Advanced Algorithms

Introduction: Min, Max, and Sparsest Cuts

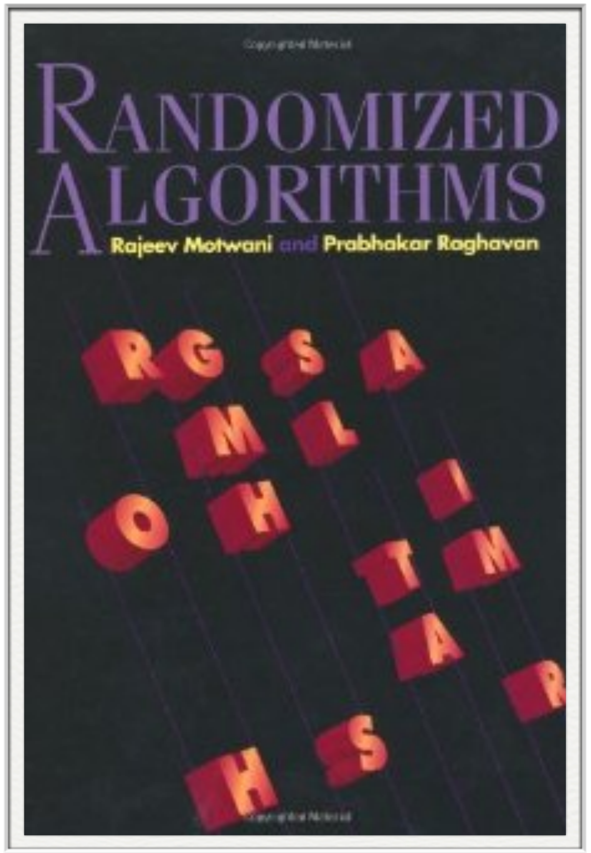
尹一通 Nanjing University, 2023 Fall

Course Info

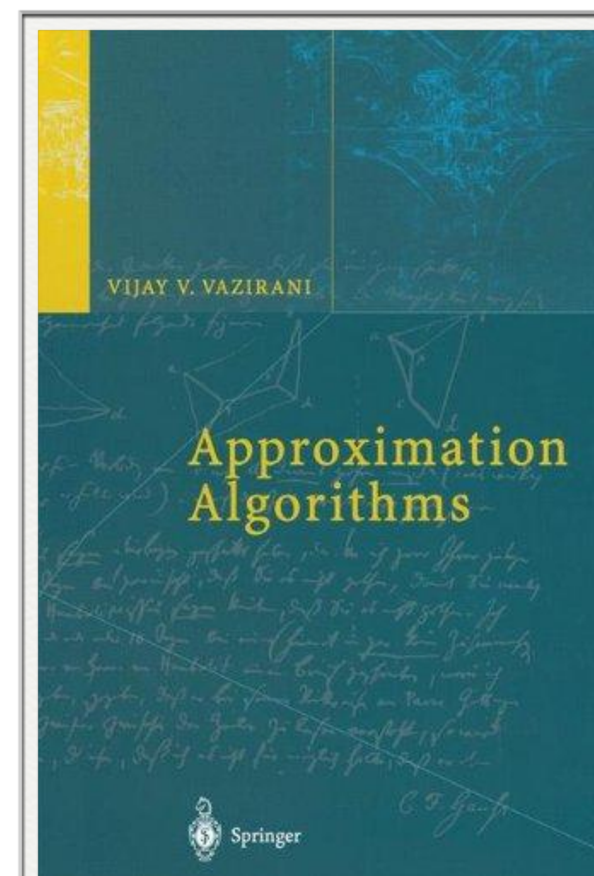
高级算法（南京大学…
群号: 876680440

- Instructors（授课时间顺序）：
  - 尹一通、刘景铖、栗师
  - {yinyt, liu, shili}@nju.edu.cn
- Office hour:
  - 804 Thursday 2-4pm（尹一通）
- QQ group: 876680440
- Course webpage: http://tcs.nju.edu.cn/wiki/

Textbooks

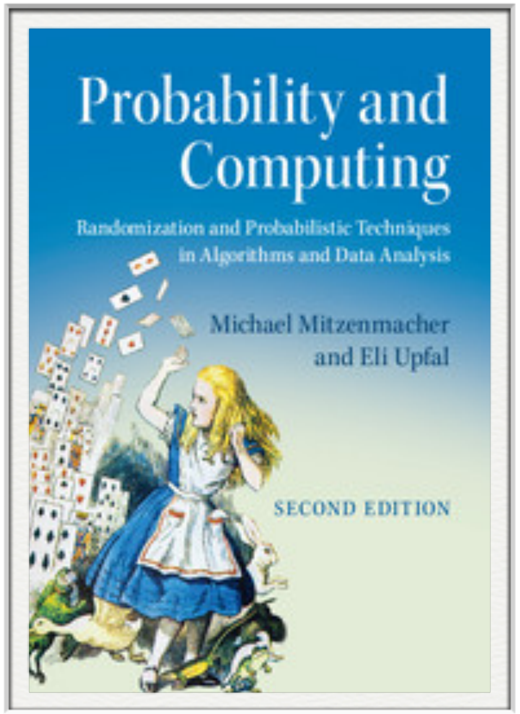


Rajeev Motwani and Prabhakar Raghavan.
Randomized Algorithms.
Cambridge University Press, 1995.



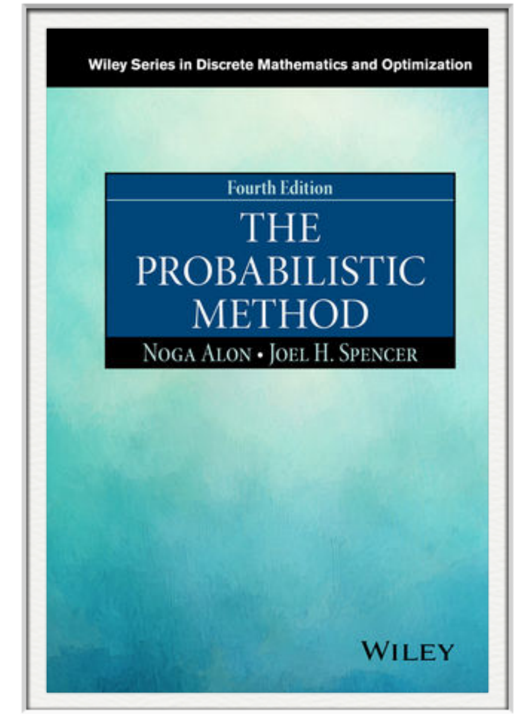
Vijay Vazirani
Approximation Algorithms.
Spinger-Verlag, 2001.

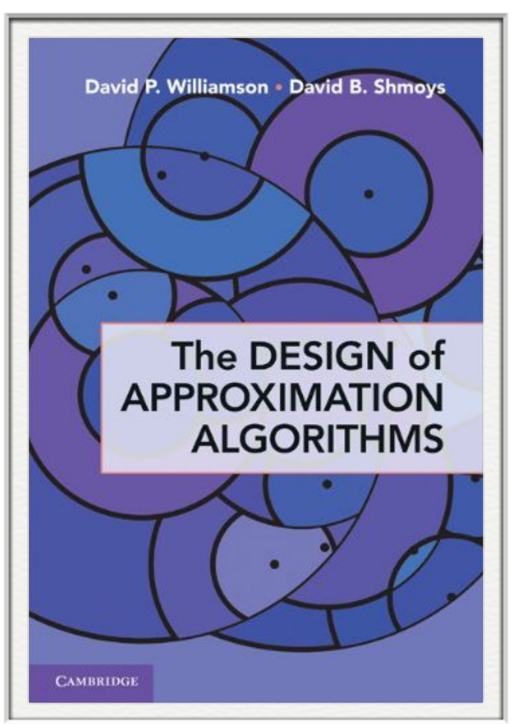
References



Mitzenmacher and Upfal.
Probability and Computing,
2nd Ed.

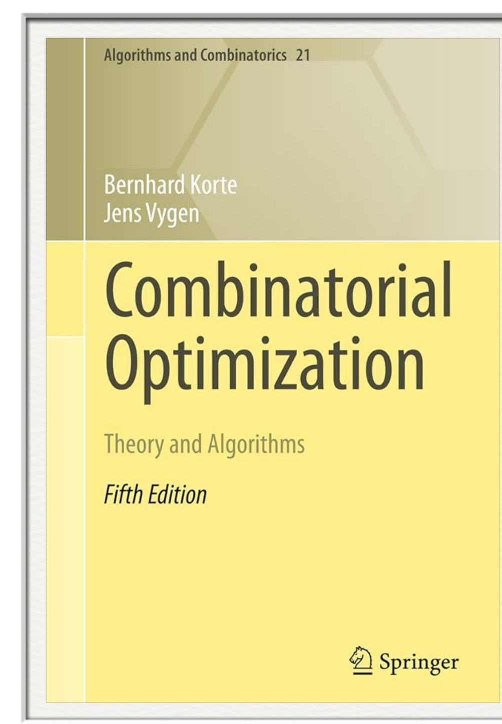
Alon and Spencer
The Probabilistic Method,
4th Ed.



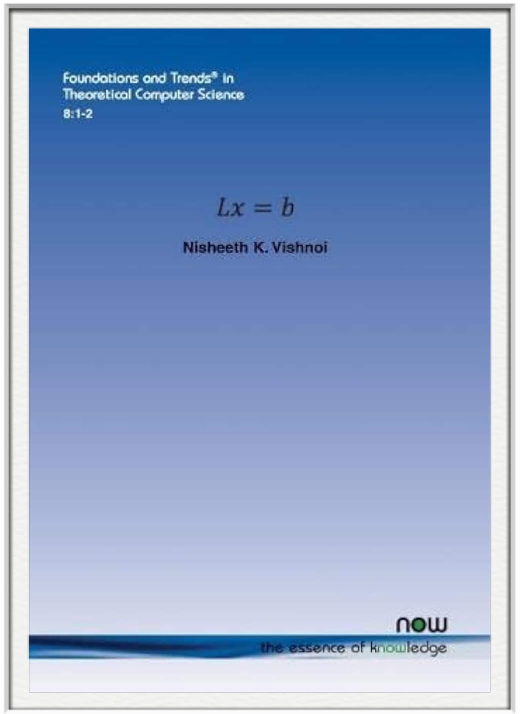


Williamson and Shmoys
The Design of
Approximation Algorithms

Korte and Vygen
Combinatorial Optimization:
Theory and Algorithms

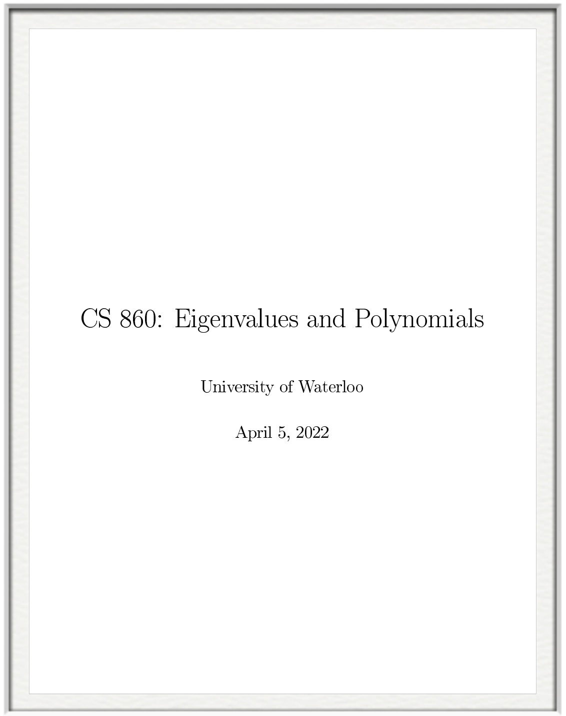


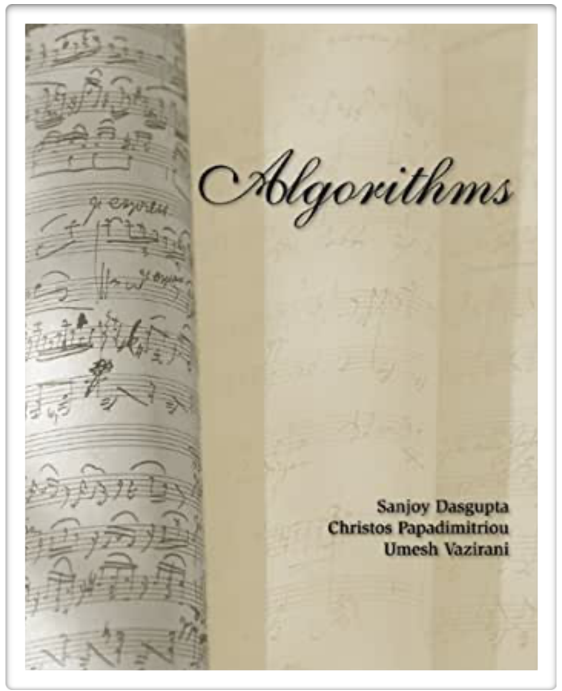
References



Nisheeth Vishnoi
$Lx = b$

Lap Chi Lau
Eigenvalues and Polynomials





DPV
Algorithms

Muḥammad ibn Mūsā
al-Khwārizmī
(780-850)

