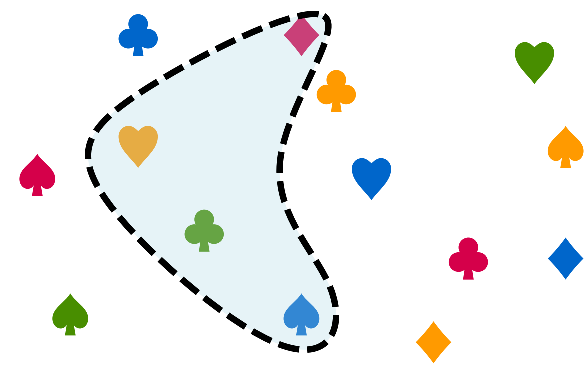
Fast Algorithms via Dynamic-Oracle Matroids

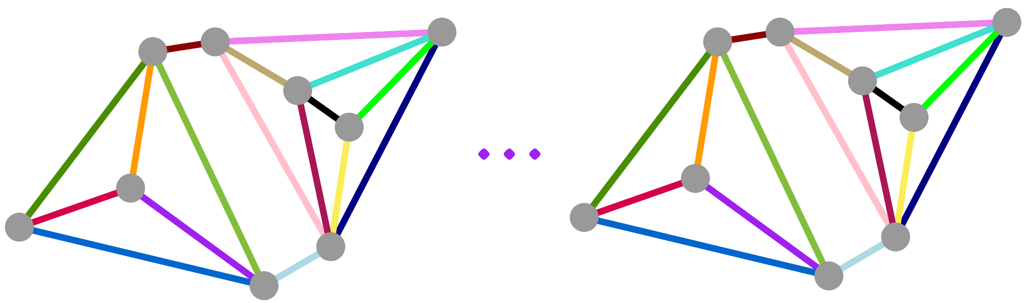
Joakim Blikstad[*] Sagnik Mukhopadhyay[‡]
Danupon Nanongkai[*†] Ta-Wei Tu[†]

ETH Zürich A&C online seminar

May 2023

To appear at STOC'23

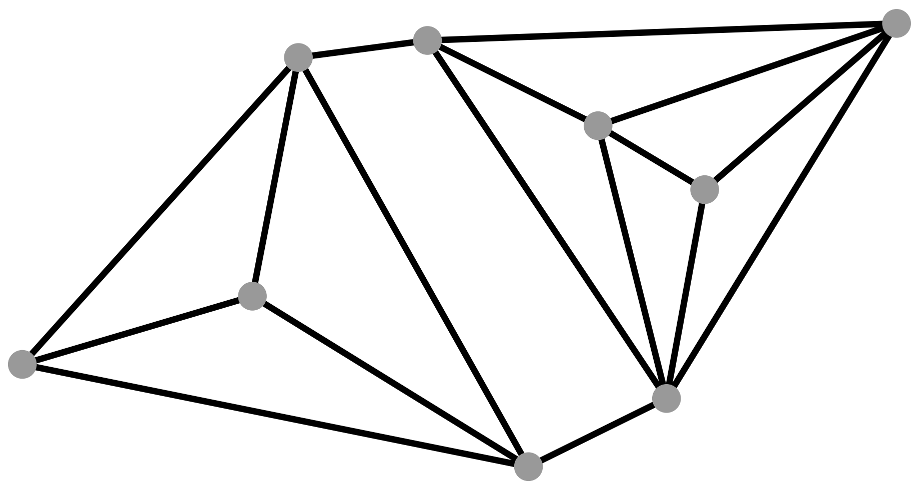




[*] KTH Royal Institute of Technology, Sweden
[†] Max-Planck Institute of Informatics, Germany
[‡] University of Sheffield, UK

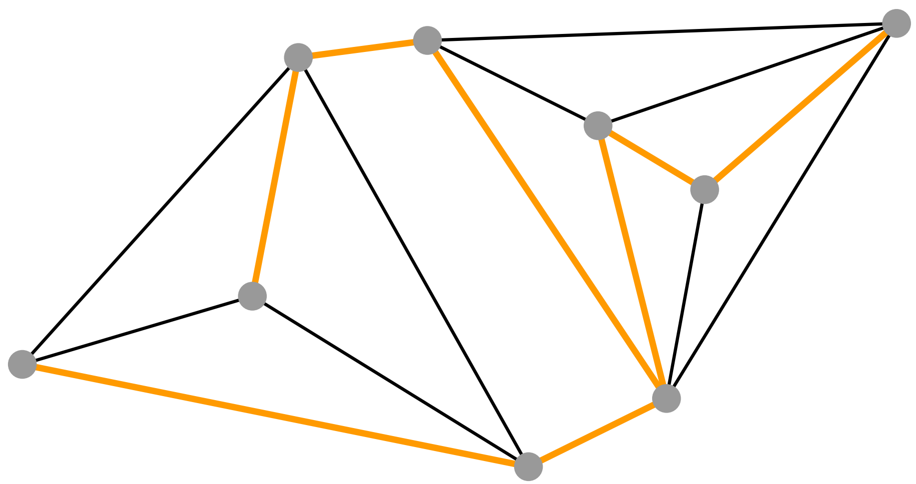
k-Disjoint Spanning Tree

Given: Graph $G = (V, E)$, integer $k \geq 1$;
Goal: Find k disjoint spanning trees.



k-Disjoint Spanning Tree

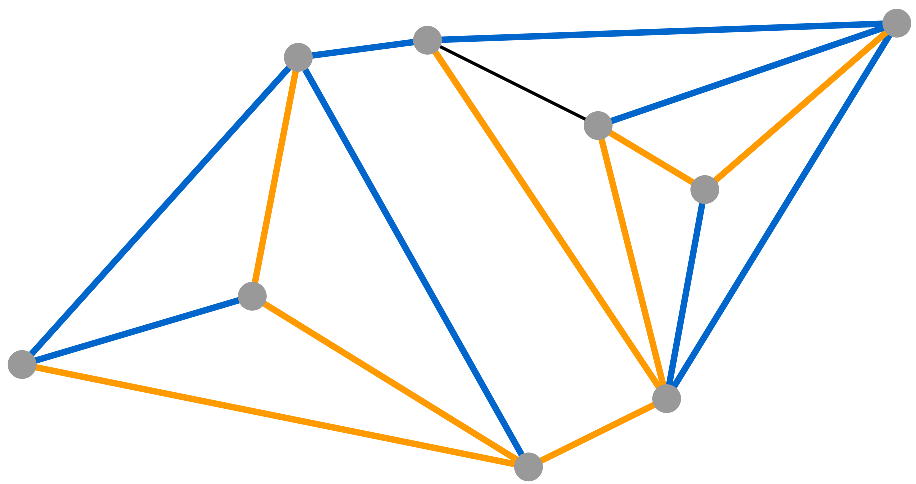
Given: Graph $G = (V, E)$, integer $k \geq 1$;
Goal: Find k disjoint spanning trees.



$k = 1$

k-Disjoint Spanning Tree

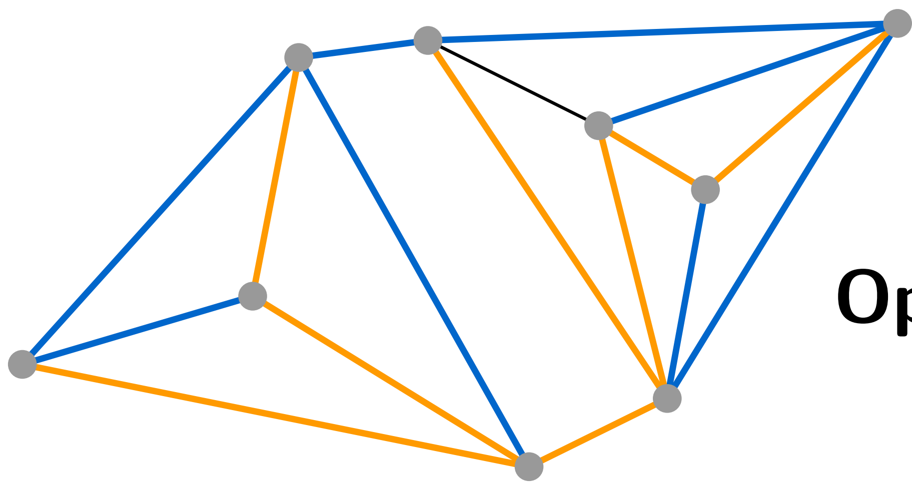
Given: Graph $G = (V, E)$, integer $k \geq 1$;
Goal: Find k disjoint spanning trees.



$k = 2$

k-Disjoint Spanning Tree

Given: Graph $G = (V, E)$, integer $k \geq 1$;
Goal: Find k disjoint spanning trees.

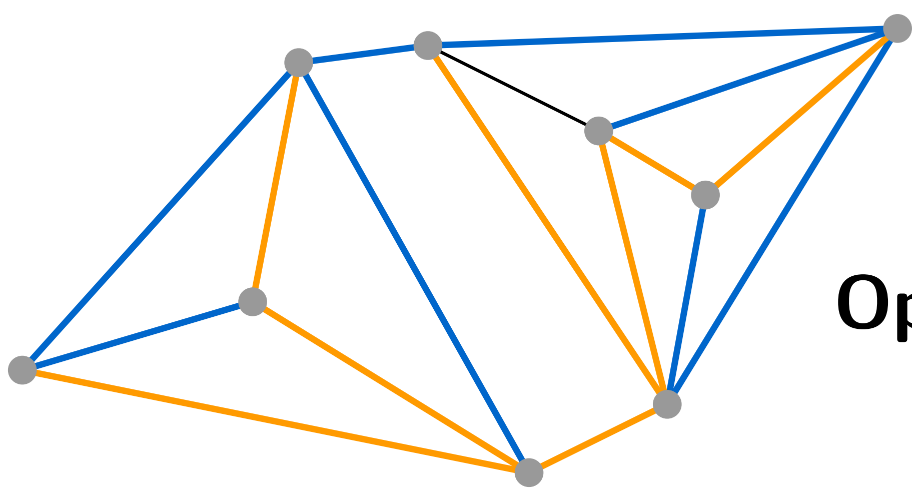


$k = 2$

Open: $\tilde{O}(|E|)$ time even for $k = 2$?

k-Disjoint Spanning Tree

Given: Graph $G = (V, E)$, integer $k \geq 1$;
Goal: Find k disjoint spanning trees.



$k = 2$

Open: $\tilde{O}(|E|)$ time even for $k = 2$?

$\tilde{O}_k(|V|\sqrt{|E|})$ [Gabow-Westerman STOC'88]
$\tilde{O}_k(|E| + |V|\sqrt{|V|})$ **Ours**†

†Also concurrently by [Quanrud'23]

Graph Problems & Reductions

Want *Unified* way to design *Efficient* algorithms.

Graph Problems & Reductions

Want *Unified* way to design *Efficient* algorithms.

Min-Cost Max-Flow†

- Directed Cut
- Closure Problem
- Gomory-Hu Trees
- Edge Connectivity
- Airline Scheduling
- Vertex Connectivity
- Bipartite Matching
- NWSSSP

†Almost linear time, [Chen, Kyng, Liu, Peng, Probst Gutenberg, Sachdeva FOCS'22]

Graph Problems & Reductions

Want *Unified* way to design *Efficient* algorithms.

Min-Cost Max-Flow†

Directed Cut

Closure Problem

Gomory-Hu Trees

Edge Connectivity

Airline Scheduling

Vertex Connectivity

Bipartite Matching

NWSSSP

Graphic Matroid Intersection

Job Scheduling Matroid Intersection

Arboricity

Tree Packing

Colorful Spanning Trees

k-Disjoint Spanning Trees

†Almost linear time, [Chen, Kyng, Liu, Peng, Probst Gutenberg, Sachdeva FOCS'22]

Graph Problems & Reductions

Want *Unified* way to design *Efficient* algorithms.

Min-Cost Max-Flow†

- Directed Cut
- Closure Problem
- Gomory-Hu Trees
- Edge Connectivity
- Airline Scheduling
- Vertex Connectivity
- Bipartite Matching
- NWSSSP

Matroid Intersection/Union

- NWSSSP
- Bipartite Matching
- Vertex Connectivity
- Graphic Matroid Intersection
- Job Scheduling Matroid Intersection
- k-Disjoint Spanning Trees
- Colorful Spanning Trees
- Tree Packing
- Arboricity

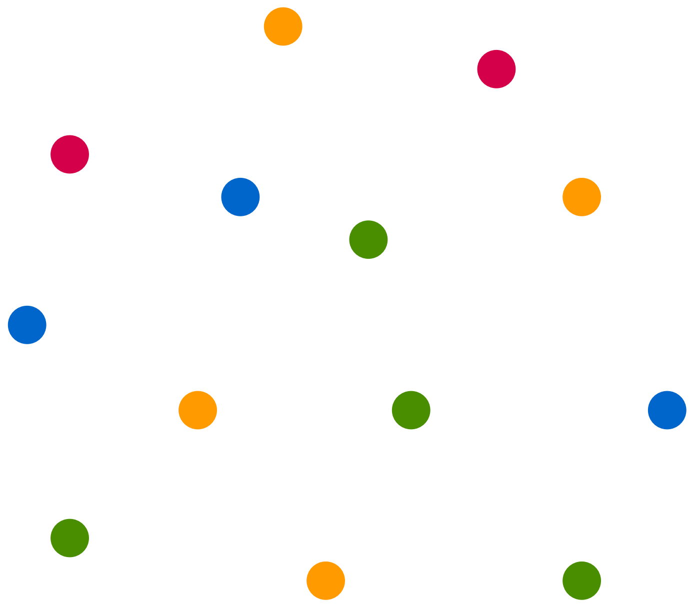
†Almost linear time, [Chen, Kyng, Liu, Peng, Probst Gutenberg, Sachdeva FOCS'22]

Matroid Problems

Matroids

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

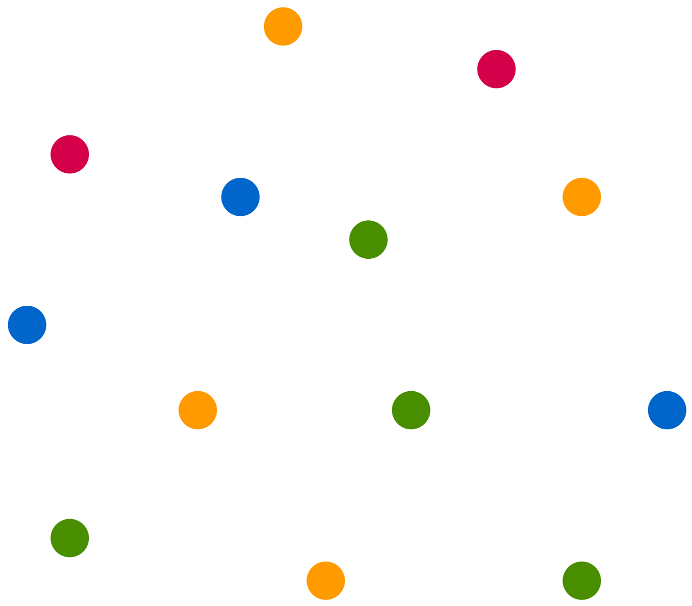
1. Ground set U of n elements



Matroids

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

1. Ground set U of n elements
2. Notion of independence $\mathcal{I}$

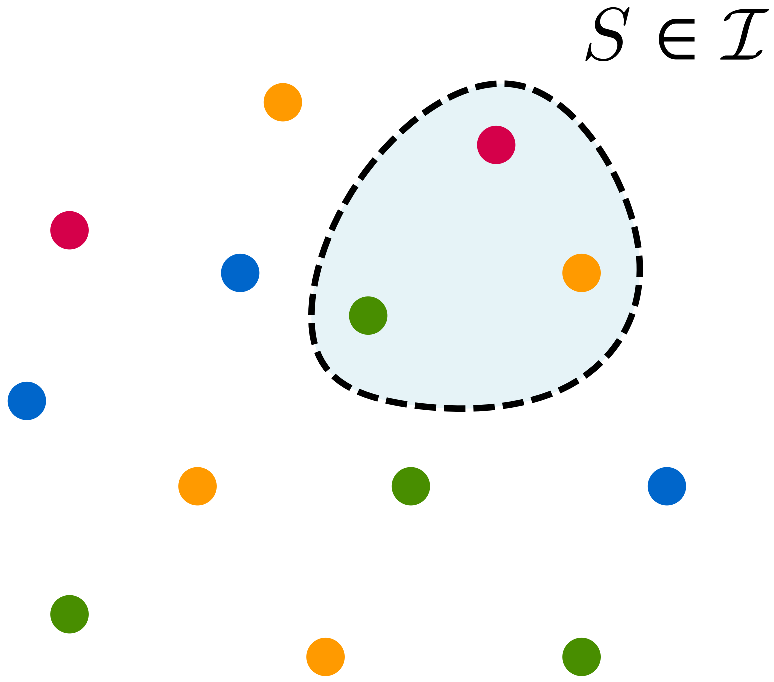


Eg. Colourful Matroid
"no duplicate colours"

Matroids

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

1. Ground set U of n elements
2. Notion of independence $\mathcal{I}$

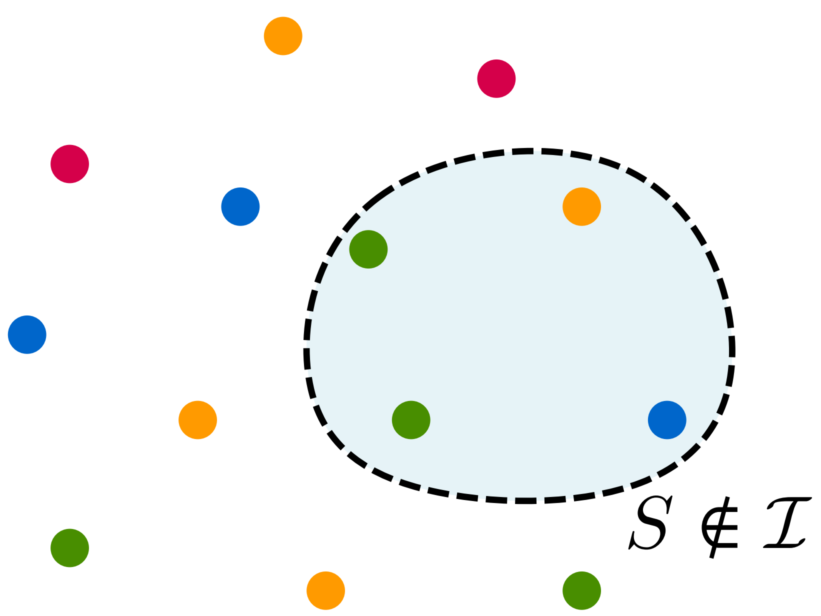


Eg. Colourful Matroid
"no duplicate colours"

Matroids

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

1. Ground set U of n elements
2. Notion of independence $\mathcal{I}$

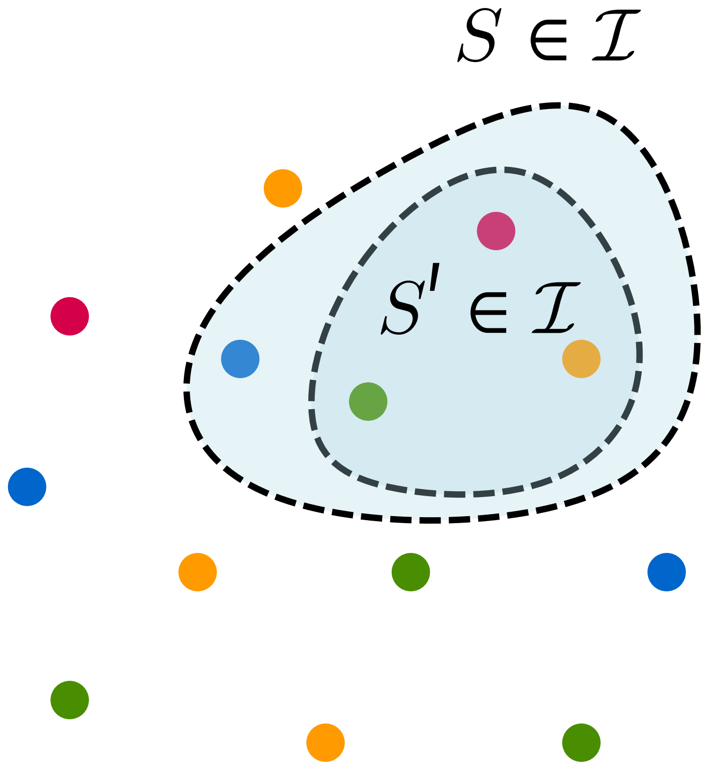


Eg. Colourful Matroid
"no duplicate colours"

