(Even More) Recent Advances

In

Cut Query

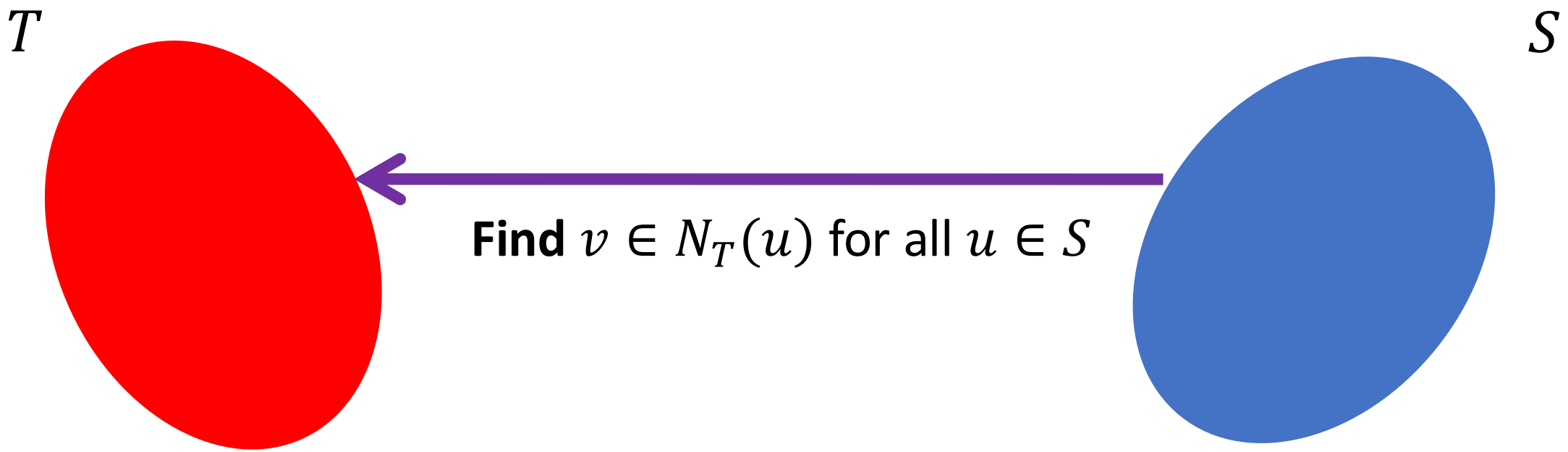
Sagnik Mukhopadhyay

UNIVERSITY OF BIRMINGHAM

Connectivity

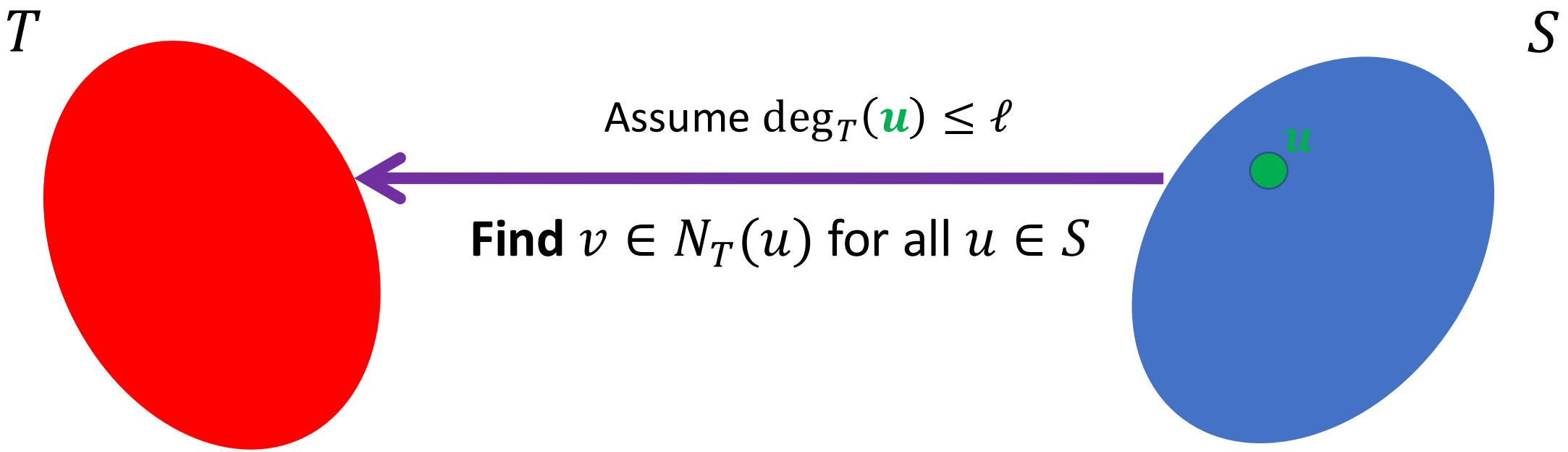
The (Easy) Problem

- Each $u \in S$ **finds** a neighbor $v \in T$



The (Easy) Problem

- Each $u \in S$ **finds** a neighbor $v \in T$

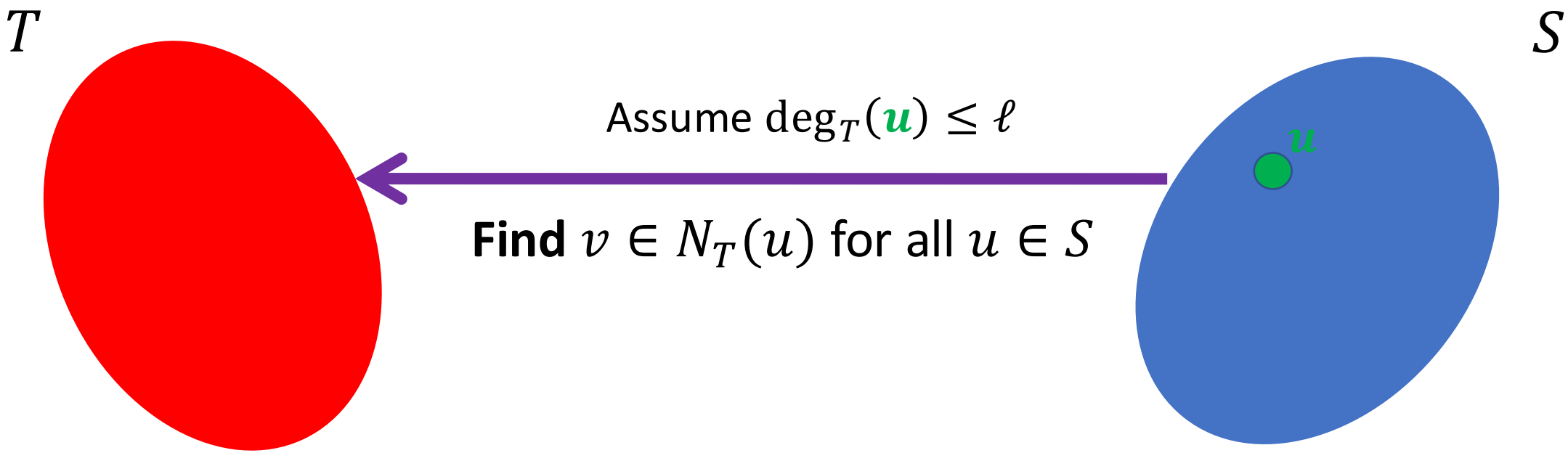


- $|S| \times |T|$ Boolean matrix $M, M[u,v] = 1 \leftrightarrow (u,v) \in E$
- M has row sparsity ℓ
- Goal, learn M!

Seperating Matrices: Learn $M \in \{0,1\}^{n \times n}$ of row sparsity ℓ with $O(\ell n)$ cut queries

The (Easy) Problem

- Each $u \in S$ **finds** a neighbor $v \in T$

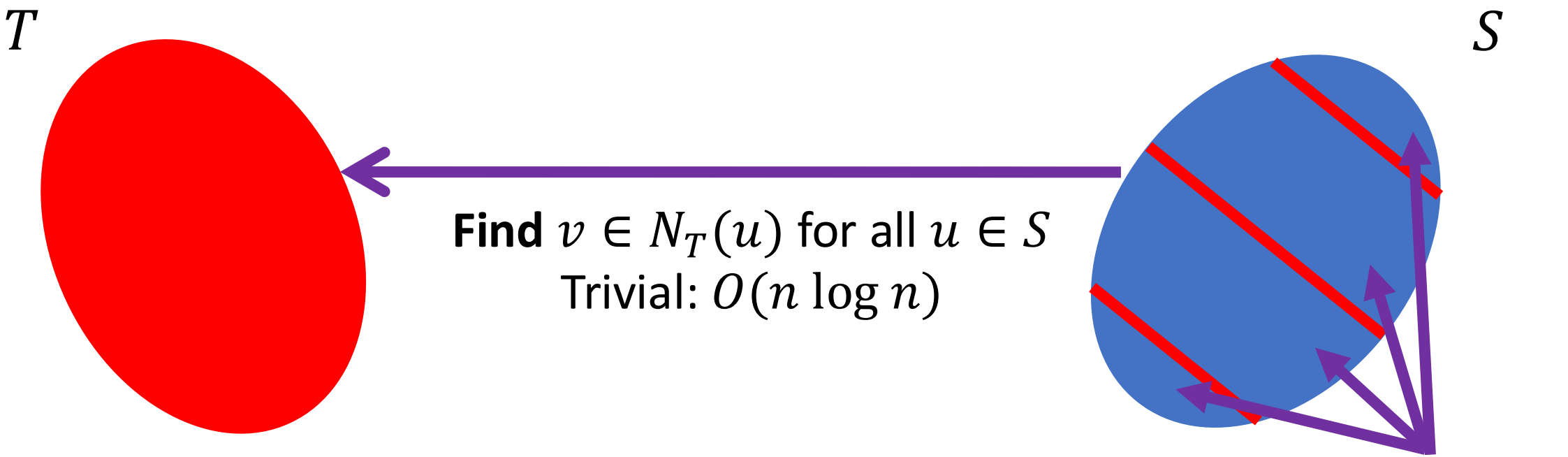


Seperating Matrices: Learn $M \in \{0,1\}^{n \times n}$ of row sparsity ℓ with $O(\ell n)$ cut queries

- **Trivial:** $O(\ell n \log n)$ cut queries using binary search.
- A cut query provides $\Omega(\log n)$ bit of information.
- Can be used to shave the log factor.

