# Faster Matroid Partition Algorithms

**Tatsuya Terao**
Kyoto University

HJ 2025 @Tokyo May 26, 2025

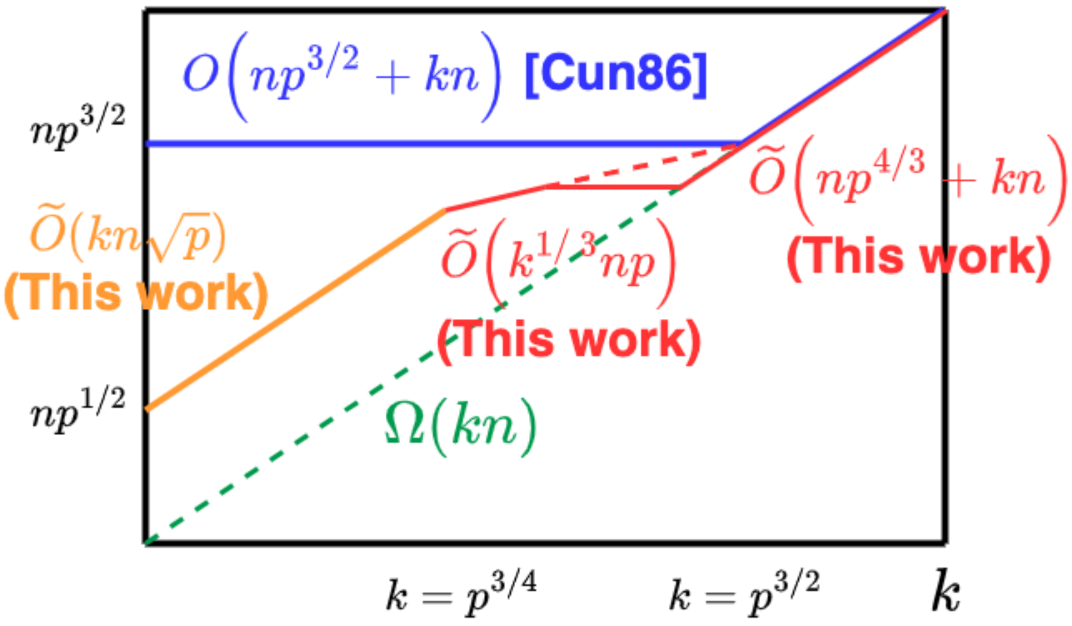
Tatsuya Terao :Faster Matroid Partition Algorithms,
A preliminary version appears in **ICALP 2023, a full version is published at TALG.** (arXiv:2303.05920)

# Summary

**Result**

Three fast algorithms for **matroid partition**

- Algorithm 1.
  $\tilde{O}(kn\sqrt{p})$ **independence** queries
- Algorithm 2.
  $\tilde{O}(k'^{1/3}np + kn)$ **independence** queries
- Algorithm 3.
  $\tilde{O}((n+k)\sqrt{p})$ **rank** queries

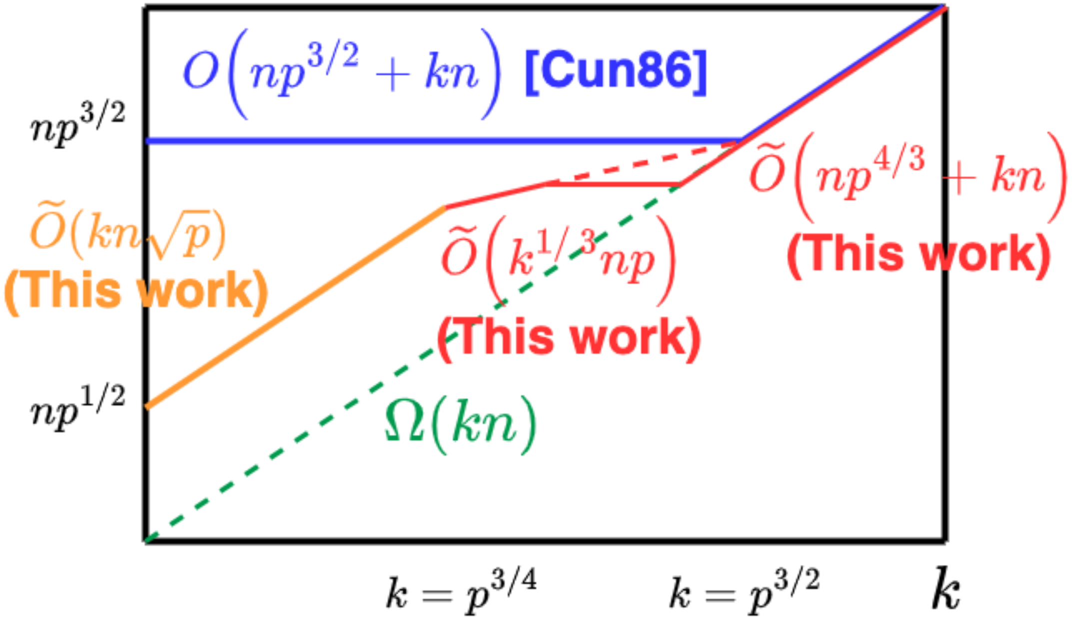



$n$ = #elements, $k$ = #matroids
$p$ = solution size, $k' = \min\{k, p\}$

# Summary

## Result
Three fast algorithms for **matroid partition**


- Algorithm 1.
  $\tilde{O}(kn\sqrt{p})$ **independence** queries
- Algorithm 2.
  $\tilde{O}(k'^{1/3}np + kn)$ **independence** queries
- Algorithm 3.
  $\tilde{O}((n+k)\sqrt{p})$ **rank** queries




$n$ = #elements, $k$ = #matroids
$p$ = solution size, $k' = \min\{k, p\}$

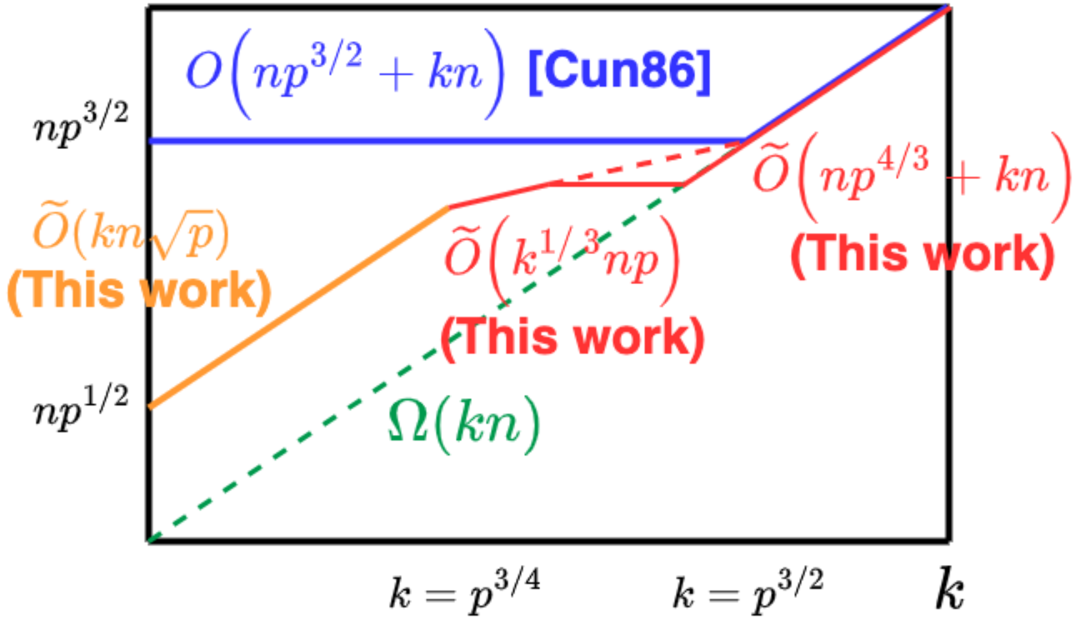
# Summary

**Result**

Three fast algorithm

the first improvement since [Cunningham'86]

- Algorithm 1.
  $\widetilde{O}(kn\sqrt{p})$ **independence** queries
- Algorithm 2.
  $\widetilde{O}(k'^{1/3}np + kn)$ **independence** queries
- Algorithm 3.
  $\widetilde{O}((n+k)\sqrt{p})$ **rank** queries




**A new approach**

Edge Recycling Augmentation

# Outline

- Summary
- Preliminaries
  - -Matroid
  - -Matroid Intersection
  - -Matroid Partition
- Result
  - -Faster Matroid Partition Algorithms
- Idea
  - -Blocking Flow
  - -Edge Recycling Augmentation
- Conclusion

# Matroid $\mathcal{M} = (V, \mathcal{I})$

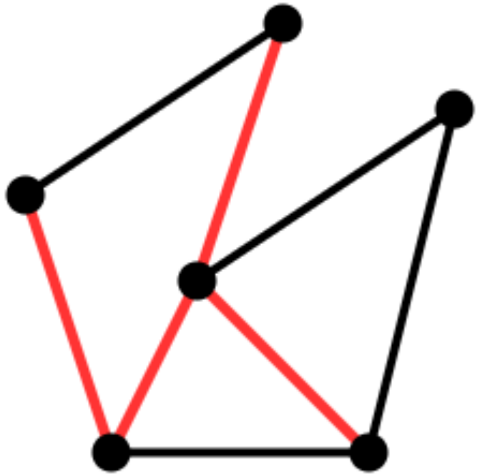
**Def**

A finite set $V$ and non-empty family of **independent** sets $\mathcal{I} \subseteq 2^V$ such that

- $S' \subseteq S \in \mathcal{I} \implies S' \in \mathcal{I}$
- $S, T \in \mathcal{I}, |S| > |T| \implies \exists\, e \in S - T$ s.t. $T \cup \{e\} \in \mathcal{I}$

**E.g.**

- **Graphic Matroid**



$V$ = edges
$\mathcal{I}$ = forests

- **Linear Matroid**

$$\begin{bmatrix} 0 & 1 & 2 & 0 \\ 3 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 1 & 2 & 3 & 0 \end{bmatrix}$$

$V$ = row vectors
$\mathcal{I}$ = linearly independent

# Matroid $\mathcal{M} = (V, \mathcal{I})$

**Def**

A finite set $V$ and non-empty family of **independent** sets $\mathcal{I} \subseteq 2^V$ such that

- $S' \subseteq S \in \mathcal{I} \implies S' \in \mathcal{I}$
- $S, T \in \mathcal{I}$, $|S| > |T| \implies \exists\, e \in S - T$ s.t. $\mathrm{T} \cup \{e\} \in \mathcal{I}$

Algorithm accesses a matroid through an **oracle**

# Matroid $\mathcal{M} = (V, \mathcal{I})$

**Def**

A finite set $V$ and non-empty family of **independent** sets $\mathcal{I} \subseteq 2^V$ such that

- $S' \subseteq S \in \mathcal{I} \implies S' \in \mathcal{I}$
- $S, T \in \mathcal{I}, |S| > |T| \implies \exists\, e \in S - T$ s.t. $T \cup \{e\} \in \mathcal{I}$

Algorithm accesses a matroid through an **oracle**

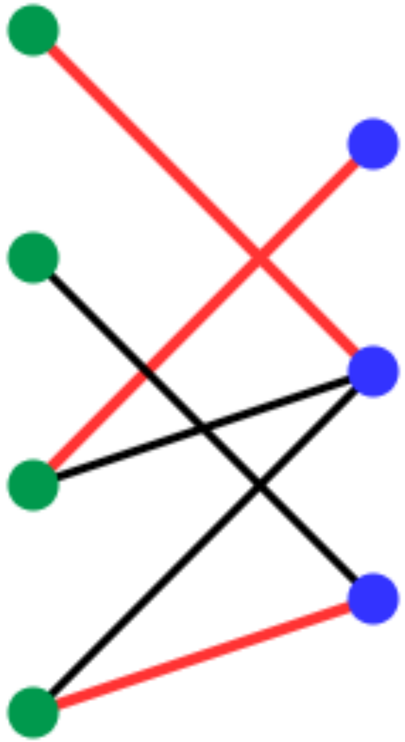
- **Independence** oracle query: Is $S \in \mathcal{I}$?

