Deterministic Almost-Linear-Time Gomory-Hu Trees

Amir Abboud (Weizmann Institute of Science),
Rasmus Kyng (ETHZ),
Jason Li (CMU),
Debmalya Panigrahi (Duke),
Maximilian Probst Gutenberg (ETHZ),
Thatchaphol Saranurak (UMich),
Weixuan Yuan (ETHZ),
Wuwei Yuan (ETHZ)

FOCS 2025

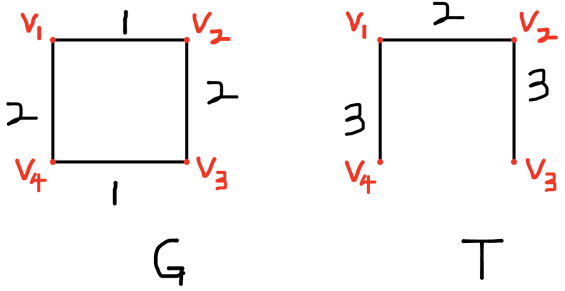
December 16, 2025

Problem Description

- Unless otherwise specified, all graphs are *undirected* and *weighted*.

Problem Description

- Unless otherwise specified, all graphs are *undirected* and *weighted*.
- Given graph $G = (V, E, w)$, compute tree T on same vertex set s.t. $\text{mincut}_G(s,t) = \text{mincut}_T(s,t)$ for all $s, t \in V$.
- T is known as **Gomory-Hu tree**.



Related Works

- Max-flow takes rand. [CKLPPS'22] / deter. [BCKLPPSS'23][BCKLMGS'24] $m^{1+o(1)}$ time.
- $\tilde{O}(\cdot)$ hides polylog(n) factors.

Reference	Rand./Deter.	Time
Gomory-Hu'61	deter.	$nm^{1+o(1)}$
Abboud-Krauthgamer-Li-Panigrahi-Saranurak-Trabelsi'21	rand.	$\tilde{O}(n^{2.875})$
Zhang'21, AKLPST'21	rand.	$\tilde{O}(n^2)$
Abboud-Li-Panigrahi-Saranurak'23	rand.	$m^{1+o(1)}$

Related Works

- Max-flow takes rand. [CKLPPS'22] / deter. [BCKLPPSS'23][BCKLMGS'24] $m^{1+o(1)}$ time.
- $\tilde{O}(\cdot)$ hides polylog(n) factors.

Reference	Rand./Deter.	Time
Gomory-Hu'61	deter.	$nm^{1+o(1)}$
Abboud-Krauthgamer-Li-Panigrahi-Saranurak-Trabelsi'21	rand.	$\tilde{O}(n^{2.875})$
Zhang'21, AKLPST'21	rand.	$\tilde{O}(n^2)$
Abboud-Li-Panigrahi-Saranurak'23	rand.	$m^{1+o(1)}$
This Work	deter.	$m^{1+o(1)}$

Related Works (Cont.)

- Assume graphs are **unweighted**.
- Assume max-flow takes deterministic $T(n, m) = \Omega(m)$ time.
- $\tilde{O}(\cdot)$ hides polylog(n) factors.
- Simple algorithm = bypass single-source min-cut.

Reference	Rand./Deter.	Time	Remark
Gomory-Hu'61	deter.	$n \cdot T(n, m)$	simple algo.
AKLPST'21	rand.	$\tilde{O}(n^{2.875})$	
AKLPST'21	rand.	$m^{1+o(1)}$	
ALPS'23	rand.	$n^{1+o(1)} + \tilde{O}(T(n, m))$	
This Work	deter.	$m^{1+o(1)}$	

Related Works (Cont.)

- Assume graphs are **unweighted**.
- Assume max-flow takes deterministic $T(n, m) = \Omega(m)$ time.
- $\tilde{O}(\cdot)$ hides polylog(n) factors.
- Simple algorithm = bypass single-source min-cut.