The (Easy) Problem

- Each $u \in S$ **finds** a neighbor $v \in T$

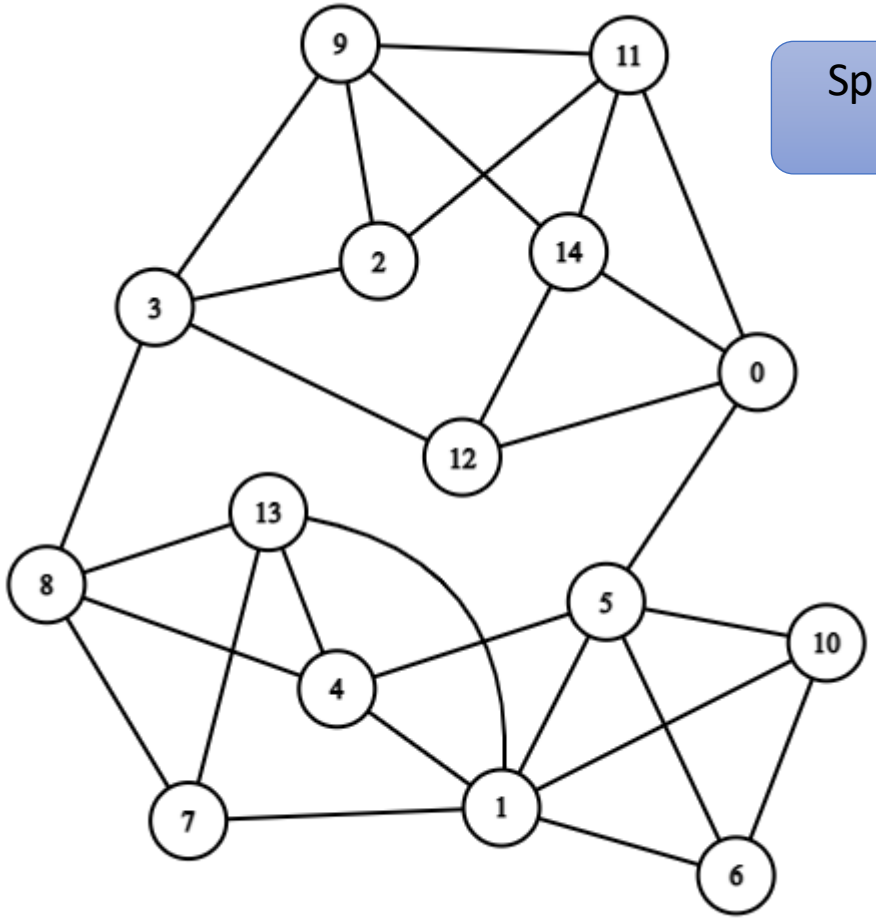


- **Learn $M \in \{0,1\}^{n \times n}$ of row sparsity ℓ with $O(\ell n)$ cut queries**
- **Sampling Lemma**: Solve **The (Easy) Problem** in $O(|S|)$ cut queries, **in expectation**
- Immediate corollary: **Connectivity** in **expected $O(n)$** cut queries!

Follow Troy's talk for more!

Minimum Cut on Simple Graphs

Background: Basic Algorithm



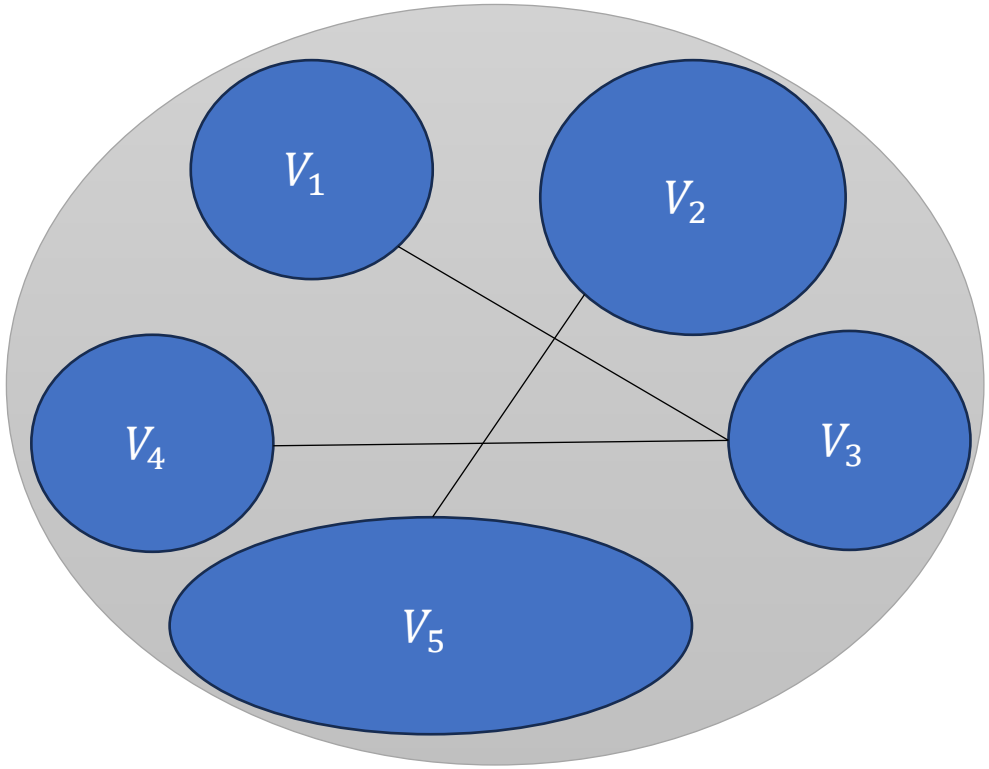
Spanning tree: $\tilde{O}(n)$ queries

- Pack δ spanning trees.
  - Each tree must cross every cut at least once.
  - $\delta \geq \lambda$.
- Complexity: $\tilde{O}(n\delta)$ $\rightarrow O(n\delta)$.

Separating matrices

- Can we do any better?

Background: Min-cut Preserving Clustering [Kawarabayashi, Thorup 2015]



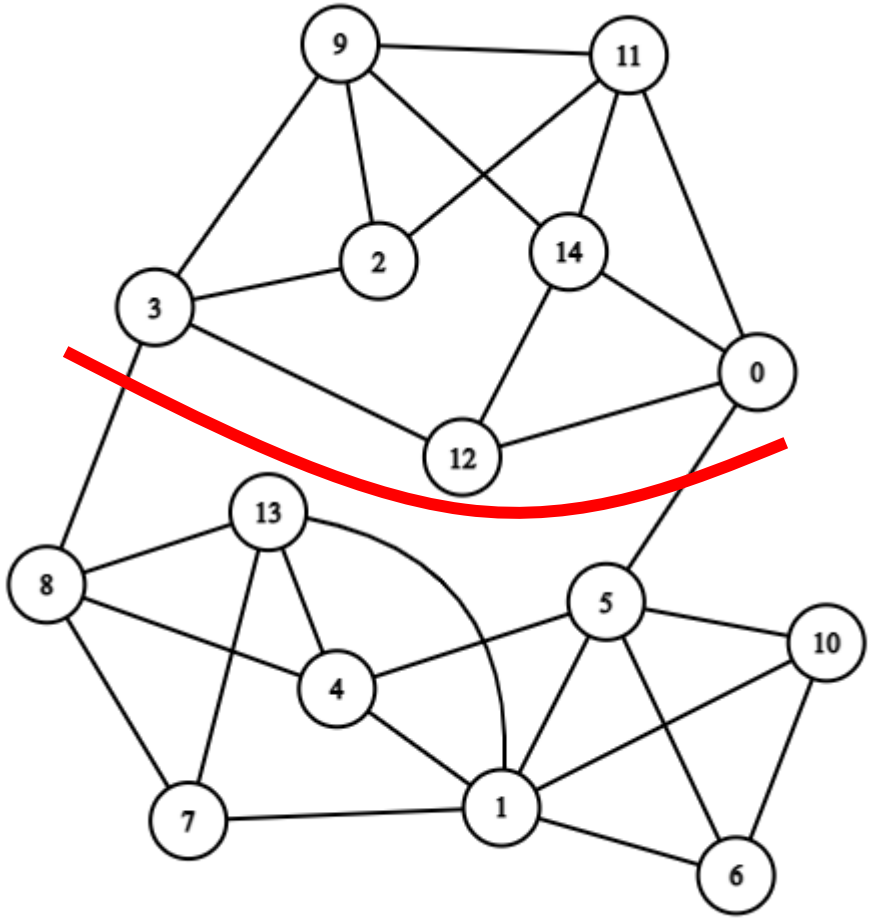
$G = (V, E)$

- Simple graph G with min deg δ.
- Contract: $G \rightarrow G'$ such that
  - G' has $\tilde{O}\left(\frac{n}{\delta}\right)$ vertices and $\tilde{O}(n)$ edges.
  - All non-trivial min-cuts are preserved.

Min-cut (G) = Min-cut (G')

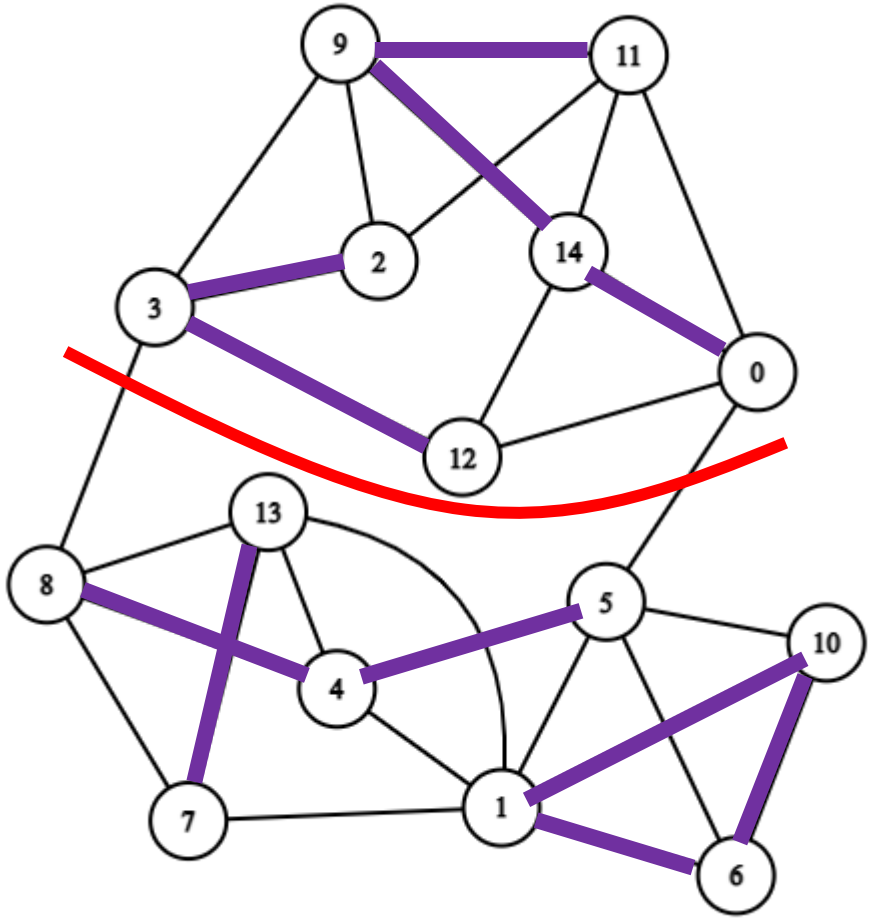
- Pack δ spanning trees in G'
  - Almost linear complexity

