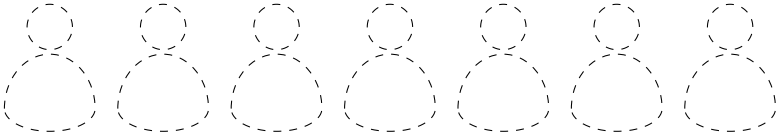
Constant-Competitive Random Assignment MSP Without Knowing the Matroid

Richard Santiago[1], **Ivan Sergeev**[1], Rico Zenklusen[1]

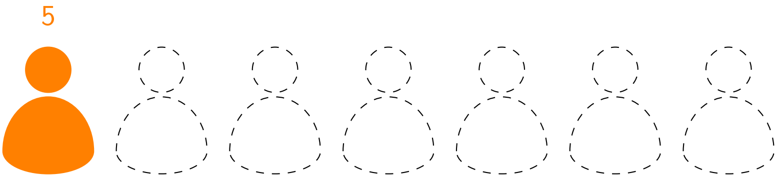
[1] ETH Zürich, Switzerland

IPCO, June 2023

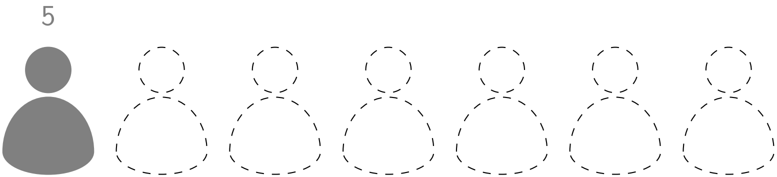
Classical Secretary Problem



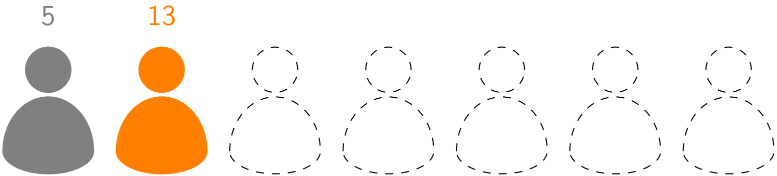
Classical Secretary Problem



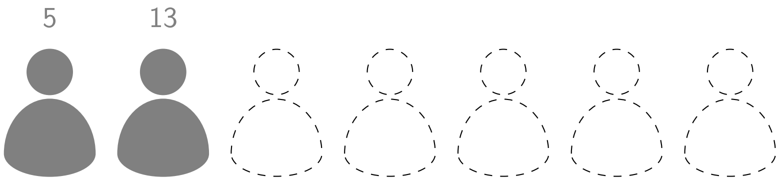
Classical Secretary Problem



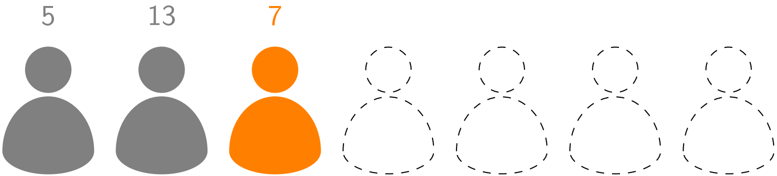
Classical Secretary Problem



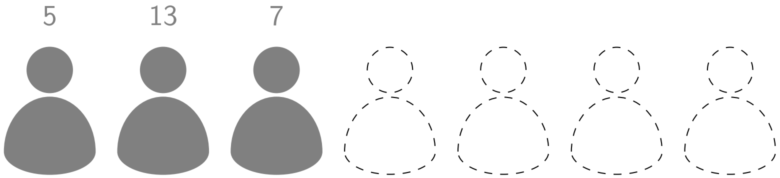
Classical Secretary Problem



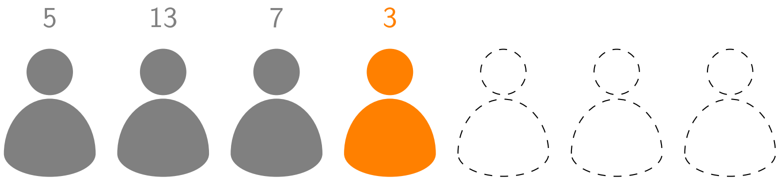
Classical Secretary Problem



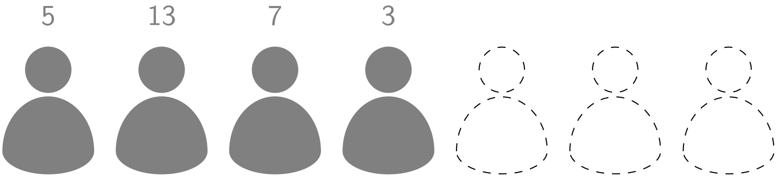
Classical Secretary Problem



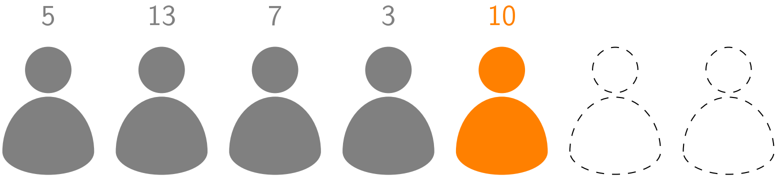
