbrace{\mathsf{G}_i \text{ is a cut-set in } G_0.} \ \mathsf{th}\text{ cut-set } C \text{ and } |C| = \ \mathsf{tion}\text{: min-degree} \geq k \ \mathsf{is}\text{ for all } G_i \text{ due to 1st observation)} \end{CD}$  $Pr[\mathcal{E}_2 | \mathcal{E}_1] \geq 1 - \frac{2}{n-1}$  $\frac{2}{n-1}$  → Pr $[\mathcal{E}_i \mid \mathcal{E}_1 \cap ... \cap \mathcal{E}_{i-1}] \ge 1 - \frac{2}{n-i}$ *n*−*i*+1  $Pr[\mathcal{E}_{n-2}] = Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdot \ldots \cdot Pr[\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$ *n*  $1 = \frac{n - i}{n - i}$  $\geq (1 - \frac{2}{n})$ *n* 2 *n*−1 2 *n*−2 2 4 2 3  $\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{4}\right)\left(1-\frac{2}{3}\right)$ **Observation**: A cut-set in  $G_i$  is a cut-set in  $G_0$ .

- 
- A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm • Motivation: distinguish** *non-essential* as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

# $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*) **return** unique cut-set in *Gn*−<sup>2</sup>

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

*n*−2

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br> **Contraction Algorithm**<br>
• Motivation: distinguish *inner* expe **Karger** $(G_0 = (V_0, E_0))$ for  $i = 1$  to  $n - 2$  do  $\blacksquare$  //  $O(n)$ **Success Probability Consider min-cut in**  $G_0$  **with cut-set** *C* **and**  $|C| = k$  $\mathcal{E}_i = "C$  in  $G_i" \quad |$  **Observation**: min-degree  $\geq k$  $Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$  (holds for all  $G_i$  $\begin{CD} \mathsf{non}\text{-essential}\\ \mathsf{w}\text{-}\mathsf{essential} \ \overbrace{\mathsf{G}_i \text{ is a cut-set in } G_0.} \ \mathsf{th}\text{ cut-set } C \text{ and } |C| = \ \mathsf{tion}\text{: min-degree} \geq k \ \mathsf{is}\text{ for all } G_i \text{ due to 1st observation)} \end{CD}$  $Pr[\mathcal{E}_2 | \mathcal{E}_1] \geq 1 - \frac{2}{n-1}$  $\frac{2}{n-1}$  → Pr $[\mathcal{E}_i \mid \mathcal{E}_1 \cap ... \cap \mathcal{E}_{i-1}] \ge 1 - \frac{2}{n-i}$  $Pr[\mathcal{E}_{n-2}] = Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdot \ldots \cdot Pr[\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$ *n* ≥ *n* 2 2 2 2 *n*−1 *n*−2 4  $\left(\frac{n}{n}-\frac{2}{n}\right)\left(\frac{n-1}{n-1}-\frac{2}{n-1}\right)\left(\frac{n-2}{n-2}-\frac{2}{n-2}\right)\cdots\left(\frac{4}{n}-\frac{2}{n}\right)\left(\frac{3}{3}-\frac{2}{3}\right)$ **Observation**: A cut-set in  $G_i$  is a cut-set in  $G_0$ .

*n*

*n*

*n*−1

*n*−1

*n*−2

- 
- A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm • Motivation: distinguish** *non-essential* as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

 $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*) **return** unique cut-set in *Gn*−<sup>2</sup>

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

*w x*





*n*−*i*+1

3 3 2 3

4

4
*n | n*−1 *| n−2 | (* 4 *)* ( 3

**Karger's Algorithm**<br>
Edge Contraction<br>
• Merge two adjacent nodes in a multigraph without self-loops<br>
• A (multi) graph with two nodes has a unique cut-set<br> **Contraction Algorithm**<br>
• Motivation: distinguish *inner* exse **Karger** $(G_0 = (V_0, E_0))$ **Success Probability Consider min-cut in**  $G_0$  **with cut-set** *C* **and**  $|C| = k$  $\mathcal{E}_i = "C$  in  $G_i" \quad |$  **Observation**: min-degree  $\geq k$  $Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$  (holds for all  $G_i$  $\begin{CD} \mathsf{non}\text{-essential}\\ \mathsf{w}\text{-}\mathsf{essential} \ \overbrace{\mathsf{G}_i \text{ is a cut-set in } G_0.} \ \mathsf{th}\text{ cut-set } C \text{ and } |C| = \ \mathsf{tion}\text{: min-degree} \geq k \ \mathsf{is}\text{ for all } G_i \text{ due to 1st observation)} \end{CD}$  $Pr[\mathcal{E}_2 | \mathcal{E}_1] \geq 1 - \frac{2}{n-1}$  $\frac{2}{n-1}$  → Pr $[\mathcal{E}_i \mid \mathcal{E}_1 \cap ... \cap \mathcal{E}_{i-1}] \ge 1 - \frac{2}{n-i}$  $Pr[\mathcal{E}_{n-2}] = Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdot \ldots \cdot Pr[\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$ *n*  $\geq \left(\frac{n-2}{n}\right)\left(\frac{n-1-2}{n-1}\right)\left(\frac{n-2-2}{n-2}\right)\cdots\left(\frac{4-2}{n}\right)\left(\frac{3-2}{2}\right)$ 

- 
- A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm • Motivation: distinguish** *non-essential* as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

for  $i = 1$  to  $n - 2$  do  $\qquad$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*) **return** unique cut-set in *Gn*−<sup>2</sup>

Running time in  $O(n^2)$ ■ Can be implemented to run in  $O(m)$  *w x*

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

$$
\mathcal{L}^{\mathcal{L}}(\mathcal{L}^{\mathcal{L}}(\mathcal{L}^{\mathcal{L}}))
$$

*n*−*i*+1

**Karger's Algorithm**<br>
Edge Contraction<br> **Edge Contraction**<br>
Merge two adjacent nodes in a multigraph without self-loops<br> **A** (multi) graph with two nodes has a unique cut-set<br>
Contraction Algorithm<br>
Mean of a finical summ **Karger** $(G_0 = (V_0, E_0))$ **Success Probability Consider min-cut in**  $G_0$  **with cut-set** *C* **and**  $|C| = k$  $\mathcal{E}_i = "C$  in  $G_i" \quad |$  **Observation**: min-degree  $\geq k$  $Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$  (holds for all  $G_i$  $Pr[\mathcal{E}_2 | \mathcal{E}_1] \geq 1 - \frac{2}{n-1}$  $\frac{2}{n-1}$  → Pr $[\mathcal{E}_i \mid \mathcal{E}_1 \cap ... \cap \mathcal{E}_{i-1}] \ge 1 - \frac{2}{n-i}$  $Pr[\mathcal{E}_{n-2}] = Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdot \ldots \cdot Pr[\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-3}]$ *n*  $\geq \left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{2}{n-4}\right)\left(\frac{1}{n-2}\right)$ *n | n*−1 *| n−2 | (* 4 *)* ( 3 **Observation**: A cut-set in  $G_i$  is a cut-set in  $G_0$ .

