Breaking $O(nr)$ for Matroid Intersection

Joakim Blikstad

ICALP 2021

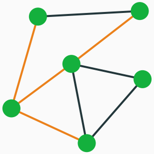
KTH Royal Institute of Technology, Sweden

What is a Matroid?

- Set of elements V. $n = |V|$.
- Notion of independence $\mathcal{I} \subseteq 2^V$.

Exact definition is not important for this presentation.

Graphic Matroid



$V =$ edges

$\mathcal{I} =$ forests

Linear Matroid

$$\begin{bmatrix} 0 & 1 & 2 & 0 & 1 \\ 1 & 0 & 1 & 2 & 0 \\ 2 & 0 & 2 & 4 & 0 \\ 1 & 1 & 3 & 2 & 1 \\ 0 & 0 & 1 & 0 & 5 \end{bmatrix}$$

$V =$ row vectors

$\mathcal{I} =$ linearly independent

Matroid Intersection

Given: two matroid $\mathcal{M}_1 = (V, \mathcal{I}_1)$ and $\mathcal{M}_2 = (V, \mathcal{I}_2)$

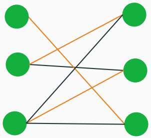
Goal: find a common independent set $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ of maximum size

Queries: independence-oracle *Is* $X \in \mathcal{I}_1$*?* *Is* $X \in \mathcal{I}_2$*?*

Matroid Intersection models many combinatorial optimization problems.

E.g. Bipartite Matching:

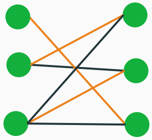
- $\mathcal{M}_1 =$ "≤ 1 edge per vertex on the left"
- $\mathcal{M}_2 =$ "≤ 1 edge per vertex on the right"



Augmenting Paths & The Exchange Graph

- Special case: Bipartite Matching
- Augmenting Path algorithms
  - Similar idea works for matroid intersection too!
  - Find augmenting paths in the Exchange Graph.

[Edmonds 60s]



The $\tilde{O}(nr)$ query bound

($n = $ #elements, $r = $ size of answer)

- $\tilde{O}(nr)$ bound $\approx$ "find each of the r augmenting path in $O(n)$ queries".

[Nguyen, CLSSW 2019]

The $\tilde{O}(nr)$ query bound

($n = \#$elements, $r =$ size of answer)

- $\tilde{O}(nr)$ bound $\approx$ "find each of the r augmenting path in $O(n)$ queries".
  [Nguyen, CLSSW 2019]
- To beat this we need to find several paths "in parallel".

The $\tilde{O}(nr)$ query bound

($n = \#$elements, $r =$ size of answer)

- $\tilde{O}(nr)$ bound $\approx$ "find each of the r augmenting path in $O(n)$ queries".

  [Nguyen, CLSSW 2019]

- To beat this we need to find several paths "in parallel".

- **Challenge:** Exchange Graph changes after each augmentation:
  - Some edges added, some removed.
  - Set of vertex disjoint paths $\not\Longrightarrow$ augment along all of them.

    $\uparrow$ unlike for bipartite matching / max-flow

    (Hopcroft-Karp / Dinitz)

The $\tilde{O}(nr)$ query bound (cont.)

($n = \#$elements, $r =$ size of answer)

Breaking $\tilde{O}(nr)$:

- **Previous:** large $r = \omega(\sqrt{n})$:
  - $(1-\varepsilon)$-Approx.: $\tilde{O}\left(\frac{n\sqrt{n}}{\varepsilon\sqrt{\varepsilon}}\right)$ queries [CLSSW 2019]
  - Exact: $\tilde{O}(n^{6/5}r^{3/5})$ queries [BvdBMN 2021]

The $\tilde{O}(nr)$ query bound (cont.)