Advanced Algorithms

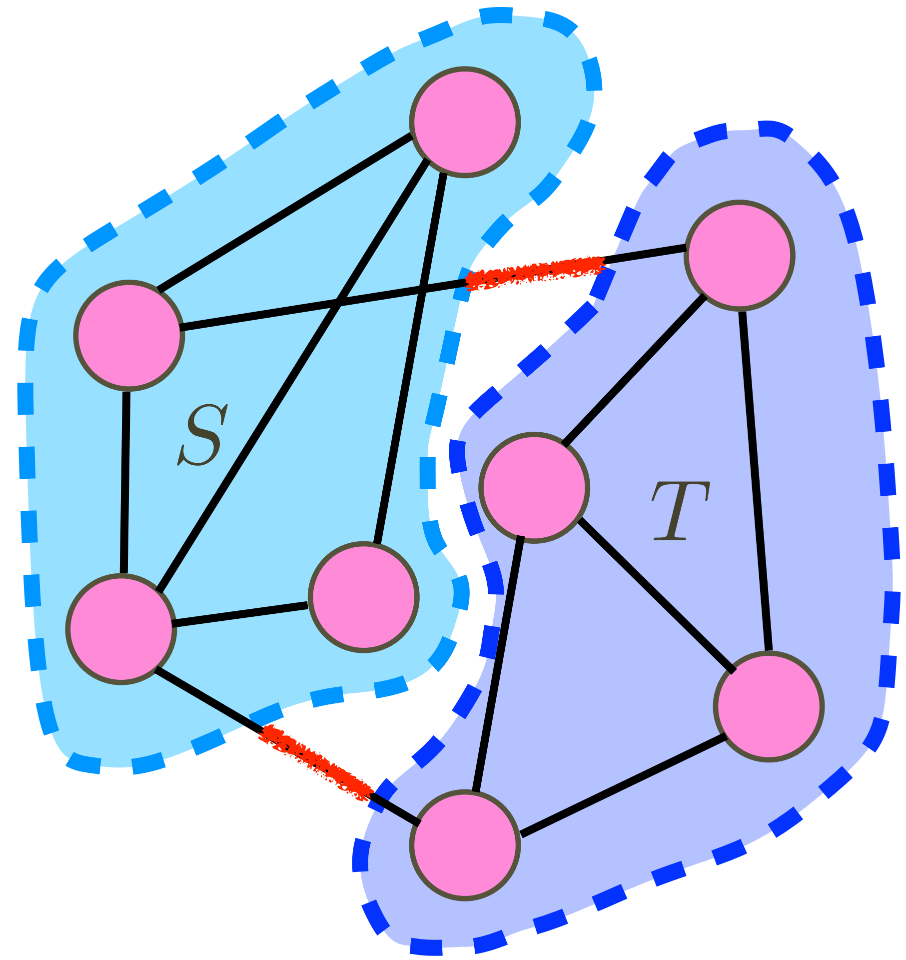
Minimum Cut

(*Randomized Algorithms*)

Min-Cut

- Undirected graph $G(V, E)$
- Bi-partition of V into nonempty S and T
- Find a cut $E(S, T)$ of smallest size (***global*** min-cut)
- **Deterministic** algorithms:
  - max-flow min-cut (**duality**)
  - best upper bound (2021): $m^{1+o(1)}$

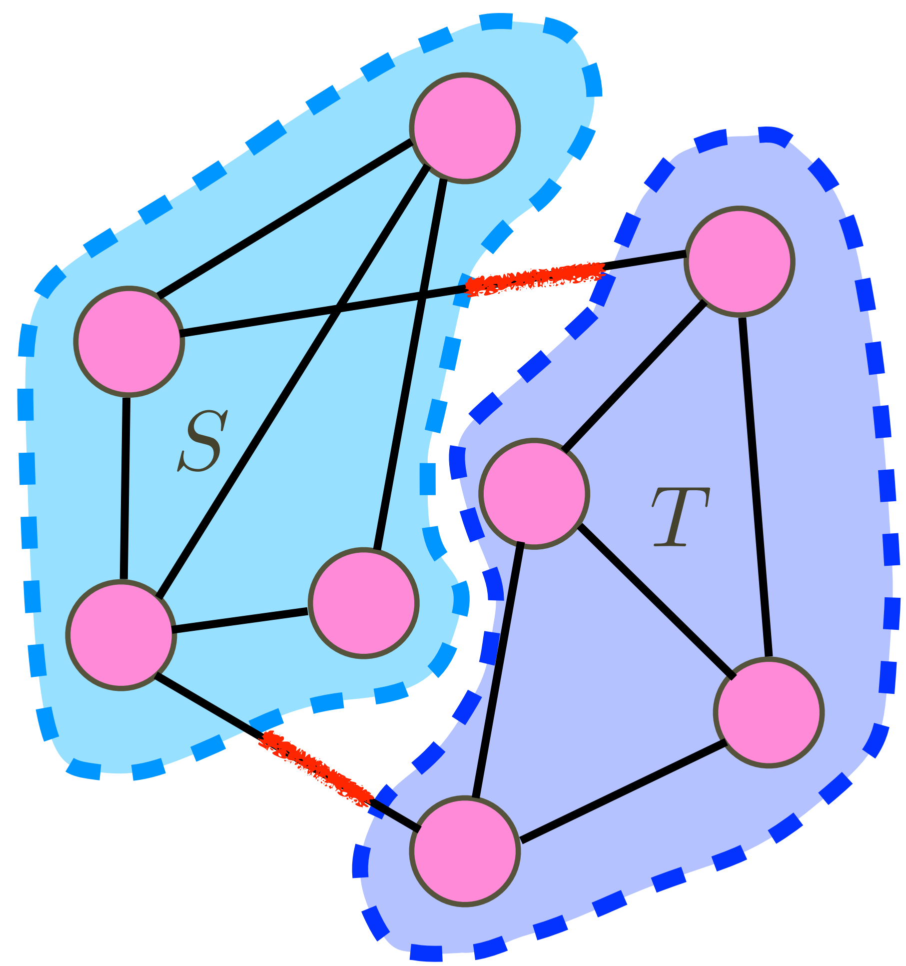
$$E(S,T) := \{uv \in E \mid u \in S, v \in T\}$$



Min-Cut

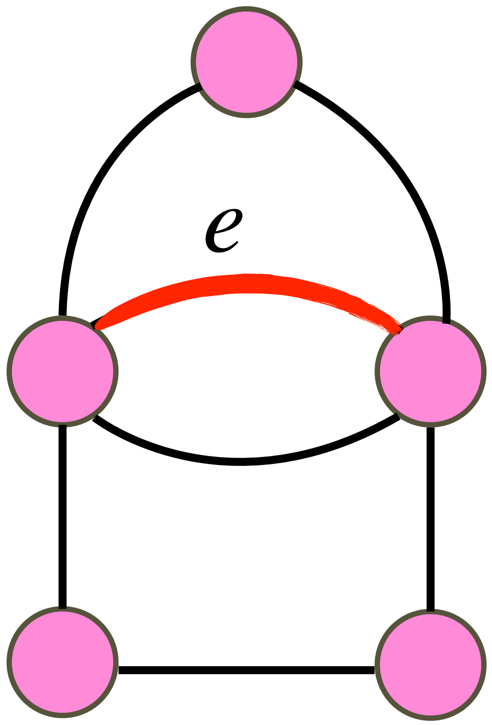
- Undirected graph $G(V, E)$
- Too many cuts:
  - there are $2^{\Omega(n)}$ bi-partitions
  - (will see later) only at most $O(n^2)$ min-cuts
- **Generate a random cut that is a min-cut with large enough probability?**

$$E(S, T) := \{uv \in E \mid u \in S, v \in T\}$$



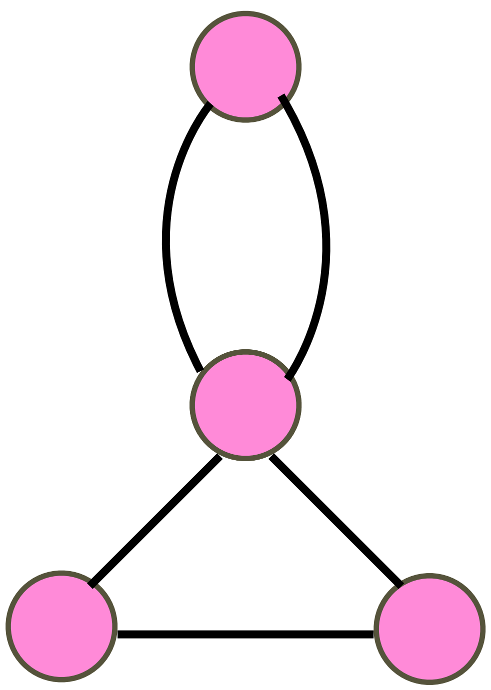
Contraction

- Undirected multigraph $G(V, E)$
- multigraph:
  - allow parallel edges
  - but no self-loop
- contract(e): $e = uv$ is an edge
  - merges two endpoints u and v



Contraction

- Undirected multigraph $G(V, E)$
- multigraph:
  - allow parallel edges
  - but no self-loop
- contract(e): $e = uv$ is an edge
  - merges two endpoints u and v



Karger's Min-Cut Algorithm

- multigraph $G(V, E)$

Karger's Algorithm:

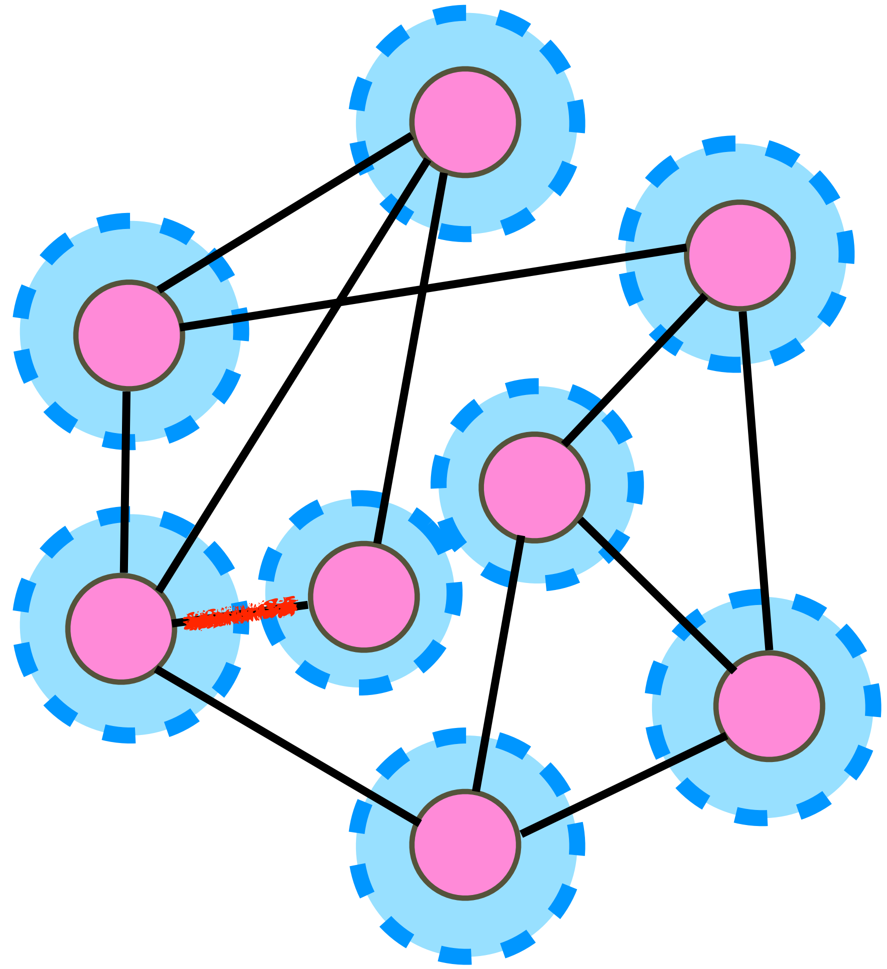
while $|V| > 2$ do:

  pick random $e \in E$;

  contract(e);

return remaining edges;

random: uniformly and independently at random



Karger's Min-Cut Algorithm

- multigraph $G(V, E)$

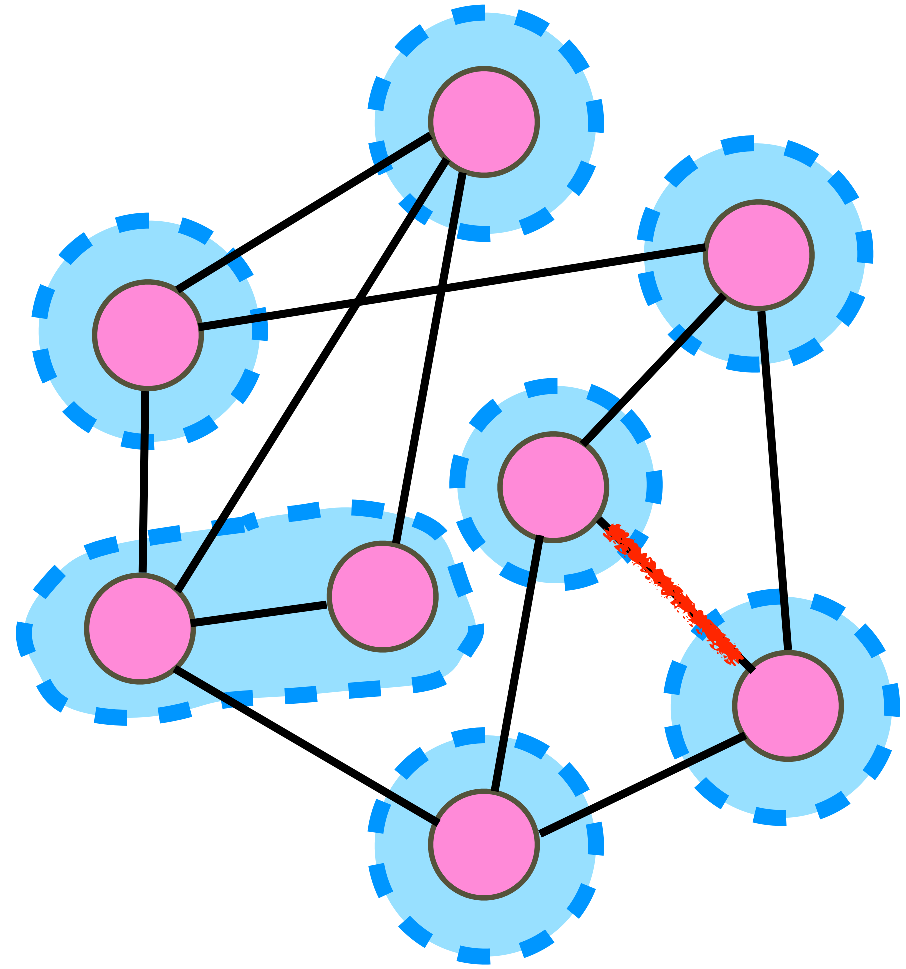
Karger's Algorithm:

while $|V| > 2$ do:

  pick random $e \in E$;

  contract(e);

return remaining edges;



