Subquadratic Submodular Maximization with a General Matroid Constraint

Yusuke Kobayashi, **Tatsuya Terao**
Kyoto University

ICALP 2024@Tallinn

Submodular Function

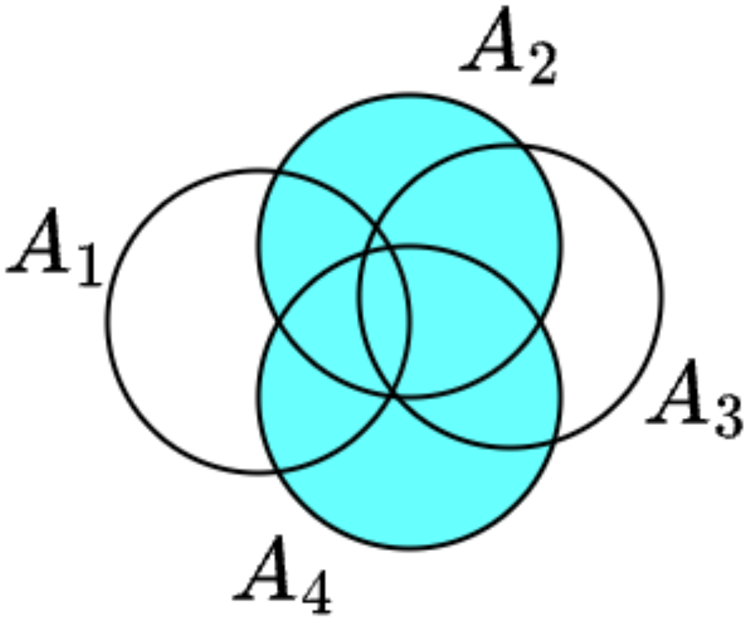
diminishing returns property

Def

$f: 2^V \to \mathbb{R}$ such that

$\boldsymbol{S} \subseteq \boldsymbol{T} \subseteq \boldsymbol{V}, \boldsymbol{v} \in \boldsymbol{V} \setminus \boldsymbol{T} \Longrightarrow \boldsymbol{f}(\boldsymbol{S} \cup \{\boldsymbol{v}\}) - \boldsymbol{f}(\boldsymbol{S}) \geq \boldsymbol{f}(\boldsymbol{T} \cup \{\boldsymbol{v}\}) - \boldsymbol{f}(\boldsymbol{T})$

e.g.) • **Coverage Function**



Given: $A_1, \ldots, A_n \subset U$

$\boldsymbol{f}(\boldsymbol{S}) = |\cup_{\boldsymbol{i} \in \boldsymbol{S}} \boldsymbol{A}_{\boldsymbol{i}}|$

Monotone Submodular Maximization

$\boldsymbol{f(S) \leq f(T) \ \ (S \subseteq T)}$

$\boldsymbol{f}: \mathbf{2}^{\boldsymbol{V}} \to \mathbb{R}_{\geq \mathbf{0}}$ **: Monotone Submodular**, $f(\emptyset) = 0$

$$\textbf{max} \quad \boldsymbol{f(S)} \qquad \text{s.t.} \qquad \boldsymbol{S \in \mathcal{C}}$$

$\boldsymbol{|S| \leq r}$**, etc.**

- Generalization of Many Problems
  e.g., Maximum Coverage, Facility Location
- Many Practical Applications
  e.g., Machine Learning, Vision, Economics

Monotone Submodular Maximization with a Cardinality Constraint

[Fisher-Nemhauser-Wolsey 1978]

$f: 2^V \to \mathbb{R}_{\geq 0}$: Monotone Submodular

$$\max \quad f(S) \qquad \text{s.t.} \quad |S| \leq r$$

☞ Greedy algorithm achieves **(1 − 1/e)-approximation**

$$f(S) \geq (1 - 1/e)\, f(OPT)$$

☞ **Approximation ratio $1 - 1/e$ is optimal**

[Nemhauser-Wolsey 1978]

Monotone Submodular Maximization with a Matroid Constraint

$f: 2^V \to \mathbb{R}_{\geq 0}$: Monotone Submodular

$$\max \quad f(S) \qquad \text{s.t.} \quad \boxed{\boldsymbol{S \in \mathcal{I}}}$$

☞ **(1 − 1/e)-approximation ?**

Monotone Submodular Maximization with a Matroid Constraint

$f: 2^V \to \mathbb{R}_{\geq 0}$: Monotone Submodular

$$\max \quad f(S) \qquad \text{s.t.} \quad \boxed{\boldsymbol{S \in \mathcal{I}}}$$

☞ **Continuous Greedy** achieves **$(1 - 1/e)$-approximation**

[Calinescu-Chekuri-Pál-Vondrák 2007]

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

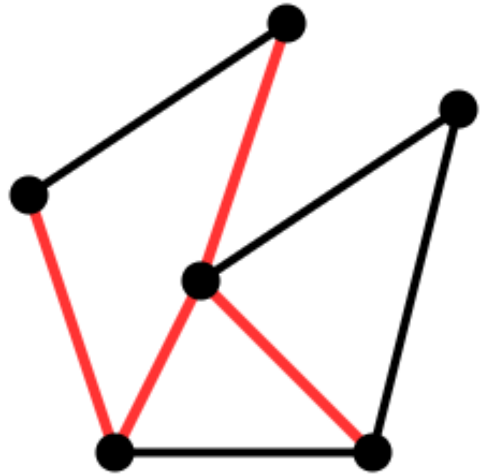
Def

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

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

e.g.)

- **Graphic Matroid**



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

- **Linear Matroid**

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

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

Time Complexity Analysis

☞ Algorithm accesses a submodular function and a matroid through an **oracle**

- **Value** oracle query: $f(S) = ?$
- **Independence** oracle query: Is $S \in \mathcal{I}$?

$(1-1/e)$-approximation for Monotone Submodular Maxization with a Matroid Constraint

		Query Complexity
2007	Calinescu-Chekuri-Pál-Vondrák	$\tilde{O}(n^8)$

$n = |V|$, $\boldsymbol{r} =$ **rank of matroid** $(\leq \boldsymbol{n})$

$(1-1/e-\epsilon)$-approximation for Monotone Submodular Maxization with a Matroid Constraint

		Query Complexity
2007	Calinescu-Chekuri-Pál-Vondrák	$\tilde{O}(n^8)$
2012	Filmus-Ward	$\tilde{O}_\epsilon(rn^4)$
2014	Badanidiyuru-Vondrák	$\tilde{O}_\epsilon(rn)$
2015	Buchbinder-Feldman-Schwartz	$\tilde{O}_\epsilon(r^2+\sqrt{r}n)$

$n = |V|$, $r =$ **rank of matroid** $(\leq n)$

$(\mathbf{1 - 1/e - \epsilon})$-approximation for Monotone Submodular Maxization with a Matroid Constraint

Query Complexity

2007	Calinescu-Chekuri-Pál-Vondrák	$\tilde{O}(n^8)$
2012	Filmus-Ward	$\tilde{O}_\epsilon(rn^4)$
2014	Badanidiyuru-Vondrák	$\tilde{O}_\epsilon(rn)$
2015	Buchbinder-Feldman-Schwartz	$\tilde{O}_\epsilon(r^2 + \sqrt{r}n)$
2024	**This work**	$\tilde{\boldsymbol{O}}_\epsilon(\sqrt{\boldsymbol{r}}\boldsymbol{n})$

$n = |V|$, $\boldsymbol{r} =$ **rank of matroid** $(\leq \boldsymbol{n})$

Continuous Greedy Algorithm

[Calinescu-Chekuri-Pál-Vondrák 2007]

