Fast Algorithms via Dynamic-Oracle Matroids

Ta-Wei Tu (MPI-INF ⟶ Stanford)

Joint work with

Joakim Blikstad
(KTH)

Sagnik Mukhopadhyay
(University of Sheffield)

Danupon Nanongkai
(MPI-INF & KTH)

Summary

Summary

- *Unified* way of solving problems *fast* via *efficient* reduction

Summary

- *Unified* way of solving problems *fast* via *efficient* reduction

$\tilde{O}(1) \times$ "max-flow".

Bipartite matching, Gomory-Hu tree, edge connectivity, vertex connectivity, …

Summary

- *Unified* way of solving problems *fast* via *efficient* reduction

$\widetilde{O}(1) \times$ "max-flow".

Bipartite matching, Gomory-Hu tree, edge connectivity, vertex connectivity, …

$\widetilde{O}(1) \times$ "matroid intersection".

Bipartite matching, k-disjoint spanning tree, colorful spanning tree, graphic MI, …

Summary

- *Unified* way of solving problems *fast* via *efficient* reduction

$\widetilde{O}(1) \times$ "max-flow".

Bipartite matching, Gomory-Hu tree, edge connectivity, vertex connectivity, …

$\widetilde{O}(1) \times$ "matroid intersection".

Bipartite matching, k-disjoint spanning tree, colorful spanning tree, graphic MI, …

- k-disjoint spanning tree

Summary

- *Unified* way of solving problems *fast* via *efficient* reduction

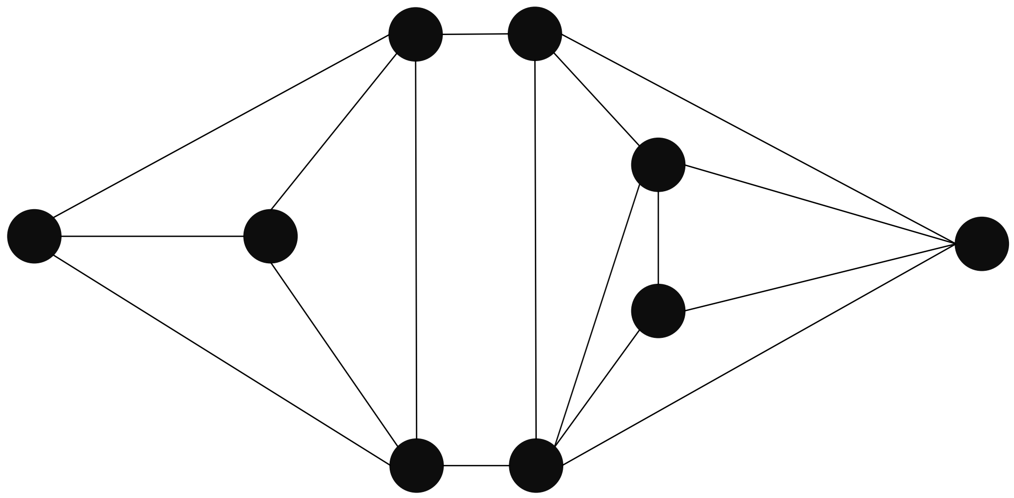
$\tilde{O}(1) \times$ "max-flow".

Bipartite matching, Gomory-Hu tree, edge connectivity, vertex connectivity, …

$\tilde{O}(1) \times$ "matroid intersection".

Bipartite matching, k-disjoint spanning tree, colorful spanning tree, graphic MI, …

- k-disjoint spanning tree



$k = 1$

Summary

- *Unified* way of solving problems *fast* via *efficient* reduction

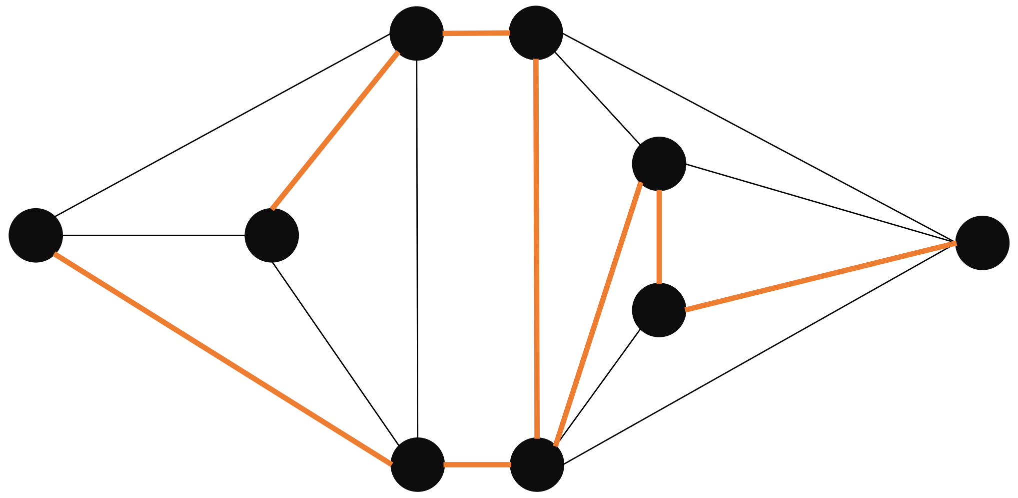
$\widetilde{O}(1) \times$ "max-flow".

Bipartite matching, Gomory-Hu tree, edge connectivity, vertex connectivity, …

$\widetilde{O}(1) \times$ "matroid intersection".

Bipartite matching, k-disjoint spanning tree, colorful spanning tree, graphic MI, …

- k-disjoint spanning tree



$k = 1$

Summary

- *Unified* way of solving problems *fast* via *efficient* reduction

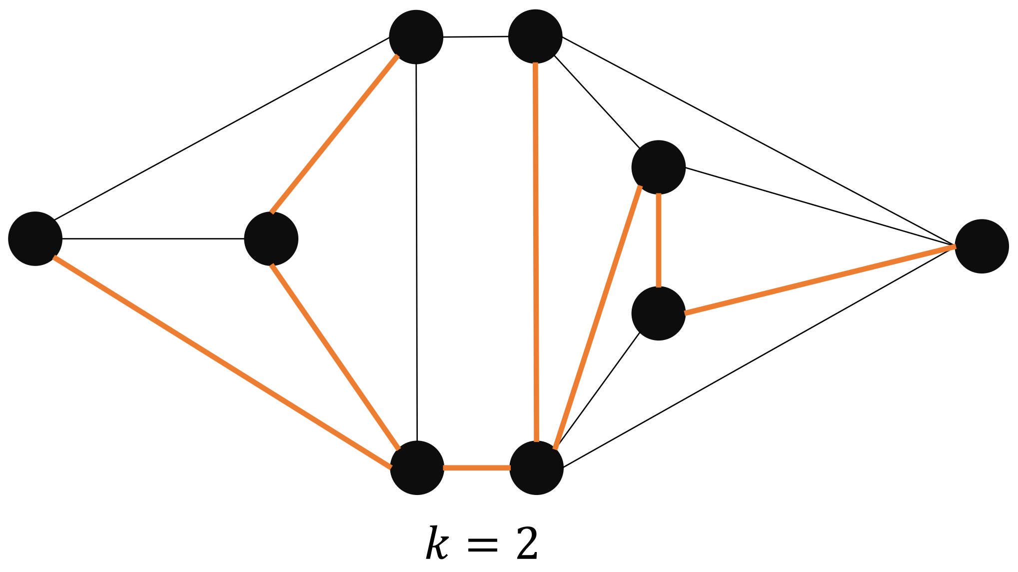
$\tilde{O}(1) \times$ "max-flow".