Karger's Min-Cut Algorithm

- multigraph $G(V, E)$

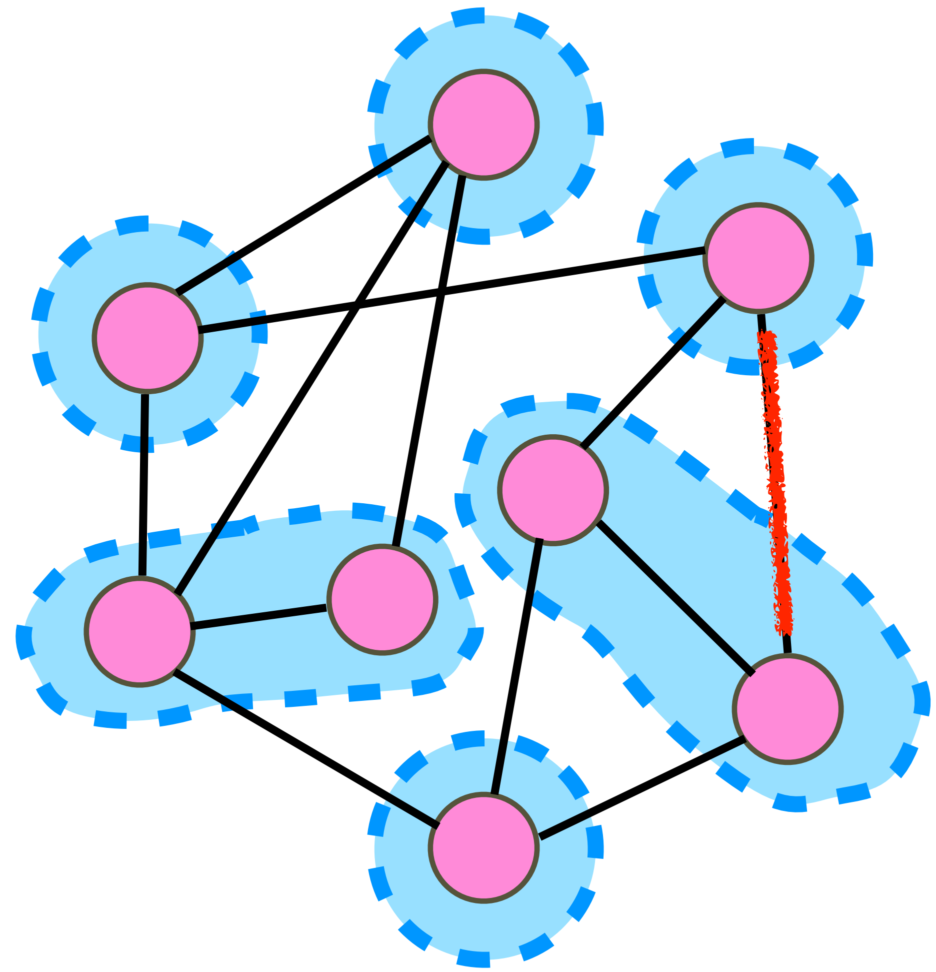
Karger's Algorithm:

while $|V| > 2$ do:

  pick random $e \in E$;

  contract(e);

return remaining edges;



Karger's Min-Cut Algorithm

- multigraph $G(V, E)$

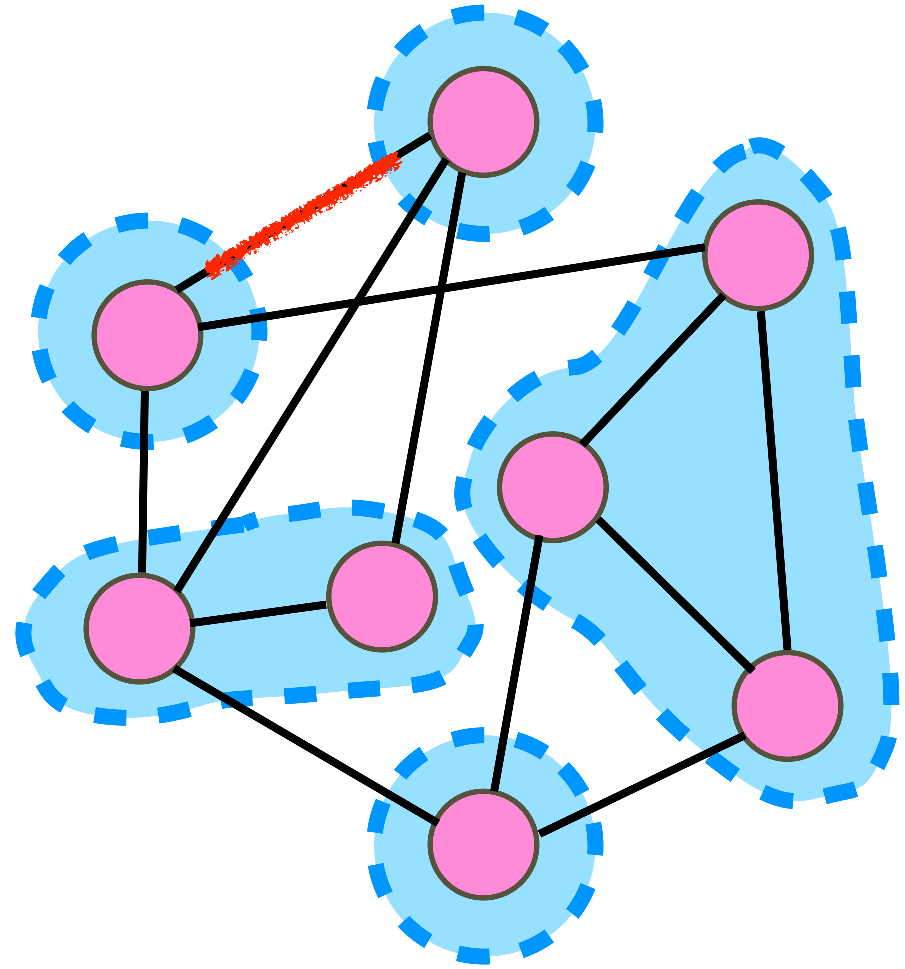
Karger's Algorithm:

while $|V| > 2$ do:

 pick random $e \in E$;

 contract(e);

return remaining edges;



Karger's Min-Cut Algorithm

- multigraph $G(V, E)$

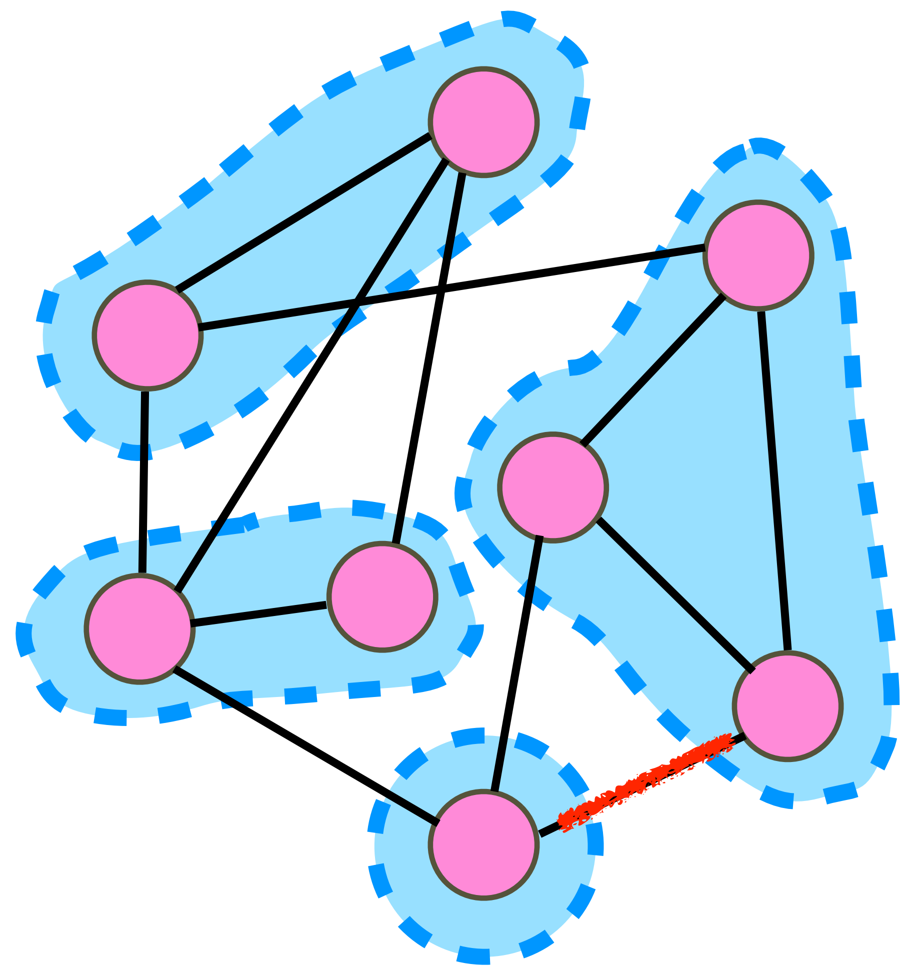
Karger's Algorithm:

while $|V| > 2$ do:

  pick random $e \in E$;

  contract(e);

return remaining edges;



Karger's Min-Cut Algorithm

- multigraph $G(V,E)$

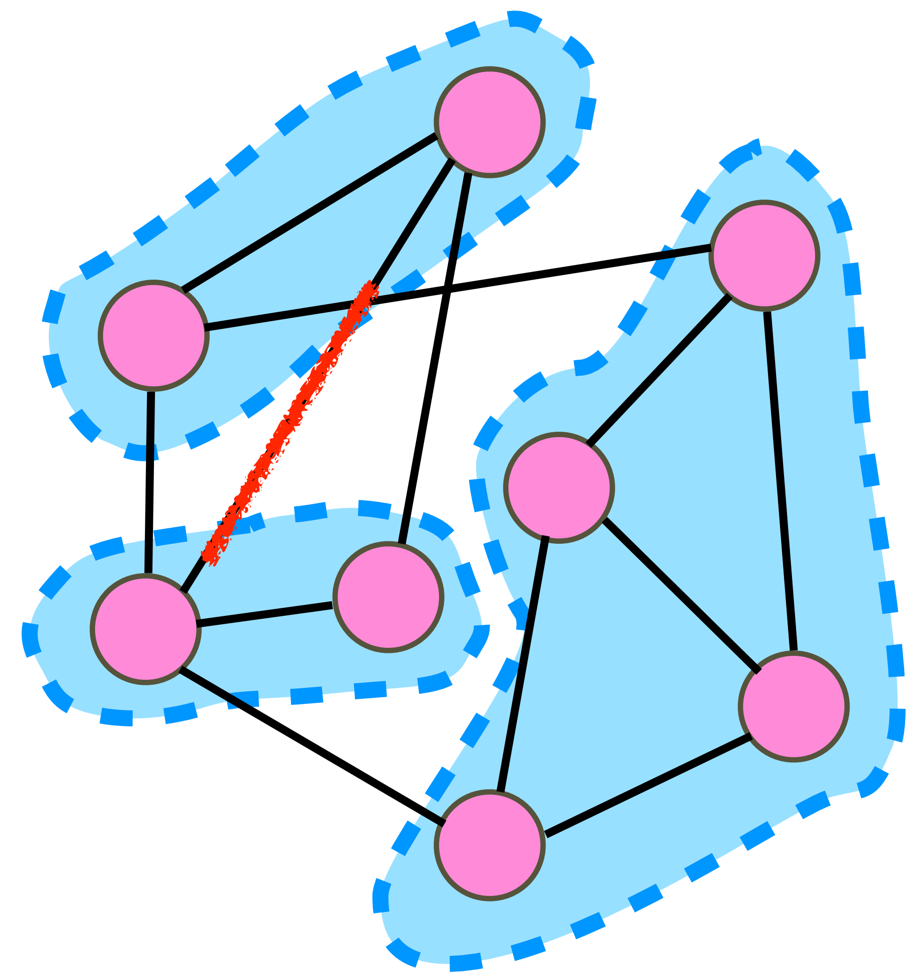
Karger's Algorithm:

while $|V| > 2$ do:

  pick random $e \in E$;

  contract(e);

return remaining edges;



