Efficient learning in graphs and in combinatorial multi-armed bandits

Ruo-Chun Tzeng

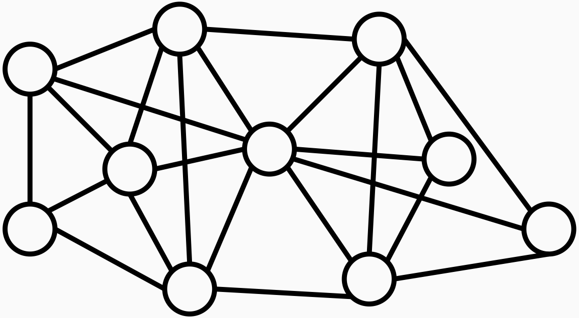
EECS, TCS division, KTH Royal Institue of Technology, Sweden

Included papers

- [TOG20] "Discovering conflicting groups in signed networks." In Proc. of NeurIPS 2020.
- [TWA$^+$22] "Improved analysis of randomized SVD for top-eigenvector approximation." In Proc. of AISTATS 2022.
- [TWPL23] "Closing the Computational-Statistical Gap in Best Arm Identification for Combinatorial Semi-bandits." In Proc. of NeurIPS 2023.
- [TOA24] "Matroid Semi-bandits in Sublinear Time." In Proc. of ICML 2024.

Introduction

Given a graph $G = (V, E)$ with $w : E \to \mathbb{R}$,



we'd like to solve a combinatorial optimization problem:

$$\max_{\boldsymbol{x} \in \mathcal{X}} f_w(\boldsymbol{x})$$

for some function f_w and some combinatorial set $\mathcal{X}$.

Part I - Graph mining

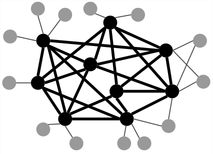
- the edge weight w is known
- $f_w(\boldsymbol{x})$ quantifies how much the solution $\boldsymbol{x}$ matches a specific pattern that we are searching for

We focus on graph mining tasks of the following form:

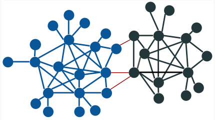
$$\max_{\boldsymbol{x} \in \mathcal{X}} \frac{\boldsymbol{x}^T \mathbf{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}.$$

Examples include:

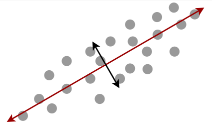
densest subgraph detection



2-community detection



PCA



Part I - Graph mining

- the edge weight w is known
- $f_w(\boldsymbol{x})$ quantifies how much the solution $\boldsymbol{x}$ matches a specific pattern that we are searching for

We focus on graph mining tasks of the following form:

$$\max_{\boldsymbol{x} \in \mathcal{X}} \frac{\boldsymbol{x}^T \mathbf{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}}.$$

Included papers:

- "Discovering conflicting groups in signed networks." In Proc. of NeurIPS 2020.
- "Improved analysis of randomized SVD for top-eigenvector approximation." In Proc. of AISTATS 2022.

Part II - Combinatorial multi-armed bandits

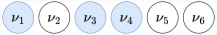
- the edge weight w is unknown and observed through stochastic samples
- $f_w(\boldsymbol{x})$ is the reward function whose dependency on w is known

We focus on stochastic semi-bandit with linear reward

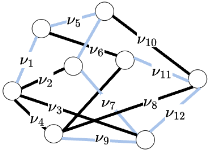
$$f_w(\boldsymbol{x}) = \sum_{k \in \mathrm{supp}(\boldsymbol{x})} w(k)$$

Applications include:

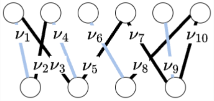
online ranking



network routing



crowdsourcing



Part II - Combinatorial multi-armed bandits

- the edge weight w is unknown and observed through stochastic samples
- $f_w(\boldsymbol{x})$ is the reward function whose dependency on w is known

We focus on stochastic semi-bandit with linear reward

$$f_w(\boldsymbol{x}) = \sum_{k \in \mathrm{supp}(\boldsymbol{x})} w(k)$$

Included papers:

- "Closing the Computational-Statistical Gap in Best Arm Identification for Combinatorial Semi-bandits." In Proc. of NeurIPS 2023.
- "Matroid Semi-bandits in Sublinear Time." In Proc. of ICML 2024.

Papers to be presented

- [TOG20] "Discovering conflicting groups in signed networks." In Proc. of NeurIPS 2020.
- [TWA$^+$22] "Improved analysis of randomized SVD for top-eigenvector approximation." In Proc. of AISTATS 2022.
- [TWPL23] "Closing the Computational-Statistical Gap in Best Arm Identification for Combinatorial Semi-bandits." In Proc. of NeurIPS 2023.
- [TOA24] "Matroid Semi-bandits in Sublinear Time." In Proc. of ICML 2024.

Part I - Graph mining: explicit structure mining

Included papers

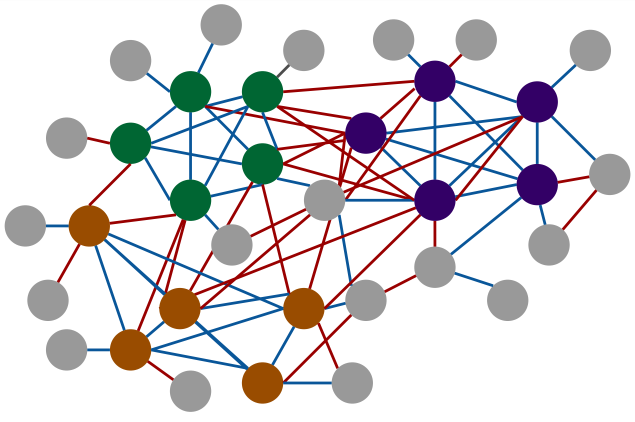
- [TOG20] "Discovering conflicting groups in signed networks." In Proc. of NeurIPS 2020.
- [TWA+22] "Improved analysis of randomized SVD for top-eigenvector approximation." In Proc. of AISTATS 2022.
- [TWPL23] "Closing the Computational-Statistical Gap in Best Arm Identification for Combinatorial Semi-bandits." In Proc. of NeurIPS 2023.
- [TOA24] "Matroid Semi-bandits in Sublinear Time." In Proc. of ICML 2024.

[TOG20]: one-page summary

- We show each conflicting groups is characterized by a maximum discrete Rayleigh's quotient (`MAX-DRQ`) problem
- We present an eigenvector-based algorithm with provable guarantee to the `MAX-DRQ` problem

[TOG20] Problem: k-conflicting groups detection

Given a signed graph, the goal is to find k disjoint groups $S_1, \cdots, S_k \subseteq V$ with mostly $+$ intra-group edges and mostly $-$ inter-group edges



Prior work: 2-conflicting group detection

- It has been studied by [BCMV12, BGG+19]

$$\max_{\substack{S_1,S_2 \subseteq V \\ S_1 \cap S_2 = \emptyset}} \frac{\sum_{\substack{(i,j)\in E(S_h) \\ h=1,2}} A_{i,j} + \sum_{(i,j)\in E(S_1,S_2)}(-A_{i,j})}{|S_1 \cup S_2|} = \max_{\boldsymbol{x}\in\{0,\pm 1\}^n \setminus \{\mathbf{0}_n\}} \frac{\boldsymbol{x}^T \mathbf{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}},$$

  where $\mathbf{A} \in \{0, \pm 1\}^{n \times n}$ is the signed adjacency matrix

  - $A_{i,j}x_i x_j = 1$: $+$ intra-group edges and $-$ inter-group edges
  - $A_{i,j}x_i x_j = -1$: $-$ intra-group edges and $+$ inter-group edges

Prior work: 2-conflicting group detection

- It has been studied by [BCMV12, BGG$^+$19]

$$\max_{\substack{S_1,S_2\subseteq V\\ S_1\cap S_2=\emptyset}} \frac{\sum_{\substack{(i,j)\in E(S_h)\\ h=1,2}} A_{i,j} + \sum_{(i,j)\in E(S_1,S_2)}(-A_{i,j})}{|S_1\cup S_2|} = \max_{\boldsymbol{x}\in\{0,\pm1\}^n\setminus\{\boldsymbol{0}_n\}} \frac{\boldsymbol{x}^T\mathbf{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}},$$

where $\mathbf{A}\in\{0,\pm1\}^{n\times n}$ is the signed adjacency matrix

  - [BCMV12, BGG$^+$19] showed the problem is APX-hard
  - $\mathcal{O}(\sqrt{n})$-approx eigenvector-based algorithm by [BGG$^+$19]
  - $\tilde{\mathcal{O}}(n^{\frac{1}{3}})$-approx SDP-based algorithm by [BCMV12]