Bipartite matching, Gomory-Hu tree, edge connectivity, vertex connectivity, …

$\tilde{O}(1) \times$ "matroid intersection".

Bipartite matching, k-disjoint spanning tree, colorful spanning tree, graphic MI, …

- k-disjoint spanning tree



Summary

- *Unified* way of solving problems *fast* via *efficient* reduction

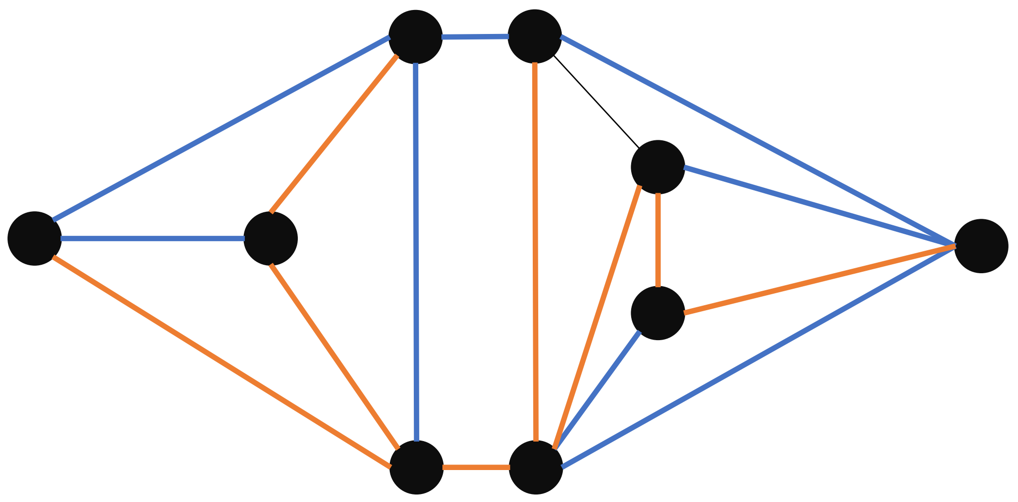
$\tilde{O}(1) \times$ "max-flow".

Bipartite matching, Gomory-Hu tree, edge connectivity, vertex connectivity, …

$\tilde{O}(1) \times$ "matroid intersection".

Bipartite matching, k-disjoint spanning tree, colorful spanning tree, graphic MI, …

- k-disjoint spanning tree



$k = 2$

Summary

- *Unified* way of solving problems *fast* via *efficient* reduction

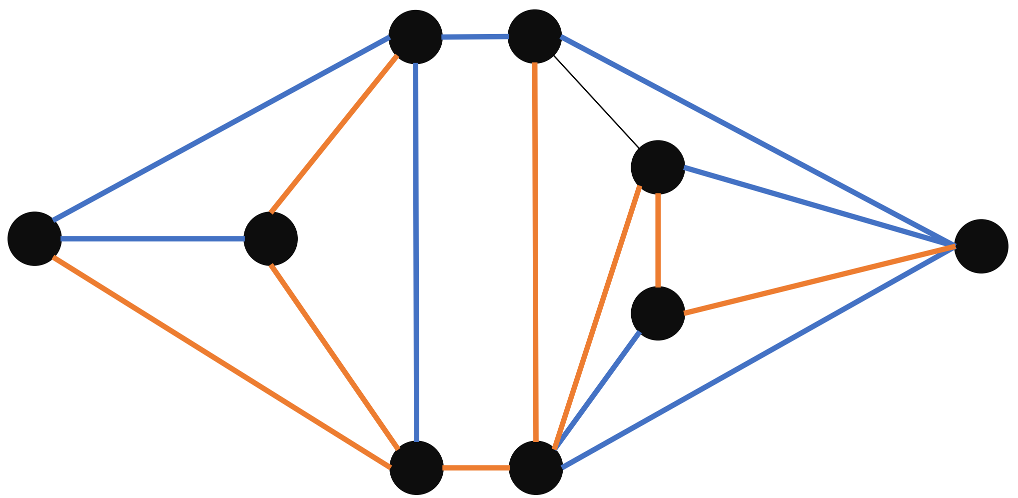
$\tilde{O}(1) \times$ "max-flow".

Bipartite matching, Gomory-Hu tree, edge connectivity, vertex connectivity, …

$\tilde{O}(1) \times$ "matroid intersection".

Bipartite matching, k-disjoint spanning tree, colorful spanning tree, graphic MI, …

- k-disjoint spanning tree



$k = 2$

- $\tilde{O}(k^{3/2}|V|\sqrt{|E|})$ by [GW'92]
- $\tilde{O}\left(|E| + k^{3/2}|V|\sqrt{|V|}\right)$ [this paper, Quanrud'23]

Summary

- *Unified* way of solving problems *fast* via *efficient* reduction

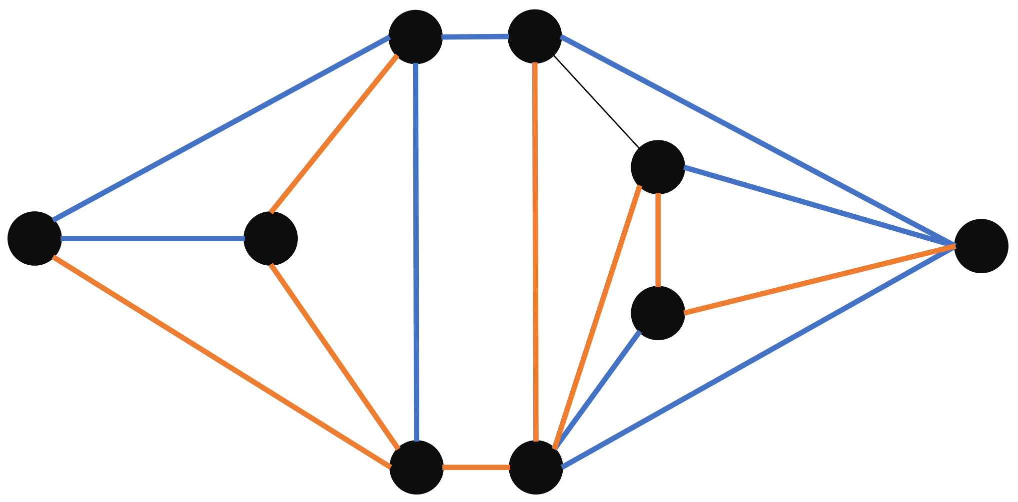
$\widetilde{O}(1) \times$ "max-flow".

