Karlsruhe Institute of Technology

# Probability & Computing

## Probability Amplification

# The Segmentation Problem

**Input**

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

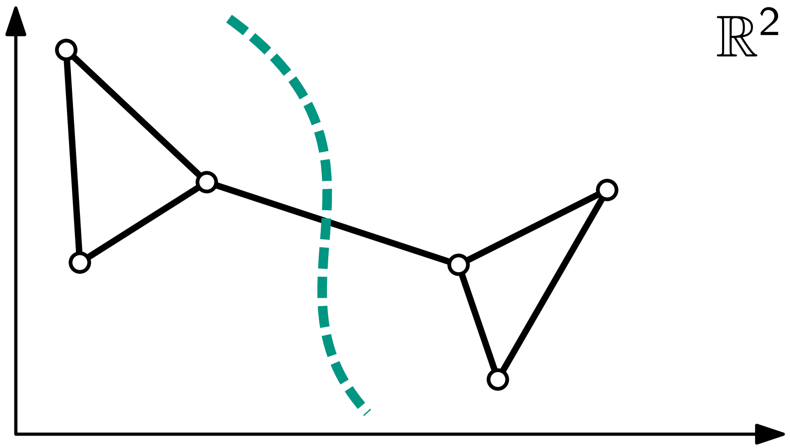
**Output**: $P_1, \ldots, P_k$ such that

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

**Applications**: Compression, medical diagnosis, etc.

**Approach**: Model as graph

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




**Example**

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

**Today**

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

# The Edge-Connectivity Problem

## Cuts

- $G = (V, E)$ an unweighted, undirected, connected graph
- *Cut*: partition of $V$ into parts $V_1$, $V_2$ such that $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$. (in general one can consider more than two parts)
- *Cut-set*: set of edges with one endpoint in $V_1$ and the other in $V_2$
- *Weight*: size of the cut-set (or sum of weights in a weighted graph)

## Excursion: Cuts with Terminals

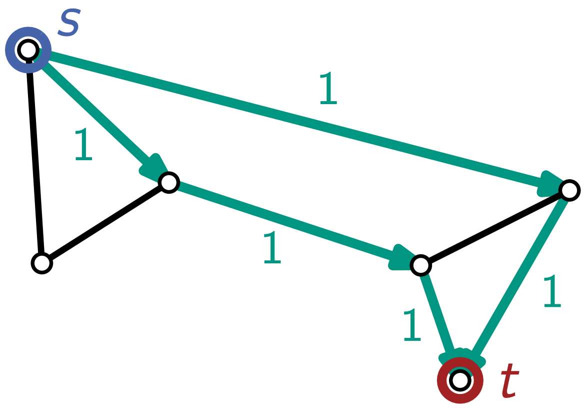
- each part contains exactly one of a specified vertex set

## $k$-Edge-Connectivity

- $k$*-edge-connected*: a minimum cut has weight at least $k$ (we cannot disconnect the graph by removing less than $k$ edges)

## Edge-Connectivity

- max. $k$ such that $G$ is $k$-edge-connected (exactly the weight of a min-cut)




## Excursion: Flows

- given source $s$ and target $t$
- assign *flow* to edges s.t.
  - in-flow = out-flow for all vertices (not $s$ and $t$)
  - flow of an edge bounded by edge-capacity (here: $\leq 1$)
  - flow in $t$ is maximized

**Thm.** Max-Flow = Min-Cut.

# Deterministic Algorithms for Edge-Connectivity

Karlsruhe Institute of Technology

## Flow-based

“Max flows in O(nm) time, or better”, Orlin, STOC’13

- Compute max-flow between all vertex pairs $\rightarrow O(n^2 \cdot \underbrace{T_{\text{max-flow}}}_{O(nm)}) \subseteq O(n^3 m)$
- Compute max-flow between $v$ and all others $\rightarrow O(n \cdot T_{\text{max-flow}}) \subseteq O(n^2 m) \rightarrow \Omega(n^3)$

  (if a cut of size $k$ exists, it has to cut $v$ from some vertex)

## Matroid-based

“A Matroid Approach to Finding Edge Connectivity and Packing Arborescences”, Gabow, JCSS, 1995