Classical Secretary Problem



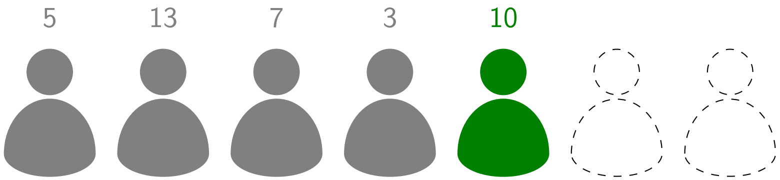
Classical Secretary Problem



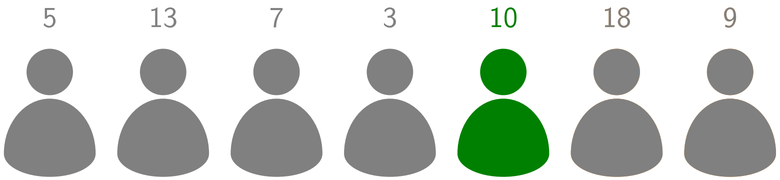
Classical Secretary Problem



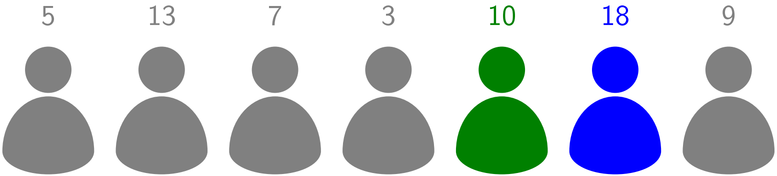
Classical Secretary Problem



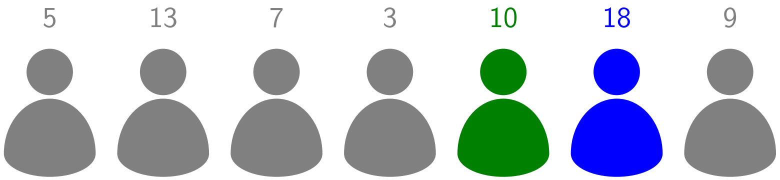
Classical Secretary Problem



Classical Secretary Problem

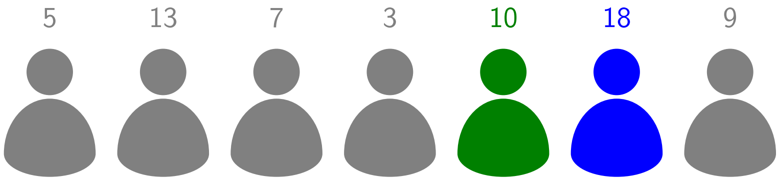


Classical Secretary Problem



$$\text{Competitive Ratio} = \mathbb{E}\left[\text{selected}\right] / \text{optimal}$$

Classical Secretary Problem

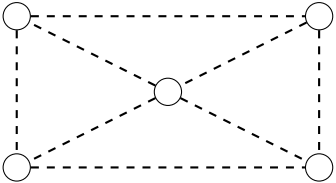


Competitive Ratio $= \mathbb{E}\,[\text{selected}]\,/\text{optimal}$

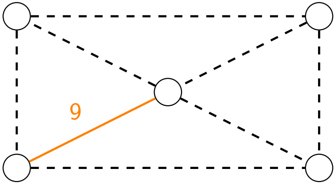
Theorem [Dynkin 1963]

There exists an optimal $\frac{1}{e}$-competitive algorithm.