Prior work: 2-conflicting group detection

- It has been studied by [BCMV12, BGG+19]

$$\max_{\substack{S_1,S_2\subseteq V\\ S_1\cap S_2=\emptyset}} \frac{\sum_{\substack{(i,j)\in E(S_h)\\ h=1,2}} A_{i,j} + \sum_{(i,j)\in E(S_1,S_2)}(-A_{i,j})}{|S_1\cup S_2|} = \max_{\mathbf{x}\in\{0,\pm1\}^n\setminus\{\mathbf{0}_n\}} \frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}},$$

  where $\mathbf{A}\in\{0,\pm1\}^{n\times n}$ is the signed adjacency matrix
  - [BCMV12, BGG+19] showed the problem is APX-hard
  - $\mathcal{O}(\sqrt{n})$-approx eigenvector-based algorithm by [BGG+19]
  - $\tilde{\mathcal{O}}(n^{\frac{1}{3}})$-approx SDP-based algorithm by [BCMV12]
- ($k\geq 2$) **Our contribution**: generalize the (i) problem formulation and (ii) algorithmic framework to the case $k\geq 2$

Problem formulation: k-conflicting group detection

- We propose the following problem formulation of k-conflicting group detection:

$$\max_{\substack{S_1,\cdots,S_k \subseteq V:\\ S_i \cap S_j = \emptyset, \forall i,j}} \frac{\sum_{\substack{(i,j)\in E(S_h,S_h)\\ h=1,\cdots,k}} A_{i,j} - \frac{1}{k-1} \sum_{h\neq\ell} \sum_{(i,j)\in E(S_h,S_\ell)} A_{i,j}}{\sum_{h\in[k]} |S_h|}.$$

- Because under (i) equally-sized groups and (ii) uniform edge density, # of inter-group edges $\approx (k-1) \times \#$ of intra-group edges.

An equivalent formulation by spectral graph theory

$$\max_{\mathbf{Y}\in\mathbb{R}^{n\times(k-1)}\setminus\{\mathbf{0}\}} \frac{\mathrm{Tr}(\mathbf{Y}^T\mathbf{A}\mathbf{Y})}{\mathrm{Tr}(\mathbf{Y}^T\mathbf{Y})} \tag{1}$$

subject to $Y_{i,j} = \begin{cases} c_j(k-j) & \text{if } i \in S_j \\ 0 & \text{if } i \in \cup_{h=1}^{j-1} S_h \text{ or } i \notin \cup_{h\in[k]} S_h \\ -c_j & \text{if } i \in \cup_{h=j+1}^{k} S_h \end{cases}$, where $\{c_j\}_{j\in[k-1]}$ are constants.

An equivalent formulation by spectral graph theory

$$\max_{\mathbf{Y}\in\mathbb{R}^{n\times(k-1)}\setminus\{\mathbf{0}\}} \frac{\mathrm{Tr}(\mathbf{Y}^T\mathbf{A}\mathbf{Y})}{\mathrm{Tr}(\mathbf{Y}^T\mathbf{Y})} \tag{1}$$

subject to $Y_{i,j} = \begin{cases} c_j(k-j) & \text{if } i \in S_j \\ 0 & \text{if } i \in \cup_{h=1}^{j-1} S_h \text{ or } i \notin \cup_{h\in[k]} S_h \\ -c_j & \text{if } i \in \cup_{h=j+1}^{k} S_h \end{cases}$, where $\{c_j\}_{j\in[k-1]}$ are constants.

- Fix $S_1, \cdots, S_{j-1}$. Find $S_j = \{i \in [n] : \mathbf{u}_i^\star = k-j\}$ by solving

$$\mathbf{u}^\star \in \operatorname{argmax}\left\{\frac{\mathbf{x}^T\mathbf{A}^{(j-1)}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} : \mathbf{x} \in \{-1, 0, k-j\}^n\setminus\{\mathbf{0}\}\right\}, \tag{2}$$

where $\mathbf{A}^{(j-1)}$ is the adj. matrix after removing $\cup_{h\in[j-1]} S_h$ and $\mathbf{A}^{(0)} = \mathbf{A}$.

Solving the maximum discrete Rayleigh's quotient problem

Fix $S_1, \cdots, S_{j-1}$. Find $S_j = \{i \in [n] : \mathbf{u}_i^\star = k - j\}$ by solving

$$\mathbf{u}^\star \in \operatorname{argmax} \left\{ \frac{\mathbf{x}^T \mathbf{A}^{(j-1)} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} : \mathbf{x} \in \{-1, 0, k-j\}^n \setminus \{\mathbf{0}\} \right\}, \qquad (2)$$

where $\mathbf{A}^{(j-1)}$ is the adj. matrix after removing $\cup_{h \in [j-1]} S_h$ and $\mathbf{A}^{(0)} = \mathbf{A}$.

- Eigenvector-based: let $\mathbf{u}$ be the leading eigenvector of $\mathbf{A}^{(j-1)}$.
  Randomized rounding: $\mathcal{O}((k-j)\sqrt{n})$-approx generalizes [BGG+19]

$$\tilde{\mathbf{u}}_i = \begin{cases} (k-j) \cdot \text{Bernoulli}(|\mathbf{u}_i|/(k-j)) & \text{if } \mathbf{u}_i > 0 \\ -1 \cdot \text{Bernoulli}(|\mathbf{u}_i|) & \text{if } \mathbf{u}_i < 0 \end{cases}$$

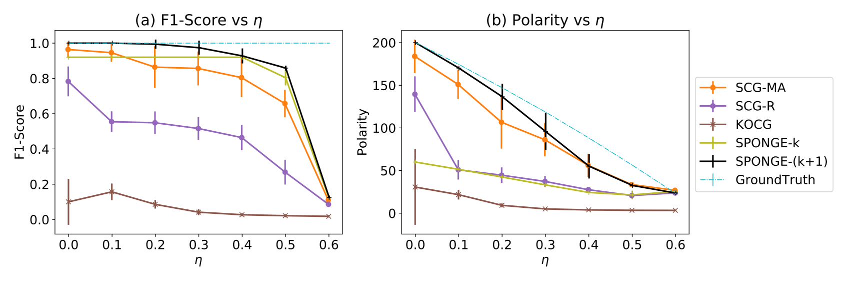
  [TOG20] There exists an instance such that the integrality gap of any eigenvector-based algorithm is $\Omega(\sqrt{n})$.

Experiment results

- Real-world networks:

	Bitcoin	WikiVote	Referendum	Slashdot	WikiConflict	Epinions	Wikipolitics
$\|V\|$	5 881	7 115	10 884	82 140	116 717	131 580	138 587
$\|E\|$	21 492	100 693	251 406	500 481	2 026 646	711 210	715 883
$\|E_-\|/\|E\|$	0.2	0.2	0.1	0.2	0.6	0.2	0.1
SCG-MA	**14.6**	**45.5**	**84.9**	**37.8**	**102.6**	**88.8**	**57.5**
SCG-R	5.0	9.7	39.8	7.3	16.2	39.4	5.5
KOCG [CWP+16]	4.4	5.5	8.8	2.6	4.5	8.7	4.8
SPONGE-k [CDGT19]	5.0	15.8	41.5	—	—	—	—
SPONGE-(k+1) [CDGT19]	0.8	1.0	1.0	—	—	—	—

- Synthetic:



Future works

- Can the approximation guarantee of (2) be improved?

$$\mathbf{u}^\star \in \operatorname{argmax}\left\{\frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}} : \mathbf{x} \in \{-1, 0, q\}^n \backslash \{\mathbf{0}\}\right\}. \tag{2}$$

- Is it possible to design an algorithm with provable guarantee to the k-conflicting group detection?
- What is the fundamental limit in detecting k-conflicting groups in the synthetic model?

Included papers

- [TOG20] "Discovering conflicting groups in signed networks." In Proc. of NeurIPS 2020.
- [TWA^{+}22] "Improved analysis of randomized SVD for top-eigenvector approximation." In Proc. of AISTATS 2022.
- [TWPL23] "Closing the Computational-Statistical Gap in Best Arm Identification for Combinatorial Semi-bandits." In Proc. of NeurIPS 2023.
- [TOA24] "Matroid Semi-bandits in Sublinear Time." In Proc. of ICML 2024.

[TWA$^+$22]: one-page summary

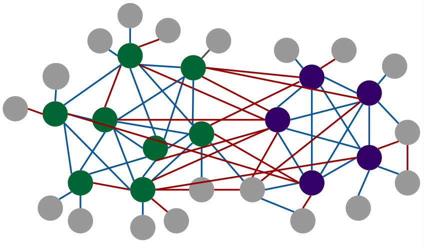
- Given a symmetric matrix $\mathbf{A}$, we study the leading eigenvector $\boldsymbol{u}$ returned by Randomized SVD (RSVD) w.r.t. $R(\boldsymbol{u}) = \frac{\boldsymbol{u}^T \mathbf{A} \boldsymbol{u}}{\lambda_1}$, where λ_1 is $\mathbf{A}$'s leading eigenvalue
- By relating $R(\boldsymbol{u})$ with random projection lemma, we sharpen the analysis of RSVD in the pass-efficient setting

Multiplicative analysis of RSVD using $R(\boldsymbol{u})$

- Given a symmetric matrix $\mathbf{A}$, let $\boldsymbol{u}$ be the leading eigenvector returned by RSVD using $\mathcal{O}(dn)$-memory and q passes over $\mathbf{A}$
- We analyze $\boldsymbol{u}$ w.r.t.