Karger's Min-Cut Algorithm

- multigraph $G(V, E)$

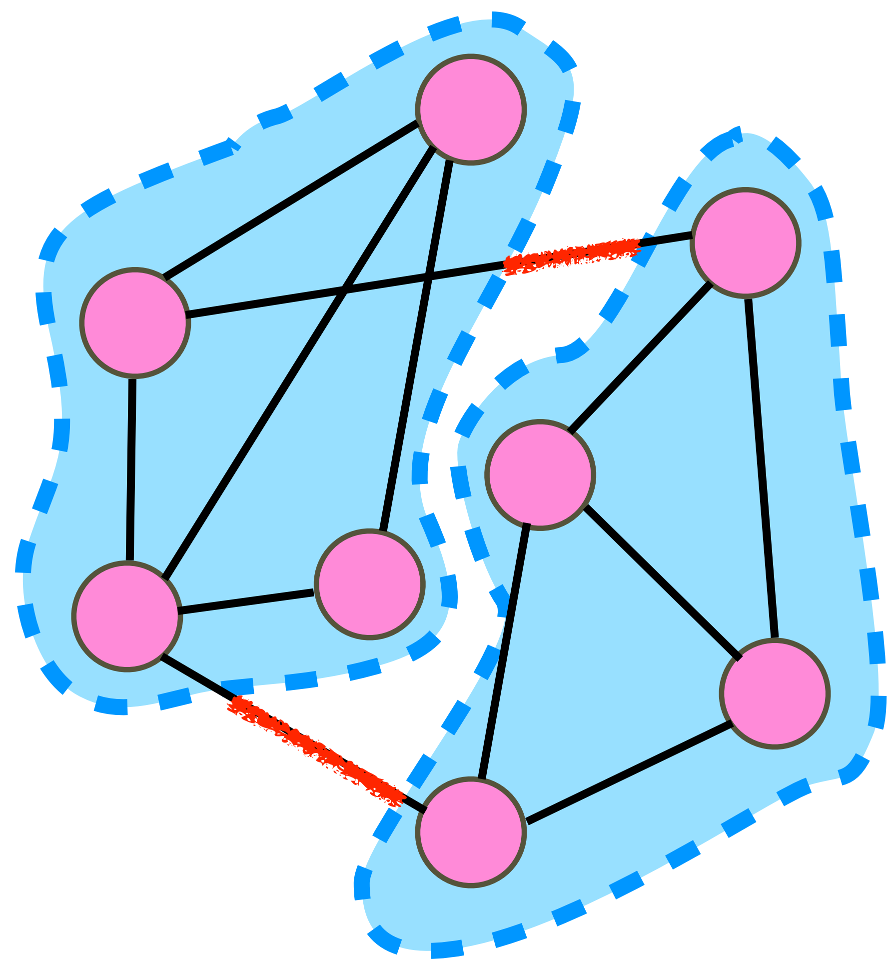
Karger's Algorithm:

while $|V| > 2$ do:

pick random $e \in E$;

contract(e);

return remaining edges;



Karger's Algorithm:

while $|V| > 2$ do:

pick random $e \in E$;

contract(e);

return remaining edges;

Theorem (Karger 1993).

$$\Pr[\text{ a min-cut is returned }] \geq \frac{2}{n(n-1)}$$

repeat independently for $\frac{1}{2}n(n-1)\ln n$ times and return the smallest cut

$$\Pr[\text{ fail to output a min-cut at last }]$$

$$= \Pr[\text{ fail to output a min-cut in one trial }]^{\frac{n(n-1)}{2}\ln n}$$

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^{\frac{n(n-1)}{2}\ln n} < \left(\frac{1}{\mathrm{e}}\right)^{\ln n} = \frac{1}{n}$$

Succeed *with high probability* (*w.h.p.*)!

Karger's Algorithm:

while $|V| > 2$ do:

pick random $e \in E$;

contract(e);

return remaining edges;

Sequence of contracted edges:

$$e_1, e_2, \ldots, e_{n-2}$$

Initially: $G_0 = G$ input multigraph

i-th iteration:

$$G_i = \text{contract}(G_{i-1}, e_i)$$

Contraction does not create new edges or cuts:

$C \subseteq E_i$ is a cut in $G_i \implies C \subseteq E_{i-1}$ and C is a cut in G_{i-1}

Non-contracted cut persists: $C \subseteq E_{i-1}$ is a cut in G_{i-1}

$e_i \notin C \implies C \subseteq E_i$ and C is a cut in G_i

$$\Downarrow$$

Observation: Suppose $C \subseteq E_{i-1}$ is a min-cut in G_{i-1}.

$e_i \notin C \implies C$ is a min-cut in G_i

Karger's Algorithm:

while $|V| > 2$ do:

pick random $e \in E$;

contract(e);

return remaining edges;

Sequence of contracted edges:

$$e_1, e_2, \ldots, e_{n-2}$$

Initially: $G_0 = G$ input multigraph

i-th iteration:

$$G_i = \text{contract}(G_{i-1}, e_i)$$

Observation: Suppose $C \subseteq E_{i-1}$ is a min-cut in G_{i-1}.

$e_i \notin C \implies C$ is a min-cut in G_i

Fix an arbitrary min-cut C in G.

$$\Pr[\, C \text{ is returned} \,] \geq \Pr[e_1, e_2, \ldots, e_{n-2} \notin C]$$

chain rule: $$= \prod_{i=1}^{n-2} \Pr[e_i \notin C \mid e_1, e_2, \ldots, e_{i-1} \notin C]$$

Sequence of contracted edges: $e_1, e_2, \ldots, e_{n-2}$

Initially: $G_0 = G$ $\quad$ i-th iteration: $G_i = \text{contract}(G_{i-1}, e_i)$

Observation: Suppose $C \subseteq E_{i-1}$ is a min-cut in G_{i-1}.

$e_i \notin C \implies C$ is a min-cut in G_i

Fix an arbitrary min-cut C in G.

$$\Pr[\, C \text{ is returned}\,] \geq \prod_{i=1}^{n-2} \Pr[e_i \notin C \mid e_1, e_2, \ldots, e_{i-1} \notin C]$$

C is a min-cut in G_{i-1}

Observation:

C is a min-cut in $G(V, E)$

$\implies |E| \geq \frac{1}{2}|C||V|$

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

Proof:

min-degree of $G \geq |C|$

$$= \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} = \frac{2}{n(n-1)}$$

$O(n^2)$ time

Karger's Algorithm:

while $|V| > 2$ do:

pick random $e \in E$;

$\boxed{\text{contract}(e)}$; $O(n)$ time (with help of some data structure)

return remaining edges;

Theorem (Karger 1993).

For any min-cut C in an input graph $G(V,E)$,

$$\Pr\left[\, C \text{ is returned} \,\right] \geq \frac{2}{n(n-1)}$$

repeat independently for $\frac{1}{2}n(n-1)\ln n$ **times**

and return the smallest cut

Find a min-cut *with high probability* (w.h.p.) in $\tilde{O}(n^4)$ time.

$\tilde{O}(\cdot)$: ignore poly-logarithmic factors

Number of Min-Cuts

Theorem (Karger 1993)**.**

For any min-cut C in an input graph $G(V, E)$,

$$\Pr\left[\, C \text{ is returned} \,\right] \geq \frac{2}{n(n-1)}$$

Corollary (Karger 1993)**.**

The number of distinct min-cuts in any graph of n vertices is at most $n(n-1)/2$.

Suppose there are M distinct min-cuts: $C_1, C_2, \ldots, C_M$

$$1 \geq \Pr[\text{ a min-cut is returned }] = \sum_{i=1}^{M} \Pr[\, C_i \text{ is returned }]$$

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

An Observation

Contract(G, t):

while $|V| > t$ do:

pick random $e \in E$;

contract(e);

return remaining edges;

- multigraph $G(V, E)$
- sequence of contracted edges: $e_1, e_2, \ldots, e_{n-t}$

C: a min-cut in G.

$\mathscr{E}$: no edge in C is contracted in Contract(G, t).

$$\Pr[\,\mathscr{E}\,] \geq \prod_{i=1}^{n-t} \Pr[e_i \notin C \mid e_1, e_2, \ldots, e_{i-1} \notin C]$$

$$\geq \prod_{i=1}^{n-t} \frac{n-i-1}{n-i+1} = \frac{t(t-1)}{n(n-1)}$$

only getting bad when t is small

Faster Min-Cut

Contract(G, t):

while $|V| > t$ do:

  pick random $e \in E$;

  contract(e);

return remaining edges;

FastCut(G):

if $|V| \le 6$ then return a min-cut by brute force;

else: (t to be fixed later)

$$\left.\begin{aligned} G_1 &= \text{Contract}(G, t); \\ G_2 &= \text{Contract}(G, t); \end{aligned}\right\} \text{(independently)}$$

return $\min\left\{\text{FastCut}(G_1), \text{FastCut}(G_2)\right\}$;

C: a min-cut in G.

$\mathscr{E}$: no edge in C is contracted in Contract(G, t).

$$\Pr[\,\mathscr{E}\,] \ge \frac{t(t-1)}{n(n-1)}$$

Contract(G, t):

while $|V| > t$ do:

  pick random $e \in E$;

  contract(e);

return remaining edges;

FastCut(G):

if $|V| \le 6$ then return a min-cut by brute force;

else: set $t = n/\sqrt{2} + 1$

$$\left.\begin{aligned} G_1 &= \text{Contract}(G, t); \\ G_2 &= \text{Contract}(G, t); \end{aligned}\right\} \text{(independently)}$$

return $\min\left\{\text{FastCut}(G_1), \text{FastCut}(G_2)\right\}$;

C: a min-cut in G.

$\mathscr{E}$: no edge in C is contracted in Contract(G, t).

$$\Pr[\,\mathscr{E}\,] \ge \frac{t(t-1)}{n(n-1)} \ge \frac{1}{2}$$

$$p(n) = \min_{G:|V|=n} \Pr\left[\text{FastCut}(G)\ \text{succeeds}\right]$$

returns a min-cut in G

$$\ge 1 - \left(1 - \Pr[\mathscr{E}]\Pr\left[\text{FastCut}(G_1)\ \text{succeeds} \mid \mathscr{E}\right]\right)^2$$

$$\ge 1 - \left(1 - \frac{t(t-1)}{n(n-1)}p(t)\right)^2 \ge p\left(\frac{n}{\sqrt{2}} + 1\right) - \frac{1}{4}p\left(\frac{n}{\sqrt{2}} + 1\right)^2$$

Contract(G, t):

while $|V| > t$ do:

  pick random $e \in E$;

  contract(e);

return remaining edges;

FastCut(G):

if $|V| \le 6$ then return a min-cut by brute force;

else: set $t = n/\sqrt{2} + 1$

$$\left.\begin{aligned} G_1 &= \text{Contract}(G, t); \\ G_2 &= \text{Contract}(G, t); \end{aligned}\right\} \text{(independently)}$$

return $\min\left\{\text{FastCut}(G_1), \text{FastCut}(G_2)\right\}$;

$$p(n) = \min_{G:|V|=n} \Pr\left[\,\text{FastCut}(G)\text{ succeeds}\,\right]$$

$$\ge p\left(\frac{n}{\sqrt{2}} + 1\right) - \frac{1}{4} p\left(\frac{n}{\sqrt{2}} + 1\right)^2$$

- **Verified by induction:** $p(n) = \Omega\left(\frac{1}{\log n}\right)$
- **Recursion for time cost:** $T(n) \le 2T\left(n/\sqrt{2} + 1\right) + O(n^2)$
- **Verified by induction:** $T(n) = O(n^2 \log n)$

Contract(G, t):

while $|V| > t$ do:

pick random $e \in E$;

contract(e);

return remaining edges;

FastCut(G):

if $|V| \leq 6$ then return a min-cut by brute force;

else: set $t = n/\sqrt{2} + 1$

$\left.\begin{array}{l} G_1 = \text{Contract}(G, t); \\ G_2 = \text{Contract}(G, t); \end{array}\right\}$ (independently)

return $\min\left\{\text{FastCut}(G_1), \text{FastCut}(G_2)\right\}$;

Theorem (Karger and Stein 1996)**.**

On any graph G of n vertices, FastCut(G) runs in $O(n^2 \log n)$ time and returns a min-cut in G with probability $\Omega(1/\log n)$.

repeat independently for $O(\log^2 n)$ times and return the smallest cut

Find a min-cut *with high probability* (w.h.p.) in $\tilde{O}(n^2)$ time.

Min-Cut

- **Randomized** (*Monte Carlo*) algorithms: (correct w.h.p.)
  - Karger's *Contraction* algorithm (1993): $\tilde{O}(n^4)$
  - Karger-Stein Algorithm (1996): $\tilde{O}(n^2)$
  - Karger's *Tree-packing* Algorithm (2000): $\tilde{O}(m)$
- **Deterministic** algorithms:
  - max-flow min-cut (**duality**): $n \times$ max-flow computation
  - Stoer-Wagner Algorithm (1997): $\tilde{O}(mn)$
  - (only for single graphs) Kawarabayashi-Thorup (2015): $\tilde{O}(m)$
  - Jason Li (2021): $m^{1+o(1)}$

conductance-based

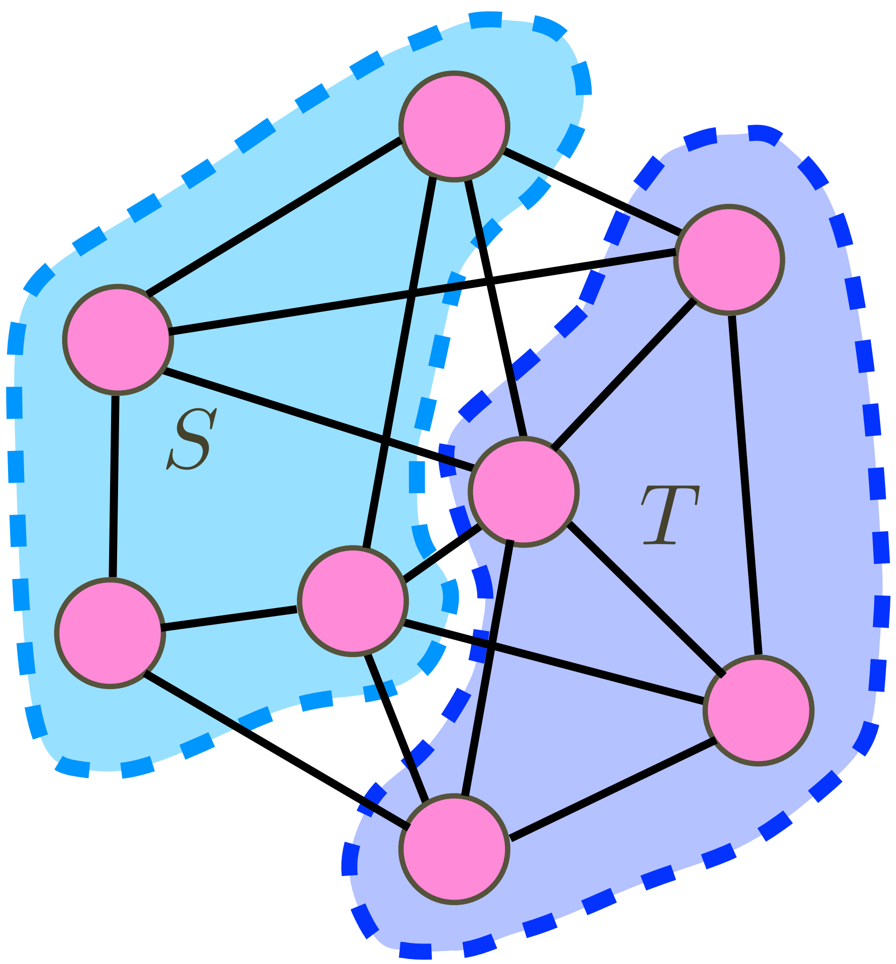
Maximum Cut

(*Approximation Algorithms*)