■ A (multi) graph with two nodes has a unique cut-set

**Contraction Algorithm • Motivation: distinguish** *non-essential* as well as *essential* edges **ontraction Algorithm**<br>Motivation: distinguish *non-essentia*<br>as well as *essential* edges} part of a min-cut<br>& hope there are few essential ones

for  $i = 1$  to  $n - 2$  do  $\qquad$  //  $O(n)$  $e := U(E_{i-1})$  //  $O(1)$ *G<sup>i</sup>* = *Gi*−1*:***contract**(*e*) // *O*(*n*) **return** unique cut-set in *Gn*−<sup>2</sup>

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



*n* <sup>1</sup> *n*−1 <sup>1</sup> *n*−2 <sup>2</sup> 1 4 <sup>1</sup> 3

Karger's Algorithm			
Edge Contraction	more essential		
Merge two adjacent nodes in a multigraph without self-loops	$uv$		
A (multi) graph with two nodes has a unique cut-set	$uv$		
Contraction Algorithm	and part of a mirror	$success$ Probability	$essential$
Motivation: distinguish $non-essential$	$Quccess$ Probability	$essential$	
as well as essential edges} $real$ and edges	$l$	$Consolution: A cut-set in G_i is a cut-set in G_0.$	
8 hope there are few essential ones	$Consider min-cut$ in $G_0$ with cut-set $C$ and $ C  = k$		
$Karger(G_0 = (V_0, E_0))$	$l$ $O(n)$	$P_r[\xi_1] \geq 1 - \frac{2}{n}$	$(holds for all G_i due to 1st observation)$
$e := U(E_{i-1})$	$l$ $O(1)$	$Pr[\xi_2   \xi_1] \geq 1 - \frac{2}{n-1}$	$Pr[\xi_1   \xi_1 \cap ... \cap \xi_{i-1}] \geq 1 - \frac{2}{n-i+1}$
$G_i = G_{i-1}$ .contract(e) $l$ $O(n)$	$Pr[\xi_{n-2}] = Pr[\xi_1] \cdot Pr[\xi_2   \xi_1] \cdot ... \cdot Pr[\xi_{n-2}   \xi_1 \cap ... \cap \xi_{n-3}]$		
return unique cut-set in $G_{n-2}$	$\geq \frac{(p-2)}{n}$ ( $\frac{n-3}{n-1}$ ) ( $\frac{n-4}{n-2}$		

■ A (multi) graph with two nodes has a unique cut-set

**Contraction n** Motivatic as well as *essential* edges

 $$  $e \coloneqq \mathcal{U}$ (  $G_i = G_i$ **return** u

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



 $\overline{n}$   $\left|\left(\begin{array}{cc} n-1 \end{array}\right)$   $\left|\left(\begin{array}{cc} n-2 \end{array}\right) \right|$ 

Karger's Algorithm			
Edge Contraction	more essential		
Merge two adjacent nodes in a multigraph without self-loops	$uv$		
A (multi) graph with two nodes has a unique cut-set	$uv$		
Contraction Algorithm	and part of a mirror et		
Notivation: distinguish $non-essential$	Success Probability	essential	
As well as essential edges} $real$ and edges	Constant and one		
the same few essential ones	Consider min-cut in $G_0$ with cut-set $C$ and $ C  = k$		
Range( $G_0 = (V_0, E_0)$ )	$U(0, E_0)$		
for $i = 1$ to $n - 2$ do	$  O(n)  $	$Pr[\mathcal{E}_1] \ge 1 - \frac{2}{n}$	$  O(\log \text{for all } G_i \text{ due to 1st observation})$
for $i = 1$ to $n - 2$ do	$  O(1)$	$Pr[\mathcal{E}_1] \ge 1 - \frac{2}{n}$	$  O(\log \text{for all } G_i \text{ due to 1st observation})$
for $i = 1$ to $n - 2$ do	$  O(1)$	$Pr[\mathcal{E}_1] \ge 1 - \frac{2}{n}$	$  O(\log \text{for all } G_i \text{ due to 1st observation})$
for $i = 1$ to $n - 2$ do	$  O(1)$	$Pr[\mathcal{E}_2   \mathcal{E}_1] \ge 1 - \frac{2}{n-1}$	$Pr[\mathcal{E}_1   \mathcal{E}_1   \cdots  $

 $\blacksquare$  A (multi)

**Contraction n** Motivation as well as *essential* edges

 $e$  :=  $\mathcal{U}$ (  $G_i = G_i$ **return** u

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

 $\mid$   $\cal{A}$   $\mid$   $\cal{X}$ 

Karger's Algorithm			
Edge Contraction	more essential		
Merge two adjacent nodes in a multigraph without self-loops	$uv$		
A (multi) graph with two nodes has a unique cut-set	$uv$		
Contraction Algorithm	and part of a mirror	$success Probability$	$ssential$
Motivation: distinguish $non-essential$	$Quccess Probability$	$ssential$	
as well as essential edges} $real$ and $real$ and $real$	$Conservation: A cut-set in G_i is a cut-set in G_0.$		
8 hope there are few essential ones	$Consider min-cut in G_0 with cut-set C and  C  = k$		
$Karger(G_0 = (V_0, E_0))$	$l$ $O(n)$	$Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n}$	$(holds for all G_i due to 1st observation)$
$e := U(E_{i-1})$	$l$ $O(1)$	$Pr[\mathcal{E}_1] \geq 1 - \frac{2}{n-1}$	$Pr[\mathcal{E}_1   \mathcal{E}_1   \dots   P[\mathcal{E}_1   \mathcal{E}_1   \dots   P[\mathcal{E}_{n-2}]   \mathcal{E}_1   \dots   P[\mathcal{E}_{n-2}] = Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2   \mathcal{E}_1   \dots   P[\mathcal{E}_{n-2}]   \mathcal{E}_1   \dots   \mathcal{E}_{n-2}   \mathcal{E}_1   \dots   \mathcal{E}_{n-2}   \mathcal{E}_1   \dots   \mathcal{E}_{n-2}   \mathcal{E}_1   \dots   \mathcal{E}_{n-2}   \mathcal{E}_1   \dots   \mathcal{E}_{n-$