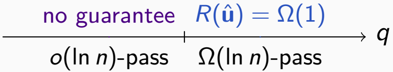
$$R(\boldsymbol{u}) = \lambda_1^{-1}(\mathbf{A}) \frac{\boldsymbol{u}^T \mathbf{A} \boldsymbol{u}}{\boldsymbol{u}^T \boldsymbol{u}}$$

- Motivation: the 2-conflicting-group-detection algorithm [BGG$^+$19] is $\mathcal{O}(R(\boldsymbol{u})^{-1}\sqrt{n})$-approx by rounding $\boldsymbol{u}$



Our contribution

- Prior to our [TWA+22]: for any $\mathbf{A} \succcurlyeq 0$, w.p. $\geq 1 - e^{-\Omega(d)}$,



Our contribution

- Prior to our [TWA+22]: for any $\mathbf{A} \succcurlyeq 0$, w.p. $\geq 1 - e^{-\Omega(d)}$,

  no guarantee | $R(\hat{\mathbf{u}}) = \Omega(1)$ → q

  $o(\ln n)$-pass | $\Omega(\ln n)$-pass

- Our contribution: for any $\mathbf{A} \succcurlyeq 0$, w.p. $\geq 1 - e^{-\Omega(d)}$,

  $R(\hat{\mathbf{u}}) = \Omega\left(\left(\frac{d}{n}\right)^{\frac{1}{2q+1}}\right)$ → q

  $o(\ln n)$-pass | $\Omega(\ln n)$-pass

  and for some indefinite $\mathbf{A}$, w.p. $\geq 1 - e^{-\Omega(\sqrt{d})}$,

  $R(\hat{\mathbf{u}}) = \Omega\left(\left(\frac{d}{n}\right)^{\frac{1}{2q+1}}\right)$ → q

  $o(\ln n)$-pass | $\Omega(\ln n)$-pass

Recap: Randomized SVD

Algorithm 1: RSVD $(\mathbf{A}, q, d)$

1 $\mathbf{Y} \leftarrow \mathbf{A}^q \mathbf{S}$ where $\mathbf{S} \sim \mathcal{N}(0,1)^{n \times d}$;
2 $\mathbf{Y} = \mathbf{QR}$;
3 $\mathbf{B} \leftarrow \mathbf{Q}^T \mathbf{AQ}$;
4 $\hat{\mathbf{u}} = \mathbf{Q}\, \mathbf{u}_1(\mathbf{B})$;
5 return $\hat{\mathbf{u}}$;

(Step 1: line 1) $\mathbf{Y}_{:,j} = \mathbf{A}^q \mathbf{S}_{:,j} = \sum_{i=1}^{n} \lambda_i^q (\mathbf{u}_i^T \mathbf{S}_{:,j}) \mathbf{u}_i, \ \forall j \in [d]$

(Step 2: line 2-4) $\hat{\mathbf{u}} = \operatorname{argmax}\{\mathbf{v}^T \mathbf{A} \mathbf{v} : \mathbf{v} \in \operatorname{range}(\mathbf{Y}) \cap \mathbb{S}^{n-1}\}$

Effect of $q \uparrow$: $\mathbf{Y}_{:,j}$ align more to eigenspace of λ_1

Effect of $d \uparrow$: (i) $\uparrow$ concentration around $\mathbb{E}[R(Y_{:,j})]$; (ii) $\uparrow \mathbb{E}[R(\hat{\mathbf{u}})]$

Main technique: random projection lemma

- We relate $R(\hat{\boldsymbol{u}})$ with random projection length by Cauchy-Schwarz inequality:

$$\underbrace{\max_{\mathbf{a}\in\mathbb{S}^{d-1}} \frac{\sum_{i\in[n]} \alpha_i^{2q+1} \langle \mathbf{S}^T \mathbf{u}_i, \mathbf{a}\rangle^2}{\sum_{i\in[n]} \alpha_i^{2q} \langle \mathbf{S}^T \mathbf{u}_i, \mathbf{a}\rangle^2}}_{R(\hat{\mathbf{u}})} \Longrightarrow \underbrace{\max_{\mathbf{a}\in\mathbb{S}^{d-1}} \frac{\langle \mathbf{S}^T \mathbf{u}_1, \mathbf{a}\rangle^2}{\sum_{i\in[n]} \langle \mathbf{S}^T \mathbf{u}_i, \mathbf{a}\rangle^2}}_{\cos^2\theta(\mathbf{u}_1, \mathbf{S})},$$

  where $\alpha_i = \frac{\lambda_i}{\lambda_1}, \forall i \in [n]$
- **(Lemma [Ver18])** $\forall \mathbf{v} \in \mathbb{S}^{n-1}$ and $d \ll n$, w.p. $\geq 1 - e^{-\Omega(d)}$,

$$\cos^2\theta(\mathbf{v}, \mathbf{S}) = \Theta\left(\frac{d}{n}\right).$$

Our results

(Theorem 1) $\forall \mathbf{A} \succcurlyeq 0$, $R(\hat{\mathbf{u}}) = \left(\Omega\left(\frac{d}{n}\right)\right)^{\frac{1}{2q+1}}$ w.p. $\geq 1 - e^{-\Omega(d)}$.

(Theorem 2) $\exists \mathbf{A} \succcurlyeq 0$ s.t. $R(\hat{\mathbf{u}}) = \mathcal{O}\left(\left(\frac{d}{n}\right)^{\frac{1}{2q+1}}\right)$ w.p. $\geq 1 - e^{-\Omega(d)}$.

(Theorem 3) $\forall \mathbf{A} \succcurlyeq 0$ with (i_0, γ)-power-law decay, $i_0 \in [n]$ and $\gamma > 1/2q$,

$$R(\hat{\mathbf{u}}) = \Omega\left(\left(\frac{d}{d + i_0}\right)^{\frac{1}{2q+1}}\right) \text{ w.p. } \geq 1 - e^{-\Omega(d)}.$$

(Assumption 1) $\exists \kappa \in (0, 1]$ s.t. $\sum_{i=2}^{n} \lambda_i^{2q+1} \geq \kappa \sum_{i=2}^{n} |\lambda_i|^{2q+1}$.

(Theorem 4) $\forall \mathbf{A}$ with (i_0, γ)-power-law decay, $i_0 \in [n]$ and $\gamma > 1/2q$, and satisfying Assumption 1, there exists a constant $c_\kappa > 0$ such that

$$R(\hat{\mathbf{u}}) = \Omega\left(c_\kappa \left(\frac{d}{d + i_0}\right)^{\frac{1}{2q+1}}\right) \text{ w.p. } \geq 1 - e^{-\Omega(\sqrt{d}\kappa^2)}.$$

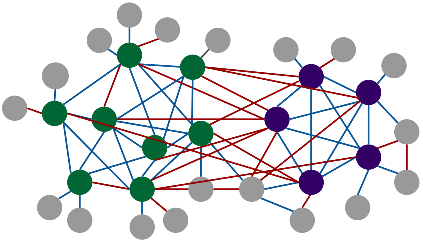
Algorithm 2: RandSum($\mathbf{A}, q, d, p$)

$\mathbf{S}_1 \sim \mathcal{N}(0,1)^{n\times\lceil\frac{d}{2}\rceil}$, $\mathbf{S}_2 \sim \text{Bernoulli}(p)^{n\times\lfloor\frac{d}{2}\rfloor}$;

$\mathbf{S} \leftarrow [\mathbf{S}_1 \quad \mathbf{S}_2]$;

return RSVD($\mathbf{A}$,$\mathbf{S}$,q,d);

- **Motivation:** Consider the case when $\langle \mathbf{u}_1, \mathbf{1}_n\rangle = \Theta(\sqrt{n})$:



RandSum: combining Gaussian with 0/1-Bernoulli

Algorithm 2: RandSum($\mathbf{A}, q, d, p$)

1 $\mathbf{S}_1 \sim \mathcal{N}(0,1)^{n \times \lceil \frac{d}{2} \rceil}$, $\mathbf{S}_2 \sim \text{Bernoulli}(p)^{n \times \lfloor \frac{d}{2} \rfloor}$;

2 $\mathbf{S} \leftarrow [\mathbf{S}_1 \quad \mathbf{S}_2]$;

3 return RSVD($\mathbf{A}$,$\mathbf{S}$,q,d);

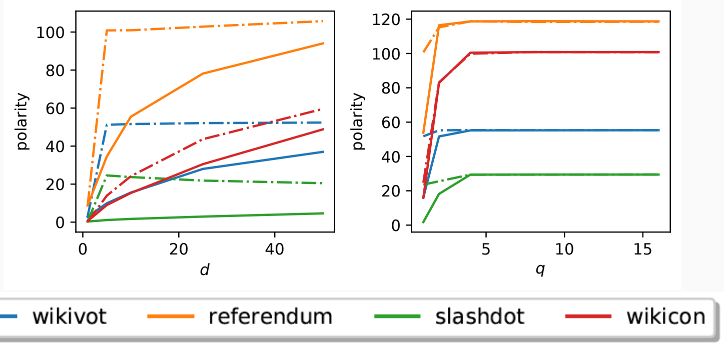
- **(Theorem 5)** For $\mathbf{A} \succcurlyeq 0$, RandSum($\mathbf{A}$,$q$,$d$,$p$) returns $\hat{\mathbf{u}}$ satisfying

$$R(\hat{\mathbf{u}}) = \left(\Omega \left(\frac{\max \left\{ d, \langle \mathbf{u}_1, \mathbf{1}_n \rangle^2 \right\}}{n} \right) \right)^{\frac{1}{2q+1}} \quad \text{with prob.} \quad \geq 1 - e^{-\Omega(d)}.$$

Theorem 5 extends to some indefinite **A** under an assumption similar to **Assumption 1**.