# Matroid Intersection

Input : two matroids $\mathcal{M}_1 = (V, \mathcal{I}_1), \mathcal{M}_2 = (V, \mathcal{I}_2)$
Find : maximum **common independent set** $S \in \mathcal{I}_1 \cap \mathcal{I}_2$

**E.g.** Bipartite Matching

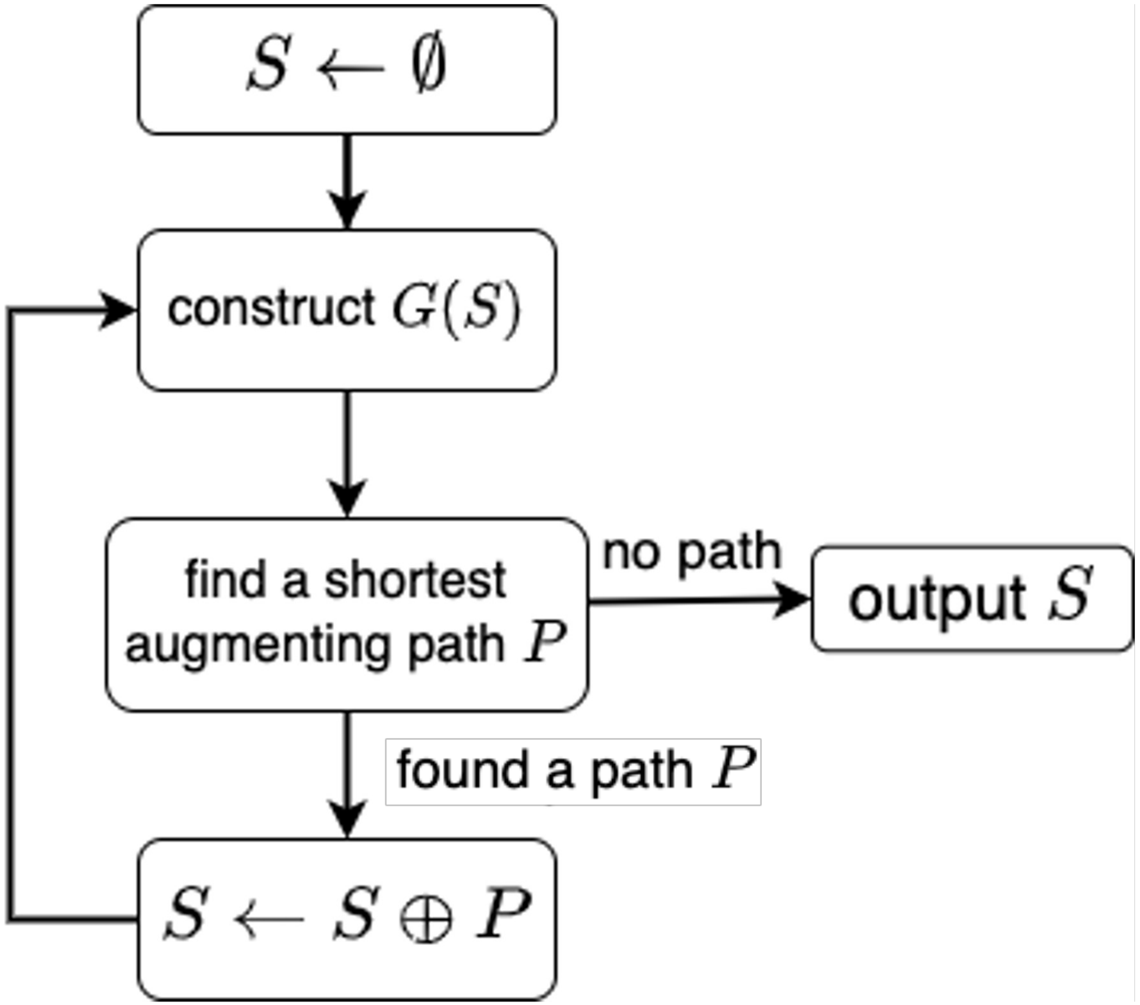
$\boldsymbol{V} =$ **edges**
$\boldsymbol{\mathcal{I}_1} =$ **each left vertex has at most 1 edge**
$\boldsymbol{\mathcal{I}_2} =$ **each right vertex has at most 1 edge**

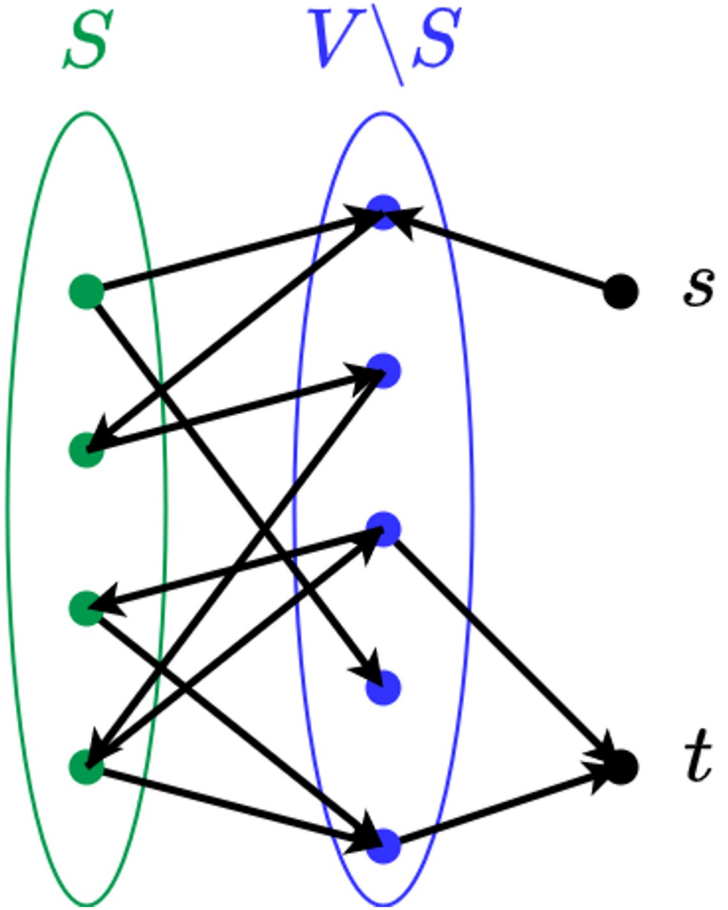


given: $(V, \mathcal{I}_1), (V, \mathcal{I}_2)$
max $|S|$ s.t. $S \in \mathcal{I}_1 \cap \mathcal{I}_2$

# Algorithm for Matroid Intersection

**[Edmonds 1970, Aigner-Dowling 1971, Lawler 1975]**



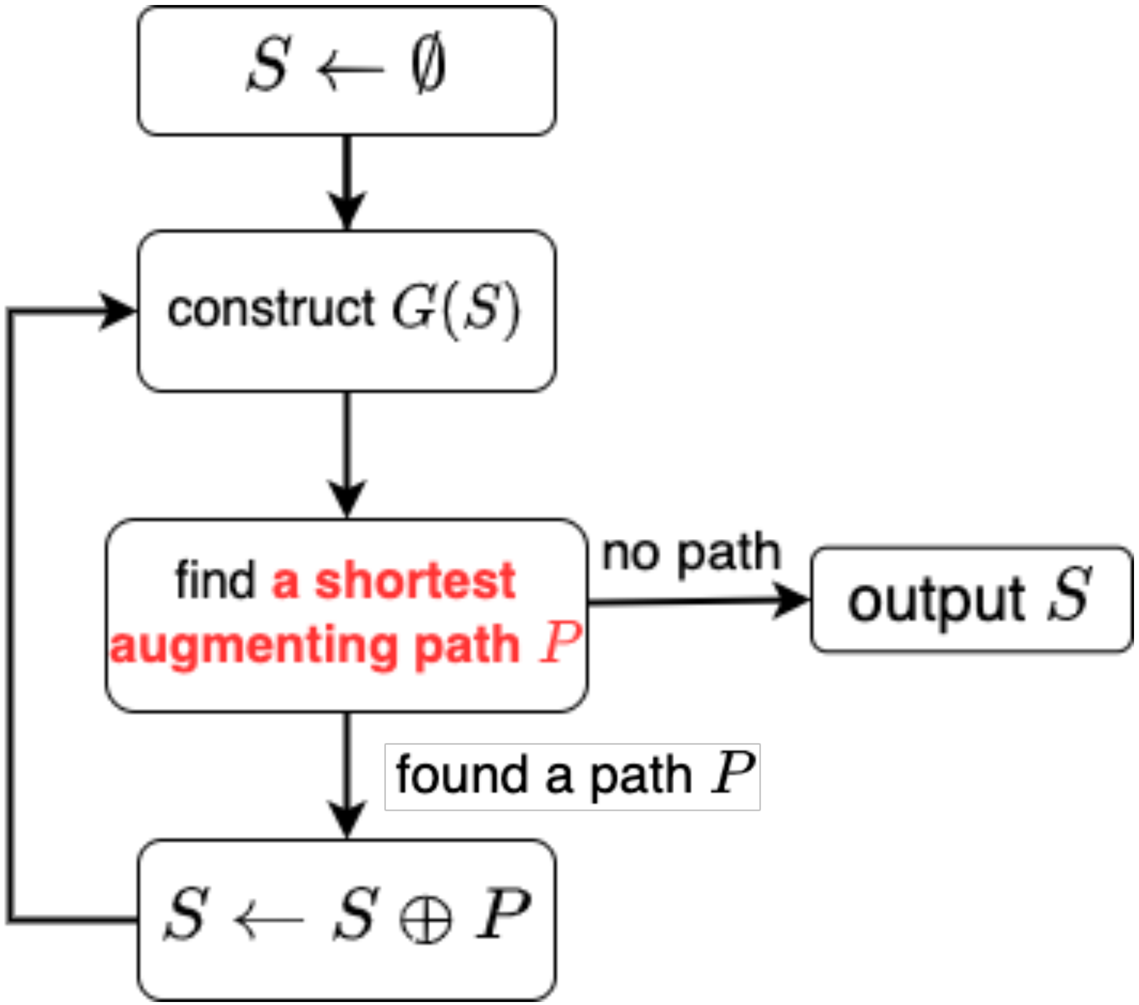



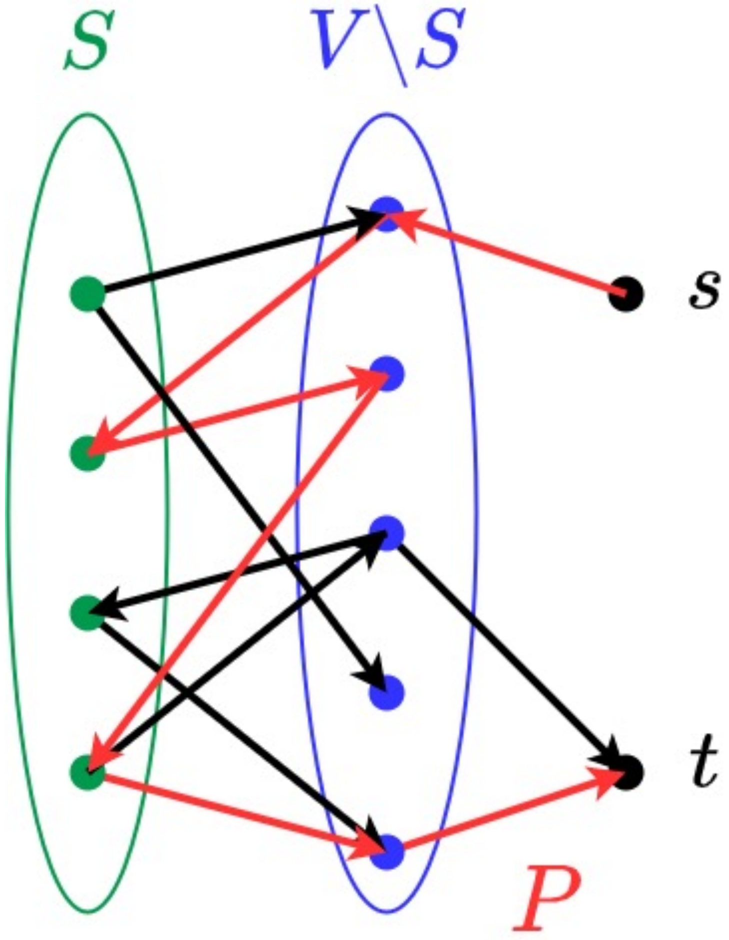

**Exchange graph $G(S)$**

given: $(V, \mathcal{I}_1), (V, \mathcal{I}_2)$
max $|S|$ s.t. $S \in \mathcal{I}_1 \cap \mathcal{I}_2$

