マトロイド分割問題に対する高速なアルゴリズム

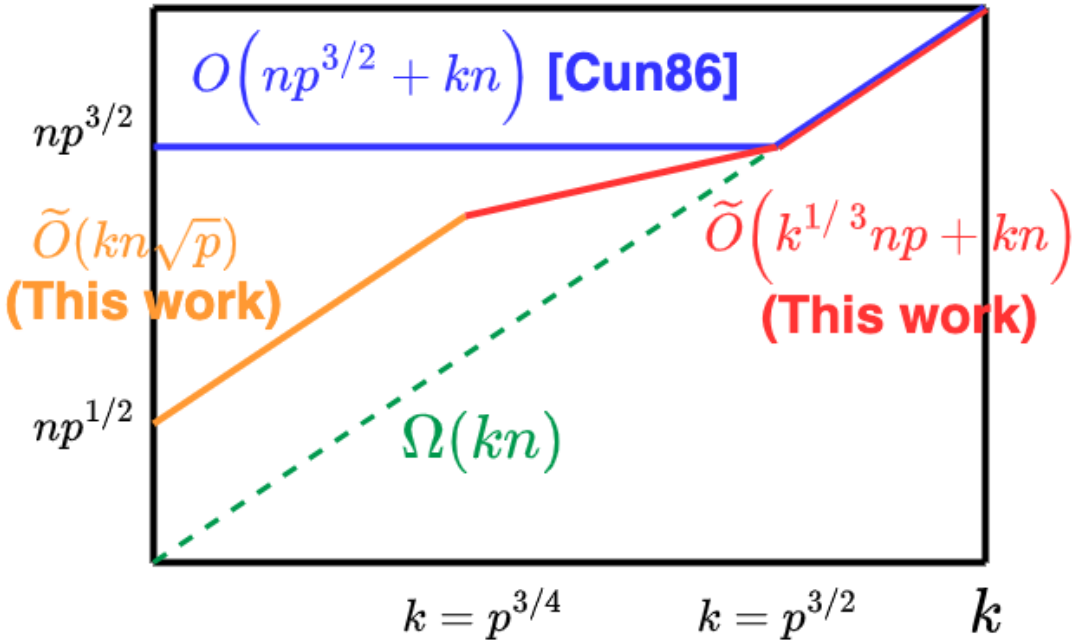
寺尾 樹哉
京都大学数理解析研究所 M2

離散数学とその応用研究集会 2023 (JCCA2023)@愛知教育大
8月30日(水)

発表概要

主結果
マトロイド分割問題を高速に解く3つのアルゴリズムを設計

- アルゴリズム①
  独立性オラクル $\tilde{O}(kn\sqrt{p})$回クエリ
- アルゴリズム②
  独立性オラクル $\tilde{O}(k^{1/3}np + kn)$回クエリ
- アルゴリズム③
  ランクオラクル $\tilde{O}((n+k)\sqrt{p})$回クエリ

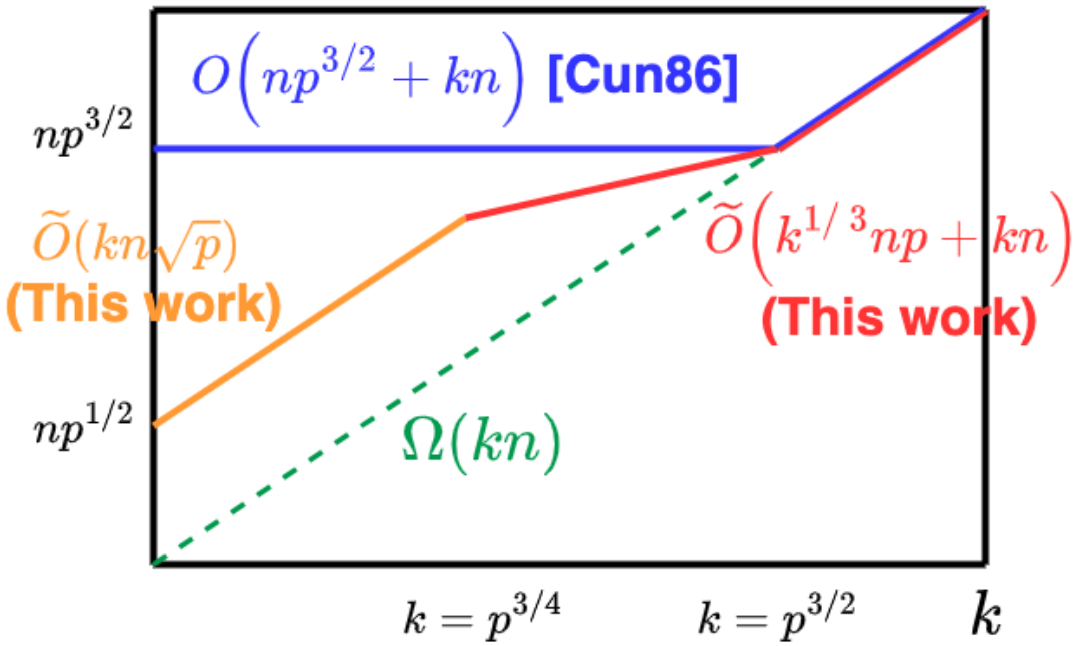


Tatsuya Terao：**Faster Matroid Partition Algorithms,** Proceedings of the 50th EATCS International Colloquium on Automata, Languages and Programming (**ICALP 2023**). (arXiv: 2303.05920)

発表概要

主結果
マトロイド分割問題を高速に

[Cunningham,1986]から約40年ぶり



- アルゴリズム①
  独立性オラクル $\tilde{O}(kn\sqrt{p})$回クエリ
- アルゴリズム②
  独立性オラクル $\tilde{O}(k^{1/3}np+kn)$回クエリ
- アルゴリズム③
  ランクオラ

辺再利用型増加路探索

Tatsuya Terao：**Faster Matroid Partition Algorithms,** Proceedings of the 50th EATCS International Colloquium on Automata, Languages and Programming (**ICALP 2023**). (arXiv: 2303.05920)

目次

マトロイド $\mathcal{M} = (V, \mathcal{I})$: 線形独立性の一般化

$\mathcal{I}$の要素を**独立**集合と呼ぶ

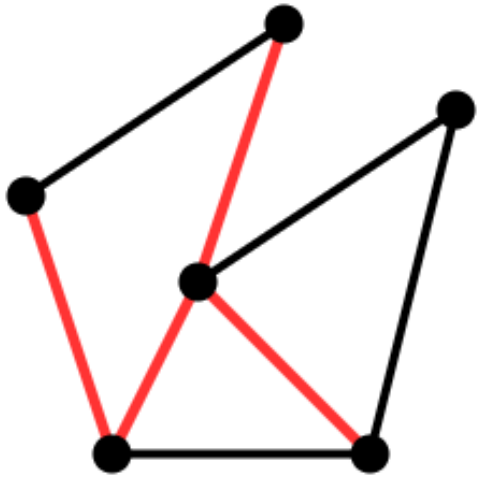
定義

有限集合 V 上の空でない部分集合族 $\mathcal{I} \subseteq 2^V$ で次のよい性質を持つもの

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

例）

- **グラフ的マトロイド**



V ＝辺集合
$\mathcal{I}$ ＝森全体

- **線形マトロイド**

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

V ＝行ベクトル
$\mathcal{I}$ ＝線形独立

マトロイド $\mathcal{M} = (V, \mathcal{I})$

$\mathcal{I}$の要素を独立集合と呼ぶ

有限集合 V 上の部分集合族 $\mathcal{I}$ で良い性質を持つもの

Q. マトロイドをどう扱うのか？

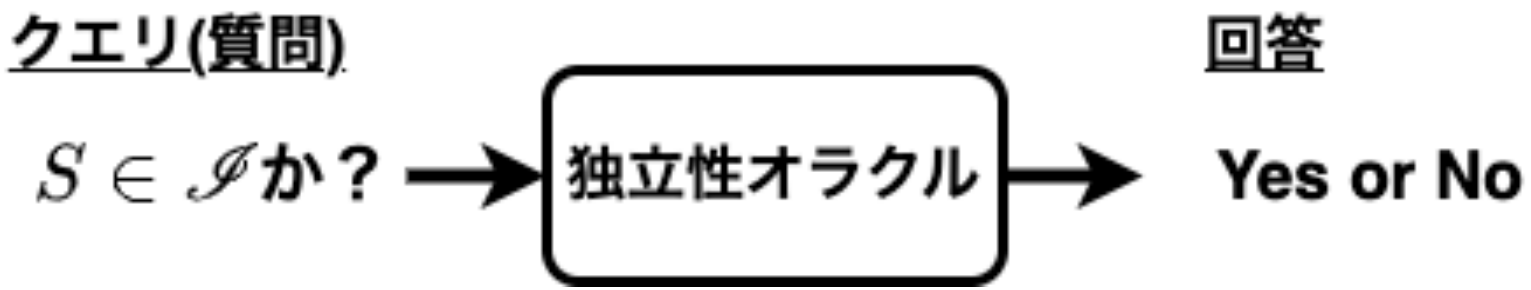
マトロイド $\mathcal{M} = (V, \mathcal{I})$

$\mathcal{I}$の要素を独立集合と呼ぶ

有限集合 V 上の部分集合族 $\mathcal{I}$ で良い性質を持つもの

Q. マトロイドをどう扱うのか？

独立性オラクル

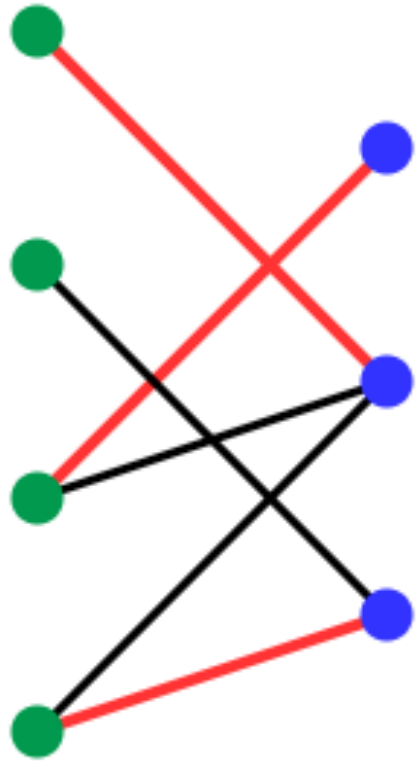


マトロイド交叉問題

入力：2つのマトロイド $\mathcal{M}_1 = (V, \mathcal{I}_1), \mathcal{M}_2 = (V, \mathcal{I}_2)$
出力：最大サイズの**共通独立集合** $S \in \mathcal{I}_1 \cap \mathcal{I}_2$