- Involved technique based on the fact that min-cut $=$ max. number of dijsoint, directed spanning trees $\rightarrow O(m + k^2 n \log(n/k))$
- Good if $k$ is small but still $\Omega(n^3)$ in the worst case

## Contraction-based

“A simple min-cut algorithm”, Stoer & Wagner, JACM, 1997

- Iteratively pick two vertices (in a smart way) and compare the min-cuts where they are / are not in the same part $\rightarrow O(mn + n^2 \log(n)) \rightarrow \Omega(n^3)$

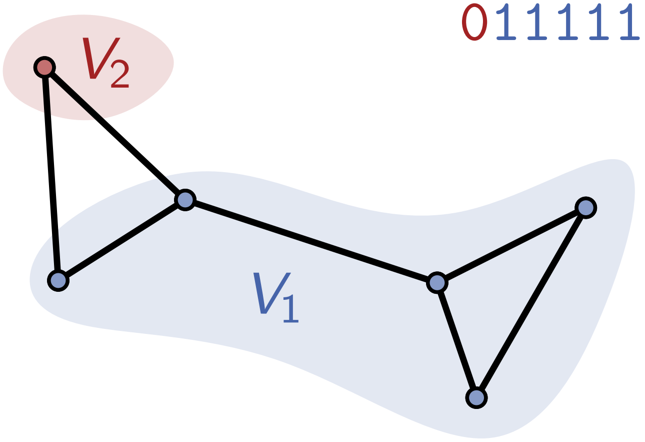
*Enter: The Power of Randomness!*

# A Simple(?) Randomized Algorithm

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

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

011111




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

**Algorithm: Simple(?) Randomized Cut**

- Simple idea: choose a cut at random among all possible cuts and return it.

  What do we mean?
  What distribution?

- Uniform distribution: We do not want to potentially favor non-minimum cuts
- Problem: How do we choose a cut *uniformly* at random?
  - Represent cut using bit-string
  - How can we choose a unfiorm random bit-string *while avoiding* `11...1` and `00...0`?

$n$ random bits? $\rightarrow$ does not avoid `11...1` and `00...0`

random number from $\{1, \ldots, 2^n - 2\}$? $\rightarrow$ exponential in input size

rejection sampling? running time not deterministic (though probably what you'd do in practice)

# Excursion: Uniform Non-Identical Bit Strings

`[For educational purposes only!]`

- **Goal**: Choose uniformly at random from the length $n$ bit-strings that are not $\mathtt{0}^n$ or $\mathtt{1}^n$
- Number of valid bit-strings:

$$2^n - 2 = \left(\sum_{k=0}^{n}\binom{n}{k}\right) - 2 = \sum_{k=1}^{n-1}\binom{n}{k}$$

$$2^n = \sum_{k=0}^{n}\binom{n}{k} \qquad \binom{n}{0} = \binom{n}{n} = 1$$

**Assumptions**: We can sample ...
- uniformly from $\{0, ..., O(n+m)\}$ in $O(1)$ time
- uniformly from $[0, 1]$ in $O(1)$ time

Not possible in theory. Reasonable in practice.

- 2-step process: choose $k$ & choose $k$ 1s in $n$ bits

**unibs**$(n)$

$b := 00\ldots0$ // $n$ zeros
$k :=$ **rand**$(\{1, \ldots, n-1\})$ // number of 1s
$P :=$ **randSet**$(\{1, \ldots, n\}, k)$ // positions of 1s
$b[P] = 1$ // set 1s in $b$
**return** $b$

**How to sample $k$?**
- uniform?

$$\left.\begin{aligned}\Pr[1000] &= 1/3 \cdot 1/4 = 1/12\\ \Pr[1100] &= 1/3 \cdot 1/6 = 1/18\end{aligned}\right\} \neq 1/\underbrace{14}_{2^n-2}$$

- choose $k$ with prob $\binom{n}{k}/(2^n - 2)$
- Reduce to uniform using *Inverse Transform Sampling*

**How to sample $P$?**

$n = 4$

$$\left.\begin{matrix}1000\\0100\\0010\\0001\end{matrix}\right\} k = 1 \qquad \left.\begin{matrix}1100\\1010\\1001\\0110\\0101\\0011\end{matrix}\right\} k = 2 \qquad \left.\begin{matrix}1110\\1101\\1011\\0111\end{matrix}\right\} k = 3$$

# Excursion-Excursion: Reservoir Sampling

[For educational purposes only!]