Discrete

max $\boldsymbol{f}(\boldsymbol{S})$ s.t. $\boldsymbol{S} \in \boldsymbol{\mathcal{I}}$

submodular function $\boldsymbol{f}: \mathbf{2}^{\boldsymbol{V}} \rightarrow \mathbb{R}_{\geq \mathbf{0}}$

Continuous Greedy Algorithm

[Calinescu-Chekuri-Pál-Vondrák 2007]

Step1. Continuous Relaxation

Discrete

max $\boldsymbol{f}(\boldsymbol{S})$ s.t. $\boldsymbol{S} \in \boldsymbol{\mathcal{I}}$

Continuous

max $\boldsymbol{F}(\boldsymbol{x})$ s.t. $\boldsymbol{x} \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{M}})$

multilinear extension $\boldsymbol{F}\colon [\boldsymbol{0},\boldsymbol{1}]^{V} \to \mathbb{R}_{\geq \boldsymbol{0}}$

$\boldsymbol{F}(\boldsymbol{x}) = \sum_{S \subseteq V} \boldsymbol{f}(\boldsymbol{S}) \prod_{v \in S} \boldsymbol{x}_v \prod_{v \in V \setminus S} (\boldsymbol{1} - \boldsymbol{x}_v)$

matroid base polytope

$\mathcal{B}(\mathcal{M}) = \mathrm{conv}\ \{\chi_B \mid B \in \mathcal{B}\}$

Continuous Greedy Algorithm

[Calinescu-Chekuri-Pál-Vondrák 2007]

Step1. Continuous Relaxation

Discrete

max $\boldsymbol{f}(\boldsymbol{S})$ s.t. $\boldsymbol{S} \in \boldsymbol{\mathcal{I}}$

Continuous

max $\boldsymbol{F}(\boldsymbol{x})$ s.t. $\boldsymbol{x} \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{M}})$

Step2. Continuous Greedy Algorithm

$\boldsymbol{x} \in \mathcal{B}(\mathcal{M})$

s.t.

$\mathbb{E}[\boldsymbol{F}(\boldsymbol{x})] \geq (\mathbf{1} - \mathbf{1}/\boldsymbol{e} - \boldsymbol{\epsilon})\boldsymbol{f}(\boldsymbol{OPT})$

Continuous Greedy Algorithm

[Calinescu-Chekuri-Pál-Vondrák 2007]

Step1. Continuous Relaxation

Discrete

max $\boldsymbol{f}(\boldsymbol{S})$ s.t. $\boldsymbol{S} \in \boldsymbol{\mathcal{I}}$

Continuous

max $\boldsymbol{F}(\boldsymbol{x})$ s.t. $\boldsymbol{x} \in \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{M}})$

Step2. Continuous Greedy Algorithm

$\boldsymbol{x} \in \mathcal{B}(\mathcal{M})$

s.t.

$\mathbb{E}[\boldsymbol{F}(\boldsymbol{x})] \geq (\mathbf{1} - \mathbf{1}/\boldsymbol{e} - \boldsymbol{\epsilon})\boldsymbol{f}(\boldsymbol{OPT})$

$\boldsymbol{S} \in \mathcal{I}$

s.t.

$\mathbb{E}[\boldsymbol{f}(\boldsymbol{S})] \geq (\mathbf{1} - \mathbf{1}/\boldsymbol{e} - \boldsymbol{\epsilon})\boldsymbol{f}(\boldsymbol{OPT})$

Step3. Rounding

Fast Continuous Greedy Algorithm

Discrete

max $\boldsymbol{f(S)}$ s.t. $\boldsymbol{S \in \mathcal{I}}$

Continuous

max $\boldsymbol{F(x)}$ s.t. $\boldsymbol{x \in \mathcal{B}(\mathcal{M})}$

Continuous Greedy Alg

$\boldsymbol{\widetilde{O}_{\epsilon}(rn)}$ **queries**

[Badanidiyuru-Vondrák 2014]

$\boldsymbol{x} \in \mathcal{B}(\mathcal{M})$

s.t.

$\mathbb{E}[\boldsymbol{F}(\boldsymbol{x})] \geq (\boldsymbol{1} - \boldsymbol{1}/\boldsymbol{e} - \boldsymbol{\epsilon})\boldsymbol{f}(\boldsymbol{OPT})$

Rounding

$\boldsymbol{S} \in \mathcal{I}$

s.t.

$\mathbb{E}[\boldsymbol{f}(\boldsymbol{S})] \geq (\boldsymbol{1} - \boldsymbol{1}/\boldsymbol{e} - \boldsymbol{\epsilon})\boldsymbol{f}(\boldsymbol{OPT})$

Fast Continuous Greedy Algorithm

Discrete

$\max f(S)$ s.t. $S \in \mathcal{I}$

Continuous

$\max F(x)$ s.t. $x \in \mathcal{B}(\mathcal{M})$

Continuous Greedy Alg

$\tilde{O}_\epsilon(rn)$ **queries**

[Badanidiyuru-Vondrák 2014]

$x \in \mathcal{B}(\mathcal{M})$

s.t.

$\mathbb{E}[F(x)] \geq (1 - 1/e - \epsilon) f(OPT)$

Rounding

$O_\epsilon(r^2)$ **queries** [Chekuri-Vondrák-Zenklusen 2010]

$S \in \mathcal{I}$

s.t.

$\mathbb{E}[f(S)] \geq (1 - 1/e - \epsilon) f(OPT)$

Fast Continuous Greedy Algorithm

Discrete

max $\boldsymbol{f(S)}$ s.t. $\boldsymbol{S \in \mathcal{I}}$

Continuous

max $\boldsymbol{F(x)}$ s.t. $\boldsymbol{x \in \mathcal{B}(\mathcal{M})}$

Continuous Greedy Alg

$\boldsymbol{\tilde{O}_\epsilon(\sqrt{r}n)}$ **queries**

[Buchbinder-Feldman-Schwartz 2015]

$\boldsymbol{x} \in \mathcal{B}(\mathcal{M})$

s.t.

$\mathbb{E}[\boldsymbol{F(x)}] \geq (\boldsymbol{1} - \boldsymbol{1}/\boldsymbol{e} - \boldsymbol{\epsilon})\boldsymbol{f(OPT)}$

$\boldsymbol{S} \in \mathcal{I}$

s.t.

$\mathbb{E}[\boldsymbol{f(S)}] \geq (\boldsymbol{1} - \boldsymbol{1}/\boldsymbol{e} - \boldsymbol{\epsilon})\boldsymbol{f(OPT)}$

Rounding

$\boldsymbol{O_\epsilon(r^2)}$ **queries** [Chekuri-Vondrák-Zenklusen 2010]

Fast Continuous Greedy Algorithm

Discrete

max $f(S)$ s.t. $S \in \mathcal{I}$

Continuous

max $F(x)$ s.t. $x \in \mathcal{B}(\mathcal{M})$

Continuous Greedy Alg

$\tilde{O}_\epsilon(\sqrt{r}n)$ **queries**

[Buchbinder-Feldman-Schwartz 2015]

$x \in \mathcal{B}(\mathcal{M})$

s.t.

$\mathbb{E}[F(x)] \geq (1 - 1/e - \epsilon) f(OPT)$

$S \in \mathcal{I}$

s.t.