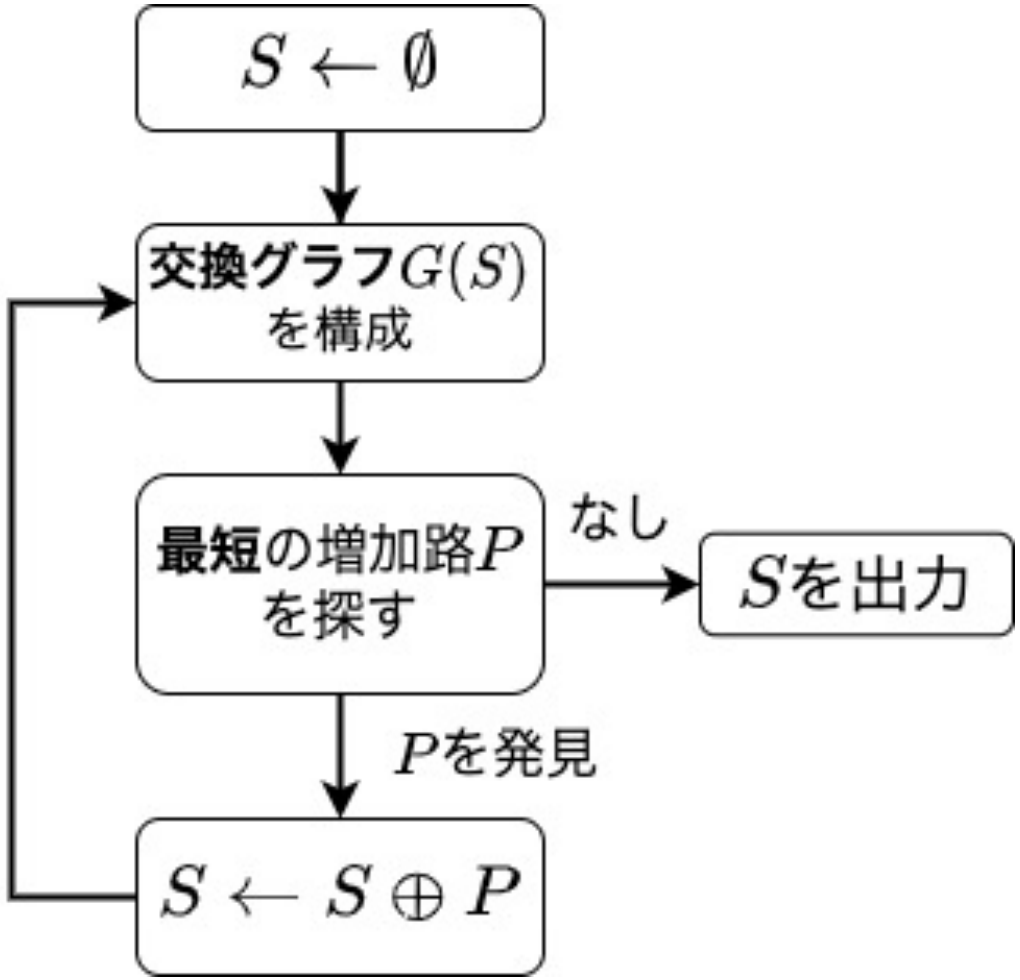
例）二部グラフの最大マッチング

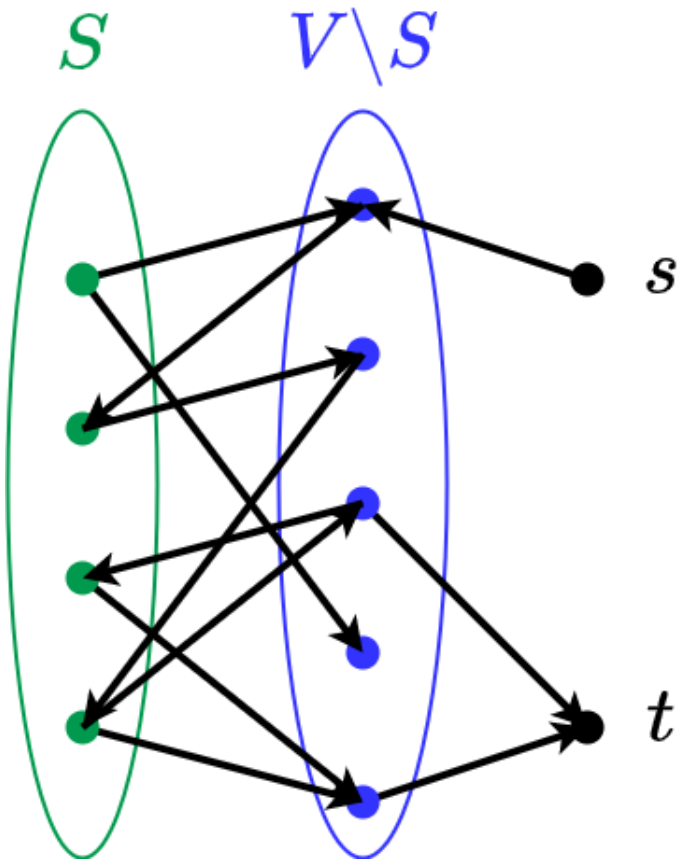
- $\mathcal{I}_1$ ＝ **左側の各頂点から辺を高々1本**
- $\mathcal{I}_2$ ＝ **右側の各頂点から辺を高々1本**



マトロイド交叉問題に対するアルゴリズム

(Edmonds 1970, Aigner-Dowling 1971, Lawler 1975)

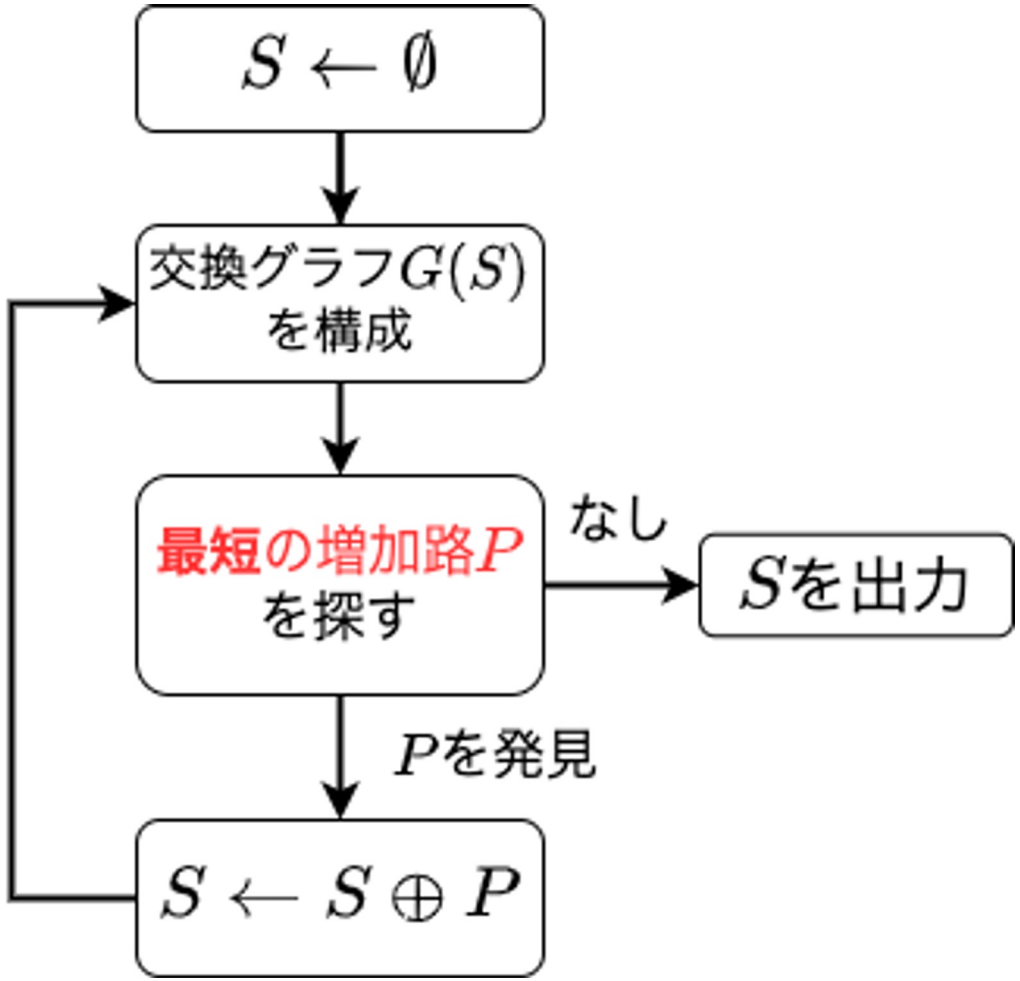


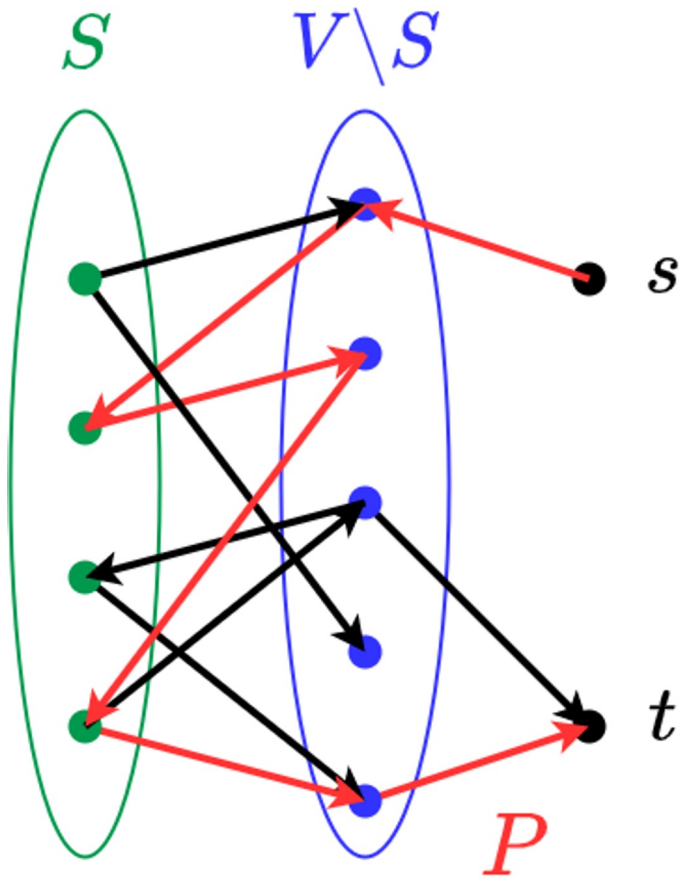


交換グラフ $G(S)$

マトロイド交叉問題に対するアルゴリズム

（Edmonds 1970, Aigner-Dowling 1971, Lawler 1975）





交換グラフ $G(S)$

マトロイド交叉問題の高速化

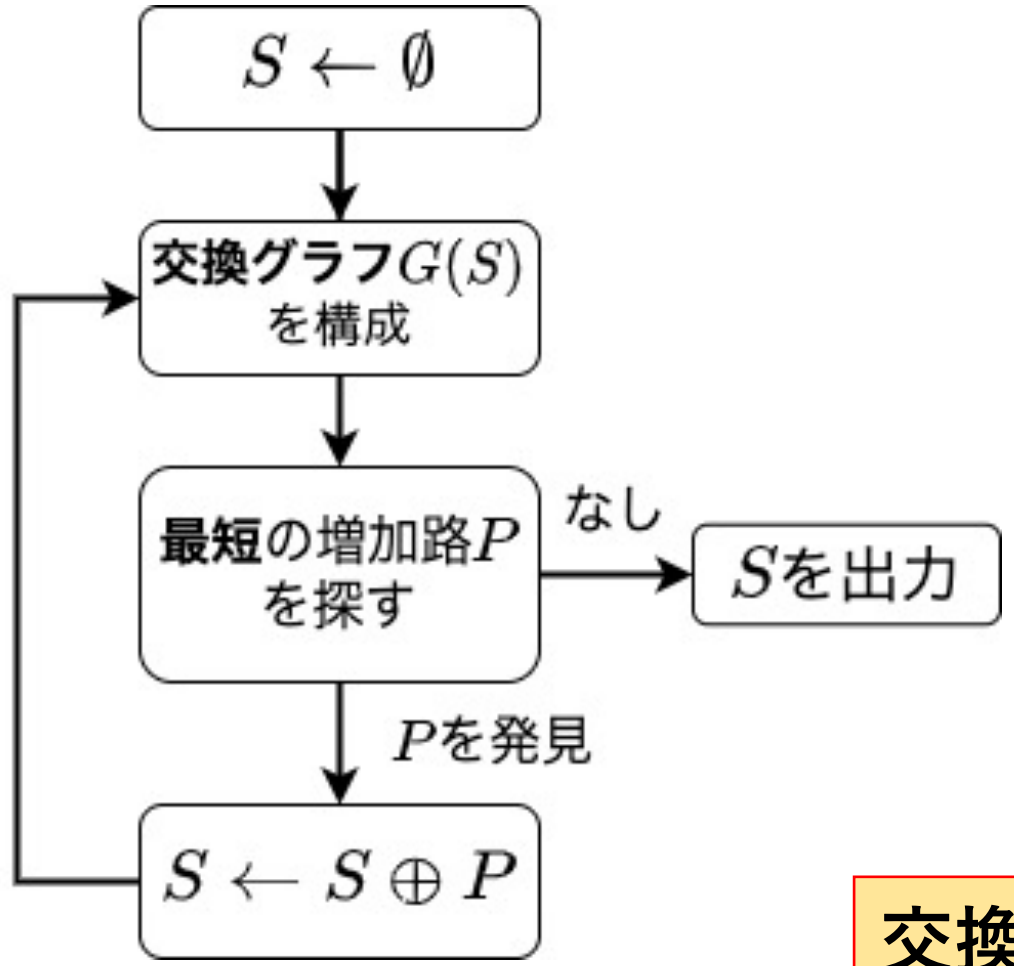
独立性オラクルへのクエリ回数

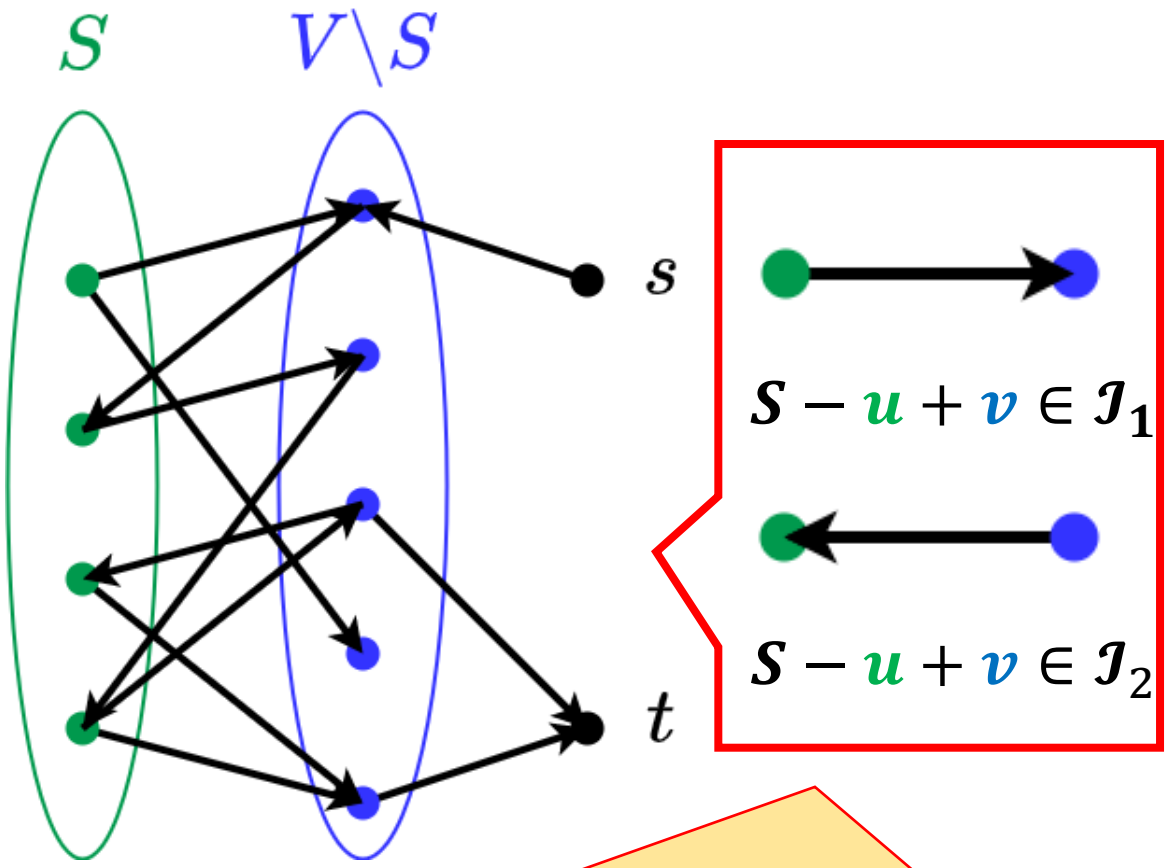
1970s	Edmonds, Lawler, Aigner-Dowling	$O(nr^2)$
1986	Cunningham	$O(nr^{3/2})$
2015	Lee-Sidford-Wong	$\tilde{O}(n^2)$
2019	Nguyễn, Chakrabarty-Lee-Sidford-Singla-Wong	$\tilde{O}(nr)$
2021	Blikstad-v.d.Brand-Mukhopadhyay-Nanongkai	$\tilde{O}(n^{9/5})$
2021	Blikstad	$\tilde{O}(nr^{3/4})$