Matroids

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

1. Ground set U of n elements
2. Notion of independence $\mathcal{I}$
   - Downward closure



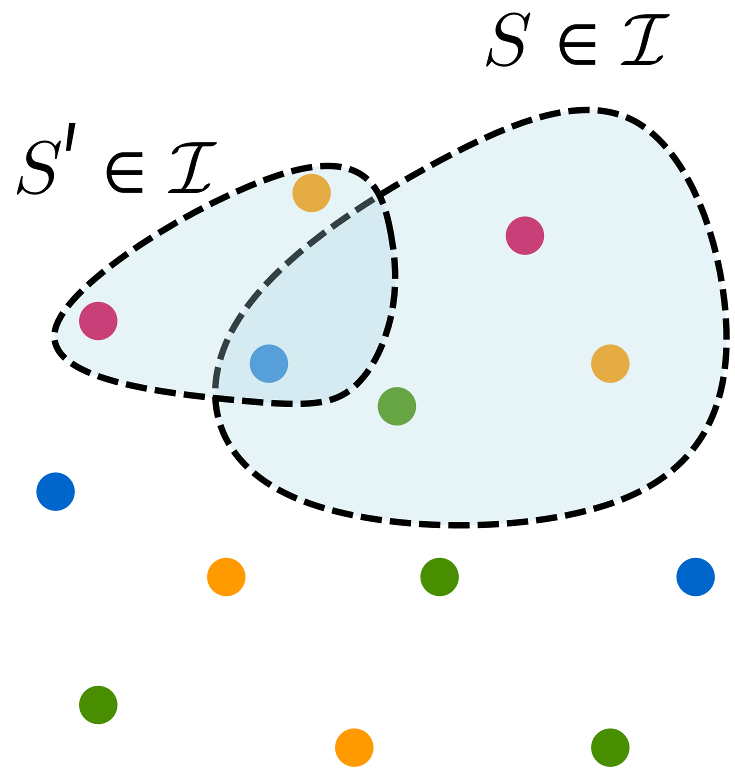
Eg. Colourful Matroid
"no duplicate colours"

Matroids

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

1. Ground set U of n elements
2. Notion of independence $\mathcal{I}$
   - Downward closure
   - Exchange property

   "All maximal independent sets have the same size"



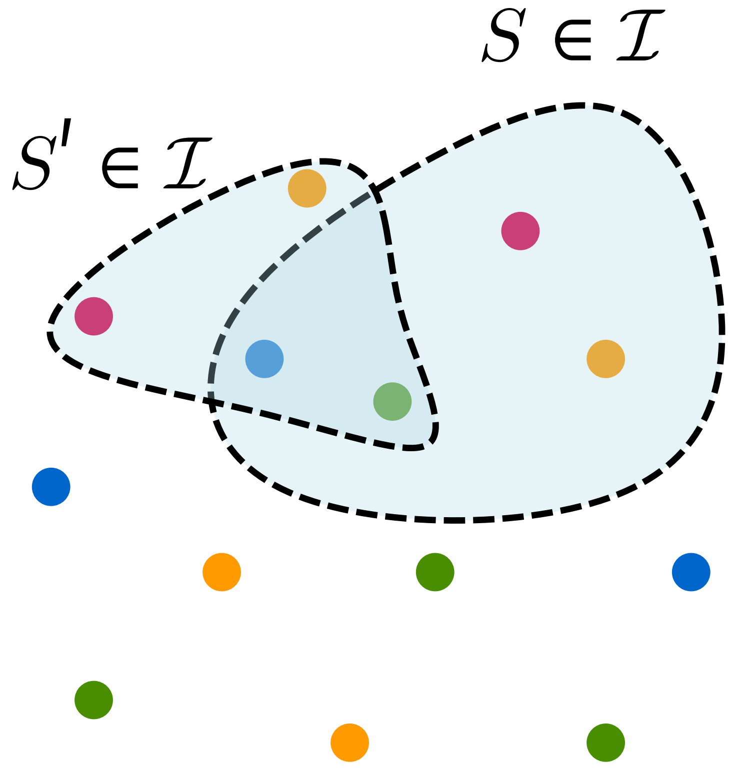
Eg. Colourful Matroid
"no duplicate colours"

Matroids

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

1. Ground set U of n elements
2. Notion of independence $\mathcal{I}$
   - Downward closure
   - Exchange property

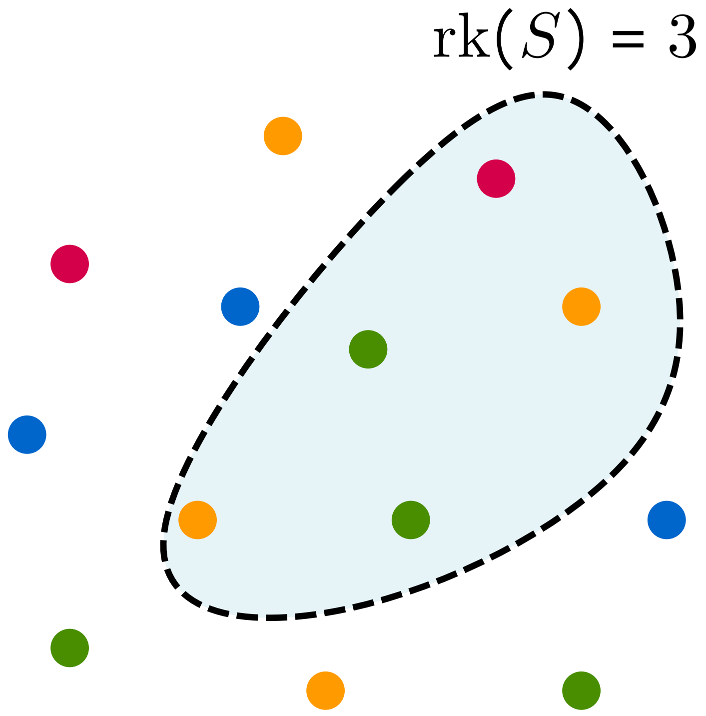
   "All maximal independent sets have the same size"



Eg. Colourful Matroid
"no duplicate colours"

Matroid Rank

$$\mathrm{rk}(S) = \max\{|A| : A \subseteq S, A \in \mathcal{I}\}$$

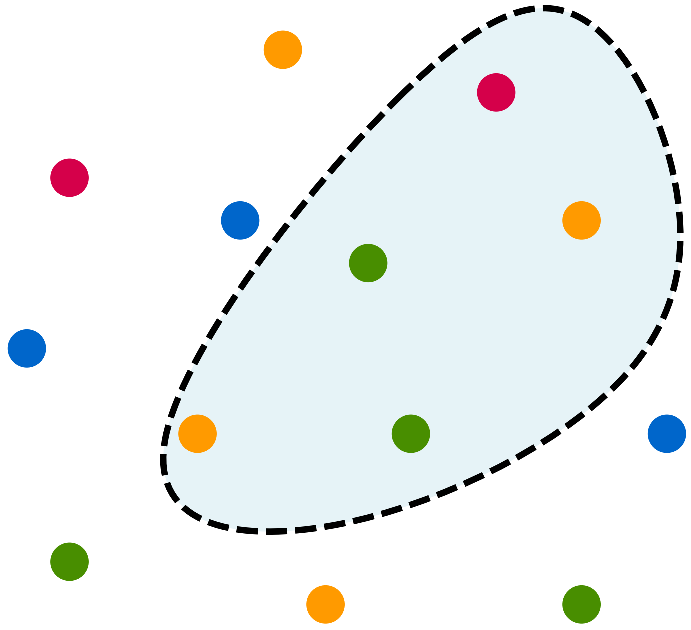


Matroid Rank

$$\mathrm{rk}(S) = \max\{|A| : A \subseteq S, A \in \mathcal{I}\}$$

$$= \text{size of a maximum independent set in } S$$

$\mathrm{rk}(S) = 3$

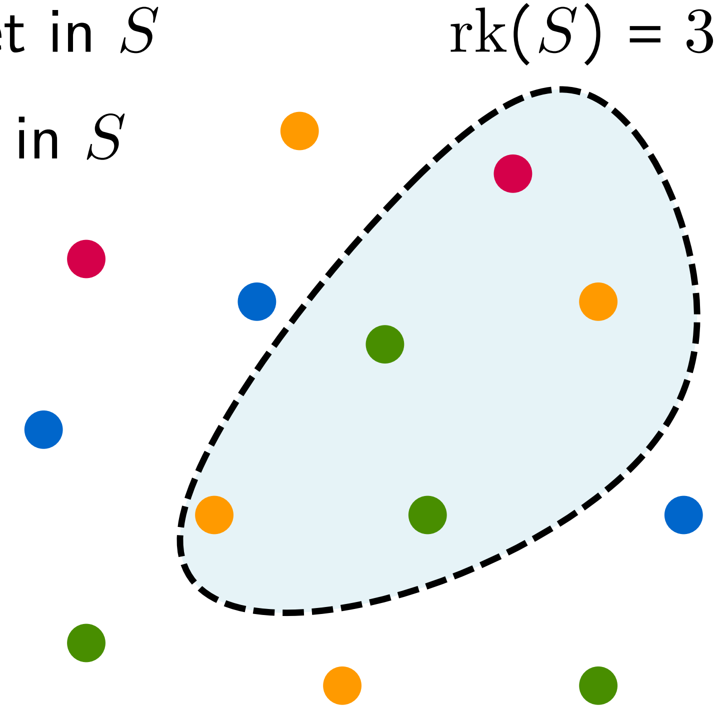


Matroid Rank

$$\mathrm{rk}(S) = \max\{|A| : A \subseteq S, A \in \mathcal{I}\}$$

= size of a maximum independent set in S

= size of a *maximal* independent set in S



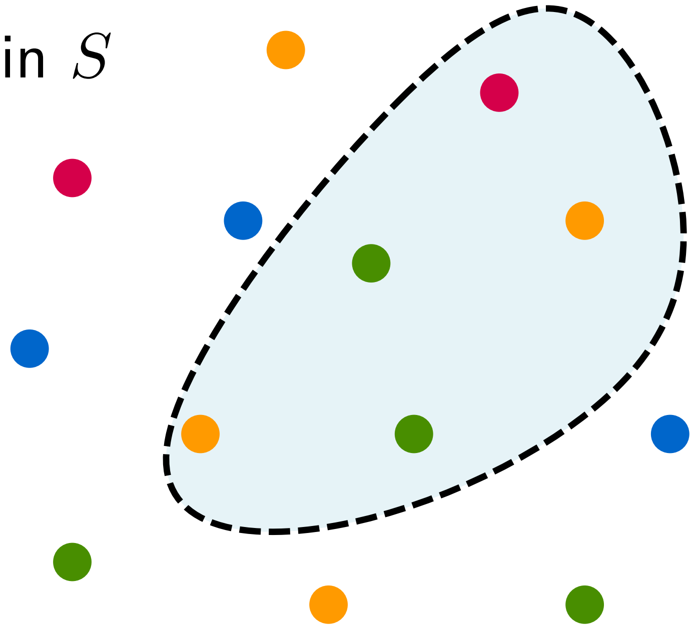
Matroid Rank

$$\mathrm{rk}(S) = \max\{|A| : A \subseteq S, A \in \mathcal{I}\}$$

= size of a maximum independent set in S

= size of a *maximal* independent set in S

$\mathrm{rk}(S) = 3$

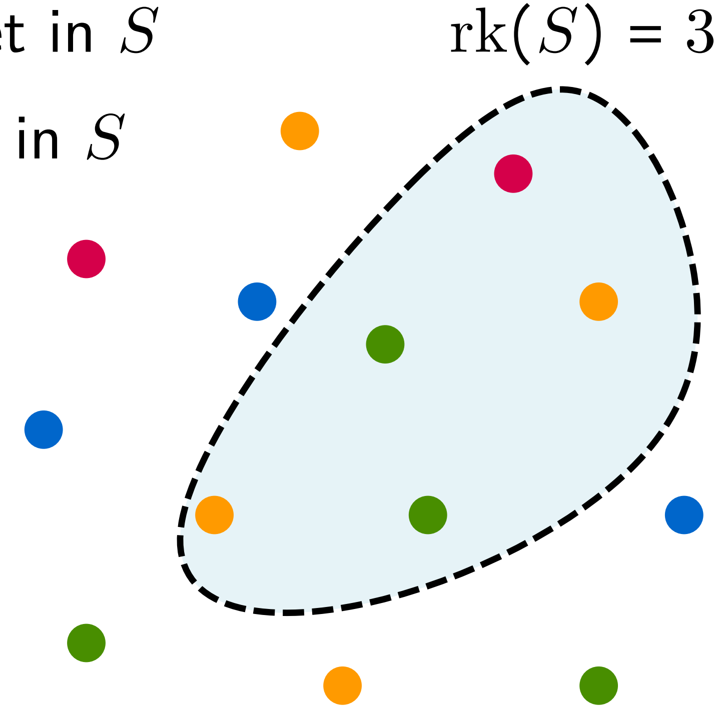


Properties:

- $S \in \mathcal{I} \iff \mathrm{rk}(S) = |S|$

Matroid Rank

$$\begin{aligned}\mathrm{rk}(S) &= \max\{|A| : A \subseteq S, A \in \mathcal{I}\}\\ &= \text{size of a maximum independent set in } S\\ &= \text{size of a \textit{maximal} independent set in } S\end{aligned}$$

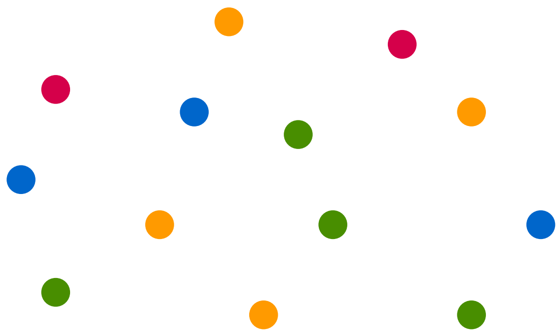


Properties:

- $S \in \mathcal{I} \iff \mathrm{rk}(S) = |S|$
- Submodular (Diminishing returns)
  If $A \subseteq B$, and $x \notin B$ then:
  $\mathrm{rk}(A+x) - \mathrm{rk}(A) \geq \mathrm{rk}(B+x) - \mathrm{rk}(B)$

Matroids: Examples

Colourful Matroid

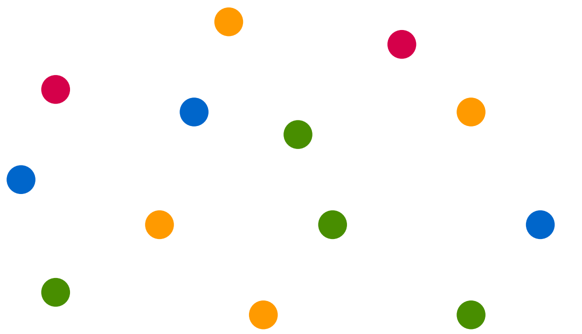


$\mathcal{I}$ =“no duplicate colours”

rk =“number of distinct colours”

Matroids: Examples

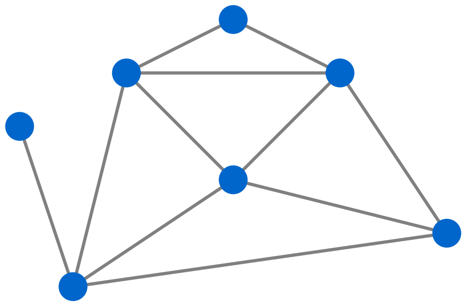
Colourful Matroid



$\mathcal{I}$ =“no duplicate colours”

rk =“number of distinct colours”

Graphic Matroid



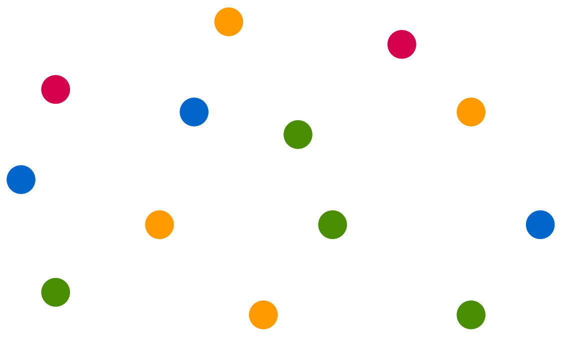
U = edges

$\mathcal{I}$ =“no cycles”

rk =“#vertices - #components”

Matroids: Examples

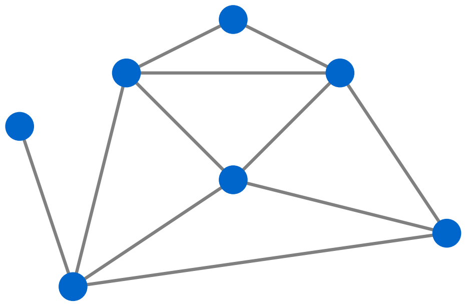
Colourful Matroid



$\mathcal{I}$ =“no duplicate colours”

rk =“number of distinct colours”

Graphic Matroid



U = edges

$\mathcal{I}$ =“no cycles”

rk =“#vertices - #components”

Linear Matroid

$$(2,1,4,2,3,3)$$

$$(1,0,1,0,1,0)$$

$$(3,1,5,2,4,3)$$

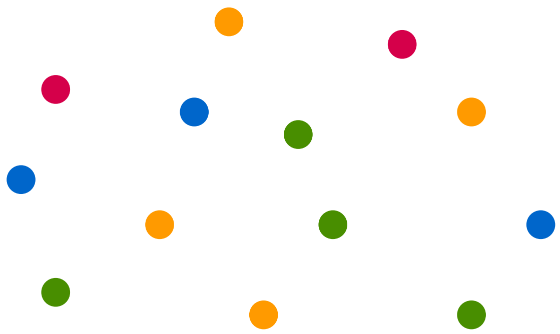
U = vectors

$\mathcal{I}$ =“linear independence”

rk =rank

Matroids: Examples

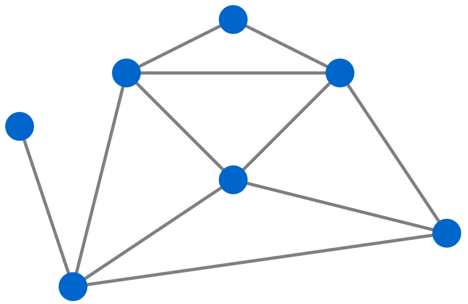
Colourful Matroid



$\mathcal{I}$ =“no duplicate colours”

rk =“number of distinct colours”

Graphic Matroid



U = edges

$\mathcal{I}$ =“no cycles”

rk =“#vertices - #components”

Linear Matroid

$(2, 1, 4, 2, 3, 3)$

$(1, 0, 1, 0, 1, 0)$

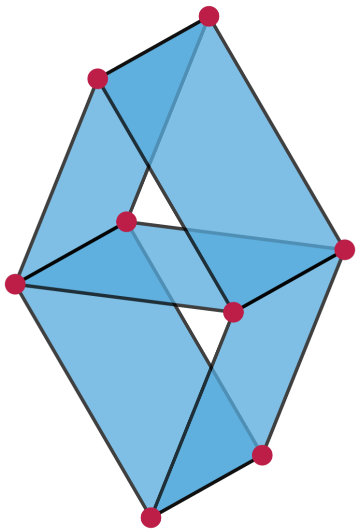
$(3, 1, 5, 2, 4, 3)$

U = vectors

$\mathcal{I}$ =“linear independence”

rk =rank

Vámos Matroid



Matroid Problems

Matroid Intersection:
Given two matroids $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$,
find a set S of maximum size in $\mathcal{I}_1 \cap \mathcal{I}_2$.

Matroid Union: (a.k.a. matroid sum)
Given k matroids $\mathcal{M}_i = (U, \mathcal{I}_i)$,
find a set $S = S_1 \cup S_2 \cup \cdots \cup S_k$ of maximum size, where $S_i \in \mathcal{I}_i$.

k-**Fold Matroid Union:** (a.k.a. partitioning)
Special case of matroid union where all k matroids are the same.