Bipartite matching, Gomory-Hu tree, edge connectivity, vertex connectivity, …

$\widetilde{O}(1) \times$ "matroid intersection".

Bipartite matching, k-disjoint spanning tree, colorful spanning tree, graphic MI, …

- k-disjoint spanning tree



$k = 2$

- $\tilde{O}(k^{3/2}|V|\sqrt{|E|})$ by [GW'92]
- $\tilde{O}\left(|E| + k^{3/2}|V|\sqrt{|V|}\right)$ [this paper, Quanrud'23]

$\tilde{O}\left(k^{O(1)}(|E| + |V|\sqrt{|V|})\right)$ by e.g. [Karger'98, FNSYY'20] (implicitly)

Background

Matroid: $\mathcal{M} = (U, \mathcal{I})$ where $\mathcal{I} \subseteq 2^U$ satisfies certain properties

- U: ground set of n elements, $\mathcal{I}$: notion of independence

Background

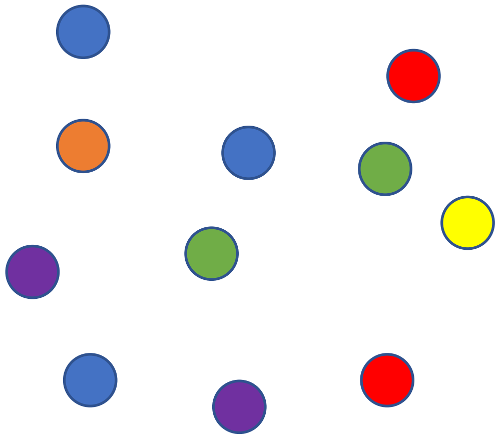
Matroid: $\mathcal{M} = (U, \mathcal{I})$ where $\mathcal{I} \subseteq 2^U$ satisfies certain properties

- U: ground set of n elements, $\mathcal{I}$: notion of independence
- $\text{rank}(S) = \max\{|A|: A \subseteq S, A \in \mathcal{I}\}$
- Basis = maximal/maximum independent set

Background

Matroid: $\mathcal{M} = (U, \mathcal{I})$ where $\mathcal{I} \subseteq 2^U$ satisfies certain properties

- U: ground set of n elements, $\mathcal{I}$: notion of independence
- $\text{rank}(S) = \max\{|A|: A \subseteq S, A \in \mathcal{I}\}$
- Basis = maximal/maximum independent set

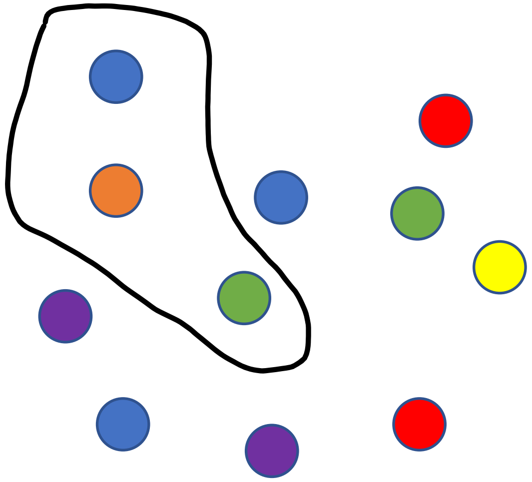


$\mathcal{M}$ = no duplicate colors
$\text{rank}(S)$ = #distinct colors

Background

Matroid: $\mathcal{M} = (U, \mathcal{I})$ where $\mathcal{I} \subseteq 2^U$ satisfies certain properties

- U: ground set of n elements, $\mathcal{I}$: notion of independence
- $\text{rank}(S) = \max\{|A|:\ A \subseteq S, A \in \mathcal{I}\}$
- Basis = maximal/maximum independent set

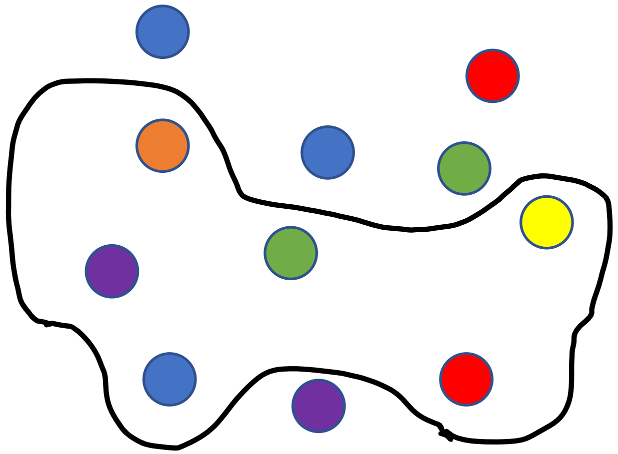


$\mathcal{M}$ = no duplicate colors
rank(S) = #distinct colors

Background

Matroid: $\mathcal{M} = (U, \mathcal{I})$ where $\mathcal{I} \subseteq 2^U$ satisfies certain properties

- U: ground set of n elements, $\mathcal{I}$: notion of independence
- $\text{rank}(S) = \max\{|A|: A \subseteq S, A \in \mathcal{I}\}$
- Basis = maximal/maximum independent set

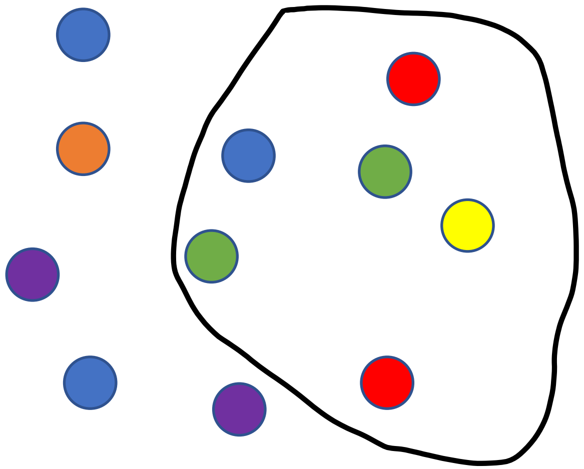


$\mathcal{M}$ = no duplicate colors
$\text{rank}(S)$ = #distinct colors

Background

Matroid: $\mathcal{M} = (U, \mathcal{I})$ where $\mathcal{I} \subseteq 2^U$ satisfies certain properties