($n = $ #elements, $r = $ size of answer)

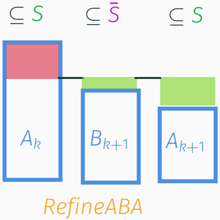
Breaking $\tilde{O}(nr)$:

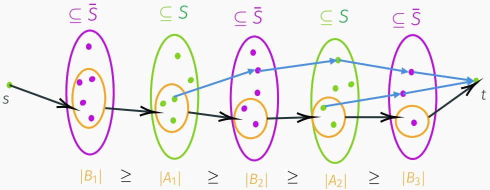
- **Previous:** large $r = \omega(\sqrt{n})$:
  - $(1-\varepsilon)$-Approx.: $\tilde{O}\left(\frac{n\sqrt{n}}{\varepsilon\sqrt{\varepsilon}}\right)$ queries [CLSSW 2019]
  - Exact: $\tilde{O}(n^{6/5}r^{3/5})$ queries [BvdBMN 2021]
- **This paper:** full range of r:
  - $(1-\varepsilon)$-Approx.: $\tilde{O}\left(\frac{n\sqrt{r}}{\varepsilon}\right)$ queries
  - Exact: $\tilde{O}(nr^{3/4})$ queries

Technique

Approximation

Improve algorithm of [CLSSW] with two new ideas.





Exact

Plug in the approximation algorithm in the framework of [BvdBMN].

Open Problems

- Gap between lower and upper bounds for matroid intersection.
  - No $\Omega(n^{1+\delta})$ lower-bound is known for $\delta > 0$.
- Can one also solve **weighted** matroid intersection in $o(nr)$ queries?

Thanks!

Extra Slides

Summary

Result: ($n = $#elements, $r = $ size of answer)

First independence-query matroid intersection algorithms breaking $\tilde{O}(nr)$.

- $(1-\varepsilon)$-approximation
  - Previous best: $O(nr/\varepsilon)$ and $\tilde{O}(n^{1.5}/\varepsilon^{1.5})$.
  - **Ours:** $\tilde{O}(n\sqrt{r}/\varepsilon)$ queries.
- Exact:
  - Previous best: $\tilde{O}(nr)$ and $\tilde{O}(n^{6/5}r^{3/5})$
  - **Ours:** $\tilde{O}(nr^{3/4})$ queries

Technique:

- $(1-\varepsilon)$-approximate: Improve CLSSW's algorithm with two new ideas.
- Exact: Plug in approximate algorithm in the framework of BvdBMN.

Exact Algorithm

Algorithm [BvdBMN]:

1. Many short paths: $(1-\varepsilon)$-approximation algorithm
2. Few remaining long paths: find them one by one

Old Query Complexity: $\tilde{O}(n^{6/5}r^{3/5})$

Exact Algorithm

Algorithm [BvdBMN]:

1. Many short paths: $(1-\varepsilon)$-approximation algorithm
2. Few remaining long paths: find them one by one

Old Query Complexity: $\tilde{O}(n^{6/5}r^{3/5})$

Bottleneck: $\tilde{O}\left(\frac{n\sqrt{n}}{\varepsilon\sqrt{\varepsilon}}\right)$ approximation algorithm by [CLSSW].

Exact Algorithm

Algorithm [BvdBMN]:

1. Many short paths: $(1-\varepsilon)$-approximation algorithm
2. Few remaining long paths: find them one by one

Old Query Complexity: $\tilde{O}(n^{6/5}r^{3/5})$

Bottleneck: $\tilde{O}\left(\frac{n\sqrt{n}}{\varepsilon\sqrt{\varepsilon}}\right)$ approximation algorithm by [CLSSW].

Replace with our improved $\tilde{O}\left(\frac{n\sqrt{r}}{\varepsilon}\right)$ approximation algorithm:
New Query Complexity: $\tilde{O}(nr^{3/4})$

Approximation Algorithm

We improve the $\tilde{O}\left(\frac{n\sqrt{n}}{\varepsilon\sqrt{\varepsilon}}\right)$ approx-algorithm [CLSSW]:

Algorithm [CLSSW]
Run in $O(1/\varepsilon)$ phases and find "blocking-flow":

Stage 1: Keep refining a partial augmenting set.

Stage 2: When progress stagnates, find remaining paths one at a time.

Blocking-Flow: (think Hopcroft-Karp / Dinitz's)
Find a maximal set of "compatible" augmenting paths of the same length.

Approximation Algorithm — Improvements

Algorithm [CLSSW]
Run in $O(1/\varepsilon)$ phases and find “blocking-flows”:

Stage 1: Keep refining a partial augmenting set.

Stage 2: When progress stagnates, find remaining paths one at a time.

This Paper: Two new improvements:

Approximation Algorithm — Improvements

Algorithm [CLSSW]
Run in $O(1/\varepsilon)$ phases and find "blocking-flows":

Stage 1: Keep refining a partial augmenting set.