- **Goal**: Choose a set of size $k$ uniformly at random from the $n$ elements.
- **Idea**:
  - initialize **reservoir** with first $k$ elements
  - replace reservoir elements at random

**Assumptions**: We can sample . . .
- uniformly from $\{0, ..., O(n + m)\}$ in $O(1)$ time
- uniformly from $[0, 1]$ in $O(1)$ time

Not possible in theory. Reasonable in practice.

**randSet**$(\{1, \ldots, n\}, k)$

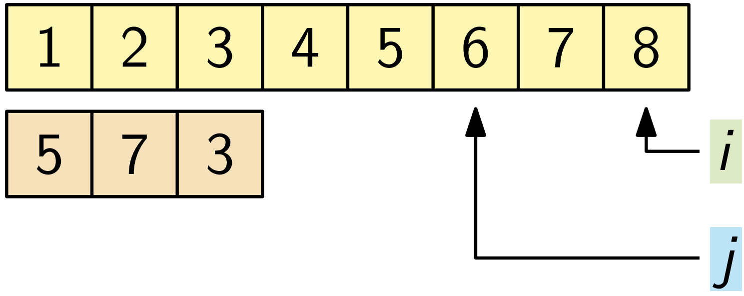
$r := [1, \ldots, k]$ // reservoir // $O(k)$

**for** $i$ from $k + 1$ to $n$ **do** // $O(n - k)$

  $j :=$ **unif**$(\{1, \ldots, i\})$ // $O(1)$

  **if** $j \leq k$ **then** $r[j] = i$ // $O(1)$

**return** $r$




- **Running time**: $O(n)$

# Excursion: Uniform Non-Homogeneous Bit Strings

[For educational purposes only!]

- **Goal**: Choose uniformly at random from the length $n$ bit strings that are not $0^n$ or $1^n$
- 2-step process:
  - choose $k$
  - choose $k$ 1s in $n$ bits

**Assumptions**: We can sample ...
- uniformly from $\{0, ..., O(n+m)\}$ in $O(1)$ time
- uniformly from $[0, 1]$ in $O(1)$ time

Not possible in theory. Reasonable in practice.

**unibs**$(n)$

$b := 00...0$ // $n$ zeros — // $O(n)$

$k :=$ **rand**$(\{1, ..., n-1\})$ // number of 1s — // $O(\log(n))$ via Inverse Transform Sampling

$P :=$ **randSet**$(\{1, ..., n\}, k)$ // positions of 1s — // $O(n)$ via Reservoir Sampling

$b[P] = 1$ // set 1s in $b$ — // $O(k) \subseteq O(n)$

**return** $b$

Under our assumptions, we can sample a length $n$ bit string that is not $0^n$ or $1^n$ uniformly at random in time $O(n)$.

# Simple Randomized Cut

- Simple idea: choose a cut uniformly at random among all possible cuts and return it.
- Running time: $O(n)$ much better than the $\Omega(n^3)$ in the deterministic setting , but...

## Success probability

- $2^{n-1} - 1$ cuts in a graph with $n$ nodes
- How many min-cuts? $\rightarrow$ pessimistic assumption: 1

**Observation**: On a graph with $n$ nodes, **Simple Randomized Cut** runs in $O(n)$ time and returns a minimum cut with probability at least $1/(2^{n-1} - 1)$.$\rightarrow$ exponentially small!

## Amplification

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

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

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

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

# Probability Amplification

**Definition**: A **Monte Carlo Algorithm** is a randomized algorithm that terminates deterministically and whose output is correct only with a certain probability $p \in (0, 1)$.

- 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

bias
majority
best

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

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

- After $t$ (independent) runs return the ...

  $\Pr[\text{“success”}] \geq 1 - (1-p)^t \geq 1 - e^{-pt}$ (for two-sided errors it's a bit more complicated)

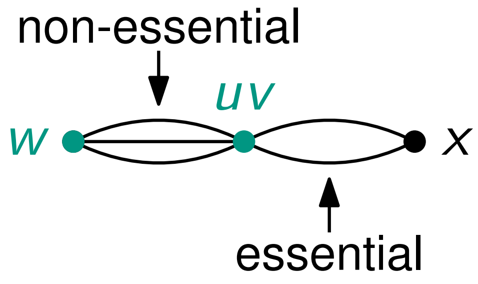
- Error probability decreases exponentially in $t$

For Simple Randomized Cut we had to pay with exponentially large running time ...