$\mathbb{E}[f(S)] \geq (1 - 1/e - \epsilon) f(OPT)$

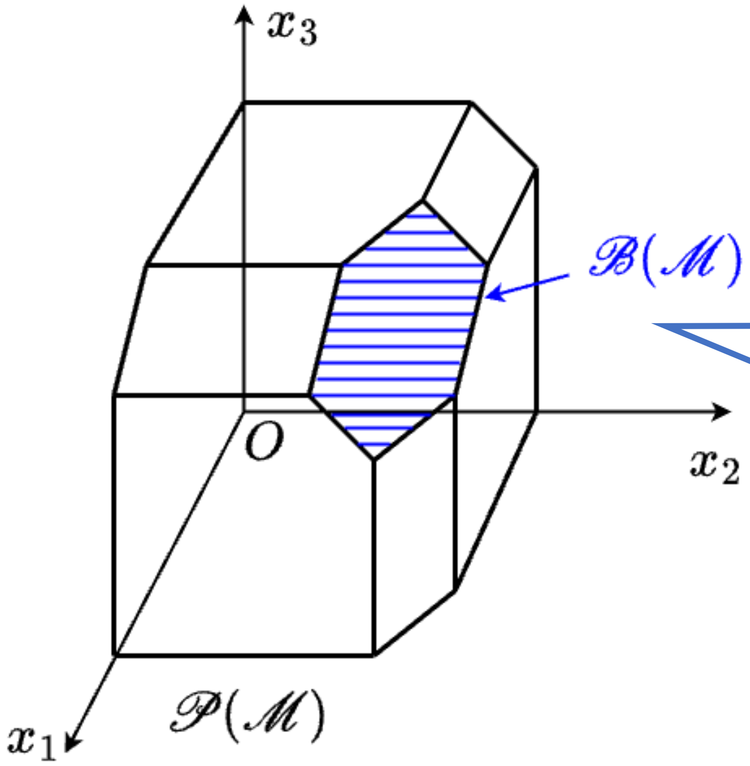
Rounding

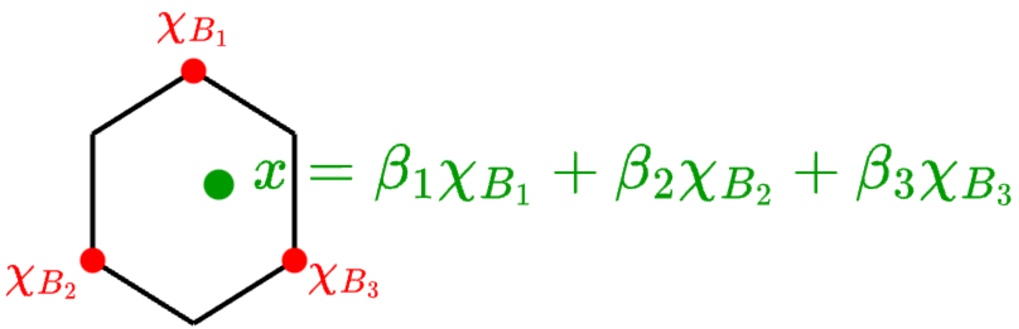
$\tilde{O}_\epsilon(r^{3/2})$ **queries** [This work]

Swap Rounding Algorithm [Chekuri-Vondrák-Zenklusen 2010]

$\boldsymbol{x} = \boldsymbol{\beta_1 \chi_{B_1}} + \cdots + \boldsymbol{\beta_t \chi_{B_t}}$

Input: $x \in \mathcal{B}(\mathcal{M})$ represented as a convex combination of t bases



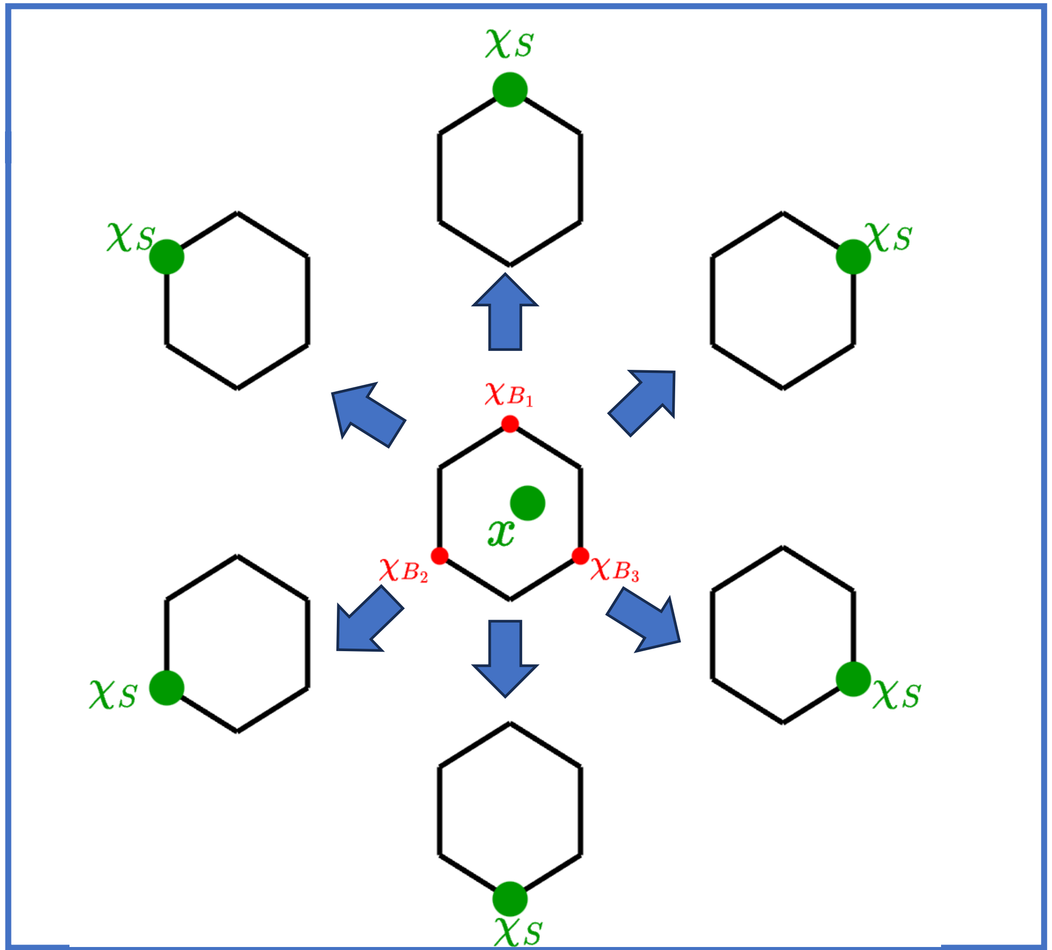


Swap Rounding Algorithm [Chekuri-Vondrák-Zenklusen 2010]

Input: $x \in \mathcal{B}(\mathcal{M})$ represented as a convex combination of t bases

Output: basis S s.t. $\mathbb{E}[\boldsymbol{F}(\boldsymbol{\chi_S})] \geq \boldsymbol{F}(\boldsymbol{x})$ for any submodular function f

$\boldsymbol{F}(\boldsymbol{\chi_S}) = \boldsymbol{f}(\boldsymbol{S})$



Fast Rounding Algorithm

Input: $x \in \mathcal{B}(\mathcal{M})$ represented as a convex combination of t bases
Output: basis S s.t. $\mathbb{E}[\boldsymbol{f(S)}] \geq \boldsymbol{(1-\epsilon)F(x)}$ for any submodular function f

Thm [Chekuri-Vondrák-Zenklusen 2010]
Algorithm using $\boldsymbol{O(r^2 t)}$ independence queries

Thm [This work]
Algorithm using $\boldsymbol{\tilde{O}_\epsilon(r^{3/2} t)}$ independence queries

Swap Rounding Algorithm of [Chekuri-Vondrák-Zenklusen 2010]