Matroid Intersection

Given two matroids:

- $\mathcal{M}_1 = (V, \mathcal{I}_1)$
- $\mathcal{M}_2 = (V, \mathcal{I}_2)$

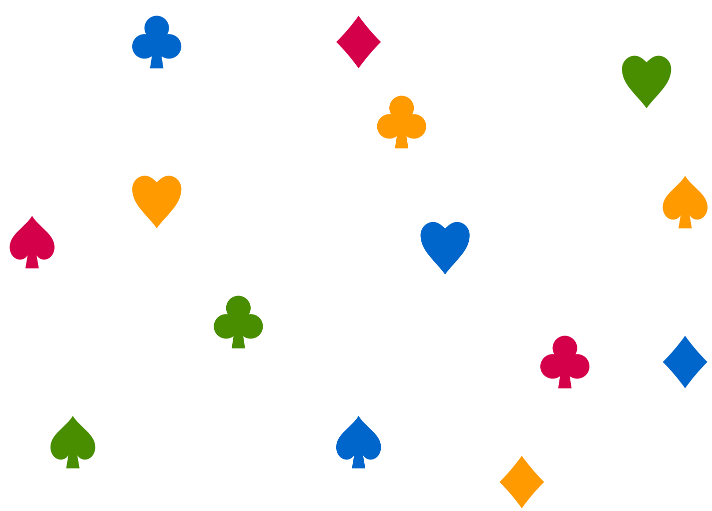
Find a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size.

Matroid Intersection

Given two matroids:

- $\mathcal{M}_1 = (V, \mathcal{I}_1)$
- $\mathcal{M}_2 = (V, \mathcal{I}_2)$

Find a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size.



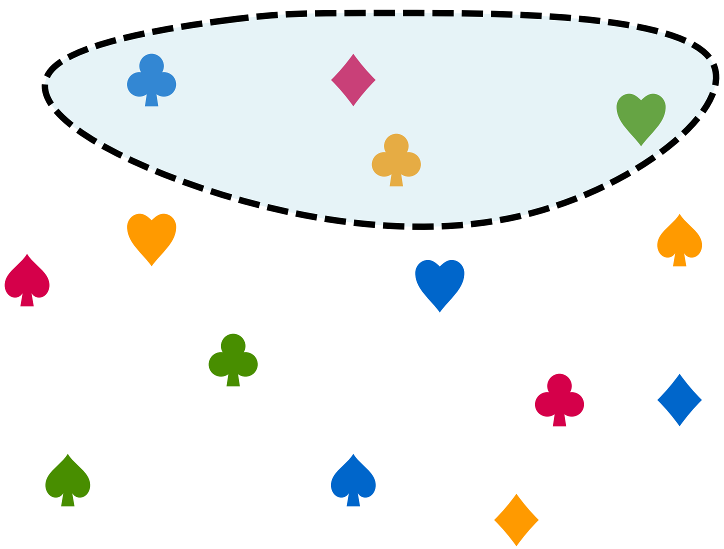
$\mathcal{M}_1 =$ "distinct suits"
$\mathcal{M}_2 =$ "distinct colours"

Matroid Intersection

Given two matroids:

- $\mathcal{M}_1 = (V, \mathcal{I}_1)$
- $\mathcal{M}_2 = (V, \mathcal{I}_2)$

Find a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size.



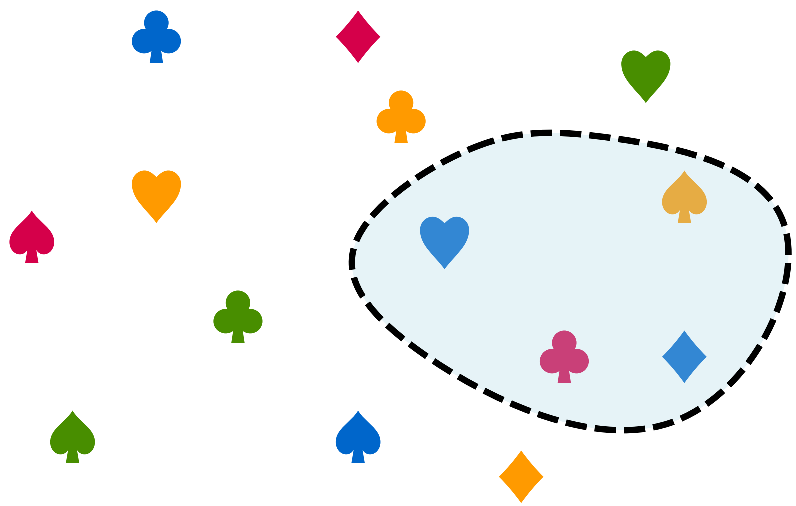
$\mathcal{M}_1 =$ "distinct suits"
$\mathcal{M}_2 =$ "distinct colours"

Matroid Intersection

Given two matroids:

- $\mathcal{M}_1 = (V, \mathcal{I}_1)$
- $\mathcal{M}_2 = (V, \mathcal{I}_2)$

Find a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size.



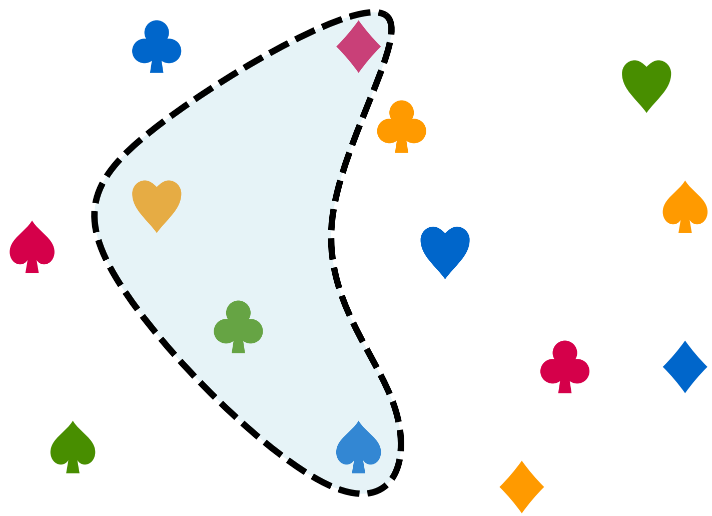
$\mathcal{M}_1 =$ "distinct suits"
$\mathcal{M}_2 =$ "distinct colours"

Matroid Intersection

Given two matroids:

- $\mathcal{M}_1 = (V, \mathcal{I}_1)$
- $\mathcal{M}_2 = (V, \mathcal{I}_2)$

Find a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size.



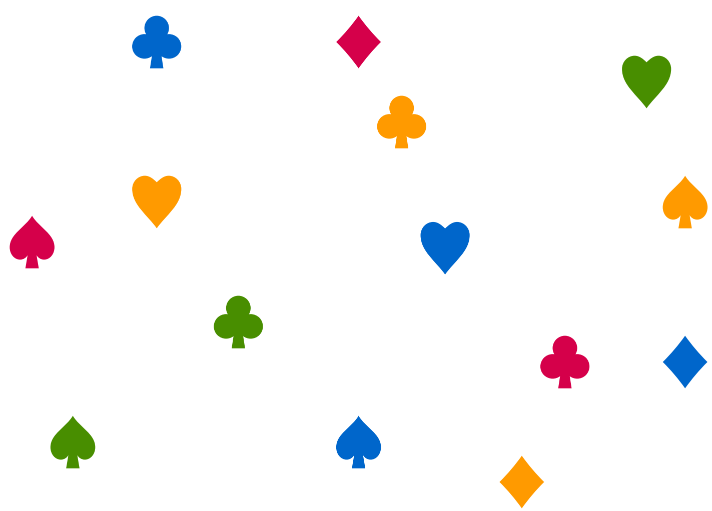
$\mathcal{M}_1 =$ "distinct suits"
$\mathcal{M}_2 =$ "distinct colours"

Matroid Intersection

Given two matroids:

- $\mathcal{M}_1 = (V, \mathcal{I}_1)$
- $\mathcal{M}_2 = (V, \mathcal{I}_2)$

Find a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size.



$\mathcal{M}_1 =$ "distinct suits"
$\mathcal{M}_2 =$ "distinct colours"

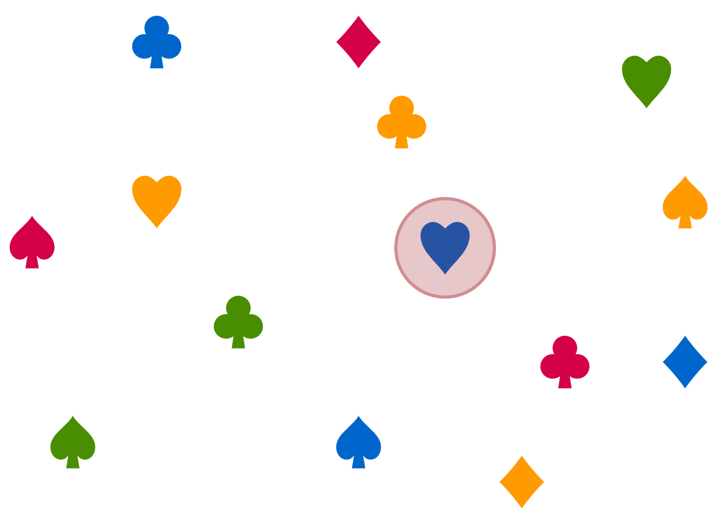
$\clubsuit$	Blue
$\blacklozenge$	Red
$\spadesuit$	Green
$\heartsuit$	Yellow

Matroid Intersection

Given two matroids:

- $\mathcal{M}_1 = (V, \mathcal{I}_1)$
- $\mathcal{M}_2 = (V, \mathcal{I}_2)$

Find a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size.



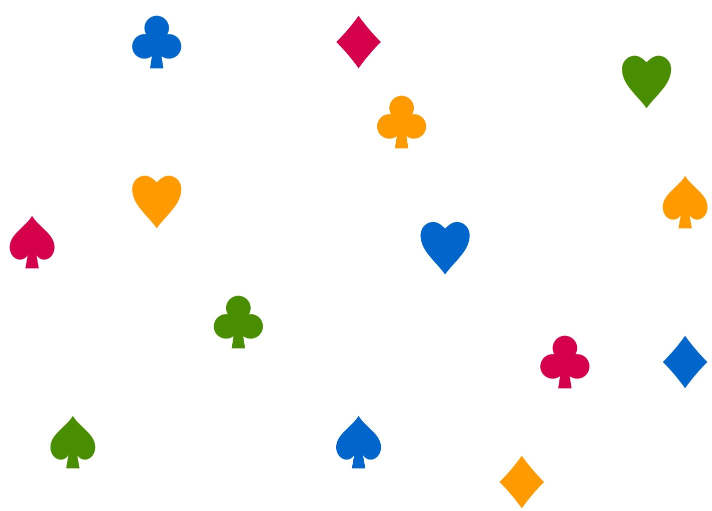
$\mathcal{M}_1 =$ "distinct suits"
$\mathcal{M}_2 =$ "distinct colours"

Matroid Intersection

Given two matroids:

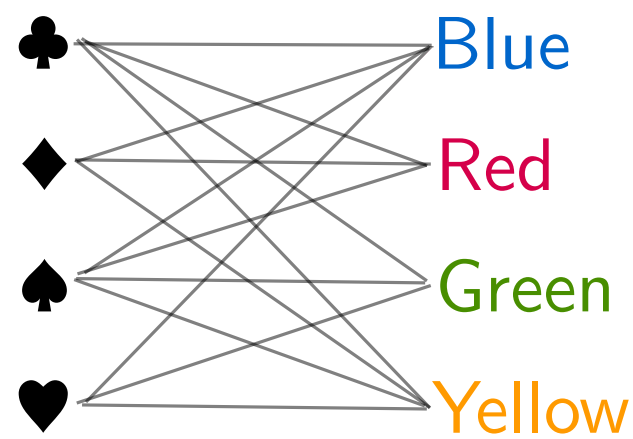
- $\mathcal{M}_1 = (V, \mathcal{I}_1)$
- $\mathcal{M}_2 = (V, \mathcal{I}_2)$

Find a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size.



$\mathcal{M}_1 =$ "distinct suits"

$\mathcal{M}_2 =$ "distinct colours"

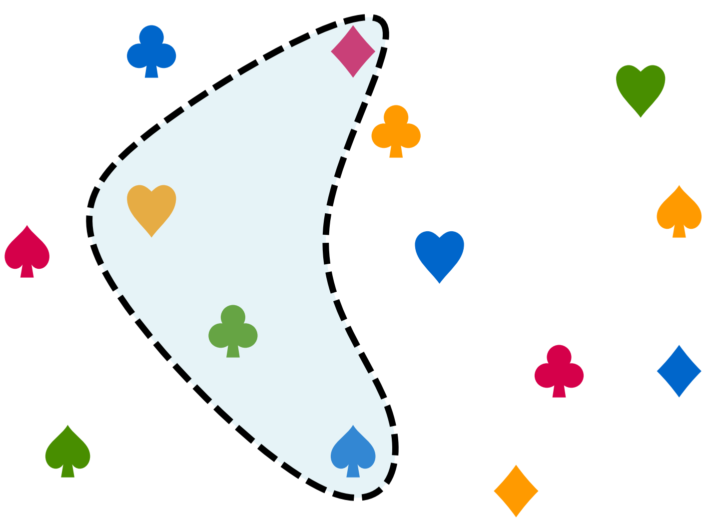


Matroid Intersection

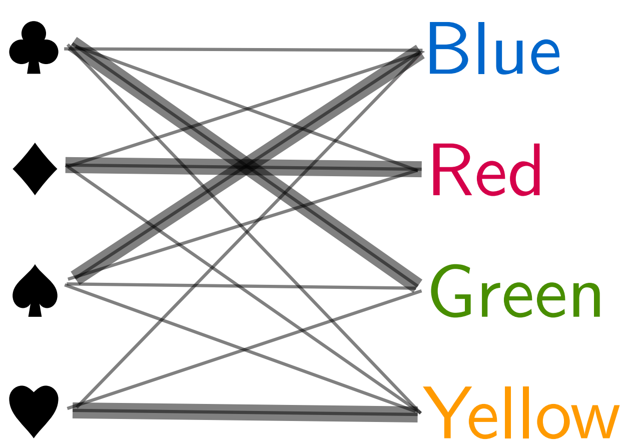
Given two matroids:

- $\mathcal{M}_1 = (V, \mathcal{I}_1)$
- $\mathcal{M}_2 = (V, \mathcal{I}_2)$

Find a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size.



$\mathcal{M}_1 =$ "distinct suits"
$\mathcal{M}_2 =$ "distinct colours"



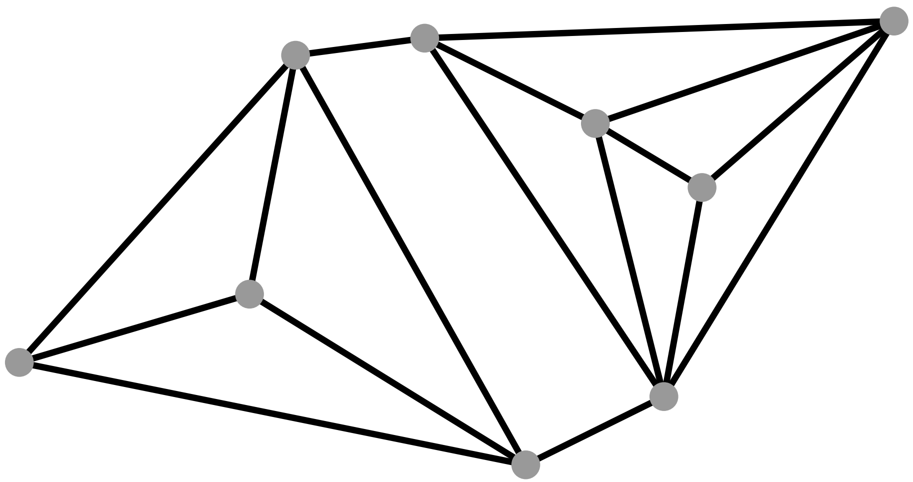
k-fold Matroid Union

Given: Matroid $\mathcal{M} = (U, \mathcal{I})$, integer k;
Goal: Find $S = S_1 \cup \cdots \cup S_k$ (where $S_i \in \mathcal{I}$) of maximum size.

.

k-fold Matroid Union

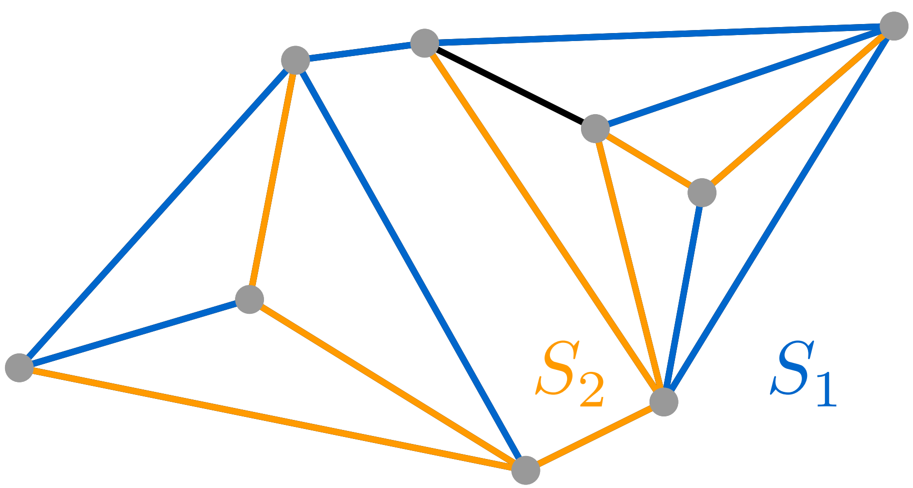
Given: Matroid $\mathcal{M} = (U, \mathcal{I})$, integer k;
Goal: Find $S = S_1 \cup \cdots \cup S_k$ (where $S_i \in \mathcal{I}$) of maximum size.



$k = 2$
$\mathcal{M}$ = graphic matroid
$S_i \subseteq E$ is in $\mathcal{I}$ iff no cycle.

k-fold Matroid Union

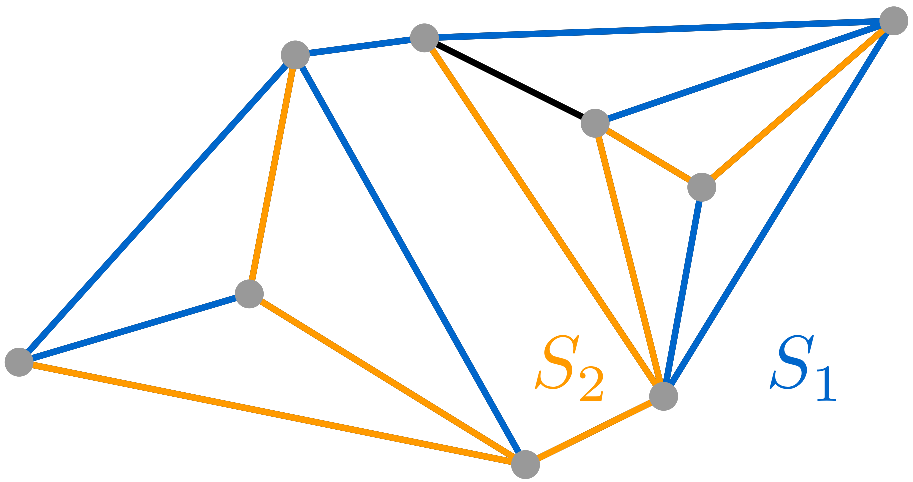
Given: Matroid $\mathcal{M} = (U, \mathcal{I})$, integer k;
Goal: Find $S = S_1 \cup \cdots \cup S_k$ (where $S_i \in \mathcal{I}$) of maximum size.



$k = 2$
$\mathcal{M}$ = graphic matroid
$S_i \subseteq E$ is in $\mathcal{I}$ iff no cycle.

k-fold Matroid Union

Given: Matroid $\mathcal{M} = (U, \mathcal{I})$, integer k;
Goal: Find $S = S_1 \cup \cdots \cup S_k$ (where $S_i \in \mathcal{I}$) of maximum size.

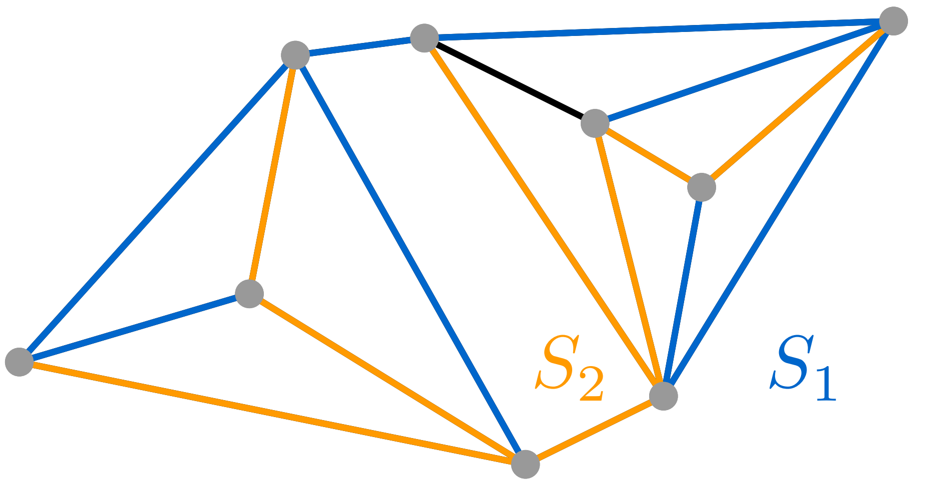


$k = 2$
$\mathcal{M}$ = graphic matroid
$S_i \subseteq E$ is in $\mathcal{I}$ iff no cycle.