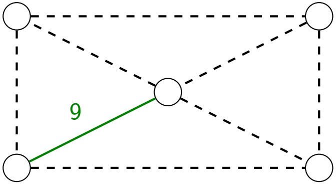
Matroid Secretary Problem



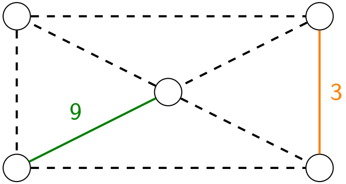
Matroid Secretary Problem



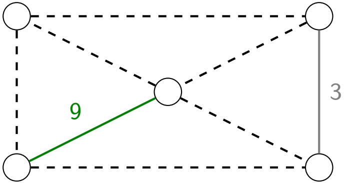
Matroid Secretary Problem



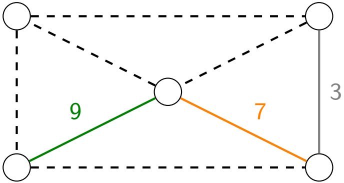
Matroid Secretary Problem



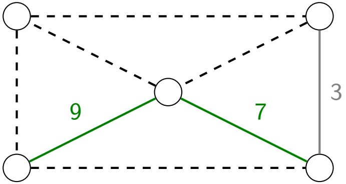
Matroid Secretary Problem



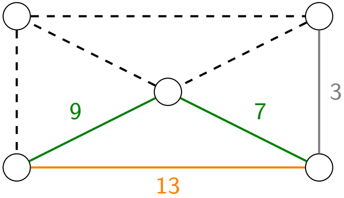
Matroid Secretary Problem



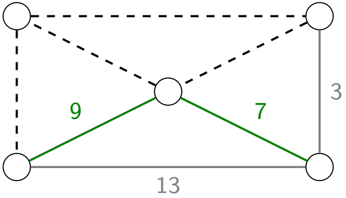
Matroid Secretary Problem



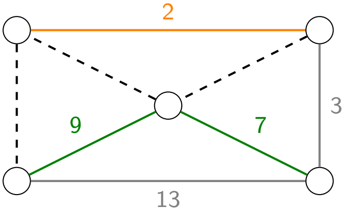
Matroid Secretary Problem



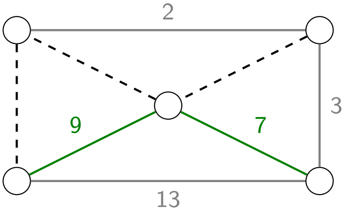
Matroid Secretary Problem



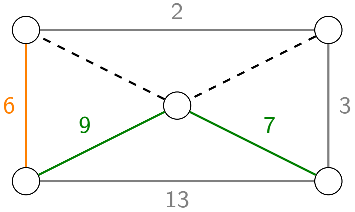
Matroid Secretary Problem



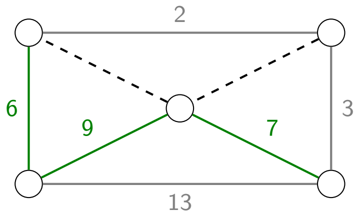
Matroid Secretary Problem



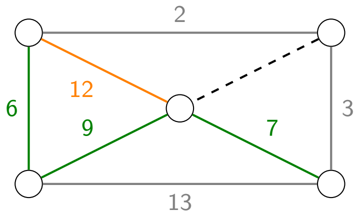
Matroid Secretary Problem



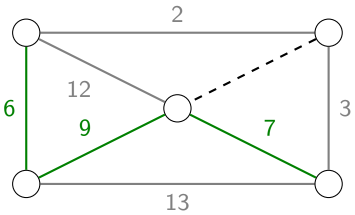
Matroid Secretary Problem



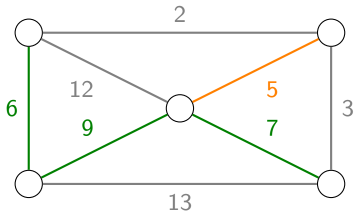
Matroid Secretary Problem



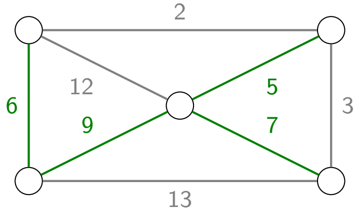
Matroid Secretary Problem



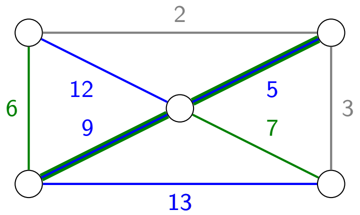
Matroid Secretary Problem



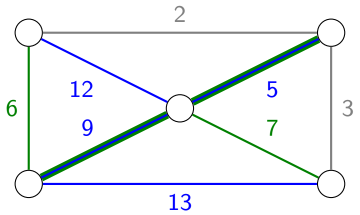
Matroid Secretary Problem



Matroid Secretary Problem

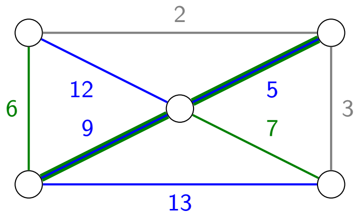


Matroid Secretary Problem



Competitive Ratio $= \mathbb{E}\left[\text{selected}\right] / \text{optimal}$

Matroid Secretary Problem



Competitive Ratio $= \mathbb{E}\,[\text{selected}]\,/\text{optimal}$

Conjecture [Babaioff, Immorlica, Kleinberg 2007]

There exists a constant-competitive algorithm for general matroids.

Prior Work

- **Current best for general matroids:** $O(\log \log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]

Prior Work

- **Current best for general matroids:** $O(\log \log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**

Prior Work

- **Current best for general matroids:** $O(\log\log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**
  - Graphic [Soto, Turkieltaub, Verdugo 2018]

Prior Work

- **Current best for general matroids:** $O(\log \log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**
  - Graphic [Soto, Turkieltaub, Verdugo 2018]
  - Laminar [Soto, Turkieltaub, Verdugo 2018]

Prior Work

- **Current best for general matroids:** $O(\log \log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**
  - Graphic [Soto, Turkieltaub, Verdugo 2018]
  - Laminar [Soto, Turkieltaub, Verdugo 2018]
  - Transversal [Kasselheim et al. 2013]

Prior Work

- **Current best for general matroids:** $O(\log\log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**
  - Graphic [Soto, Turkieltaub, Verdugo 2018]
  - Laminar [Soto, Turkieltaub, Verdugo 2018]
  - Transversal [Kasselheim et al. 2013]
  - Regular [Dinitz, Kortsarz 2013]

Prior Work

- **Current best for general matroids:** $O(\log \log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**
  - Graphic [Soto, Turkieltaub, Verdugo 2018]
  - Laminar [Soto, Turkieltaub, Verdugo 2018]
  - Transversal [Kasselheim et al. 2013]
  - Regular [Dinitz, Kortsarz 2013]

Prior Work

- **Current best for general matroids:** $O(\log \log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**
  - Graphic [Soto, Turkieltaub, Verdugo 2018]
  - Laminar [Soto, Turkieltaub, Verdugo 2018]
  - Transversal [Kasselheim et al. 2013]
  - Regular [Dinitz, Kortsarz 2013]
  - Cographic [Soto 2011]

Prior Work

- **Current best for general matroids:** $O(\log \log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**
  - Graphic [Soto, Turkieltaub, Verdugo 2018]
  - Laminar [Soto, Turkieltaub, Verdugo 2018]
  - Transversal [Kasselheim et al. 2013]
  - Regular [Dinitz, Kortsarz 2013]
  - Cographic [Soto 2011]
  - ...

Prior Work

- **Current best for general matroids:** $O(\log \log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**
  - Graphic [Soto, Turkieltaub, Verdugo 2018]
  - Laminar [Soto, Turkieltaub, Verdugo 2018]
  - Transversal [Kasselheim et al. 2013]
  - Regular [Dinitz, Kortsarz 2013]
  - Cographic [Soto 2011]
  - ...
- **Constant-competitive for certain variants:**

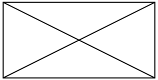
Prior Work

- **Current best for general matroids:** $O(\log\log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**
  - Graphic [Soto, Turkieltaub, Verdugo 2018]
  - Laminar [Soto, Turkieltaub, Verdugo 2018]
  - Transversal [Kasselheim et al. 2013]
  - Regular [Dinitz, Kortsarz 2013]
  - Cographic [Soto 2011]
  - ...
- **Constant-competitive for certain variants:**
  - Free order model [Jaillet, Soto, Zenklusen 2013]