- U: ground set of n elements, $\mathcal{I}$: notion of independence
- $\mathrm{rank}(S) = \max\{|A|: A \subseteq S, A \in \mathcal{I}\}$
- Basis = maximal/maximum independent set

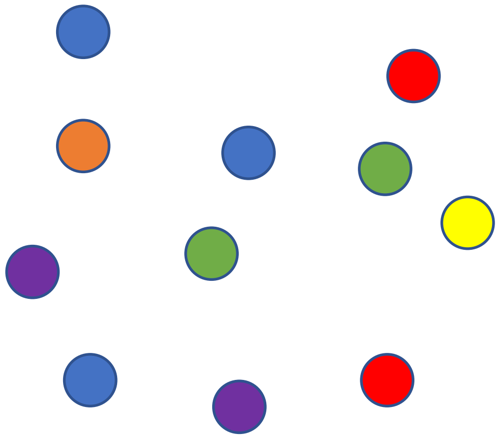


$\mathcal{M}$ = no duplicate colors
$\mathrm{rank}(S)$ = #distinct colors

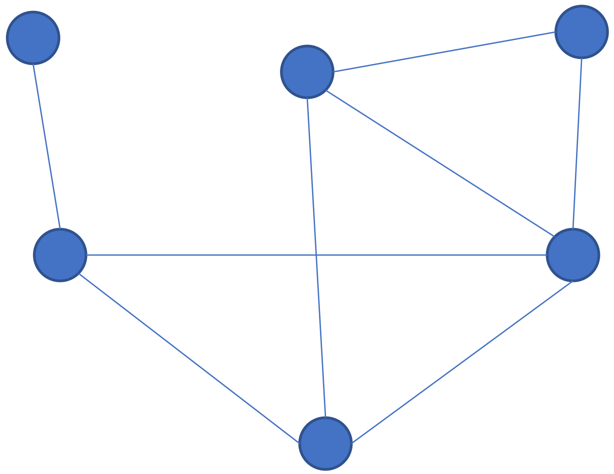
Background

Matroid: $\mathcal{M} = (U, \mathcal{I})$ where $\mathcal{I} \subseteq 2^U$ satisfies certain properties

- U: ground set of n elements, $\mathcal{I}$: notion of independence
- $\text{rank}(S) = \max\{|A|:\ A \subseteq S, A \in \mathcal{I}\}$
- Basis = maximal/maximum independent set



$\mathcal{M}$ = no duplicate colors
$\text{rank}(S)$ = #distinct colors



U = edges
$\mathcal{M}$ = acyclic subgraphs
$\text{rank}(S) = |V| - \#\text{CC}(G[S])$

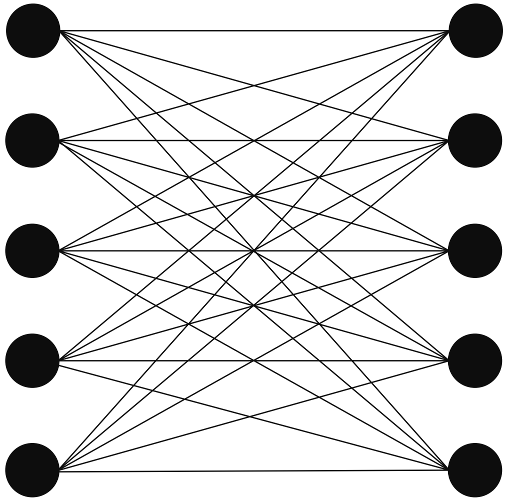
Matroid Problems

Matroid Problems

- **Matroid intersection**
  - Input: $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$
  - Output: the maximum-size $S \in \mathcal{I}_1 \cap \mathcal{I}_2$.

Matroid Problems

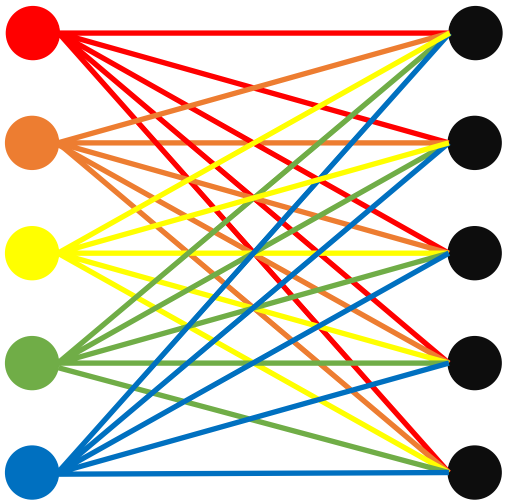
- **Matroid intersection**
  - Input: $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$
  - Output: the maximum-size $S \in \mathcal{I}_1 \cap \mathcal{I}_2$.



U = edges of bipartite graph

Matroid Problems

- **Matroid intersection**
  - Input: $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$
  - Output: the maximum-size $S \in \mathcal{I}_1 \cap \mathcal{I}_2$.

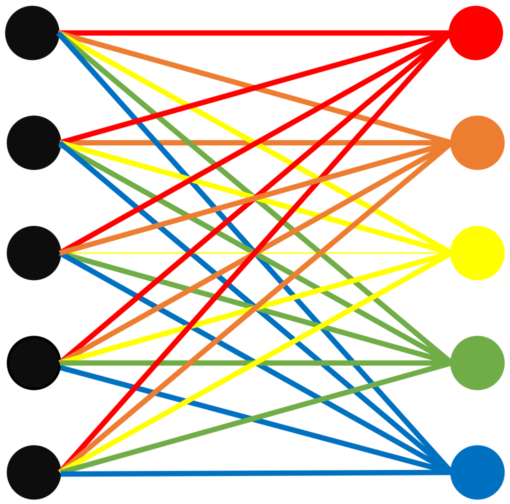


U = edges of bipartite graph

$\mathcal{M}_1$ = left vertex incident to at most one edge

Matroid Problems

- **Matroid intersection**
  - Input: $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$
  - Output: the maximum-size $S \in \mathcal{I}_1 \cap \mathcal{I}_2$.