A (multi) graph with two nodes has a unique cut-set

**Contraction n** Motivation

as well as *essential* edges

 $$  $e \coloneqq \mathcal{U}$  $G_i = G_i$ **return u** 

- **n** Running
- **n** Can be

*w x*



**8 8** Maximilian Katzmann, Stefan Walzer – Probability at least  $\frac{2}{n^2}$ .<br> **8** Maximilian Katzmann, Stefan Walzer – Probability at least  $\frac{2}{n^2}$ . **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  $\frac{2}{n^2}$ .



**Example 18 Algorithm Amplified**<br> **8** Maximum cut with probability at least  $\frac{2}{n^2}$ .<br> **8** Maximum cut with probability at least  $\frac{2}{n^2}$ .<br> **8** Success probability  $\geq p$ <br> **8** Maximum of repetitions *t*<br> **8** Amplifi **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.  $\geq 1 - e^{-pt}$ 



**Example 18 Algorithm Amplified**<br> **18 Maximum** cut with probability at least  $\frac{2}{n^2}$ .<br>  $\Pr[\text{``min-cut found''}] \ge 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$ <br>  $\begin{array}{|l|l|} \hline \text{Success probability} > p \\ \hline \text{for } t = \frac{n^2}{2} \ln(n) \end{array}$  Amplified prob.  $\ge 1 - e^{-pt}$ <br> **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  $\frac{2}{n^2}$ .

$$
\Pr[\text{"min-cut found"}] \ge 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}
$$
\n
$$
\left[\begin{array}{c}\text{Success probability} \ge \rho \\ \text{Number of repetitions } t \\ \text{for } t = \frac{n^2}{2} \ln(n) \end{array}\right]
$$
\n
$$
\text{Number of repetitions } t
$$
\n
$$
\text{Amplified prob.} \ge 1 - e^{-pt}
$$

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

Karger's Algorithm Amplified	
Theorem: On a graph with <i>n</i> nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$ .	
$Pr["min-cut found"] \geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	$\frac{Success probability \geq p}{Number of repetitions t}$
<b>Corollary:</b> On a graph with <i>n</i> 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.	
$Waximinian Katzman, Stefan Walzer-Probability & Computing$	Insitute of Theoretical Informatics, Algorithm Engineering & Scalable Algorithm

**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	
Theorem: On a graph with <i>n</i> nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$ .	
$Pr[$ "min-cut found"] $\geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	$\frac{Success probability \geq p}{Number of repetitions t}$
<b>Corollary:</b> On a graph with <i>n</i> 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.	$\frac{Much better than exp. time of Randomized Cut!}{Much better than exp. time of Randomized Cut!}$

\n**Maximum Ragian Katzmann, Stefan Walzer – Probability & Computing**

\n**Maximum Ragineering & Scalable Algorithm**

**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	
Theorem: On a graph with <i>n</i> nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$ .	
$Pr["min-cut found"] \geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	Success probability $\geq p$
$Pr["min-cut found"] \geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	Success probability $\geq p$
$Corollary: On a graph with n nodes, O(n^2 \log(n)) Karger repetitions run in O(n^4 \log(n))$	
$total time and return a min-cut with high probability.$	Much better than exp. time of Randomized Cut!
$Let C_1, ..., C_\ell$ be all the min-cuts in <i>G</i> and $\mathcal{E}_{n-2}^i$ for $i \in [\ell]$ be the event that $C_i$ is returned by Karger's algorithm	
$log Karger's algorithm$	Just little of Theoretical Informatics, Algorithm Engineering a Scalable Algorithm

**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.

### **Sidenote: Number of minimum cuts**

Let  $C_1, \ldots, C_\ell$  be all the min-cuts in *G* and  $\mathcal{E}_r^i$  $\sum_{n=2}^{\infty}$  for  $i \in [\ell]$  be the event that  $C_i$  is returned biand the and return a mini-cut with high probability. Much better than exp. time of Randomized Cut!<br> **idenote: Number of minimum cuts**<br>
Let  $C_1, \ldots, C_\ell$  be all the min-cuts in G and  $\mathcal{E}'_{n-2}$  for  $i \in [\ell]$  be the eve



Karger's Algorithm Amplified	
Theorem: On a graph with <i>n</i> nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$ .	
$Pr[$ "min-cut found"] $\geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	Success probability $\geq p$
$Pr[$ "min-cut found"] $\geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	Success probability $\geq p$
$Corollary: On a graph with n nodes, O(n^2 \log(n)) Karger repetitions run in O(n^4 \log(n))$	
$total time and return a min-cut with high probability.$	Much better than exp. time of Randomized Cut!
$Let C_1, \ldots, C_{\ell}$ be all the min-cuts in <i>G</i> and $\mathcal{E}_{n-2}^{\prime}$ for $i \in [\ell]$ be the event that $C_i$ is returned by Karger's algorithm	
$Just seen: Pr[\mathcal{E}_{n-2}^{\prime}] \geq \frac{2}{n^2}$	
Assimilar Katzman, Stefan Walzer – Probability & Computing	

**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  $\mathcal{E}_r^i$  $\sum_{n=2}^{\infty}$  for  $i \in [\ell]$  be the event that  $C_i$  is returned by Karger's algorithm

### Just seen: Pr[E *i*  $\binom{n}{n-2} \geq \frac{2}{n^2}$  $\overline{n^2}$



Karger's Algorithm Amplified	
Theorem: On a graph with <i>n</i> nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$ .	
$Pr[$ "min-cut found"] $\geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	$\frac{Success probability \geq p}{Number of repetitions t}$
$Corollary: On a graph with n nodes, O(n^2 \log(n)) Karger repetitions run in O(n^4 \log(n))$	
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 + e^{-pt}}$
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 + e^{-pt}}$
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 + e^{-2}}$ for $i \in [\ell]$ be the event that $C_i$ is returned by Karger's algorithm
<b>Just seen:</b> $Pr[\mathcal{E}_{n-2}^{\ell}] \geq \frac{2}{n^2}$	
<b>Matrix</b>	$\frac{1}{1 + e^{-2}}$