Experimental results

	WikiVot	Referendum	Slashdot	WikiCon
$\lvert V \rvert$	7 115	10 884	82 140	116 717
$\lvert E \rvert$	100 693	251 406	500 481	2 026 646
(γ, i_0)	(4.6, 15)	(4.5, 16)	(5.3, 17)	(2.8, 22)
κ	0.397	0.620	0.204	0.034
$\cos\theta(\mathbf{u}_1, \mathbf{1}_n)$	0.378	0.399	0.194	0.193



Future works

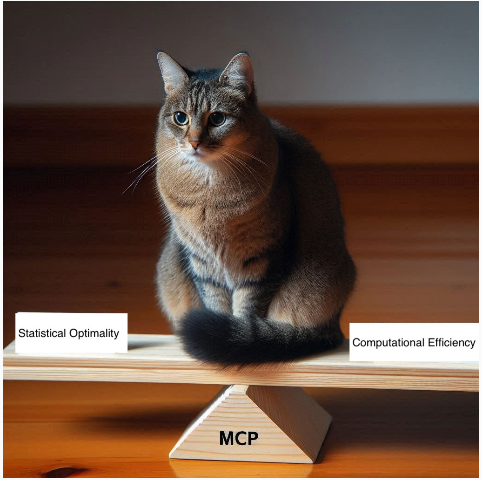
- Do the results generalize to (row/column)-stochastic matrices?
- Do the results of `RandSum` hold for other non-centered subgaussian distributions?
- Can we extend the analysis to top-k eigenvectors approximations?
- What is the fundamental limit of $R(\hat{\mathbf{u}})$ for any q-pass $\tilde{\mathcal{O}}(n)$-space algorithm?
- Can we reduce the space complexity while keeping the same guarantees?

Part II - Combinatorial MAB: implicit structure exploitation

Included papers

- [TOG20] "Discovering conflicting groups in signed networks." In Proc. of NeurIPS 2020.
- [TWA^{+}22] "Improved analysis of randomized SVD for top-eigenvector approximation." In Proc. of AISTATS 2022.
- [TWPL23] "Closing the Computational-Statistical Gap in Best Arm Identification for Combinatorial Semi-bandits." In Proc. of NeurIPS 2023.
- [TOA24] "Matroid Semi-bandits in Sublinear Time." In Proc. of ICML 2024.

[TWPL23]: one-page summary



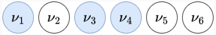
- We show there is no computational-statistical gap for combinatorial BAI with fixed confidence and semi-bandit feedback
- The main bottleneck is the computation for the most confusing parameter (MCP)
- We propose an approximate MCP algorithm and use it to design an optimal algorithm that runs in poly. time

Combinatorial semi-bandits

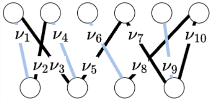
Given

- K arms $(\nu_k)_{k\in[K]}$ with mean $\boldsymbol{\mu} \in \mathbb{R}^K$
- Set $\mathcal{X} \subseteq \{0,1\}^K$ of actions

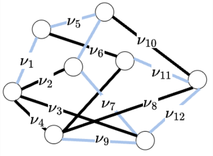
m-sets



matchings



spanning trees



In round t, a learner

- pulls action $\boldsymbol{x}(t) \in \mathcal{X}$
- receives the reward $y_k(t) \sim \nu_k(t)$ if $x_k(t) = 1$

Sequential sampling strategy: $\boldsymbol{x}(t) \in \mathcal{F}_t := \sigma(\boldsymbol{x}(1), \boldsymbol{y}(1), \ldots, \boldsymbol{x}(t-1), \boldsymbol{y}(t-1))$.

Best action identification with fixed confidence

Goal: identify the best action $\boldsymbol{i}^\star(\boldsymbol{\mu}) \in \text{argmax}_{x \in \mathcal{X}} \langle \boldsymbol{x}, \boldsymbol{\mu} \rangle$.

A strategy consist of

- (sampling rule) $\boldsymbol{x}(t) \in \mathcal{F}_t$ (action to explore)
- (stopping rule) τ (round to stop)
- (decision rule) $\mathcal{F}_\tau$-measurable $\hat{\boldsymbol{\imath}} \in \mathcal{X}$ (guess of best action to return)

Wish to minimize $\mathbb{E}_{\boldsymbol{\mu}}[\tau]$ subject to $\mathbb{P}_{\boldsymbol{\mu}}[\hat{\boldsymbol{\imath}} \neq \boldsymbol{i}^\star(\boldsymbol{\mu})] \leq \delta$.

Typical assumptions in combinatorial BAI

Throughout this work, we assume

(i) $\forall \boldsymbol{\mu} \in \Lambda$, $\boldsymbol{i}^\star(\boldsymbol{\mu})$ is unique, where Λ denotes all possible parameters.

(ii) $\forall k \in [K]$, arm k's reward distribution is $\nu_k = \mathcal{N}(\mu_k, 1)$.

(iii) $\forall \boldsymbol{v} \in \mathbb{R}^K$, $\boldsymbol{i}^\star(\boldsymbol{v})$ can be found in poly. time.

(iv) $\mathcal{X}$ is inclusion-wise maximal, i.e., no $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ such that $\boldsymbol{x} < \boldsymbol{x}'$

Here $\boldsymbol{i}^\star(\boldsymbol{v}) \in \operatorname{argmax}_{x \in \mathcal{X}} \langle \boldsymbol{x}, \boldsymbol{v} \rangle$.

A standard way to design a statistically optimal algorithm

[GK16] showed that: for any δ-PAC algorithm (stop in finite time a.e. and $\mathbb{P}_{\boldsymbol{\mu}}[\hat{\imath} \neq i^\star(\boldsymbol{\mu})] \leq \delta$),

$$\liminf_{\delta \to 0} \frac{\mathbb{E}_{\boldsymbol{\mu}}[\tau]}{\ln(1/\delta)} \geq \left(\sup_{\boldsymbol{\omega} \in \Sigma} \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{k=1}^{K} \frac{\omega_k (\mu_k - \lambda_k)^2}{2} \right)^{-1} . \tag{3}$$

- $\Sigma = \{\sum_{\boldsymbol{x} \in \mathcal{X}} w_{\boldsymbol{x}} \boldsymbol{x} : \boldsymbol{w} \in \Sigma_{|\mathcal{X}|}\}$: all possible arm allocations
- $\mathrm{Alt}(\boldsymbol{\mu}) = \{\boldsymbol{\lambda} \in \Lambda : \boldsymbol{i}^\star(\boldsymbol{\lambda}) \neq \boldsymbol{i}^\star(\boldsymbol{\mu})\}$: confusing parameters

(3)$\Rightarrow$ An optimal algorithm has a sampling strategy described by

$$\hat{\boldsymbol{\omega}}(t) \xrightarrow{t \to \infty} \boldsymbol{\omega}^\star(\boldsymbol{\mu}) := \arg\max_{\boldsymbol{\omega} \in \Sigma} F_{\boldsymbol{\mu}}(\boldsymbol{\omega}),$$

where $\hat{\boldsymbol{\omega}}(t)$ is the empirical allocation of arm draws up to round t, and

$$F_{\boldsymbol{\mu}}(\boldsymbol{\omega}) = \inf_{\lambda \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{k=1}^{K} \frac{\omega_k (\mu_k - \lambda_k)^2}{2} .$$

- most confusing parameter (MCP): $\arg\inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{k=1}^{K} \frac{\omega_k (\mu_k - \lambda_k)^2}{2}$

A standard way to design a statistically optimal algorithm

A standard way [GK16] consists of:

- (sampling rule) pull an action by tracking $\text{argmax}_{\boldsymbol{\omega} \in \Sigma} F_{\hat{\boldsymbol{\mu}}(t)}(\boldsymbol{\omega})$
- (stopping rule) Generalized Likelihood Ratio Test (GLRT):

$$\tau = \inf \left\{ t : tF_{\hat{\boldsymbol{\mu}}(t)}(\hat{\boldsymbol{\omega}}(t)) > \ln\left(\frac{t}{\delta}\right) + o(1) \right\} \tag{4}$$