SwapRound($\boldsymbol{x} = \boldsymbol{\beta_1 \chi_{B_1}} + \cdots + \boldsymbol{\beta_t \chi_{B_t}}$)

$C_1 \leftarrow B_1, \gamma_1 \leftarrow \beta_1$

For $i = 1, \ldots t - 1$:

$C_{i+1} \leftarrow$ **MergeBases**$(\gamma_i, C_i, \beta_{i+1}, B_{i+1})$

$\gamma_{i+1} \leftarrow \gamma_i + \beta_{i+1}$

Return C_t

Swap Rounding Algorithm of [Chekuri-Vondrák-Zenklusen 2010]

SwapRound($\boldsymbol{x} = \boldsymbol{\beta_1 \chi_{B_1}} + \cdots + \boldsymbol{\beta_t \chi_{B_t}}$)

  $C_1 \leftarrow B_1, \gamma_1 \leftarrow \beta_1$

  For $i = 1, \ldots t - 1$:

    $C_{i+1} \leftarrow$ **MergeBases**$(\gamma_i, C_i, \beta_{i+1}, B_{i+1})$

    $\gamma_{i+1} \leftarrow \gamma_i + \beta_{i+1}$

  Return C_t

MergeBases$(\beta_1, B_1, \beta_2, B_2)$

  While $B_1 \neq B_2$:

    (Step1) Find v, u such that $B_1 + v - u \in \mathcal{I}$ and $B_2 + u - v \in \mathcal{I}$

    (Step2) $B_1 \leftarrow B_1 + v - u$ w.p. $\beta_2/(\beta_1 + \beta_2)$

    $B_2 \leftarrow B_2 + u - v$ w.p. $\beta_1/(\beta_1 + \beta_2)$

Swap Rounding Algorithm of [Chekuri-Vondrák-Zenklusen 2010]

SwapRound($\boldsymbol{x} = \boldsymbol{\beta_1 \chi_{B_1}} + \cdots + \boldsymbol{\beta_t \chi_{B_t}}$)

$C_1 \leftarrow B_1, \gamma_1 \leftarrow \beta_1$

For $i = 1, \ldots t - 1$:

$C_{i+1} \leftarrow$ **MergeBases**$(\gamma_i, C_i, \beta_{i+1}, B_{i+1})$

$\gamma_{i+1} \leftarrow \gamma_i + \beta_{i+1}$

Return C_t

MergeBases$(\beta_1, B_1, \beta_2, B_2)$

While $B_1 \neq B_2$:

(Step1) **Find $\boldsymbol{v}, \boldsymbol{u}$ such that $\boldsymbol{B_1 + v - u \in \mathcal{I}}$ and $\boldsymbol{B_2 + u - v \in \mathcal{I}}$**

(Step2) $B_1 \leftarrow B_1 + v - u$ w.p. $\beta_2/(\beta_1 + \beta_2)$

$B_2 \leftarrow B_2 + u - v$ w.p. $\beta_1/(\beta_1 + \beta_2)$

$\boldsymbol{O(r)}$ queries

Swap Rounding Algorithm of [Chekuri-Vondrák-Zenklusen 2010]

Thm [Chekuri-Vondrák-Zenklusen 2010]
Use $\boldsymbol{O(r^2 t)}$ independence queries

SwapRound($\boldsymbol{x = \beta_1 \chi_{B_1} + \cdots + \beta_t \chi_{B_t}}$)
  $C_1 \leftarrow B_1, \gamma_1 \leftarrow \beta_1$
  For $i = 1, \ldots t - 1$:
    $C_{i+1} \leftarrow$ **MergeBases**$(\gamma_i, C_i, \beta_{i+1}, B_{i+1})$
    $\gamma_{i+1} \leftarrow \gamma_i + \beta_{i+1}$
  Return C_t

MergeBases$(\beta_1, B_1, \beta_2, B_2)$
  While $B_1 \neq B_2$:
    (Step1) **Find $\boldsymbol{v, u}$ such that $\boldsymbol{B_1 + v - u \in \mathcal{I}}$ and $\boldsymbol{B_2 + u - v \in \mathcal{I}}$**
    (Step2) $B_1 \leftarrow B_1 + v - u$ w.p. $\beta_2/(\beta_1 + \beta_2)$
        $B_2 \leftarrow B_2 + u - v$ w.p. $\beta_1/(\beta_1 + \beta_2)$

$\boldsymbol{O(r)}$ queries

Fact: Step1 is executed $\boldsymbol{O(rt)}$ times

Fact: $\mathbb{E}[F(x)]$ does not decrease in Step2

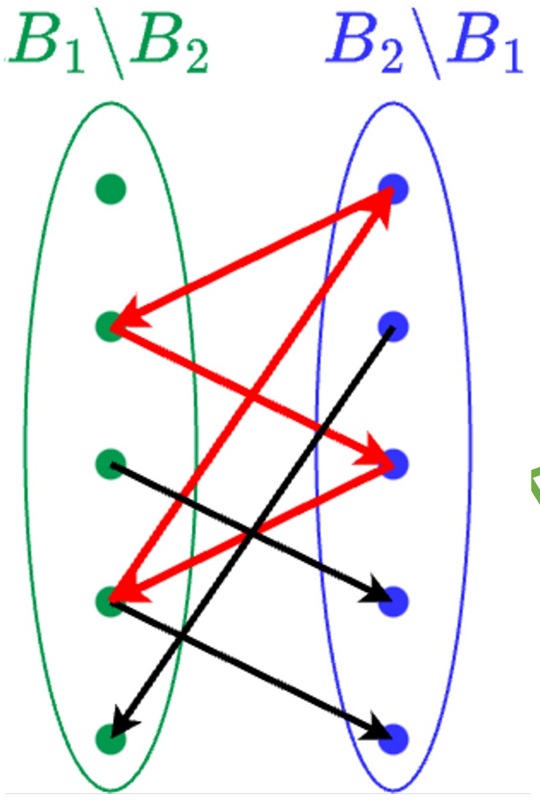
Contribution 1: Use of a dicycle of arbitrary length

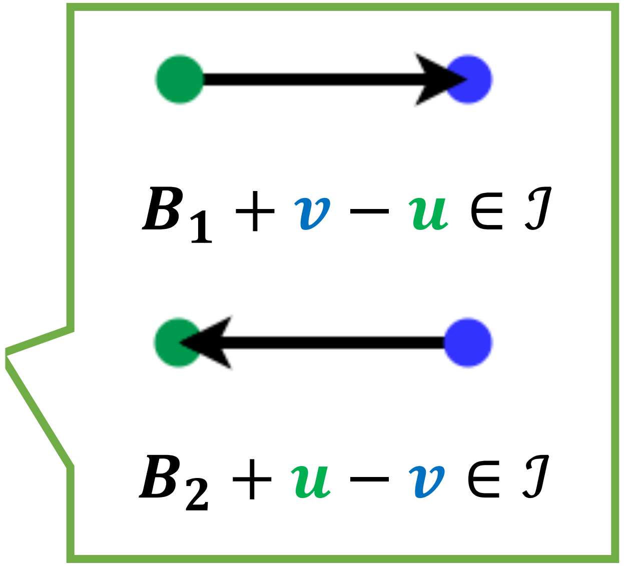
- [Chekuri-Vondrák-Zenklusen 2010]

Use of v, u such that $B_1 + v - u \in \mathcal{J}$ and $B_2 + u - v \in \mathcal{J}$

- [This work]