Reference	Rand./Deter.	Time	Remark
Gomory-Hu'61	deter.	$n \cdot T(n, m)$	simple algo.
AKLPST'21	rand.	$\tilde{O}(n^{2.875})$	
AKLPST'21	rand.	$m^{1+o(1)}$	
ALPS'23	rand.	$n^{1+o(1)} + \tilde{O}(T(n, m))$	
This Work	deter.	$m^{1+o(1)}$	
Kyng-Probst Gutenberg-**Y**-**Y**'25	rand.	$\tilde{O}(T(n, m))$	simple algo.
Probst Gutenberg-**Y**'25	deter.	$\tilde{O}(T(n, m))$[1]	simple algo.

[1] Assuming expander decomposition takes deterministic $\tilde{O}(m)$ time.

Related Works (Cont.)

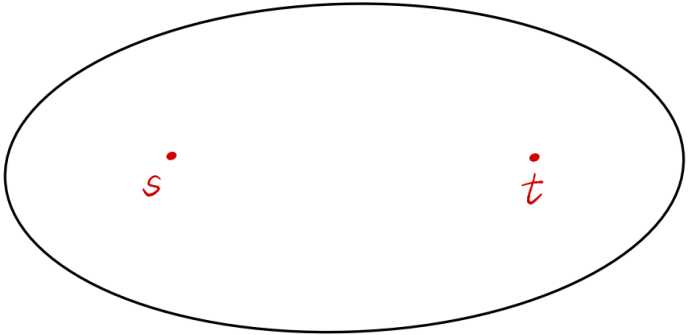
- Assume graphs are **unweighted**.
- Assume max-flow takes deterministic $\tilde{O}(m)$ time.
- $\tilde{O}(\cdot)$ hides polylog(n) factors.
- Simple algorithm = bypass single-source min-cut.

Reference	Rand./deter.	Time	Remark
Gomory-Hu'61	deter.	$\tilde{O}(nm)$	simple algo.
AKLPST'21	rand.	$\tilde{O}(n^{2.875})$	
AKLPST'21	rand.	$m^{1+o(1)}$	
ALPS'23	rand.	$n^{1+o(1)} + \tilde{O}(m)$	
This Work	deter.	$m^{1+o(1)}$	
Kyng-Probst Gutenberg-**Y**-**Y**'25	rand.	$\tilde{O}(m)$	simple algo.
Probst Gutenberg-**Y**'25	deter.	$\tilde{O}(m)$[1]	simple algo.

[1] Assuming expander decomposition takes deterministic $\tilde{O}(m)$ time.