Background: One-out contraction



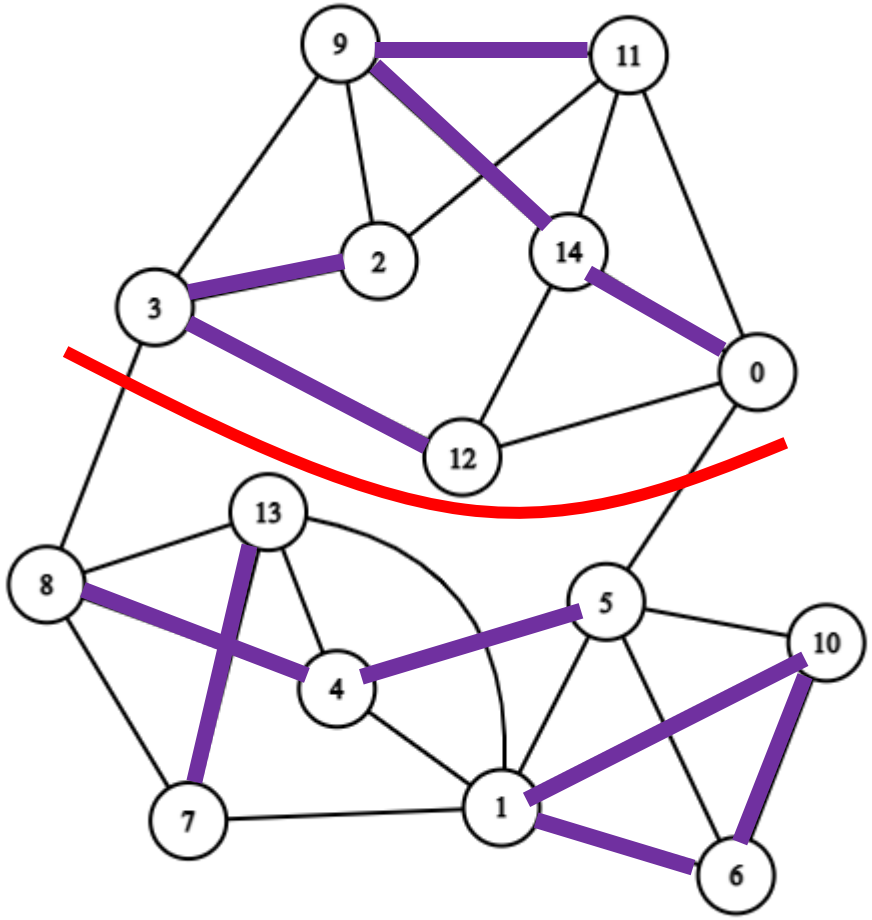
- Let C be some min cut. Every vertex v chooses uniformly random neighbor $u \in N(v)$.

Background: One-out contraction



- Let C be some min cut. Every vertex v chooses uniformly random neighbor $u \in N(v)$.
- **S- sampled edges.**
- $\Pr[S \cap C = \emptyset]$?
- $\Pr[S \cap C = \emptyset] \geq \frac{1}{2}^4 = \frac{1}{16}$
- Constant Prob!

Background: One-out contraction



- Let C be some min cut. Every vertex v chooses uniformly random neighbor $u \in N(v)$.
- ***S*- sampled edge**
- $\Pr[S \cap C = \emptyset] = \prod_{v \in N(C)} \left(1 - \frac{c(v)}{d(v)}\right)$

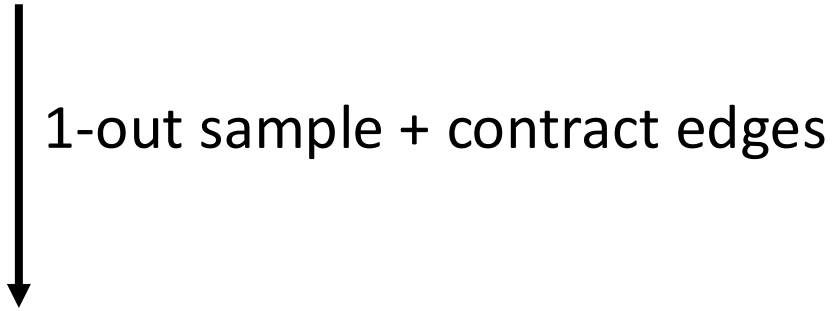
$\geq \frac{1}{16}$

$$\frac{c(v)}{d(v)} \leq 1/2 \text{ for every } v \in N(C)$$

$$\sum_{v \in N(C)} \frac{c(v)}{d(v)} \leq 2\frac{|C|}{\delta(G)} \leq 2.$$

Background: One-out contraction

- $G = (V, E)$



$G' = (V', E')$

- With constant probability, $\lambda(G) = \lambda(G')$
- Solve on multigraph G'?
- One can show: Exists graphs s.t. $|V'| = \Theta\left(\frac{n}{\sqrt{\delta}}\right)$ w.h.p.

We want: $O\left(\frac{n}{\delta}\right)$

Background: **Two**-out contraction [Ghaffari, Nowicki, Thorup 2020]

- $G = (V, E)$

2-out sample + contract edges

$G' = (V', E')$

Room for improvement:
- Cluster diameter $\Theta(\log^2 n)$
- Analysis made simpler

2-out sample complexity: $O(n \log n)$.

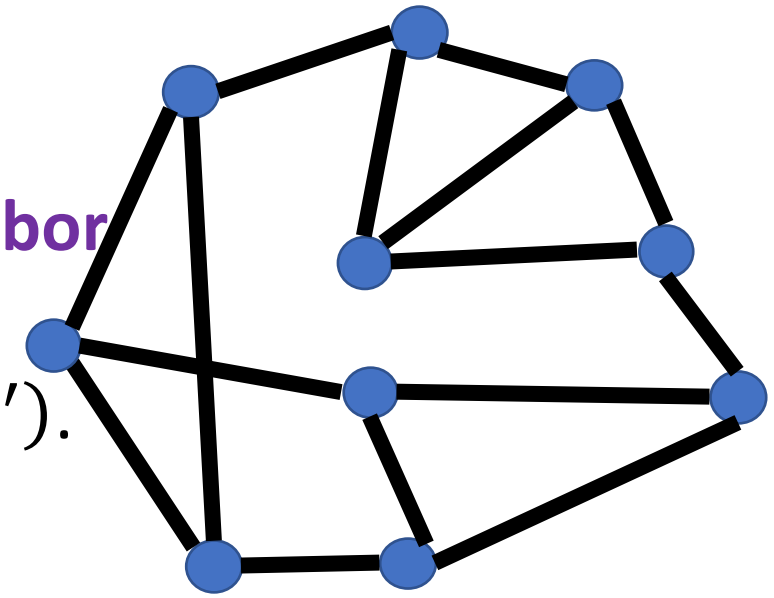
- With constant probability, $\lambda(G) = \lambda(G')$
- Solve on multigraph G'.
- Non-trivial analysis: $|V'| = O\left(\frac{n}{\delta}\right)$ w.h.p.

No logs!!

Complexity: $O\left(\frac{n}{\delta} . \delta\right) = O(n)$.

Star-Contraction

- Idea: Sample from a subset of neighbors
- **Construct set R with each $v \in R$ w.p. $p = \Theta\left(\frac{\log n}{\delta}\right)$**
- **Each $u \notin R$ independently samples neighbor $v \in N_R(u)$**
- Contract sampled edges S into $G' = (V', E')$.

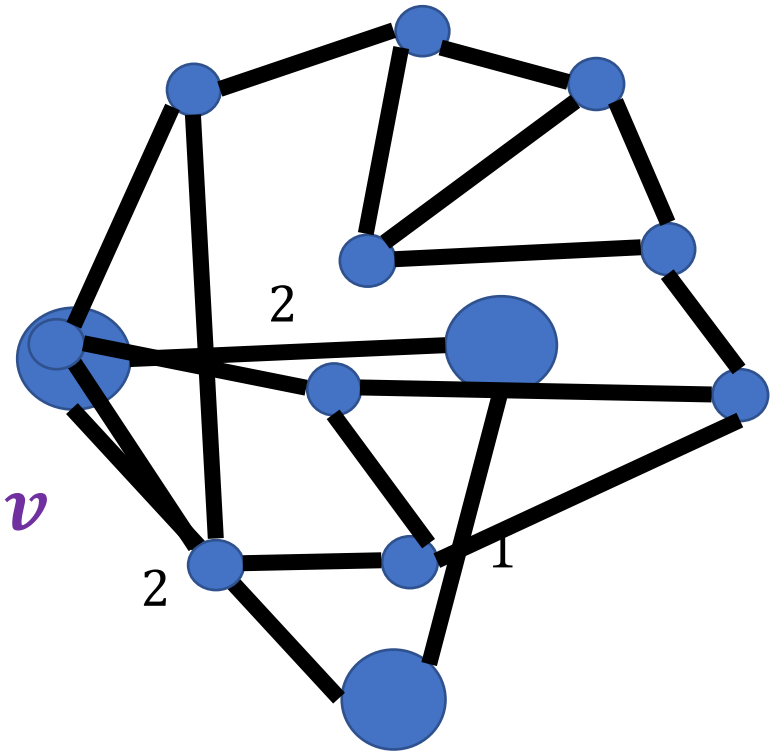


$$\mathbb{E}_R\left[\frac{c_R(v)}{d_R(v)} \,\middle|\, d_R(v) > 0\right] = \frac{c(v)}{d(v)} \ .$$