Use of a **dicycle** of arbitrary length





Contribution 1: Use of a dicycle of arbitrary length

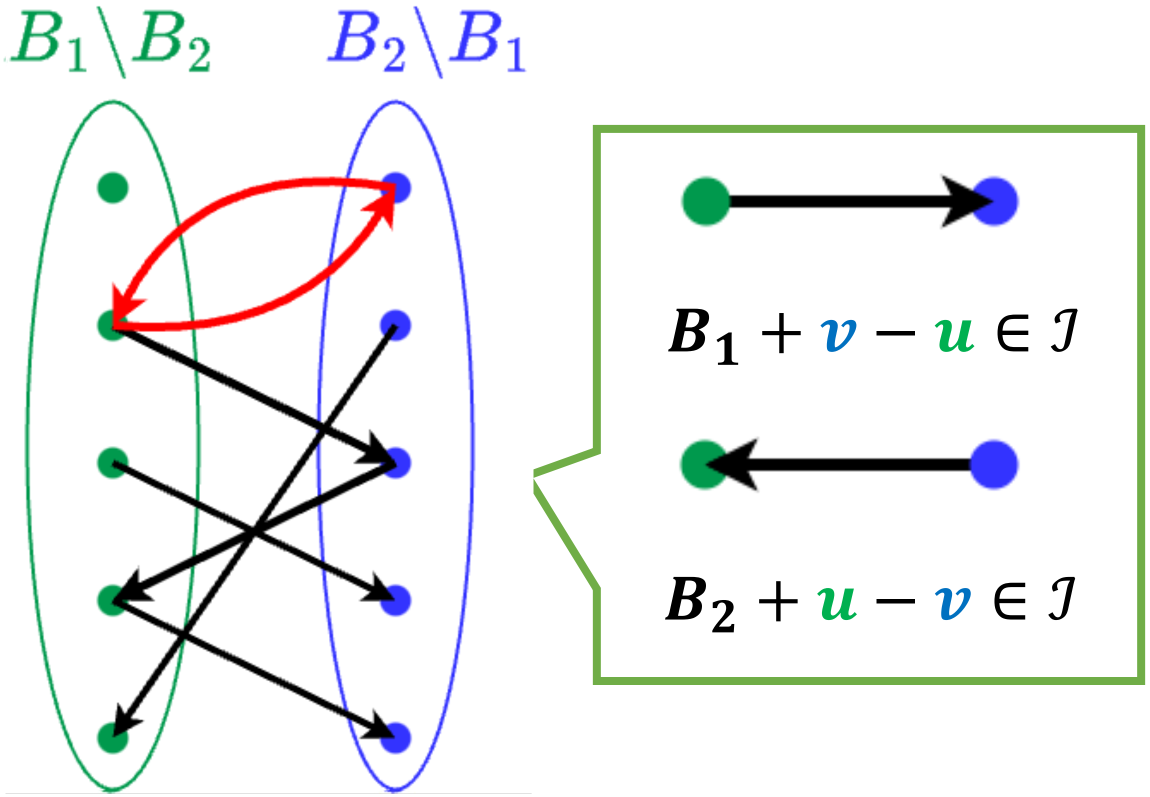
- [Chekuri-Vondrák-Zenklusen 2010]

Use of v, u such that $B_1 + v - u \in \mathcal{I}$ and $B_2 + u - v \in \mathcal{I}$

dicycle of length **two** (bidirected edge)

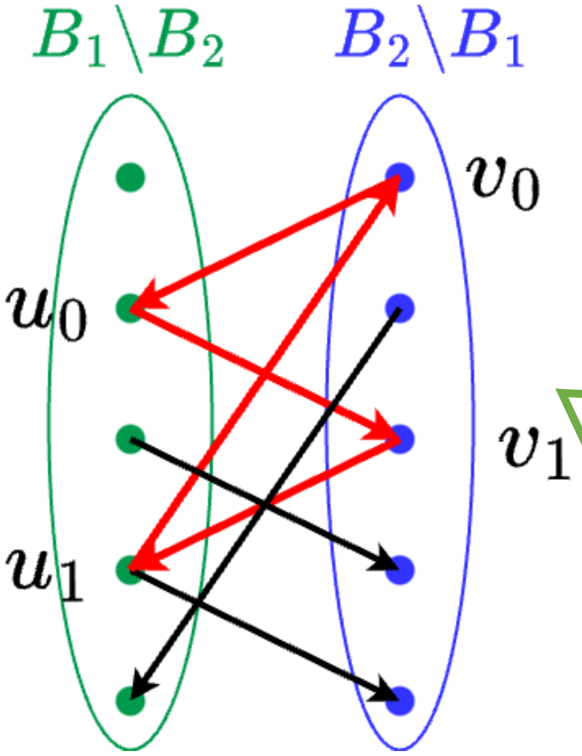
- [This work]

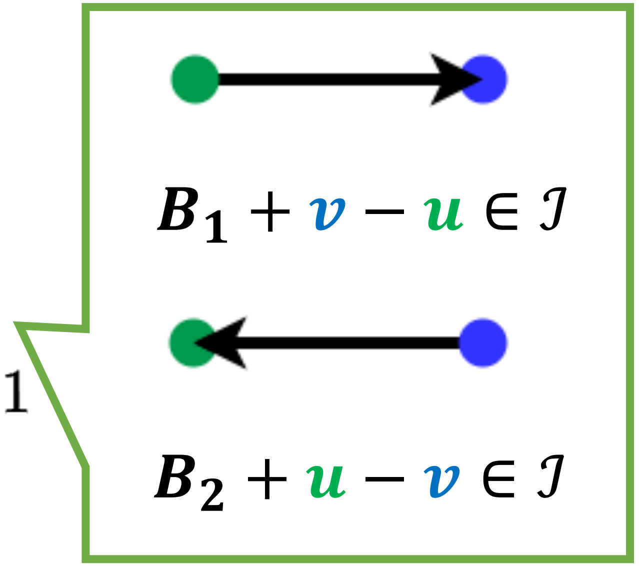
Use of a **dicycle** of arbitrary length



Contribution 1: Update via dicycle

w.p. $\beta_2/(\beta_1+\beta_2)$:
  Update B_1
w.p. $\beta_1/(\beta_1+\beta_2)$:
  Update B_2





Contribution 1: Update via dicycle

w.p. $\beta_2/(\beta_1+\beta_2)$:

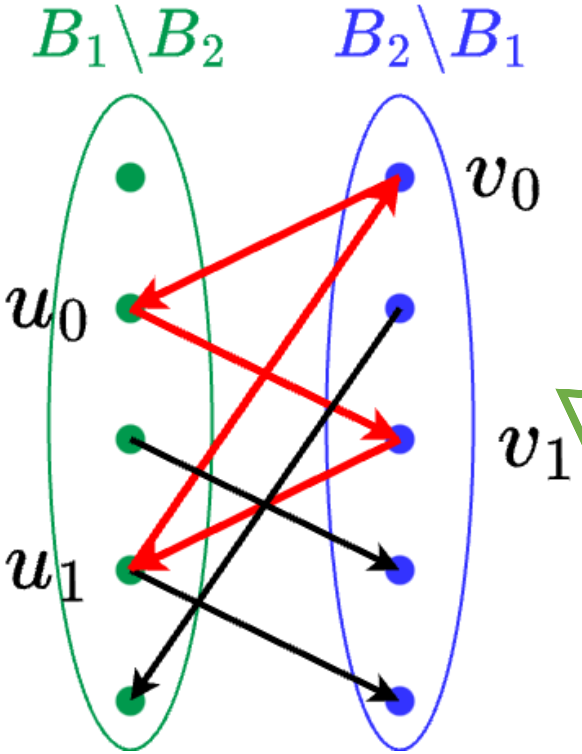
Update B_1

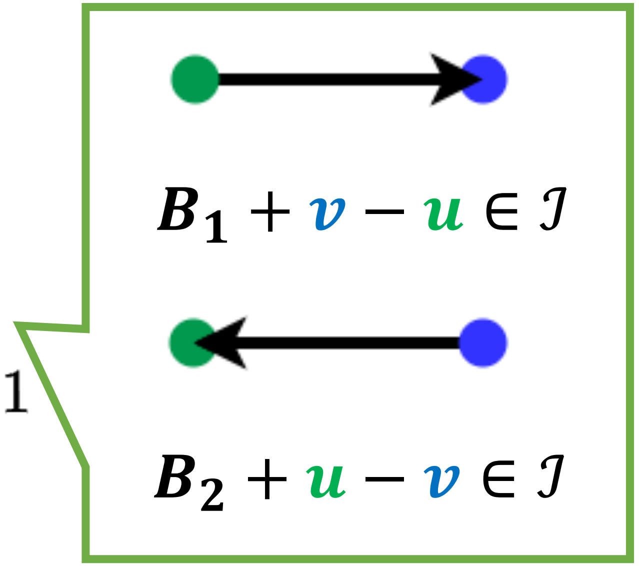
w.p. $\beta_1/(\beta_1+\beta_2)$:

Update B_2

Pick i unif. at random from $\{0,1\}$

$B_2 \leftarrow B_2 + u_{i+1} - v_i$

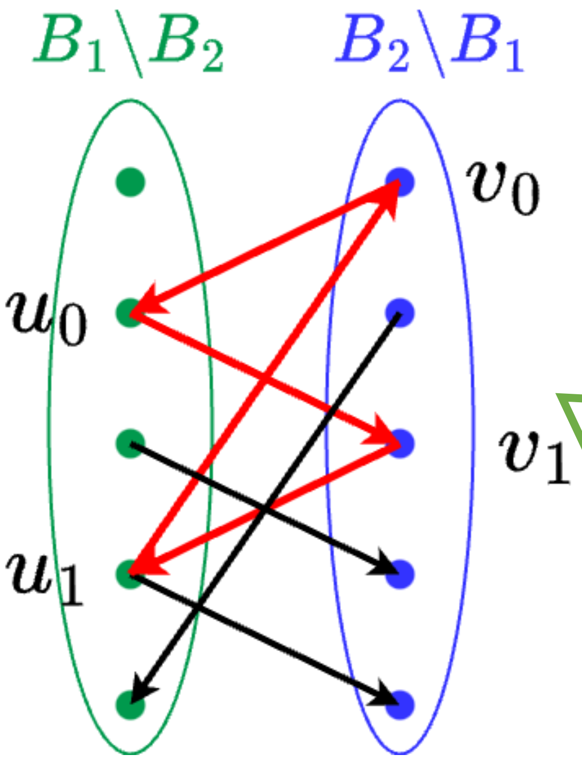


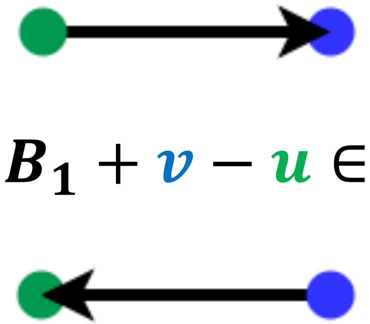


Contribution 1: Update via dicycle

w.p. $\beta_2/(\beta_1+\beta_2)$:
  Update B_1
w.p. $\beta_1/(\beta_1+\beta_2)$:
  Update B_2

Pick i unif. at random from $\{0,1\}$
$B_2 \leftarrow B_2 + u_{i+1} - v_i$



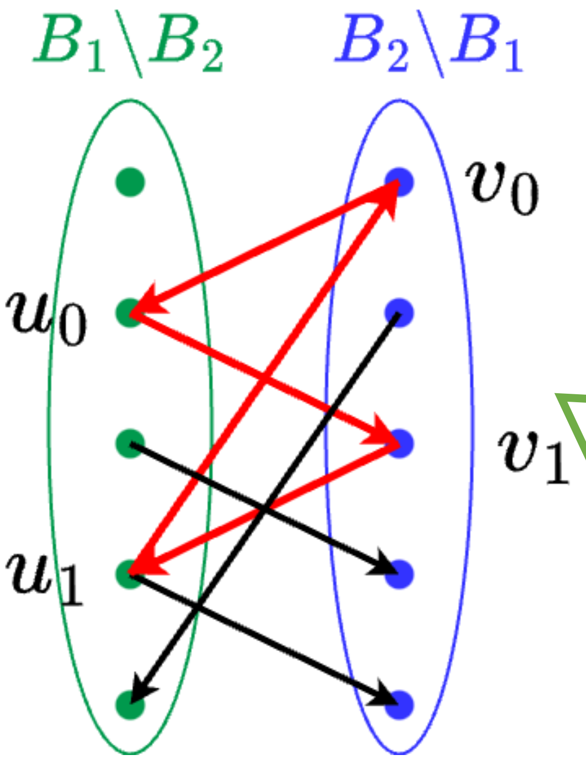


Lem: $\mathbb{E}\left[\boldsymbol{\beta_1 \chi_{B_1^{\mathrm{new}}}} + \boldsymbol{\beta_2 \chi_{B_2^{\mathrm{new}}}}\right] = \boldsymbol{\beta_1 \chi_{B_1^{\mathrm{old}}}} + \boldsymbol{\beta_2 \chi_{B_2^{\mathrm{old}}}}$

Contribution 1: Update via dicycle

w.p. $\beta_2/(\beta_1+\beta_2)$:
  Update B_1
w.p. $\beta_1/(\beta_1+\beta_2)$:
  Update B_2

Pick i unif. at random from $\{0,1\}$
$B_2 \leftarrow B_2 + u_{i+1} - v_i$



$\boldsymbol{B_1} + \boldsymbol{v} - \boldsymbol{u} \in \mathcal{I}$

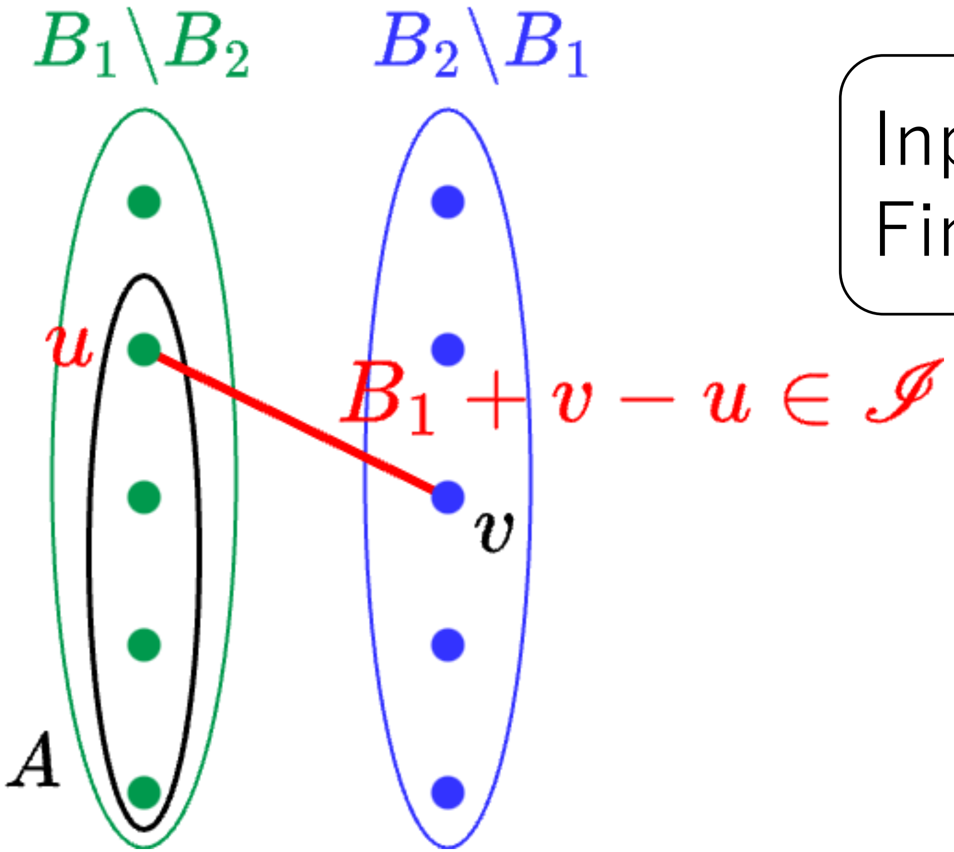
$\boldsymbol{B_2} + \boldsymbol{u} - \boldsymbol{v} \in \mathcal{I}$