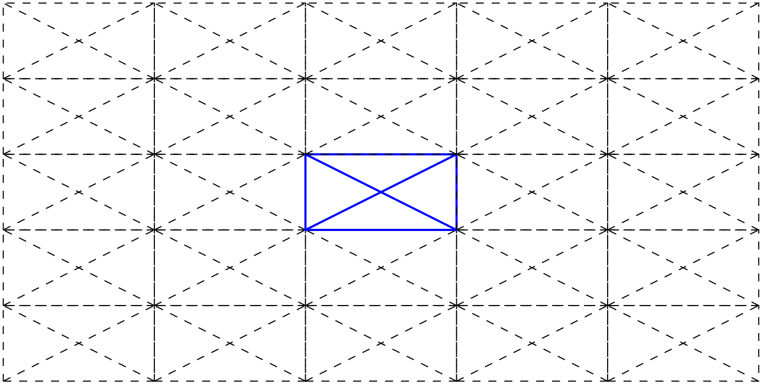
Prior Work

- **Current best for general matroids:** $O(\log\log \mathrm{rank})$-competitive
  - [Lachish 2014], [Feldman, Svensson, Zenklusen 2014]
- **Constant-competitive for specific matroid classes:**
  - Graphic [Soto, Turkieltaub, Verdugo 2018]
  - Laminar [Soto, Turkieltaub, Verdugo 2018]
  - Transversal [Kasselheim et al. 2013]
  - Regular [Dinitz, Kortsarz 2013]
  - Cographic [Soto 2011]
  - ...
- **Constant-competitive for certain variants:**
  - Free order model [Jaillet, Soto, Zenklusen 2013]
  - Random assignment [Soto 2011; Oveis Gharan, Vondrák 2013]

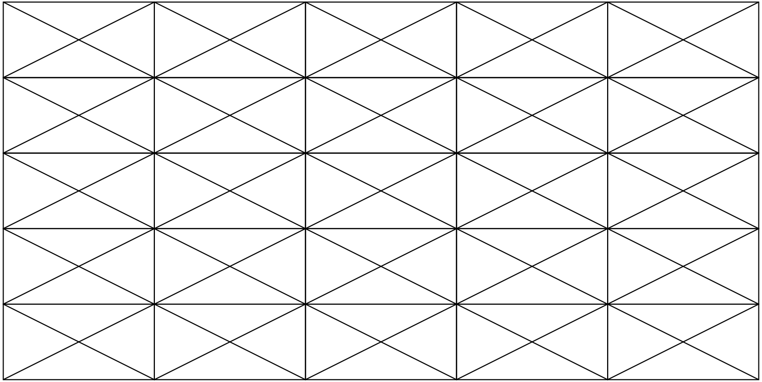
Prior Information



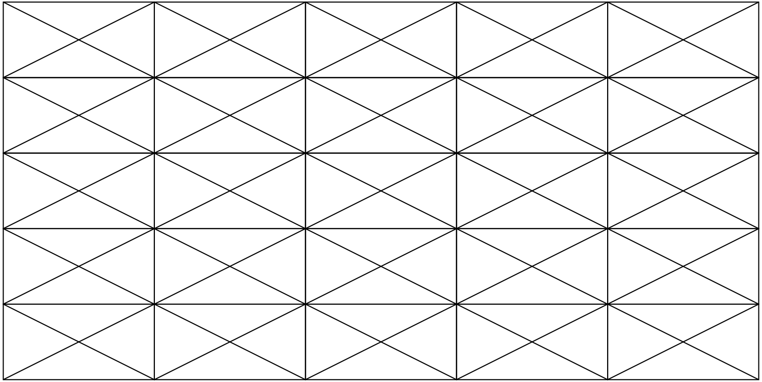
Prior Information



Prior Information

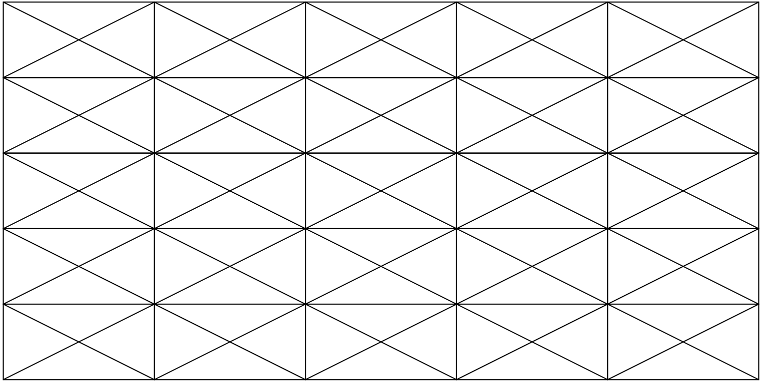


Prior Information



Intuition: Knowing complete matroid structure in advance shouldn't help

Prior Information



Intuition: Knowing complete matroid structure in advance shouldn't help

Goal: Reduce amount of required prior information in RAMSP

Random Assignment Model

- **Setting:**

Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one

Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one
  - Weights chosen adversarially, but assigned to elements randomly

Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one
  - Weights chosen adversarially, but assigned to elements randomly
  - Goal: Select $S \in \mathcal{I}$ with weight $w(S)$ as large as possible

Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one
  - Weights chosen adversarially, but assigned to elements randomly
  - Goal: Select $S \in \mathcal{I}$ with weight $w(S)$ as large as possible
- **Motivation:**

Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one
  - Weights chosen adversarially, but assigned to elements randomly
  - Goal: Select $S \in \mathcal{I}$ with weight $w(S)$ as large as possible
- **Motivation:**
  - Left as open question in [Babaioff, Immorlica, Kleinberg 2007]

Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one
  - Weights chosen adversarially, but assigned to elements randomly
  - Goal: Select $S \in \mathcal{I}$ with weight $w(S)$ as large as possible
- **Motivation:**
  - Left as open question in [Babaioff, Immorlica, Kleinberg 2007]
  - Easier than MSP

Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one
  - Weights chosen adversarially, but assigned to elements randomly
  - Goal: Select $S \in \mathcal{I}$ with weight $w(S)$ as large as possible
- **Motivation:**
  - Left as open question in [Babaioff, Immorlica, Kleinberg 2007]
  - Easier than MSP
- **Prior work:**

Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one
  - Weights chosen adversarially, but assigned to elements randomly
  - Goal: Select $S \in \mathcal{I}$ with weight $w(S)$ as large as possible
- **Motivation:**
  - Left as open question in [Babaioff, Immorlica, Kleinberg 2007]
  - Easier than MSP
- **Prior work:**
  - [Soto 2011]: Constant-competitive algorithm, but need full matroid in advance

Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one
  - Weights chosen adversarially, but assigned to elements randomly
  - Goal: Select $S \in \mathcal{I}$ with weight $w(S)$ as large as possible
- **Motivation:**
  - Left as open question in [Babaioff, Immorlica, Kleinberg 2007]
  - Easier than MSP
- **Prior work:**
  - [Soto 2011]: Constant-competitive algorithm, but need full matroid in advance
  - [Oveis Gharan, Vondrák 2013]: Extended to adversarial arrival order, but same limitation