Star-Contraction

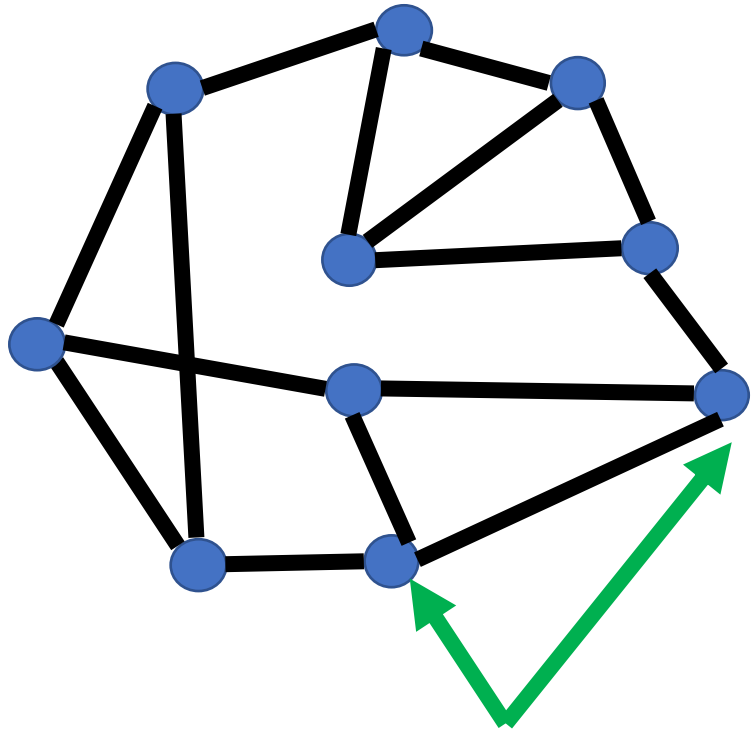
- Idea: Sample from a subset of neighbors
- **Construct set R with each $v \in R$ w.p. $p = \Theta\left(\frac{\log n}{\delta}\right)$**
- **Each $u \notin R$ independently samples neighbor $v \in N_R(u)$**
- Contract sampled edges S into $G' = (V', E')$.
- **Immediate**: $|V'| = O\left(\frac{n \log n}{\delta}\right)$ w.h.p.
- $\Pr[S \cap C = \emptyset] = \Omega(1)$.
- $\lambda($

Complexity: $O\left(\frac{n}{\delta} \log n \,.\, \delta\right) = O(n \log n)$.



Beyond $n \log n$ step 1: Refined star contraction

- $|R|$ and therefore $|V'|$ are too large
- Replace $p = \Theta\left(\frac{\log n}{\delta}\right)$ with $p = \Theta\left(\frac{\log \delta}{\delta}\right)$
- **Immediate**: $|R| = O\left(\frac{n \log \delta}{\delta}\right)$
- $|\boldsymbol{V^*}| = |\{\boldsymbol{v} \in \boldsymbol{V} \mid \boldsymbol{N(v)} \cap \boldsymbol{R} = \emptyset| = \boldsymbol{O\left(\frac{n}{\delta}\right)}$
- $V' = \boldsymbol{R} \cup \boldsymbol{V^*}$
- Each $u \notin V'$, independently samples neighbor $v \in N_R(u)$
- Contract into $G' = (V', E')$
- $O(n \log \delta) \Rightarrow O(n \log \log n)$ queries.

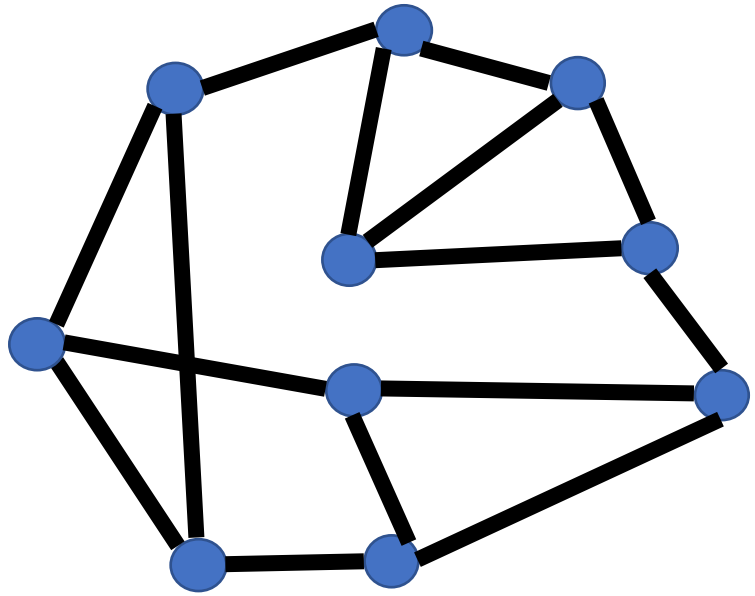


No neighbor in R

If $\delta \geq polylog(n)$, use
[Mukhopadhyay, Nanongkai 2020]

Beyond $n \log n$ step 1: Refined star contraction

- $|R|$ and therefore $|V'|$ are too large
- Replace $p = \Theta\left(\frac{\log n}{\delta}\right)$ with $p = \Theta\left(\frac{\log \delta}{\delta}\right)$
- **Immediate**: $|R| = O\left(\frac{n \log \delta}{\delta}\right)$
- $|\boldsymbol{V^*}| = |\{\boldsymbol{v} \in \boldsymbol{V} \mid \boldsymbol{N(v)} \cap \boldsymbol{R} = \emptyset| = \boldsymbol{O\left(\frac{n}{\delta}\right)}$
- $V' = \boldsymbol{R} \cup \boldsymbol{V^*}$
- Each $u \notin V'$, independently samples neighbor $v \in N_R(u)$
- Contract into $G' = (V', E')$
- $O(n \log \delta) \Rightarrow O(n \log \log n)$ queries.



How?

Separating Matrices

Trivial: $O(n \log n)$

If $\delta \geq polylog(n)$, use
[Mukhopadhyay, Nanongkai 2020]

The Problem

- Each $u \notin V'$ independently samples neighbor $v \in N_R(u)$

$$\Pr[S \cap C = \emptyset] = \prod_{v \in N(C)} \left(1 - \frac{c(v)}{d(v)}\right) > \frac{1}{16}$$ as long as

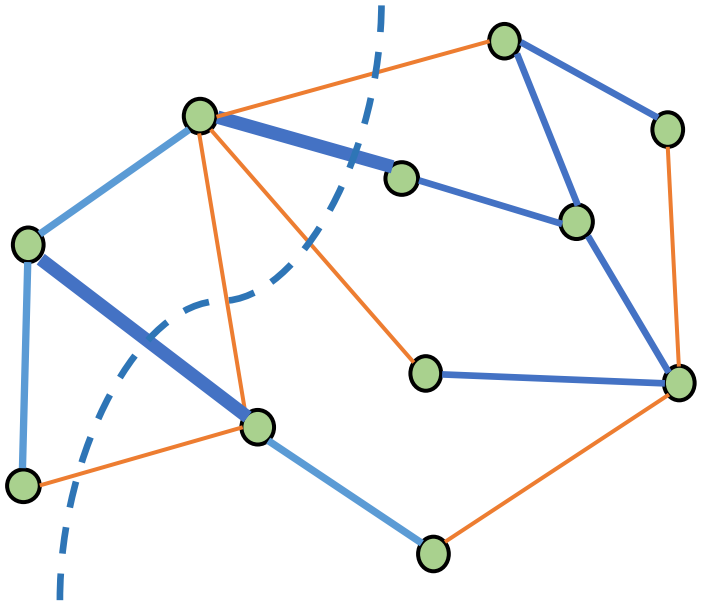
$$\frac{c(v)}{d(v)} \leq 1/2 \text{ for every } v \in N(C)$$

$$\sum_{v \in N(C)} \frac{c(v)}{d(v)} \leq 2\frac{|C|}{\delta(G)} \leq 2.$$

Seperating Matrix Works!