Max-Cut

- Undirected graph $G(V, E)$
- Bi-partition of V into nonempty S and T
- Find a cut $E(S, T)$ of largest size
- **NP-hard:**
    - one of Karp's 21 NP-complete problems
- Approximation algorithms?

$$E(S,T) := \{uv \in E \mid u \in S, v \in T\}$$



Greedy *Heuristics*

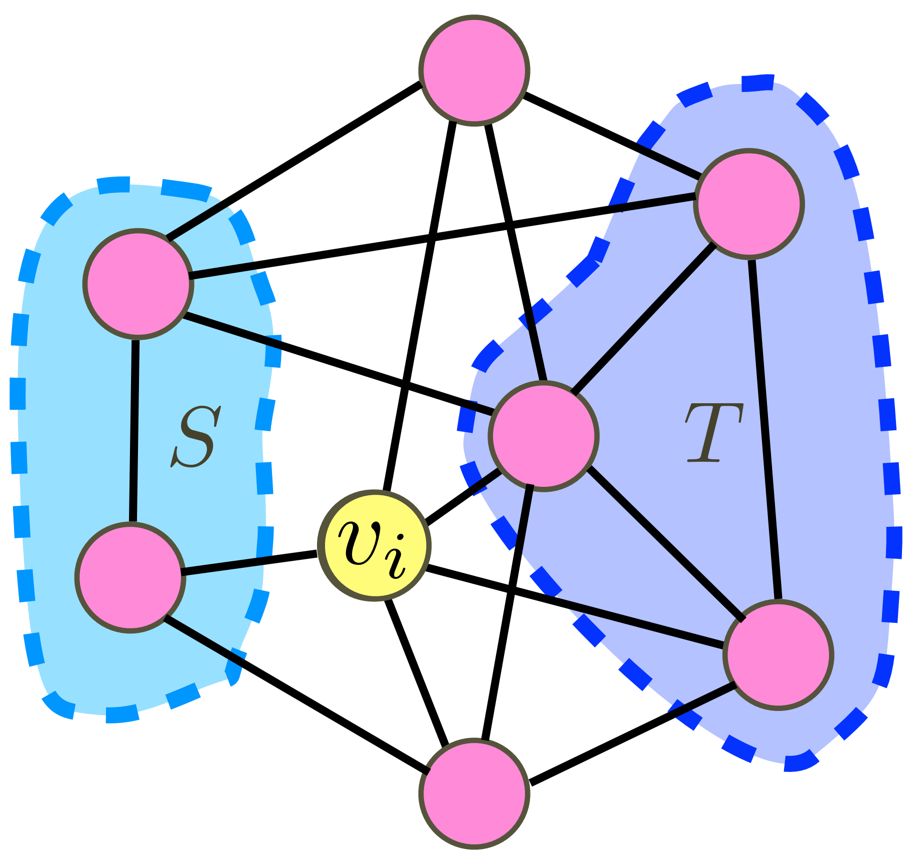
Greedy Cut:

initially, $S = T = \varnothing$;

for $i = 1, 2, \ldots, n$:

v_i joins one of S, T

to maximize current $E(S, T)$;



$$E(S, T) = \{uv \in E \mid u \in S, v \in T\}$$

Greedy *Heuristics*

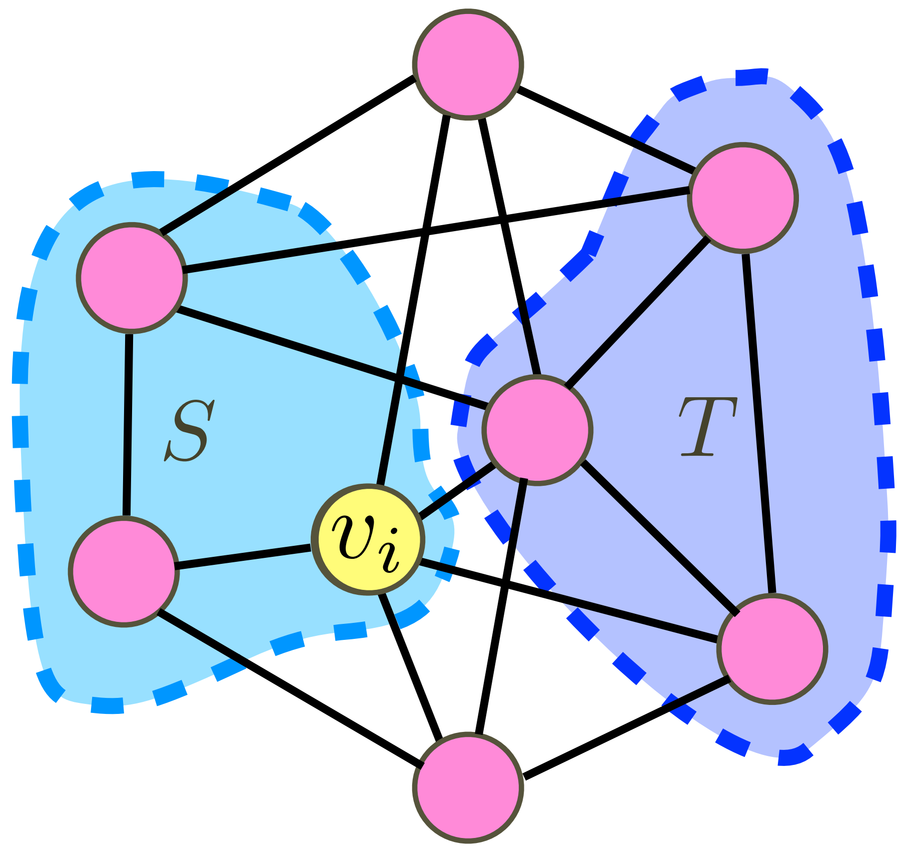
Greedy Cut:

initially, $S = T = \varnothing$;

for $i = 1,2,\ldots,n$:

  v_i joins one of S, T

  to maximize current $E(S, T)$;



$$E(S, T) = \{uv \in E \mid u \in S, v \in T\}$$

Approximation Ratio

Algorithm $\mathscr{A}$:

Greedy Cut:

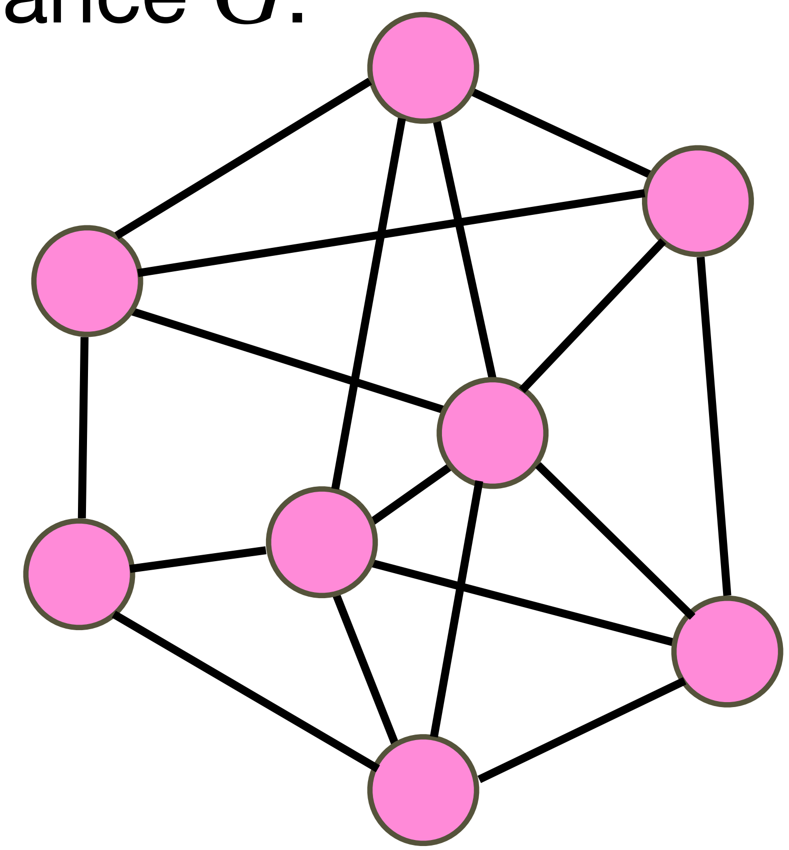
initially, $S = T = \emptyset$;

for $i = 1,2,\ldots,n$:

v_i joins one of S, T

to maximize current $E(S,T)$;

Instance G:



OPT_G: value of max-cut in G

SOL_G: value of the cut returned by $\mathscr{A}$ on G

Algorithm $\mathscr{A}$ has approximation ratio α if

$$\forall \text{ instance } G, \quad \frac{SOL_G}{OPT_G} \geq \alpha$$

Approximation Algorithm

Greedy Cut:

initially, $S = T = \emptyset$;

for $i = 1,2,\ldots,n$:

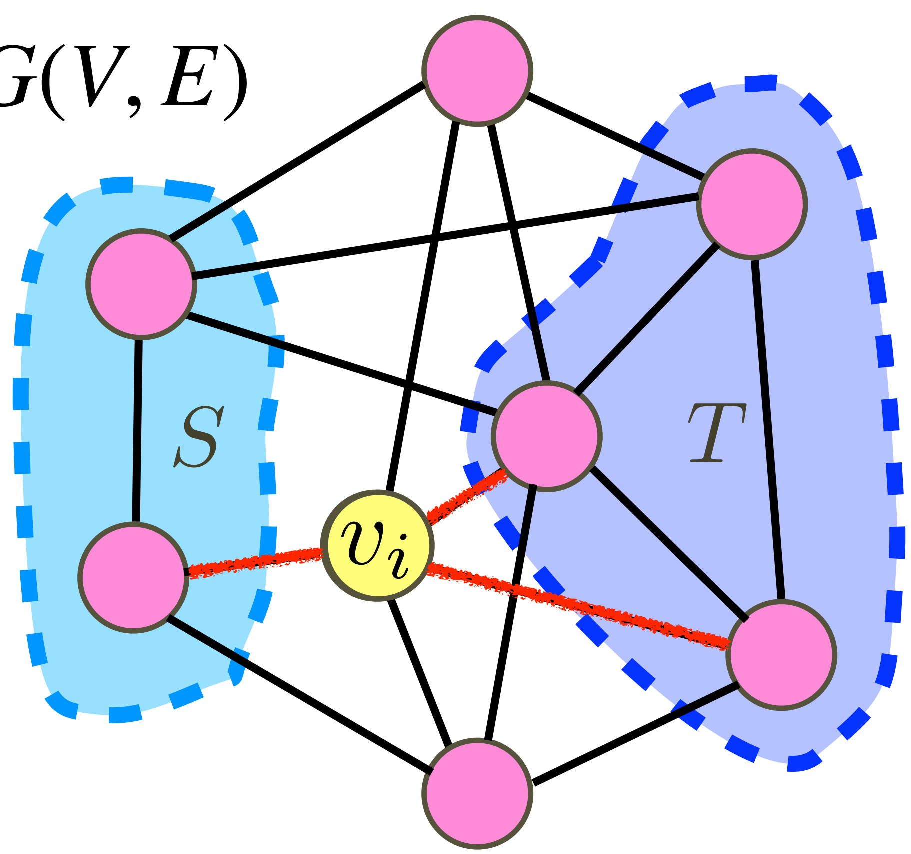
v_i joins one of S, T

to maximize current $E(S,T)$;