Can solve using Matroid Intersection!

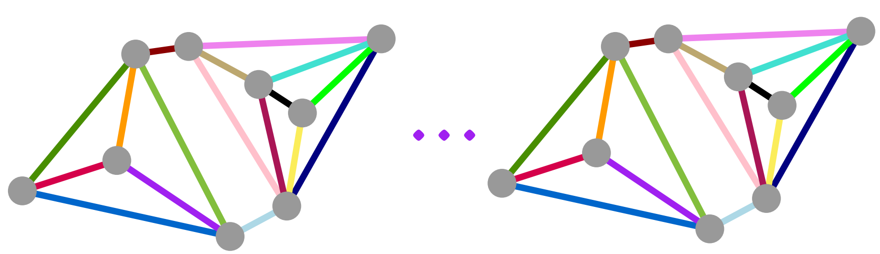
k-fold Matroid Union

Given: Matroid $\mathcal{M} = (U, \mathcal{I})$, integer k;
Goal: Find $S = S_1 \cup \cdots \cup S_k$ (where $S_i \in \mathcal{I}$) of maximum size.



$k = 2$
$\mathcal{M}$ = graphic matroid
$S_i \subseteq E$ is in $\mathcal{I}$ iff no cycle.

Can solve using Matroid Intersection!



$\mathcal{M}_1$ = colorful matroid
$\mathcal{M}_2$ = k independent copies of $\mathcal{M}$

Matroid Intersection & Union: Examples

- Bipartite matching
- k-disjoint spanning trees
- Arborescence (directed spanning tree)
- Colourful spanning tree
- Tree/Arborescence packing
- Some scheduling problems
- Some routing problems
- Some graph orientation problems
- ...

Also connections to *Submodular Function Minimization*

Query Access

How to access a matroid?

Query Access

How to access a matroid?

Oracle Access

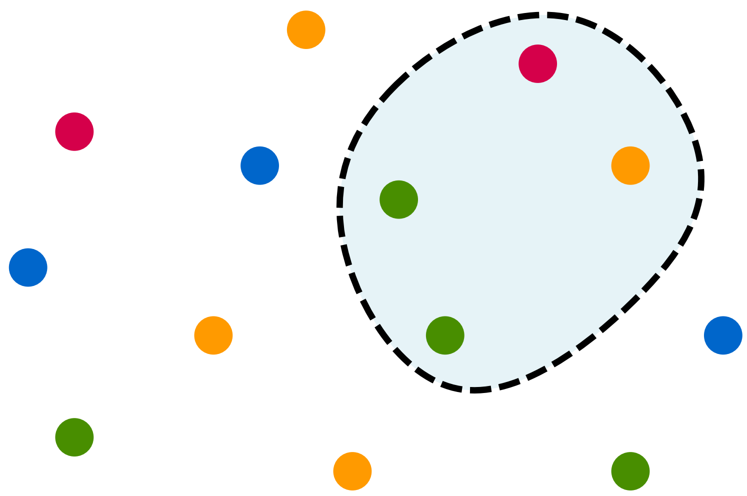
- Independence query: "Is $S \in \mathcal{I}$?"
- Rank query: "What is $\mathrm{rk}(S)$?"

Query Access

How to access a matroid?

Oracle Access

- Independence query: "Is $S \in \mathcal{I}$?"
- Rank query: "What is $\mathrm{rk}(S)$?"



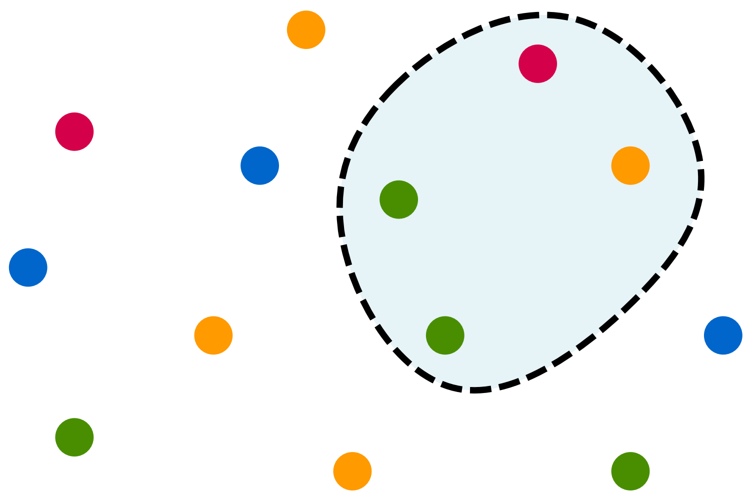
Query Access

How to access a matroid?

Oracle Access

- Independence query: "Is $S \in \mathcal{I}$?"
- Rank query: "What is $\mathrm{rk}(S)$?"

Important:
We do not know the underlying structure of the matroids!



Traditional Model

Traditional Model:

Minimize number of indep./rank queries measured in terms of:

- $n = |U|$ = number of elements (= #edges)
- $r = |S|$ = size of answer (≤ #vertices)

Traditional Model

Traditional Model:
Minimize number of indep./rank queries measured in terms of:

- $n = |U|$ = number of elements (= #edges)
- $r = |S|$ = size of answer (≤ #vertices)

State-of-the-art: Matroid Intersection & Union

- $\tilde{O}(n\sqrt{r})$ rank-queries [CLSSW FOCS'19]
- $\tilde{O}(nr^{3/4})$ indep-queries [BvdBMN STOC'21, Blikstad ICALP'21]

Traditional Model

Traditional Model:
Minimize number of indep./rank queries measured in terms of:
- $n = |U|$ = number of elements (= #edges)
- $r = |S|$ = size of answer (≤ #vertices)

State-of-the-art: Matroid Intersection & Union
- $\tilde{O}(n\sqrt{r})$ rank-queries [CLSSW FOCS'19]
- $\tilde{O}(nr^{3/4})$ indep-queries [BvdBMN STOC'21, Blikstad ICALP'21]

Open: $\tilde{O}(n)$ possible?

Traditional Model

Traditional Model:
Minimize number of indep./rank queries measured in terms of:

- $n = |U|$ = number of elements (= #edges)
- $r = |S|$ = size of answer (≤ #vertices)

State-of-the-art: Matroid Intersection & Union

- $\tilde{O}(n\sqrt{r})$ rank-queries [CLSSW FOCS'19]
- $\tilde{O}(nr^{3/4})$ indep-queries [BvdBMN STOC'21, Blikstad ICALP'21]

Open: $\tilde{O}(n)$ possible?

Caveat:
Does not imply *Efficient* algorithms.
Query "$\mathrm{rk}(Q)$?" takes $O(|Q|)$ time to specify, let alone answer.

Traditional Model

Traditional Model:
Minimize number of indep./rank queries measured in terms of:
- $n = |U|$ = number of elements (= #edges)
- $r = |S|$ = size of answer ($\leq$ #vertices)

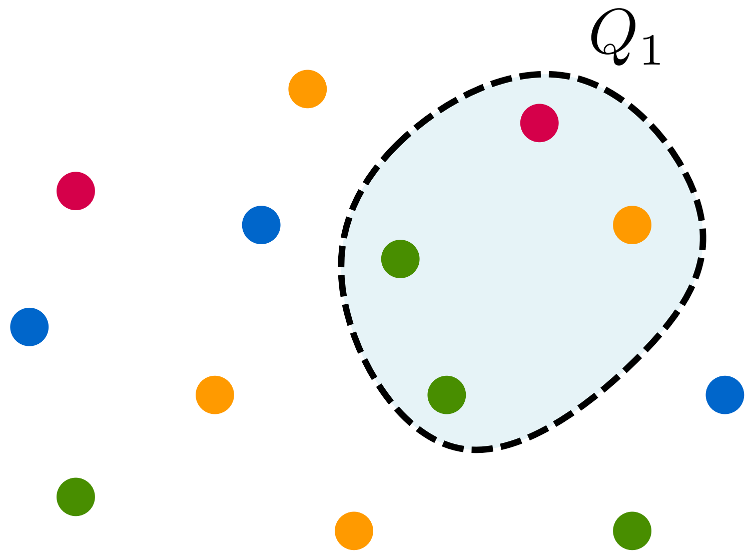
State-of-the-art: Matroid Intersection & Union
- $\tilde{O}(n\sqrt{r})$ rank-queries [CLSSW FOCS'19]
- $\tilde{O}(nr^{3/4})$ indep-queries [BvdBMN STOC'21, Blikstad ICALP'21]

Open: $\tilde{O}(n)$ possible?

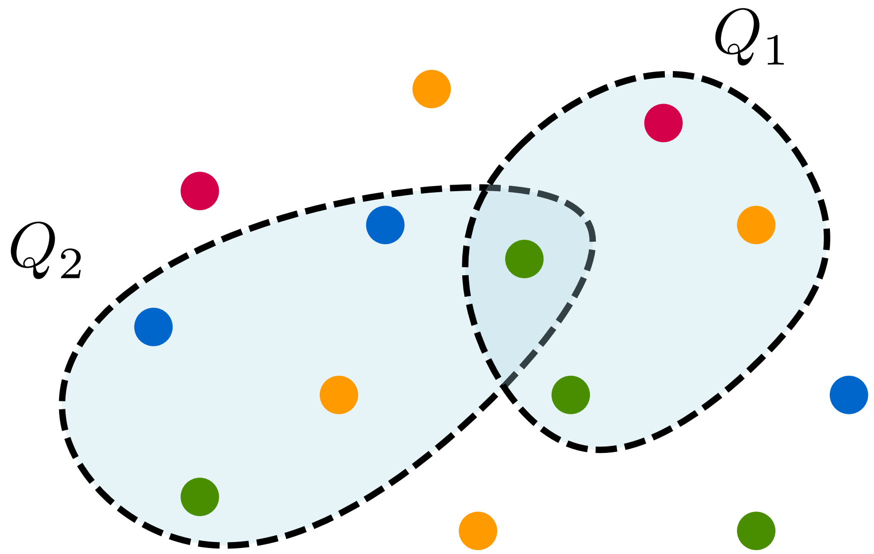
Caveat:
Does not imply *Efficient* algorithms.
Query "$\mathrm{rk}(Q)$?" takes $O(|Q|)$ time to specify, let alone answer.

Many papers in the 80s/90s specialize matroid intersection/union framework to specific problems

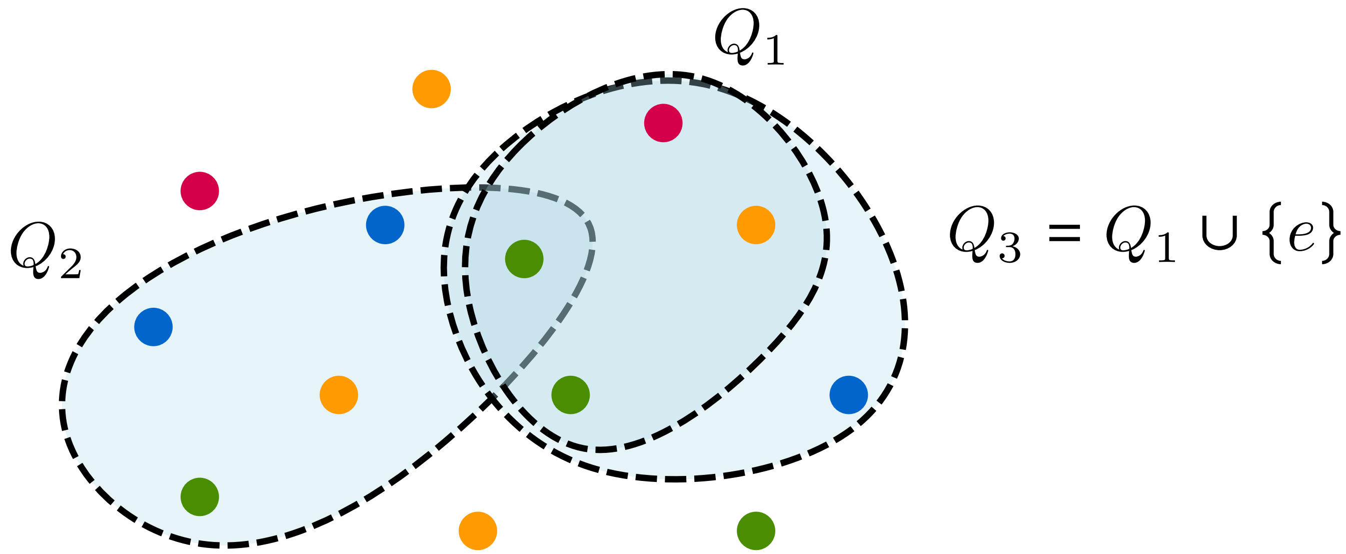
Dynamic Oracle



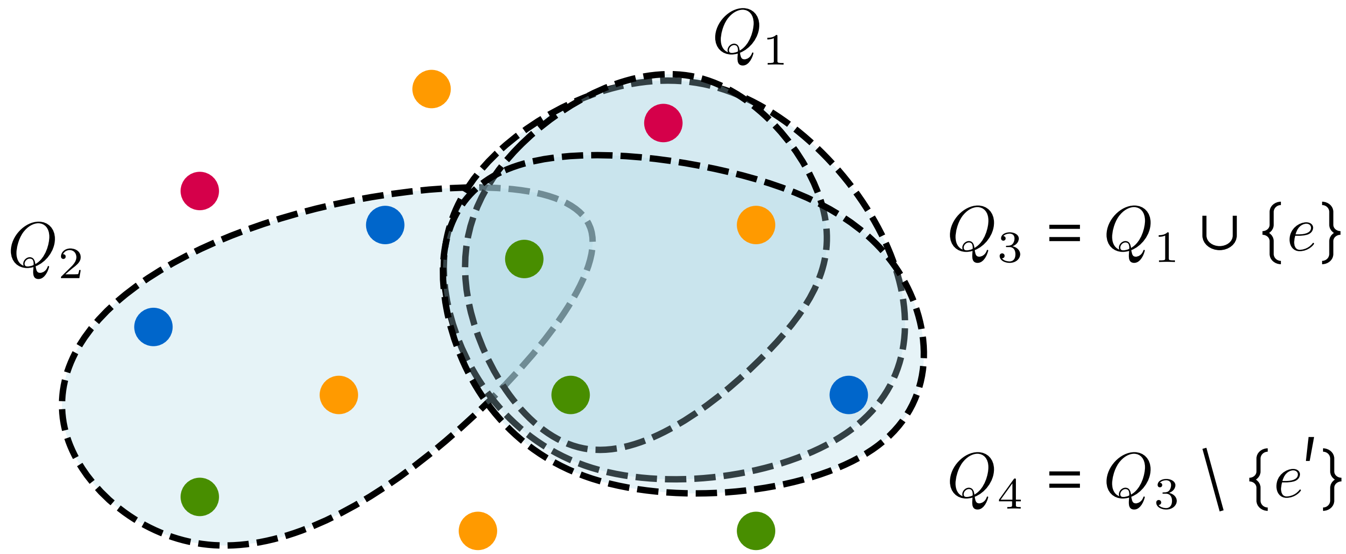
Dynamic Oracle



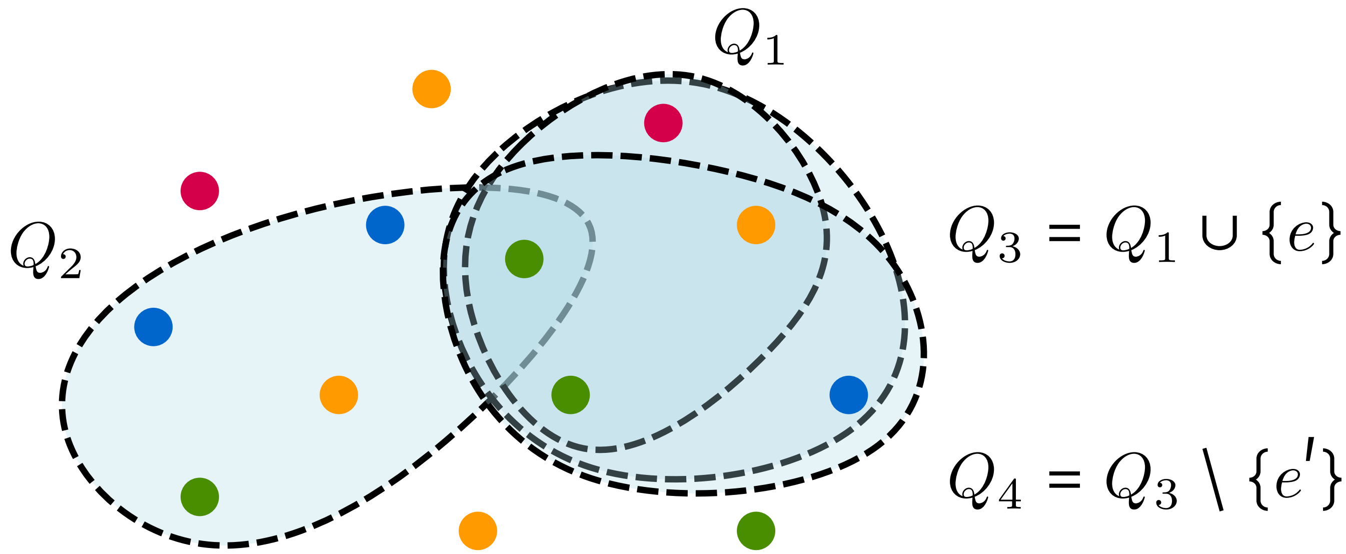
Dynamic Oracle



Dynamic Oracle



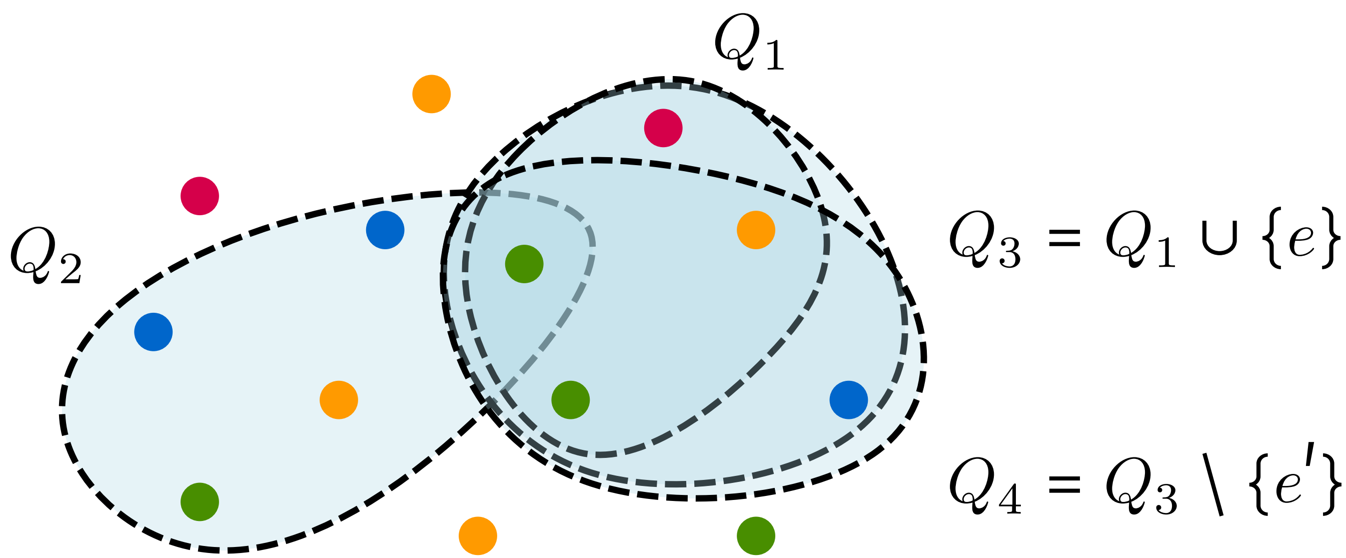
Dynamic Oracle



Main Motivation:

Cost to answer a query ≈ how different it is to previous queries.

Dynamic Oracle



Main Motivation:

Cost to answer a query ≈ how different it is to previous queries.