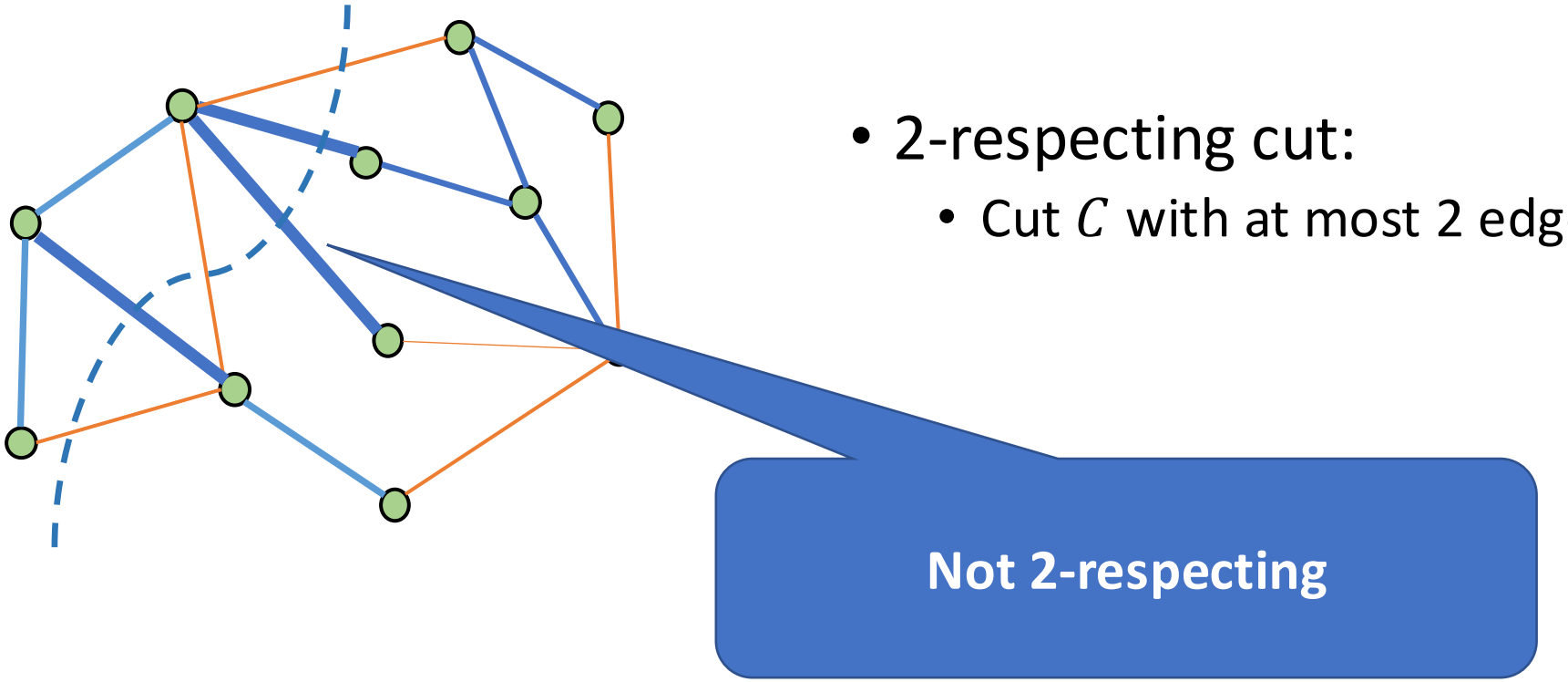
Minimum Cut on Weighted Graphs

From Approx to Exact: 2-respecting cut



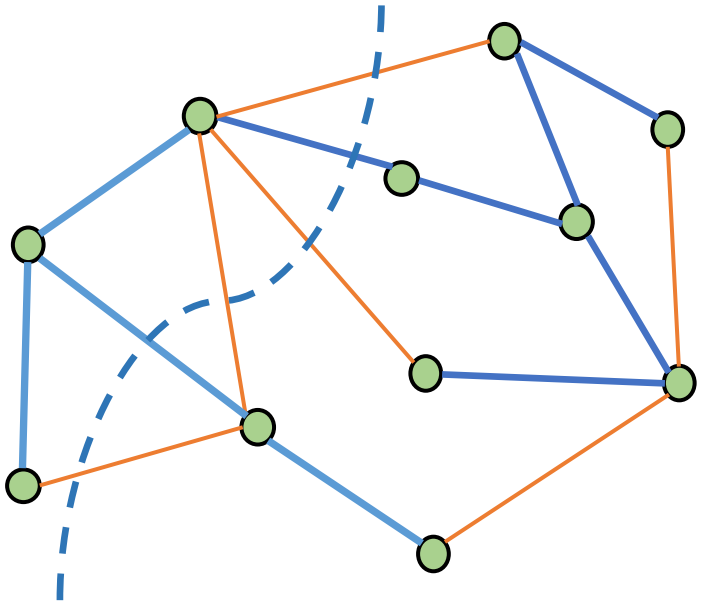
- Main subroutine of Karger's algorithm.
- Spanning tree:
  - A tree T with edges from $\mathrm{E}(G)$ that spans all vertices.
- 2-respecting cut:
  - Cut C with at most 2 edges from T.

2-respecting cut



- Spanning tree:
  - A tree T with edges from $\mathrm{E}(G)$ that spans all vertices.
- 2-respecting cut:
  - Cut C with at most 2 edges from T.

2-respecting cut



- Spanning tree:
  - A tree T with edges from $\mathrm{E}(G)$ that spans all vertices.
- 2-respecting cut
  - Cut C with at most 2 edges from T.

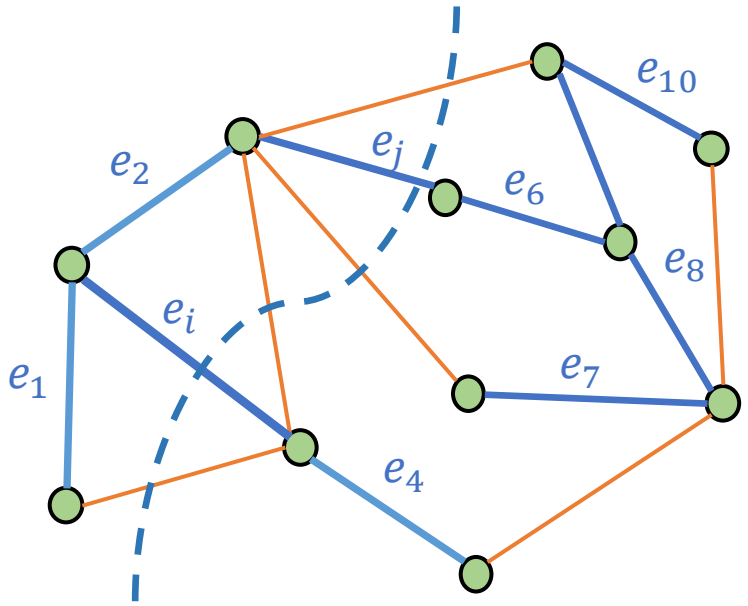
Main bottleneck!

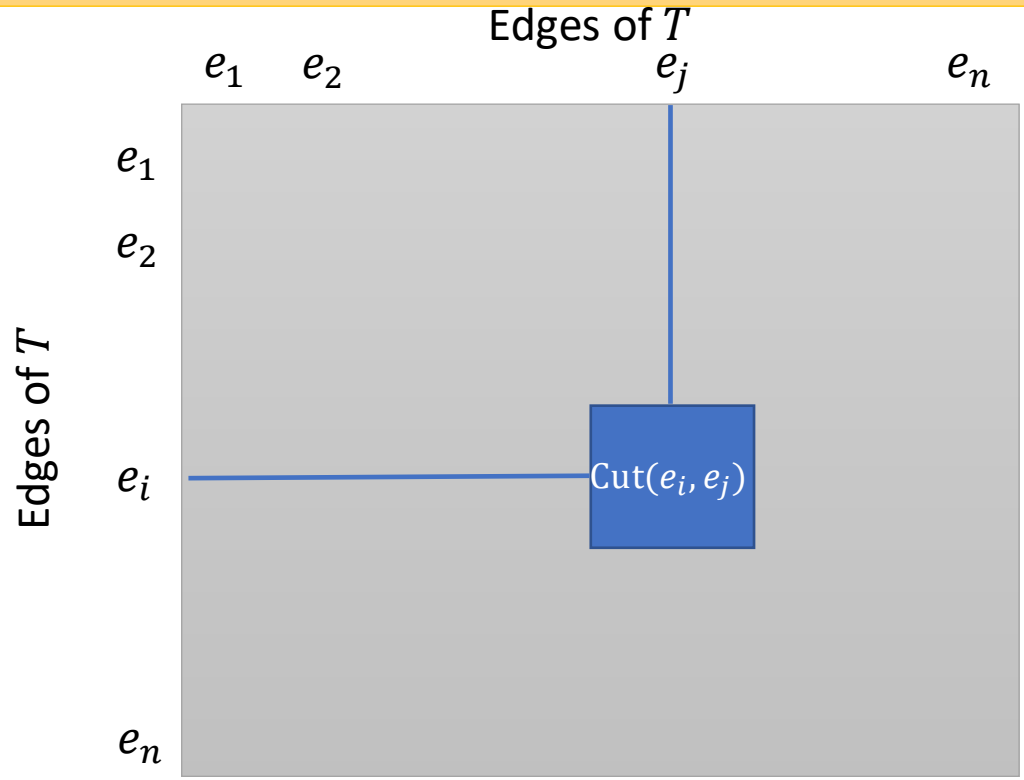
Goal: Find a 2 –respecting min-cut in a weighted graph given a spanning tree T.

Theorem (Karger). Efficient algorithm for solving 2-respecting min-cut implies efficient algorithm for solving min-cut.

A closer look…

Simple and efficient algorithm for **min 2-resp cut** for all models



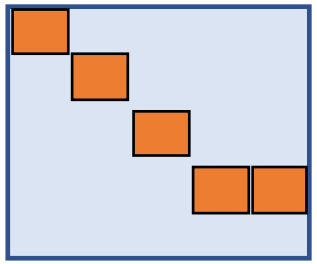


Matrix min-entry problem

Puzzle time!

Input: I will write down an $n \times n$ matrix M (you don't see M but know n).

Promise: **Monotonicity** - column minimums never move up as we go right



= min of each column

Example:

0	4	2	3
4	2	2	1
3	3	1	0
5	5	3	2

Goal: Find the minimum entry in M.

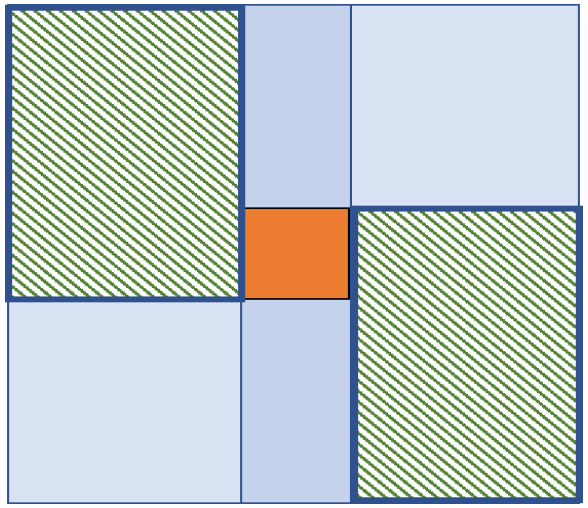
Query: You can ask for one entry at a time. **How many entries do you need?**

Solution for matrix min-entry

Puzzle time!

$O(n \log n)$ queries by divide and conquer.

1. Find min in the middle column
   - by asking for everything in that column.
2. The recurse on both sides.
3. Search area decreases by half.



In fact, there is an O(n)-query solution [SMAWK]. But it's not useful for us (too sequential)