(S_i, T_i):

current (S,T) in the beginning of i-th iteration

$G(V,E)$



$E(S,T) = \{uv \in E \mid u \in S, v \in T\}$

$$\frac{SOL_G}{OPT_G} \geq \frac{SOL_G}{|E|} \geq \frac{1}{2}$$

$\forall v_i$, $\geq 1/2$ of $|E(S_i, v_i)| + |E(T_i, v_i)|$ contributes to SOL_G

$$|E| = \sum_{i=1}^{n} \left(|E(S_i, v_i)| + |E(T_i, v_i)| \right)$$

Approximation Algorithm

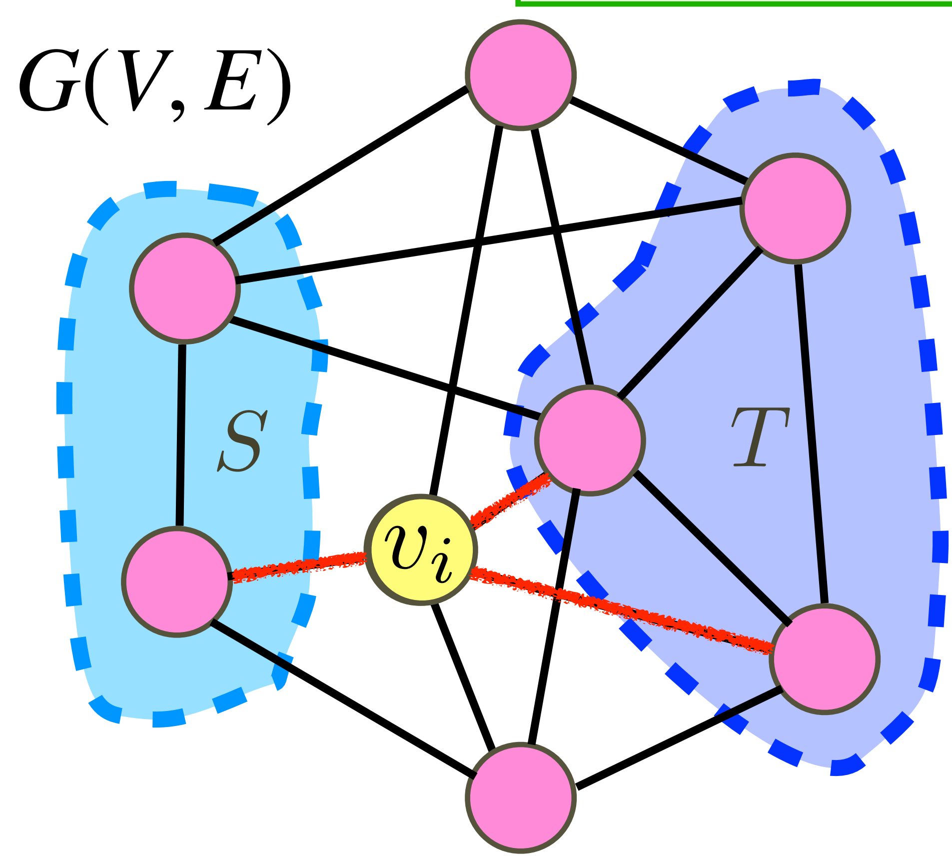
Greedy Cut:

initially, $S = T = \varnothing$;

for $i = 1,2,\ldots,n$:

    v_i joins one of S, T

    to maximize current $E(S, T)$;



$E(S, T) = \{uv \in E \mid u \in S, v \in T\}$

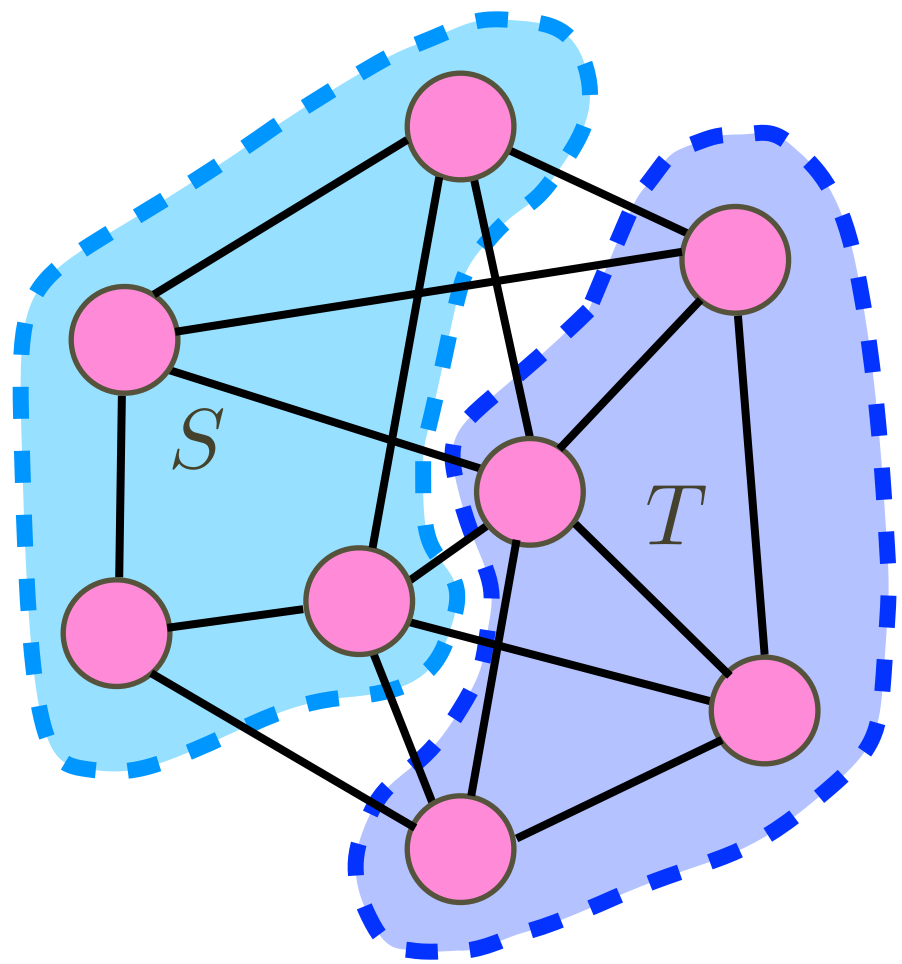
$$\frac{SOL_G}{OPT_G} \geq \frac{SOL_G}{|E|} \geq \frac{1}{2}$$

- **Approximation ratio:** $1/2$
- **Time cost:** $O(m)$

Max-Cut

- Find a cut $E(S, T)$ of largest size
- **NP-hard:**
  - one of Karp's 21 NP-complete problems
- Greedy algorithm: 0.5-approximation

$$E(S, T) = \{uv \in E \mid u \in S, v \in T\}$$

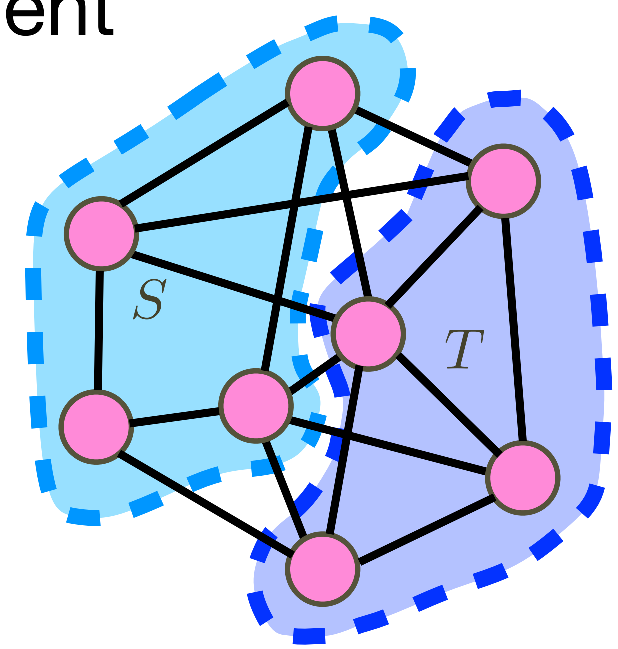


Random Cut

Theorem: For uniform random cut $E(S, T)$ in graph $G(V, E)$,

$$\mathbf{E}[\,|E(S,T)|\,] = \frac{|E|}{2}.$$

- $\forall v \in V$: let $X_v \in \{0,1\}$ be uniform & independent



$$X_v = 0 \implies v \text{ joins } S$$
$$X_v = 1 \implies v \text{ joins } T$$

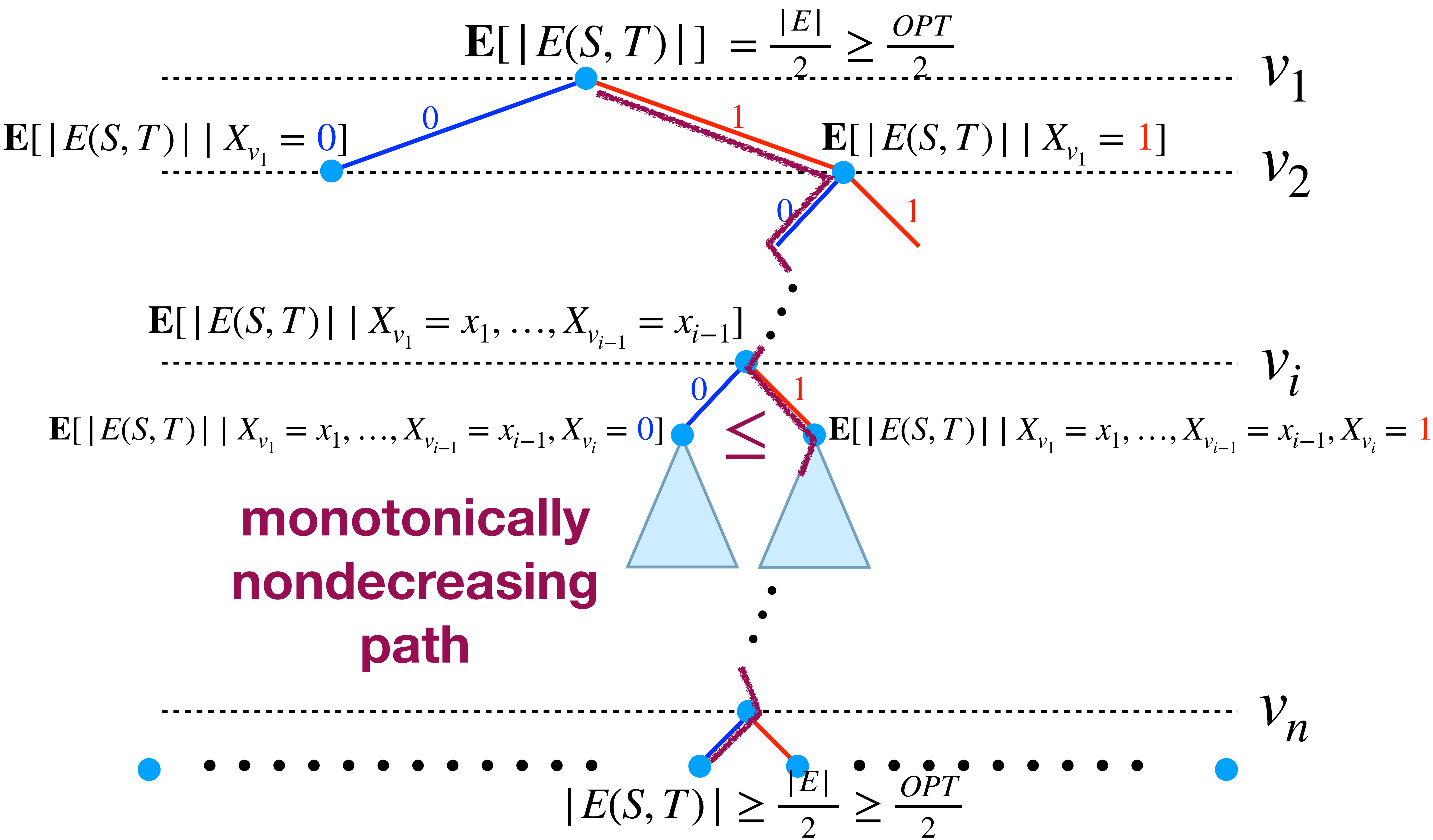
$$|E(S,T)| = \sum_{uv \in E} \boxed{I[X_u \neq X_v]}$$

indicator of $X_u \neq X_v$

- Linearity of expectation:

$$\mathbf{E}[\,|E(S,T)|\,] = \sum_{uv \in E} \Pr[X_u \neq X_v] = \frac{|E|}{2} \geq \frac{OPT}{2}$$

De-randomization (By Conditional Expectation)



All 2^n possible bi-partitions (S,T) of V.

De-randomization (By Conditional Expectation)

Greedy Cut:

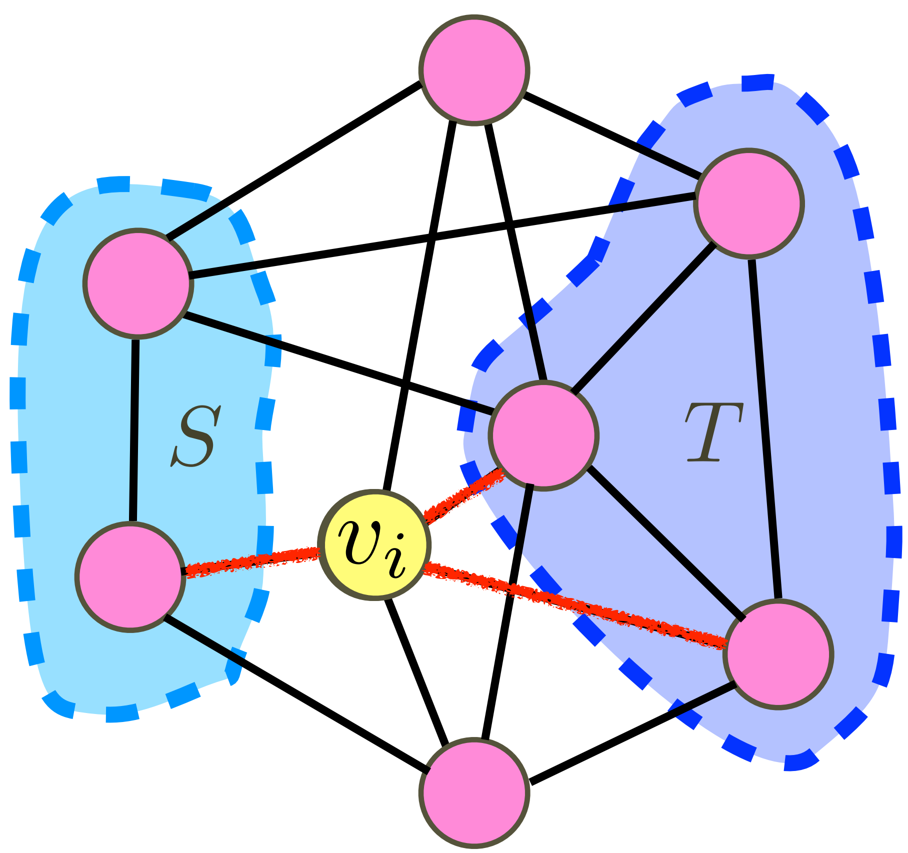
initially, $S = T = \varnothing$;

for $i = 1,2,\ldots,n$:

v_i joins one of S, T

to maximize current $E(S, T)$;

with bigger expected cut conditional on current (S, T)