# Karger's Algorithm

## Edge Contraction

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




## Contraction Algorithm

- Motivation: distinguish $\overbrace{\textit{non-essential}}^{\text{not part of a min-cut}}$ as well as *essential* edges } part of a min-cut & hope there are few essential ones

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

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

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

## Success Probability

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

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

**Observation**: min-degree $\geq k$

(holds for all $G_i$ due to 1st observation)

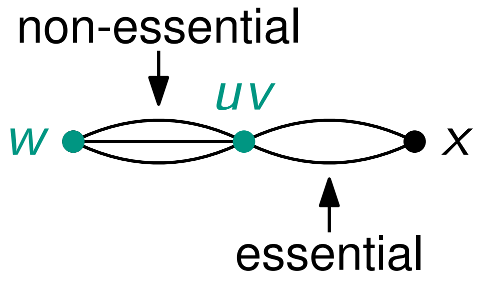
$$\Pr[\mathcal{E}_1] = 1 - \frac{k}{m} \geq 1 - \frac{k}{nk/2} = 1 - \frac{2}{n}$$

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

# Karger's Algorithm

## Edge Contraction

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




## Contraction Algorithm

- Motivation: distinguish $\overbrace{\textit{non-essential}}^{\text{not part of a min-cut}}$ as well as *essential* edges $\}$ part of a min-cut
  & hope there are few essential ones

**Karger**$(G_0 = (V_0, E_0))$
  **for** $i = 1$ to $n-2$ **do** // $O(n)$
    $e :=$ **unif**$(E_{i-1})$ // $O(1)$
    $G_i = G_{i-1}$.**contract**$(e)$ // $O(n)$
  **return** unique cut in $G_{n-2}$

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

## Success Probability

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

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

**Observation**: min-degree $\geq k$

(holds for all $G_i$ due to 1st observation)

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

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

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

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

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

# Karger's Algorithm Amplified

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

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

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

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

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

Much better than exp. time Simple Randomized Cut!

## Sidenote: Number of minimum cuts

- Let $C_1, \ldots, C_\ell$ be all the min-cuts in $G$ and $\underbrace{\mathcal{E}^i_{n-2}}_{\text{disjoint, since the algorithm returns only one cut}}$ for $i \in [\ell]$ be the event that $C_i$ is returned by Karger's algorithm
- Just seen: $\Pr[\mathcal{E}^i_{n-2}] \geq \frac{2}{n(n-1)}$

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

**Observation**: A graph on $n$ nodes contains at most $\frac{n(n-1)}{2}$ minimum cuts.


Institute of Theoretical Informatics, Algorithm Engineering & Scalable Algorithms

# More Amplification: Karger-Stein

## Motivation

- Probability that a min-cut survives $i$ contractions

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

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

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

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

- With increasing number of steps the probability for a min-cut to survive decreases

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

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

$$\Pr[\mathcal{E}_t] = \frac{(\cancel{n-n}+n/\sqrt{2}+1)(\cancel{n-n}+n/\sqrt{2}+\cancel{1-1})}{n(n-1)} = \frac{n^2/2+n/\sqrt{2}}{n(n-1)} = \frac{\cancel{n}(n/2+1/\sqrt{2})}{\cancel{n}(n-1)} = \frac{1}{2} \cdot \underbrace{\frac{n+\sqrt{2}}{n-1}}_{\geq 1} \geq \frac{1}{2}$$

- Probability that no mistake made after $t$ steps still large

# Karger-Stein: Running Time

Karlsruhe Institute of Technology

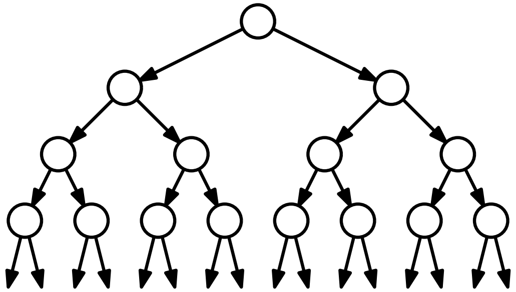
## Recursion

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

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

## Recursion tree

- Layers: $\log_{\sqrt{2}}(n)$
- Nodes on layer $j$: $2^j$
- Time on layer $j$: $O\left(\left(\frac{n}{\sqrt{2}^j}\right)^2\right)$