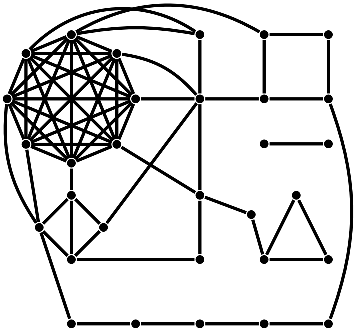
Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one
  - Weights chosen adversarially, but assigned to elements randomly
  - Goal: Select $S \in \mathcal{I}$ with weight $w(S)$ as large as possible
- **Motivation:**
  - Left as open question in [Babaioff, Immorlica, Kleinberg 2007]
  - Easier than MSP
- **Prior work:**
  - [Soto 2011]: Constant-competitive algorithm, but need full matroid in advance
  - [Oveis Gharan, Vondrák 2013]: Extended to adversarial arrival order, but same limitation
- **Our main result:**

Random Assignment Model

- **Setting:**
  - Elements of matroid $\mathcal{M} = (N, \mathcal{I})$ revealed online one by one
  - Weights chosen adversarially, but assigned to elements randomly
  - Goal: Select $S \in \mathcal{I}$ with weight $w(S)$ as large as possible
- **Motivation:**
  - Left as open question in [Babaioff, Immorlica, Kleinberg 2007]
  - Easier than MSP
- **Prior work:**
  - [Soto 2011]: Constant-competitive algorithm, but need full matroid in advance
  - [Oveis Gharan, Vondrák 2013]: Extended to adversarial arrival order, but same limitation
- **Our main result:**
  - Constant-competitive algorithm without knowing matroid upfront

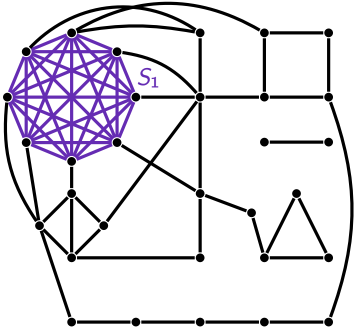
Approach from [Soto 2013]



- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

Approach from [Soto 2013]

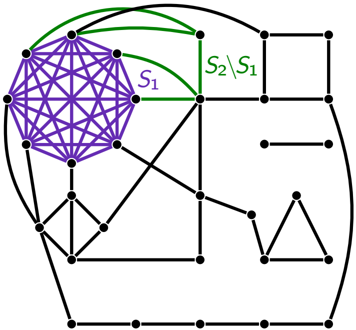


- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

- S_1 is the densest set in $\mathcal{M}$

Approach from [Soto 2013]

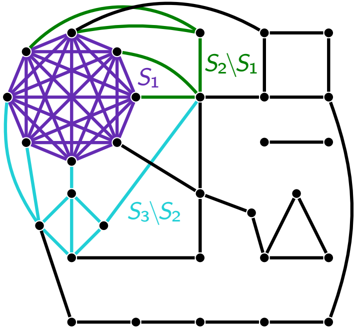


- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

- S_1 is the densest set in $\mathcal{M}$
- $S_2 \setminus S_1$ is the densest set in $\mathcal{M}/S_1$

Approach from [Soto 2013]

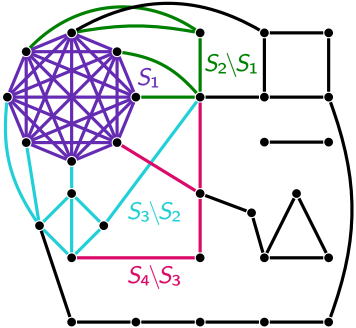


- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

- S_1 is the densest set in $\mathcal{M}$
- $S_2 \setminus S_1$ is the densest set in $\mathcal{M}/S_1$
- $\ldots$

Approach from [Soto 2013]

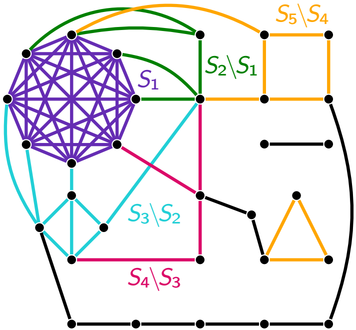


- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

- S_1 is the densest set in $\mathcal{M}$
- $S_2 \setminus S_1$ is the densest set in $\mathcal{M}/S_1$
- ...
- $S_{i+1} \setminus S_i$ is the densest set in $\mathcal{M}/S_i$

Approach from [Soto 2013]

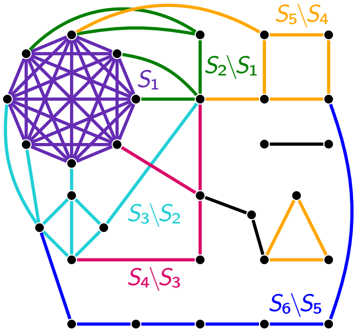


- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

- S_1 is the densest set in $\mathcal{M}$
- $S_2 \setminus S_1$ is the densest set in $\mathcal{M}/S_1$
- ...
- $S_{i+1} \setminus S_i$ is the densest set in $\mathcal{M}/S_i$

Approach from [Soto 2013]

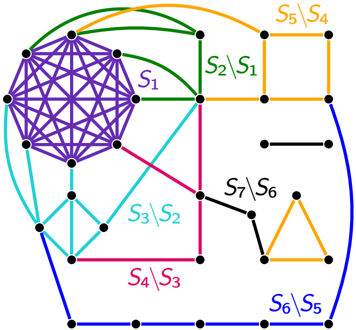


- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

- S_1 is the densest set in $\mathcal{M}$
- $S_2 \setminus S_1$ is the densest set in $\mathcal{M}/S_1$
- ...
- $S_{i+1} \setminus S_i$ is the densest set in $\mathcal{M}/S_i$

Approach from [Soto 2013]