$n = |V|$, 解のサイズ $\boldsymbol{r}(\leq \boldsymbol{n})$

マトロイド交叉問題に対するアルゴリズム

(Edmonds 1970, Aigner-Dowling 1971, Lawler 1975)

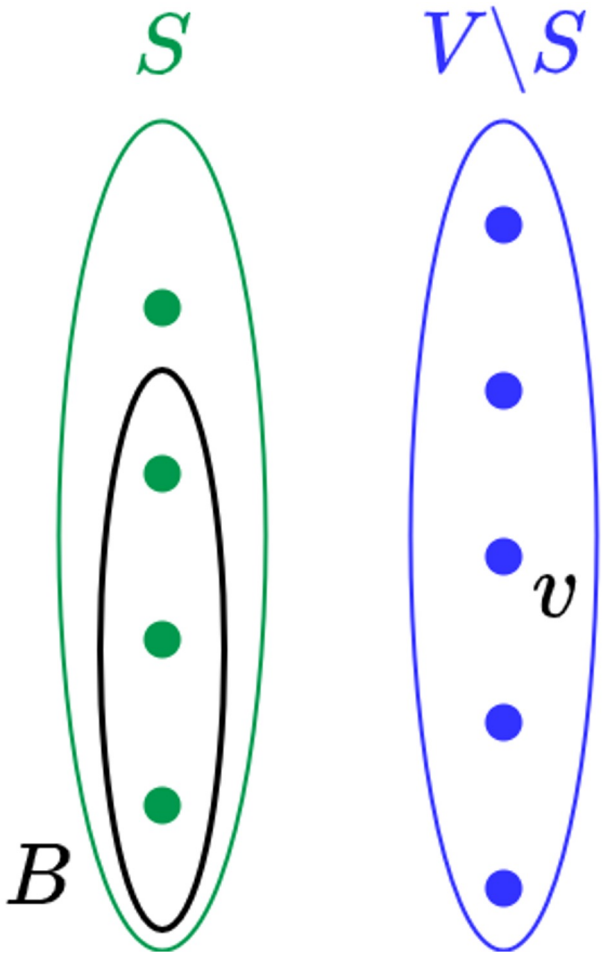




交換グラフ $G(S)$ の辺の候補を全て見る！

マトロイド交叉の高速化の道具

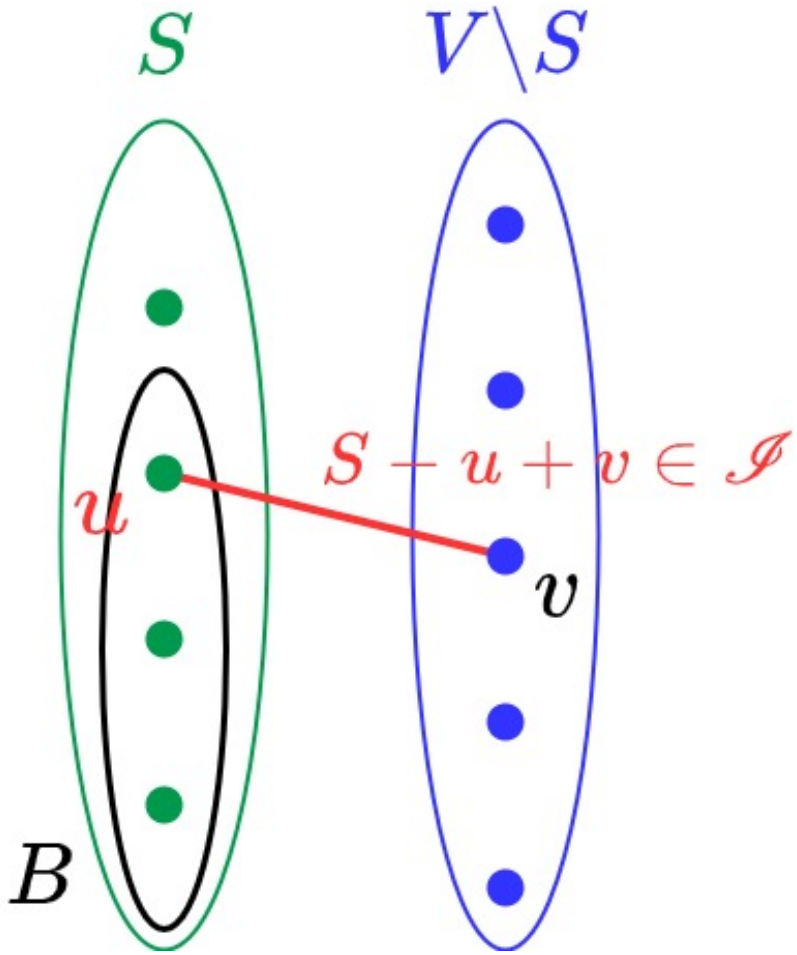
(Nguyễn 2019, Chakrabarty et al. 2019)



入力：$\mathcal{M} = (V, \mathcal{I})$, $S \in \mathcal{I}$, $v \in V \setminus S$, $B \subseteq S$

マトロイド交叉の高速化の道具

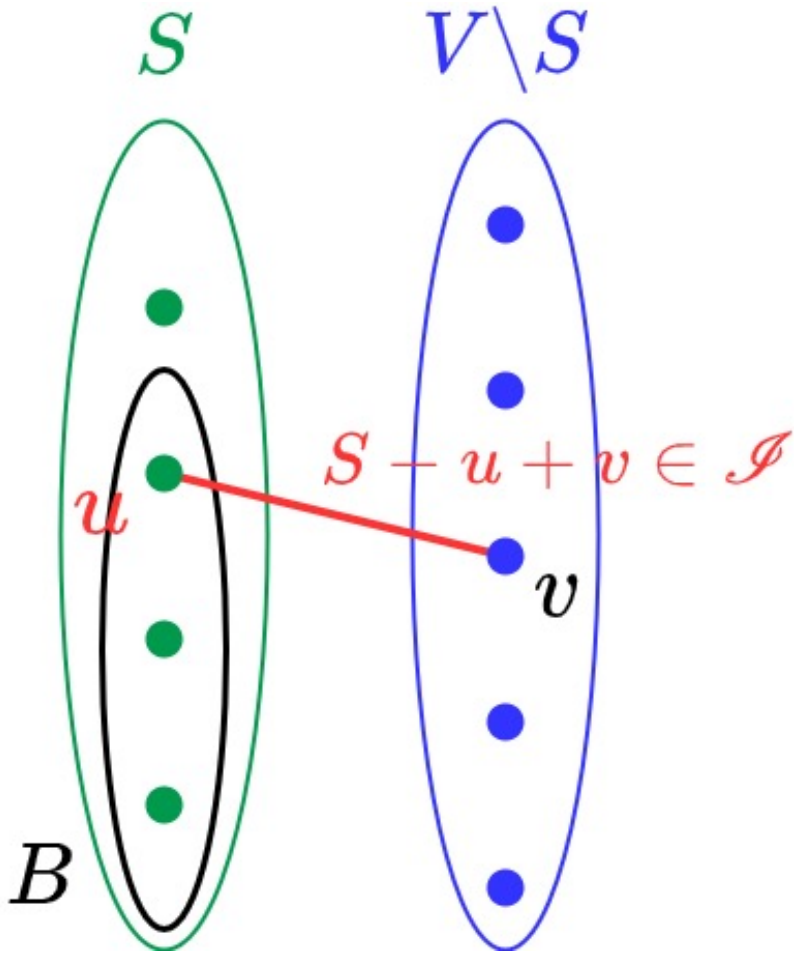
(Nguyễn 2019, Chakrabarty et al. 2019)



入力：$\mathcal{M} = (V, \mathcal{I})$, $S \in \mathcal{I}$, $v \in V \setminus S$, $B \subseteq S$
出力：$S - u + v \in \mathcal{I}$ なる $u \in B$ を一つ

マトロイド交叉の高速化の道具

(Nguyễn 2019, Chakrabarty et al. 2019)

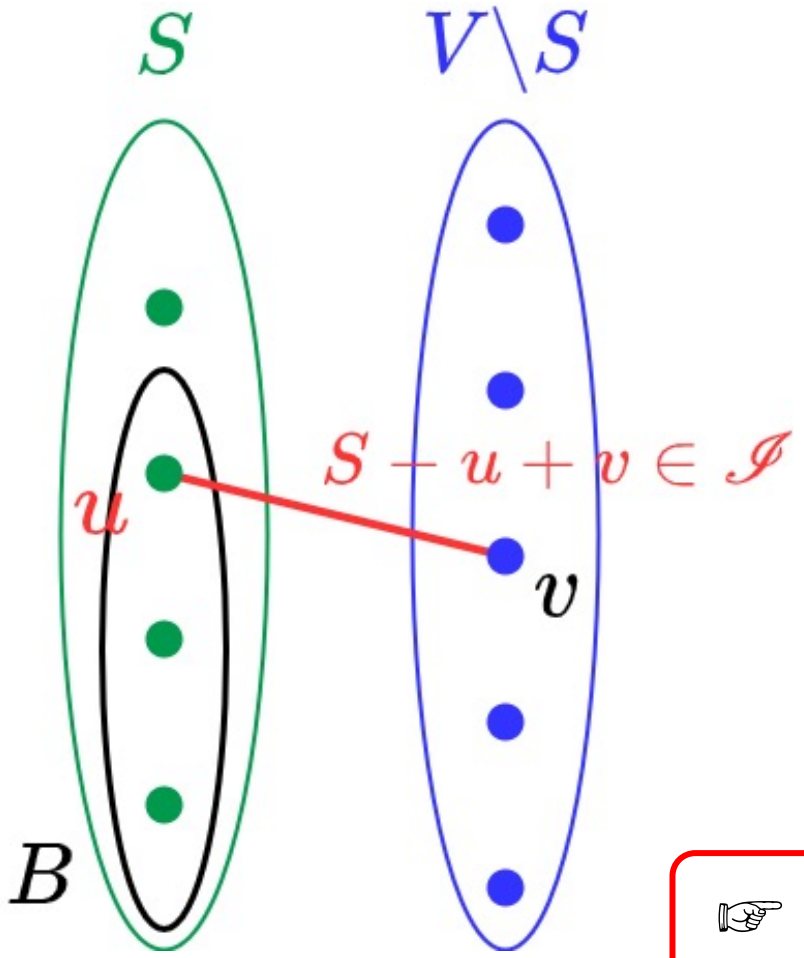


入力：$\mathcal{M} = (V, \mathcal{I})$, $S \in \mathcal{I}$, $v \in V \setminus S$, $B \subseteq S$
出力：$S - u + v \in \mathcal{I}$ なる $u \in B$ を一つ

二分探索を用いることで、
$O(\log |B|)$ **回**の独立性オラクル
へのクエリでできる

マトロイド交叉の高速化の道具

(Nguyễn 2019, Chakrabarty et al. 2019)



入力：$\mathcal{M} = (V, \mathcal{I})$, $S \in \mathcal{I}$, $v \in V \setminus S$, $B \subseteq S$
出力：$S - u + v \in \mathcal{I}$ なる $u \in B$ を一つ

二分探索を用いることで、
$O(\log |B|)$ 回の独立性オラクル
へのクエリでできる

☞ **交換グラフ$G(S)$ の辺の候補を全て見る必要はない！**

マトロイド分割問題

入力：**k 個のマトロイド** $\mathcal{M}_1 = (V, \mathcal{I}_1), \ldots, \mathcal{M}_k = (V, \mathcal{I}_k)$
出力：**分割可能**な最大サイズの集合 $S \subseteq V$

$S_i \in \mathcal{I}_i$なるSの分割$S = S_1 \cup \cdots \cup S_k$が存在

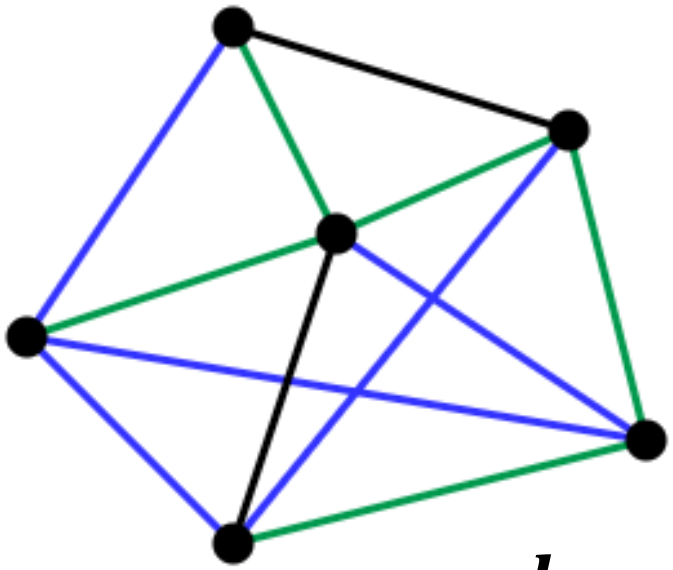
マトロイド分割問題

入力：**k 個のマトロイド** $\mathcal{M}_1 = (V, \mathcal{I}_1), \ldots, \mathcal{M}_k = (V, \mathcal{I}_k)$
出力：**分割可能**な最大サイズの集合 $S \subseteq V$