- (decision rule) return $\hat{\boldsymbol{\imath}} \leftarrow \boldsymbol{i}^\star(\hat{\boldsymbol{\mu}}(\tau))$

where $\hat{\boldsymbol{\mu}}(t)$ is the estimate of $\boldsymbol{\mu}$, and $\hat{\boldsymbol{\omega}}(t)$ is the empirical pulled arm allocation up to t

Difficulty in identifying MCP (or equivalently $F_\cdot(\cdot)$)

Prior approach is to solve $|\mathcal{X}| - 1$ many convex programs by partitioning

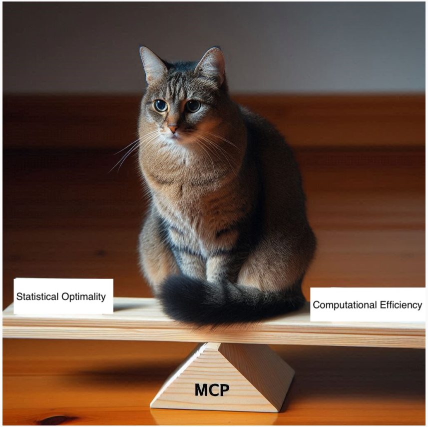
$\text{Alt}(\boldsymbol{\mu}) = \cup_{\boldsymbol{x} \neq \boldsymbol{i}^\star(\boldsymbol{\mu})} \{\boldsymbol{\lambda} \in \Lambda : \langle \boldsymbol{i}^\star(\boldsymbol{\mu}) - \boldsymbol{x}, \boldsymbol{\lambda} \rangle < 0\}$

However, $|\mathcal{X}| = \mathcal{O}(2^K)$!

Other statistically optimal sampling rules

In addition to the GLRT stopping rule, MCP is also required by

- FWS [WTP21]: apply the first-order methods to maximize $F_{\mu}(\omega)$, where the gradient is derived by a MCP (Envelop theorem), and enlarge the subsdifferential to cope with non-smoothness .
- CombGame [JMKK21]: need a MCP for chasing a saddle point.



Our Perturbed Frank-Wolfe Sampling (`P-FWS`)

- `P-FWS` copes with non-smoothness actions by *stochastic smoothing*.
- All `P-FWS` needs are
  (i) linear maximization oracle $\boldsymbol{i}^\star(\cdot)$ (standard),
  (ii) (ϵ, θ)-MCP, where $\epsilon > 0, \theta \in (0, 1)$ (established in this work).

Roadmap

1. Designing the (ϵ, θ)-MCP oracle
2. Solving lowerbound problem $\sup_{\boldsymbol{\omega} \in \Sigma} F_{\boldsymbol{\mu}}(\boldsymbol{\omega})$ by `P-FWS`

Let $\boldsymbol{x} \neq \boldsymbol{i}^\star(\boldsymbol{\mu})$, define $f_{\boldsymbol{x}}(\boldsymbol{\omega}, \boldsymbol{\mu}) = \inf_{\boldsymbol{\lambda} \in \mathbb{R}^K : \langle \boldsymbol{i}^\star(\boldsymbol{\mu}) - \boldsymbol{x}, \boldsymbol{\lambda} \rangle < 0} \sum_{k=1}^K \frac{\omega_k (\mu_k - \lambda_k)^2}{2}$.

Property of $f_{\boldsymbol{x}}$ and its Lagrangian dual $g_{\boldsymbol{\omega},\boldsymbol{\mu}}$

$$f_{\boldsymbol{x}}(\boldsymbol{\omega}, \boldsymbol{\mu}) = \max_{\alpha \geq 0} g_{\boldsymbol{\omega},\boldsymbol{\mu}}(\boldsymbol{x}, \alpha) \text{ (strong duality)}$$

$g_{\boldsymbol{\omega},\boldsymbol{\mu}}(\boldsymbol{x}, \alpha)$ is *linear* in $\boldsymbol{x}$ and *concave* in α

$$\Rightarrow F_{\boldsymbol{\mu}}(\boldsymbol{\omega}) = \min_{\boldsymbol{x} \neq \boldsymbol{i}^\star(\boldsymbol{\mu})} f_{\boldsymbol{x}}(\boldsymbol{\omega}, \boldsymbol{\mu}) = \min_{\boldsymbol{x} \neq \boldsymbol{i}^\star(\boldsymbol{\mu})} \max_{\alpha \geq 0} g_{\boldsymbol{\omega},\boldsymbol{\mu}}(\boldsymbol{x}, \alpha) \tag{5}$$

(5) motivates us a design:

- $\boldsymbol{x}$-player employs a regret min. algorithm to minimize $g_{\boldsymbol{\omega},\boldsymbol{\mu}}(\boldsymbol{x}, \alpha)$
- α-player employs a regret min. algorithm to maximize $g_{\boldsymbol{\omega},\boldsymbol{\mu}}(\boldsymbol{x}, \alpha)$

1. Designing the approximate MCP oracle

However, the choices are limited since...

1. $\{\boldsymbol{x} \neq \boldsymbol{i}^\star(\boldsymbol{\mu})\}$ is a discrete set which consist of $\mathcal{O}(2^K)$ elements
2. $\min\max = \max\min$ could not hold
3. we need the *equilibrium* action $\boldsymbol{x}_e$ s.t. $F_{\boldsymbol{\mu}}(\boldsymbol{\omega}) = \max_{\alpha \geq 0} g_{\boldsymbol{\omega},\boldsymbol{\mu}}(\boldsymbol{x}_e, \alpha)$
4. we want computationally efficient algorithms, i.e. approximate $\boldsymbol{x}_e$ in polynomial time

Algorithm 3: (ϵ, θ)-$\texttt{MCP}(\boldsymbol{\omega}, \boldsymbol{\mu})$

for $n = 1, 2, \cdots$ **do**

(Follow-the-Perturbed-Leader) $\mathcal{Z}_n \sim \exp(1)^K$ and $\eta_n = \frac{c_0}{\sqrt{n}}$

$$\boldsymbol{x}^{(n)} \in \operatorname*{argmin}_{\boldsymbol{x} \neq \boldsymbol{i}^\star(\boldsymbol{\mu})} \left(\sum_{m=1}^{n-1} g_{\boldsymbol{\omega},\boldsymbol{\mu}}(\boldsymbol{x}, \alpha^{(m)}) + \frac{\langle \mathcal{Z}_n, \boldsymbol{x} \rangle}{\eta_n} \right)$$

(Best-Response) $\alpha^{(n)} \in \operatorname*{argmax}_{\alpha \geq 0} g_{\boldsymbol{\omega},\boldsymbol{\mu}}(\boldsymbol{x}^{(n)}, \alpha)$

if $\sqrt{n} > \frac{c_\theta(1+\epsilon)}{\epsilon \hat{F}}$, *where* $\begin{cases} \hat{F} = g_{\boldsymbol{\omega},\boldsymbol{\mu}}(\boldsymbol{x}^{(n_\star)}, \alpha^{(n_\star)}) \\ n_\star \in \operatorname{argmin}_{m \leq n} g_{\boldsymbol{\omega},\boldsymbol{\mu}}(\boldsymbol{x}^{(m)}, \alpha^{(m)}) \end{cases}$ **then return** $(\hat{F}, \boldsymbol{x}^{(n_\star)})$;

end

1. Designing the approximate MCP oracle

Theorem 1 (MCP)

Let $(\boldsymbol{\omega}, \boldsymbol{\mu}) \in \Sigma_+ \times \Lambda$. The (ϵ, θ)-MCP$(\boldsymbol{\omega}, \boldsymbol{\mu})$ algorithm outputs $(\hat{F}, \hat{\boldsymbol{x}})$:

- $\mathbb{P}\left[F_{\boldsymbol{\mu}}(\boldsymbol{\omega}) \leq \hat{F} \leq (1+\epsilon) F_{\boldsymbol{\mu}}(\boldsymbol{\omega})\right] \geq 1 - \theta$
- the number of calls to $\boldsymbol{i}^\star(\cdot)$: $\mathcal{O}\left(\frac{\|\boldsymbol{\mu}\|_\infty^4 \left\|\boldsymbol{\omega}^{-1}\right\|_\infty^2 K^3 D^5 \ln K \ln \theta^{-1}}{\epsilon^2 F_{\boldsymbol{\mu}}(\boldsymbol{\omega})^2}\right)$

By envelop theorem [WTP21], we estimate the (sub)gradient of $F_{\boldsymbol{\mu}}(\boldsymbol{\omega})$ by

$$\nabla_{\boldsymbol{\omega}} f_{\hat{\boldsymbol{x}}}(\boldsymbol{\omega}, \boldsymbol{\mu}) = \left(\frac{(\mu_k - \lambda_k^\star)^2}{2}\right)_{k \in [K]},$$

where $\boldsymbol{\lambda}^\star$ is the minimizer to the optimization problem of $f_{\hat{\boldsymbol{x}}}(\boldsymbol{\omega}, \boldsymbol{\mu})$.