# Algorithm for Matroid Intersection

**[Edmonds 1970, Aigner-Dowling 1971, Lawler 1975]**




**Exchange graph $G(S)$**

# Prior Work on Matroid Intersection

given: $(V, \mathcal{I}_1), (V, \mathcal{I}_2)$
max $|S|$ s.t. $S \in \mathcal{I}_1 \cap \mathcal{I}_2$
$n = |V|$, $r =$ sol. size

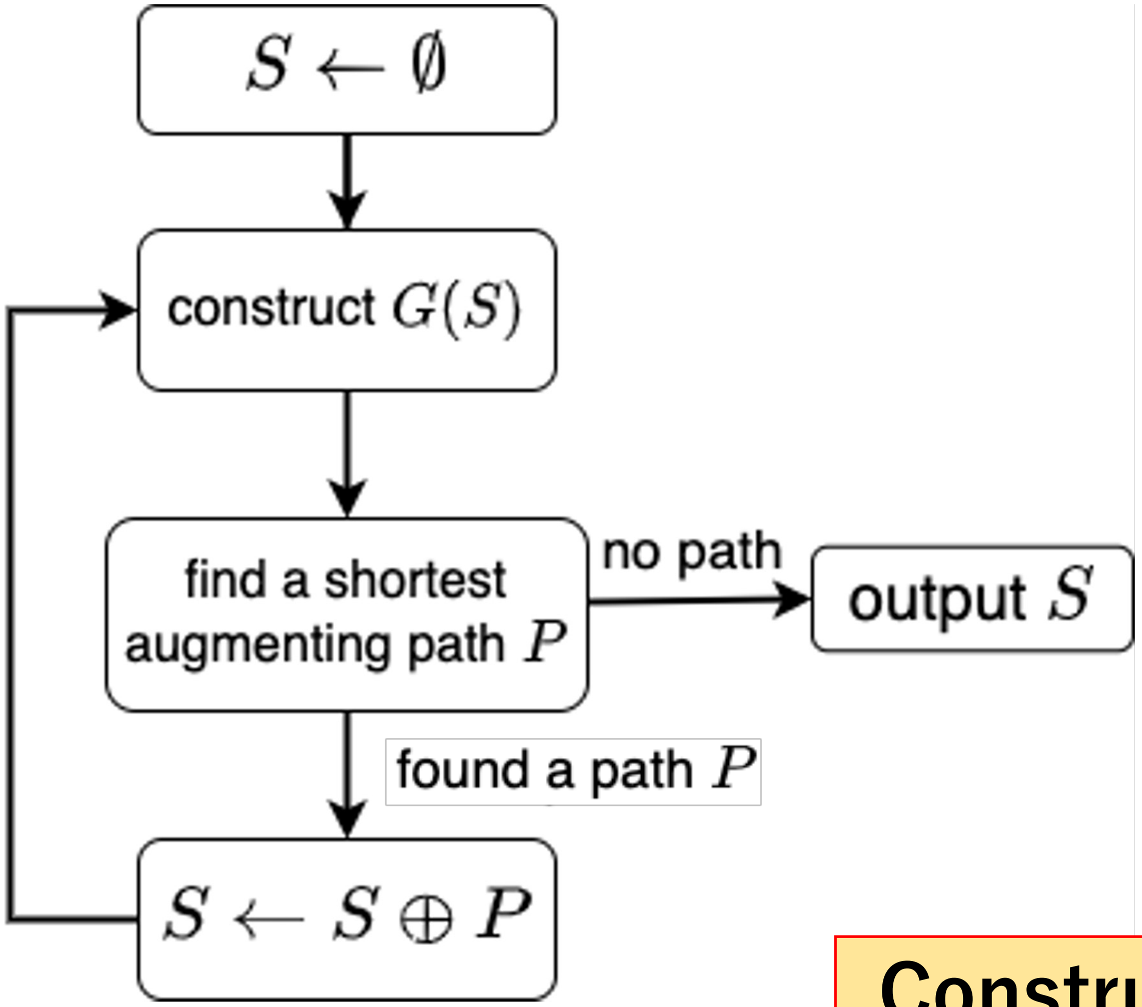
**Independence** query complexity

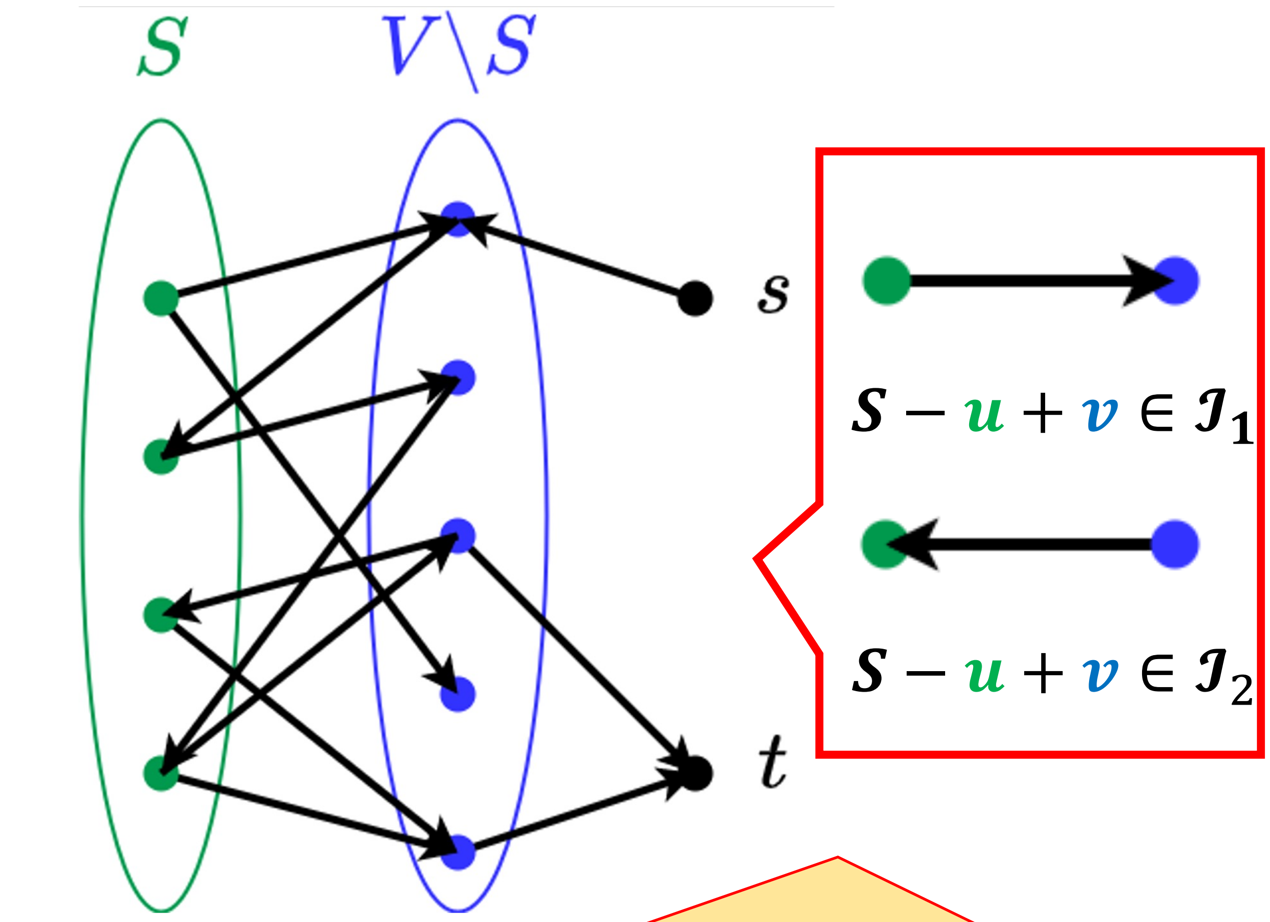
| | | |
|---|---|---|
| 1970s | Edmonds, Lawler, Aigner-Dowling | $O(nr^2)$ |
| 1986 | Cunningham | $O(nr^{3/2})$ |
| 2015 | Lee-Sidford-Wong | $\tilde{O}(n^2)$ |
| 2019 | Nguyễn, Chakrabarty-Lee-Sidford-Singla-Wong | $\tilde{O}(nr)$ |
| 2021 | Blikstad-v.d.Brand-Mukhopadhyay-Nanongkai | $\tilde{O}(n^{9/5})$ |
| 2021 | Blikstad | $\tilde{O}(nr^{3/4})$ |

# Algorithm for Matroid Intersection

given: $(V,\mathcal{I}_1),(V,\mathcal{I}_2)$
max $|S|$ s.t. $S \in \mathcal{I}_1 \cap \mathcal{I}_2$

**[Edmonds 1970, Aigner-Dowling 1971, Lawler 1975]**

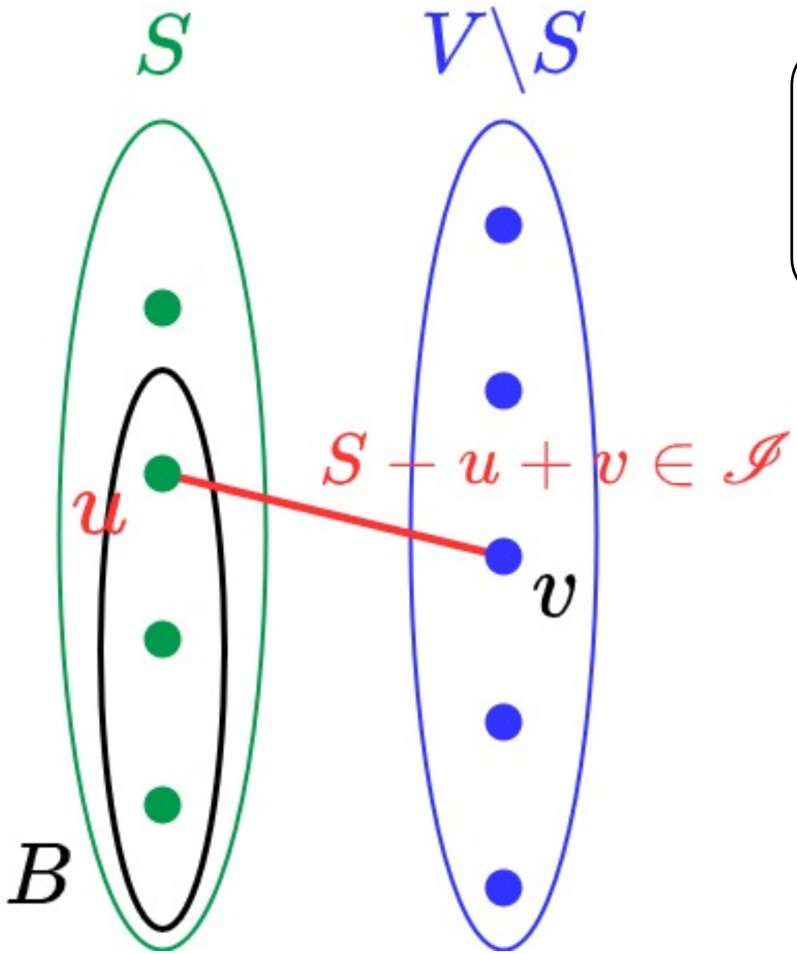



**Construct exchange graph $G(S)$ explicitly**

# Tool for Faster Matroid Intersection

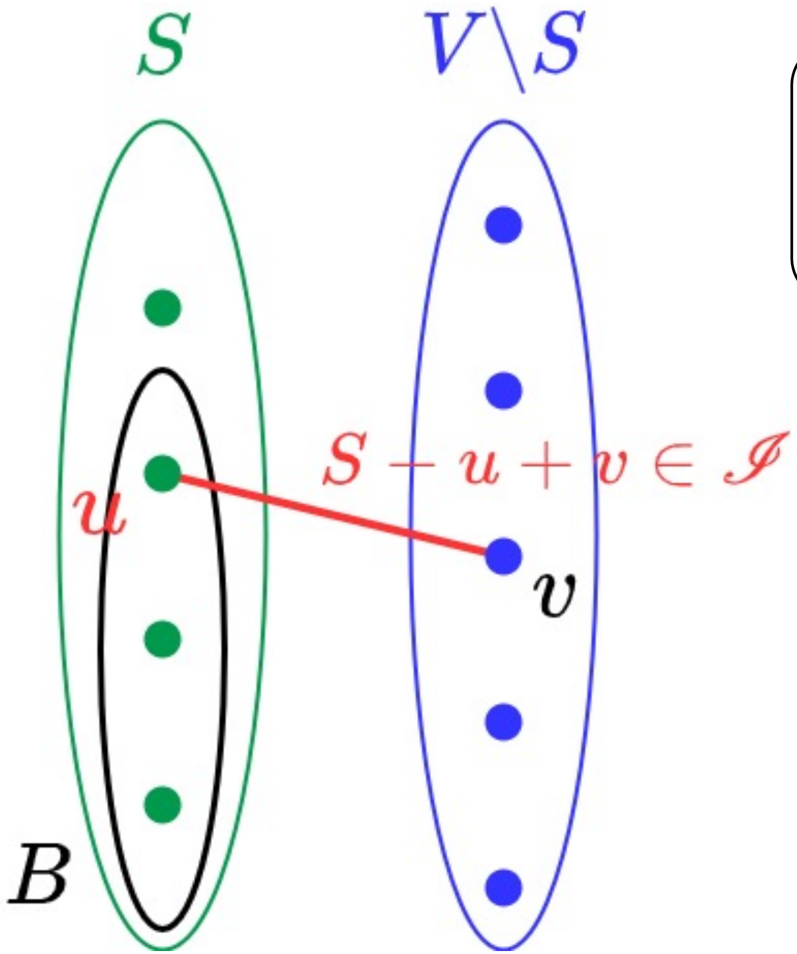
**[Nguyễn 2019, Chakrabarty et al. 2019]**