$S_i \in \mathcal{I}_i$ なる S の分割 $S = S_1 \cup \cdots \cup S_k$ が存在

例） k-全域森問題

互いに辺素な森 k 個に**分割できる**
最大サイズの辺集合



$\boldsymbol{k = 2}$

マトロイド分割とマトロイド交叉

マトロイド分割問題は**マトロイド交叉問題に帰着**して解ける

☞ $\boldsymbol{V} \times \{\mathbf{1}, \ldots, \boldsymbol{k}\}$ 上の**直和マトロイド**と**分割マトロイド**の交叉で解ける

マトロイド分割とマトロイド交叉

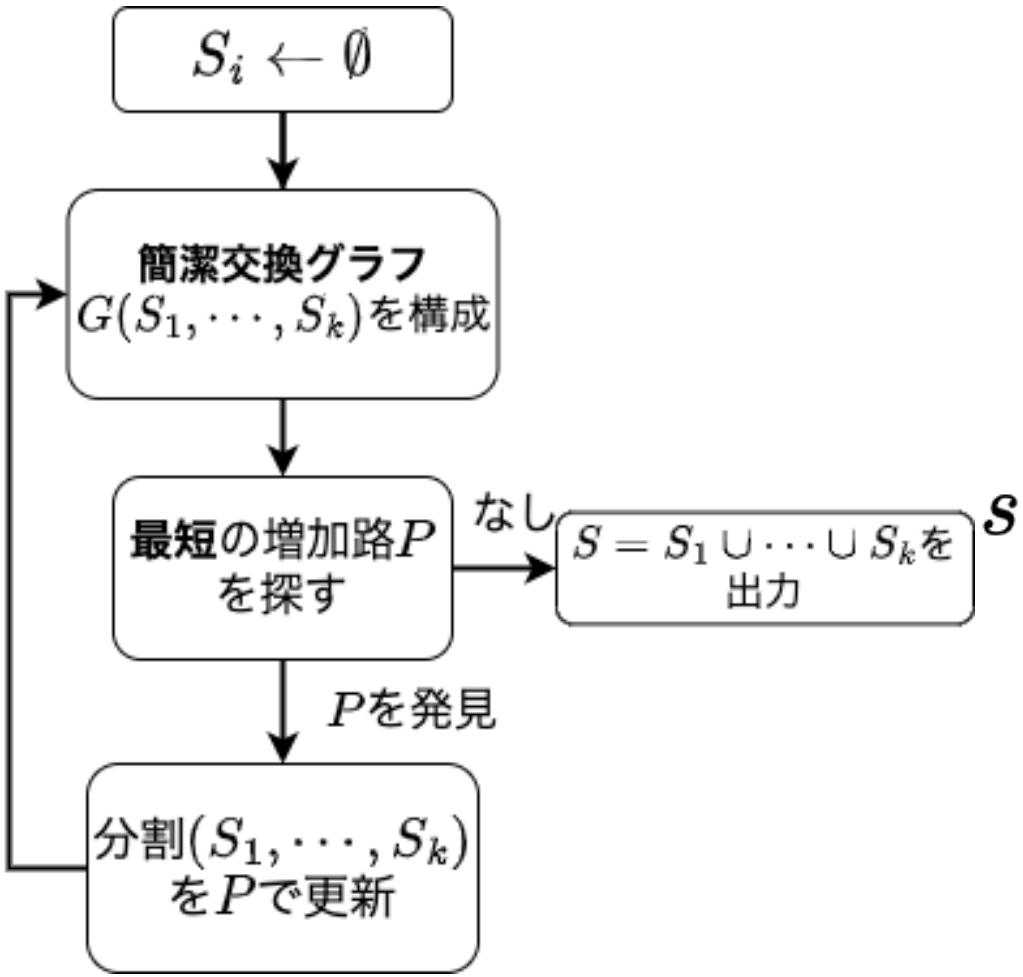
マトロイド分割問題は**マトロイド交叉問題に帰着**して解ける

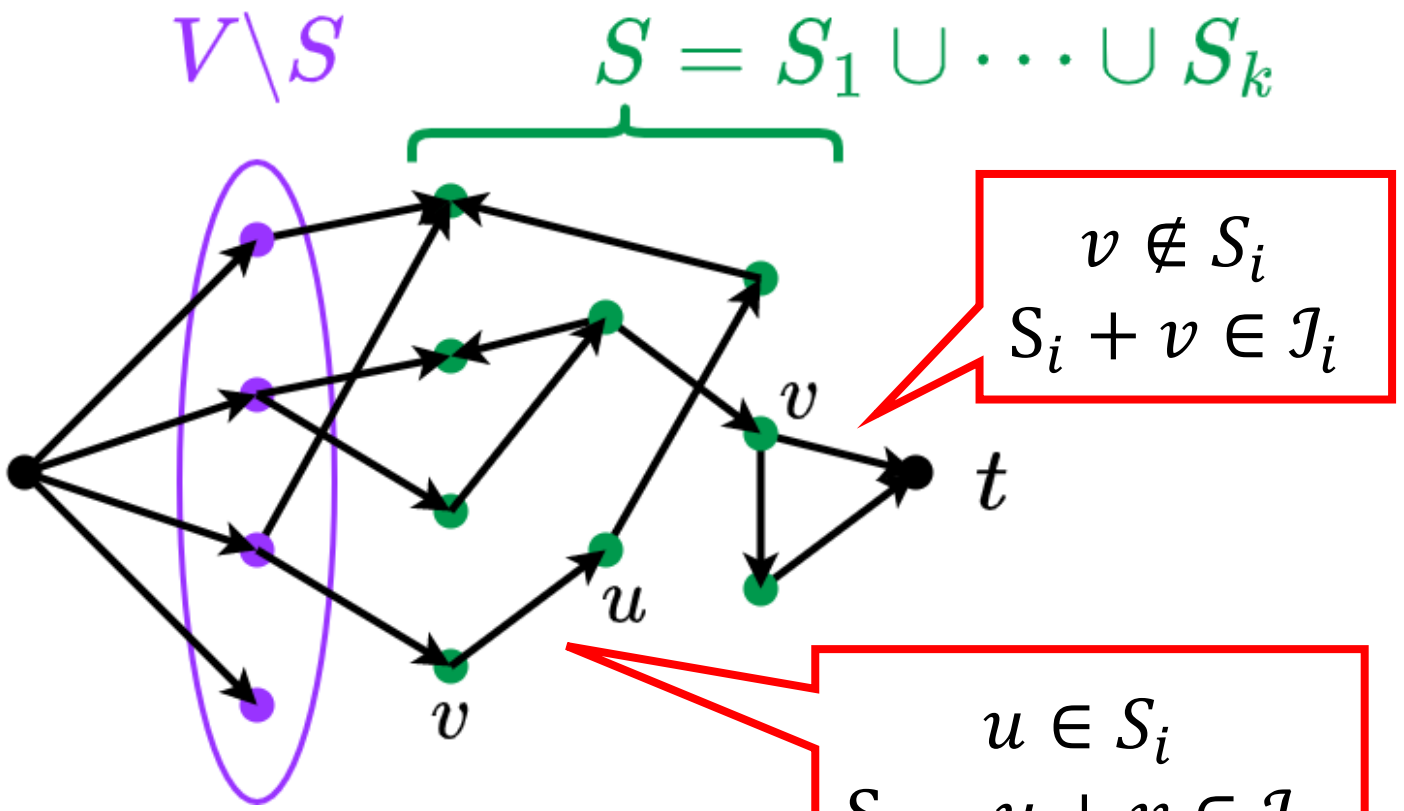
☞ $V \times \{1, \ldots, k\}$ 上の**直和マトロイド**と**分割マトロイド**の交叉で解ける

台集合のサイズがknと大 ⇒ 計算量が大きくなってしまう

マトロイド分割問題に対するアルゴリズム

(Edmonds 1968)

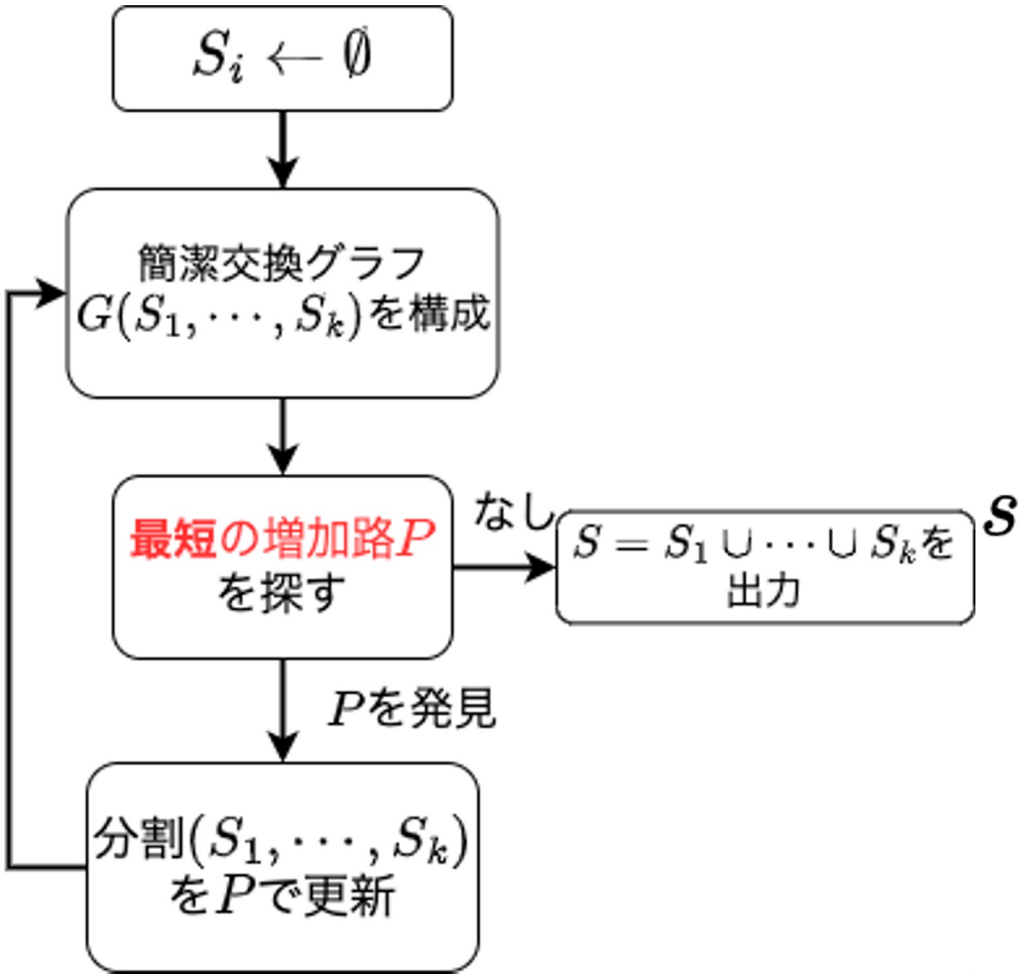


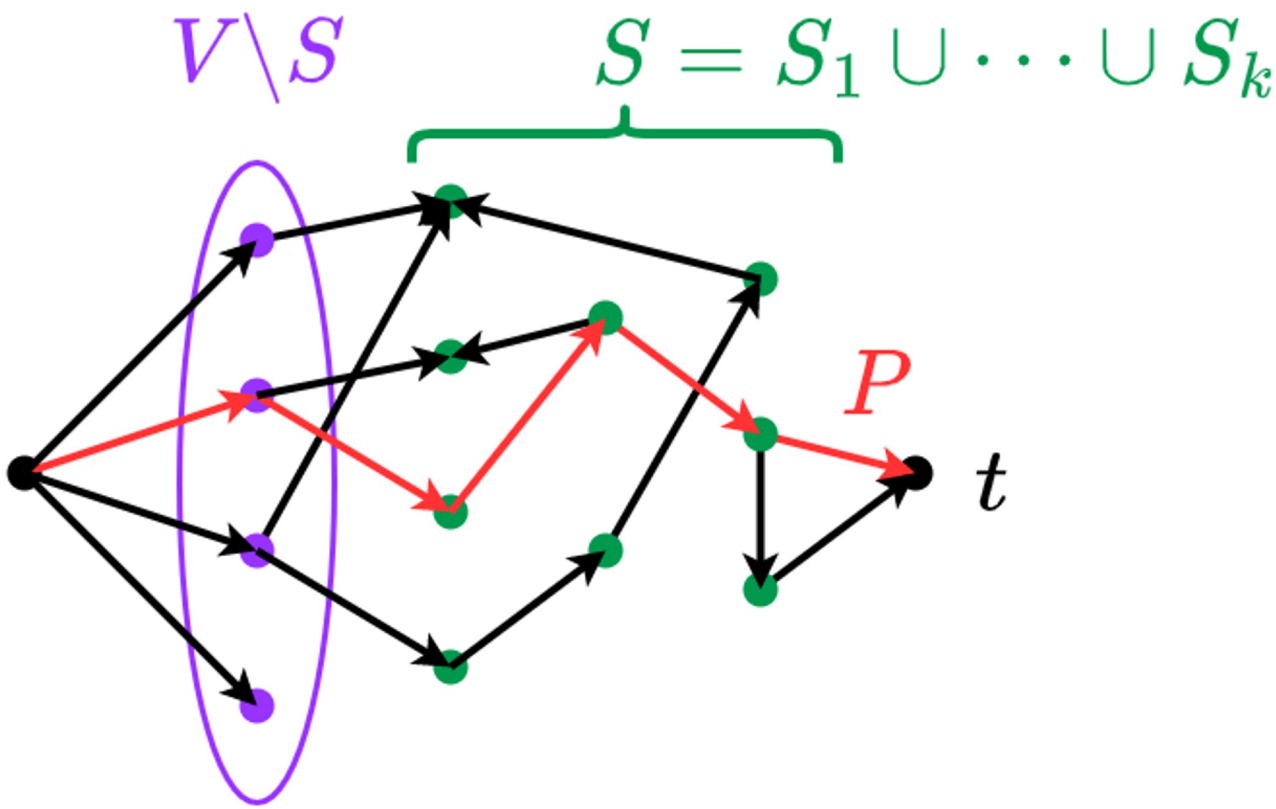


簡潔交換グラフ $G(S_1, \ldots, S_k)$

マトロイド分割問題に対するアルゴリズム

(Edmonds 1968)