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

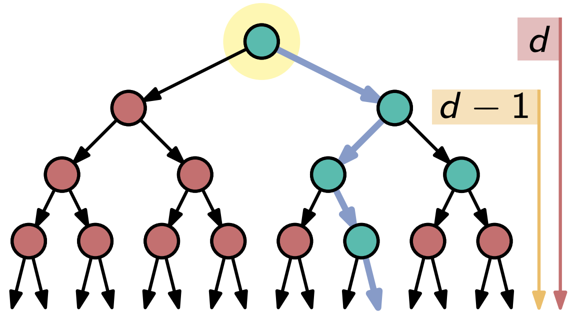
```
// O(1)   if |V_0| = 2 then return unique cut
// O(n)   for i = 1 to t = |V_0| - |V_0|/√2 - 1 do
// O(1)       e := unif(E_{i-1})
// O(n)       G_i = G_{i-1}.contract(e)
          C_1 := KargerStein(G_t)   // inde-
          C_2 := KargerStein(G_t)   // pendent
                                  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

$$T(n) = \sum_{j=1}^{\log_{\sqrt{2}}(n)} 2^j \cdot O\left(\left(\frac{n}{\sqrt{2}^j}\right)^2\right) = O\left(n^2 \cdot \sum_{j=1}^{\log_{\sqrt{2}}(n)} \frac{2^j}{2^j}\right) = O(n^2 \log_{\sqrt{2}}(n)) = O(n^2 \log(n))$$

# Karger-Stein: Success Probability

- After $t = n - n/\sqrt{2} - 1$ steps we have $\Pr[\mathcal{E}_t] \geq 1/2$ ($t$ was chosen to achieve exactly that)

## Recursion tree




- A node is a successful node if it still contains a min-cut of the original graph
- A path is a successful path if it contains only successful nodes
- $\mathcal{P}_d$: there exists a successful path of length $d$ starting at the root

$\Pr[\mathcal{P}_0] \geq 1/2$

$$\Pr[\mathcal{P}_d] = \Pr[\mathcal{P}_0] \cdot \left(1 - (1 - \Pr[\mathcal{P}_{d-1}])^2\right) \geq \frac{1}{2} \cdot \left(1 - (1 - \Pr[\mathcal{P}_{d-1}])^2\right) = \Pr[\mathcal{P}_{d-1}] - \frac{1}{2}\Pr[\mathcal{P}_{d-1}]^2$$

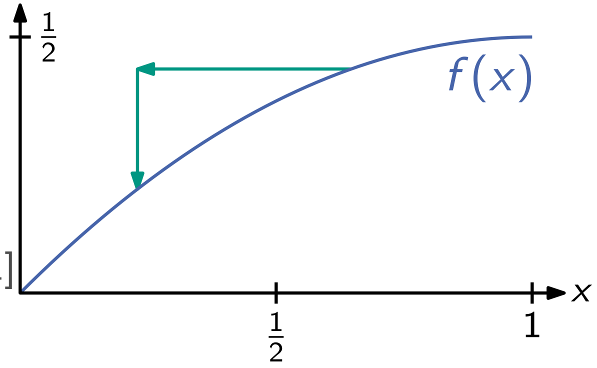
**Claim** $\Pr[\mathcal{P}_d] \geq \frac{1}{d+2}$ (proof via induction)

- Base case $d = 0$: $\Pr[\mathcal{P}_0] \geq 1/2$, Assumption: $\Pr[\mathcal{P}_{d-1}] \geq \frac{1}{d+1}$
- Step: $\Pr[\mathcal{P}_d] \geq \Pr[\mathcal{P}_{d-1}] - \frac{1}{2}\Pr[\mathcal{P}_{d-1}]^2$
  $\geq \frac{1}{d+1} - \frac{1}{2}\left(\frac{1}{d+1}\right)^2$

$f(x) = x - \frac{1}{2}x^2$

$f(x)\frac{\mathrm{d}}{\mathrm{d}x} = \underbrace{1 - x}_{\geq 0}$

for $x \in [0, 1]$

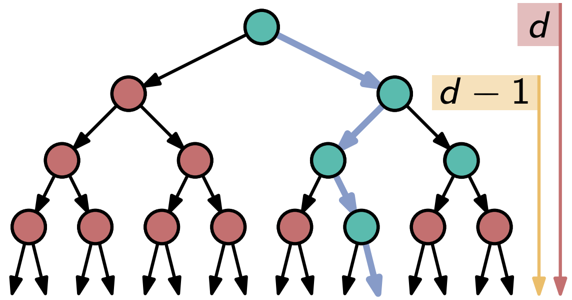



smaller $x$ yields smaller $f(x)$

# Karger-Stein: Success Probability

- After $t = n - n/\sqrt{2} - 1$ steps we have $\Pr[\mathcal{E}_t] \geq 1/2$ (t was chosen to achieve exactly that)

## Recursion tree




- A node is a successful node if it still contains a min-cut of the original graph
- A path is a successful path if it contains only successful nodes