Technical

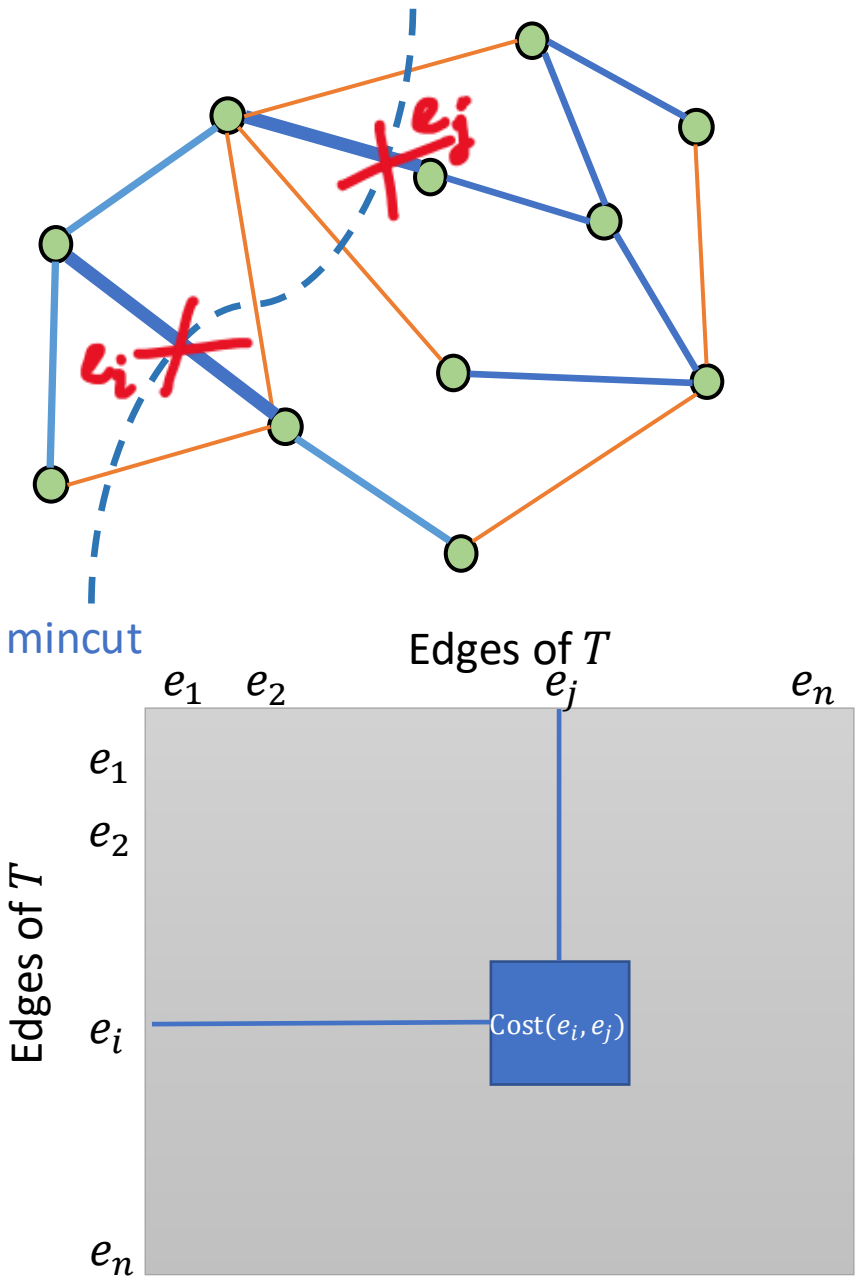
Mincut → Matrix min-entry

2-respecting min-cut: Easy to find spanning tree **T** that 2-constains the min-cut.

Cost matrix: Guess two tree-edges crossing the cuts
Entries in $n \times n$ matrix

There are $O(n^2)$ candidate cuts

Cut-query model: Can find **1** entry by **1** query.

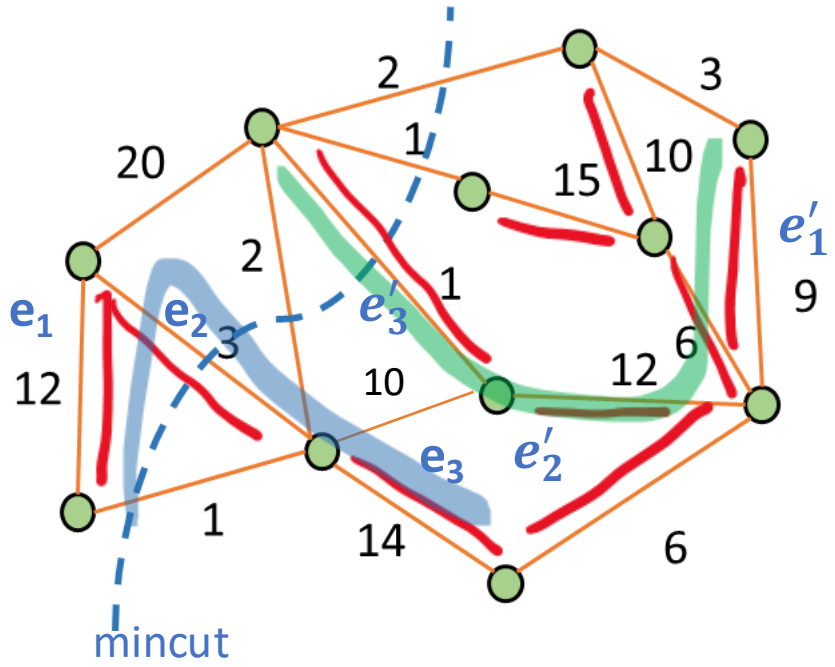


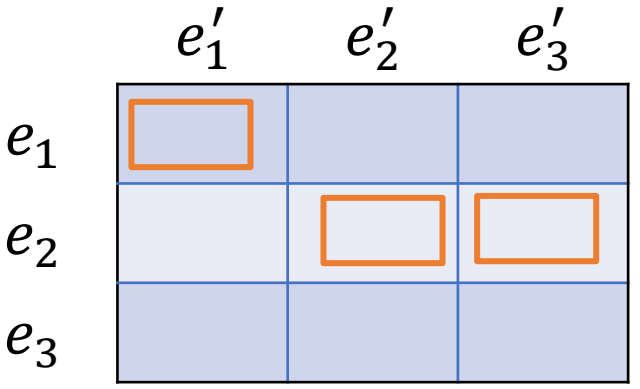
Technical

Mincut → Matrix min-entry

Cute trick: Assume we know two paths in **T** where the best candidates are in

Lemma: The corresponding cost matrix is **monotone** like in the puzzle!





Monotone matrix

Technical

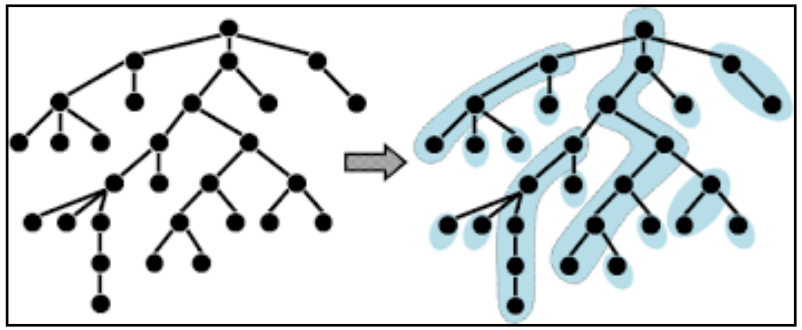
Mincut → Matrix min-entry (2)

Cute trick: Assume we know two paths in **T** where the best candidates are in

Lemma: The corresponding cost matrix is **monotone** like in the puzzle!

Decompose **T** into paths using **path decomposition** (e.g. heavy-light, bough/layering decomposition) & the simulate matrix min-entry algorithm for **some** pairs of paths

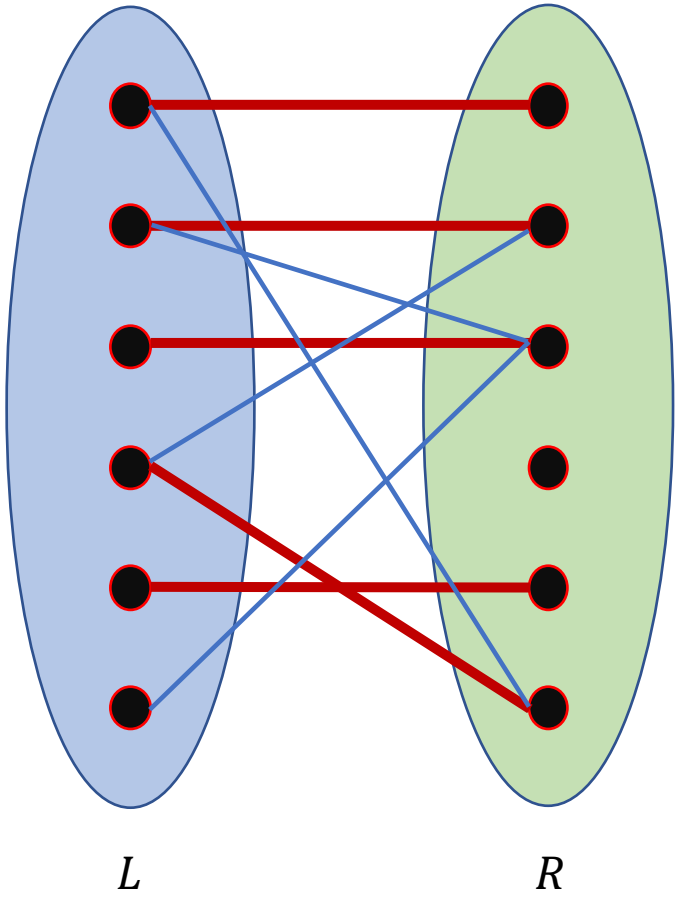
Leftover details: Picking a few pairs



Heavy-light decomposition

Bipartite matching

Maximum Cardinality Bipartite Matching



- Input: Bipartite graph $G = (V = L \cup R, E)$.
- Matching: Set of disjoint edges

Edge variables

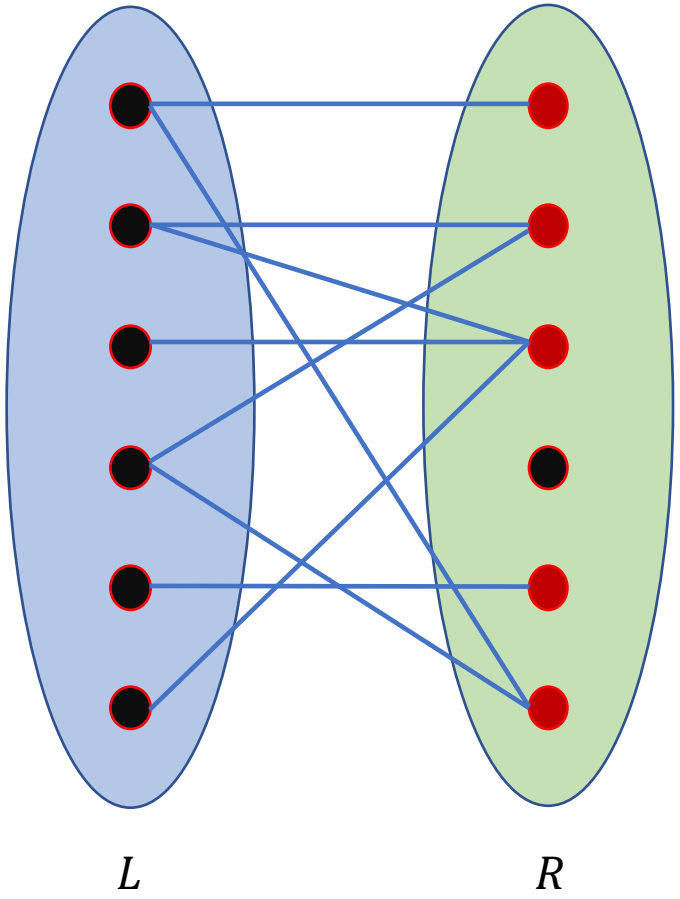
$$\text{maximize} \sum_{e \in E} z_e$$

Each vertex has at most 1 edge incident

$$\text{subject to} \quad \sum_{e \text{ incident on } v} z_e \leq 1 \qquad \text{for all node } v \in V,$$
$$z_e \in [0,1] \qquad \text{for all edge } e \in E.$$

Linear program of the maximum matching

Maximum Cardinality Bipartite Matching



$$\text{minimize} \sum_{v \in V} x_v$$

Each edge is covered by at least 1 vertex

subject to $x_u + x_v \geq 1$ for all edge $(u, v) \in E$,
$x_v \in [0,1]$ for all vertex $v \in V$.