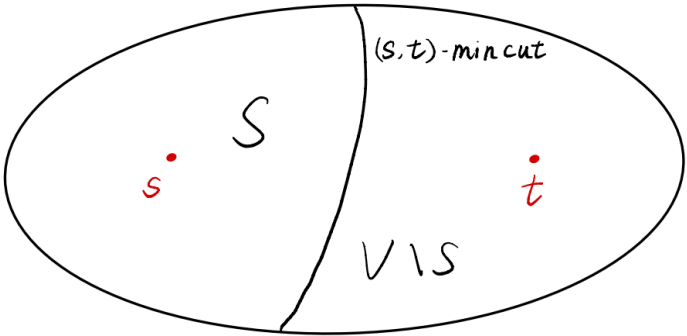
Gomory and Hu's Algorithm

Arbitrary $s, t \in V$



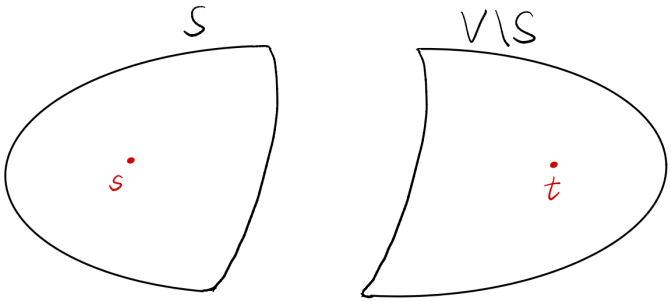
Gomory and Hu's Algorithm

Arbitrary $s, t \in V \implies (s,t)$-mincut $(S, V \setminus S)$



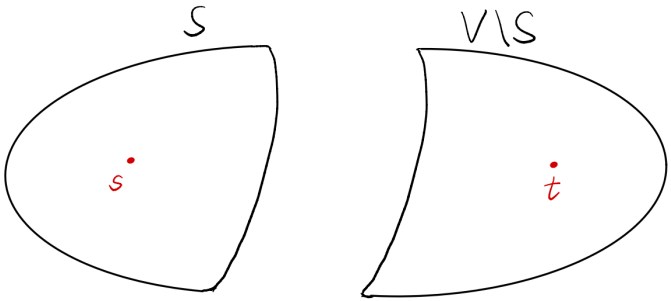
Gomory and Hu's Algorithm

Arbitrary $s, t \in V \implies (s,t)$-mincut $(S, V \setminus S) \implies$ recursion



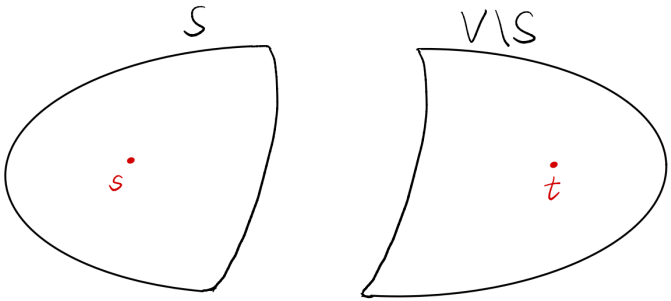
Gomory and Hu's Algorithm

Arbitrary $s, t \in V \implies (s,t)$-mincut $(S, V \setminus S) \implies$ recursion



Gomory and Hu's Algorithm

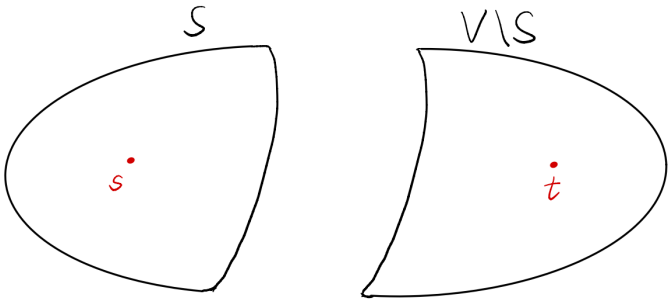
Arbitrary $s, t \in V \implies (s,t)$-mincut $(S, V \setminus S) \implies$ recursion



- The (s,t)-mincut $\approx$ one edge in the Gomory-Hu tree

Gomory and Hu's Algorithm

Arbitrary $s, t \in V \implies (s,t)$-mincut $(S, V \setminus S) \implies$ recursion



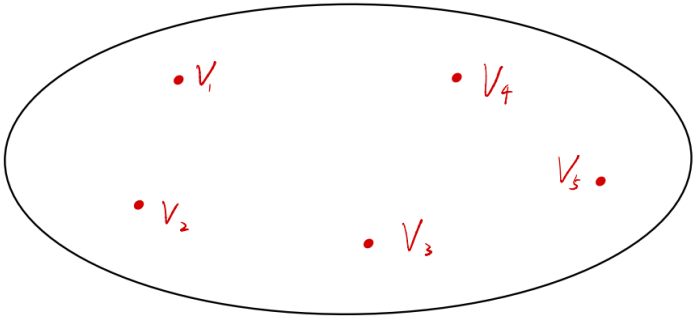
- The (s,t)-mincut $\approx$ one edge in the Gomory-Hu tree
- Unbalanced decomposition $\implies$ large recursion depth $\implies$ superlinear complexity

Isolating Cuts

- **Isolating cuts** (Li-Panigrahi'20)
  Input: G, vertices $P = \{v_1, \ldots, v_l\} \subseteq V$
  Output: disjoint vertex sets $S_1, \ldots, S_l$, such that S_i is the inclusion-minimal min $(v_i, P \setminus v_i)$ cut.