New Dynamic Oracle Model:

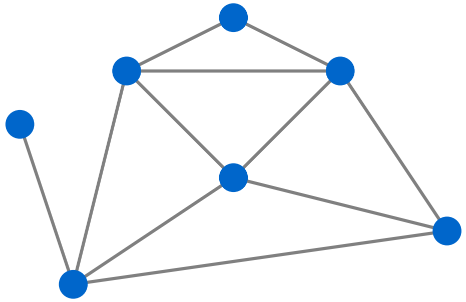
Cost to issue the k'th query Q_k is $\min_{i<k} |Q_k \oplus Q_i|$.

$\iff$

Query $Q_k = Q_i \pm \{e\}$.

Data Structures for Dynamic Rank Oracle

- **Colourful/partition matroid:**
  Count colors in $O(1)$ update time.
- **Graphic matroid:**
  Count components in $O(\text{polylog} n)$ (or $O(n^{o(1)})$) update time.
  [KKM'13, GKKKT'15, CGLNPS'20, NSW'17]

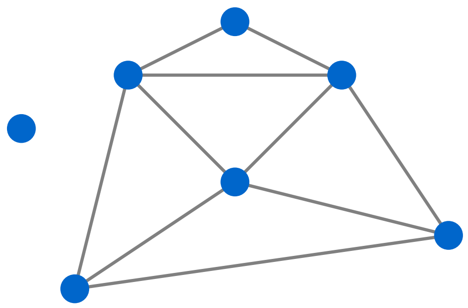


(delete / add edges)

Data Structures for Dynamic Rank Oracle

- **Colourful/partition matroid:**
  Count colors in $O(1)$ update time.
- **Graphic matroid:**
  Count components in $O(\text{polylog} n)$ (or $O(n^{o(1)})$) update time.

[KKM'13, GKKKT'15, CGLNPS'20, NSW'17]

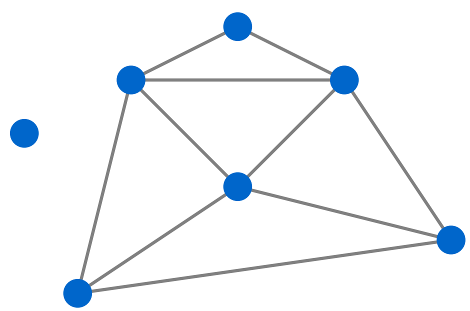


(delete / add edges)

Data Structures for Dynamic Rank Oracle

- **Colourful/partition matroid:**
  Count colors in $O(1)$ update time.
- **Graphic matroid:**
  Count components in $O(\text{polylog} n)$ (or $O(n^{o(1)})$) update time.

[KKM'13, GKKKT'15, CGLNPS'20, NSW'17]



(delete / add edges)

~~Caveat~~

$O(f(n, r))$ dynamic query matroid intersection/union algorithm

+

fast data structures

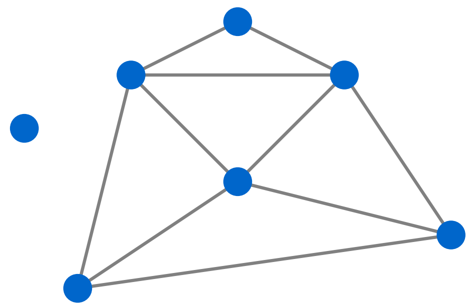
=

$\tilde{O}(f(n, r))$ time algorithm

Data Structures for Dynamic Rank Oracle

- **Colourful/partition matroid:**
  Count colors in $O(1)$ update time.
- **Graphic matroid:**
  Count components in $O(\text{polylog} n)$ (or $O(n^{o(1)})$) update time.

[KKM'13, GKKKT'15, CGLNPS'20, NSW'17]



(delete / add edges)

~~Caveat~~

$O(f(n,r))$ dynamic query matroid intersection/union algorithm

$+$

fast data structures $<$ Need to be worst-case.
Oblivious advesary is okay.

$=$

$\tilde{O}(f(n,r))$ time algorithm

Our Results

- Dynamic-oracle algorithms matching previous query-bounds:
  - $\tilde{O}(n\sqrt{r})$-dynamic-rank-query.
  - $\tilde{O}(nr^{3/4})$-dynamic-indep.-query.
- Improved Matroid Union:
  - $\tilde{O}(n + r\sqrt{r})$-dynamic-rank-query.
  - Concurrently & independently shown† by [Quanrud'23]
  - Compare $O(|E|\sqrt{|V|})$ vs $O(|E| + |V|^{1.5})$ for graph problems.
- First super-linear lower-bounds:
  - $\Omega(n \log n)$ dynamic-rank-queries needed
  - $\Omega(n \log n)$ traditional-indep.-queries needed
  - Improves $\log_2(3)n - o(n) \approx 1.58n$ lower-bound by [Harvey SODA'08]

†In traditional model + specialized to graphic and partition matroids.

Applications

problems	our bounds	state-of-the-art results
(Via k-fold matroid union)		
k-forest[8]	$\tilde{O}(\|E\| + (k\|V\|)^{3/2})$ ✓	$\tilde{O}(k^{3/2}\|V\|\sqrt{\|E\|})$ [GW88]
k-pseudoforest	$\tilde{O}(\|E\| + (k\|V\|)^{3/2})$ ✗	$\|E\|^{1+o(1)}$ [CKL+22]
k-disjoint spanning trees	$\tilde{O}(\|E\| + (k\|V\|)^{3/2})$ ✓	$\tilde{O}(k^{3/2}\|V\|\sqrt{\|E\|})$ [GW88]
arboricity[9]	$\tilde{O}(\|E\|\|V\|)$ ✗	$\tilde{O}(\|E\|^{3/2})$ [Gab95]
tree packing	$\tilde{O}(\|E\|^{3/2})$	$\tilde{O}(\|E\|^{3/2})$ [GW88]
Shannon Switching Game	$\tilde{O}(\|E\| + \|V\|^{3/2})$ ✓	$\tilde{O}(\|V\|\sqrt{\|E\|})$ [GW88]
graph k-irreducibility	$\tilde{O}(\|E\| + (k\|V\|)^{3/2} + k^2\|V\|)$ ✓	$\tilde{O}(k^{3/2}\|V\|\sqrt{\|E\|})$ [GW88]
(Via matroid union)		
(f,p)-mixed forest-pseudoforest	$\tilde{O}_{f,p}(\|E\| + \|V\|\sqrt{\|V\|})$ ✓	$\tilde{O}((f+p)\|V\|\sqrt{f\|E\|})$ [GW88]
(Via matroid intersection)		
bipartite matching (combinatorial[12])	$\tilde{O}(\|E\|\sqrt{\|V\|})$	$O(\|E\|\sqrt{\|V\|})$ [HK73]
bipartite matching (continuous)	$\tilde{O}(\|E\|\sqrt{\|V\|})$ ✗	$\|E\|^{1+o(1)}$ [CKL+22]
graphic matroid intersection	$\tilde{O}(\|E\|\sqrt{\|V\|})$	$\tilde{O}(\|E\|\sqrt{\|V\|})$ [GX89]
simple job scheduling matroid intersection	$\tilde{O}(n\sqrt{r})$	$\tilde{O}(n\sqrt{r})$ [XG94]
convex transversal matroid [EF65] intersection	$\tilde{O}(\|V\|\sqrt{\mu})$	$\tilde{O}(\|V\|\sqrt{\mu})$ [XG94]
linear matroid intersection[10]	$\tilde{O}(n^{2.529}\sqrt{r})$ ✗	$\tilde{O}(nr^{\omega-1})$ [Har09]
colorful spanning tree	$\tilde{O}(\|E\|\sqrt{\|V\|})$	$\tilde{O}(\|E\|\sqrt{\|V\|})$ [GS85]
maximum forest with deadlines	$\tilde{O}(\|E\|\sqrt{\|V\|})$ ✓	(no prior work)

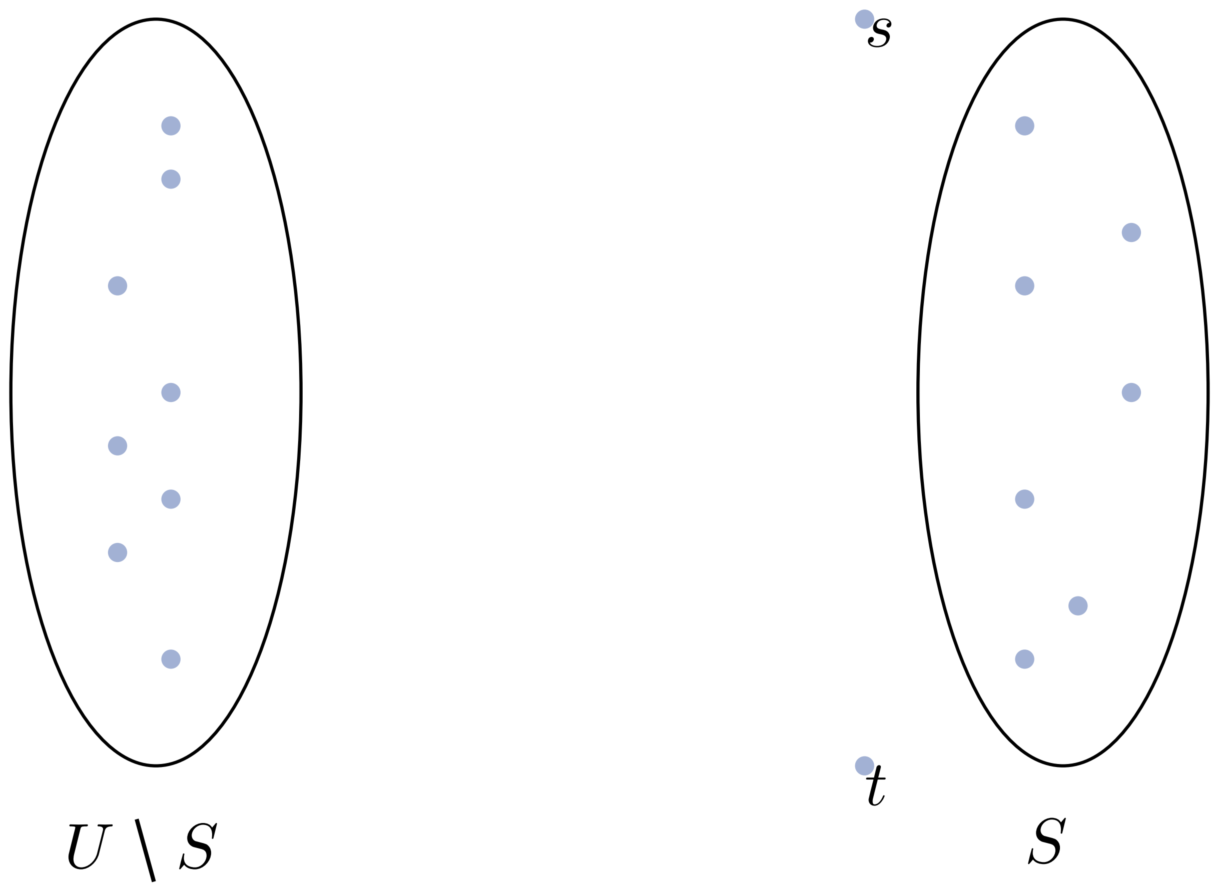
Techniques

Technical part — Overview

1. Exchange Graph & Augmenting Paths
2. Matroid Intersection
   - Matching previous algorithms with Dynamic Oracle
   - Main Idea: “Exchange-Binary-Search-Tree”
3. Matroid Union
   - Improving $\tilde{O}(n\sqrt{r})$ to $\tilde{O}(n + r\sqrt{r})$
   - Main Idea: Sparsifying the Exchange Graph
4. Lower Bound
   - $\Omega(n \log n)$
   - Main Idea: Communication Complexity of Reachability

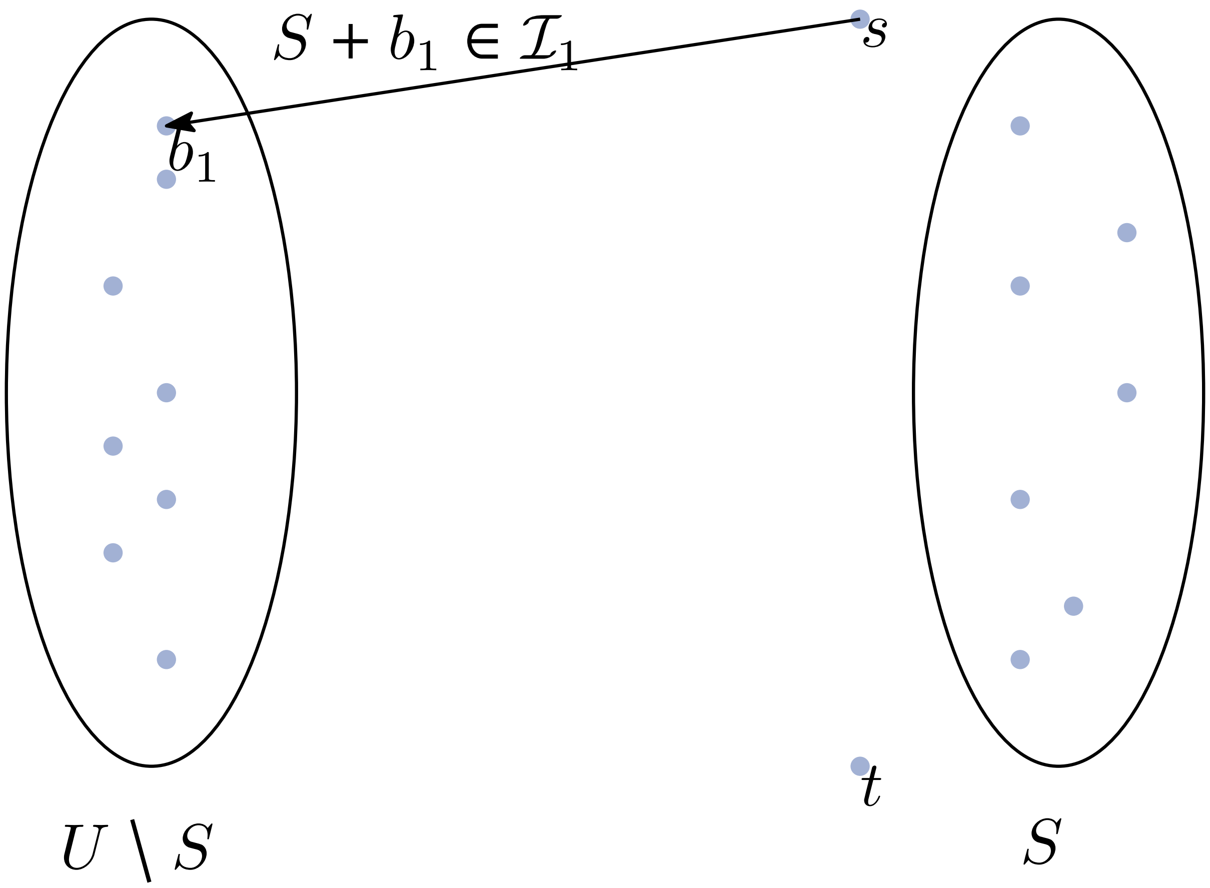
Exchange Graph & Augmenting Paths [Edmonds'60s]

Definition:
The *Exchange Graph* $G(S)$ for a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ looks as follows:



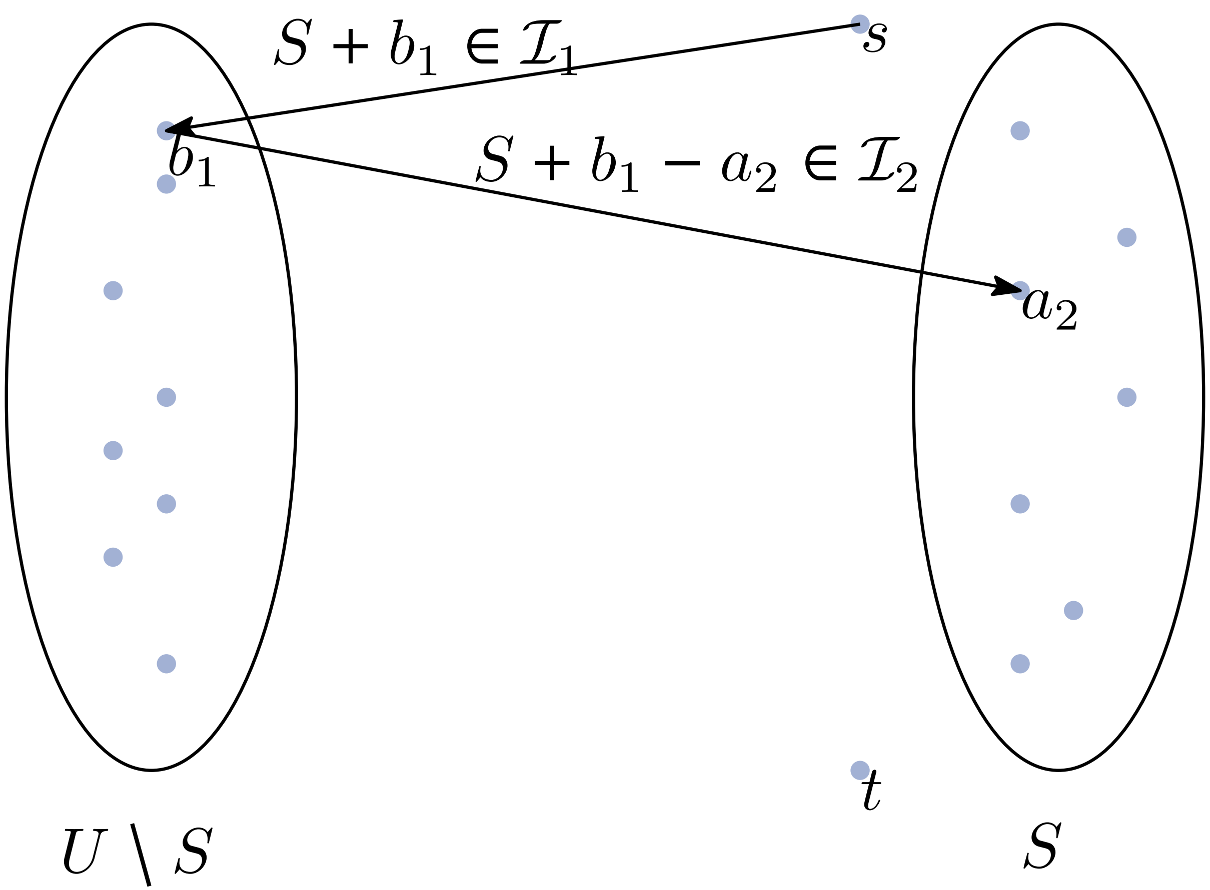
Exchange Graph & Augmenting Paths [Edmonds'60s]

Definition:
The *Exchange Graph* $G(S)$ for a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ looks as follows:



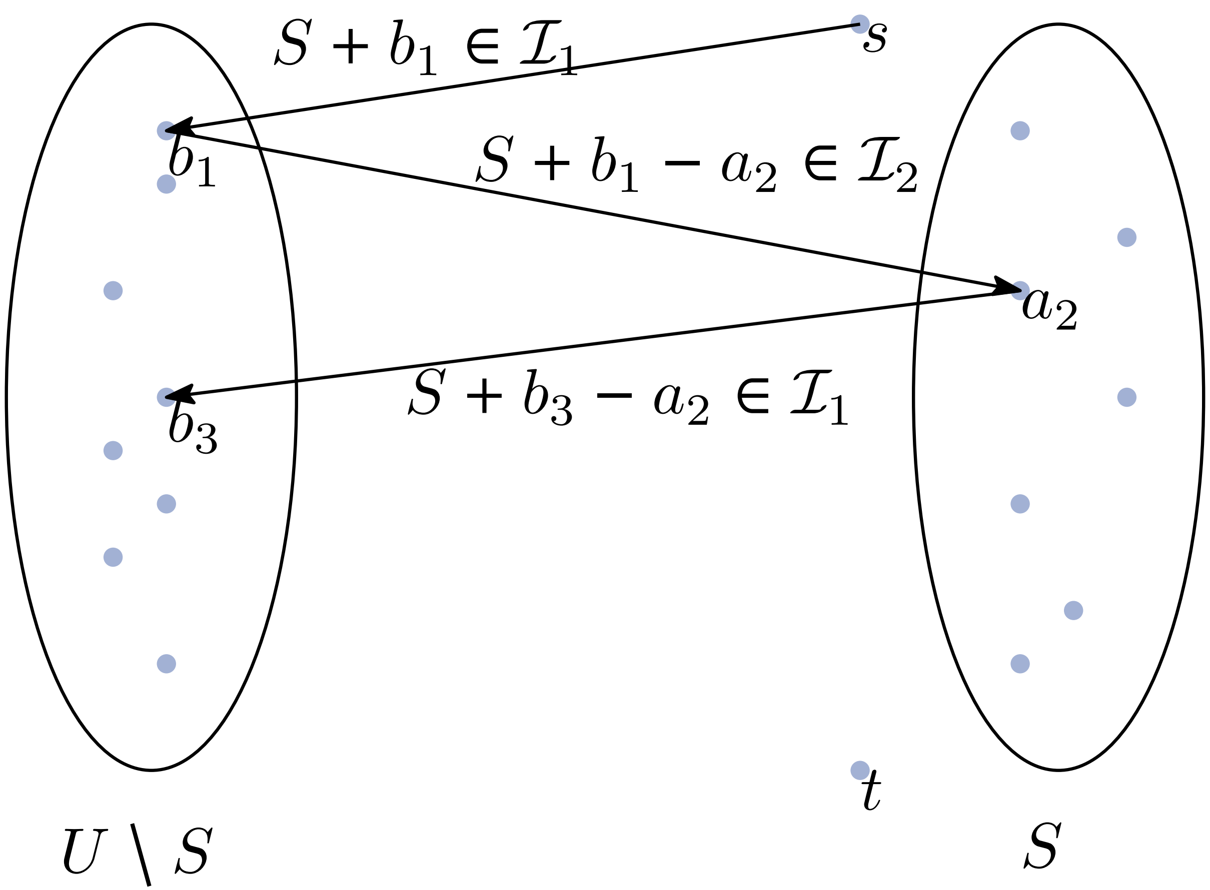
Exchange Graph & Augmenting Paths [Edmonds'60s]