Input：$\mathcal{M} = (V, \mathcal{I}),\ S \in \mathcal{I},\ v \in V \setminus S,\ B \subseteq S$
Find ：$u \in B$ s.t. $S - u + v \in \mathcal{I}$

# Tool for Faster Matroid Intersection

**[Nguyễn 2019, Chakrabarty et al. 2019]**

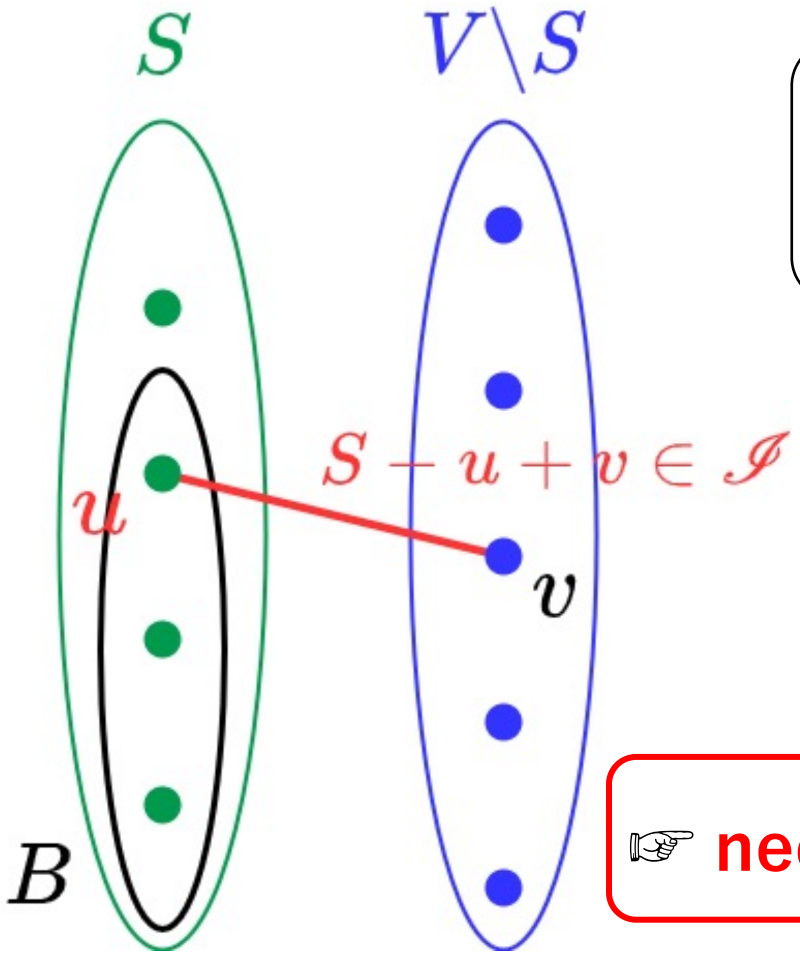



Input : $\mathcal{M} = (V, \mathcal{I}),\ S \in \mathcal{I},\ v \in V \setminus S,\ B \subseteq S$
Find : $u \in B$ s.t. $S - u + v \in \mathcal{I}$

$O(\log|B|)$ independence query using **binary search**

# Tool for Faster Matroid Intersection

[**Nguyễn 2019, Chakrabarty et al. 2019**]




Input : $\mathcal{M} = (V, \mathcal{I})$, $S \in \mathcal{I}$, $v \in V \setminus S$, $B \subseteq S$
Find : $u \in B$ s.t. $S - u + v \in \mathcal{I}$

$O(\log|B|)$ independence query using **binary search**

☞ **need not** construct **exchange graph $G(S)$** **explicitly**

# Matroid Partition

Input : $\boldsymbol{k}$ **matroids** $\mathcal{M}_1 = (V, \mathcal{I}_1), \ldots, \mathcal{M}_k = (V, \mathcal{I}_k)$
Find : maximum **partitionable** set $S \subseteq V$

There exists a **partition** $\boldsymbol{S = S_1 \cup \cdots \cup S_k}$ s.t. $\boldsymbol{S_i \in \mathcal{I}_i}$

# Matroid Partition

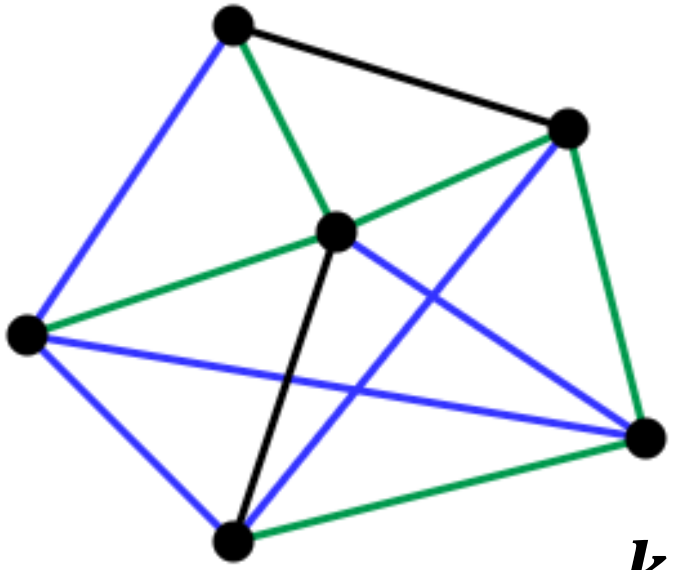
Input : ***k* matroids** $\mathcal{M}_1 = (V, \mathcal{I}_1), \ldots, \mathcal{M}_k = (V, \mathcal{I}_k)$

Find : maximum **partitionable** set $S \subseteq V$

There exists a partition $S = S_1 \cup \cdots \cup S_k$ s.t. $S_i \in \mathcal{I}_i$

**E.g.** $k$-forest

Find a maximum-size union of $k$ forests



$\boldsymbol{k = 2}$

# Matroid Partition and Matroid Intersection

Matroid partition can be solved by **the reduction to matroid intersection**

☞ Intersection of two matroids on $\boldsymbol{V} \times \{\mathbf{1}, \dots, \boldsymbol{k}\}$

# Matroid Partition and Matroid Intersection

Matroid partition can be solved by **the reduction to matroid intersection**

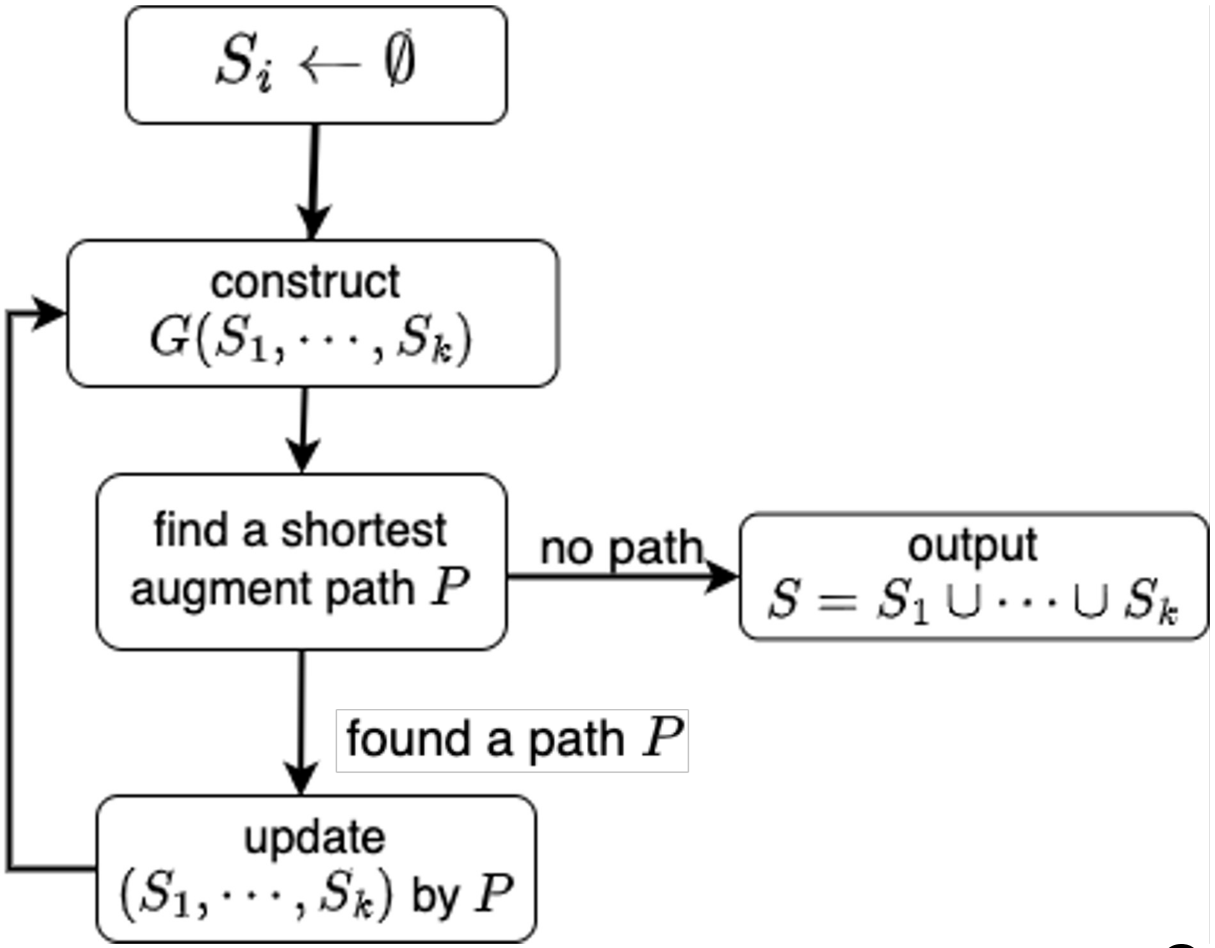
☞ Intersection of two matroids on $\boldsymbol{V} \times \{\mathbf{1}, \dots, \boldsymbol{k}\}$