- $\mathcal{P}_d$: there exists a successful path of length $d$ starting at the root

$\Pr[\mathcal{P}_0] \geq 1/2$

$\Pr[\mathcal{P}_d] = \Pr[\mathcal{P}_0] \cdot \left(1 - (1 - \Pr[\mathcal{P}_{d-1}])^2\right) \geq \frac{1}{2} \cdot \left(1 - (1 - \Pr[\mathcal{P}_{d-1}])^2\right) = \Pr[\mathcal{P}_{d-1}] - \frac{1}{2}\Pr[\mathcal{P}_{d-1}]^2$

**Claim** $\Pr[\mathcal{P}_d] \geq \frac{1}{d+2}$ (proof via induction)

- Base case $d = 0$: $\Pr[\mathcal{P}_0] \geq 1/2$, Assumption: $\Pr[\mathcal{P}_{d-1}] \geq \frac{1}{d+1}$
- Step: $\Pr[\mathcal{P}_d] \geq \Pr[\mathcal{P}_{d-1}] - \frac{1}{2}\Pr[\mathcal{P}_{d-1}]^2$

$$\geq \frac{1}{d+1} - \frac{1}{(2d+2)(d+1)}$$

$2d \geq d$ for $d \geq 0$

$$\geq \frac{1}{d+1} - \frac{1}{(d+2)(d+1)}$$

$$= \frac{\cancel{d+1}}{(\cancel{d+1})(d+2)} = \frac{1}{d+2}$$

- $\Pr[\text{“min-cut on layer } d\text{”}] \geq \frac{1}{d+2}$
- How many layers in the tree?
  $\rightarrow \log_{\sqrt{2}}(n)$
- $\Pr[\text{“min-cut returned”}] \geq \frac{1}{O(\log(n))}$

# Karger-Stein Amplified

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

Reminder: Karger $\rightarrow 1/O(n^2)$ in $O(n^2)$ time

## Amplification

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

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

Success probability $\geq p$

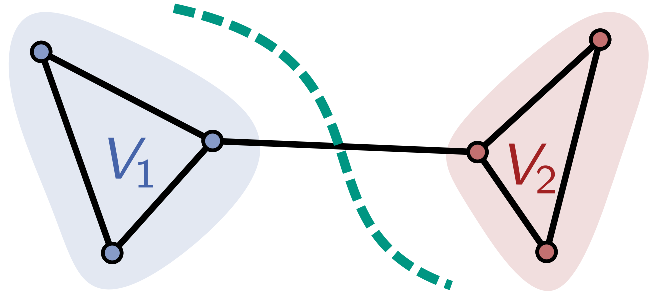
Number of repetitions $t$

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

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

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

# Conclusion




## Cuts

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

**Assumptions**: We can sample …
- uniformly from $\{0, \ldots, O(n+m)\}$ in $O(1)$ time
- uniformly from $[0, 1]$ in $O(1)$ time

Not possible in theory. Reasonable in practice.

## Randomized Algorithms

- Simple randomized cut via reservoir sampling
- Karger's edge-contraction algorithm

## Probability Amplification

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

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

## Outlook

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

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

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

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