Stage 2: When progress stagnates, find remaining paths one at a time.

This Paper: Two new improvements:

- In stage 1: We refine on three consecutive layers instead of two.
  - Guarantees we make "progress" on "even" layers $\subseteq S$. $|S| \leq r$.
  - Replaces $\sqrt{n}$ term with $\sqrt{r}$.

Approximation Algorithm — Improvements

Algorithm [CLSSW]
Run in $O(1/\varepsilon)$ phases and find "blocking-flows":

Stage 1: Keep refining a partial augmenting set.

Stage 2: When progress stagnates, find remaining paths one at a time.

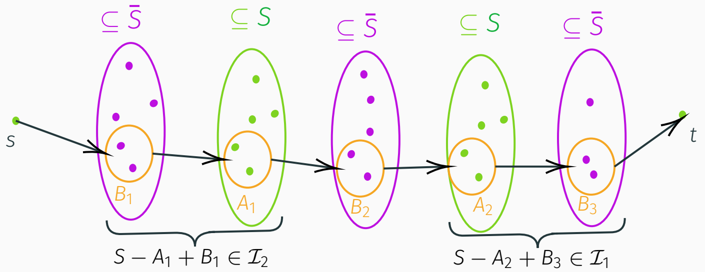
This Paper: Two new improvements:

- In stage 1: We refine on three consecutive layers instead of two.
  - Guarantees we make "progress" on "even" layers $\subseteq S$. $|S| \leq r$.
  - Replaces $\sqrt{n}$ term with $\sqrt{r}$.
- In stage 2: We find paths directly on top of the output of stage 1.
  - Fewer path need to be found.
  - Shaves of $1/\sqrt{\varepsilon}$-factor.

Augmenting Sets [CLSSW]

Augmenting Sets ≈ Collection of “compatible” augmenting paths.

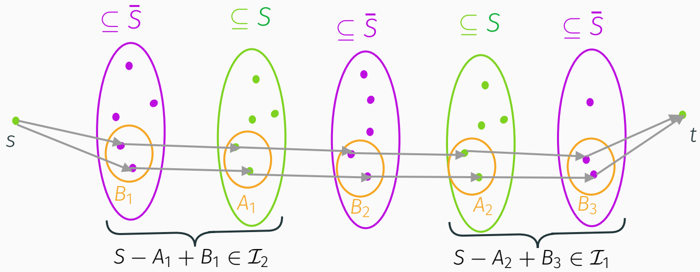
Only **local** constraints:
“$S - A + B \in \mathcal{I}$” where A and B are in adjacent distance-layers.



Augmenting Sets [CLSSW]

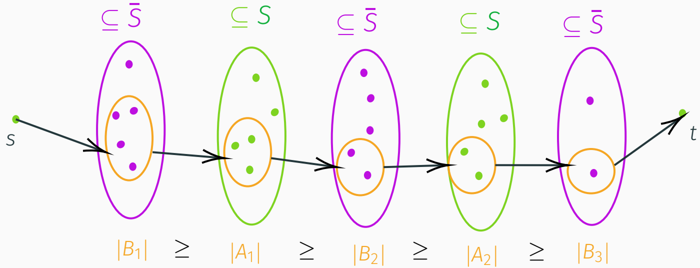
Augmenting Sets $\approx$ Collection of “compatible” augmenting paths.

Only **local** constraints:
“$S - A + B \in \mathcal{I}$” where A and B are in adjacent distance-layers.



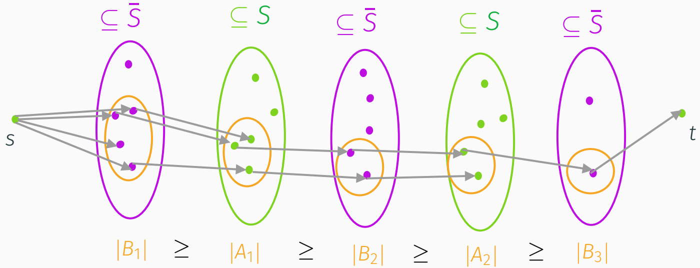
Finding a maximal augmenting set ("Blocking-Flow")

Keep track of a partial augmenting set.