簡潔交換グラフ $G(S_1, \ldots, S_k)$

マトロイド分割問題の高速化

独立性オラクルへのクエリ回数

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

$n = |V|$, マトロイドの個数 $\boldsymbol{k}$
解のサイズ $\boldsymbol{p}(\leq \boldsymbol{n})$

マトロイド分割問題の高速化

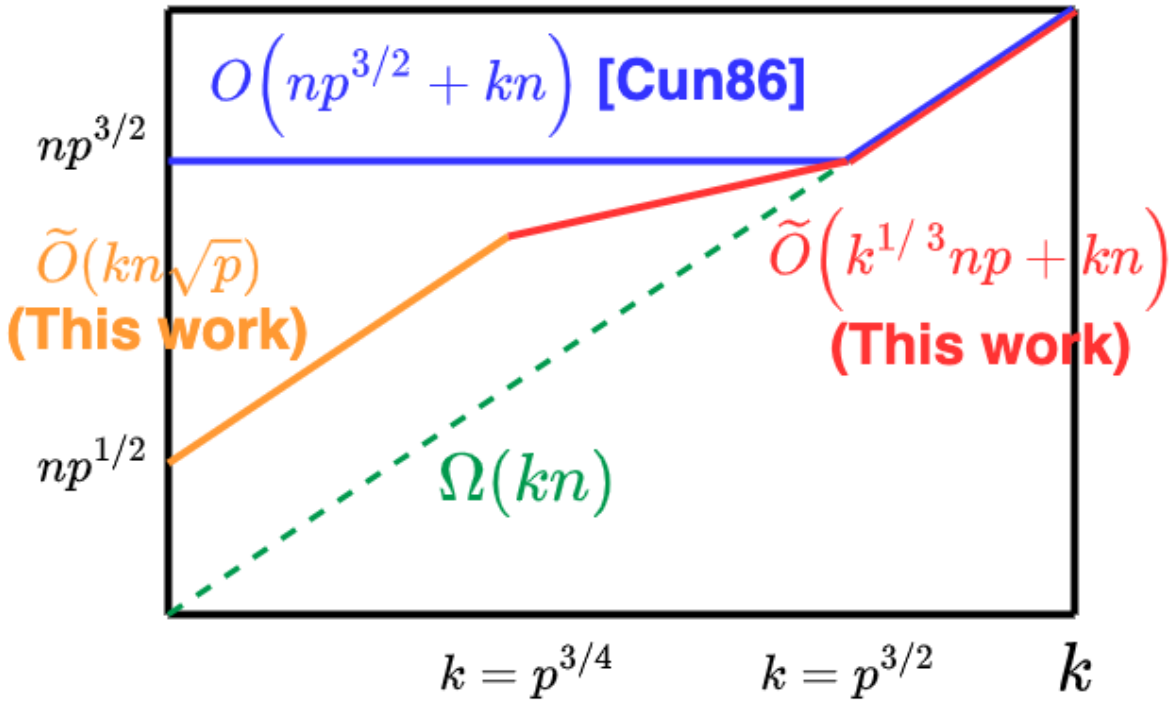
		独立性オラクルへのクエリ回数
1968	Edmonds	$O(np^2 + kn)$
1986	Cunningham	$O(np^{3/2} + kn)$
2023	**本研究**	$\boldsymbol{\tilde{O}(kn\sqrt{p})}$
2023	**本研究**	$\boldsymbol{\tilde{O}(k^{1/3}np + kn)}$

$n = |V|$, マトロイドの個数 $\boldsymbol{k}$
解のサイズ $\boldsymbol{p}(\leq \boldsymbol{n})$

マトロイド分割問題の高速化

独立性オラクルへのクエリ回数

1968	Edmonds	$O(np^2 + kn)$
1986	Cunningham	$O(np^{3/2} + kn)$
2023	本研究	$\tilde{O}(kn\sqrt{p})$
2023	本研究	$\tilde{O}(k^{1/3}np + kn)$



$n = |V|$, マトロイドの個数 k
解のサイズ $p(\leq n)$

提案アルゴリズム①：ブロッキングフロー

定理1
マトロイド分割問題は$\tilde{O}(kn\sqrt{p})$回の**独立性オラクル**へのクエリで解ける

$n = |V|$, マトロイドの個数 k
解のサイズ $p(\leq n)$

提案アルゴリズム①：ブロッキングフロー

定理1
マトロイド分割問題は$\tilde{O}(kn\sqrt{p})$回の**独立性オラクル**へのクエリで解ける

アイデア

ブロッキングフロー (Cunningham 1986)
☞ Hopcroft-KarpやDinicと類似

＋

二分探索で辺をみつける
(Nguyễn 2019, Chakrabarty et al. 2019)

同じ長さの増加路をまとめてみつける

提案アルゴリズム①：ブロッキングフロー

定理1
マトロイド分割問題は$\tilde{O}(kn\sqrt{p})$回の**独立性オラクル**へのクエリで解ける

アルゴリズム

Repeat:

Step 1: 幅優先探索

Step 2: 同じ長さの増加路を全てみつける

提案アルゴリズム①：ブロッキングフロー

定理1
マトロイド分割問題は$\tilde{O}(kn\sqrt{p})$回の**独立性オラクル**へのクエリで解ける

アルゴリズム

Repeat:

Step 1: 幅優先探索 ⇐ $\tilde{O}(kn)$回クエリ

Step 2: 同じ長さの増加路を全てみつける ⇐ $\tilde{O}(kn)$回クエリ

提案アルゴリズム①：ブロッキングフロー

定理1
マトロイド分割問題は$\tilde{O}(kn\sqrt{p})$回の**独立性オラクル**へのクエリで解ける

アルゴリズム

Repeat:

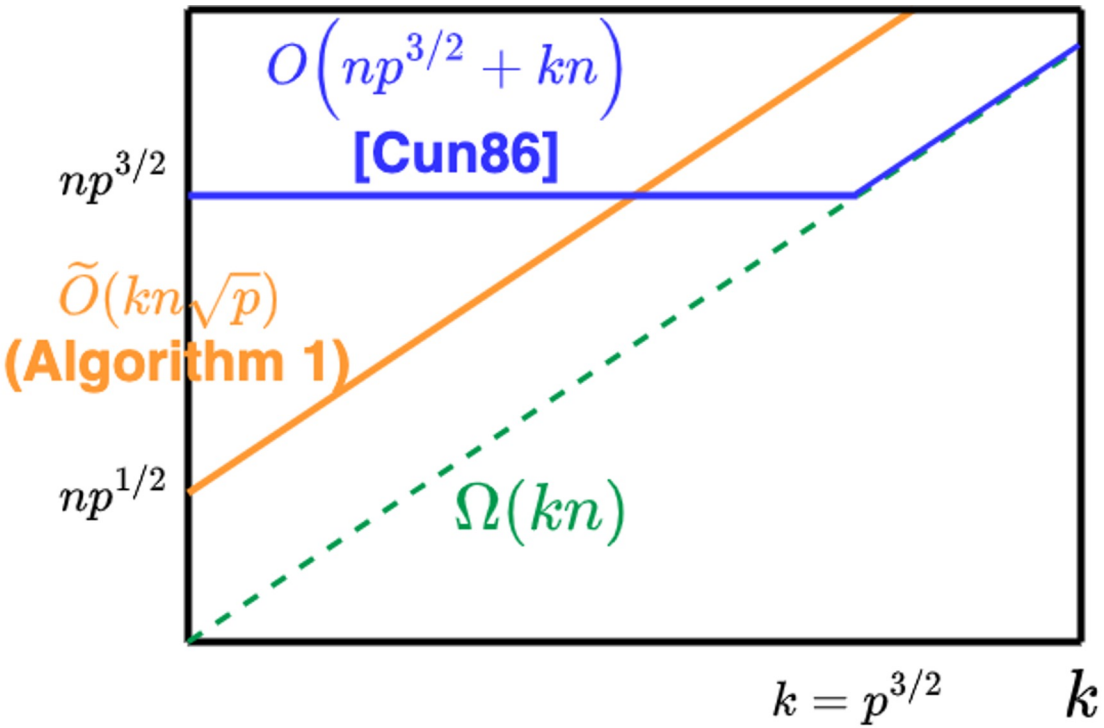
Step 1: 幅優先探索 ⇐ $\tilde{O}(kn)$回クエリ

Step 2: 同じ長さの増加路を全てみつける ⇐ $\tilde{O}(kn)$回クエリ

事実: $\Theta(\sqrt{p})$回繰り返せば十分

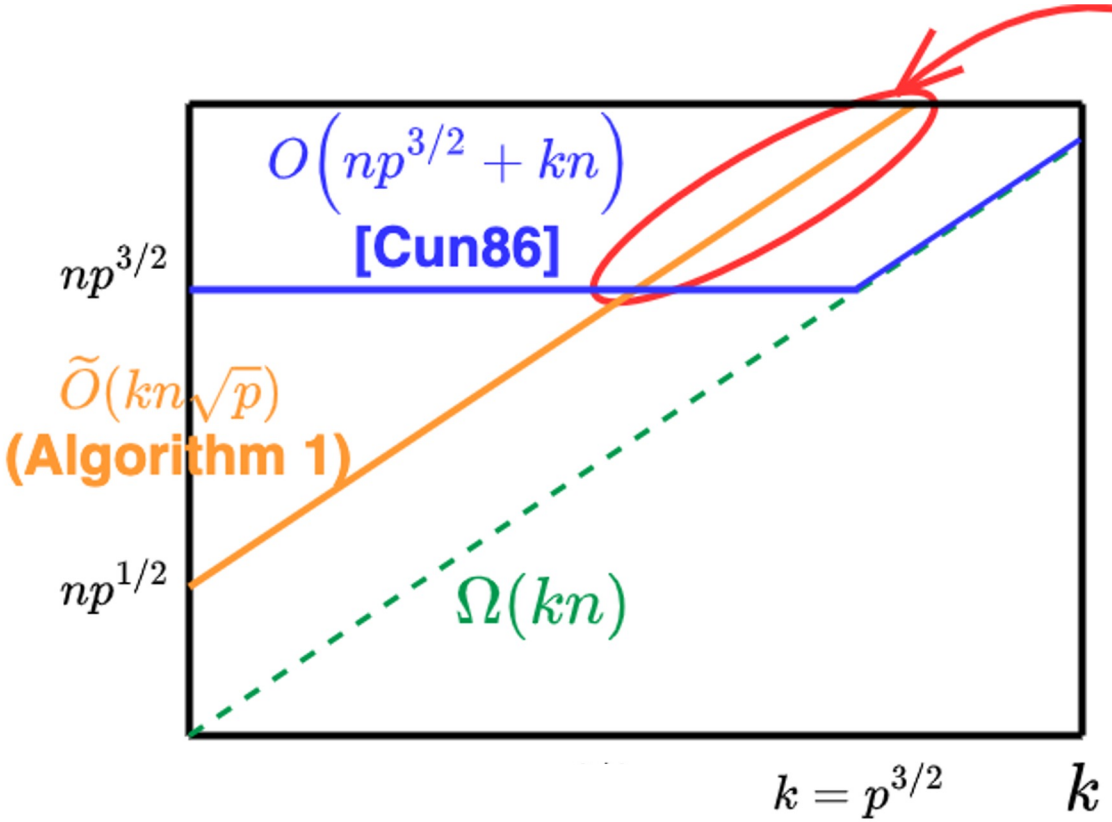
提案アルゴリズム①：ブロッキングフロー

定理1
マトロイド分割問題は$\tilde{O}(kn\sqrt{p})$回の**独立性オラクル**へのクエリで解ける



提案アルゴリズム①：ブロッキングフロー

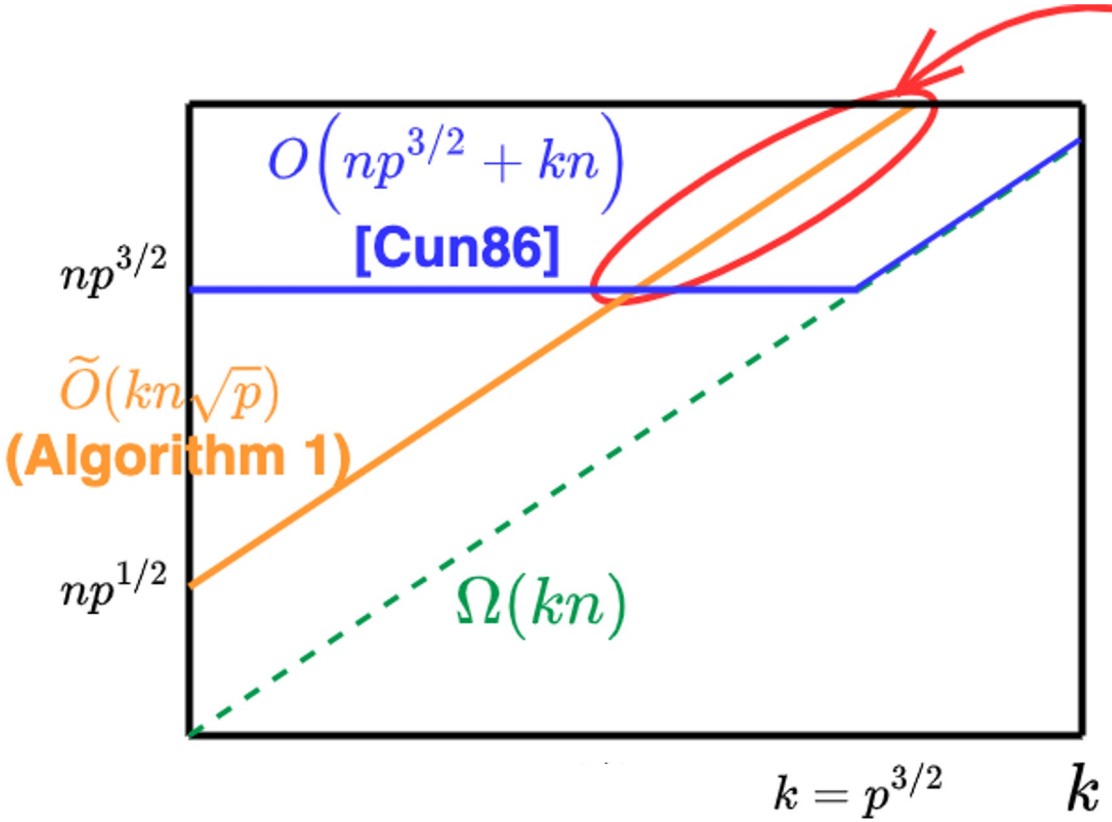
定理1
マトロイド分割問題は$\tilde{O}(kn\sqrt{p})$回の**独立性オラクル**へのクエリで解ける