2. Solving the lowerbound problem $\max_{\omega \in \Sigma} F_\mu(\omega)$

$F_{\boldsymbol{\mu}}$ is nonsmooth $\Rightarrow$ using stochastic smoothing [FKM05, DBW12],
$\bar{F}_{\boldsymbol{\mu},\eta}(\boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{\mathcal{Z}} \sim \text{Uniform}(B_2)}[F_{\boldsymbol{\mu}}(\boldsymbol{\omega} + \eta \boldsymbol{\mathcal{Z}})]$ with noise level $\eta > 0$ satisfies:

- $\nabla \bar{F}_{\boldsymbol{\mu},\eta}(\boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{\mathcal{Z}} \sim \text{Uniform}(B_2)}[\nabla F_{\boldsymbol{\mu}}(\boldsymbol{\omega} + \eta \boldsymbol{\mathcal{Z}})]$
- $\bar{F}_{\boldsymbol{\mu},\eta}$ is $\frac{\ell K}{\eta}$-smooth and $\bar{F}_{\boldsymbol{\mu},\eta}(\boldsymbol{\omega}) \xrightarrow{\eta \downarrow 0} F_{\boldsymbol{\mu}}(\boldsymbol{\omega})$

High-level design of `P-FWS`

Let $\mathcal{X}_0$ be a set s.t. $\forall k \in [K]$, there exists $\boldsymbol{x} \in \mathcal{X}_0$ s.t. $x_k = 1$.

`P-FWS` alternate between two phases:

$$\begin{cases} \text{pull each } \boldsymbol{x} \in \mathcal{X}_0 \text{ once} & \text{(to avoid high cost and boundary cases)} \\ \text{pull } \boldsymbol{x}(t) \in \arg\max_{\boldsymbol{x} \in \mathcal{X}} \langle \nabla \bar{F}_{\hat{\boldsymbol{\mu}}(t-1),\eta_t}(\hat{\boldsymbol{\omega}}(t-1)), \boldsymbol{x} \rangle & \text{(ideal FW update)} \end{cases}$$

Theorem 2 (P-FWS)

Let $\boldsymbol{\mu} \in \Lambda$ and $\delta \in (0,1)$. `P-FWS` is δ-PAC, and

(i) $\limsup_{\delta \to 0} \frac{\mathbb{E}_{\boldsymbol{\mu}}[\tau]}{\ln(1/\delta)} \leq \left(\sup_{\boldsymbol{\omega} \in \Sigma} \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{k=1}^{K} \frac{\omega_k (\mu_k - \lambda_k)^2}{2} \right)^{-1}$

(ii) $\mathbb{E}_{\boldsymbol{\mu}}[\tau] = \mathrm{poly}\,(K)$ in moderate confidence regime

(iii) the expected number of calls for linear maximization is upper bounded by $\mathrm{poly}\,(K)$

Note: To our best knowledge, (ii) is even not yet established in unstructured BAI.

Conclusion and future works

- P-FWS closes the computational-statistical gap for combinatorial BAI by exploring the structural properties of the lowerbound problem, and it shows strong empirical performance:

Algorithm	Sample Complexity
P-FWS (ours)	1 176
CombGame [JMKK21]	1 277

Table 1: Averaged sample complexity at $\delta = 0.1$ over 100 independent runs on a graph with $|\mathcal{X}| = 21\,025$ spanning trees.

Conclusion and future works

- The computational-statistical gap widely exists for other tasks

Table 2: Computational-statistical gap in combinatorial semi-bandits.

	Reward Distribution	Statistical Optimality	Computational Efficiency & Statistical Optimality
Combinatorial BAI (semi-bandit)	Gaussian	[JMKK21, WTP21]	[TWPL23]
	Bernoulli		open
Combinatorial BAI (full-bandit)	Gaussian	[WTP21]	open
	Bernoulli		
Combinatorial Regret Minimization	Gaussian	[CMP17]	open
	Bernoulli		

Summary of the thesis

For graph mining,

- [TOG20]: solve a discrete Rayleigh's quotient maximization problem
- [TWA$^+$22]: sharpen the analysis of Randomized SVD for the ratio objective

For combinatorial semi-bandits,

- [TWPL23]: close the computational-statistical gap for combinatorial BAI with fixed-confidence
- [TOA24]: design the first sublinear-time algorithm for the matroid semi-bandits

[TOA24]: one-page summary

- We study the matroid semi-bandit problem and propose an algorithm that runs in sublinear time while matching the gap-dependent regret lower bound asymptotically
- The main technique is rounding and the minimum hitting set.

Matroid semi-bandits: problem setting and existing works

- Given an instance of matroid semi-bandit $([K], \mathcal{X}, \boldsymbol{\mu})$,
  - $[K] = \{1, \cdots, K\}$ is the ground set;
  - $\mathcal{X} \subseteq \{0,1\}^K$ is the set of bases of $\mathcal{M} = ([K], \mathcal{I})$[1];
  - $\boldsymbol{\mu} \in (0,1)^K$ is the mean of the arms' rewards $(\nu_1, \cdots, \nu_K)$;[2]
- At each round t, the learner pulls $\boldsymbol{x}(t) \in \mathcal{X}$ and observes $y_k(t) \sim \nu_k$ for each arm $k \in \operatorname{supp}(\boldsymbol{x}(t))$.
- The goal is to minimize the expected cumulative regret

$$R(T) = T \langle \boldsymbol{i}^\star, \boldsymbol{\mu} \rangle - \sum_{t=1}^{T} \mathbb{E}[\langle \boldsymbol{x}(t), \boldsymbol{\mu} \rangle],$$

where $\boldsymbol{i}^\star \in \operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}} \langle \boldsymbol{x}, \boldsymbol{\mu} \rangle$.

[1] $\mathcal{I}$ satisfy hereditary property and augmentation property.

[2] We assume for each $k \in [K]$, the support of ν_k is $[a, b] \subseteq (0,1)$.

Matroid semi-bandits: problem setting and existing works

- Define $\triangle_{\min} = \min_{i \in \mathrm{supp}(\boldsymbol{i}^\star), j \notin \mathrm{supp}(\boldsymbol{i}^\star): \mu_i - \mu_j > 0} (\mu_i - \mu_j)$.
  - CUCB [KWA+14] achieves $R(T) = \mathcal{O}(\frac{(K-D)\log T}{\triangle_{\min}})$, which matches the gap-dependent LB $R(T) = \Omega(\frac{(K-D)\log T}{\triangle_{\min}})$
  - KL-OSM [TP16] achieves $\limsup_{T\to\infty} \frac{R(T)}{\log T} \le c(\boldsymbol{\mu})$, matching instance-specific LB $\liminf_{T\to\infty} \frac{R(T)}{\log T} \ge c(\boldsymbol{\mu})$, where $c(\boldsymbol{\mu}) \le \frac{K-D}{\triangle_{\min}}$
- Per-round time complexity:
  - CUCB and KL-OSM require at least $\Omega(K)$
  - CUCB takes $\mathcal{O}(K(\log K + \mathcal{T}_{\mathrm{member}}))$ and KL-OSM takes $\mathcal{O}(K(\log K + \mathcal{T}_{\mathrm{member}} + \text{line search}))$.

[1] $\mathcal{I}$ satisfy hereditary property and augmentation property.
[2] We assume for each $k \in [K]$, the support of ν_k is $[a, b] \subseteq (0, 1)$.

Summary of our results

- Nearly optimal per-round time complexity of `FasterCUCB` on uniform matroid, partition matroid, and graphical matroid
- `FasterCUCB` has the same regret guarantee as `CUCB`

	`CUCB`	`FasterCUCB`
Per-round Time Complexity	$\mathcal{O}(K(\log K + \mathcal{T}_{\text{member}}))$	$\mathcal{O}(D \operatorname{polylog}(T)\, \mathcal{T}_{\text{update}}(\mathcal{A}))$
Uniform Matroid	$\mathcal{O}(K \log K)$	$\mathcal{O}(D \log K \operatorname{polylog}(T))$
Partition Matroid	$\mathcal{O}(K \log K)$	$\mathcal{O}(D \log K \operatorname{polylog}(T))$
Graphical Matroid	$\mathcal{O}(K \log K)$	$\mathcal{O}(D \operatorname{polylog}(K) \operatorname{polylog}(T))$
Transversal Matroid	$\mathcal{O}(K(\log K + DK))$	$\mathcal{O}(D\sqrt{K} \operatorname{polylog}(T))$

Table 3: Per-round time complexity on different matroids. K is the number of arms and $D = \max_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x}\|_0$.