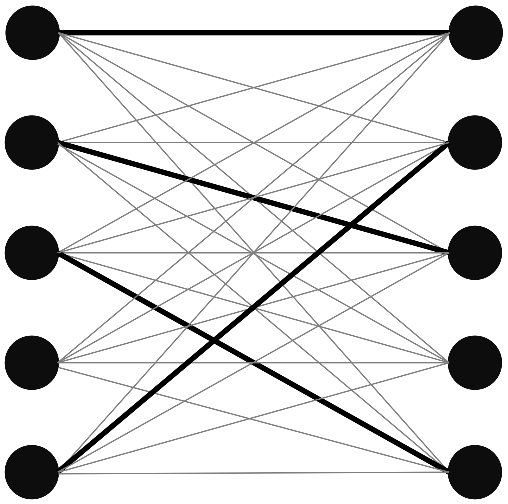
U = edges of bipartite graph

$\mathcal{M}_1$ = left vertex incident to at most one edge

$\mathcal{M}_2$ = right vertex incident to at most one edge

Matroid Problems

- **Matroid intersection**
  - Input: $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$
  - Output: the maximum-size $S \in \mathcal{I}_1 \cap \mathcal{I}_2$.



U = edges of bipartite graph

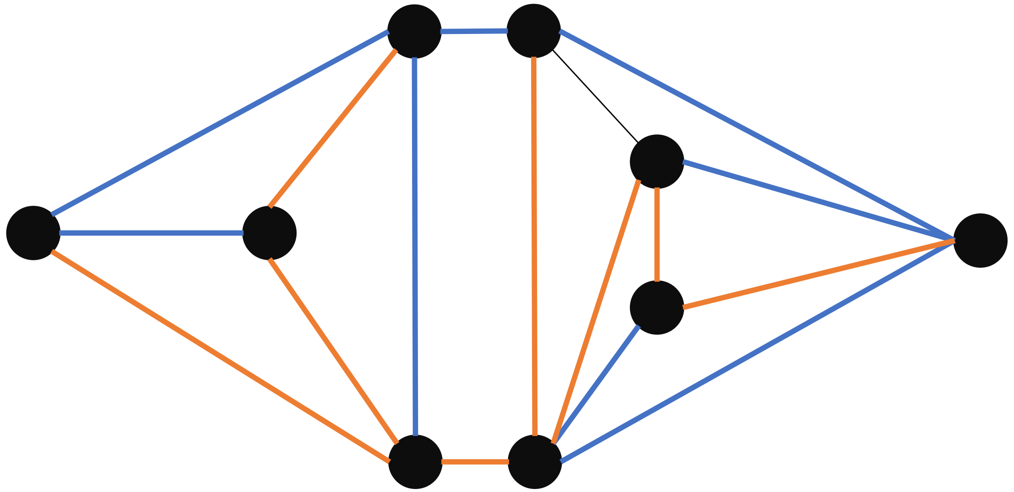
$\mathcal{M}_1$ = left vertex incident to at most one edge

$\mathcal{M}_2$ = right vertex incident to at most one edge

Matroid Problems

- **Matroid intersection**
  - Input: $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$
  - Output: the maximum-size $S \in \mathcal{I}_1 \cap \mathcal{I}_2$.
- **Matroid union**
  - Input $\mathcal{M}_1 = (U_1, \mathcal{I}_1)$, …, $\mathcal{M}_k = (U_k, \mathcal{I}_k)$.
  - Output: the maximum-size $S = S_1 \cup \cdots \cup S_k$ where $S_i \in \mathcal{I}_i$.
- **$\boldsymbol{k}$-fold Matroid union**
  - Input $\mathcal{M} = (U, \mathcal{I})$.
  - Output: the maximum-size $S = S_1 \cup \cdots \cup S_k$ where $S_i \in \mathcal{I}$.

Matroid Problems



$\mathcal{M}$ = graphic matroid, $k = 2$

- **Matroid intersection**
  - Input: $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$
  - Output: the maximum-size $S \in \mathcal{I}_1 \cap \mathcal{I}_2$.
- **Matroid union**
  - Input $\mathcal{M}_1 = (U_1, \mathcal{I}_1)$, …, $\mathcal{M}_k = (U_k, \mathcal{I}_k)$.
  - Output: the maximum-size $S = S_1 \cup \cdots \cup S_k$ where $S_i \in \mathcal{I}_i$.
- ***k*-fold Matroid union**
  - Input $\mathcal{M} = (U, \mathcal{I})$.
  - Output: the maximum-size $S = S_1 \cup \cdots \cup S_k$ where $S_i \in \mathcal{I}$.

Matroid Problems

- **Matroi**
  - Input
  - Outp
- **Matroi**
  - Input
  - Outp
- **k-fold I**
  - Input
  - Outp

Examples:

- Bipartite matching
- k-Disjoint spanning tree
- Colorful spanning tree
- Arboricity & spanning tree packing number
- Some scheduling problems
- Some graph orientation problems
- …

A special case of submodular function minimization (SFM).

Matroid Problems

- **Matroid intersection**
  - Input: $\mathcal{M}_1 = (U, \mathcal{I}_1)$ and $\mathcal{M}_2 = (U, \mathcal{I}_2)$
  - Output: the maximum-size $S \in \mathcal{I}_1 \cap \mathcal{I}_2$.
- **Matroid union**
  - Input $\mathcal{M}_1 = (U_1, \mathcal{I}_1)$, …, $\mathcal{M}_k = (U_k, \mathcal{I}_k)$.
  - Output: the maximum-size $S = S_1 \cup \cdots \cup S_k$ where $S_i \in \mathcal{I}_i$.
- ***k*-fold Matroid union**
  - Input $\mathcal{M} = (U, \mathcal{I})$.
  - Output: the maximum-size $S = S_1 \cup \cdots \cup S_k$ where $S_i \in \mathcal{I}$.
- Notation: $n = \max |U_i|$, $r = \max \operatorname{rank}(\mathcal{M}_i)$

(number of edges $|E|$) (number of vertices $|V|$)

Oracle Complexity vs Runtime

- How do algorithms access matroids?

Oracle Complexity vs Runtime

- How do algorithms access matroids?
  - *Independence* oracle: determine if S is independent.
  - *Rank* oracle: compute $\mathrm{rank}(S)$.

Oracle Complexity vs Runtime

- How do algorithms access matroids?
  - *Independence* oracle: determine if S is independent.
  - *Rank* oracle: compute $\mathrm{rank}(S)$.
- "Traditional" model: minimize the number of queries.

Oracle Complexity vs Runtime

- How do algorithms access matroids?
  - *Independence* oracle: determine if S is independent.
  - *Rank* oracle: compute $\mathrm{rank}(S)$.
- "Traditional" model: minimize the number of queries.

$\tilde{O}(n\sqrt{r})$ rank-query algorithm [CLSSW'19]

Oracle Complexity vs Runtime

- How do algorithms access matroids?
  - *Independence* oracle: determine if S is independent.
  - *Rank* oracle: compute $\mathrm{rank}(S)$.
- "Traditional" model: minimize the number of queries.