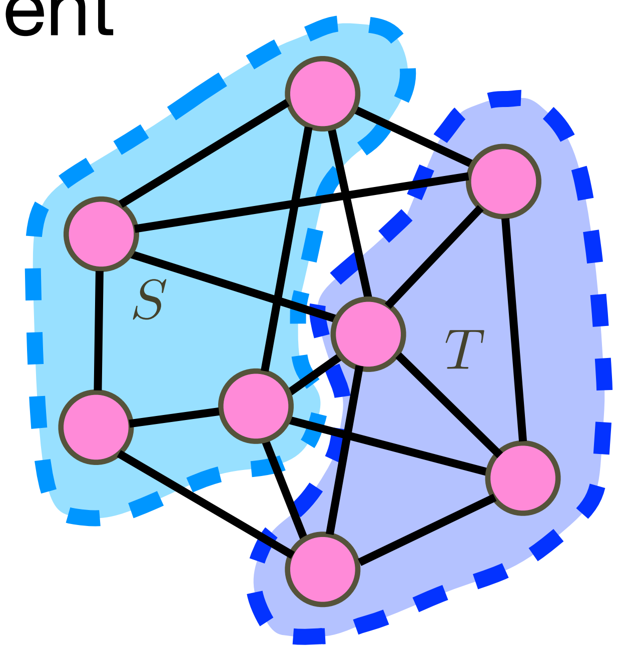
Random Cut

Theorem: For uniform random cut $E(S,T)$ in graph $G(V,E)$,

$$\mathbf{E}[\,|E(S,T)|\,] = \frac{|E|}{2}.$$

- $\forall v \in V$: let $X_v \in \{0,1\}$ be uniform & independent

$$X_v = 0 \implies v \text{ joins } S$$
$$X_v = 1 \implies v \text{ joins } T$$



$$|E(S,T)| = \sum_{uv \in E} \boxed{I[X_u \neq X_v]}$$

indicator of $X_u \neq X_v$

- **Linearity of expectation**:

$$\mathbf{E}[\,|E(S,T)|\,] = \sum_{uv \in E} \Pr[X_u \neq X_v] = \frac{|E|}{2} \geq \frac{OPT}{2}$$

Holds for pairwise independent X_v's!

Mutual Independence

Definition (mutual independence):

Events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n$ are mutually independent if for any subset $I \subseteq \{1, \ldots, n\}$,

$$\Pr\left[\bigwedge_{i \in I} \mathcal{E}_i\right] = \prod_{i \in I} \Pr\left[\mathcal{E}_i\right]$$

Definition (mutual independence of random variables):

Random variables $X_1, X_2, \ldots, X_n$ are mutually independent if for any subset $I \subseteq \{1, \ldots, n\}$ and any values x_i, where $i \in I$,

$$\Pr\left[\bigwedge_{i \in I} (X_i = x_i)\right] = \prod_{i \in I} \Pr\left[X_i = x_i\right]$$

k-wise Independence

Definition (k-wise independence):

Events $\mathscr{E}_1, \mathscr{E}_2, \ldots, \mathscr{E}_n$ are mutually independent if for any subset $I \subseteq \{1,\ldots,n\}$ of size at most k,

$$\Pr\left[\bigwedge_{i\in I} \mathscr{E}_i\right] = \prod_{i\in I} \Pr\left[\mathscr{E}_i\right]$$

Definition (k-wise independence of random variables):

Random variables $X_1, X_2, \ldots, X_n$ are mutually independent if for any subset $I \subseteq \{1,\ldots,n\}$ of size at most k, and any values x_i, where $i \in I$,

$$\Pr\left[\bigwedge_{i\in I}(X_i = x_i)\right] = \prod_{i\in I} \Pr\left[X_i = x_i\right]$$

Pairwise: 2-wise

Pairwise Independent Bits

- Mutually independent uniform random bits: (random source)

$$b_1, \ldots, b_l \in \{0,1\}$$

a	b	a⊕b
0	0	0
0	1	1
1	0	1
1	1	0

Enumerate all nonempty subsets:

$$S_1, \ldots, S_{2^l-1} \subseteq \{1, \ldots, l\}$$

Parity Construction:

$$X_j = \bigoplus_{i \in S_j} b_i$$

Theorem:

$X_1, \ldots, X_{2^l-1}$ are pairwise independent uniform random bits.

- Pairwise independent uniform random bits: (for $n \gg l$)

$$X_1, \ldots, X_n \in \{0,1\}$$

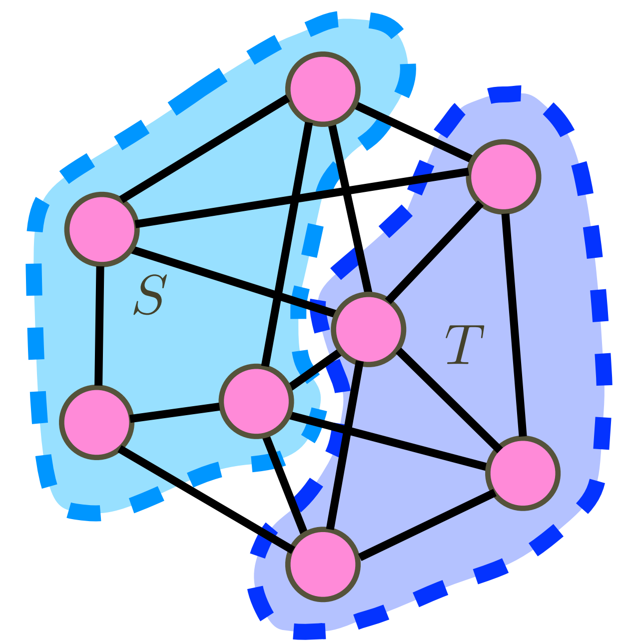
De-randomization (By Pairwise Independence)

- Pairwise independent uniform $X_v \in \{0,1\}$ for all $v \in V$

$$X_v = 0 \implies v \text{ joins } S$$
$$X_v = 1 \implies v \text{ joins } T$$

$$|E(S,T)| = \sum_{uv \in E} \boxed{I[X_u \neq X_v]}$$

indicator of $X_u \neq X_v$



- Linearity of expectation:

$$\mathbf{E}[\,|E(S,T)|\,] = \sum_{uv \in E} \Pr[X_u \neq X_v] = \frac{|E|}{2} \geq \frac{OPT}{2}$$

Theorem: For (S,T) generated from pairwise independent uniform random bits, $\mathbf{E}[\,|E(S,T)|\,] = \frac{|E|}{2}$.

De-randomization (By Pairwise Independence)

- Let $b_1, \ldots, b_{\lceil \log_2(n+1) \rceil} \in \{0,1\}$ be mutually independent uniform bits.

$\Downarrow$ parity construction

- Pairwise independent uniform $X_v \in \{0,1\}$ for all $v \in V$

$$X_v = 0 \implies v \text{ joins } S$$
$$X_v = 1 \implies v \text{ joins } T$$

> **Theorem:** For (S, T) generated from pairwise independent uniform random bits, $\mathbf{E}[\,|E(S,T)|\,] = \frac{|E|}{2}$.

- Enumerate all $b_1, \ldots, b_{\lceil \log_2(n+1) \rceil} \in \{0,1\}$: (only $O(n)$ in total)
  - There must exist an assignment of $b_1, \ldots, b_{\lceil \log_2(n+1) \rceil} \in \{0,1\}$ which corresponds to a cut with $|E(S,T)| \geq \frac{|E|}{2}$.

De-randomization (By Pairwise Independence)

Parity Search:

for all $\boldsymbol{b} \in \{0,1\}^{\lceil \log_2(n+1) \rceil}$:

initialize $S_{\boldsymbol{b}} = T_{\boldsymbol{b}} = \emptyset$;

for $i = 1,2,\ldots,n$:

if $\bigoplus_{j:\lfloor i/2^j \rfloor \bmod 2 = 1} b_j = 1$ then v_i joins $S_{\boldsymbol{b}}$;

else v_i joins $T_{\boldsymbol{b}}$;

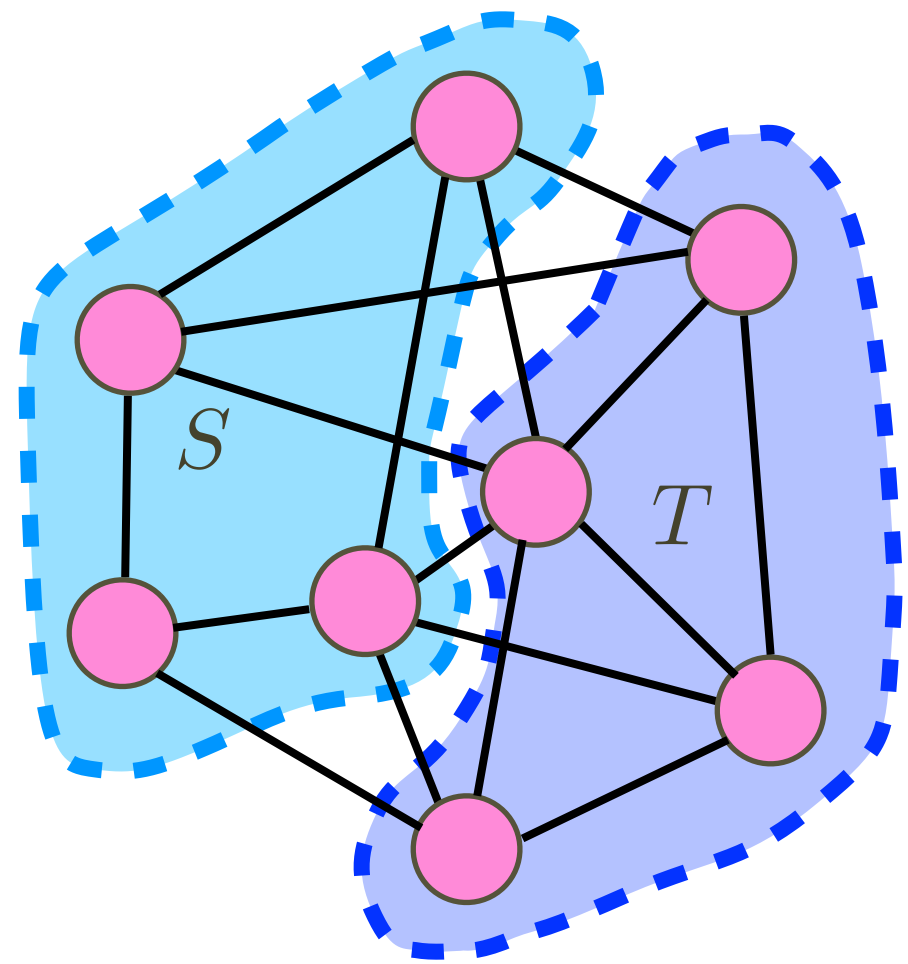
return the $(S_{\boldsymbol{b}}, T_{\boldsymbol{b}})$ with the largest $|E(S_{\boldsymbol{b}}, T_{\boldsymbol{b}})|$;

- **Approximation ratio:** $1/2$
- **Time cost:** $O(n^2 \log n)$

Max-Cut

- Find a cut $E(S,T)$ of largest size
- **NP-hard:**
  - one of Karp's 21 NP-complete problems
- Greedy algorithm: 0.5-approximation
- Best known approx. ratio for poly-time algorithm: 0.878~
- Unique games conjecture $\Longrightarrow$ computationally hard to do better

$$E(S,T) = \{uv \in E \mid u \in S, v \in T\}$$



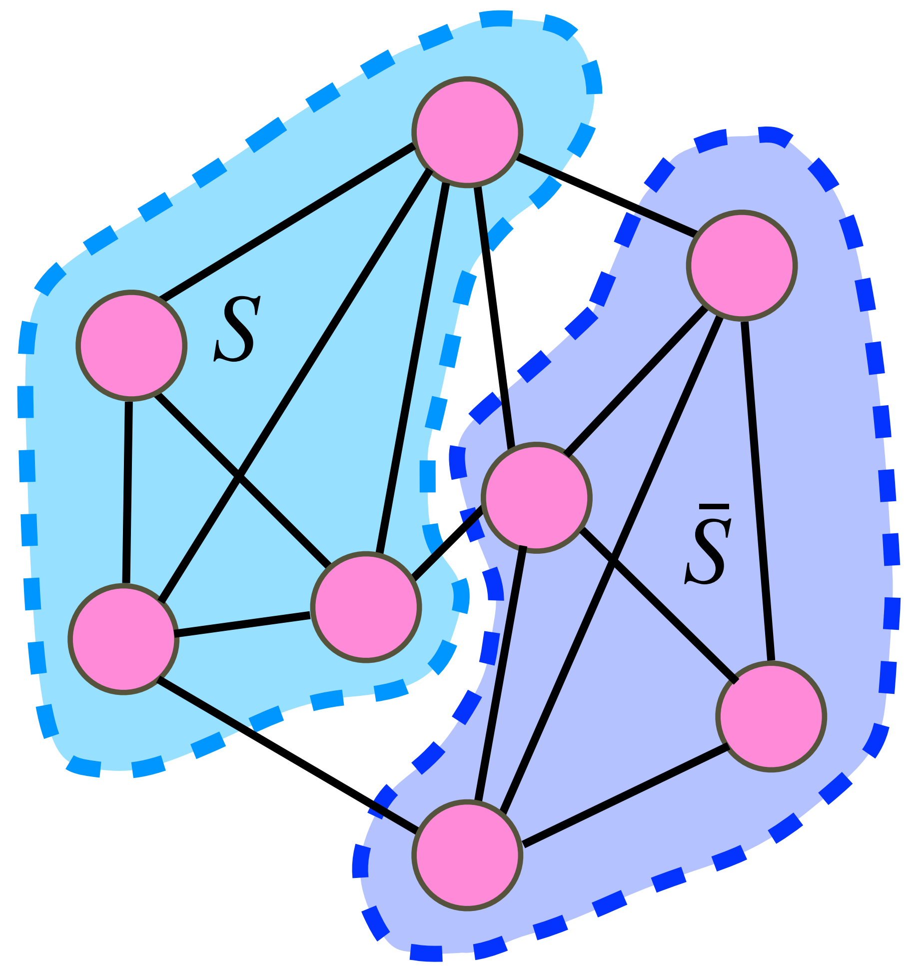
Sparsest Cut
(*Spectral Algorithms*)