Definition:
The *Exchange Graph* $G(S)$ for a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ looks as follows:



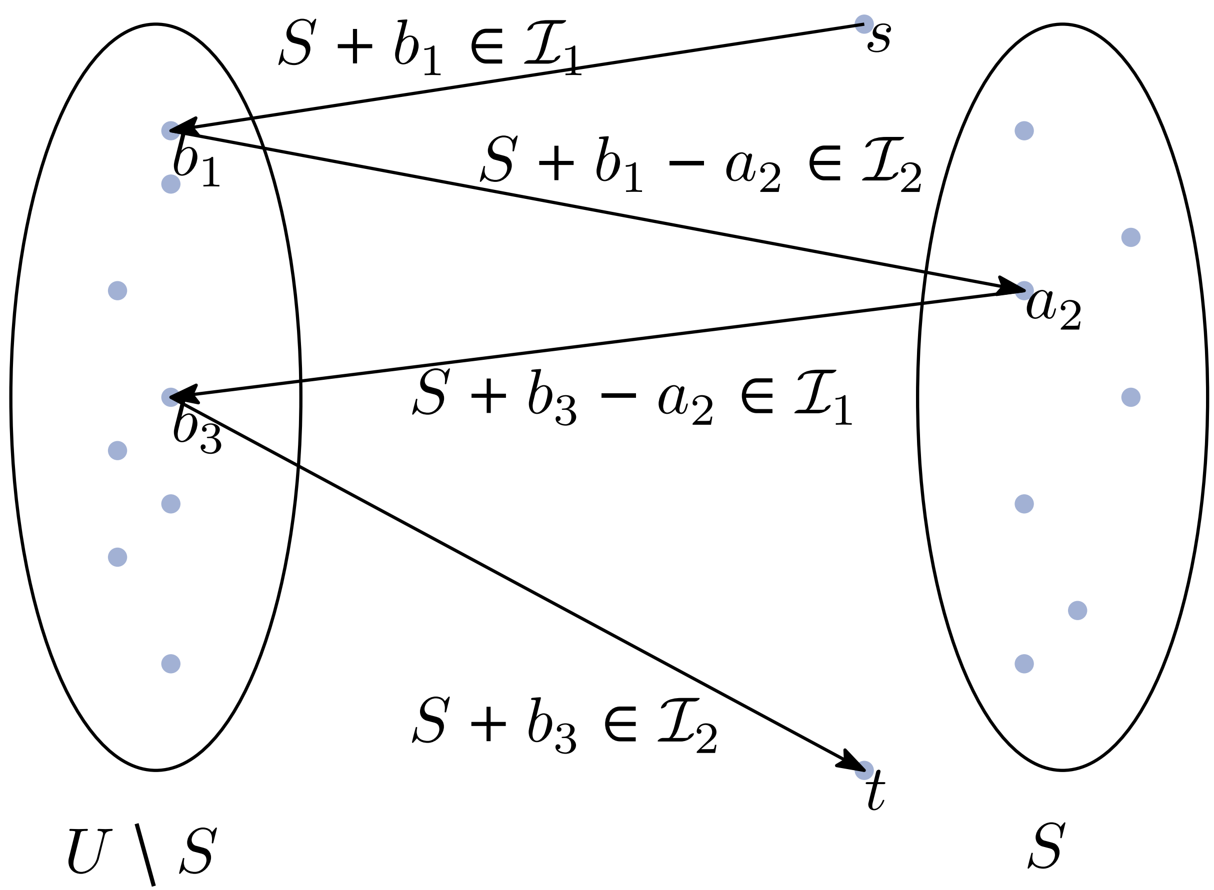
Exchange Graph & Augmenting Paths [Edmonds'60s]

Definition:
The *Exchange Graph* $G(S)$ for a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ looks as follows:

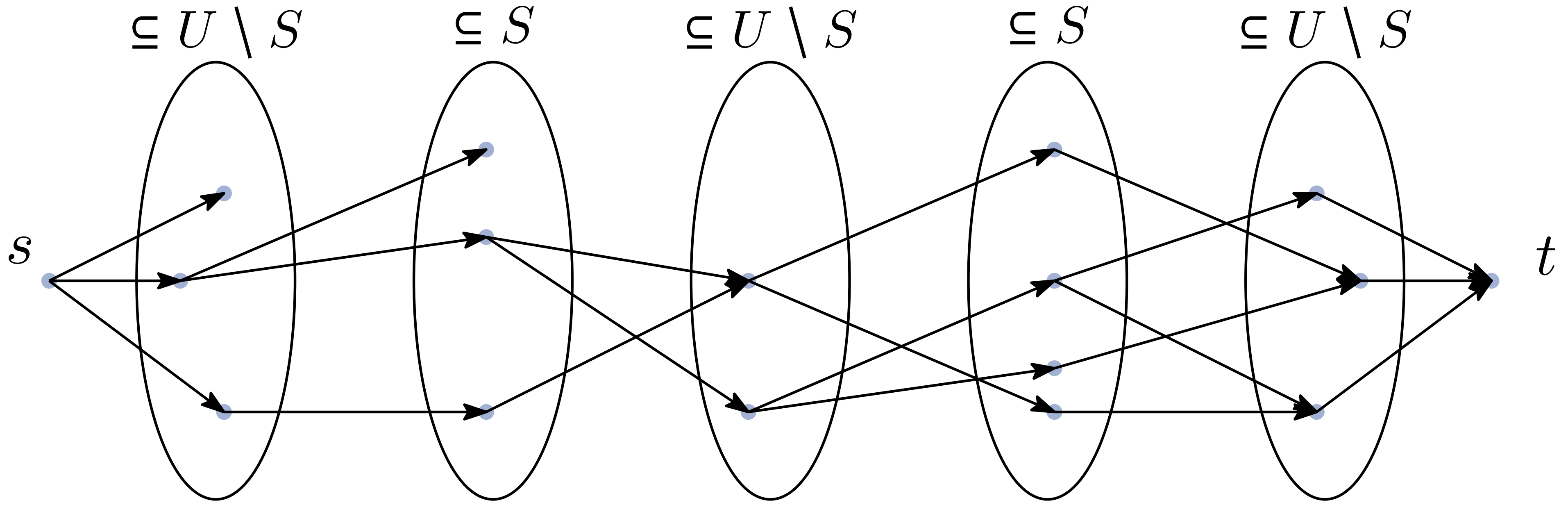


Exchange Graph & Augmenting Paths [Edmonds'60s]

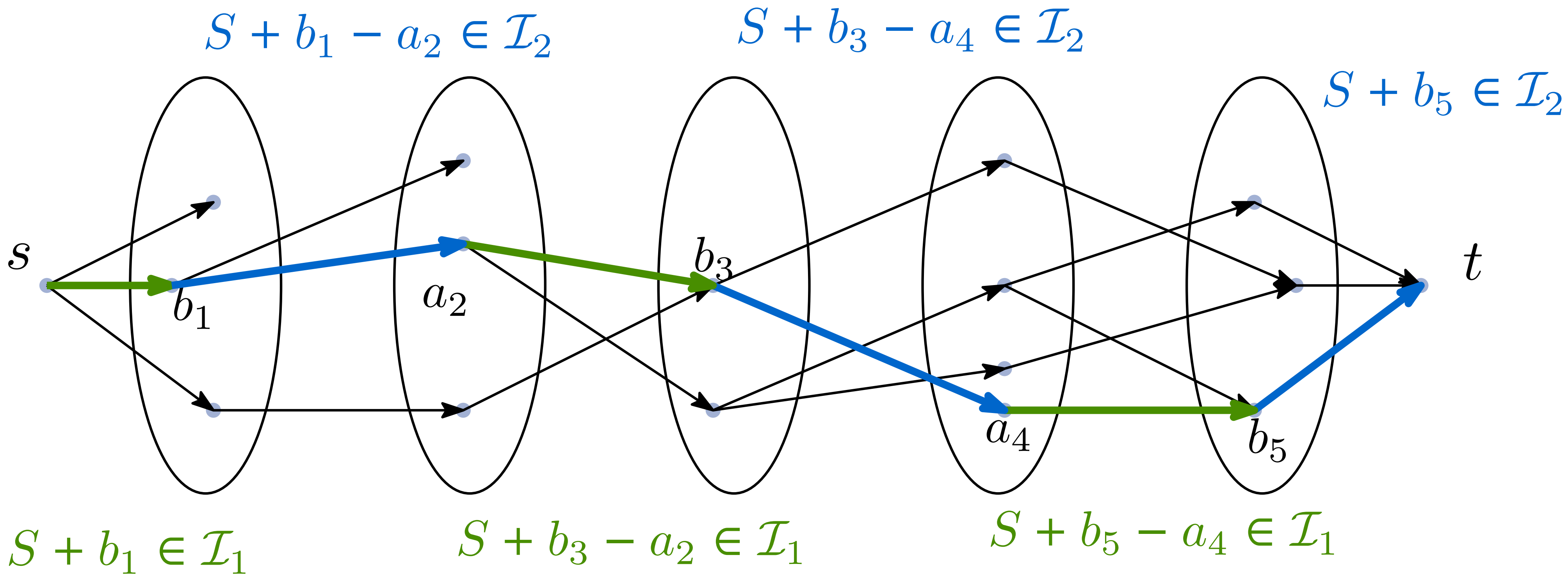
Definition:
The *Exchange Graph* $G(S)$ for a *common independent set* $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ looks as follows:



Exchange Graph & Augmenting Paths [Edmonds'60s]



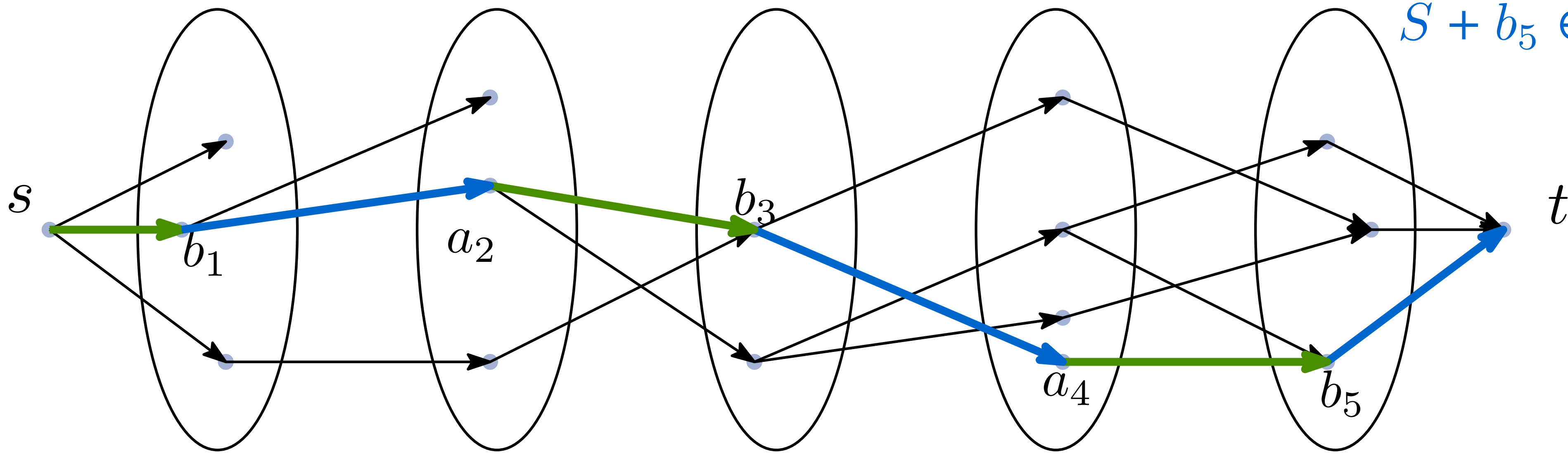
Exchange Graph & Augmenting Paths [Edmonds'60s]



Exchange Graph & Augmenting Paths [Edmonds'60s]

$$\implies S + b_1 - a_2 + b_3 - a_4 + b_5 \in \mathcal{I}_2$$

$S + b_1 - a_2 \in \mathcal{I}_2$ $\quad$ $S + b_3 - a_4 \in \mathcal{I}_2$ $\quad$ $S + b_5 \in \mathcal{I}_2$

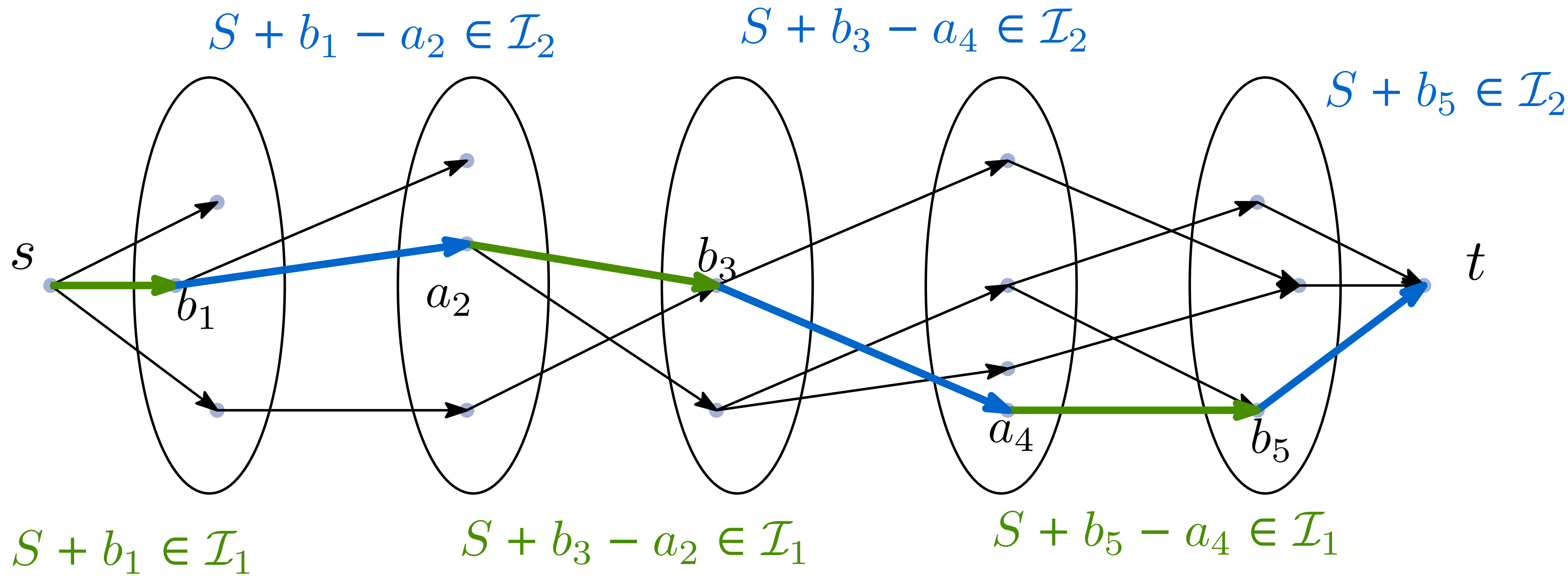


$S + b_1 \in \mathcal{I}_1$ $\quad$ $S + b_3 - a_2 \in \mathcal{I}_1$ $\quad$ $S + b_5 - a_4 \in \mathcal{I}_1$

$$\implies S + b_1 - a_2 + b_3 - a_4 + b_5 \in \mathcal{I}_1$$

Exchange Graph & Augmenting Paths [Edmonds'60s]

$$\implies S + b_1 - a_2 + b_3 - a_4 + b_5 \in \mathcal{I}_2$$

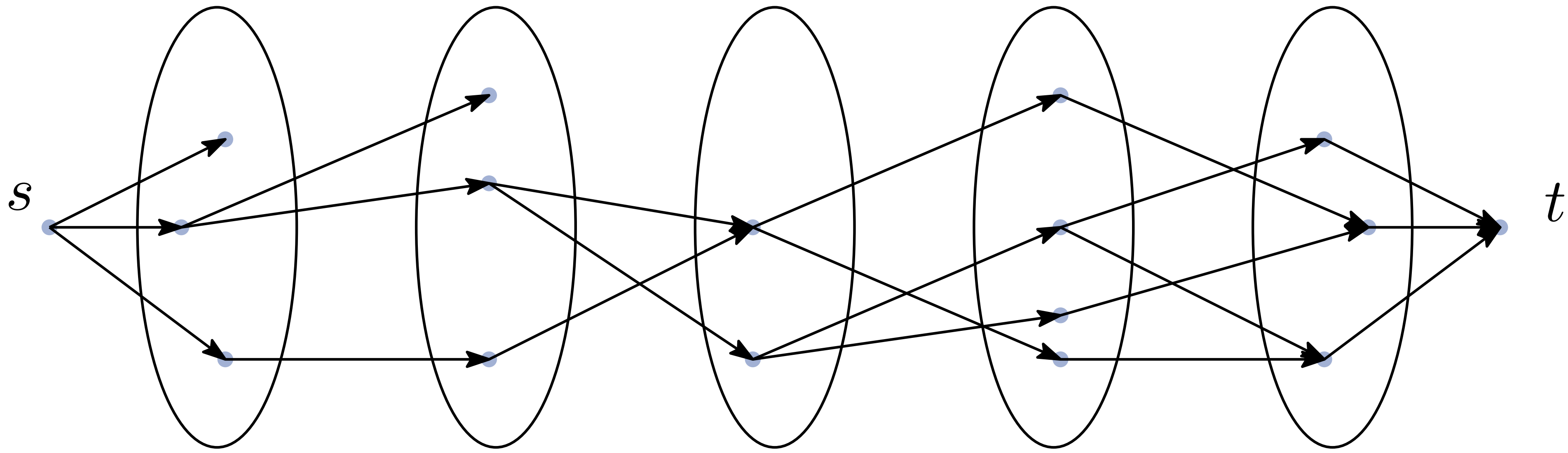


$$\implies S + b_1 - a_2 + b_3 - a_4 + b_5 \in \mathcal{I}_1$$

Common independent set $S' := S + b_1 - a_2 + b_3 - a_4 + b_5$ of size $|S'| = |S| + 1$

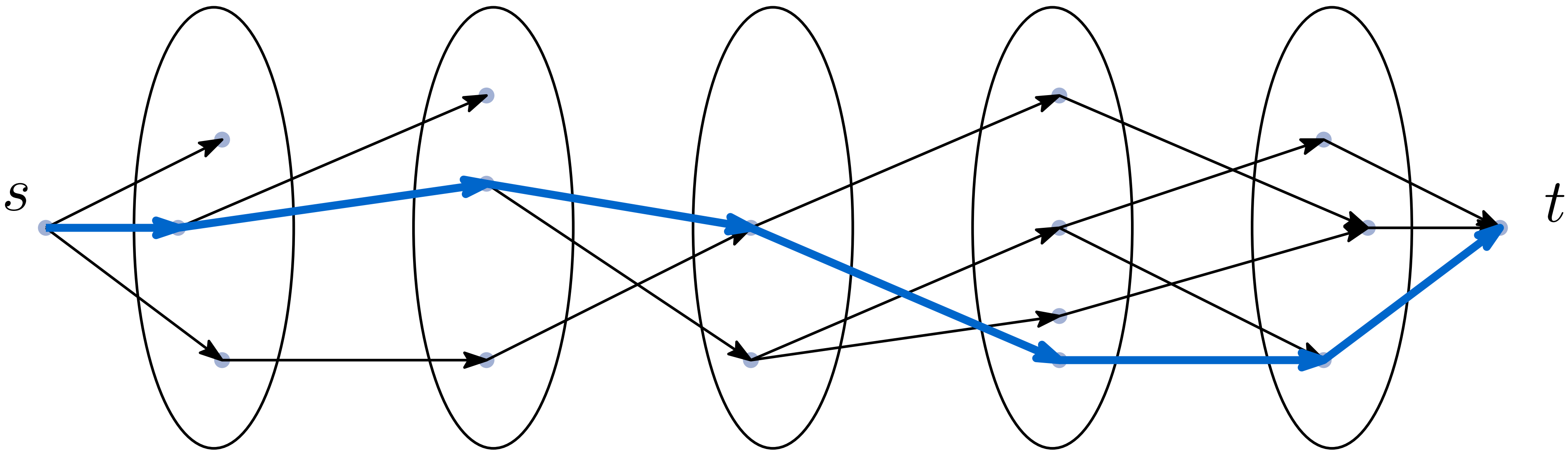
Exchange graph $G(S)$ behaves weirdly...