$\tilde{O}(n\sqrt{r})$ rank-query algorithm [CLSSW'19]

$\tilde{O}(|E|\sqrt{|V|})$ bipartite matching [HK'73]
$\tilde{O}(|E|\sqrt{|V|})$ colorful spanning tree [GS'85]
$\tilde{O}(|E|\sqrt{|V|})$ graphic MI [GX'89]
$\tilde{O}(n\sqrt{r})$ convex transversal MI [XG'94]
…

Oracle Complexity vs Runtime

- How do algorithms access matroids?
  - *Independence* oracle: determine if S is independent.
  - *Rank* oracle: compute $\mathrm{rank}(S)$.
- "Traditional" model: minimize the number of queries.

$\tilde{O}(n\sqrt{r})$ rank-query algorithm [CLSSW'19]

$\tilde{O}(|E|\sqrt{|V|})$ bipartite matching [HK'73]
$\tilde{O}(|E|\sqrt{|V|})$ colorful spanning tree [GS'85]
$\tilde{O}(|E|\sqrt{|V|})$ graphic MI [GX'89]
$\tilde{O}(n\sqrt{r})$ convex transversal MI [XG'94]
…

Question. Does *efficient* matroid algorithm imply *efficient* algorithms on the right via a *black-box* reduction?

Oracle Complexity vs Runtime

- How do algorithms access matroids?
  - *Independence* oracle: determine if S is independent.
  - *Rank* oracle: compute $\mathrm{rank}(S)$.
- "Traditional" model: minimize the number of queries.

$\tilde{O}(n\sqrt{r})$ rank-query algorithm [CLSSW'19]

$\tilde{O}(|E|\sqrt{|V|})$ bipartite matching [HK'73]
$\tilde{O}(|E|\sqrt{|V|})$ colorful spanning tree [GS'85]
$\tilde{O}(|E|\sqrt{|V|})$ graphic MI [GX'89]
$\tilde{O}(n\sqrt{r})$ convex transversal MI [XG'94]
…

<u>Question.</u> Does *efficient* matroid algorithm imply *efficient* algorithms on the right via a *black-box* reduction? No! $\mathrm{rank}(S)$ takes at least $\Omega(|S|)$ time!

New “Dynamic” Oracle

Question. Can we get *efficient* reduction to matroid algorithms and a *unified* way of solving matroid problem instances?

New "Dynamic" Oracle

Question. Can we get *efficient* reduction to matroid algorithms and a *unified* way of solving matroid problem instances?

- Hard: Compute $\mathrm{rank}(S)$ from scratch, takes $\Omega(|S|)$ time.
- (Potentially) Easier: Update $\mathrm{rank}(S)$ to $\mathrm{rank}(S \pm \{e\})$.
  - Colorful matroid: counting, $O(1)$
  - Graphic matroid: connectivity, $O(\mathrm{polylog}\, n)$

New “Dynamic” Oracle

Question. Can we get *efficient* reduction to matroid algorithms and a *unified* way of solving matroid problem instances?

- Hard: Compute $\mathrm{rank}(S)$ from scratch, takes $\Omega(|S|)$ time.
- (Potentially) Easier: Update $\mathrm{rank}(S)$ to $\mathrm{rank}(S \pm \{e\})$.
  - Colorful matroid: counting, $O(1)$
  - Graphic matroid: connectivity, $O(\mathrm{polylog}\ n)$
- Goal: Issue “close” queries.

New "Dynamic" Oracle

<u>Question.</u> Can we get *efficient* reduction to matroid algorithms and a *unified* way of solving matroid problem instances?

- Hard: Compute $\mathrm{rank}(S)$ from scratch, takes $\Omega(|S|)$ time.
- (Potentially) Easier: Update $\mathrm{rank}(S)$ to $\mathrm{rank}(S \pm \{e\})$.
  - Colorful matroid: counting, $O(1)$
  - Graphic matroid: connectivity, $O(\mathrm{polylog}\, n)$
- Goal: Issue "close" queries.

<u>Definition (Dynamic Oracle).</u> Starting from $Q_0 = \emptyset$, in the k-th query the algorithm can only ask $Q_k = Q_i \pm \{e\}$ for some $i < k$.

New "Dynamic" Oracle

<u>Question.</u> Can we get *efficient* reduction to matroid algorithms and a *unified* way of solving matroid problem instances?

- Hard: Compute $\text{rank}(S)$ from scratch, takes $\Omega(|S|)$ time.
- (Potentially) Easier: Update $\text{rank}(S)$ to $\text{rank}(S \pm \{e\})$.
  - Colorful matroid: counting, $O(1)$
  - Graphic matroid: connectivity, $O(\text{polylog}\, n)$
- Goal: Issue "close" queries.

<u>Definition (Dynamic Oracle).</u> Starting from $Q_0 = \emptyset$, in the k-th query the algorithm can only ask $Q_k = Q_i \pm \{e\}$ for some $i < k$.

Minimize the number of "dynamic" queries.

“Dynamic” Oracle Complexity vs Runtime

$\tilde{O}(n\sqrt{r})$ “dynamic”-rank-query algorithm [this paper]

$\tilde{O}(|E|\sqrt{|V|})$ bipartite matching [HK’73]
$\tilde{O}(|E|\sqrt{|V|})$ colorful spanning tree [GS’85]
$\tilde{O}(|E|\sqrt{|V|})$ graphic MI [GX’89]
$\tilde{O}(n\sqrt{r})$ convex transversal MI [XG’94]
…

“Dynamic” Oracle Complexity vs Runtime

$\tilde{O}(n\sqrt{r})$ “dynamic”-rank-query algorithm [this paper]

Dynamic connectivity [KKM’13]
Dynamic convex bipartite matching [BGHK’07]
Dynamic rank maintenance [vdBNS’19]
…

Worst-case to fully-persistence [DSST’86]

$\tilde{O}(|E|\sqrt{|V|})$ bipartite matching [HK’73]
$\tilde{O}(|E|\sqrt{|V|})$ colorful spanning tree [GS’85]
$\tilde{O}(|E|\sqrt{|V|})$ graphic MI [GX’89]
$\tilde{O}(n\sqrt{r})$ convex transversal MI [XG’94]
…

Main Results

Main Results

Theorem 1. Matroid intersection can be solved in $\tilde{O}(n\sqrt{r})$ time and dynamic rank queries.

Main Results

Theorem 1. Matroid intersection can be solved in $\tilde{O}(n\sqrt{r})$ time and dynamic rank queries.

We also have dynamic-independence-query algorithm.

Main Results

Theorem 1. Matroid intersection can be solved in $\tilde{O}(n\sqrt{r})$ time and dynamic rank queries.