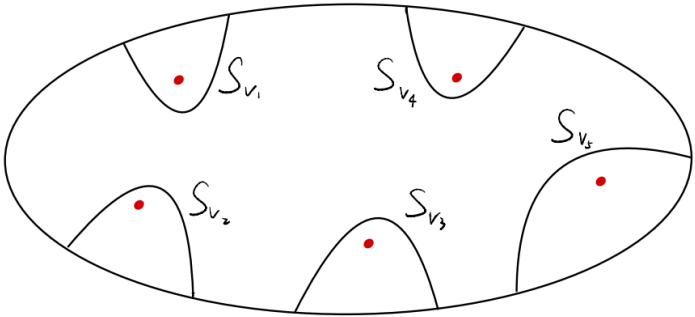
Isolating Cuts

- **Isolating cuts** (Li-Panigrahi'20)
  Input: G, vertices $P = \{v_1, \ldots, v_l\} \subseteq V$
  Output: disjoint vertex sets $S_1, \ldots, S_l$, such that S_i is the inclusion-minimal min $(v_i, P \setminus v_i)$ cut.



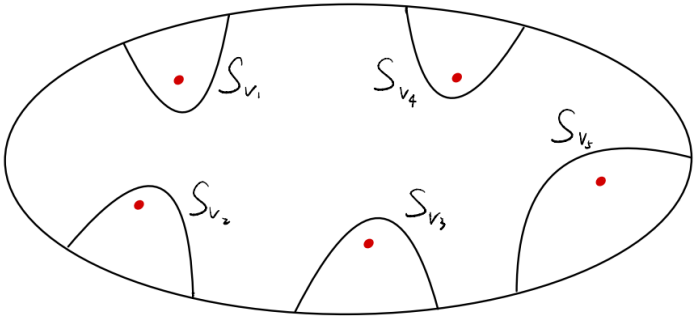
Isolating Cuts

- **Isolating cuts** (Li-Panigrahi'20)
  Input: G, vertices $P = \{v_1, \ldots, v_l\} \subseteq V$
  Output: disjoint vertex sets $S_1, \ldots, S_l$, such that S_i is the inclusion-minimal min $(v_i, P \setminus v_i)$ cut.



Isolating Cuts

- **Isolating cuts** (Li-Panigrahi'20)
  Input: G, vertices $P = \{v_1, \ldots, v_l\} \subseteq V$
  Output: disjoint vertex sets $S_1, \ldots, S_l$, such that S_i is the inclusion-minimal min $(v_i, P \setminus v_i)$ cut.



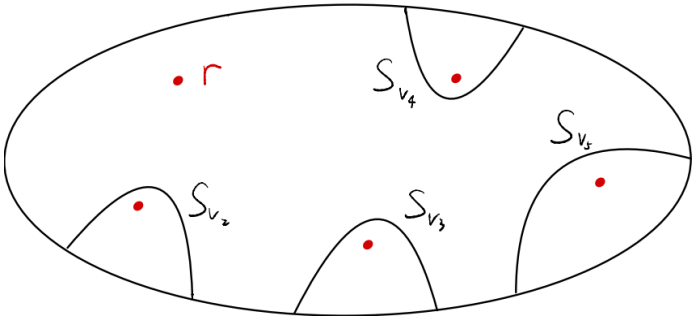
- Time cost: $\tilde{O}(T_{maxflow}(m)) = m^{1+o(1)}$.