**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.

# **Sidenote: Number of minimum cuts**

- Let  $C_1, \ldots, C_\ell$  be all the min-cuts in G and  $\mathcal{E}_r^i$  $\sum_{n=2}^{i}$  for  $i \in [\ell]$  be the event that  $C_i$  is returned by Karger's algorithm  $\mathcal{E}_{n-2}^i$  for  $i \in [\ell]$  be the event that  $C_i$  is return<br>disjoint, since the algorithm returns only one cut
- Just seen: Pr[E *i*  $\binom{n}{n-2} \geq \frac{2}{n^2}$  $\overline{n^2}$



Karger's Algorithm Amplified	
Theorem: On a graph with <i>n</i> nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$ .	
$Pr[$ "min-cut found"] $\geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	$\frac{Success probability \geq p}{Number of repetitions t}$
$Corollary: On a graph with n nodes, O(n^2 \log(n)) Karger repetitions run in O(n^4 \log(n))$	
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 - e^{-pt}}$
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 - e^{-pt}}$
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 - e^{-pt}}$
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 - e^{-pt}}$
<b>Set</b> $C_1, \ldots, C_{\ell}$ be all the min-cuts in $G$ and $\frac{C_{n-2}^{\ell}}{C_{n-2}^{\ell}}$ for $i \in [\ell]$ be the event that $C_i$ is returned by Karger's algorithm	
<b>Just seen:</b> $\Pr[\hat{C}_{n-2}^{\ell}] \geq \frac{2}{n^2}$	
<b>Pr</b> $\left[ \bigcup_{i \in [\ell]} \mathcal{E}_{n-2}^{\ell} \right] = \sum_{i \in [\ell]} \Pr[\hat{\mathcal{E}}_{n-2}^{\ell}] \geq \frac{2\ell}{n^2}$	
<b>Maximilian Katzman, Stefan Water– Probability &amp; Computing</b>	$\frac{1}{1 - e^{-pt}}$

**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  $\mathcal{E}_r^i$  $\sum_{n=2}^{i}$  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[E *i*  $\binom{n}{n-2} \geq \frac{2}{n^2}$  $\overline{n^2}$  $\Pr\left[\bigcup_{i\in[\ell]}\mathcal{E}_r^i\right]$ *n*−2  $\Big] = \sum_{i \in [\ell]} \mathsf{Pr}[\mathcal{E}_r^i]$  $\binom{n}{n-2} \geq \frac{2 \cdot \ell}{n^2}$ *n*2



Karger's Algorithm Amplified	
Theorem: On a graph with <i>n</i> nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$ .	
$Pr[$ "min-cut found"] $\geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	$\frac{Success probability \geq p}{Number of repetitions t}$
$Corollary: On a graph with n nodes, O(n^2 \log(n)) Karger repetitions run in O(n^4 \log(n))$	
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 - e^{-pt}}$
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 - e^{-pt}}$
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 - e^{-pt}}$
<b>Sidence: Number of minimum cuts</b>	$\frac{1}{1 - e^{-pt}}$
<b>Solve: Number of minimum cuts</b>	$\frac{1}{1 - e^{-pt}}$
<b>Set</b> $C_1, \ldots, C_{\ell}$ be all the min-cuts in $G$ and $\frac{1}{C_{n-2}}$ for $i \in [\ell]$ be the event that $C_i$ is returned by Karger's algorithm	
<b>Just seen:</b> $\Pr[\hat{E}_{n-2}^i] \geq \frac{2}{n^2}$	
<b>1</b> $\geq \Pr[\bigcup_{i \in [\ell]} \mathcal{E}_{n-2}^i] = \sum_{i \in [\ell]} \Pr[\hat{E}_{n-2}^i] \geq \frac{2\frac{2\ell}{n^2}}{2\frac{2\frac{2\ell}{n^2}}$	
<b>Maximum State</b> Markim Ragenering a Scale Map orithms	

**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  $\mathcal{E}_r^i$  $\sum_{n=2}^{i}$  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[E *i*  $\binom{n}{n-2} \geq \frac{2}{n^2}$  $\overline{n^2}$  $\Pr\left[\bigcup_{i\in[\ell]}\mathcal{E}_r^i\right]$ *n*−2  $\Big] = \sum_{i \in [\ell]} \mathsf{Pr}[\mathcal{E}_r^i]$  $\left|\bigcup_{i\in[\ell]}\mathcal{E}_{n-2}^i\right|=\sum_{i\in[\ell]} \mathsf{Pr}[\mathcal{E}_{n-2}^i]\geq \frac{2\cdot \ell}{n^2}$

Karger's Algorithm Amplified	
Theorem: On a graph with <i>n</i> nodes, Karger's algorithm runs in $O(n^2)$ time and returns a minimum cut with probability at least $\frac{2}{n^2}$ .	
$Pr[$ "min-cut found"] $\geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	Success probability $\geq p$
$Pr[$ "min-cut found"] $\geq 1 - \exp(-\frac{2}{n^2} \cdot t) = 1 - \frac{1}{n}$	Success probability $\geq p$
$Corollary: On a graph with n nodes, O(n^2 \log(n)) Karger repetitions run in O(n^4 \log(n))$	
$total time$ and return a min-cut with high probability. Model: Number of minimum cuts	Let $C_1, \ldots, C_\ell$ be all the min-cuts in $G$ and $\mathcal{E}_{n-2}^i$ for $i \in [\ell]$ be the event that $C_i$ is returned by Karger's algorithm
$Just seen: Pr[\mathcal{E}_{n-2}^i] \geq \frac{2}{n^2}$	Observation: $\ell \leq \frac{n^2}{2}$
$1 \geq Pr \left[ U_{i \in [\ell]} \mathcal{E}_{n-2}^i \right] = \sum_{i \in [\ell]} Pr[\mathcal{E}_{n-2}^i] \geq \frac{2\ell}{n^2}$	Observation: $\ell \leq \frac{n^2}{2}$
$u_{\text{aximinian Katzman, Stefan Wardzer - Probability's Computing}}$	Insert value of Theoretical Inform ratios, Algorithm Engineering a Scalable Algorithm

**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  $\mathcal{E}_r^i$  $\sum_{n=2}^{i}$  for  $i \in [\ell]$  be the event that  $C_i$  is returned by Karger's algorithm disjoint, since the algorithm returns only one cut

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

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

 $\frac{2}{2}$ .