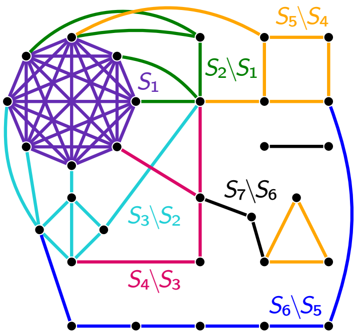


- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

- S_1 is the densest set in $\mathcal{M}$
- $S_2 \setminus S_1$ is the densest set in $\mathcal{M}/S_1$
- ...
- $S_{i+1} \setminus S_i$ is the densest set in $\mathcal{M}/S_i$

Approach from [Soto 2013]



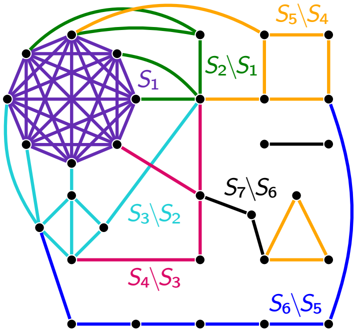
- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

- S_1 is the densest set in $\mathcal{M}$
- $S_2 \setminus S_1$ is the densest set in $\mathcal{M}/S_1$
- ...
- $S_{i+1} \setminus S_i$ is the densest set in $\mathcal{M}/S_i$

- Build matroids $\mathcal{M}_i = (\mathcal{M}/S_i)|_{S_{i+1} \setminus S_i}$ (principal sequence of $\mathcal{M}$)

Approach from [Soto 2013]



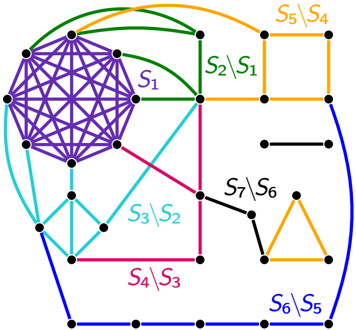
- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

- S_1 is the densest set in $\mathcal{M}$
- $S_2 \setminus S_1$ is the densest set in $\mathcal{M}/S_1$
- ...
- $S_{i+1} \setminus S_i$ is the densest set in $\mathcal{M}/S_i$

- Build matroids $\mathcal{M}_i = (\mathcal{M}/S_i)|_{S_{i+1} \setminus S_i}$ (principal sequence of $\mathcal{M}$)
- Devise constant-competitive algorithm for (very well-structured) $\mathcal{M}_i$'s

Approach from [Soto 2013]



- Density of a set of edges U is

$$\frac{|U|}{r(U)} = \frac{|U|}{\text{max size of forest contained in } U}$$

- S_1 is the densest set in $\mathcal{M}$
- $S_2 \setminus S_1$ is the densest set in $\mathcal{M}/S_1$
- ...
- $S_{i+1} \setminus S_i$ is the densest set in $\mathcal{M}/S_i$

- Build matroids $\mathcal{M}_i = (\mathcal{M}/S_i)|_{S_{i+1} \setminus S_i}$ (principal sequence of $\mathcal{M}$)
- Devise constant-competitive algorithm for (very well-structured) $\mathcal{M}_i$'s
- Use $\mathrm{OPT}(\mathcal{M}) = \Theta\left(\sum_{i=1}^{k} \mathrm{OPT}(\mathcal{M}_i)\right)$

Issues With Generalization

- Can't compute principal decomposition without knowing full structure

Issues With Generalization

- Can't compute principal decomposition without knowing full structure
  - Natural approach: sample constant fraction of elements

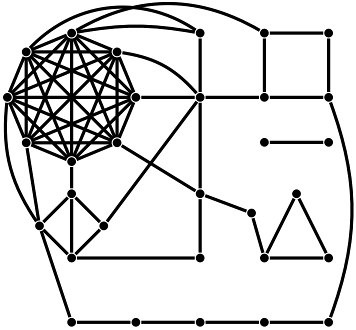
Issues With Generalization

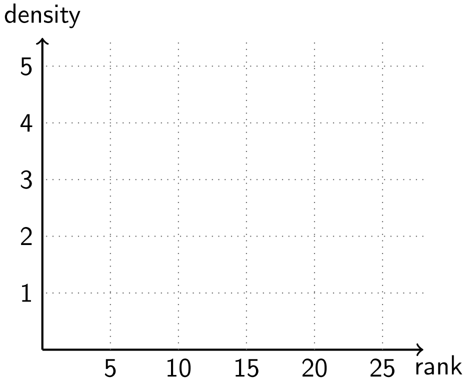
- Can't compute principal decomposition without knowing full structure
  - Natural approach: sample constant fraction of elements
- Decompositions for sample and whole matroid might differ significantly

Issues With Generalization

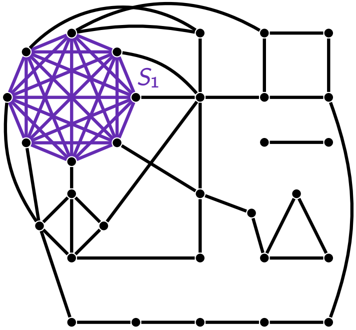
- Can't compute principal decomposition without knowing full structure
  - Natural approach: sample constant fraction of elements
- Decompositions for sample and whole matroid might differ significantly
- Elements might end up in different partitions depending on sample