Single-Source Mincuts(SSMC)

- **Single-source Mincuts(SSMC)**
  Input: $r \in V$
  Output: $\{\lambda(r,t) : t \in V \setminus \{r\}\}$
  Complexity: $m^{1+o(1)}$

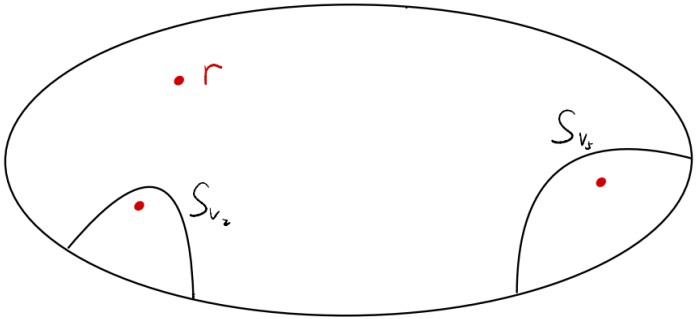
Single-Source Mincuts(SSMC)

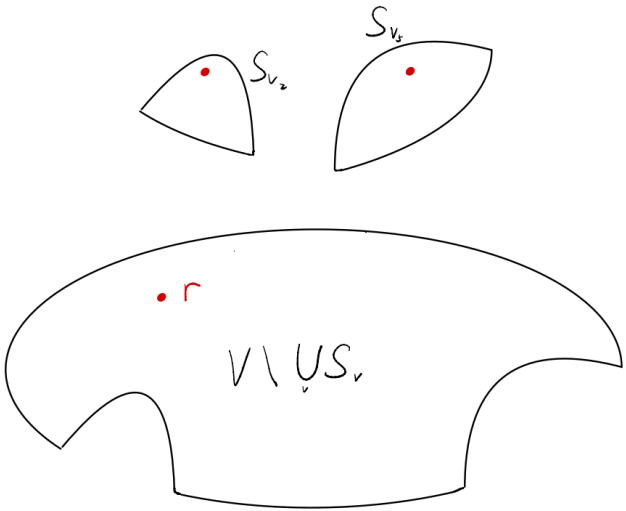
- **Single-source Mincuts(SSMC)**
  Input: $r \in V$
  Output: $\{\lambda(r,t) : t \in V \setminus \{r\}\}$
  Complexity: $m^{1+o(1)}$



Single-Source Mincuts(SSMC)

- **Single-source Mincuts(SSMC)**
  Input: $r \in V$
  Output: $\{\lambda(r,t) : t \in V \setminus \{r\}\}$
  Complexity: $m^{1+o(1)}$





- We want a balanced decomposition.

A General Framework for Decomposition

- Find pivot r and vertex set $P \subseteq V$

A General Framework for Decomposition

- Find pivot r and vertex set $P \subseteq V$
- Run isolating cuts on $\{r\} \cup P \Longrightarrow S_v : v \in P$

A General Framework for Decomposition

- Find pivot r and vertex set $P \subseteq V$
- Run isolating cuts on $\{r\} \cup P \Longrightarrow S_v : v \in P$
- $P' := \{v : S_v \text{ is a } (v, r)\text{-mincut, and } |S_v| \leq n/2\}$

A General Framework for Decomposition

- Find pivot r and vertex set $P \subseteq V$
- Run isolating cuts on $\{r\} \cup P \Longrightarrow S_v : v \in P$
- $P' := \{v : S_v \text{ is a } (v, r)\text{-mincut, and } |S_v| \leq n/2\}$
- Recurse on S_v for each $v \in P'$ and $V \setminus \bigcup_{v \in P'} S_v$

A General Framework for Decomposition

- Find pivot r and vertex set $P \subseteq V$
- Run isolating cuts on $\{r\} \cup P \Longrightarrow S_v : v \in P$
- $P' := \{v : S_v \text{ is a } (v, r)\text{-mincut, and } |S_v| \leq n/2\}$
- Recurse on S_v for each $v \in P'$ and $V \setminus \bigcup_{v \in P'} S_v$

To make the decomposition balanced, we also need $V \setminus \bigcup_{v \in P'} S_v$ small, i.e. $\bigcup_{v \in P'} S_v$ is large ($\left|\bigcup_{v \in P'} S_v\right| \geq n/n^{o(1)}$)