- $\Theta(nr)$ edges — expensive to compute



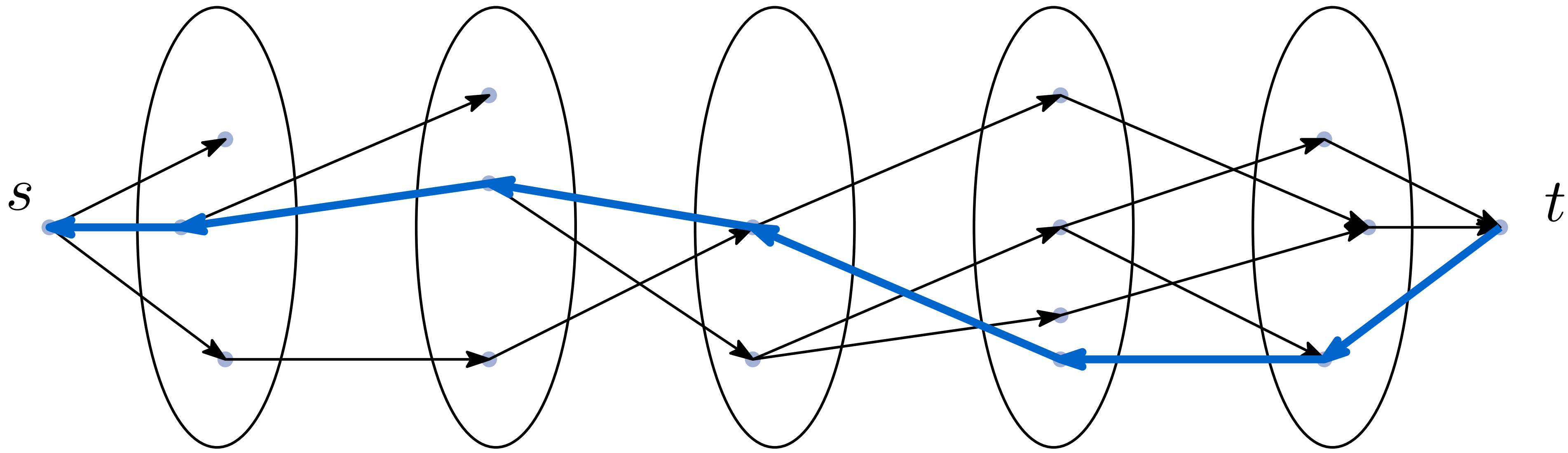
Exchange graph $G(S)$ behaves weirdly...

- $\Theta(nr)$ edges — expensive to compute



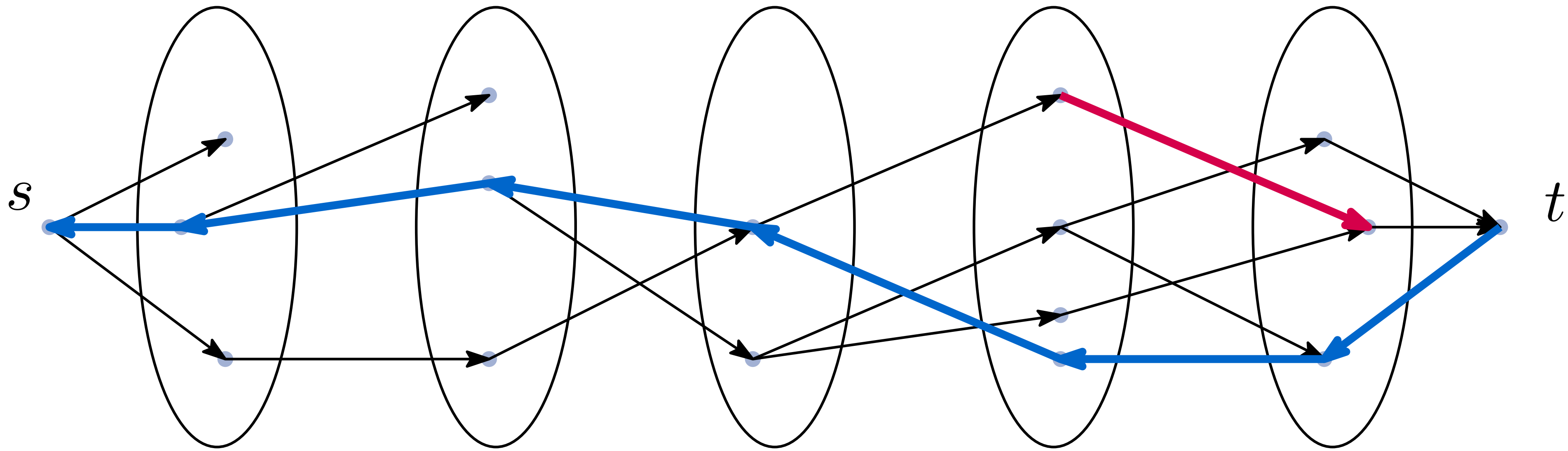
Exchange graph $G(S)$ behaves weirdly...

- $\Theta(nr)$ edges — expensive to compute



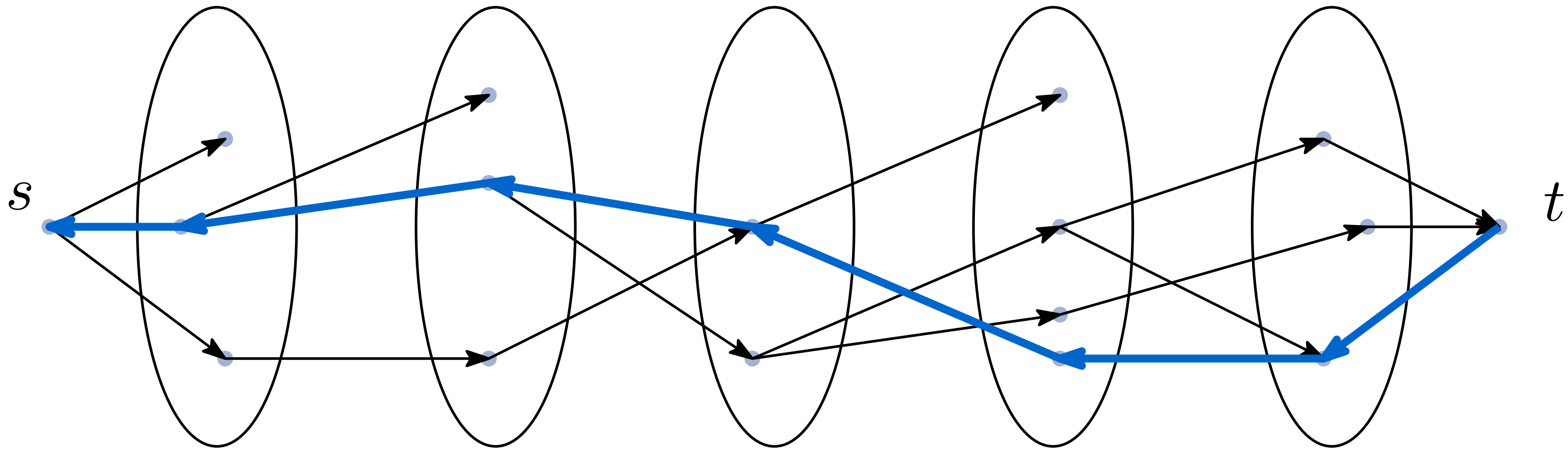
Exchange graph $G(S)$ behaves weirdly...

- $\Theta(nr)$ edges — expensive to compute



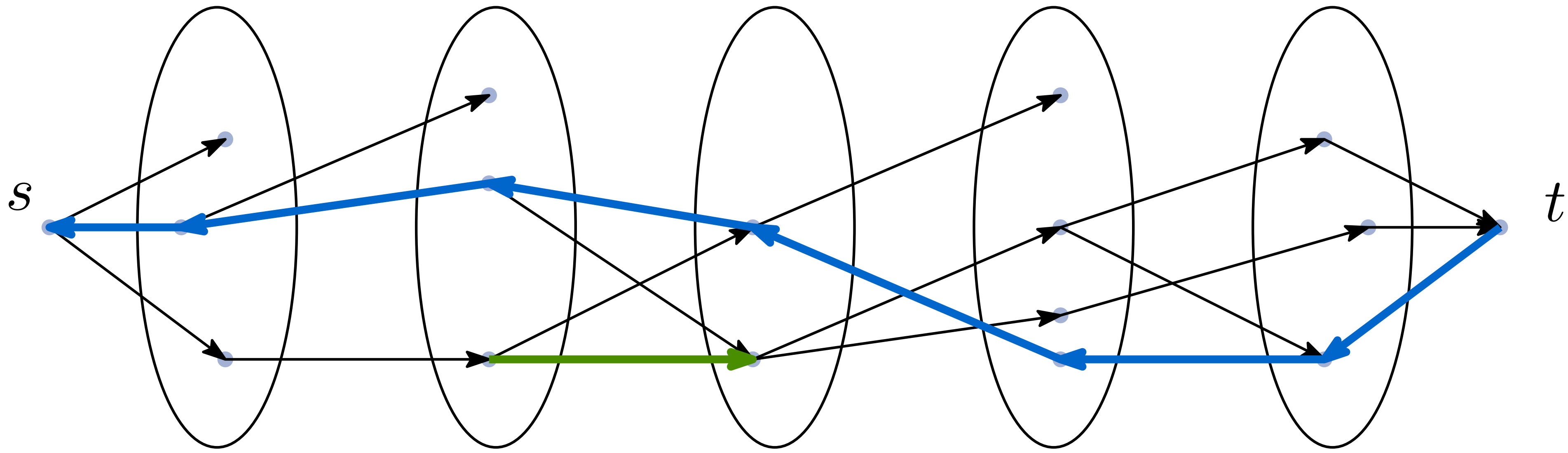
Exchange graph $G(S)$ behaves weirdly. . .

- $\Theta(nr)$ edges — expensive to compute



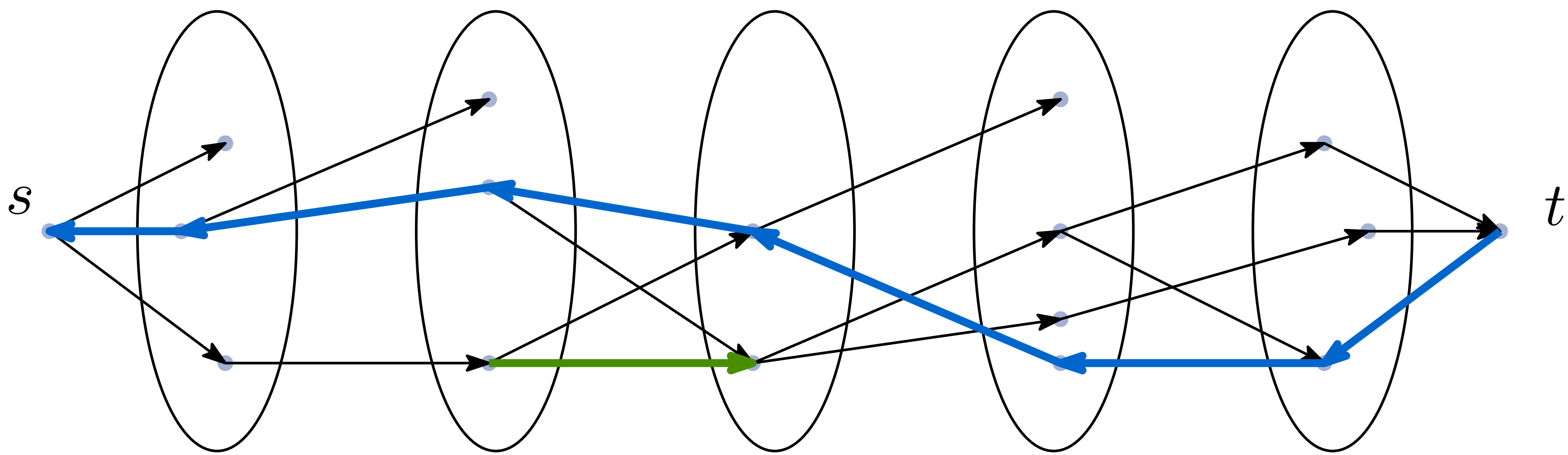
Exchange graph $G(S)$ behaves weirdly...

- $\Theta(nr)$ edges — expensive to compute



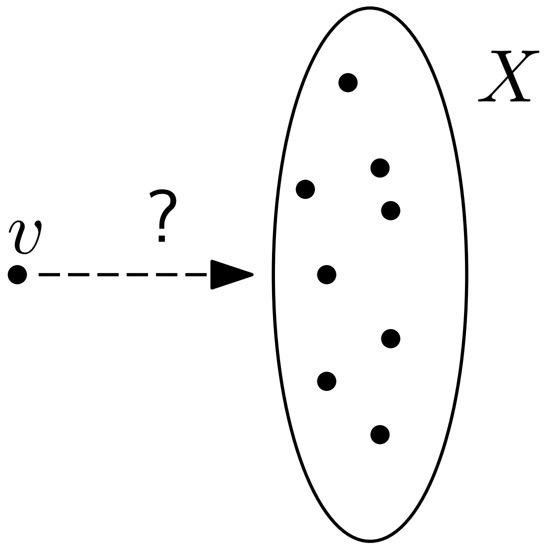
Exchange graph $G(S)$ behaves weirdly. . .

- $\Theta(nr)$ edges — expensive to compute

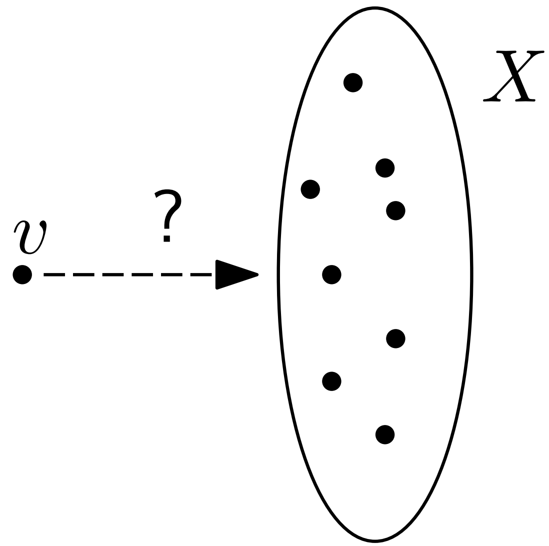


- Disjoint paths not necessarily “compatible”
- Need recompute to handle inserted and deleted edges.

Graph Exploration — Exchange Pairs



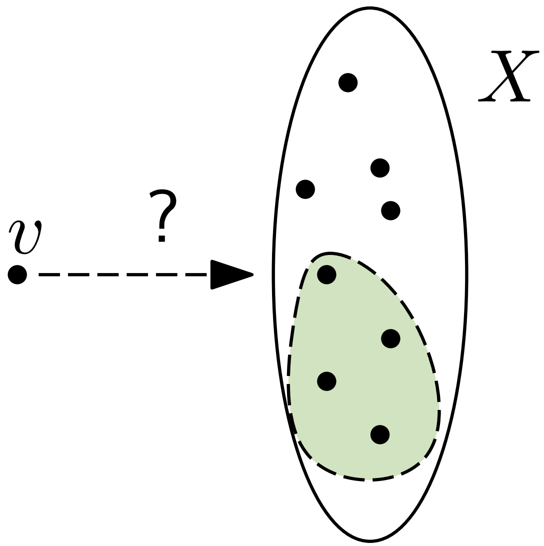
Graph Exploration — Exchange Pairs



“$\mathrm{rk}_1(S + v - X) = |S + v - X|$?”

“$\mathrm{rk}_2(S - v + X) = |S|$?”

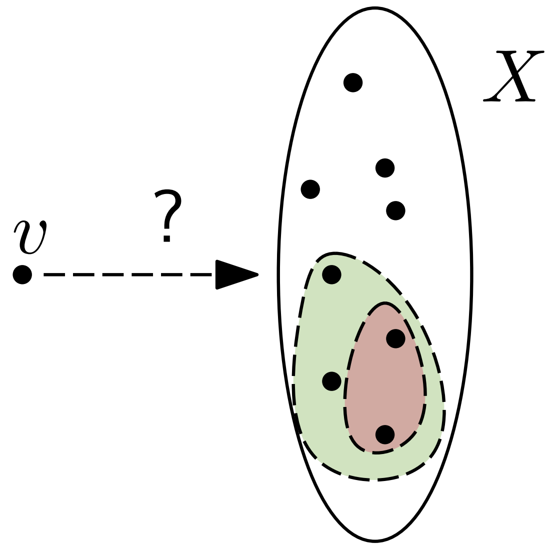
Graph Exploration — Exchange Pairs



“$\mathrm{rk}_1(S + v - X) = |S + v - X|$?”

“$\mathrm{rk}_2(S - v + X) = |S|$?”

Graph Exploration — Exchange Pairs



"$\mathrm{rk}_1(S + v - X) = |S + v - X|$?"

"$\mathrm{rk}_2(S - v + X) = |S|$?"

Binary-Search! [CLSSW, Nguyễn]

Technical part — Overview

1. Exchange Graph & Augmenting Paths

2. Matroid Intersection
   - Matching previous algorithms with Dynamic Oracle
   - Main Idea: "Exchange-Binary-Search-Tree"

3. Matroid Union
   - Improving $\tilde{O}(n\sqrt{r})$ to $\tilde{O}(n + r\sqrt{r})$
   - Main Idea: Sparsifying the Exchange Graph

4. Lower Bound
   - $\Omega(n \log n)$
   - Main Idea: Communication Complexity of Reachability

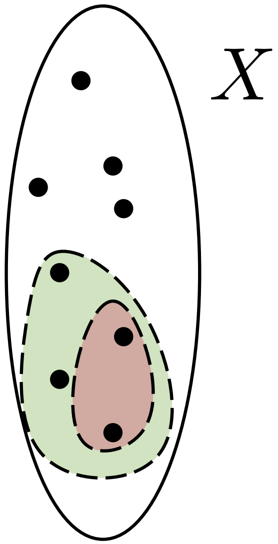
Matroid Intersection — Main Idea

"$\mathrm{rk}_2(S - v + X) = |S|$?"

Matroid Intersection — Main Idea

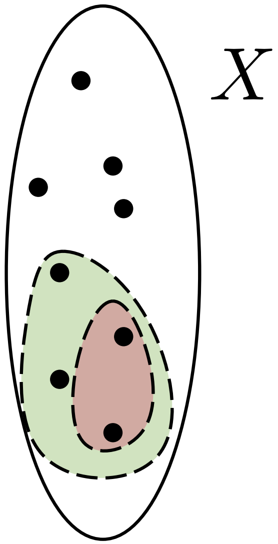
"$\mathrm{rk}_2(S - v + X) = |S|$?"

Challenge: Query sets far apart in binary search.