Vertex variables

Dual of the maximum matching

König's Theorem: G has minimum VC of size F

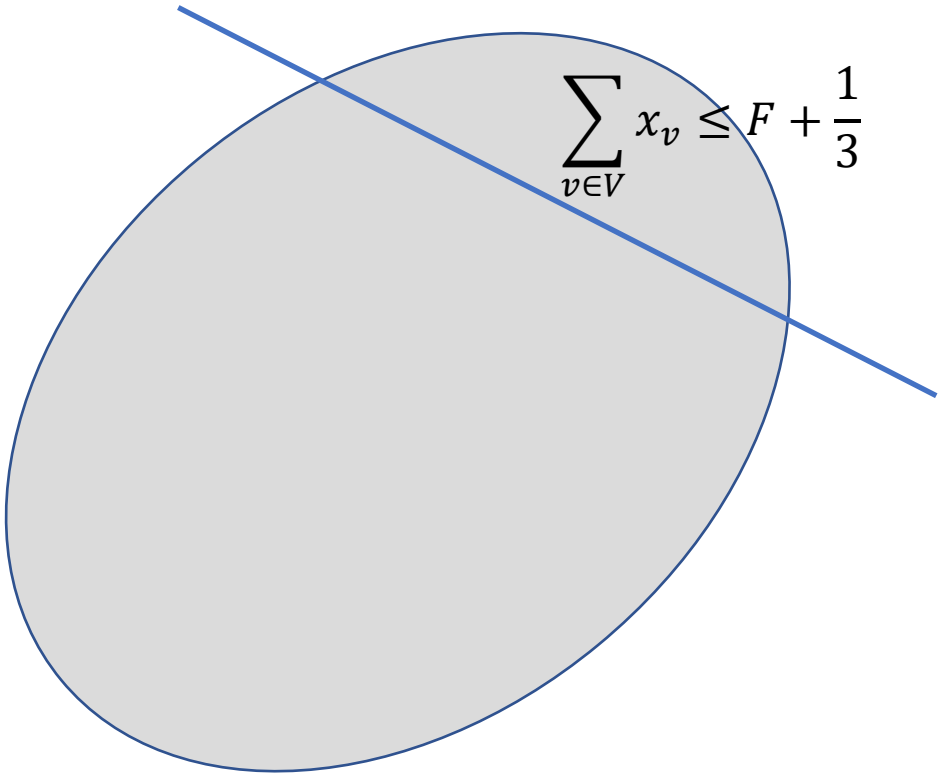
Size of maximum matching is F.

LP duality

Goal: To find out if the minimum VC is of size F.

Decision problem!

Distributed Linear System Solving



Does to change the solution!

$$\sum_{v \in V} x_v \leq F + \frac{1}{3}$$

$x_u + x_v \geq 1$ for all edge $(u, v) \in E$,

$x_v \in [0,1]$ for all vertex $v \in V$.

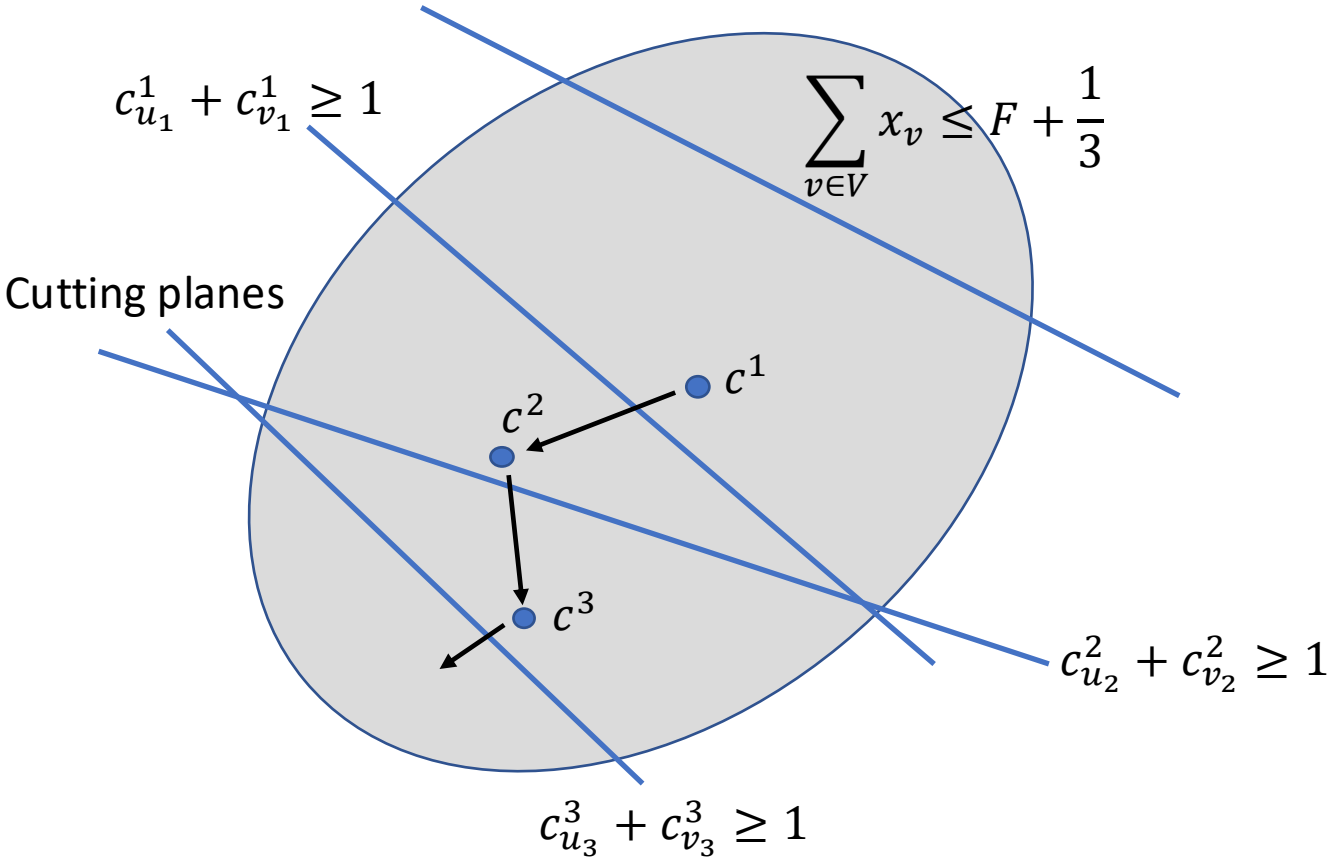
Feasibility problem

Communication complexity of Linear System/Program.

Fairly new and contemporary area of research!

- A good place to start: Vempala-Wang-Woodruff [SODA '20]
- For d-dimension system, their complexity is $\Theta(d^2 L)$.
  - L= bitlength to represent coefficient.
  - d= number of vertices for this case.

Cutting Plane Method



Does to change the solution!

$$\sum_{v \in V} x_v \leq F + \frac{1}{3}$$

$x_u + x_v \geq 1$ for all edge $(u, v) \in E$,

$x_v \in [0,1]$ for all vertex $v \in V$.

Feasibility problem

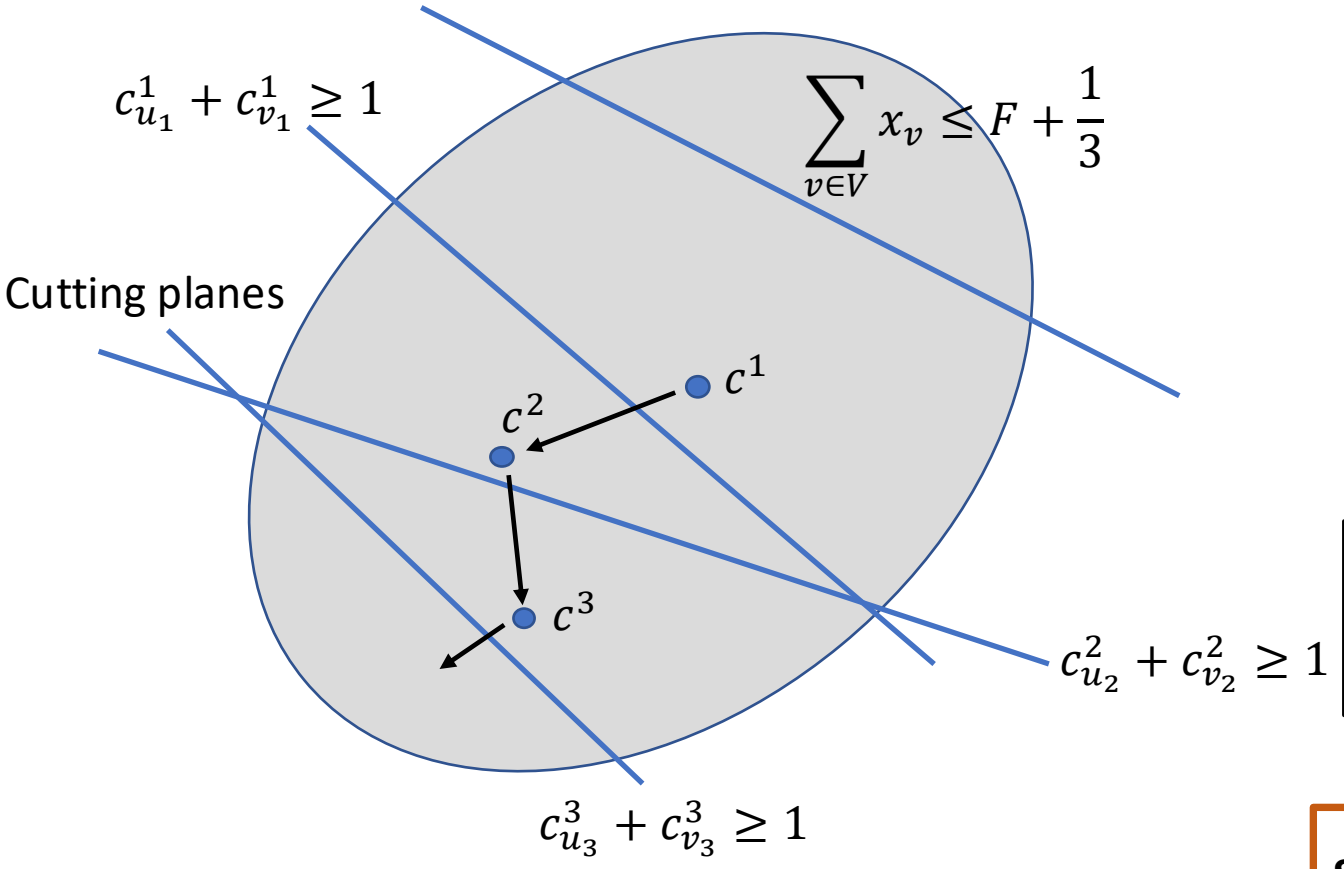
Solvable by Cutting Plane Method.