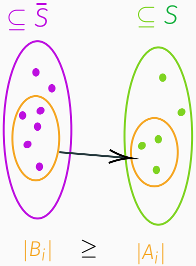
Finding a maximal augmenting set (“Blocking-Flow”)

Keep track of a partial augmenting set.



Refining

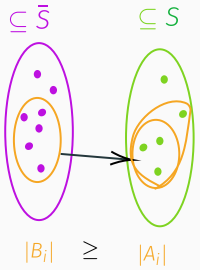
Locally improve two consecutive layers.



Refining

Locally improve two consecutive layers.

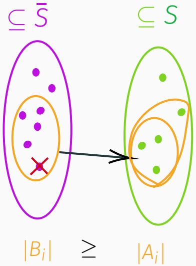
1. Extend A_i while it can be “matched” from B_i.



Refining

Locally improve two consecutive layers.

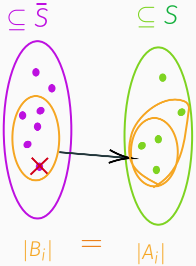
1. Extend A_i while it can be "matched" from B_i.
2. Throw away "unmatched" elements of B_i.



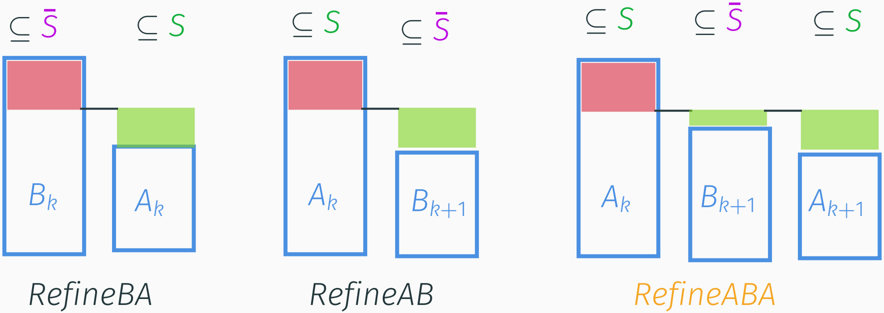
Refining

Locally improve two consecutive layers.

1. Extend A_i while it can be "matched" from B_i.
2. Throw away "unmatched" elements of B_i.
3. Now $|A_i| = |B_i|$.



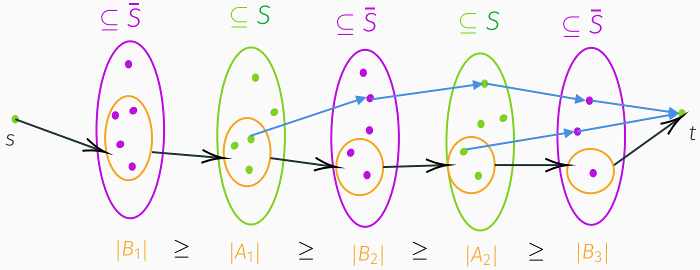
New Idea 1: Refining 3 consecutive layers



Guarantees that we make progress on "even" layers $\subseteq S$. $|S| \leq r$.

Replaces $\sqrt{n}$ term with $\sqrt{r}$. $\longleftarrow$ allows us $o(nr)$ algorithms.

New Idea 2: Finding Paths



When refining-progress stagnates:

- Fall back to finding augmenting paths individually.
- **New Idea:** Find them with respect to partial aug-set $(B_1, A_1, \ldots B_{\ell+1})$.

Lowers dependence on ε from $O(1/\varepsilon^{1.5})$ to $O(1/\varepsilon)$.

Summary

Result: ($n = $ #elements, $r = $ size of answer)

First independence-query matroid intersection algorithms breaking $\tilde{O}(nr)$.

- $(1-\varepsilon)$-approximation
  - Previous best: $O(nr/\varepsilon)$ and $\tilde{O}(n^{1.5}/\varepsilon^{1.5})$.
  - **Ours:** $\tilde{O}(n\sqrt{r}/\varepsilon)$ queries.
- Exact:
  - Previous best: $\tilde{O}(nr)$ and $\tilde{O}(n^{6/5}r^{3/5})$
  - **Ours:** $\tilde{O}(nr^{3/4})$ queries

Technique:

- $(1-\varepsilon)$-approximate: Improve CLSSW's algorithm with two new ideas.
- Exact: Plug in approximate algorithm in the framework of BvdBMN.