Sparsest Cut

$$E(S, T) := \{uv \in E \mid u \in S, v \in T\}$$

- Undirected graph $G(V, E)$
- Find an $S \subseteq V$ with smallest

$$\frac{|E(S, \bar{S})|}{\min\{|S|, |\bar{S}|\}}$$



- Edge expansion:

$$h(G) := \min_{\substack{S \subseteq V \\ |S| \leq |V|/2}} \frac{|E(S, \bar{S})|}{|S|}$$

- **NP-hard** (and remains **NP-hard** for constant approximation assuming **Unique Games Conjecture**)

- **Applications:** community detection in social network, robust network design, data clustering, image segmentation, VLSI design, rapidly mixing random walks, …

Graphs as Matrices

- Undirected graph $G(V, E)$ has adjacency matrix $A \in \{0,1\}^{V \times V}$:

$$A_{u,v} = 1 \text{ iff } uv \in E$$

- A is a real symmetric matrix:
  - it has real **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$
  - the **eigenvectors** $v_1, v_2, \ldots, v_n$ form **orthonormal basis**

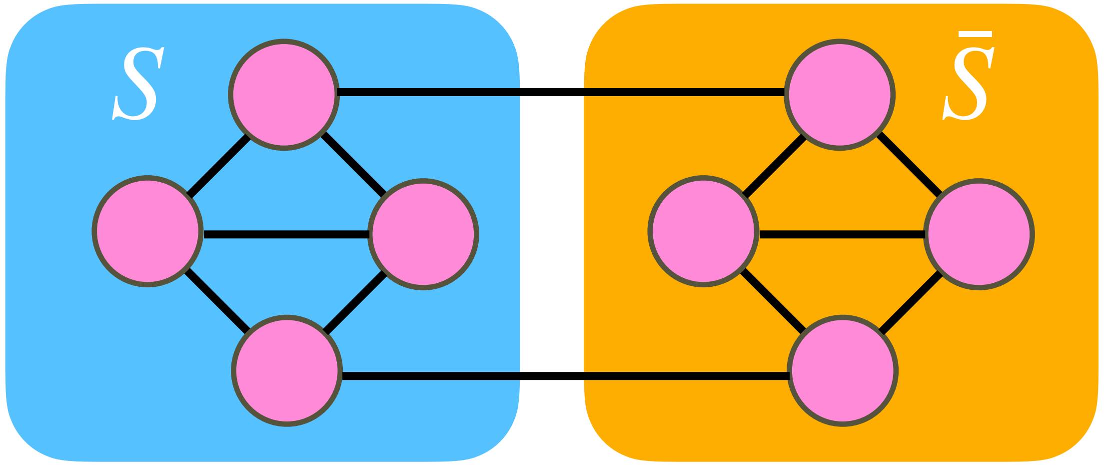
$$Av_i = \lambda_i v_i, \ \forall i,$$

$$v_i^\top v_j = \begin{cases} 1, \text{ if } i = j, \\ 0, \text{ otherwise} \end{cases}$$

Spectral Partitioning

- Undirected d-regular graph $G(V, E)$
- Conductance (normalized expansion):

$$\varphi(G) := \frac{h(G)}{d} = \min_{\substack{S \subseteq V \\ |S| \le |V|/2}} \frac{|E(S, \bar{S})|}{d|S|}$$



$$\varphi(S) := \frac{|E(S, \bar{S})|}{d|S|}$$

Spectral partitioning algorithm:

- Compute the **second largest** eigenvalue λ_2 of the adjacency matrix A and its corresponding eigenvector $\boldsymbol{x} \in \mathbb{R}^n$
- Sort the vertices $V = \{u_1, u_2 \ldots, u_n\}$ so that $x(u_1) \ge x(u_2) \ge \ldots \ge x(u_n)$
- Let $S_i := \begin{cases} \{u_1, \ldots, u_i\} & \text{if } i \le n/2 \\ V \backslash \{u_1, \ldots, u_i\} & \text{otherwise} \end{cases}$, and output $S_i = \arg\min_{1 \le i \le n} \{\varphi(S_i)\}$

Theorem: $\exists i, \varphi(S_i) \le 2\sqrt{\varphi(G)}$

Generalizable to irregular graphs

Spectral Graph Theory

- Undirected d-regular graph $G(V, E)$
- Eigenvalues of adjacency matrix: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$
- Conductance: $\varphi(G) := \min_{\substack{S \subseteq V \\ |S| \leq |V|/2}} \frac{|E(S, \bar{S})|}{d|S|}$

Theorem:

- $|\lambda_i| \leq d$
- $\lambda_1 = d$ and has eigenvector $\vec{1}$
- $\lambda_1 > \lambda_2$ iff G is connected

Theorem (Cheeger's Inequality):

$$\frac{1}{2}(1 - \lambda_2/d) \leq \varphi(G) \leq \sqrt{2(1 - \lambda_2/d)}$$

Application: Image Segmentation

One often wants to partition an image into consecutive ***regions***

- Pixels within a region are **fairly similar** to each other
- Neighboring regions are **fairly different** from each other

Source: https://cs.brown.edu/people/pfelzens/segment/

Sparsest cut based image segmentation: see Shi-Malik 2000

Application: Graph Visualization

How to *visualize* or *plot* a graph?

- The second eigenvector may sparsely partition the graph.
- Goal: "communities" are clustered together, and different "communities" are farther apart

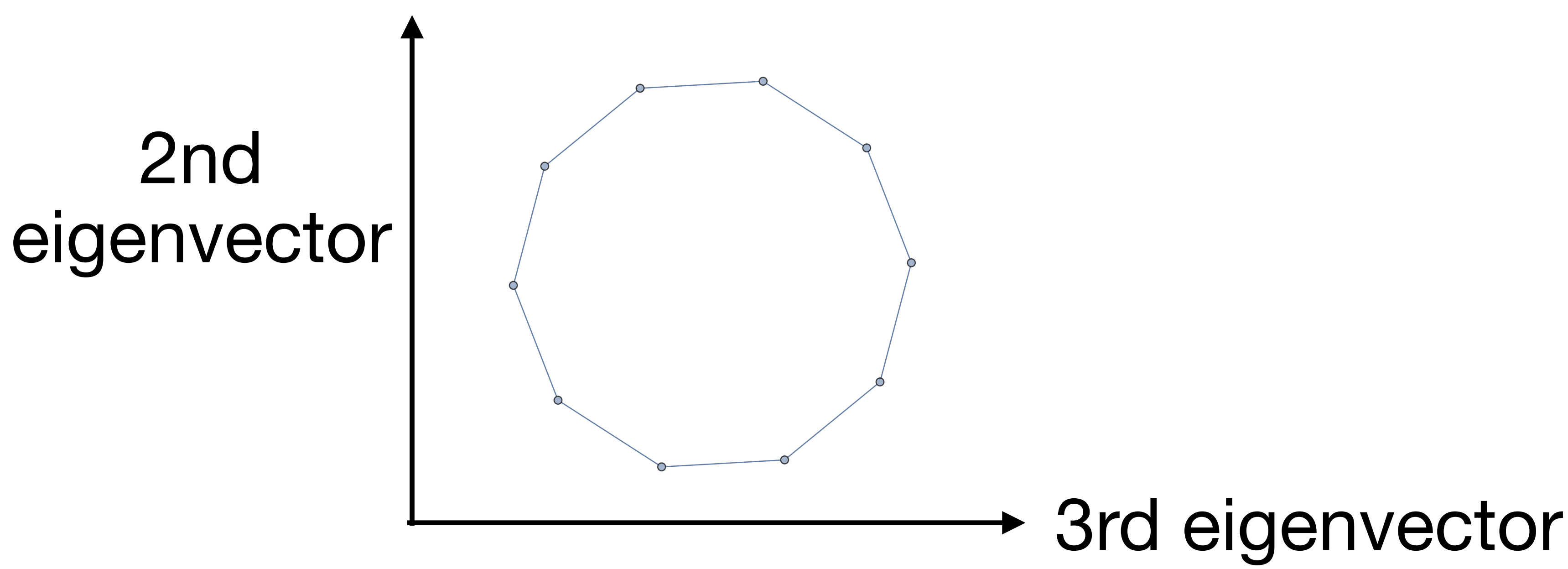
Spectral visualization algorithm:

- Compute the **second and third** eigenvectors $v_2, v_3 \in \mathbb{R}^n$ of the adjacency matrix A
- Plot the graph on a 2D plane, where vertex i is drawn at coordinate $(v_2(i), v_3(i))$

Application: Graph Visualization

Spectral visualization algorithm:

- Compute the **second and third** eigenvectors $v_2, v_3 \in \mathbb{R}^n$ of the adjacency matrix A
- Plot the graph on a 2D plane, where vertex i is drawn at coordinate $(v_2(i), v_3(i))$



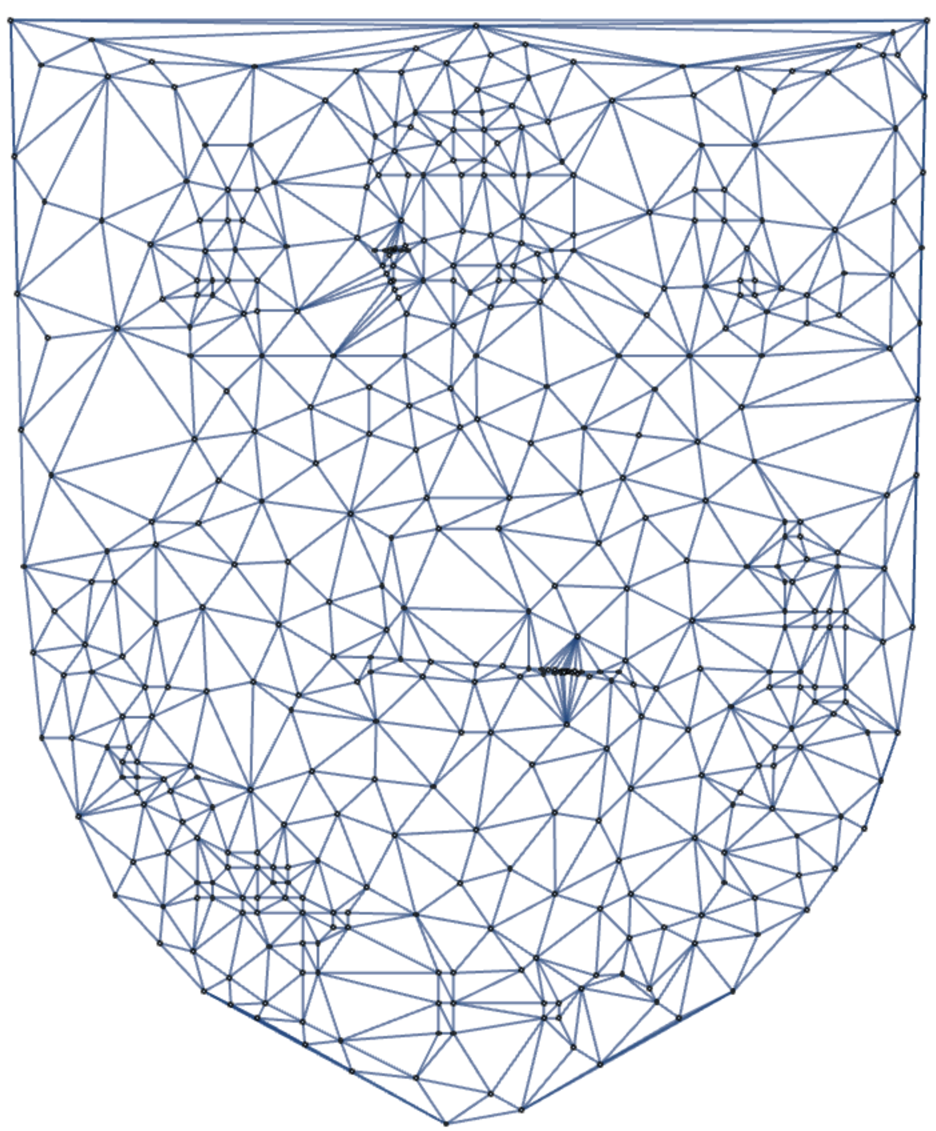
In **Mathematica:**

GraphPlot[CycleGraph[10], GraphLayout -> "SpectralEmbedding"]

Application: Graph Visualization

In[1]:=
```
Rasterize[ , RasterSize → 100] //
    ColorNegate // ImageMesh // DiscretizeRegion //
  MeshCoordinates // DelaunayMesh //
 MeshConnectivityGraph
```

Out[1]=



```
SpectralPlot[g_] :=
 AdjacencyGraph[g // AdjacencyMatrix,
  VertexCoordinates →
   (Transpose[
      Take[
       SortBy[
        Eigensystem[
          DiagonalMatrix[Total[g // AdjacencyMatrix]] -
           (g // AdjacencyMatrix // Normal // N)] //
         Transpose, #[[1]] &],
       {2, 3}]][[2]] // Transpose),
  VertexSize → Tiny]

SpectralPlot[%1]
```