Theorem 2. Matroid union over k matroids can be solved in $\tilde{O}_k(n + r\sqrt{r})$ time and dynamic rank queries.

Main Results

<u>Theorem 1.</u> Matroid intersection can be solved in $\tilde{O}(n\sqrt{r})$ time and dynamic rank queries.

<u>Theorem 2.</u> Matroid union over k matroids can be solved in $\tilde{O}_k(n + r\sqrt{r})$ time and dynamic rank queries.

Imply an $\tilde{O}_k(|E| + |V|\sqrt{|V|})$ k-disjoint spanning tree algorithm.
(Concurrently and independently by [Quanrud'23]).

Main Results

Theorem 1. Matroid intersection can be solved in $\tilde{O}(n\sqrt{r})$ time and dynamic rank queries.

Theorem 2. Matroid union over k matroids can be solved in $\tilde{O}_k(n + r\sqrt{r})$ time and dynamic rank queries.

Theorem 3. Matroid intersection requires $\Omega(n \log n)$ “dynamic” rank queries or “traditional” independence queries deterministically.

Main Results

Theorem 1. Matroid intersection can be solved in $\tilde{O}(n\sqrt{r})$ time and dynamic rank queries.

Theorem 2. Matroid union over k matroids can be solved in $\tilde{O}_k(n + r\sqrt{r})$ time and dynamic rank queries.

Theorem 3. Matroid intersection requires $\Omega(n \log n)$ "dynamic" rank queries or "traditional" independence queries deterministically.

Improve the $\log_2(3)n - o(n) \approx 1.58n$ lower bound of [Harvey'08].

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

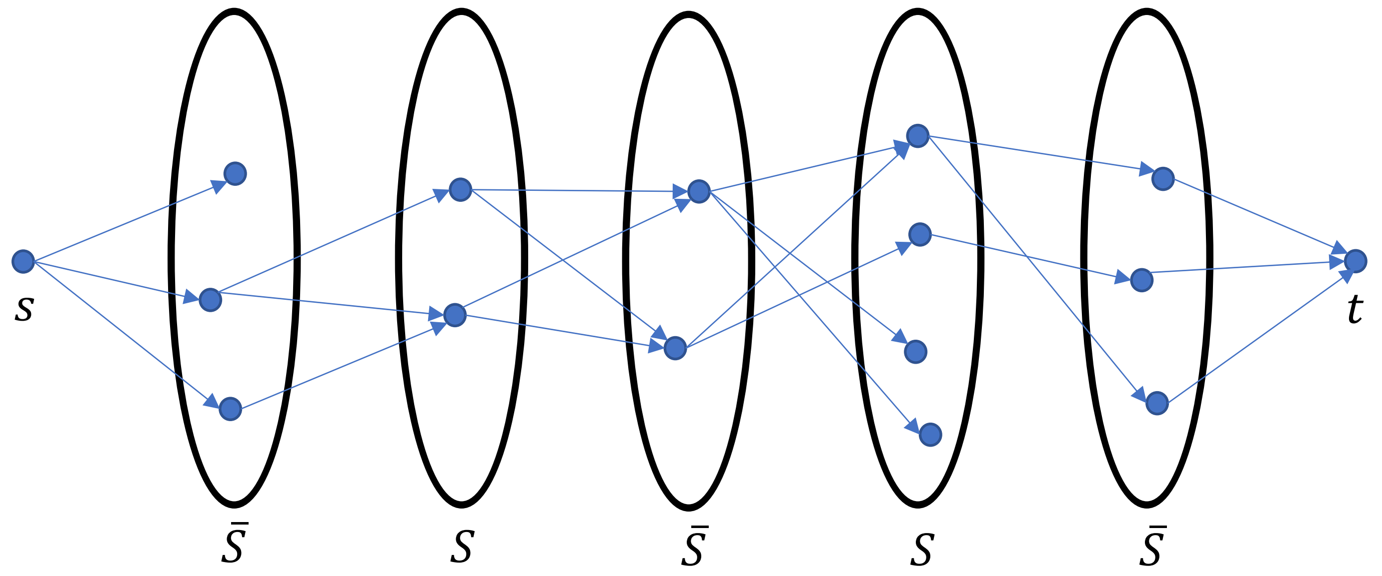
problems	our bounds	state-of-the-art results
(Via k-fold matroid union)		
k-forest[8]	$\tilde{O}(\lvert E\rvert + (k\lvert V\rvert)^{3/2})$ ✓	$\tilde{O}(k^{3/2}\lvert V\rvert\sqrt{\lvert E\rvert})$ [GW88]
k-pseudoforest	$\tilde{O}(\lvert E\rvert + (k\lvert V\rvert)^{3/2})$ ✗	$\lvert E\rvert^{1+o(1)}$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

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

Overview of Techniques

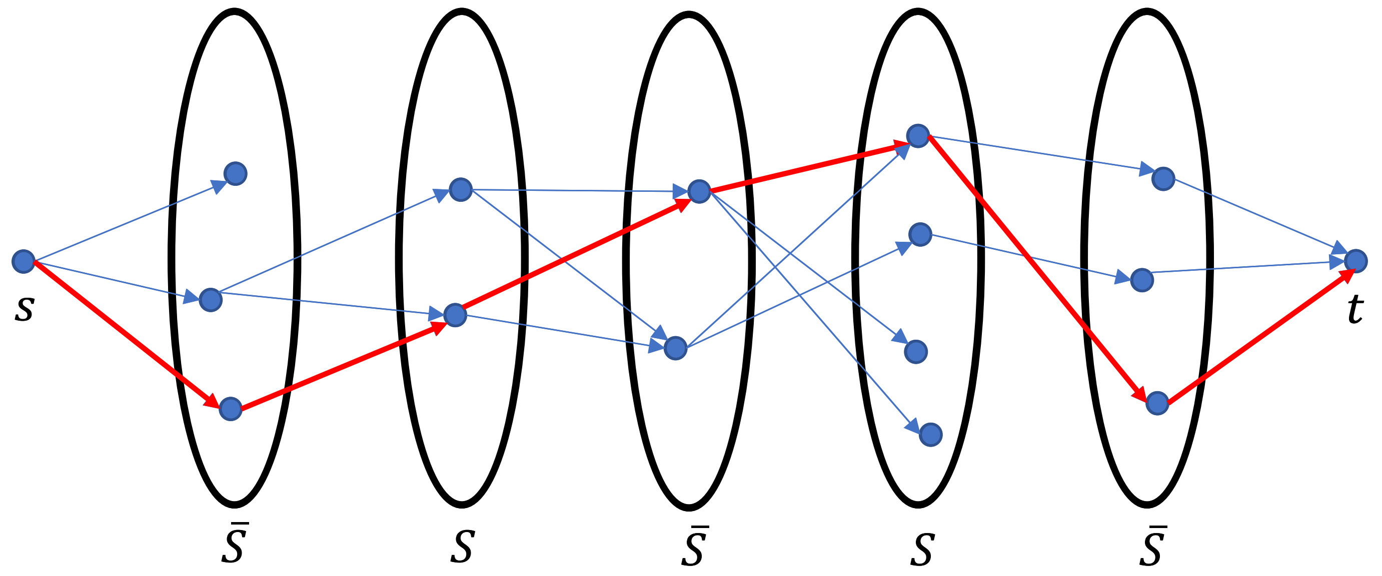
Overview of Techniques

- **Matroid intersection**
  - Construct "binary search tree" to explore a single augmenting path
  - Lazily update the tree when the solution changes
  - Apply structural lemmas of [CLSSW'19] to reduce the number of updates to the tree