$$\mathbb{E}[\boldsymbol{F}(\boldsymbol{x}^{\mathbf{new}})] \geq \boldsymbol{F}(\boldsymbol{x}^{\mathbf{old}})$$

Lem: $\mathbb{E}\left[\boldsymbol{\beta}_1 \boldsymbol{\chi}_{\boldsymbol{B}_1^{\mathbf{new}}} + \boldsymbol{\beta}_2 \boldsymbol{\chi}_{\boldsymbol{B}_2^{\mathbf{new}}}\right] = \boldsymbol{\beta}_1 \boldsymbol{\chi}_{\boldsymbol{B}_1^{\mathbf{old}}} + \boldsymbol{\beta}_2 \boldsymbol{\chi}_{\boldsymbol{B}_2^{\mathbf{old}}}$

Tool for Fast Matroid Intersection

[Nguyễn 2019, Chakrabarty-Lee-Sidford-Singla-Wong 2019]



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

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

Contribution 2: Fast Algorithm to find a dicycle

Lem.

We can find a dicycle using $\tilde{O}(\sqrt{r})$ **independence** queries **w.h.p.**

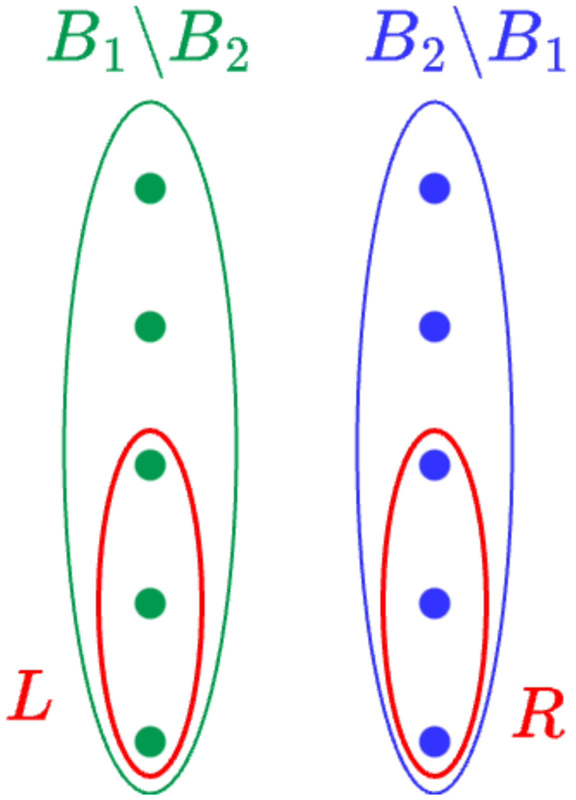
Contribution 2: Fast Algorithm to find a dicycle

Lem.

We can find a dicycle using $\tilde{O}(\sqrt{r})$ **independence** queries **w.h.p.**

Find a dicycle

① **Sample L (resp., R) of $\tilde{O}(\sqrt{r})$ elements from $B_1 \setminus B_2$ (resp., $B_2 \setminus B_1$)**



Contribution 2: Fast Algorithm to find a dicycle

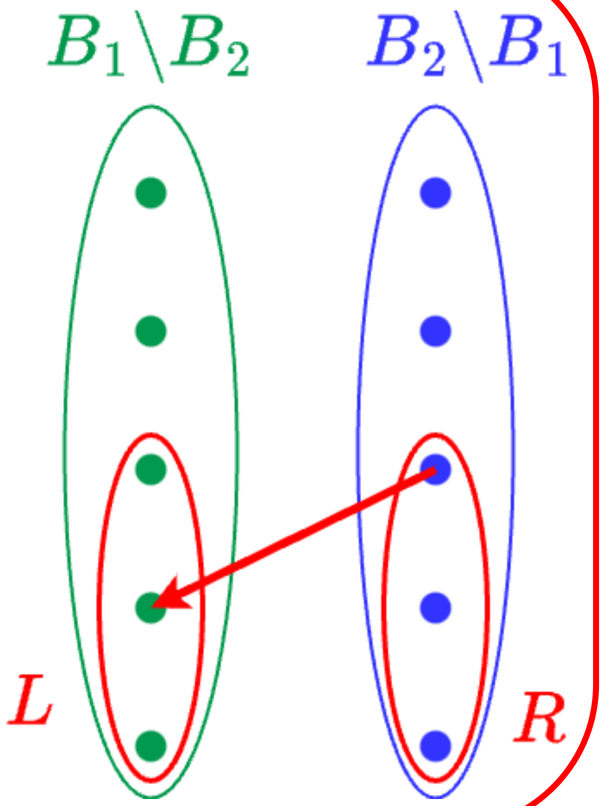
Lem.

We can find a dicycle using $\tilde{O}(\sqrt{r})$ **independence** queries **w.h.p.**

Find a dicycle

① Sample L (resp., R) of $\tilde{O}(\sqrt{r})$ elements from $B_1 \setminus B_2$ (resp., $B_2 \setminus B_1$)

② **Find a directed edge to each vertex in $G[L \cup R]$**



Contribution 2: Fast Algorithm to find a dicycle

Lem.

We can find a dicycle using $\tilde{O}(\sqrt{r})$ **independence** queries **w.h.p.**

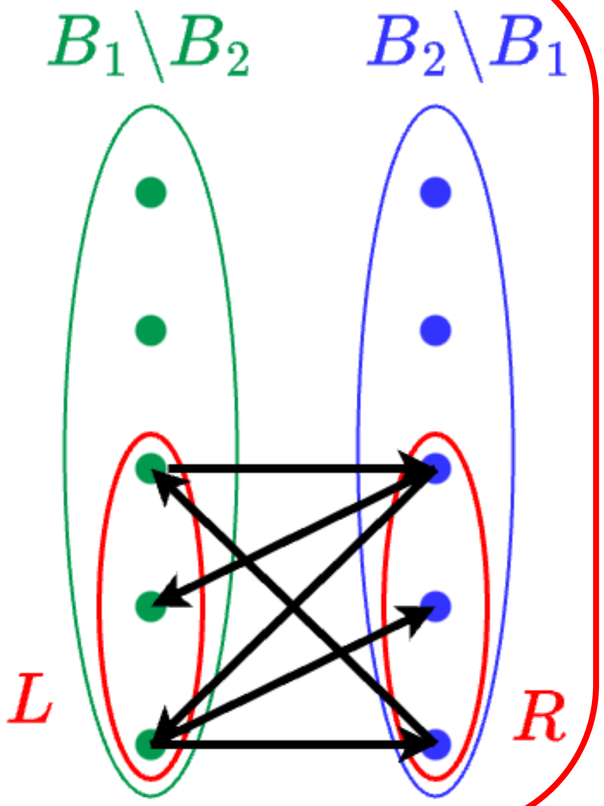
Find a dicycle

① Sample L (res

$\tilde{O}(\sqrt{r})$ queries by **binary search**

$B_1 \setminus B_2$ (resp., $B_2 \setminus B_1$)

② **Find a directed edge to each vertex in $G[L \cup R]$**



Contribution 2: Fast Algorithm to find a dicycle

Lem.