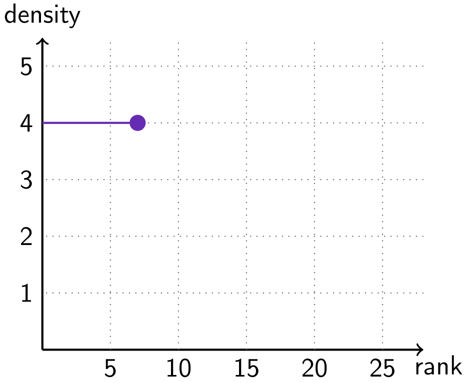
Rank-Density Curves



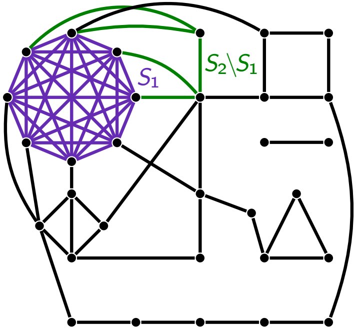


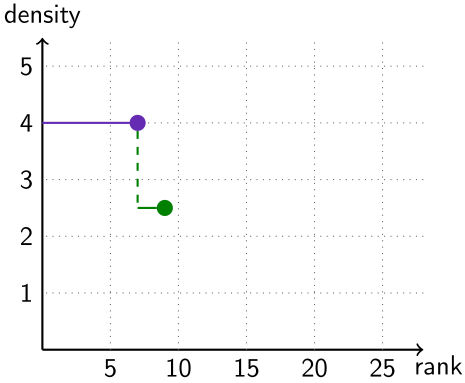
Rank-Density Curves



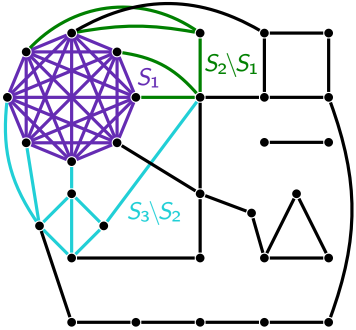


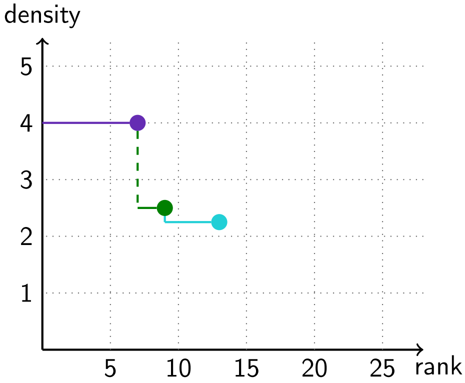
Rank-Density Curves



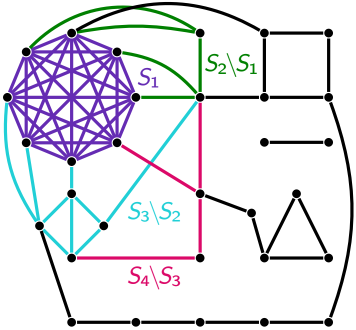


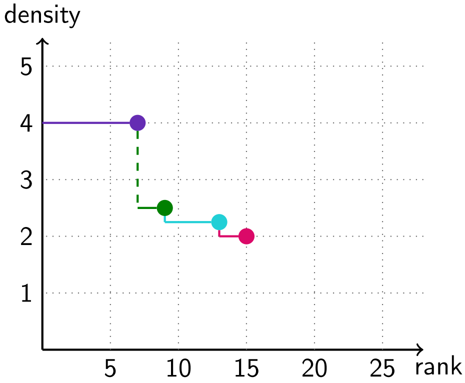
Rank-Density Curves



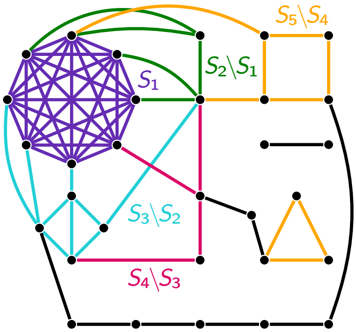


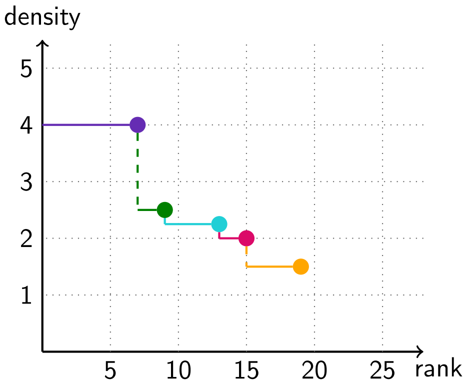
Rank-Density Curves



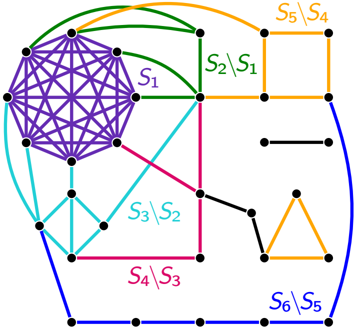


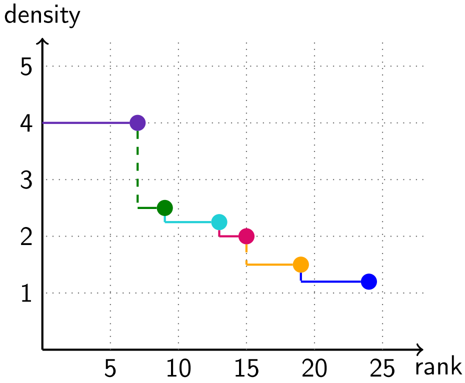
Rank-Density Curves



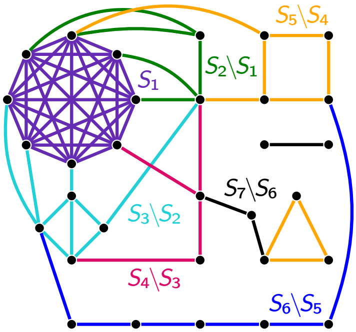


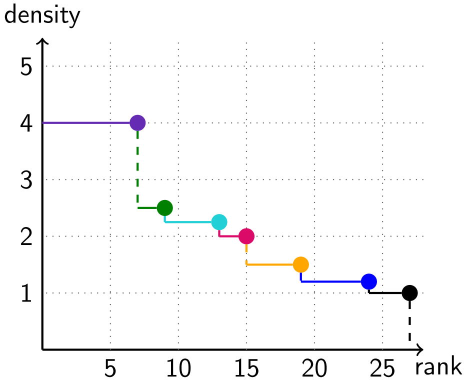
Rank-Density Curves



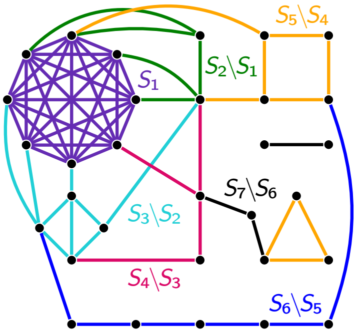


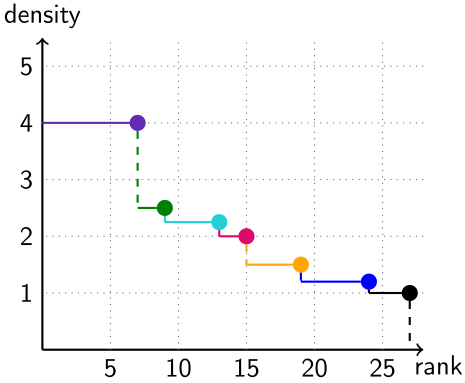
Rank-Density Curves





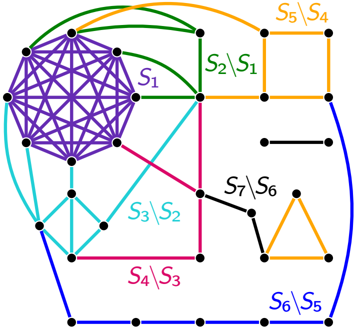
Rank-Density Curves

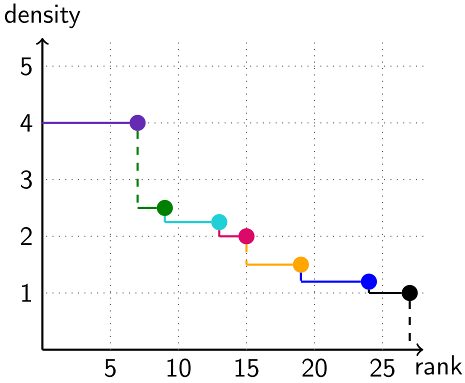




- **Advantages of RDCs:**

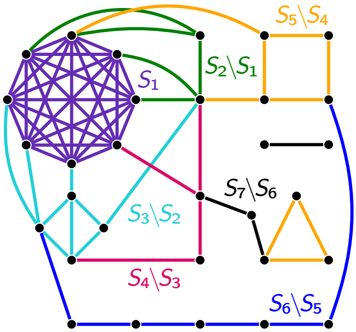
Rank-Density Curves

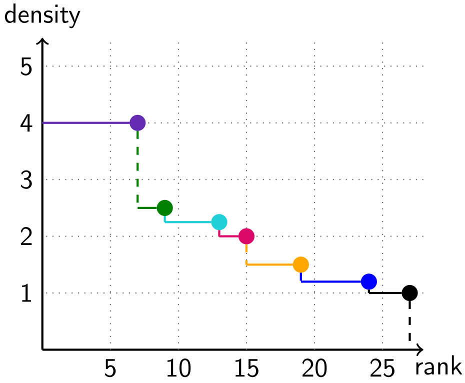




- **Advantages of RDCs:**
  - Capture key parameters of principal sequence

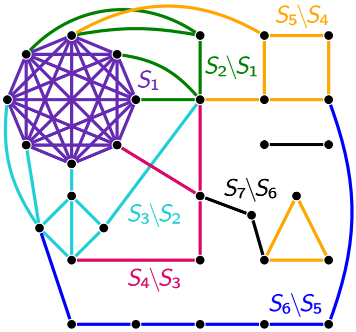
Rank-Density Curves

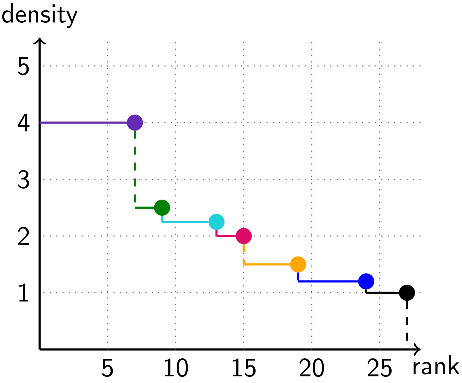




- **Advantages of RDCs:**
  - Capture key parameters of principal sequence
  - Can be compared to OPT and competitiveness

Rank-Density Curves





- **Advantages of RDCs:**
  - Capture key parameters of principal sequence