CUCB sampling rule: existing approach

Let $\boldsymbol{f}_k = (\hat{\mu}_k(t-1), \frac{1}{\sqrt{N_k(t-1)}})$ and $\boldsymbol{q} = (1, \lambda_t)$.

$$\boldsymbol{x}(t) \in \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmax}} \sum_{k=1}^{K} \langle \boldsymbol{f}_k, \boldsymbol{q} \rangle x_k \qquad \text{(CUCB)}$$

- `CUCB` and `KL-OSM` rely on the following greedy algorithm:
  (1) Find permutation $\pi : [K] \to [K]$ s.t. $\left\langle \boldsymbol{f}_{\pi(k)}, \boldsymbol{q} \right\rangle \geq \left\langle \boldsymbol{f}_{\pi(k+1)}, \boldsymbol{q} \right\rangle, \forall k$
  (2) $\boldsymbol{x} = \boldsymbol{0}_K$; While($|\mathrm{supp}\,(\boldsymbol{x})| < D$) If($\boldsymbol{x} + \boldsymbol{e}_{\pi(k)} \in \mathcal{I}$) $\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{e}_{\pi(k)}$;
- Time complexity: $\mathcal{O}(K(\log K + \mathcal{T}_{\mathrm{member}}))$

Our approach: rounding + minimum hitting set

Let $\boldsymbol{f}_k = (\hat{\mu}_k(t-1), \frac{1}{\sqrt{N_k(t-1)}})$ and $\boldsymbol{q} = (1, \lambda_t)$.

$$\boldsymbol{x}(t) \in \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmax}} \sum_{k=1}^{K} \langle \boldsymbol{f}_k, \boldsymbol{q} \rangle x_k \tag{CUCB}$$

- Challenge: Every arm k's $\langle \boldsymbol{f}_k, \boldsymbol{q} \rangle$ changes at each round.
- Idea: $\boldsymbol{q}$ is "fixed" $\Rightarrow$ only $D = \max_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x}\|_0$ arms will change

Our approach: rounding + minimum hitting set

Let $\boldsymbol{f}_k = (\hat{\mu}_k(t-1), \frac{1}{\sqrt{N_k(t-1)}})$ and $\boldsymbol{q} = (1, \lambda_t)$.

$$\boldsymbol{x}(t) \in \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}} \sum_{k=1}^{K} \langle \boldsymbol{f}_k, \boldsymbol{q} \rangle x_k \tag{CUCB}$$

- Idea: $\boldsymbol{q}$ is "fixed" $\Rightarrow$ only $D = \max_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x}\|_0$ arms will change
  - [OT23]: Construct a set $\mathcal{H}$ s.t. for any $\boldsymbol{q} \in \mathbb{R}_+^2$, $\exists \boldsymbol{h} \in \mathcal{H}$ s.t.

    $$\langle \boldsymbol{f}_k, \boldsymbol{h} \rangle > \langle \boldsymbol{f}_\ell, \boldsymbol{h} \rangle \Rightarrow \langle \boldsymbol{f}_k, \boldsymbol{q} \rangle \geq \langle \boldsymbol{f}_\ell, \boldsymbol{q} \rangle$$

    for all possible $(k, \ell) \in [K] \times [K]$ and $k \neq \ell$
  - For each $\boldsymbol{h} \in \mathcal{H}$, instantiate a dynamic algorithm $\mathcal{A}_{\boldsymbol{h}}$:
    - fast computation of a maximum-weight base (in $\mathcal{O}(D)$-time)
    - fast update of an arm's weight (in $\mathcal{T}_{\mathrm{update}}$-time)
  - Time complexity: $\mathcal{O}(D \cdot |\mathcal{H}| \cdot \mathcal{T}_{\mathrm{update}})$ Naive: $|\mathcal{H}| = \mathcal{O}(K^2)$

Our approach: rounding + minimum hitting set

Let $\boldsymbol{f}_k = (\hat{\mu}_k(t-1), \frac{1}{\sqrt{N_k(t-1)}})$ and $\boldsymbol{q} = (1, \lambda_t)$.

$$\boldsymbol{x}(t) \in \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmax}} \sum_{k=1}^{K} \langle \boldsymbol{f}_k, \boldsymbol{q} \rangle \, x_k \tag{CUCB}$$

- **(Rounding)** Generate polylog (T) bins with $\epsilon = \frac{1}{\log^m T}$.
  Represent each $\mathrm{BIN}(q,r) = (a(1+\epsilon)^{q-1}, a(1+\epsilon)^q] \times (\frac{(1+\epsilon)^{r-1}}{\sqrt{T}}, \frac{(1+\epsilon)^r}{\sqrt{T}}]$ by $\mathrm{dom}_{q,r} = (a(1+\epsilon)^q, \frac{1}{\sqrt{T}}(1+\epsilon)^r)$.
  $$\forall \boldsymbol{f} \in \mathrm{BIN}(q,r), \quad \frac{\langle \mathrm{dom}_{q,r}, \boldsymbol{q} \rangle}{1+\epsilon} < \langle \boldsymbol{f}, \boldsymbol{q} \rangle \leq \langle \mathrm{dom}_{q,r}, \boldsymbol{q} \rangle, \tag{6}$$
- **(Hitting set)** Generate $\mathcal{H}$ of size polylog (T) s.t. $\forall \boldsymbol{q} \in \mathbb{R}_+^2$, $\exists \boldsymbol{h} \in \mathcal{H}$ s.t.
  $$\langle \mathrm{dom}_{q,r}, \boldsymbol{h} \rangle > \langle \mathrm{dom}_{q',r'}, \boldsymbol{h} \rangle \Rightarrow \langle \mathrm{dom}_{q,r}, \boldsymbol{q} \rangle \geq \langle \mathrm{dom}_{q',r'}, \boldsymbol{q} \rangle \tag{7}$$
  for any $(q,r) \neq (q',r')$, and $\boldsymbol{h}$ can be found in polylog (T) time.

Our approach: rounding + minimum hitting set

Let $\boldsymbol{f}_k = (\hat{\mu}_k(t-1), \frac{1}{\sqrt{N_k(t-1)}})$ and $\boldsymbol{q} = (1, \lambda_t)$.

$$\boldsymbol{x}(t) \in \operatorname*{argmax}_{\boldsymbol{x} \in \mathcal{X}} \sum_{k=1}^{K} \langle \boldsymbol{f}_k, \boldsymbol{q} \rangle x_k \tag{CUCB}$$

- **Initialization**:
  - Generate polylog(T) bins
  - Generate the minimum hitting set $\mathcal{H}$ of size polylog(T)
  - For each $\boldsymbol{h} \in \mathcal{H}$, instantiate a dynamic algorithm $\mathcal{A}_{\boldsymbol{h}}$ with arm k's weight $\langle \mathrm{dom}(\boldsymbol{f}_k), \boldsymbol{h} \rangle$
- **FindBase($\boldsymbol{q}$)**: Find $\boldsymbol{h} \in \mathcal{H}$ such that

$$\langle \mathrm{dom}_{q,r}, \boldsymbol{h} \rangle > \langle \mathrm{dom}_{q',r'}, \boldsymbol{h} \rangle \Rightarrow \langle \mathrm{dom}_{q,r}, \boldsymbol{q} \rangle \geq \langle \mathrm{dom}_{q',r'}, \boldsymbol{q} \rangle$$

  Call $\mathcal{A}_{\boldsymbol{h}}$ to output a $(1+\epsilon)$-approx maximum-weight base
- **UpdateFeature($\boldsymbol{f}, k$)**: $\forall \boldsymbol{h} \in \mathcal{H}$, update k's weight to $\langle \mathrm{dom}(\boldsymbol{f}), \boldsymbol{h} \rangle$

Our approach: rounding + minimum hitting set

Let $\boldsymbol{f}_k = (\hat{\mu}_k(t-1), \frac{1}{\sqrt{N_k(t-1)}})$ and $\boldsymbol{q} = (1, \lambda_t)$.

$$\boldsymbol{x}(t) \in \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmax}} \sum_{k=1}^{K} \langle \boldsymbol{f}_k, \boldsymbol{q} \rangle x_k \qquad \text{(CUCB)}$$

- `FasterCUCB`: At round t,
  - Compute $\boldsymbol{x}(t)$ by **FindBase**($\boldsymbol{q}$)
  - For each $k \in \operatorname{supp}(\boldsymbol{x}(t))$, observe $y_k(t) \sim \nu_k$ and **UpdateFeature**$((\frac{(t-1)\mu_k(t-1)+y_k(t)}{t}, \frac{1}{\sqrt{N_k(t-1)+1}}), k)$
- Time complexity: $\mathcal{O}(D \operatorname{polylog}(T)\, \mathcal{T}_{\text{update}}(\mathcal{A}))$
- Regret upper bound: $\lim_{T\to\infty} \frac{R(T)}{\log T} = \mathcal{O}\left(\frac{K-D}{\triangle_{\min}}\right)$.