二分探索を使ってるにも関わらず、
[Cun 86]より悪い計算量

提案アルゴリズム①：ブロッキングフロー

定理1
マトロイド分割問題は$\tilde{O}(kn\sqrt{p})$回の**独立性オラクル**へのクエリで解ける

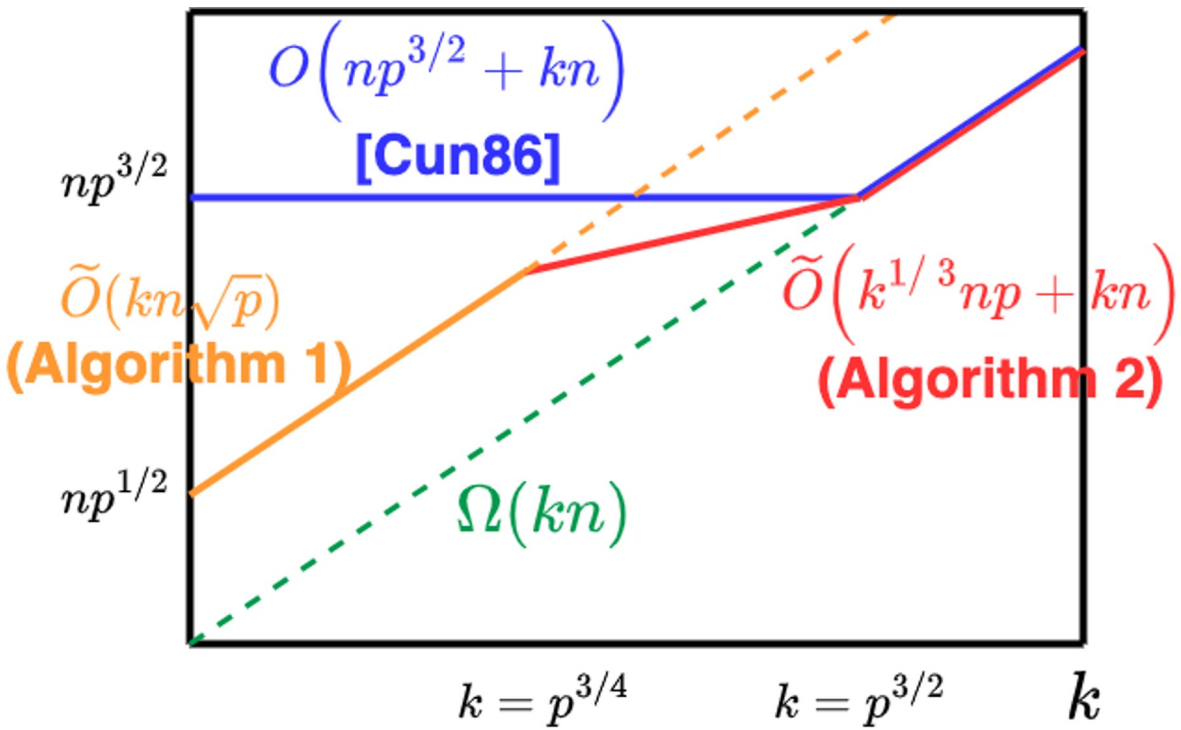


二分探索を使ってるにも関わらず、
[Cun 86]より悪い計算量

Q. k **が大きい場合**に、
[Cun 86]より良い計算量にできるか？

提案アルゴリズム②

定理2
マトロイド分割問題は$\tilde{O}(k^{1/3}np + kn)$回の**独立性オラクル**へのクエリで解ける

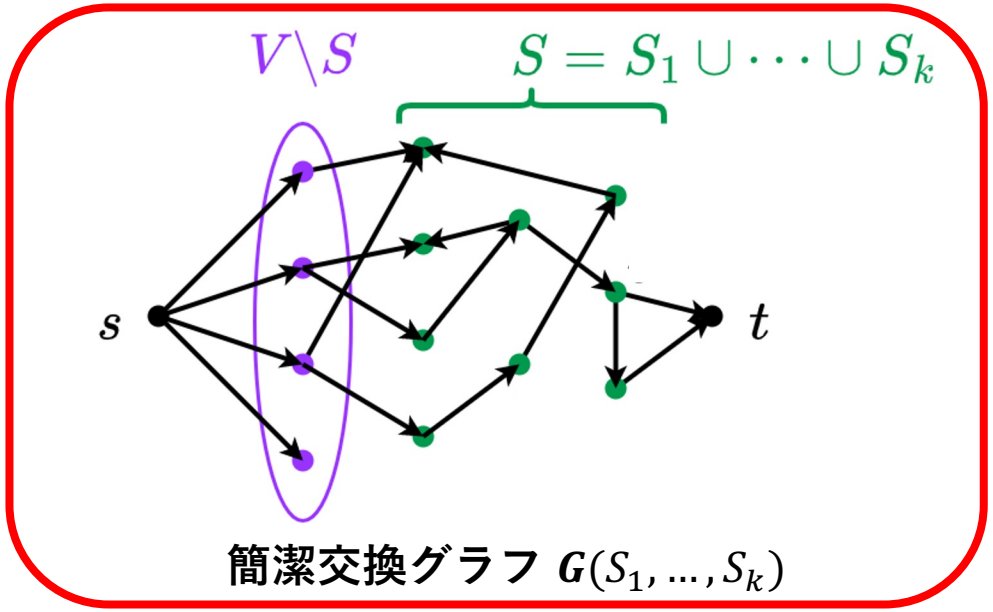


$n = |V|$, マトロイドの個数 k
解のサイズ $p(\leq n)$

1回の辺再利用型増加路探索

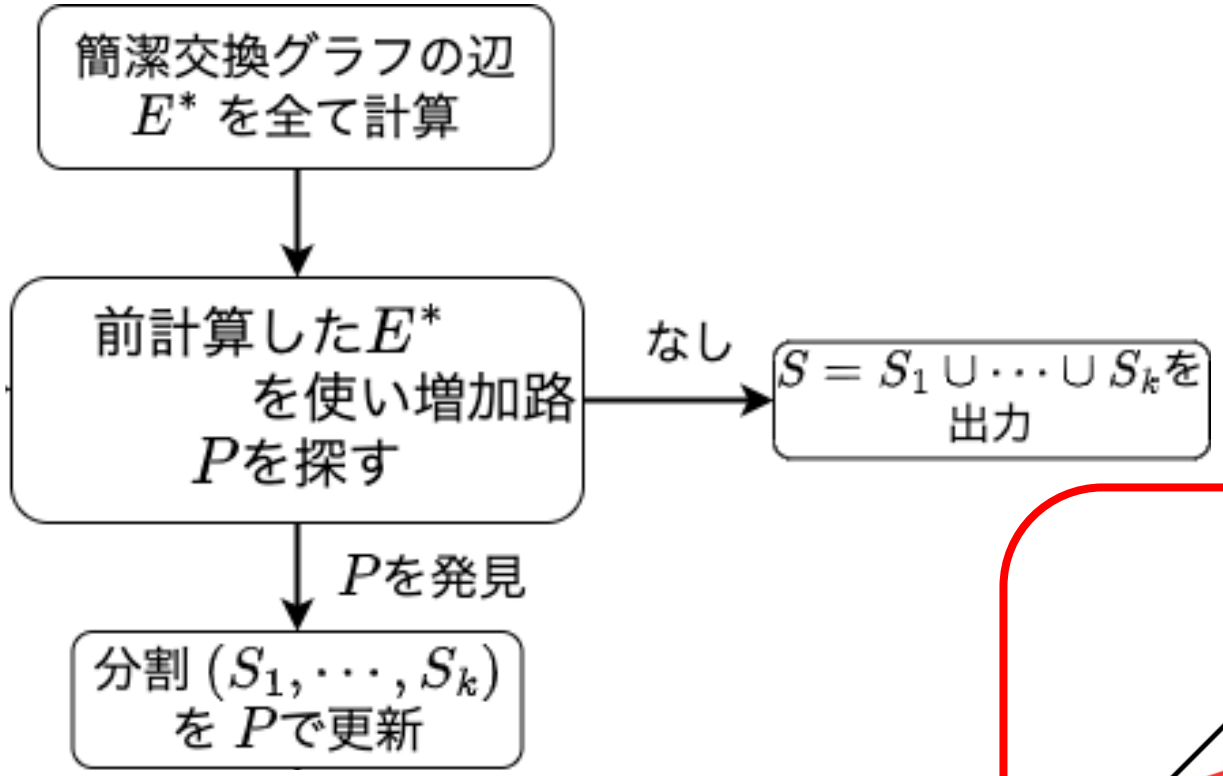
簡潔交換グラフの辺
E^* を全て計算

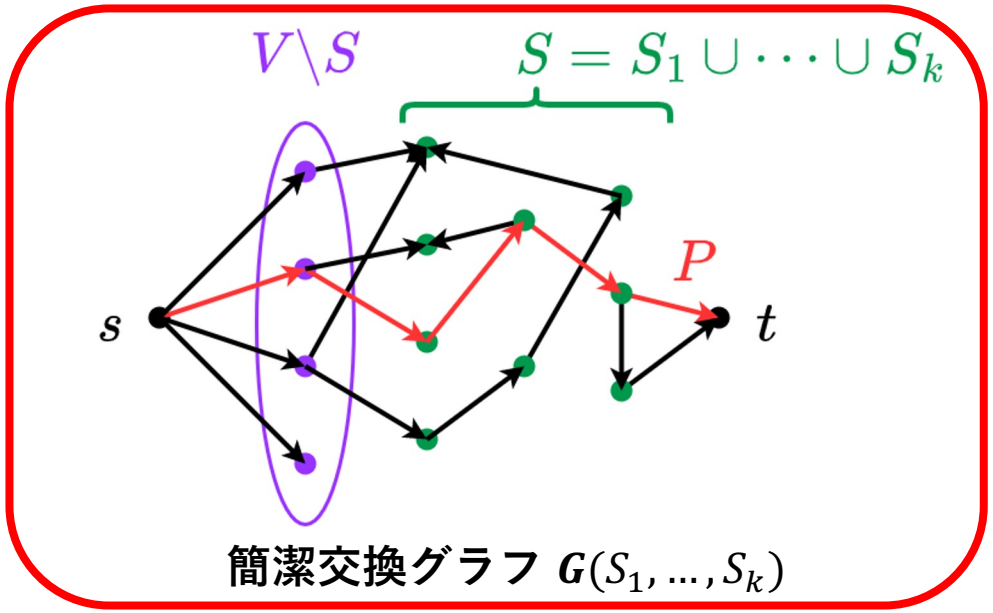
$O(np)$回クエリ



簡潔交換グラフ $G(S_1, \ldots, S_k)$

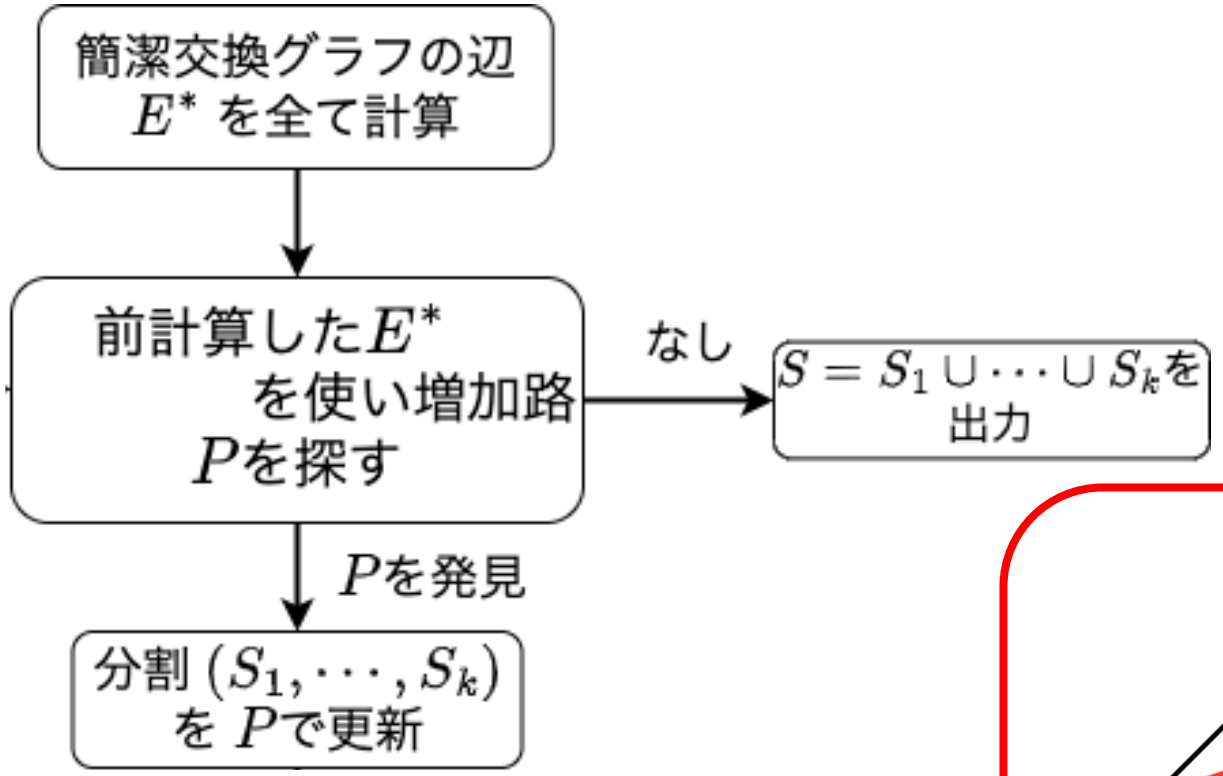
1回の辺再利用型増加路探索



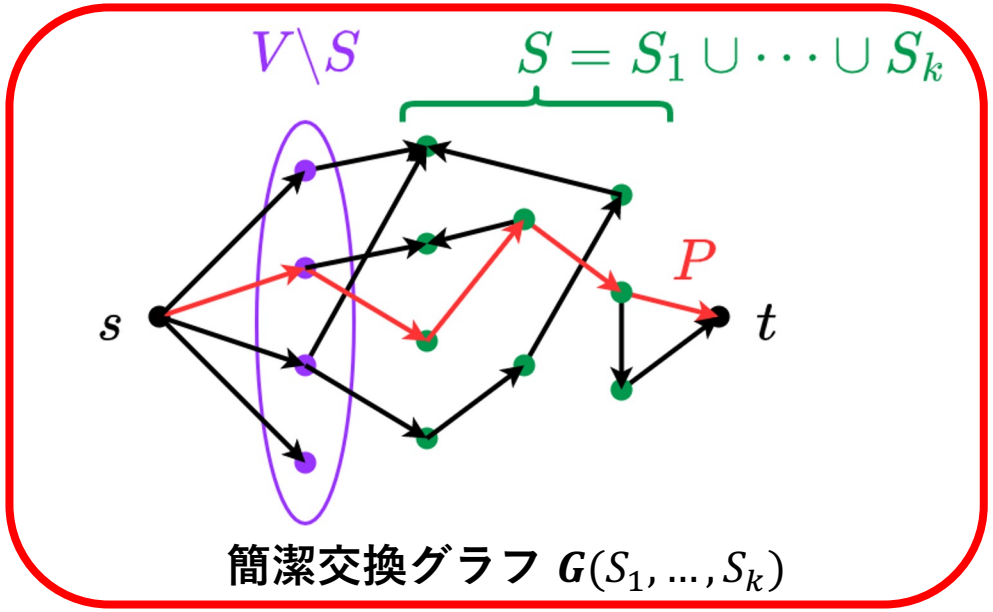


簡潔交換グラフ $\boldsymbol{G(S_1, \ldots, S_k)}$

1回の辺再利用型増加路探索

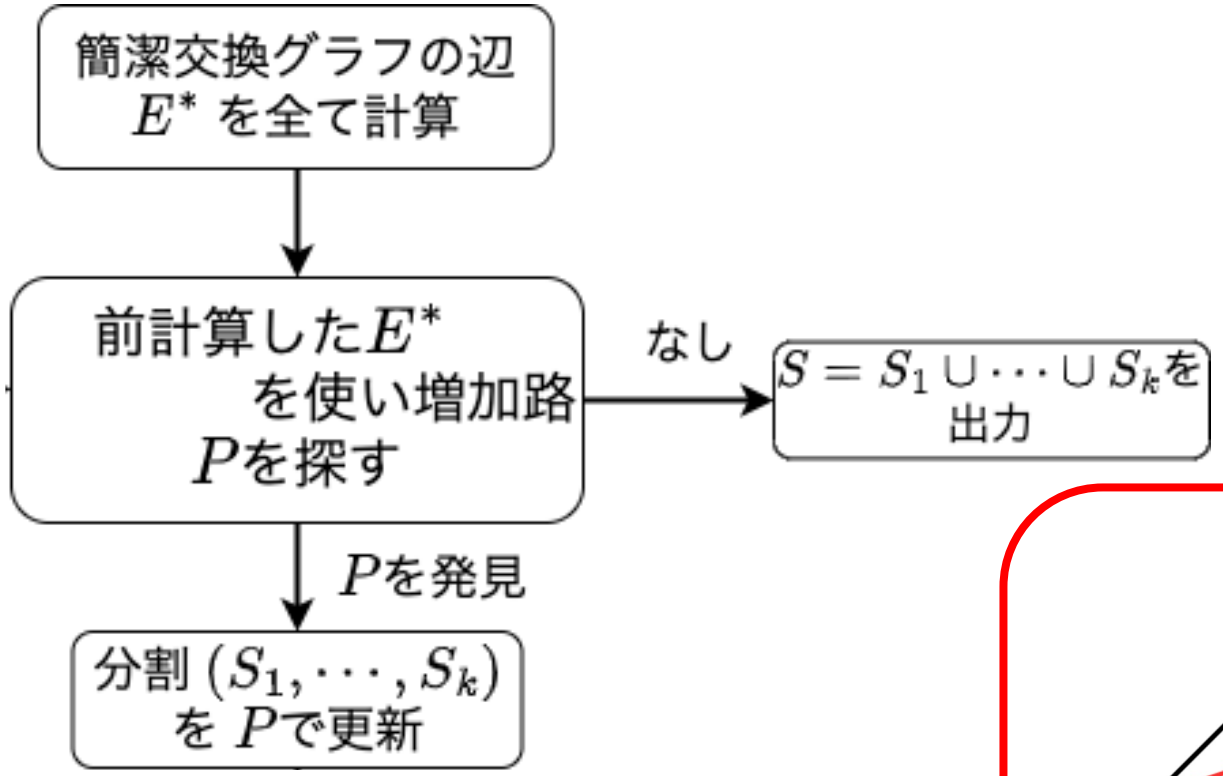


更新するとE^*は　　別のものになる！

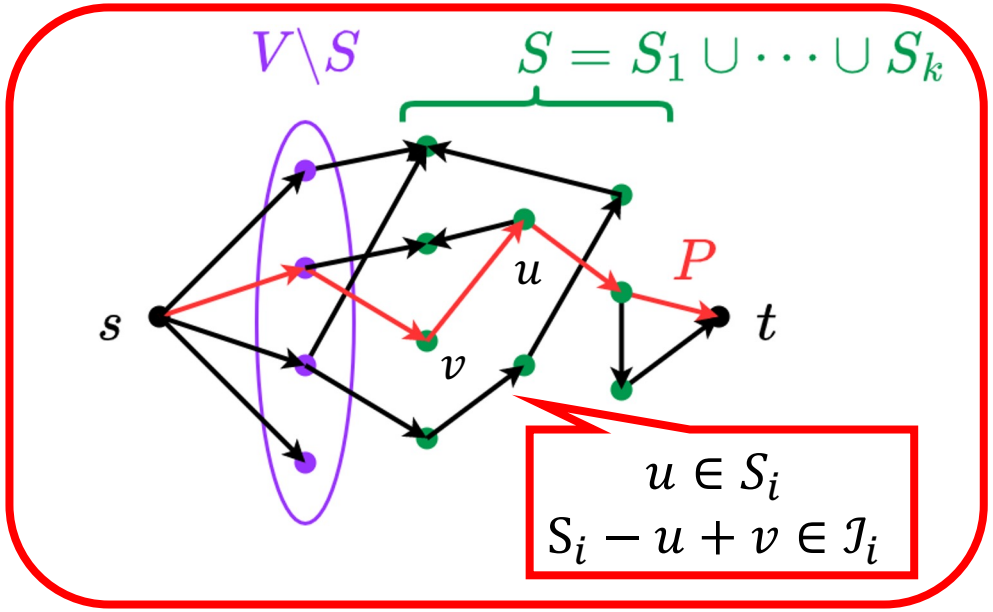


簡潔交換グラフ $G(S_1, \ldots, S_k)$

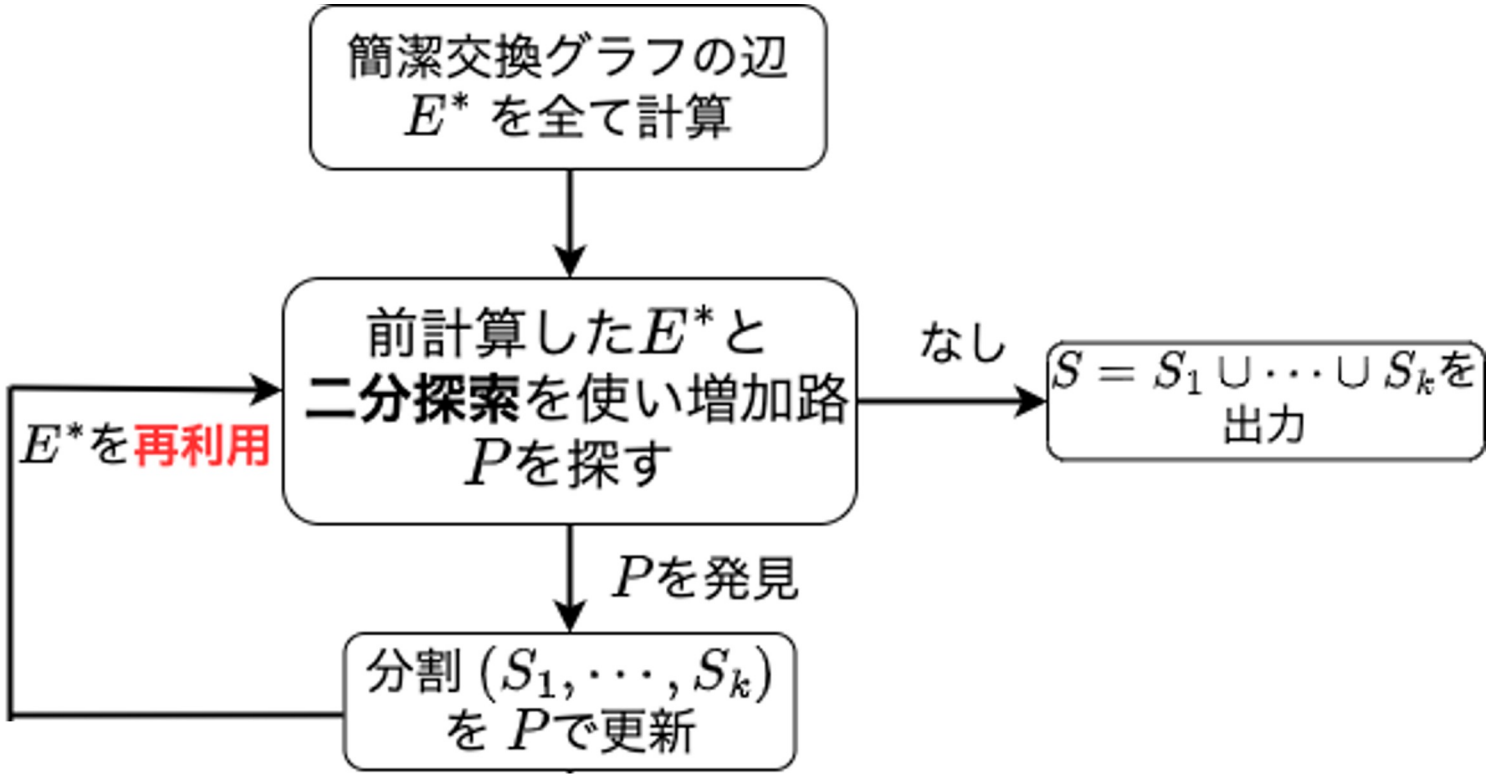
1回の辺再利用型増加路探索



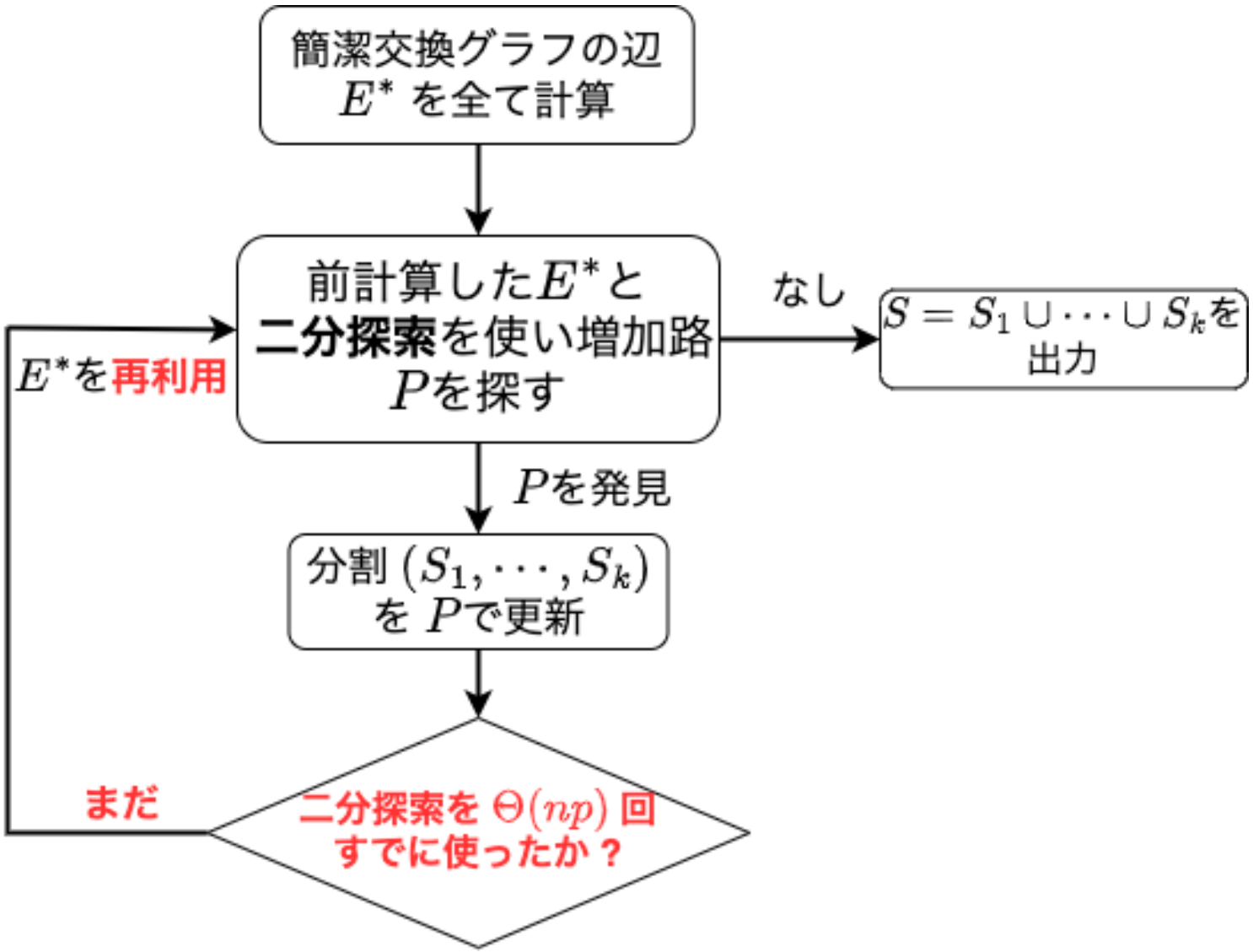
更新するとE^*は**部分的に**別のものになる！



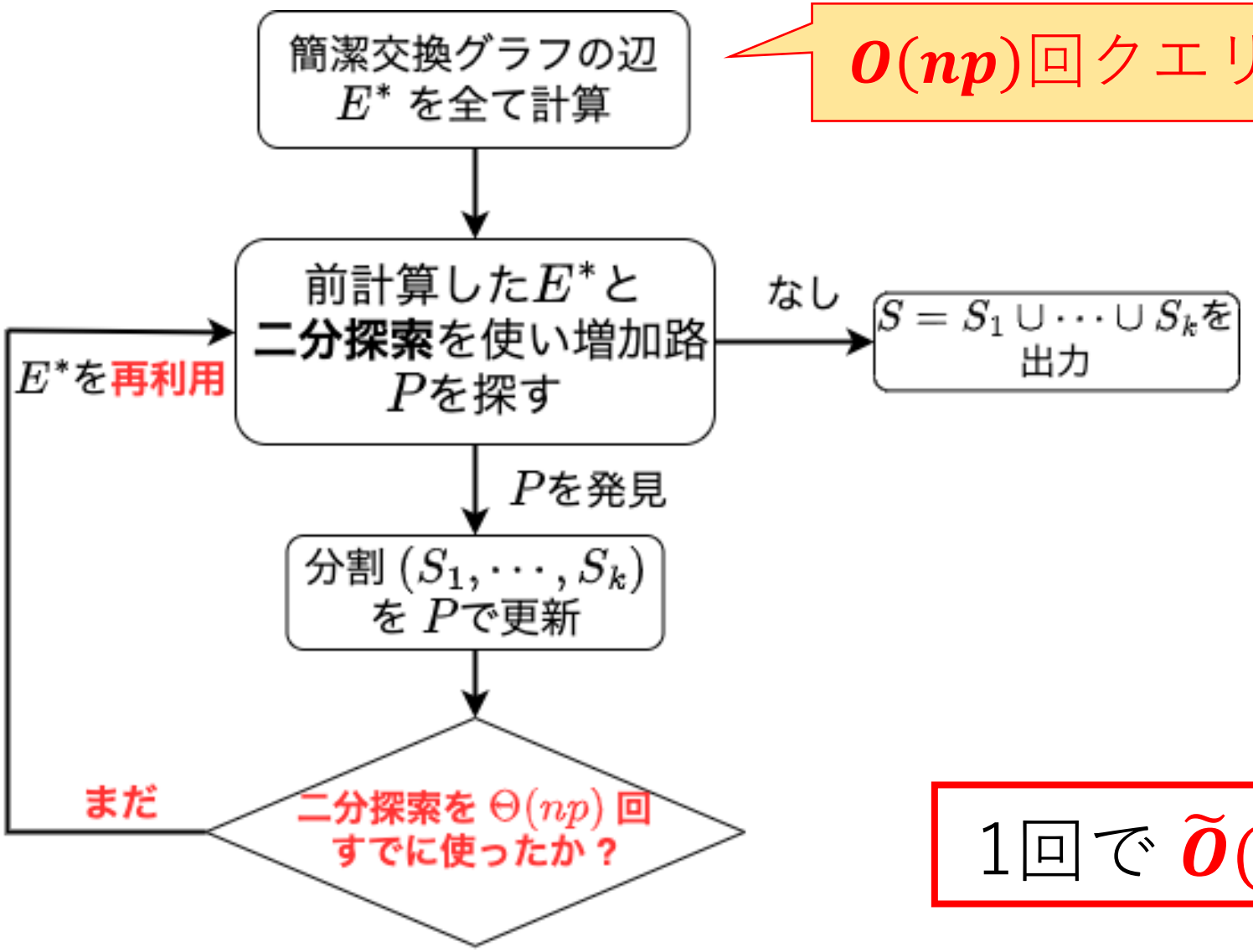
1回の辺再利用型増加路探索



1回の辺再利用型増加路探索



1回の辺再利用型増加路探索



$O(np)$回クエリ

1回で $\tilde{O}(np)$ クエリ使用

提案アルゴリズム②

定理2
マトロイド分割問題は$\tilde{O}(k^{1/3}np + kn)$回の**独立性オラクル**の使用で解ける