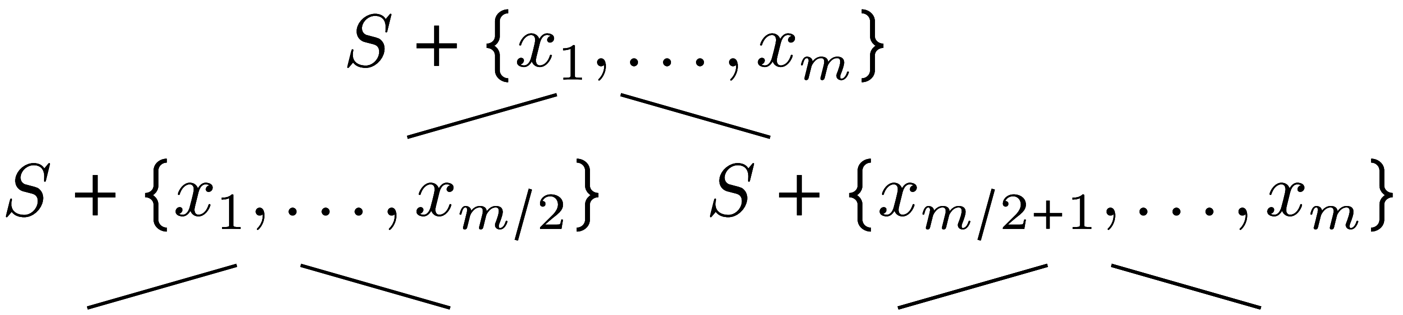
Matroid Intersection — Main Idea

"$\mathrm{rk}_2(S - v + X) = |S|$?"

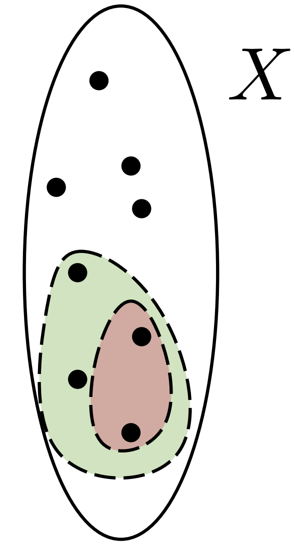


Challenge: Query sets far apart in binary search.

Solution: Prebuild sets:



Matroid Intersection — Main Idea



"$\mathrm{rk}_2(S - v + X) = |S|$?"

Challenge: Query sets far apart in binary search.

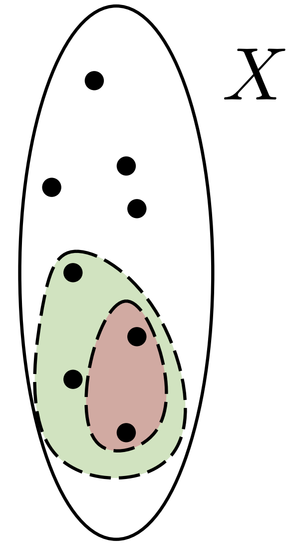
Solution: Prebuild sets:

$$S + \{x_1, \ldots, x_m\}$$

$$S + \{x_1, \ldots, x_{m/2}\} \qquad S + \{x_{m/2+1}, \ldots, x_m\}$$

Challenge: S changes when augmenting path found.

Matroid Intersection — Main Idea



"$\mathrm{rk}_2(S - v + X) = |S|$?"

Challenge: Query sets far apart in binary search.

Solution: Prebuild sets:

$$S + \{x_1, \ldots, x_m\}$$

$$S + \{x_1, \ldots, x_{m/2}\} \qquad S + \{x_{m/2+1}, \ldots, x_m\}$$

Challenge: S changes when augmenting path found.

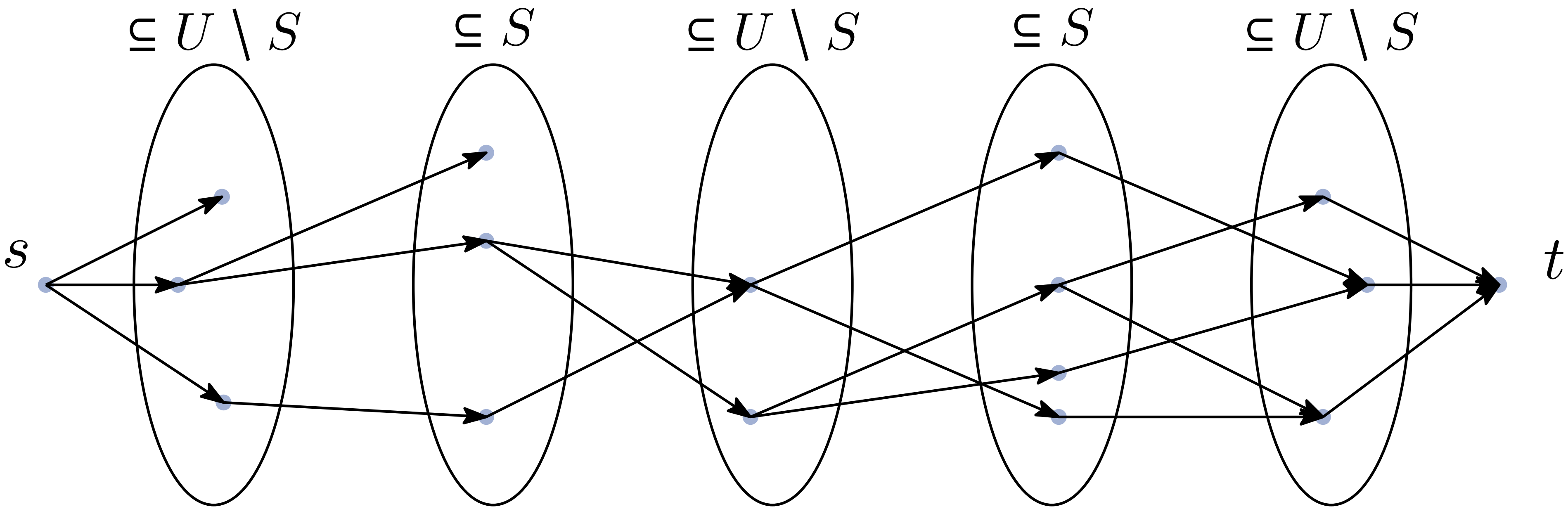
Solution:
Lazily rebuild in batched + "Augmenting Sets" Lemma [CLSSW]

Technical part — Overview

1. Exchange Graph & Augmenting Paths
2. Matroid Intersection
   - Matching previous algorithms with Dynamic Oracle
   - Main Idea: "Exchange-Binary-Search-Tree"
3. Matroid Union
   - Improving $\tilde{O}(n\sqrt{r})$ to $\tilde{O}(n + r\sqrt{r})$
   - Main Idea: Sparsifying the Exchange Graph
4. Lower Bound
   - $\Omega(n \log n)$
   - Main Idea: Communication Complexity of Reachability

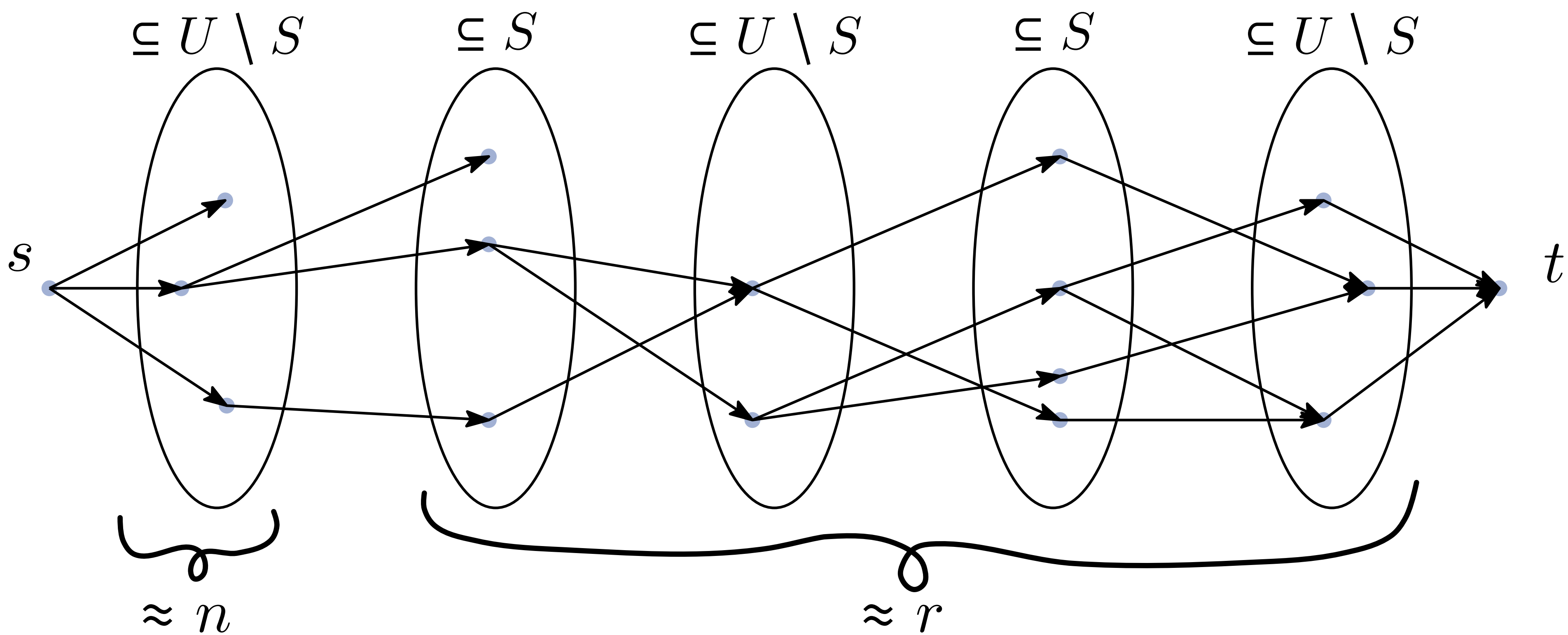
Matroid Union — Main Idea

Going from $O(n\sqrt{r})$ to $O(n + r\sqrt{r})$.



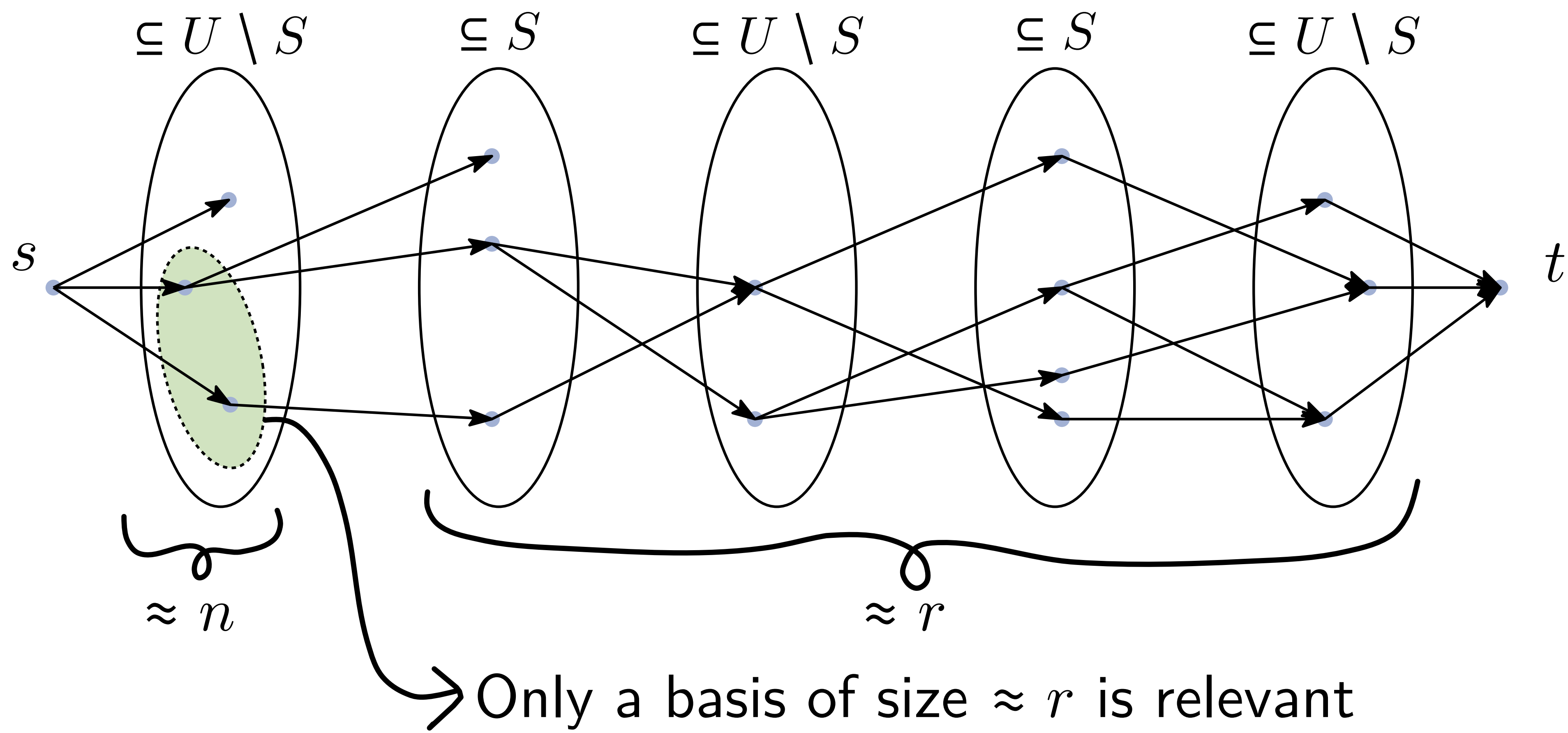
Matroid Union — Main Idea

Going from $O(n\sqrt{r})$ to $O(n + r\sqrt{r})$.



Matroid Union — Main Idea

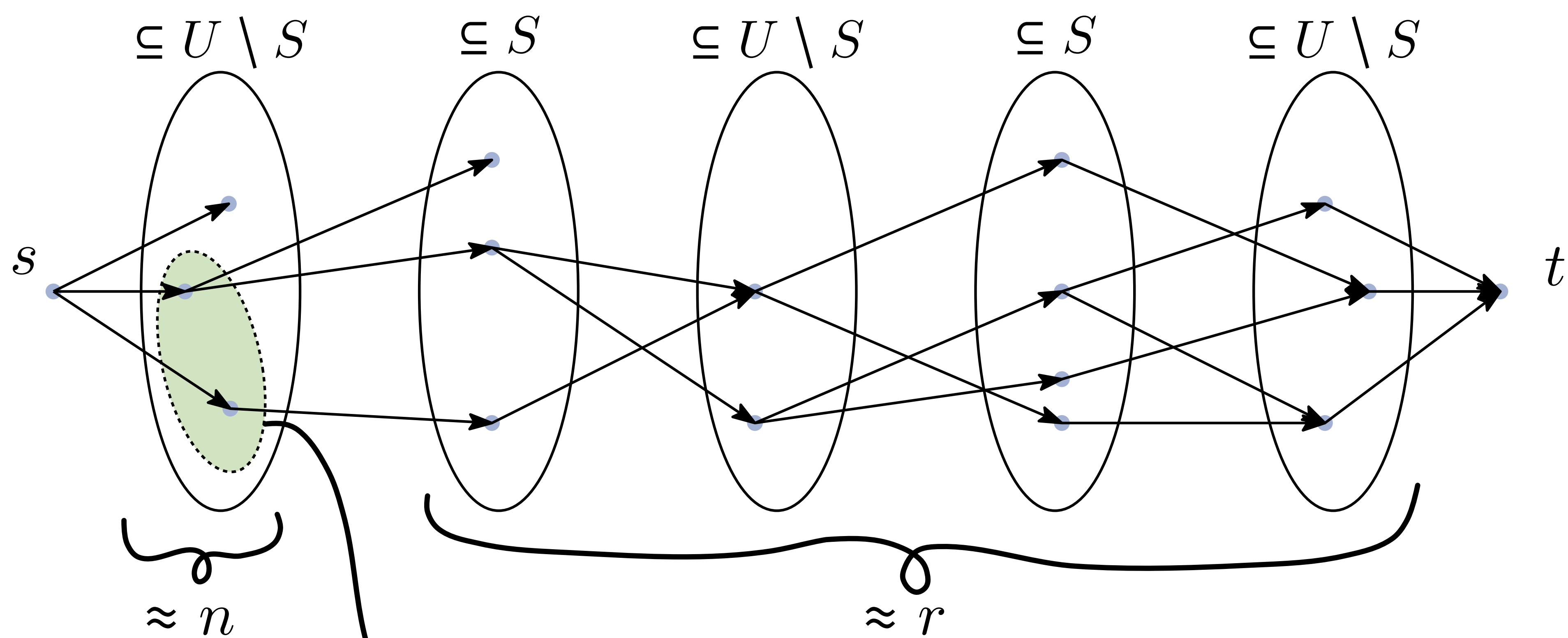
Going from $O(n\sqrt{r})$ to $O(n + r\sqrt{r})$.



Only a basis of size $\approx r$ is relevant

Matroid Union — Main Idea

Going from $O(n\sqrt{r})$ to $O(n + r\sqrt{r})$.



Only a basis of size $\approx r$ is relevant

Maintain dynamically when S changes

Binary tree of sqrt-decomposition

similar to early dynamic MST [Fre85 , EGIN97]

Technical part — Overview

1. Exchange Graph & Augmenting Paths

2. Matroid Intersection
   - Matching previous algorithms with Dynamic Oracle
   - Main Idea: "Exchange-Binary-Search-Tree"

3. Matroid Union
   - Improving $\tilde{O}(n\sqrt{r})$ to $\tilde{O}(n + r\sqrt{r})$
   - Main Idea: Sparsifying the Exchange Graph

4. Lower Bound
   - $\Omega(n \log n)$
   - Main Idea: Communication Complexity of Reachability

Lower Bound — Main Idea

Communication game

Alice

Bob

$\mathcal{M}_1 = (U, \mathcal{I}_1)$

$\mathcal{M}_2 = (U, \mathcal{I}_2)$

How many bits of communication necessary?

Lower Bound — Main Idea

Communication game

Alice $\mathcal{M}_1 = (U, \mathcal{I}_1)$

Bob $\mathcal{M}_2 = (U, \mathcal{I}_2)$

How many bits of communication necessary?

$$\leq \begin{cases} \text{indep. queries} \\ \text{rank queries } / \log(n) \\ \text{dynamic rank queries} \end{cases}$$

Lower Bound — Main Idea

Communication game

Alice $\qquad\qquad\qquad$ Bob

$\mathcal{M}_1 = (U, \mathcal{I}_1)$ $\qquad\qquad\qquad$ $\mathcal{M}_2 = (U, \mathcal{I}_2)$

How many bits of communication necessary?

$$\leq \begin{cases} \text{indep. queries} \\ \text{rank queries } / \log(n) \\ \text{dynamic rank queries} \end{cases}$$

Carefully choose matroids (gammoids) to model **Graph Reachability**

$\Omega(n \log n)$ bit lower-bound[†] [Hajnal-Maass-Turán STOC'88]

† (unconditionally for deterministic, and conjectured to hold for randomized algorithms)

Open Problems

- k-Disjoint Spanning Trees in $\tilde{O}(|E|)$ time?

Open Problems

- k-Disjoint Spanning Trees in $\tilde{O}(|E|)$ time?
- Matroid Intersection / Union in $\tilde{O}(n)$ time?
  - Dynamic or traditional query model; or communication.
  - Or any $n^{1+\Omega(1)}$ lower-bounds?

Open Problems

- k-Disjoint Spanning Trees in $\tilde{O}(|E|)$ time?
- Matroid Intersection / Union in $\tilde{O}(n)$ time?
  - Dynamic or traditional query model; or communication.
  - Or any $n^{1+\Omega(1)}$ lower-bounds?
- What about *weighted* matroid intersection?
  - Currently slower than unweighted.
  - Match using the dynamic oracle model?

Open Problems

- k-Disjoint Spanning Trees in $\tilde{O}(|E|)$ time?
- Matroid Intersection / Union in $\tilde{O}(n)$ time?
  - Dynamic or traditional query model; or communication.
  - Or any $n^{1+\Omega(1)}$ lower-bounds?
- What about *weighted* matroid intersection?
  - Currently slower than unweighted.
  - Match using the dynamic oracle model?
- Other problems where a "Dynamic Oracle" model is relevant?
  - Submodular function minimization/maximization?
  - Cut-queries in graphs?
  - ...

Open Problems

- k-Disjoint Spanning Trees in $\tilde{O}(|E|)$ time?
- Matroid Intersection / Union in $\tilde{O}(n)$ time?
  - Dynamic or traditional query model; or communication.
  - Or any $n^{1+\Omega(1)}$ lower-bounds?
- What about *weighted* matroid intersection?
  - Currently slower than unweighted.
  - Match using the dynamic oracle model?
- Other problems where a "Dynamic Oracle" model is relevant?
  - Submodular function minimization/maximization?
  - Cut-queries in graphs?
  - ...

Thanks!