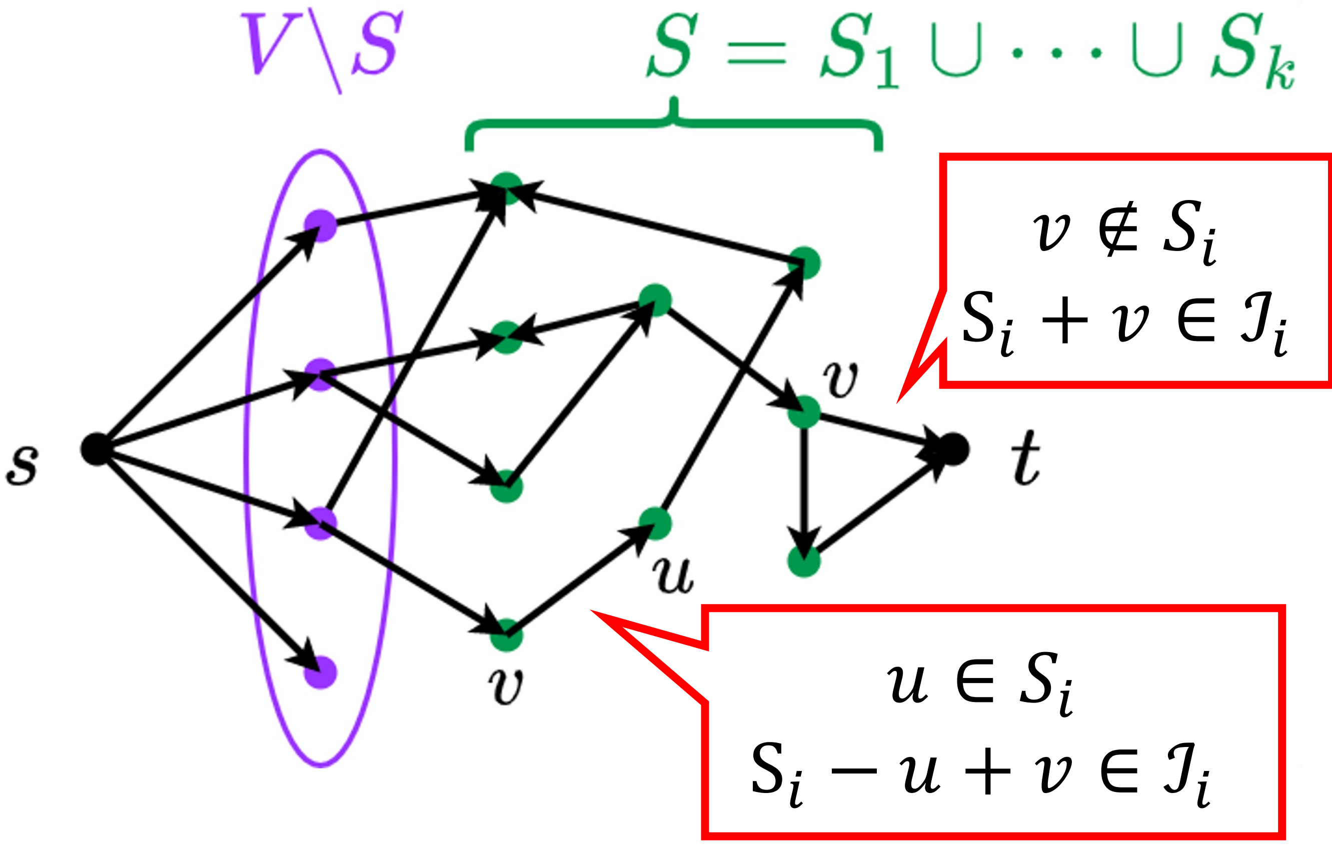
The size of ground set is $\boldsymbol{kn}$: large ⇨ **too many queries !**

given: $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$

# Algorithm for Matroid Partition

**[Edmonds 1968]**



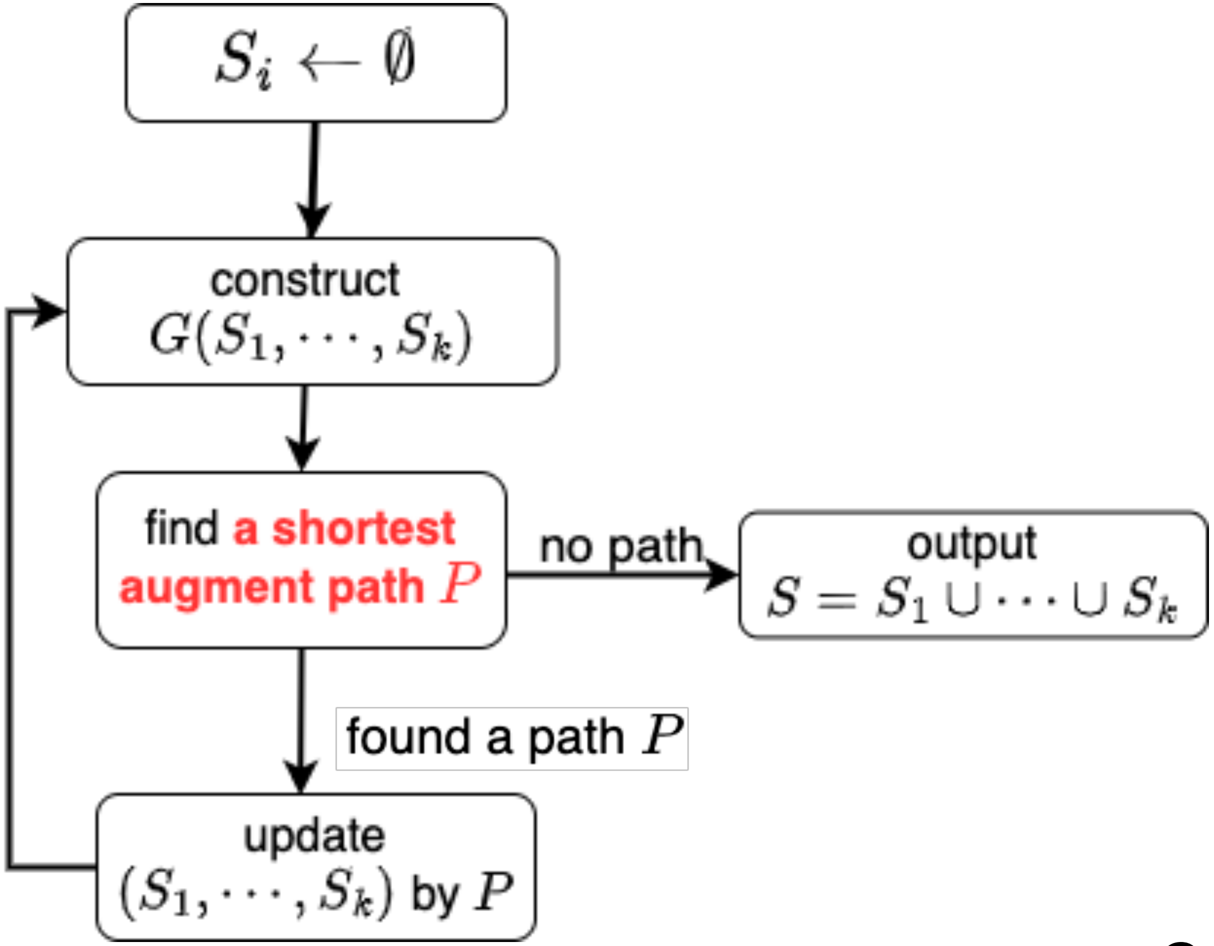



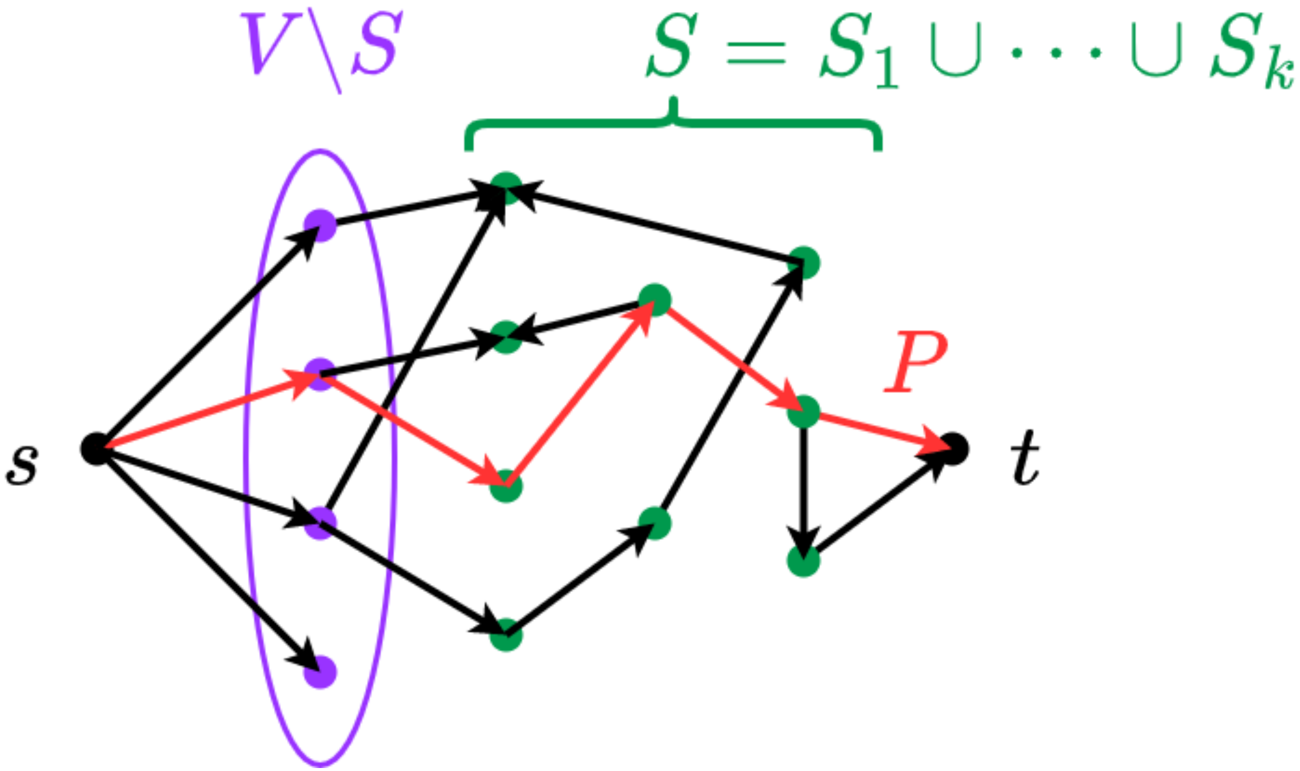

**Compressed exchange graph $G(S_1, \dots, S_k)$**

given: $(V, \mathcal{I}_1), \ldots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$

# Algorithm for Matroid Partition

**[Edmonds 1968]**




**Compressed exchange graph $G(S_1, \ldots, S_k)$**

given: $(V, \mathcal{I}_1), \ldots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $k$ = #matroids, $p$ = sol. size

# Prior Work on Matroid Partition

**Independence** query Complexity

| | | |
|---|---|---|
| 1968 | Edmonds | $O(np^2 + kn)$ |
| 1986 | Cunningham | $O(np^{3/2} + kn)$ |

# Prior Work on Matroid Partition

given: $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $k = $ #matroids, $p = $ sol. size

**Independence** query Complexity

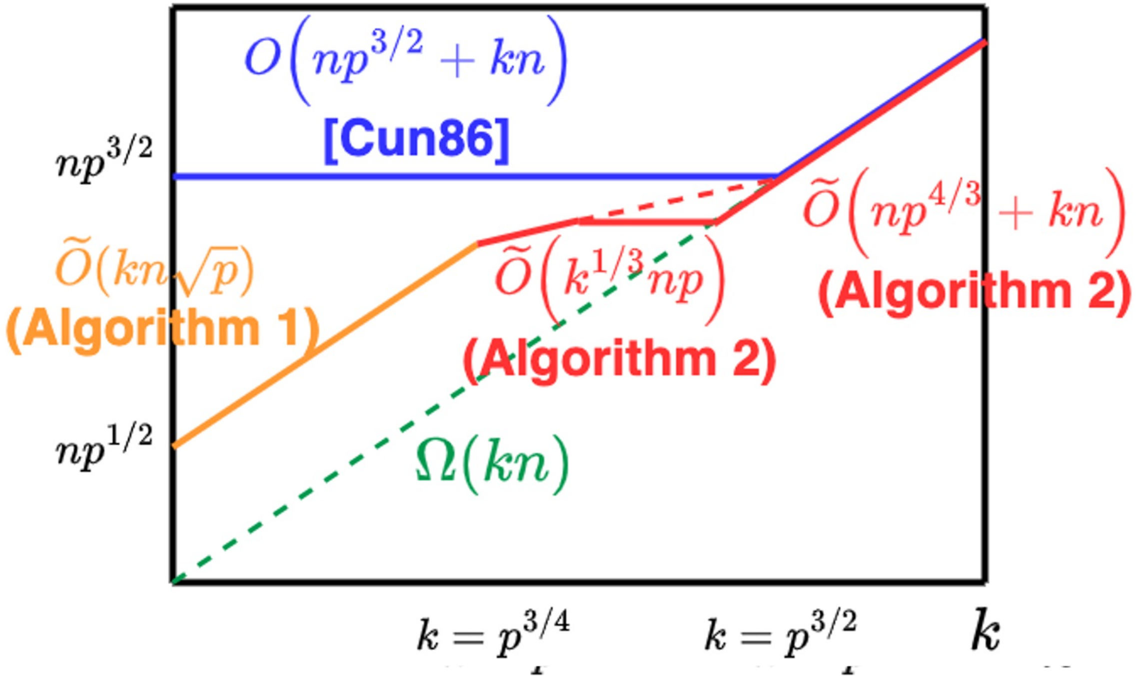
| | | |
|---|---|---|
| 1968 | Edmonds | $O(np^2 + kn)$ |
| 1986 | Cunningham | $O(np^{3/2} + kn)$ |
| **2023** | **This work** | $\tilde{O}(kn\sqrt{p})$ |
| **2023** | **This work** | $\tilde{O}(k^{1/3}np + kn)$ |

given: $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $p =$ sol. size, $k' = \min\{k, p\}$