Motivation<br>
Motivation<br>
Probability that a min-cut survives *i* contractions<br>  $Pr[\mathcal{E}_1] - Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot Pr[\mathcal{E}_l \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{l-1}]$ <br>  $\geq \Big(1-\frac{2}{n}\Big)\Big(1-\frac{2}{n-1}\Big)\Big(1-\frac{2}{n-2}\Big) \cdot \Big(1-\frac{2}{n-l+$ 

$$
\geq \Big(1-\tfrac{2}{n}\Big)\Big(1-\tfrac{2}{n-1}\Big)\Big(1-\tfrac{2}{n-2}\Big)\cdot\cdot\Big(1-\tfrac{2}{n-i+2}\Big)\Big(1-\tfrac{2}{n-i+1}\Big)
$$

Motivation<br>
Motivation<br>
Probability that a min-cut survives *i* contractions<br>  $Pr[\mathcal{E}_1] - Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot Pr[\mathcal{E}_l \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{l-1}]$ <br>  $\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdot \left(1 - \frac{2$  $\geq (1 - \frac{2}{n})$ *n* 2 *n*−1 2 *n*−2  $\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n-i+2}\right)\left(1-\frac{2}{n-i+1}\right)$  $=\left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{n-i+2-2}{n-i+2}\right)$ *n*  $\left(\frac{1}{n-1}\right)\left(n-2\right)$   $\left(n-i+2\right)$  $(n-i+1-2)$ *n*−*i*+1



Motivation<br>
Motivation<br>
Probability that a min-cut survives *i* contractions<br>  $Pr[\mathcal{E}_1] - Pr[\mathcal{E}_1] \cdot Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \ldots \cdot Pr[\mathcal{E}_l \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{l-1}]$ <br>  $\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdot \left(1 - \frac{2$  $\geq (1 - \frac{2}{n})$ *n* 2 *n*−1 2 *n*−2  $\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n-i+2}\right)\left(1-\frac{2}{n-i+1}\right)$  $=\left(\frac{n-2}{n}\right)\left(\frac{n-3}{n-1}\right)\left(\frac{n-4}{n-2}\right)\cdots\left(\frac{n-i}{n-i+2}\right)$ *n*  $\left(\frac{1}{n-1}\right)\left(n-2\right)$   $\left(n-i+2\right)$  $(n-i-1)$ *n*−*i*+1 *n*−*i* <u>) ( *n*−*i*−1</u>





More Amplification: Karger-Stein<br>
Motivation<br>
Pr[ $\mathcal{E}_i$ ]<br>
Pr[ $\mathcal{E}_i$ ] Pr[ $\mathcal{E}_j$ ] Pr[ $\mathcal{E}_j$  Pri $\mathcal{E}_j$  Prince is contractions<br>
Pr[ $\mathcal{E}_i$ ] Pr[ $\mathcal{E}_j$ ] Pr[ $\mathcal{E}_j$  Pri $\mathcal{E}_j$  Prince is contractions<br>  $\mathcal$  $\geq (1 - \frac{2}{n})$ *n* 2 *n*−1 2 *n*−2  $\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n-i+2}\right)\left(1-\frac{2}{n-i+1}\right)$  $=\left(\frac{p-2}{p-3}\right)\left(\frac{p-3}{p-2}\right)\left(\frac{p-4}{p-3}\right)\cdots\left(\frac{p-1}{p-1}\right)$ *n n*−1 *n*−2 *n*−*i*+2  $(n-i-1)$ *n*−*i*+1 *n*−*i* <u>) ( *n*−*i*−1</u> =  $(n-i)(n-i-1)$ *n*(*n* − 1) ≥  $(n-i-1)(n-i-1)$ *n* · *n*  $= (1$ *i* + 1 *n*  $\big)^2$ .





More Amplification: Karger-Stein<br>
Motivation<br>
Pr[ $\mathcal{E}_i$ ]<br>
Pr[ $\mathcal{E}_i$ ] Pr[ $\mathcal{E}_j$ ] Pr[ $\mathcal{E}_j$  Pri $\mathcal{E}_j$  Prince is contractions<br>
Pr[ $\mathcal{E}_i$ ] Pri $\mathcal{E}_j$ ] Pri $\mathcal{E}_j$  Pri $\mathcal{E}_j$  Pri $\mathcal{E}_j$  Pri $\mathcal{E}_j$  Pr  $\geq (1 - \frac{2}{n})$ *n* 2 *n*−1 2 *n*−2  $\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n-i+2}\right)\left(1-\frac{2}{n-i+1}\right)$  $=\left(\frac{p-2}{p-3}\right)\left(\frac{p-3}{p-2}\right)\left(\frac{p-4}{p-3}\right)\cdots\left(\frac{p-1}{p-1}\right)$ *n n*−1 *n*−2 *n*−*i*+2  $(n-i-1)$ *n*−*i*+1 *n*−*i* <u>) ( *n*−*i*−1</u> =  $(n-i)(n-i-1)$ *n*(*n* − 1) ≥  $(n-i-1)(n-i-1)$ *n* · *n*  $= (1$ *i* + 1 *n*  $\big)^2$ .