Regret analysis: FasterCUCB

- **(Challenge)** $\boldsymbol{x}(t) \leftarrow$**FindBase(**$\boldsymbol{q}$**)** is a $(1+\epsilon)$-approx to the maximum weight base

Regret analysis: FasterCUCB

- Fix $\boldsymbol{i}^\star \in \operatorname{argmax}_{\boldsymbol{x} \in \mathcal{X}} \langle \boldsymbol{x}, \boldsymbol{\mu} \rangle$ and let $\triangle_{j,k} = \mu_j - \mu_k$
- Construct a bijection $g_t : \operatorname{supp}(\boldsymbol{i}^\star) \mapsto \operatorname{supp}(\boldsymbol{x}(t))$ such that
  - if $j \in \operatorname{supp}(\boldsymbol{i}^\star) \cap \operatorname{supp}(\boldsymbol{x}(t))$, then $g_t(j) = j$
  - if $j \in \operatorname{supp}(\boldsymbol{i}^\star) \setminus \operatorname{supp}(\boldsymbol{x}(t))$, then $(1+\epsilon) \underbrace{\langle \boldsymbol{f}_{g_t(j)}, \boldsymbol{q} \rangle}_{\text{UCB index of arm } g_t(j)} \geq \underbrace{\langle \boldsymbol{f}_j, \boldsymbol{q} \rangle}_{\text{UCB index of arm } j}$
- $R(T) = \sum_{k \notin \operatorname{supp}(\boldsymbol{i}^\star)} \sum_{j \in \operatorname{supp}(\boldsymbol{i}^\star)} \triangle_{j,k} \sum_{t=1}^{T} \mathbb{P}[g_t(j) = k]$

Regret analysis: FasterCUCB

- Fix $\boldsymbol{i}^\star \in \text{argmax}_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{x}, \boldsymbol{\mu} \rangle$ and let $\triangle_{j,k} = \mu_j - \mu_k$
- Construct a bijection $g_t : \text{supp}(\boldsymbol{i}^\star) \mapsto \text{supp}(\boldsymbol{x}(t))$ such that
  - if $j \in \text{supp}(\boldsymbol{i}^\star) \cap \text{supp}(\boldsymbol{x}(t))$, then $g_t(j) = j$
  - if $j \in \text{supp}(\boldsymbol{i}^\star) \setminus \text{supp}(\boldsymbol{x}(t))$, then $(1+\epsilon) \underbrace{\langle \boldsymbol{f}_{g_t(j)}, \boldsymbol{q} \rangle}_{\text{UCB index of arm } g_t(j)} \geq \underbrace{\langle \boldsymbol{f}_j, \boldsymbol{q} \rangle}_{\text{UCB index of arm } j}$
- $R(T) = \sum_{k \notin \text{supp}(\boldsymbol{i}^\star)} \sum_{j \in \text{supp}(\boldsymbol{i}^\star): \triangle_{j,k} > 0} \triangle_{j,k} \sum_{t=1}^{T} \mathbb{P}[g_t(j) = k]$

Regret analysis: FasterCUCB

- Construct a bijection $g_t : \text{supp}(\boldsymbol{i}^\star) \mapsto \text{supp}(\boldsymbol{x}(t))$ such that
  - if $j \in \text{supp}(\boldsymbol{i}^\star) \cap \text{supp}(\boldsymbol{x}(t))$, then $g_t(j) = j$
  - if $j \in \text{supp}(\boldsymbol{i}^\star) \setminus \text{supp}(\boldsymbol{x}(t))$, then $(1+\epsilon) \underbrace{\langle \boldsymbol{f}_{g_t(j)}, \boldsymbol{q} \rangle}_{\text{UCB index of arm } g_t(j)} \geq \underbrace{\langle \boldsymbol{f}_j, \boldsymbol{q} \rangle}_{\text{UCB index of arm } j}$
- $R(T) \leq \sum_{k \notin \text{supp}(\boldsymbol{i}^\star)} \sum_{j \in \text{supp}(\boldsymbol{i}^\star): \triangle_{j,k} > 0} \triangle_{j,k}(I_{j,k} + II_{j,k})$, where

$$\begin{cases} I_{j,k} = \sum_{t=1}^{T} \mathbb{P}[g_t(j) = k, N_k(t-1) \leq n_{j,k,\epsilon}] \\ II_{j,k} = \sum_{t=1}^{T} \mathbb{P}[g_t(j) = k, N_k(t-1) > n_{j,k,\epsilon}] \end{cases}, \quad n_{j,k,\epsilon} \propto \frac{\log T}{(\frac{\mu_j}{1+\epsilon} - \mu_k)^2}$$

- Let $\epsilon = \frac{1}{\log^m T}$. Then, $\lim_{T \to \infty} \frac{R(T)}{\log T} = \mathcal{O}\left(\frac{K-D}{\triangle_{\min}}\right)$.

Conclusion and future works

- We have developed the first sublinear-time algorithm for matroid semi-bandits.
- Our developed techniques might be use to speedup UCB-style algorithms for other problems, e.g., combinatorial best-arm identification [CLK$^+$14, DKC21] and nonstationary semi-bandits [ZWVL20, CWZZ21]
- Another direction is to study the possibility of speeding up other forms of weights, such as those derived from gradients [TWPL23] and those in the follow-the-perturbed-leader algorithm [NB16]

- Aditya Bhaskara, Moses Charikar, Rajsekar Manokaran, and Aravindan Vijayaraghavan, *On quadratic programming with a ratio objective*, Proc. of ICALP, 2012.
- Francesco Bonchi, Edoardo Galimberti, Aristides Gionis, Bruno Ordozgoiti, and Giancarlo Ruffo, *Discovering polarized communities in signed networks*, Proc. of CIKM, 2019.
- Mihai Cucuringu, Peter Davies, Aldo Glielmo, and Hemant Tyagi, *Sponge: A generalized eigenproblem for clustering signed networks*, Proc. of AISTATS, 2019.
- Shouyuan Chen, Tian Lin, Irwin King, Michael R Lyu, and Wei Chen, *Combinatorial pure exploration of multi-armed bandits*, Proc. of NeurIPS, 2014.
- Richard Combes, Stefan Magureanu, and Alexandre Proutiere, *Minimal exploration in structured stochastic bandits*, Proc. of NeurIPS, 2017.

- Lingyang Chu, Zhefeng Wang, Jian Pei, Jiannan Wang, Zijin Zhao, and Enhong Chen, *Finding gangs in war from signed networks*, Proc. of KDD, 2016.
- Wei Chen, Liwei Wang, Haoyu Zhao, and Kai Zheng, *Combinatorial semi-bandit in the non-stationary environment*, Proc. of UAI, 2021.
- John C Duchi, Peter L Bartlett, and Martin J Wainwright, *Randomized smoothing for stochastic optimization*, SIAM Journal on Optimization (2012).
- Yihan Du, Yuko Kuroki, and Wei Chen, *Combinatorial pure exploration with full-bandit or partial linear feedback*, Proc. of AAAI, 2021.
- Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan, *Online convex optimization in the bandit setting: gradient descent without a gradient*, Proc. of SODA, 2005.

- Aurélien Garivier and Emilie Kaufmann, *Optimal best arm identification with fixed confidence*, Proc. of COLT, 2016.
- Marc Jourdan, Mojmír Mutnỳ, Johannes Kirschner, and Andreas Krause, *Efficient pure exploration for combinatorial bandits with semi-bandit feedback*, Proc. of ALT, 2021.
- Branislav Kveton, Zheng Wen, Azin Ashkan, Hoda Eydgahi, and Brian Eriksson, *Matroid bandits: fast combinatorial optimization with learning*, Proc. of UAI, 2014.
- Gergely Neu and Gábor Bartók, *Importance weighting without importance weights: An efficient algorithm for combinatorial semi-bandits*, JMLR (2016).
- Naoto Ohsaka and Riku Togashi, *Curse of" low" dimensionality in recommender systems*, Proc. of SIGIR, 2023.

Reference iv

- Ruo-Chun Tzeng, Naoto Ohsaka, and Kaito Ariu, *Matroid semi-bandits in sublinear time*, Proc. of ICML, 2024.
- Ruo-Chun Tzeng, Bruno Ordozgoiti, and Aristides Gionis, *Discovering conflicting groups in signed networks*, Proc. of NeurIPS, 2020.
- Mohammad Sadegh Talebi and Alexandre Proutiere, *An optimal algorithm for stochastic matroid bandit optimization*, Proc. of AAMAS, 2016.
- Ruo-Chun Tzeng, Po-An Wang, Florian Adriaens, Aristides Gionis, and Chi-Jen Lu, *Improved analysis of randomized svd for top-eigenvector approximation*, Proc. of AISTATS, 2022.
- Ruo-Chun Tzeng, Po-An Wang, Alexandre Proutiere, and Chi-Jen Lu, *Closing the computational-statistical gap in best arm identification for combinatorial semi-bandits*, Proc. of NeurIPS, 2023.

Reference v

- Roman Vershynin, *High-dimensional probability: An introduction with applications in data science*, Cambridge university press, 2018.
- Po-An Wang, Ruo-Chun Tzeng, and Alexandre Proutiere, *Fast pure exploration via frank-wolfe*, Proc. of NeurIPS, 2021.
- Huozhi Zhou, Lingda Wang, Lav Varshney, and Ee-Peng Lim, *A near-optimal change-detection based algorithm for piecewise-stationary combinatorial semi-bandits*, Proc. of AAAI, 2020.