# Prior Work on Matroid Partition

**Independence** query complexity

| | | |
|---|---|---|
| 1968 | Edmonds | $O(np^2 + kn)$ |
| 1986 | Cunningham | $O(np^{3/2} + kn)$ |
| 2023 | This work | $\tilde{O}(kn\sqrt{p})$ |
| 2023 | This work | $\tilde{O}(k'^{1/3}np + kn)$ |



given: $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $k = $ #matroids, $p = $ sol. size

# Algorithm 1: Blocking Flow

Thm1

Matroid partition can be solved using $\tilde{O}(kn\sqrt{p})$ **independence** queries

given: $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $k =$ #matroids, $p =$ sol. size

# Algorithm 1: Blocking Flow

Thm1

Matroid partition can be solved using $\tilde{O}(kn\sqrt{p})$ **independence** queries

Idea

**Blocking Flow** [Cunningham 1986]
☞ akin to Hopcroft-Karp / Dinic

+

**Binary Seach**
[Nguyễn 2019, Chakrabarty et al. 2019]

Finding **multiple** augmenting paths **of the same length** in one phase

given: $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $k =$ #matroids, $p =$ sol. size

# Algorithm 1: Blocking Flow

<u>Thm1</u>

Matroid partition can be solved using $\tilde{O}(kn\sqrt{p})$ **independence** queries

<u>Algorithm</u>

Repeat:

Step 1: Breadth First Search

Step 2: Find multiple augmenting paths

# Algorithm 1: Blocking Flow

given: $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $k = \#$matroids, $p =$ sol. size

Thm1
Matroid partition can be solved using $\tilde{O}(kn\sqrt{p})$ **independence** queries

Algorithm

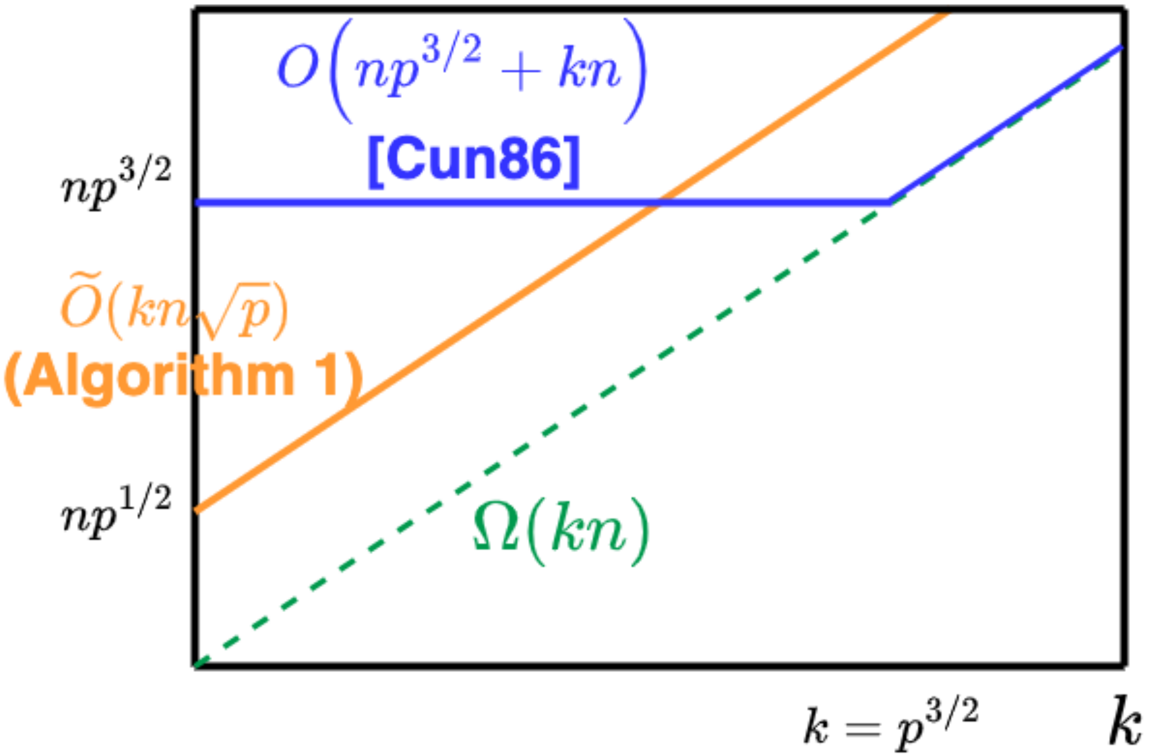
Repeat:
- Step 1: Breadth First Search ← $\tilde{O}(kn)$ **queries**
- Step 2: Find multiple augmenting paths ← $\tilde{O}(kn)$ **queries**

Fact: $\Theta(\sqrt{p})$ phases are required

# Algorithm 1: Blocking Flow

given: $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $k =$ #matroids, $p =$ sol. size

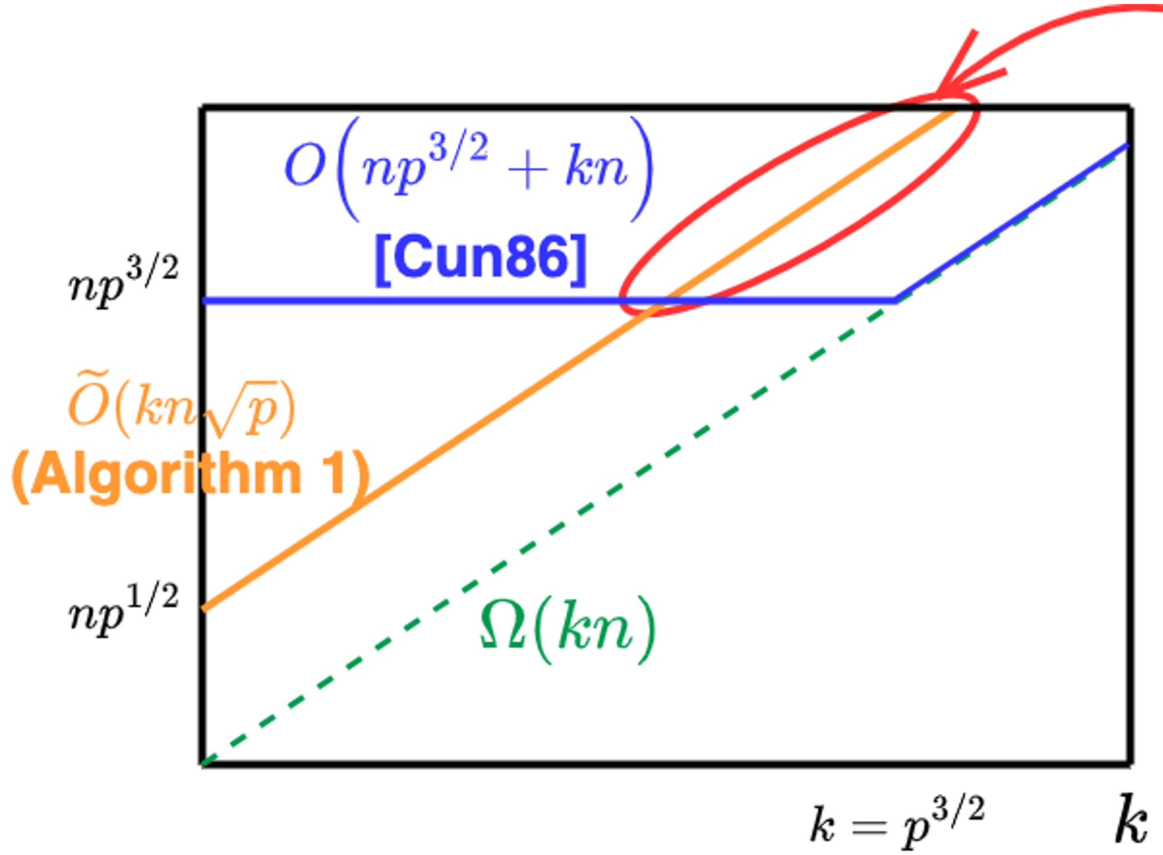
Thm1
Matroid partition can be solved using $\tilde{O}(kn\sqrt{p})$ **independence** queries



# Algorithm 1: Blocking Flow

given: $(V, \mathcal{I}_1), \ldots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $k$ = #matroids, $p$ = sol. size

Thm1
Matroid partition can be solved using $\tilde{O}(kn\sqrt{p})$ **independence** queries

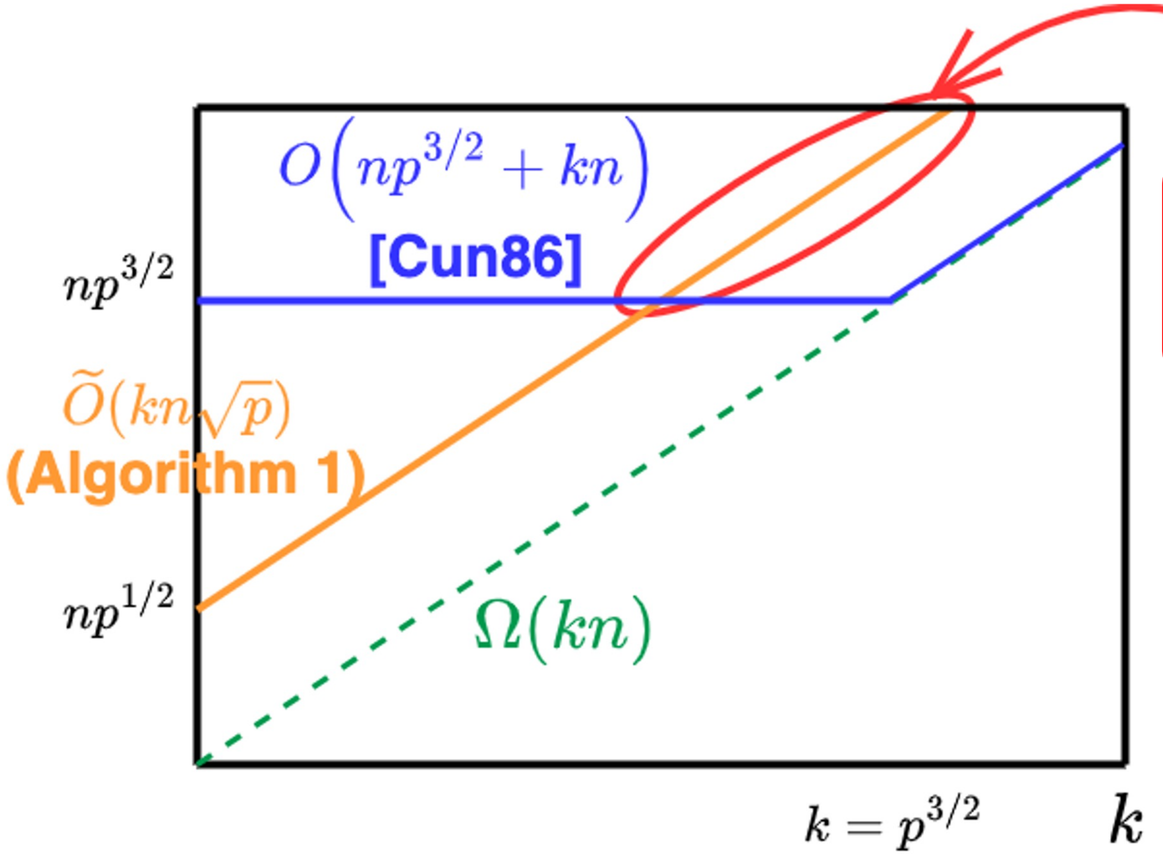



Despite of **binary search** technique, Alg. 1 is worse than [Cun 86].

given: $(V, \mathcal{I}_1), \ldots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $k$ = #matroids, $p$ = sol. size

# Algorithm 1: Blocking Flow

Thm1
Matroid partition can be solved using $\tilde{O}(kn\sqrt{p})$ **independence** queries




Despite of binary search technique, Alg. 1 is worse than [Cun 86].