Probability becomes very small only towards the  $6\sigma$ 





More Amplification: Karger-Stein<br>
Motivation<br>
Pr[ $\mathcal{E}_i$ ]<br>
Pr[ $\mathcal{E}_i$ ] Pr[ $\mathcal{E}_j$ ] Pr[ $\mathcal{E}_j$  Pri $\mathcal{E}_i$  |  $\mathcal{E}_1 \cap ... \cap \mathcal{E}_{j-1}$ ]<br>  $\geq (1 - \frac{2}{n}) (1 - \frac{2}{n-1}) (1 - \frac{2}{n-2}) \cdot (1 - \frac{2}{n-1+2}) (1 - \frac{2}{n-1+1})$ <br>  $= \left(\frac$  $\geq (1 - \frac{2}{n})$ *n* 2 *n*−1 2 *n*−2  $\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n-i+2}\right)\left(1-\frac{2}{n-i+1}\right)$  $=\left(\frac{p-2}{p-3}\right)\left(\frac{p-3}{p-2}\right)\left(\frac{p-4}{p-3}\right)\cdots\left(\frac{p-1}{p-1}\right)$ *n n*−1 *n*−2 *n*−*i*+2  $(n-i-1)$ *n*−*i*+1 *n*−*i* <u>) ( *n*−*i*−1</u> =  $(n-i)(n-i-1)$ *n*(*n* − 1) ≥  $(n-i-1)(n-i-1)$ *n* · *n*  $= (1$ *i* + 1 *n*  $\big)^2$ .

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





More Amplification: Karger-Stein<br>
Motivation<br>
Pr(E<sub>i</sub>)<br>
Pr(E<sub>i</sub>) Pr(E<sub>i</sub>) Pr(E<sub>i</sub>) E<sub>i</sub> Pr(E<sub>i</sub>) (E<sub>i</sub> (E<sub>i</sub> O..., OE<sub>i</sub>, 1]<br>  $\geq (1-\frac{2}{n}) (1-\frac{2}{n-1})(1-\frac{2}{n-2}) \cdot (1-\frac{2}{n-1+1})$ <br>  $= \frac{(n-2)(1-2)}{(n-1)(n-1)} \geq (n-1)(n-1-1) = (n-1)(n$  $\geq (1 - \frac{2}{n})$ *n* 2 *n*−1 2 *n*−2  $\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n-i+2}\right)\left(1-\frac{2}{n-i+1}\right)$  $=\left(\frac{p-2}{p-3}\right)\left(\frac{p-3}{p-2}\right)\left(\frac{p-4}{p-3}\right)\cdots\left(\frac{p-1}{p-1}\right)$ *n n*−1 *n*−2 *n*−*i*+2  $(n-i-1)$ *n*−*i*+1 *n*−*i* <u>) ( *n*−*i*−1</u> =  $(n-i)(n-i-1)$ ≥  $(n-i-1)(n-i-1)$  $= (1$ *i* + 1  $\big)^2$ .

*n*(*n* − 1) *n* · *n n* **• 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

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



More Amplification: Karger-Stein<br>
Motivation<br>
Probability that a min-cut survives *i* contractions<br>
Pr[E<sub>i</sub>] - Pr[E<sub>i</sub>] - Pr[E<sub>i</sub>] - Pr[E<sub>i</sub>] - E<sub>i</sub> [E<sub>i</sub> (E<sub>i</sub>  $\geq (1 - \frac{2}{n})$ *n* 2 *n*−1 2 *n*−2  $\left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n-i+2}\right)\left(1-\frac{2}{n-i+1}\right)$  $=\left(\frac{p-2}{p-3}\right)\left(\frac{p-3}{p-2}\right)\left(\frac{p-4}{p-3}\right)\cdots\left(\frac{p-1}{p-1}\right)$ *n n*−1 *n*−2 *n*−*i*+2  $(n-i-1)$ *n*−*i*+1 *n*−*i* <u>) ( *n*−*i*−1</u> =  $(n-i)(n-i-1)$ *n*(*n* − 1) ≥  $(n-i-1)(n-i-1)$ *n* · *n*  $= (1$ *i* + 1 *n*  $\big)^2$ . **E**<sub>i</sub>] = Pr[ $\mathcal{E}_1$ ] + Pr[ $\mathcal{E}_2$ ]  $\geq \{1-\frac{2}{n}\}$  =  $\{1-\frac{2}{n}\}$  =  $\{1-\frac{2}{n-1}\}$  =  $\$ 

**• Probability becomes very small only towards the** very end.

After  $s = n - n/\sqrt{2} - 1$  steps we have

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

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



**10** Maximilian Katzmann, Sistin Walzer – Probability & Computing  $\frac{1}{\sqrt{2}}$  (a)  $\left|\begin{array}{l} \text{H}^1 \text{K}_0 \text{H} \text{H}_0 \text{H} \text{H}_0 \text{H} \text{H}_0 \text{H}_0 \text{H}_0 \text{H}_0 \end{array}\right| = 2 \text{ then return unique cut-} \begin{array}{l} \text{H}^1 \text{K}_0 \text{H}_0 \text{H}_0 \text{H}_0 \text{H}_0 \$  $\left\| \begin{array}{c} O(n) \ O(1) \end{array} \right\|$  for  $i=1$  to  $s= \mathcal{U}(E_{i-1})$  $\mathsf{KargerStein}(G_0 = (V_0, E_0))$  $\left| \frac{1}{2} \right| \left| \frac{1}{2} \right| \left| \frac{1}{2} \right| = 2$  then return unique cut-set  $\left| \begin{array}{c} | & | \end{array} \right|$   $\left| \begin{array}{c} G_i = G_{i-1}$ . **contract**(*e*) **return** smaller of  $C_1$ ,  $C_2$ **for**  $i = 1$  to  $s = |V_0| - \frac{|V_0|}{\sqrt{2}}$ 2 − 1 **do**  $C_1$  := **KargerStein**( $G_s$ )  $C_2$  := **KargerStein**( $G_s$ ) // pendent // inde-<br>// pend<br>// runs



**Recursion**<br> **•** After  $t = n - n/\sqrt{2} - 1$  steps the number<br>
of nodes is  $n/\sqrt{2} + 1$ <br>  $\frac{n}{\sqrt{O(n)}}$ <br>  $\frac{n}{\sqrt{O(1)}} = 2T\left(\frac{n}{n} + 1\right) + O(n^2)$ <br>  $\frac{n}{\sqrt{O(1)}}$ ecursion<br>After *t = n – n/* √2 – 1 steps the number of nodes is  $n/\sqrt{2}+1$ 