アルゴリズム

提案アルゴリズム②

定理2
マトロイド分割問題は$\tilde{O}(k^{1/3}np + kn)$回の**独立性オラクル**の使用で解ける

アルゴリズム

Step 1. **ブロッキングフロー**(提案アルゴリズム①)

提案アルゴリズム②

定理2
マトロイド分割問題は$\tilde{O}(k^{1/3}np + kn)$回の**独立性オラクル**の使用で解ける

アルゴリズム

Step 1. ブロッキングフロー(提案アルゴリズム①)

Step 2. **辺再利用型増加路探索**

提案アルゴリズム②

定理2
マトロイド分割問題は$\tilde{O}(k^{1/3}np + kn)$回の**独立性オラクル**の使用で解ける

アルゴリズム

Step 1. **ブロッキングフロー**(提案アルゴリズム①)を$\Theta(\frac{p}{k^{2/3}})$ **回**

Step 2. 辺再利用型増加路探索

提案アルゴリズム②

定理2
マトロイド分割問題は$\tilde{O}(k^{1/3}np + kn)$回の**独立性オラクル**の使用で解ける

アルゴリズム

Step 1. ブロッキングフロー(提案アルゴリズム①)を$\Theta(\frac{p}{k^{2/3}})$ **回**

Step 2. **辺再利用型増加路探索**

補題: Step2では$\Theta(k^{1/3})$回繰り返せば十分

提案アルゴリズム②

定理2
マトロイド分割問題は$\tilde{O}(k^{1/3}np + kn)$回の**独立性オラクル**の使用で解ける

アルゴリズム

Step 1. ブロッキングフロー(提案アルゴリズム①)を$\Theta(\frac{p}{k^{2/3}})$ **回**

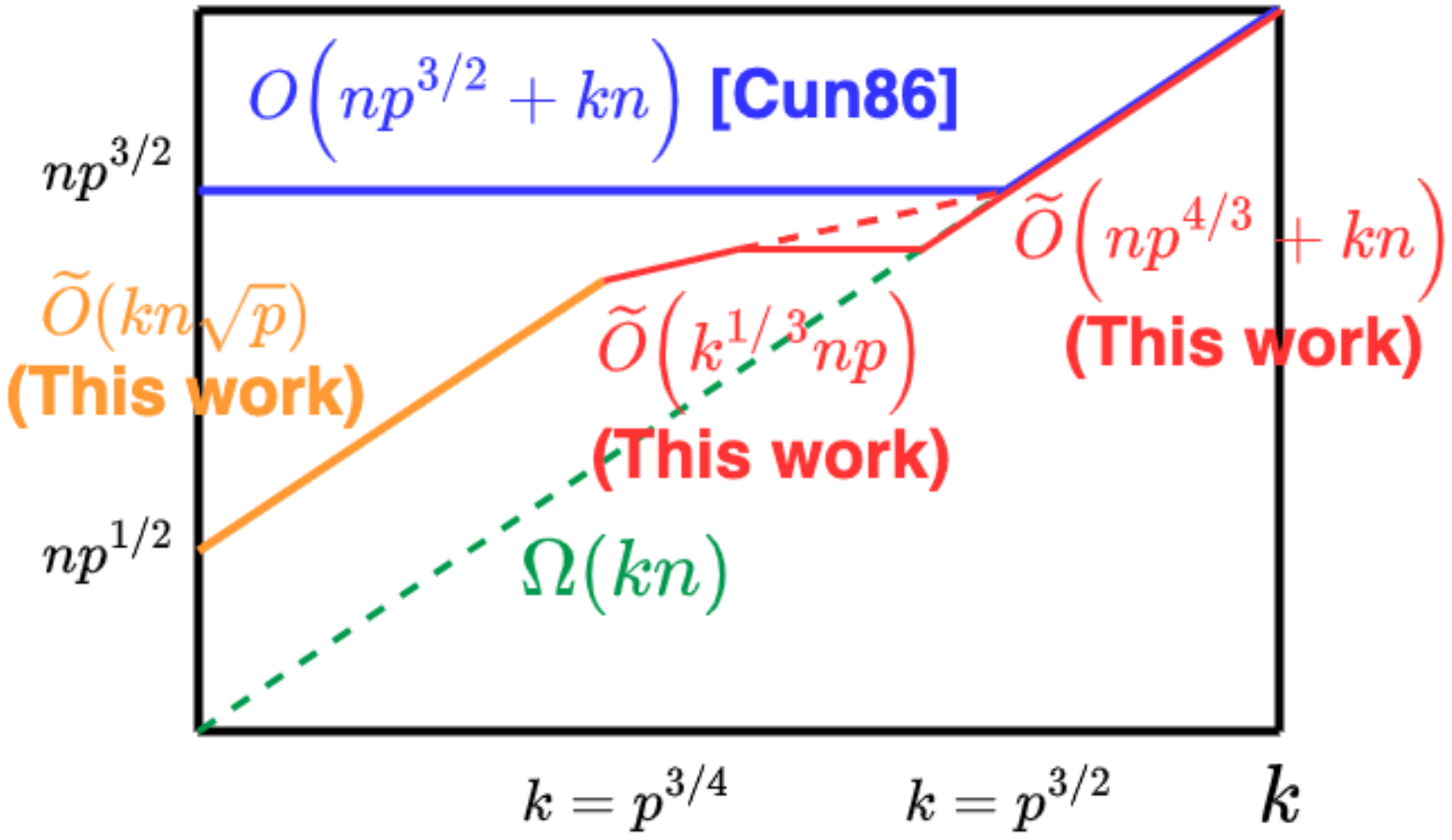
1回で $\tilde{O}(kn)$クエリ使用

Step 2. **辺再利用型増加路探索**

1回で $\tilde{O}(np)$クエリ使用

補題: Step2では$\Theta(k^{1/3})$回繰り返せば十分

補足: 実は、会議版から改善できた。



結論

マトロイド分割問題に対して**独立性オラクルの使用回数の少ない**アルゴリズムを設計

- **二分探索**により辺をみつけることで、簡潔交換グラフの辺を全て見ずに済む （Nguyễn 2019, Chakrabarty et al. 2019）
- **辺再利用型増加路探索**という新しいアイデア

Q. 本結果はさらに改善できるか？

Q. 辺再利用型増加路探索は他の問題にも適用できるか？