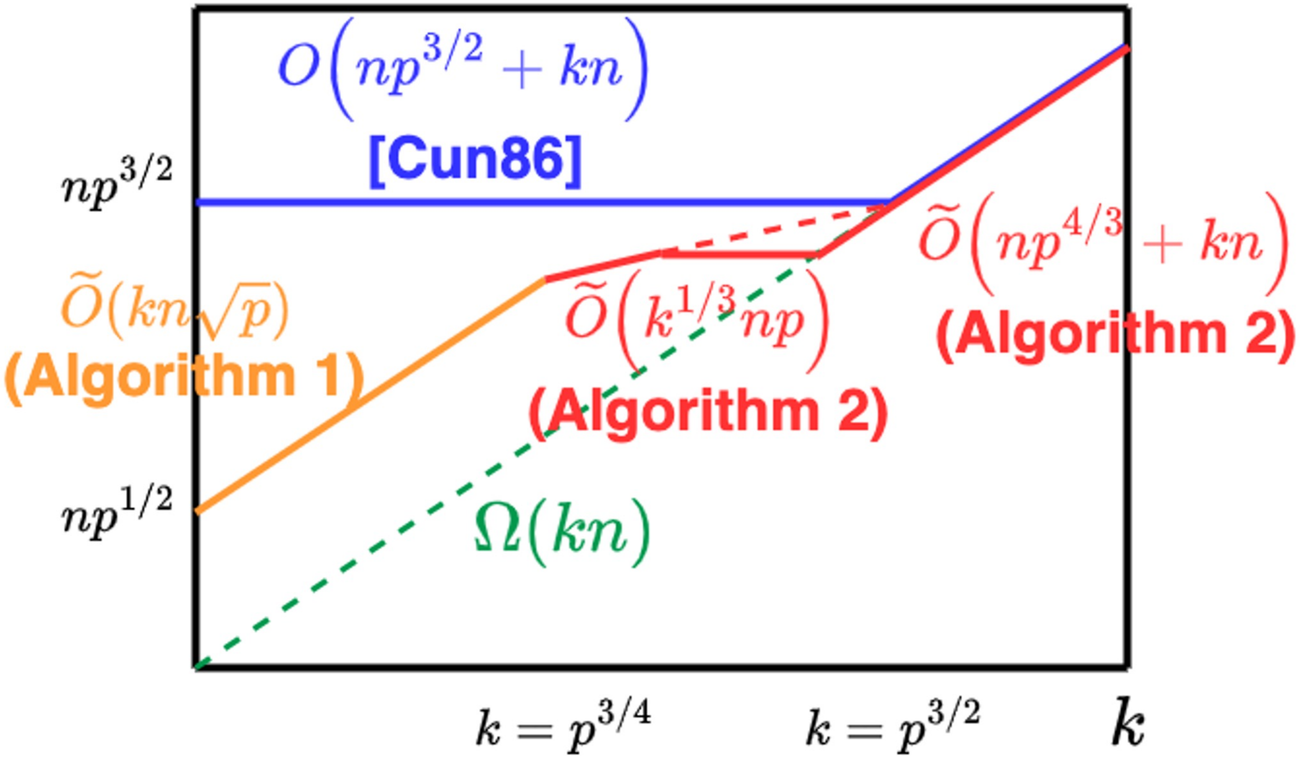
Q. Better Algorithm **when $k$ is large** ?

# Algorithm 2

given: $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $p =$ sol. size, $k' = \min\{k, p\}$

Thm2

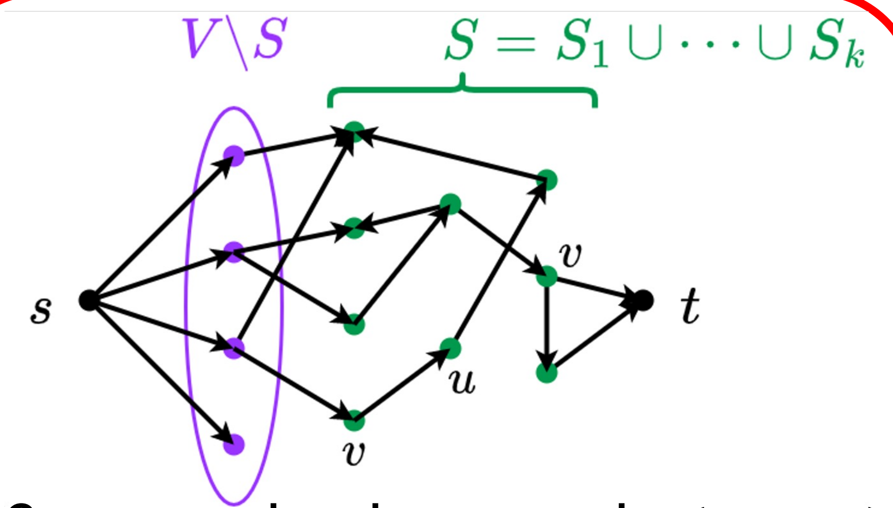
Matroid partition can be solved using $\tilde{O}(k'^{1/3}np + kn)$ **independence** queries



# One Phase of Edge Recycling Augmentation

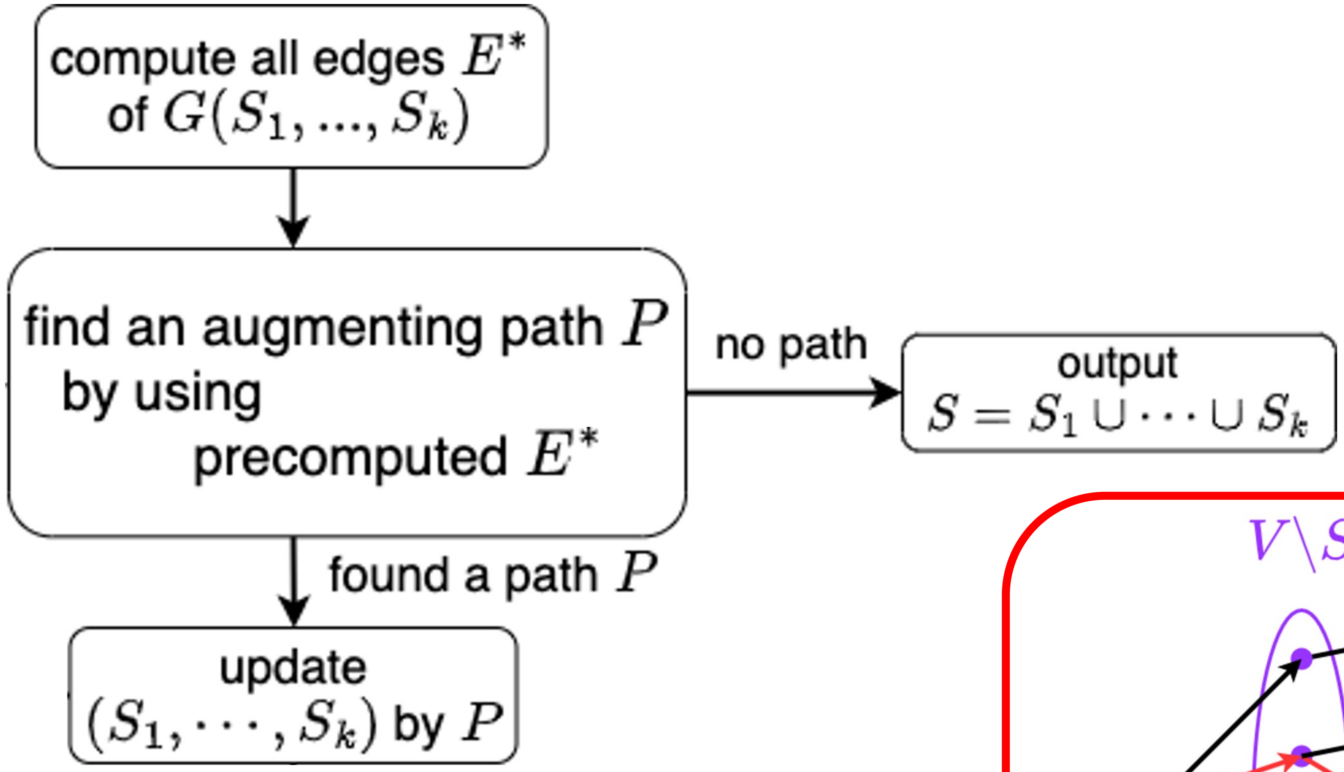
compute all edges $E^*$ of $G(S_1, \dots, S_k)$

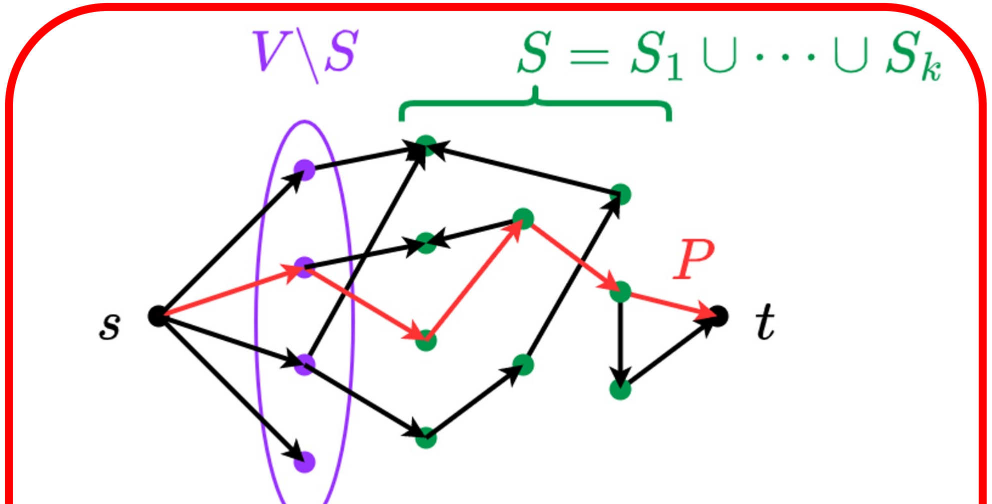
$\boldsymbol{O(np)}$ queries




Compressed exchange graph $\boldsymbol{G}(S_1, \dots, S_k)$

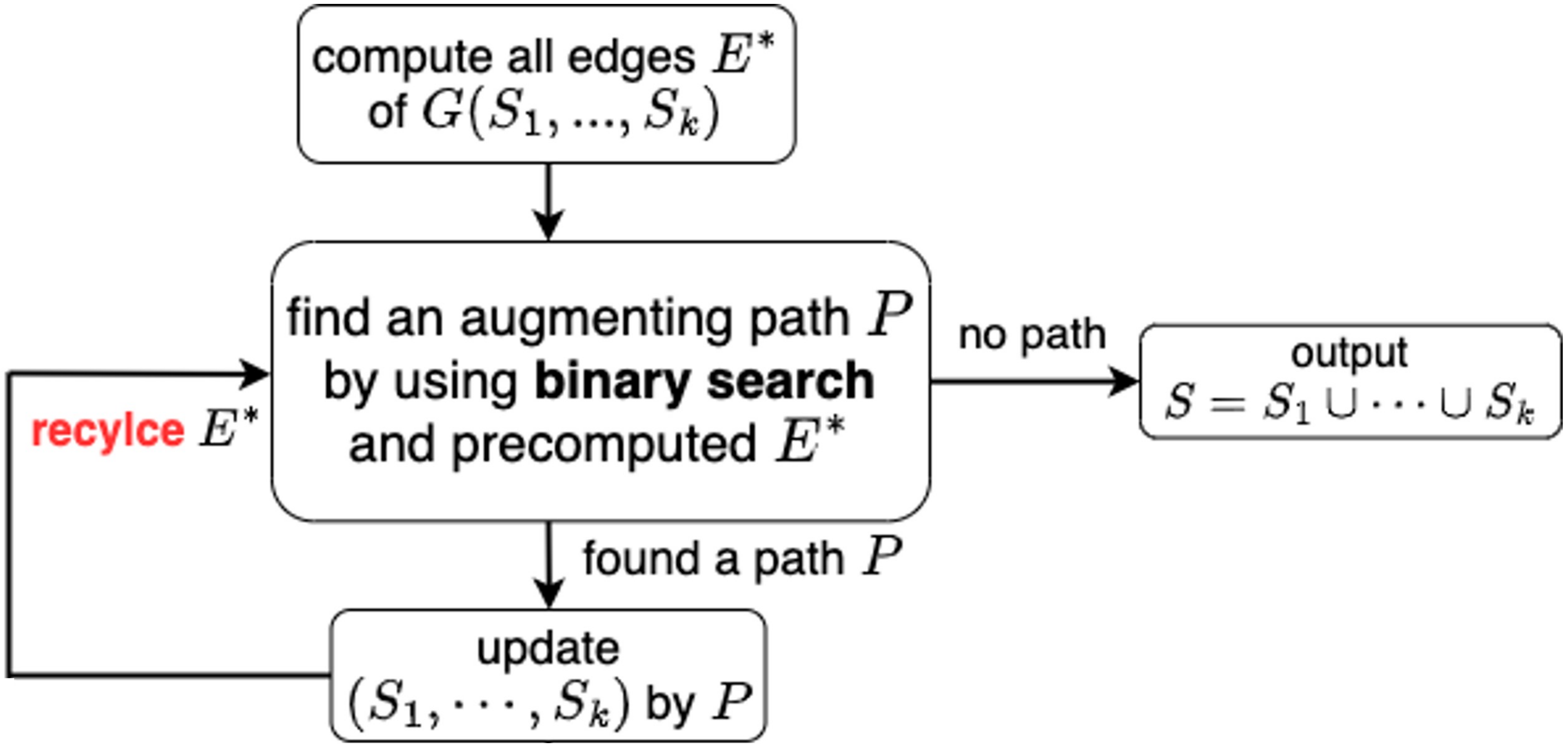
# One Phase of Edge Recycling Augmentation