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

**10** Maximilian Katzmann, Sistin Walzer – Probability & computing  $\frac{1}{2}$  Maximilian Katzmann, Sistin Walzer – Probability & computing  $\frac{1}{2}$  metal or  $\frac{1}{2}$  metal  $\frac{1}{2}$  metal  $\frac{1}{2}$  metal  $\frac{1}{2}$  metal  $\begin{array}{c|c|c|c|c|c} \hline 2 + 1 & & \end{array}$  //  $O(n)$  **for**  $i = 1$  to  $s = |V_0| - \frac{|V_0|}{\sqrt{2}}$  $\mathsf{KargerStein}(G_0 = (V_0, E_0))$  $\left| \frac{1}{2} \right| \left| \frac{1}{2} \right|$  if  $|V_0| = 2$  then return unique cut-set  $e \coloneqq \mathcal{U}(E_{i-1})$  $\left| \begin{array}{c} | & | \end{array} \right|$   $\left| \begin{array}{c} G_i = G_{i-1}.$  **contract**(*e*) **return** smaller of  $C_1$ ,  $C_2$ 2 − 1 **do**  $C_1$  := **KargerStein**( $G_s$ )  $C_2$  := **KargerStein**( $G_s$ ) // pendent // inde-<br>// pend<br>// runs



**Recursion**<br> **•** After  $t = n - n/\sqrt{2} - 1$  steps the number<br>
of nodes is  $n/\sqrt{2} + 1$ <br>  $\frac{n}{\sqrt{O(n)}}$ <br>  $\frac{n}{\sqrt{O(1)}} = 2T\left(\frac{n}{n} + 1\right) + O(n^2)$ <br>  $\frac{n}{\sqrt{O(1)}}$ ecursion<br>After *t = n – n/* √2 – 1 steps the number of nodes is  $n/\sqrt{2}+1$ 

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

2 log *n*)

**10** Maximilian Katzmann, Sistin Walzer – Probability & computing  $\frac{1}{2}$  Maximilian Katzmann, Sistin Walzer – Probability & Computing  $\frac{1}{2}$  metal or  $\frac{1}{2}$  metal  $\frac{1}{2}$  metal  $\frac{1}{2}$  metal  $\frac{1}{2}$  metal  $\begin{array}{c|c|c|c|c|c} \hline 2 + 1 & & \end{array}$  //  $O(n)$  **for**  $i = 1$  to  $s = |V_0| - \frac{|V_0|}{\sqrt{2}}$  $\mathsf{KargerStein}(G_0 = (V_0, E_0))$  $\left| \frac{1}{2} \right| \left| \frac{1}{2} \right|$  if  $|V_0| = 2$  then return unique cut-set  $e \coloneqq \mathcal{U}(E_{i-1})$  $\left| \begin{array}{c} \left| \begin{array}{c} \left| \right| \end{array} \right| \left| \begin{array}{c} \right| \end{array} \right|$  *G*<sub>*i*</sub> = *G*<sub>*i*−1</sub>**.contract**(*e*) **return** smaller of  $C_1$ ,  $C_2$ 2 − 1 **do**  $C_1$  := **KargerStein**( $G_s$ )  $C_2$  := **KargerStein**( $G_s$ ) // pendent **Solution (essentially by Master Theorem)**  $C_1 := \text{KargerStein}(G_s)$  // inde-<br>  $T(n) = O(n^2 \log n)$ <br>  $T(n) = O(n^2 \log n)$ 



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

before calling itself recursively

 $\frac{1}{2}$ .

### **Auxiliary Problem**

**Karger-Stein: Success Probability**<br> **Know:** Each call to Karger-Stein breaks the min-cut with probability at most  $\frac{1}{2}$ .<br> **Auxiliary Problem**<br>
Given complete binary tree of height d where each node is ran-<br>
don't col Given complete binary tree of height *d* where each node is randomly coloured red or green (with probability  $\frac{1}{2}$  each). **Auxiliary Problem**<br>Given complete binary tree of height *d* where each node is randomly coloured red or green (with probability  $\frac{1}{2}$  each).<br>What is the probability  $p_d$  that a green root-to-leaf path exists?



**Auxiliary Problem**

**Karger-Stein: Success Probability**<br> **Know:** Each call to Karger-Stein breaks the min-cut with probability at most  $\frac{1}{2}$ .<br> **Auxiliary Problem**<br>
Given complete binary tree of height d where each node is ran-<br>
domly col 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? before calling itself recursively<br>
where each node is ran-<br>
ability  $\frac{1}{2}$  each).<br>  $(\text{oot-to-leaf path exists?})$ <br>  $(\text{bot})^2$  // root green, **not** no path in both left and right subtree

 $p_0 = 1/2$  // root green  $p_d = \frac{1}{2}$  $\frac{1}{2}(1-(1-\rho_{d-1})^2$ 





 $\frac{1}{2}$ .

 $\frac{1}{2}$ .

### **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? **blem**<br> **duplem**<br> **eduplem**<br> **edup** 



**Karger-Stein: Success Probability**<br> **Know:** Each call to Karger-Stein breaks the min-cut with probability at most  $\frac{1}{2}$ .<br> **Auxiliary Problem**<br>
Given complete binary tree of height d where each node is ran-<br>
domly col  $p_0 = 1/2$  // root green  $p_d = \frac{1}{2}$  $\frac{1}{2}(1-(1-p_{d-1})^2)$  // root green, **not** no path in both left and right subtree Claim:  $\rho_d \geq \frac{1}{d+1}$ 

 $\frac{1}{2}$ .

### **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**<br> **Know:** Each call to Karger-Stein breaks the min-cut with probability at most  $\frac{1}{2}$ .<br> **Auxiliary Problem**<br>
Given complete binary tree of height d where each node is ran-<br>
domly col  $p_0 = 1/2$  // root green  $p_d = \frac{1}{2}$  $\frac{1}{2}(1-(1-p_{d-1})^2)$  // root green, **not** no path in both left and right subtree **Claim:**  $p_d \ge \frac{1}{d+2}$ . Proof by induction.  $\rho_0=\frac{1}{2}$  $\frac{1}{2} = \frac{1}{0+2}$   $\checkmark$  $p_d=\frac{1}{2}$  $\frac{1}{2}\big(1-\left(1- p_{d-1}\right)^2\big) \geq \frac{1}{2}$  $\frac{1}{2}\big(1-(1-\frac{1}{d+1})^2\big)=\frac{1}{2}$  $\frac{1}{2} \Big( \frac{2}{d+1} - \frac{1}{(d+1)}$  $\frac{1}{(d+1)^2}$  $=\frac{1}{2}$  $\frac{1}{2} \cdot \frac{2d+2-1}{(d+1)^2} = \frac{1}{2}$  $\frac{1}{2}\cdot\frac{2d+1}{d^2+2d+1}\geq\frac{1}{2}$  $rac{1}{2} \cdot \frac{2d}{d^2 + 2}$  $\frac{2d}{d^2+2d}=\frac{1}{d+1}$ **before calling itself recursively**<br> **where each node is ran-<br>
<b>ability**  $\frac{1}{2}$  **each).**<br> **oot-to-leaf path exists?**<br>  $p_{d-1}$ )<sup>2</sup>) // root green, **not** no path in both left<br>  $\geq \frac{1}{2}(1-(1-\frac{1}{d+1})^2) = \frac{1}{2}(\frac{2}{d+1} \frac{a}{b} \geq \frac{a-1}{b-1}$ *b*−1