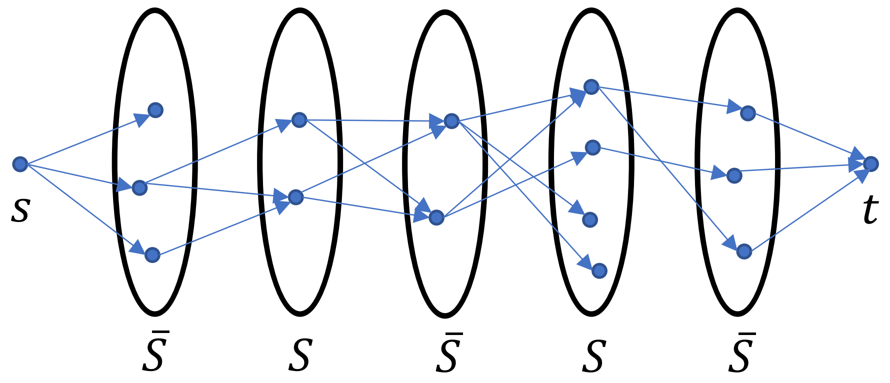


Overview of Techniques

- **Matroid intersection**
  - Construct “binary search tree” to explore a single augmenting path
  - Lazily update the tree when the solution changes
  - Apply structural lemmas of [CLSSW’19] to reduce the number of updates to the tree

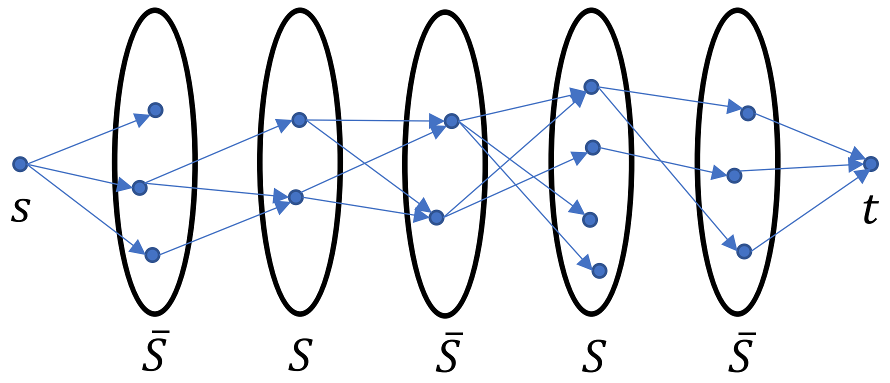


Overview of Techniques



- **Matroid intersection**
  - Construct "binary search tree" to explore a single augmenting path
  - Lazily update the tree when the solution changes
  - Apply structural lemmas of [CLSSW'19] to reduce the number of updates to the tree
- **Matroid union** (from $\tilde{O}(n\sqrt{r})$ to $\tilde{O}(n + r\sqrt{r})$, think of $n \gg r$)
  - Sparsify the exchange graph with a decremental basis data structure
  - SQRT-decomposition + sparsification ($\tilde{O}\left(\sqrt{|V|}\right)$ dynamic MST [Fre'85, EGIN'97])

Overview of Techniques



- **Matroid intersection**
  - Construct “binary search tree” to explore a single augmenting path
  - Lazily update the tree when the solution changes
  - Apply structural lemmas of [CLSSW’19] to reduce the number of updates to the tree
- **Matroid union** (from $\tilde{O}(n\sqrt{r})$ to $\tilde{O}(n + r\sqrt{r})$, think of $n \gg r$)
  - Sparsify the exchange graph with a decremental basis data structure
  - SQRT-decomposition + sparsification ($\tilde{O}\left(\sqrt{|V|}\right)$ dynamic MST [Fre’85, EGIN’97])
- **Lower bounds**
  - Reduce to communication complexity of matroid intersection
  - Then a reduction from (s, t)-connectivity: $\Omega(n \log n)$ lower bound [HMT’88]

Some Future Directions

Some Future Directions

- **Colorful spanning tree**
  - New way of using interior point methods (IPMs), or
  - Faster “combinatorial” algorithms for bipartite matching.

Some Future Directions

- **Colorful spanning tree**
  - New way of using interior point methods (IPMs), or
  - Faster "combinatorial" algorithms for bipartite matching.
- $\boldsymbol{k}$**-Disjoint spanning tree**
  - k-disjoint "arborescence" can be solved in $\tilde{O}_k(m)$ time.
  - We don't know anything better even when $k = 2$ for the undirected case.

Some Future Directions

- **Colorful spanning tree**
  - New way of using interior point methods (IPMs), or
  - Faster “combinatorial” algorithms for bipartite matching.
- **$\boldsymbol{k}$-Disjoint spanning tree**
  - k-disjoint “arborescence” can be solved in $\tilde{O}_k(m)$ time.
  - We don’t know anything better even when $k = 2$ for the undirected case.
- **Other useful dynamic oracles**
  - Submodular function minimization/maximization

Some Future Directions

- **Colorful spanning tree**
  - New way of using interior point methods (IPMs), or
  - Faster "combinatorial" algorithms for bipartite matching.
- **$\boldsymbol{k}$-Disjoint spanning tree**
  - k-disjoint "arborescence" can be solved in $\tilde{O}_k(m)$ time.
  - We don't know anything better even when $k = 2$ for the undirected case.
- **Other useful dynamic oracles**
  - Submodular function minimization/maximization
- **Improved lower bounds**
  - $\Omega(n \log n)$ "traditional"-rank-query lower bound implies better SFM lower bounds [CGJS'22].

Thanks for Listening!

Paper: https://arxiv.org/abs/2302.09796
1h-talk by Joakim at ETHZ: https://www.youtube.com/live/XgSTiseAaW8