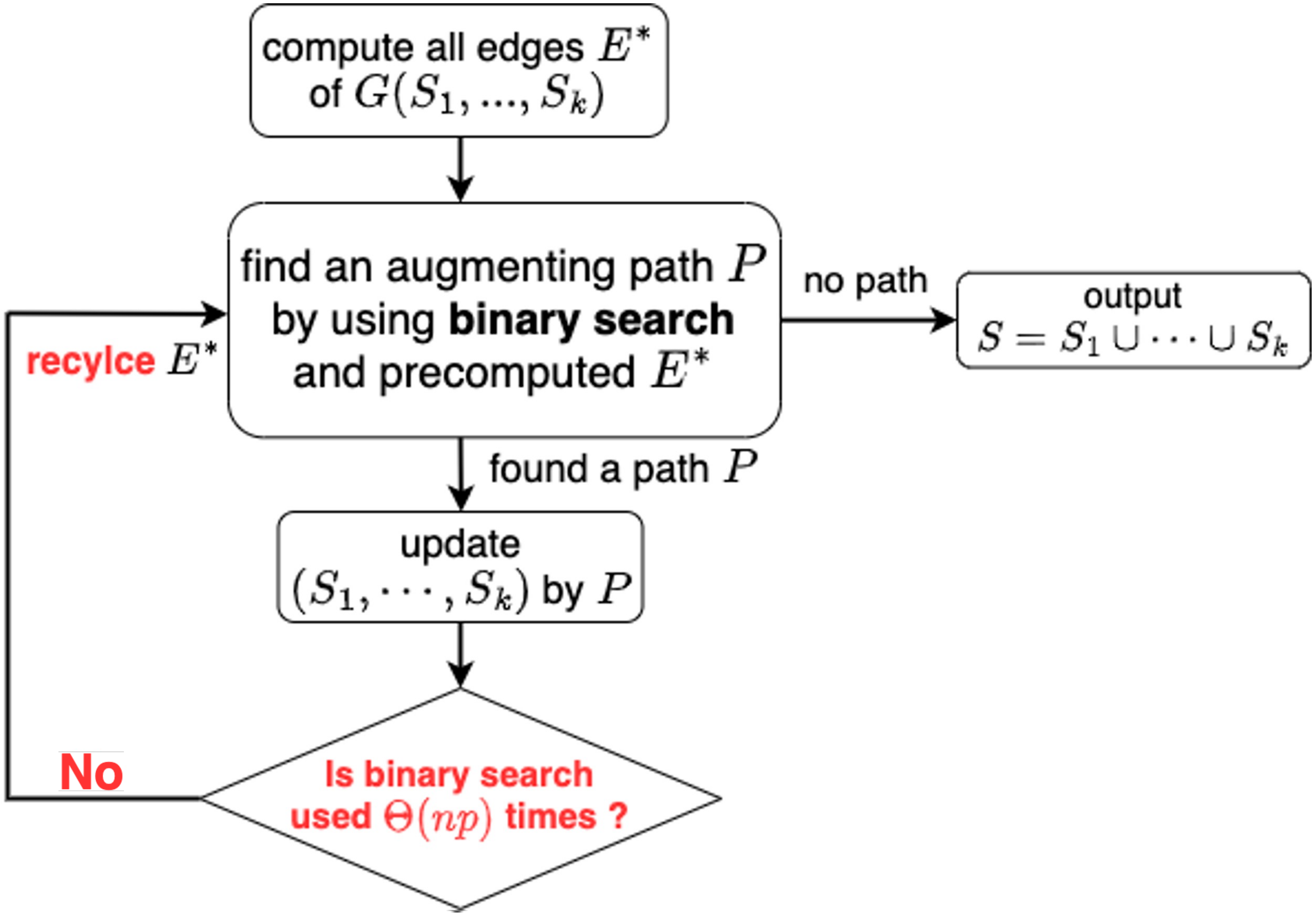




Compressed exchange graph $G(S_1, \ldots, S_k)$

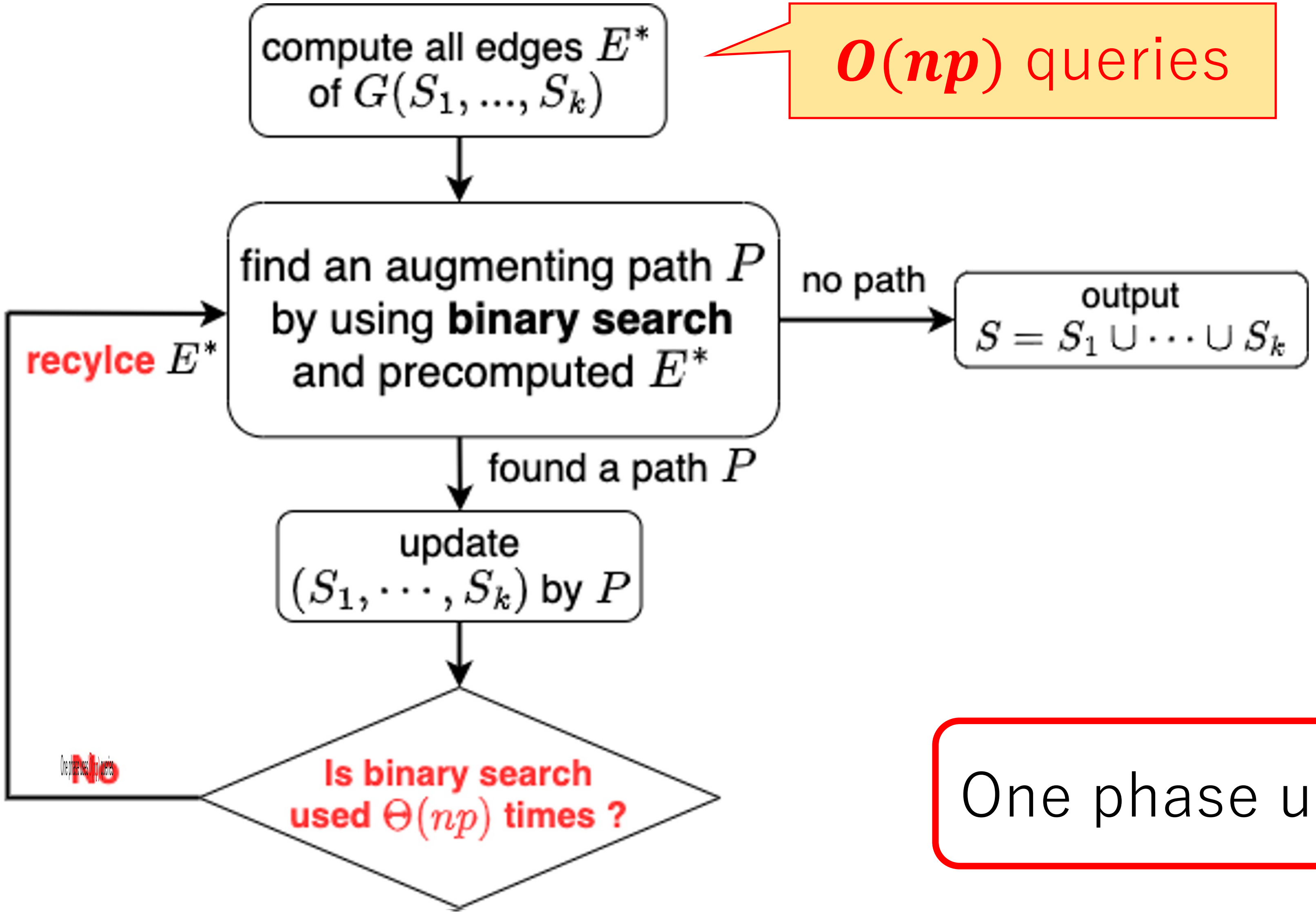
# One Phase of Edge Recycling Augmentation



# One Phase of Edge Recycling Augmentation



# One Phase of Edge Recycling Augmentation




One phase uses $\boldsymbol{\tilde{O}(np)}$ queries

# Algorithm 2: Hybrid Approach

given: $(V,\mathcal{I}_1),\dots,(V,\mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $p =$ sol. size, $k'$=min$\{k,p\}$

Thm2

Matroid partition can be solved using $\tilde{O}(k'^{1/3}np + kn)$ **independence** queries

# Algorithm 2: Hybrid Approach

given: $(V, \mathcal{I}_1), \dots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $p =$ sol. size, $k'$=min$\{k, p\}$

Thm2
Matroid partition can be solved using $\tilde{O}(k'^{1/3}np + kn)$ **independence** queries

Step 1. Apply **Blocking Flow** (Algorithm 1)

# Algorithm 2: Hybrid Approach

given: $(V,\mathcal{I}_1),\dots,(V,\mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $p =$ sol. size, $k' = \min\{k,p\}$

Thm2
Matroid partition can be solved using $\tilde{O}(k'^{1/3}np + kn)$ **independence** queries

Step 1. Apply Blocking Flow (Algorithm 1)

Step 2. Apply **Edge Recycling Augmentation**

given: $(V, \mathcal{I}_1), \ldots, (V, \mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $p =$ sol. size, $k' = \min\{k, p\}$

# Algorithm 2: Hybrid Approach

Thm2
Matroid partition can be solved using $\tilde{O}(k'^{1/3}np + kn)$ **independence** queries

Step 1. Apply **Blocking Flow** (Algorithm 1) in $\Theta(\frac{p}{k'^{2/3}})$ **phases**

Step 2. Apply Edge Recycling Augmentation

given: $(V,\mathcal{I}_1),\dots,(V,\mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $p$ = sol. size, $k'$=min$\{k,p\}$

# Algorithm 2: Hybrid Approach

<u>Thm2</u>
Matroid partition can be solved using $\tilde{O}(k'^{1/3}np + kn)$ **independence** queries

Step 1. Apply Blocking Flow (Algorithm 1) in $\Theta(\frac{p}{k'^{2/3}})$ **phases**

Step 2. Apply **Edge Recycling Augmentation**

Lemma: $\Theta(k'^{1/3})$ phases are required in Step 2

# Algorithm 2: Hybrid Approach

given: $(V,\mathcal{I}_1),\ldots,(V,\mathcal{I}_k)$
max $|S_1 \cup \cdots \cup S_k|$ s.t. $S_i \in \mathcal{I}_i (\forall i)$
$n = |V|$, $p =$ sol. size, $k' = \min\{k, p\}$

Thm2
Matroid partition can be solved using $\tilde{O}(k'^{1/3}np + kn)$ **independence** queries

Step 1. Apply Blocking Flow (Algorithm 1) in $\Theta(\frac{p}{k'^{2/3}})$ **phases**

One phase uses $\tilde{O}(k'n)$ queries

Step 2. Apply **Edge Recycling Augmentation**

One phase uses $\tilde{O}(np)$ queries

Lemma: $\Theta(k'^{1/3})$ phases are required in Step 2

# Conclusion

Improve the **independence** query complexity of **Matroid Partition**

- Use **Binary Search Technique** [Nguyễn 2019, Chakrabarty et al. 2019]
- A new approach: **Edge Recycling Augmentation**

Q. Further improvement?

Q. Apply an idea of Edge Recycling Augmentation to other problems?