A General Framework for Decomposition

- Find pivot r and vertex set $P \subseteq V$
- Run isolating cuts on $\{r\} \cup P \Longrightarrow S_v : v \in P$
- $P' := \{v : S_v \text{ is a } (v, r)\text{-mincut, and } |S_v| \leq n/2\}$
- Recurse on S_v for each $v \in P'$ and $V \setminus \bigcup_{v \in P'} S_v$

To make the decomposition balanced, we also need $V \setminus \bigcup_{v \in P'} S_v$ small, i.e. $\bigcup_{v \in P'} S_v$ is large ($\left|\bigcup_{v \in P'} S_v\right| \geq n/n^{o(1)}$)

How to select r and P?

Randomized Almost-Linear-Time Algorithm (ALPS'23)

- Sample a pivot r u.a.r.
- For $i = 1, 2, \ldots \lfloor \log n \rfloor$, sample vertex set P with $|P| = 2^i$ vertices u.a.r.

Randomized Almost-Linear-Time Algorithm (ALPS'23)

- Sample a pivot r u.a.r.
- For $i = 1, 2, \ldots \lfloor \log n \rfloor$, sample vertex set P with $|P| = 2^i$ vertices u.a.r.

In expectation, one of the decompositions is balanced.

Pivot r

τ-connected component: equivalence class under $s \sim t \iff \lambda(s,t) \geq \tau$.

Pivot r

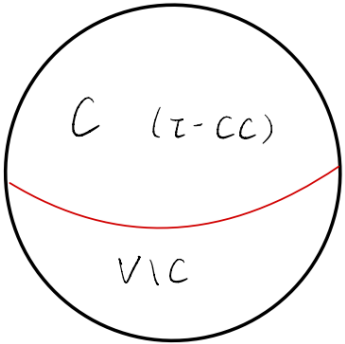
τ-connected component: equivalence class under $s \sim t \iff \lambda(s,t) \geq \tau$.
Find a threshold value τ such that:

- The largest τ-connected component has more than $|V|/2$ vertices, call it C.
- All $(\tau+1)$-connected component has no more than $|V|/2$ vertices.

Pivot r

τ-connected component: equivalence class under $s \sim t \iff \lambda(s,t) \geq \tau$.
Find a threshold value τ such that:

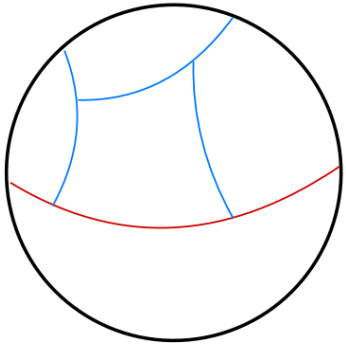
- The largest τ-connected component has more than $|V|/2$ vertices, call it C.
- All $(\tau+1)$-connected component has no more than $|V|/2$ vertices.



Pivot r

τ-connected component: equivalence class under $s \sim t \iff \lambda(s,t) \geq \tau$.
Find a threshold value τ such that:

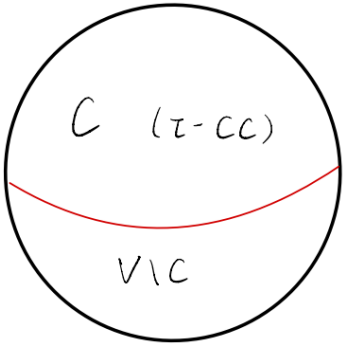
- The largest τ-connected component has more than $|V|/2$ vertices, call it C.
- All $(\tau+1)$-connected component has no more than $|V|/2$ vertices.