 $\frac{1}{2}$ . the 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**<br> **Know:** Each call to Karger-Stein breaks the min-cut with probability at most  $\frac{1}{2}$ .<br> **Auxiliary Problem**<br>
Given complete binary tree of height d where each node is ran-<br>
domly col  $p_0 = 1/2$  // root green  $p_d = \frac{1}{2}$  $\frac{1}{2}(1-(1-p_{d-1})^2)$  // root green, **not** no path in both left and right subtree **Claim:**  $p_d \ge \frac{1}{d+2}$ . Proof by induction.  $\rho_0=\frac{1}{2}$  $\frac{1}{2} = \frac{1}{0+2}$   $\checkmark$  $p_d=\frac{1}{2}$  $\frac{1}{2}\big(1-\left(1- p_{d-1}\right)^2\big) \geq \frac{1}{2}$  $\frac{1}{2}\big(1-(1-\frac{1}{d+1})^2\big)=\frac{1}{2}$  $\frac{1}{2} \Big( \frac{2}{d+1} - \frac{1}{(d+1)}$  $\frac{1}{(d+1)^2}$  $=\frac{1}{2}$  $\frac{1}{2} \cdot \frac{2d+2-1}{(d+1)^2} = \frac{1}{2}$  $\frac{1}{2}\cdot\frac{2d+1}{d^2+2d+1}\geq\frac{1}{2}$  $rac{1}{2} \cdot \frac{2d}{d^2 + 2}$  $\frac{2d}{d^2+2d}=\frac{1}{d+1}$  $\frac{1}{d+2}$  *W* for  $1 \le a \le b$  we have  $\frac{a}{b} \ge \frac{a-1}{b-1}$ *b*−1

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







**<sup>12</sup>** Maximilian Katzmann, Stefan Walzer – Probability & Computing Institute of Theoretical Informatics, Algorithm Engineering & Scalable Algorithms **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  $1/O(\log(n))$ .



**12** Maximilian Katzmann, Stefan Walzer – Probability at least  $1/O(\log(n))$ .<br> **12** Maximilian Computing Institute of Theoretical Into Institute of Theoretical Information Success probability  $\geq p$ <br> **14** Maximilian Katzmann, **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  $1/O(\log(n))$ .

Success probability  $\geq p$ Number of repetitions *t* Amplified prob.  $> 1 - e^{-pt}$ **Amplification**<br> $\begin{array}{c|c} \text{Success probability} \geq \rho \end{array}$ <br> $\begin{array}{c} \text{Success probability} \geq \rho \end{array}$ <br> $\begin{array}{c} \text{Number of repetitions } t \end{array}$ 



**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**

Karger-Stein Amplified				
Theorem: On a graph with <i>n</i> 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	$Pr["min-cut found"]$	$\geq 1 - \exp\left(-\frac{t}{O(\log(n))}\right)$	$= 1 - O\left(\frac{1}{n}\right)$	Success probability $\geq p$
Number of repetitions $t$				
Amplified prob. $\geq 1 - e^{-pt}$				
Mathilian Katzman, Stefan Walzer – Probability & Computing	Institute of Theoretical Informatics, Algorithm Engineering & Scalable Algorithm			

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


**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 <i>n</i> 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	$Pr[$ "min-cut found"] $\geq 1 - \exp\left(-\frac{t}{O(\log(n))}\right) = 1 - O\left(\frac{1}{n}\right)$	Success probability $\geq p$
Corollary: On a graph with <i>n</i> 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.		
Maximilian Katzman, Stefan Walzer-Probability & Computing		

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



**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 <i>n</i> 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	$\Pr[$ "min-cut found"] $\geq 1 - \exp\left(-\frac{t}{O(\log(n))}\right) = 1 - O\left(\frac{1}{n}\right)$	Success probability $\geq p$
Corollary: On a graph with <i>n</i> 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 $O(n^3)$ for deterministic approaches		
Maximilian Katzman, Stefan Walzer – Probability & Computing		
Maximilian Katzman, Stefan Walzer – Probability & Computing		

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  $\Omega(n^3)$ 

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



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

## **Randomized Algorithms**

■ Karger's edge-contraction algorithm





- **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**

- **Probability Amplification**<br>
 Monte Carlo algorithms with and without biases<br>
 Repetitions amplify success probability
- **Repetitions amplify success probability**
- Karger-Stein: Amplify before failure probability gets large







- **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**

- **Probability Amplification**<br>
 Monte Carlo algorithms with and without biases<br>
 Repetitions amplify success probability
- **Repetitions amplify success probability**
- **Karger-Stein: Amplify before failure probability gets large**

# **Outlook**

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

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

*i* Faster algorithms for edge connectivity via random 2-out contractions", Ghaffari & Nowicki & Thorup, SODA'20<br>
Success w.h.p. in time  $O(m \log(n))$  and  $O(m + n \log^3(n))$ 









- -
- 
- -
	-
	-
- **Mögliche Prüfungsfragen<br>
 Was ist ein Monte-Carlo-Algorithmus?<br>
 Welche Varianten gibt es?<br>
 Was versteht man unter Probability Amplification?<br>
 Welche Varianten gibt es?<br>
 Le einsetigem Fehler?<br>
 Le einsetigem Feh** 
	- -
		-
		- - Was bedeutet  $Pr[\mathcal{E}_t]$  und wie haben wir diese Wahrscheinlichkeit abgeschätzt?
			- Was ergibt sich für die Laufzeit und die Erfolgswahrscheinlichkeit?
		- - Wie haben wir die Erfolgswahrscheinlichkeit und Laufzeit abgeschätzt?
			- *n* ?