In Iteration i:

- Find a violating edge constraint on CG c^i.
- Restrict polytope on the 'feasible' side.

Theorem [Grünbaum '60]: Cutting plane cuts down at least a constant fraction of volume.

Cutting Plane Method



Does to change the solution!

$$\sum_{v \in V} x_v \leq F + \frac{1}{3}$$

$x_u + x_v \geq 1$ for all edge $(u, v) \in E$,

$x_v \in [0,1]$ for all vertex $v \in V$.

In Iteration i:

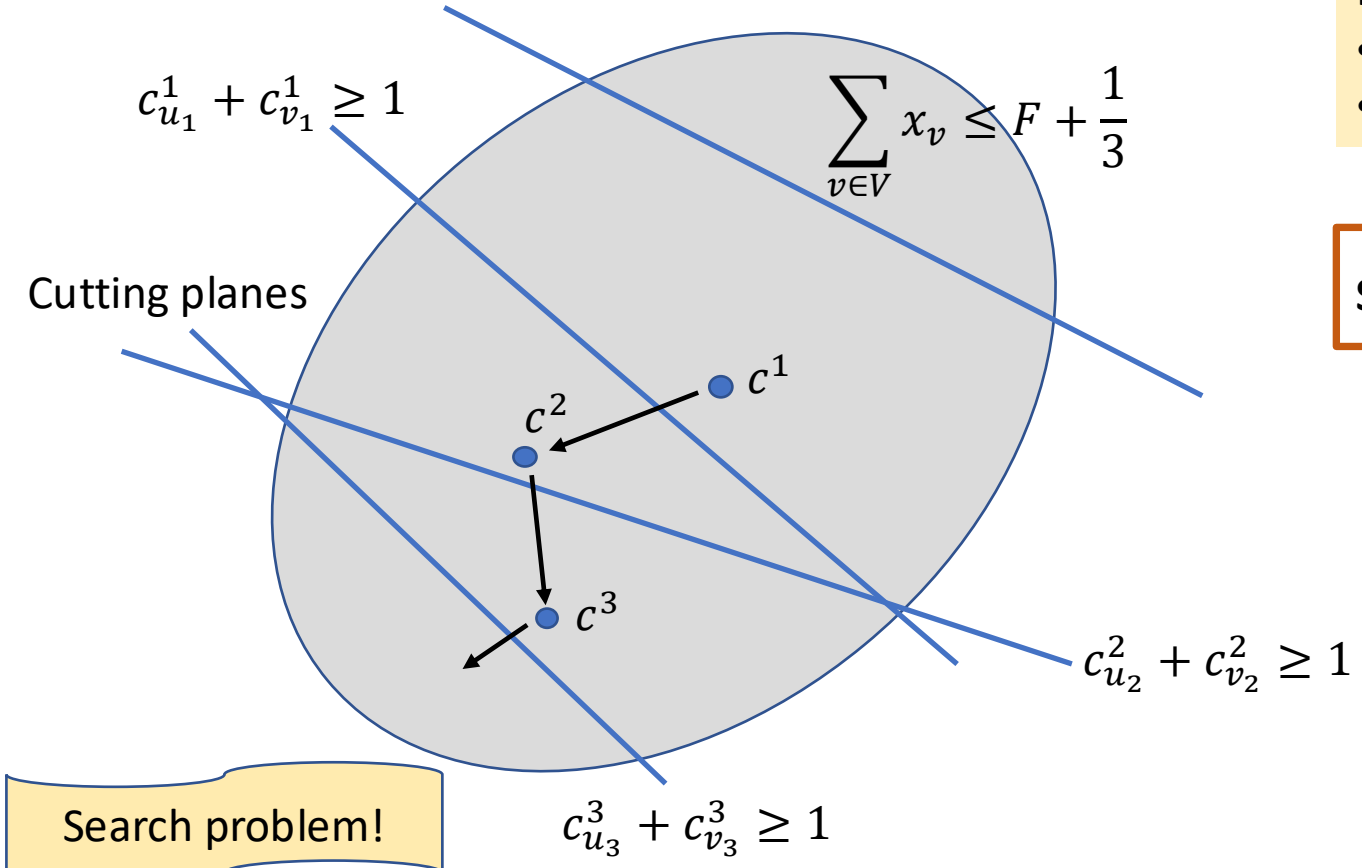
- Find a violating edge constraint on CG c^i.
- Restrict polytope on the 'feasible' side.

Theorem [Grünbaum '60]: Cutting plane cuts down at least a constant fraction of volume.

Stopping condition: Size of polytope $\leq \left(\frac{1}{20n}\right)^{2n}$.

Polytope is empty!

Cutting Plane Method



In Iteration i:

- Find a violating edge constraint on CG c^i.
- Restrict polytope on the 'good' side.

Stopping condition: Size of polytope $\leq \left(\frac{1}{20n}\right)^{2n}$.

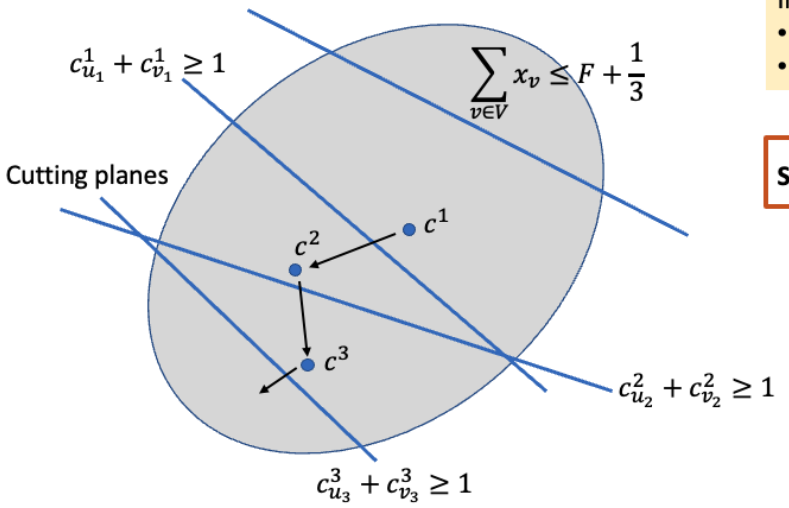
- # Iterations: $O(n \log n)$.
- Alice or Bob can find the violating constraint and exchange.

Communication complexity = $O(n \log^2 n)$.

Search problem!

Polytope is empty → A maximum matching can be found within the violating edges.

Ramifications



In Iteration i:
- Find a violating edge constraint on CG c^i.
- Restrict polytope on the 'good' side.

Stopping condition: Size of polytope $\leq \left(\frac{1}{20n}\right)^{2n}$.

- # Iterations: $O(n \log n)$.
- Alice or Bob can find the violating constraint and exchange.

Communication complexity = $O(n \log^2 n)$.

What about other models?

Quantum edge query: Allowed to ask if any edge is present from a *quantum oracle*. Answer is qubits.

OR-query protocols: Allowed to ask if any edge present in a subset $S \subseteq L \times R$.

Step 1: Compute possible violating edge set $\mathrm{Vio}(c) = \{(u, v) \mid c_u + c_v < 1\}$.

Step 2: Check if any edge from $\mathrm{Vio}(c)$ present in G.

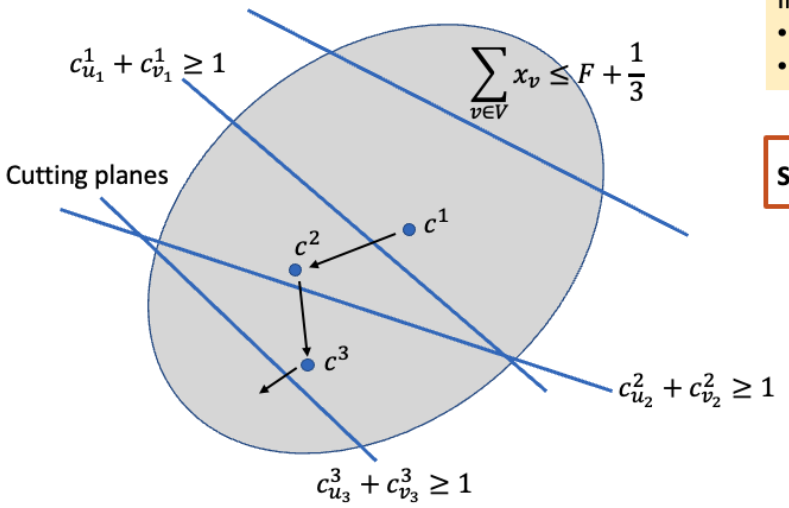
Binary search!

Grover search + Amortization

OR-query complexity: $O(n \log^2 n)$.

Quantum edge query complexity: $O(n^{1.5} \log^2 n)$.

Ramifications



In Iteration i:
- Find a violating edge constraint on CG c^i.
- Restrict polytope on the 'good' side.

Stopping condition: Size of polytope $\leq \left(\frac{1}{20n}\right)^{2n}$.

- # Iterations: $O(n \log n)$.
- Alice or Bob can find the violating constraint and exchange.

Communication complexity = $O(n \log^2 n)$.

What about other models?

Quantum edge query: Allowed to ask if any edge is present from a *quantum oracle*. Answer is qubits.

OR-query protocols: Allowed to ask if any edge present in a subset $S \subseteq L \times R$.

Step 1: Compute possible violating edge set $\mathrm{Vio}(c) = \{(u, v) \mid c_u + c_v < 1\}$.

Step 2: Check if any edge from $\mathrm{Vio}(c)$ present in G.

How about cut query?

1. If c is integral, easy to find $\mathrm{Vio}(c)$. Search within $L_{c_u=0} \times R_{c_u=0}$.
2. If c is not integral, $\exists$ integral c' such that $|c'| < |c|$ and $Vio(c) = Vio(c')$.

Thank you.