Pivot r

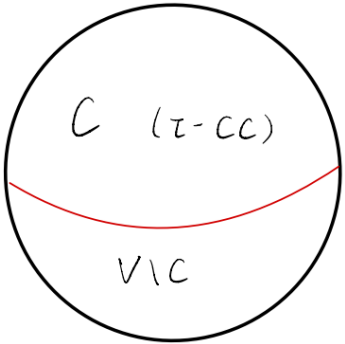
τ-connected component: equivalence class under $s \sim t \iff \lambda(s,t) \geq \tau$.
Find a threshold value τ such that:

- The largest τ-connected component has more than $|V|/2$ vertices, call it C.
- All $(\tau+1)$-connected component has no more than $|V|/2$ vertices.



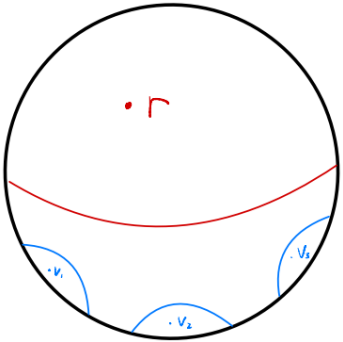
Pick an arbitrary $r \in C$ as the pivot.

Case 1: C is small



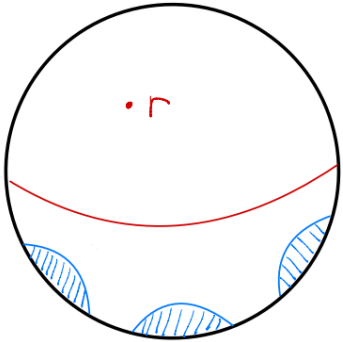
1. Start from some $A \leftarrow V \setminus C$

Case 1: C is small



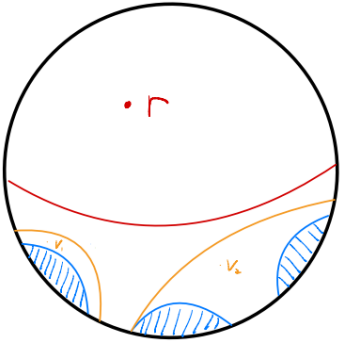
1. Start from some $A \leftarrow V \setminus C$
2. Find some $P \subseteq A \implies$ small (r, v)-mincuts $\{S_v\}$

Case 1: C is small



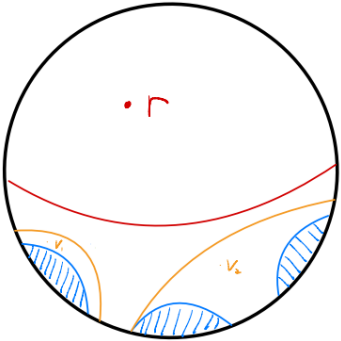
1. Start from some $A \leftarrow V \setminus C$
2. Find some $P \subseteq A \implies$ small (r, v)-mincuts $\{S_v\}$
3. $A \leftarrow A \setminus \bigcup(r, v)$-mincuts.

Case 1: C is small



1. Start from some $A \leftarrow V \setminus C$
2. Find some $P \subseteq A \Longrightarrow$ small (r, v)-mincuts $\{S_v\}$
3. $A \leftarrow A \setminus \bigcup (r, v)$-mincuts.
4. Repeat 2 and 3 for $n^{o(1)}$ times.

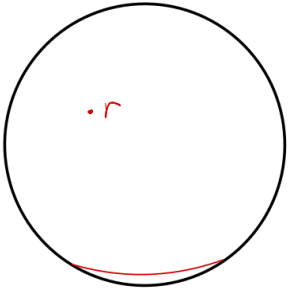
Case 1: C is small



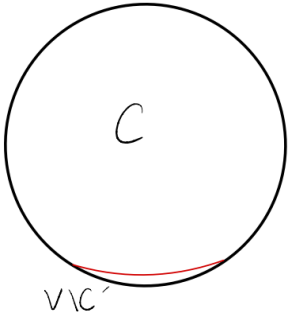
1. Start from some $A \leftarrow V \setminus C$
2. Find some $P \subseteq A \Longrightarrow$ small (r, v)-mincuts $\{S_v\}$
3. $A \leftarrow A \setminus \bigcup(r, v)$-mincuts.
4. Repeat 2 and 3 for $n^{o(1)}$ times.