We can find a dicycle using $\tilde{O}(\sqrt{r})$ **independence** queries **w.h.p.**

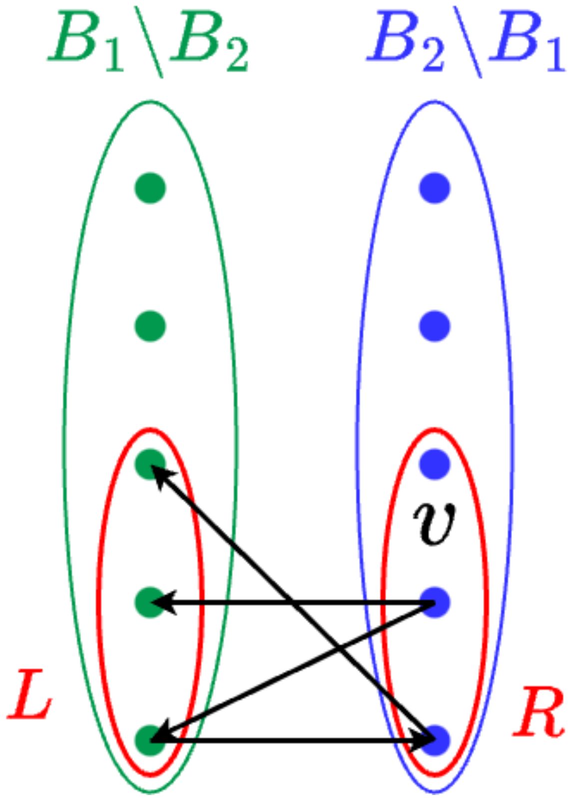
Find a dicycle

① Sample L (resp., R) of $\tilde{O}(\sqrt{r})$ elements from $B_1 \setminus B_2$ (resp., $B_2 \setminus B_1$)

② Find a directed edge to each vertex in $G[L \cup R]$

③ **If (indegree of $v \in L \cup R$ in $G[L \cup R]) = 0$**

Then Find a bidirected edge to v in G



Contribution 2: Fast Algorithm to find a dicycle

Lem.

We can find a dicycle using $\tilde{O}(\sqrt{r})$ **independence** queries **w.h.p.**

Find a dicycle

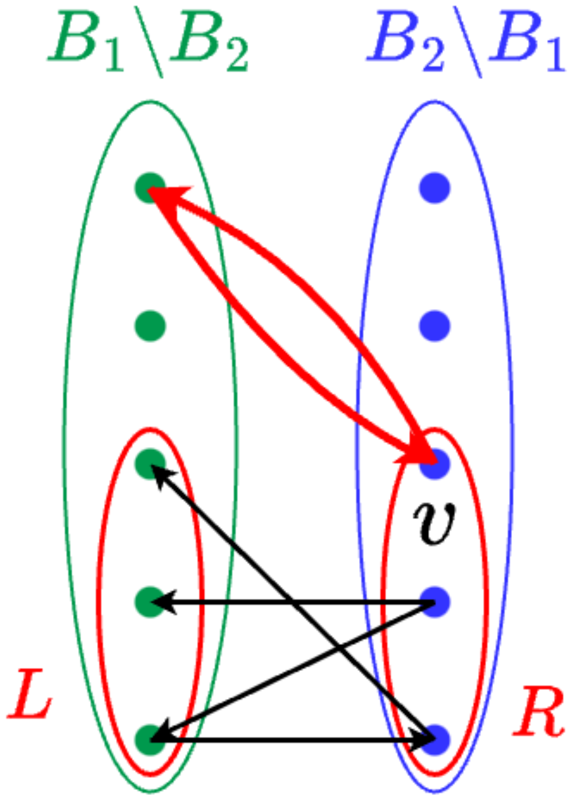
① Sample L (resp., R) of $\tilde{O}(\sqrt{r})$ elements from $B_1 \setminus B_2$ (resp., $B_2 \setminus B_1$)

② Find a directed edge to each vertex in $G[L \cup R]$

③ **If** (indegree of $v \in L \cup R$ in $G[L \cup R]$) $= 0$

Then Find a bidirected edge to v in G

Lem. $\tilde{O}(\sqrt{r})$ queries **w.h.p.** by **binary search**



Contribution 2: Fast Algorithm to find a dicycle

Lem.

We can find a dicycle using $\tilde{O}(\sqrt{r})$ **independence** queries **w.h.p.**

Find a dicycle

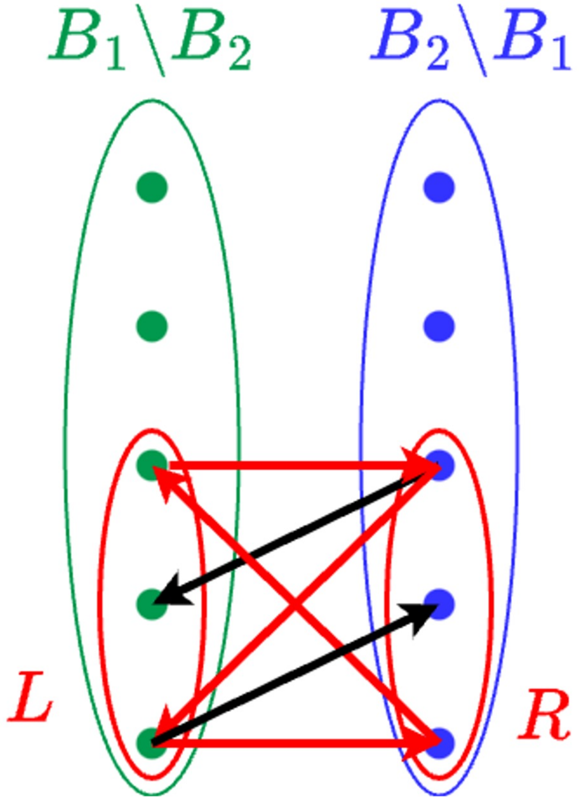
① Sample L (resp., R) of $\tilde{O}(\sqrt{r})$ elements from $B_1 \setminus B_2$ (resp., $B_2 \setminus B_1$)

② Find a directed edge to each vertex in $G[L \cup R]$

③ **If** (indegree of $v \in L \cup R$ in $G[L \cup R]$) $= 0$

Then Find a bidirected edge to v in G

Else There is a dicycle in $G[L \cup R]$



Conclusion

- Improvement on the query complexity of **Monotone Submodular Maximization with a Matroid Constraint**
- Rounding Algorithm faster than [Chekuri-Vondrák-Zenklusen 2010]

- **Use of a dicycle of arbitrary length**
- **Fast algorithm to find a dicycle**

  ☞ Binary search technique used in fast matroid intersection

Q. Further improvement ?

cf.

- rank oracle queries : $\tilde{O}_\epsilon(n + r^{3/2})$ queries [This work]
- graphic matroid, partition matroid, etc. : nearly-linear time

[Ene-Nguyễn 2019, Henzinger-Liu-Vondrák-Zheng 2023]