We get a collection of (inclusion-minimal) (r, v)-mincuts. Their union is $V \setminus C$

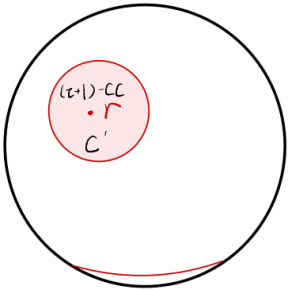
Case 2: C is large



Case 2: C is large



Case 2: C is large



1. Start from some $A \leftarrow C \setminus C'$
2. Find some $P \subseteq A \Longrightarrow$ small (r, v)-mincuts $\{S_v\}$
3. $A \leftarrow A \setminus \bigcup(r, v)$-mincuts.
4. Repeat 2 and 3 for $n^{o(1)}$ times.

How About SSMC?

Roadmap (fix some $r \in V$):

How About SSMC?

Roadmap (fix some $r \in V$):

- **Vertex sparsifier**: Input : $U \subseteq V$ containing r
  Output: A graph H defined on U with $m^{1+o(1)}$ edges, such that restricting an (s,t)-mincut in G is an approximate (s,t)-mincut in H

How About SSMC?

Roadmap (fix some $r \in V$):

- **Vertex sparsifier**: Input : $U \subseteq V$ containing r
  Output: A graph H defined on U with $m^{1+o(1)}$ edges, such that restricting an (s,t)-mincut in G is an approximate (s,t)-mincut in H
- **Guide trees** : a collection of $n^{o(1)}$ trees defined on U, such that
  1. $\forall t \in U$ and (s,t)-mincut $(S, V \setminus S)$, there exists some tree crossing the cut at most k-times.

  Edge sparsifier of H + tree packing

How About SSMC?

Roadmap (fix some $r \in V$):

- **Vertex sparsifier**: Input : $U \subseteq V$ containing r
  Output: A graph H defined on U with $m^{1+o(1)}$ edges, such that restricting an (s,t)-mincut in G is an approximate (s,t)-mincut in H
- **Guide trees** : a collection of $n^{o(1)}$ trees defined on U, such that
  1. $\forall t \in U$ and (s,t)-mincut $(S, V \setminus S)$, there exists some tree crossing the cut at most k-times.

  Edge sparsifier of H + tree packing
- **Guide trees to SSMC**

SSMC is very complicated, can we avoid SSMC?

SSMC is very complicated, can we avoid SSMC?

Yes, for unweighted graphs.

SSMC is very complicated, can we avoid SSMC?

Yes, for unweighted graphs.

A Simple and Fast Reduction from Gomory-Hu Trees to Polylog Maxflows.
by Maximilian Probst Gutenberg, Rasmus Kyng, Weixuan Yuan, Wuwei Yuan

A Simple Deterministic Reduction From Gomory-Hu Tree to Maxflow and Expander Decomposition
by Maximilian Probst Gutenberg and Weixuan Yuan

Summary: Gomory-Hu Trees

- Deterministic $m^{1+o(1)}$-time algorithm for weighted graphs. [**This Work**]
  Also:
  - Simple randomized $\tilde{O}(T(n,m))$-time algorithm for unweighted graphs. [KP**YY**'25]
  - Simple deterministic $\tilde{O}(T(n,m) + T_{ExpDecomp}(n,m))$-time algorithm for unweighted graphs. [P**Y**'25]
- Open problems:
  - Simple algorithms for weighted graphs.
  - $\tilde{O}(T(n,m))$-time algorithm for weighted graphs.
  - Efficient dynamic algorithms.
    [Kenneth-Mordoch and Krauthgamer'25]: Deterministic fully-dynamic algorithm with $n^{3/2+o(1)}$ worst-case update time.
  - Generalize Gomory-Hu tree to hypergraphs, element connectivity, and vertex connectivity.

Thanks!