  - Can be compared to OPT and competitiveness
  - Can be exploited algorithmically

Our Approach and Technical Contributions

- Sample constant fraction of elements $\rightarrow$ set S

Our Approach and Technical Contributions

- Sample constant fraction of elements $\rightarrow$ set S

Theorem

With constant probability, RDCs of $\mathcal{M}$, $\mathcal{M}|_S$, and $\mathcal{M}|_{N \setminus S}$ are close to each other.

Our Approach and Technical Contributions

- Sample constant fraction of elements $\rightarrow$ set S
- Use RDC of $\mathcal{M}|_S$ to approximate RDC of $\mathcal{M}|_{N \setminus S}$

Theorem

With constant probability, RDCs of $\mathcal{M}$, $\mathcal{M}|_S$, and $\mathcal{M}|_{N \setminus S}$ are close to each other.

Our Approach and Technical Contributions

- Sample constant fraction of elements $\to$ set S
- Use RDC of $\mathcal{M}|_S$ to approximate RDC of $\mathcal{M}|_{N \setminus S}$

Theorem

With constant probability, RDCs of $\mathcal{M}$, $\mathcal{M}|_S$, and $\mathcal{M}|_{N \setminus S}$ are close to each other.

Theorem

There is an algorithm that, given a good approximate RDC of $\mathcal{M}$, returns $\Omega(\mathrm{OPT})$ weight in expectation.

Our Approach and Technical Contributions

- Sample constant fraction of elements $\rightarrow$ set S
- Use RDC of $\mathcal{M}|_S$ to approximate RDC of $\mathcal{M}|_{N \setminus S}$
- Use approximate RDC to retrieve $\Omega(\mathrm{OPT})$ weight from $\mathcal{M}|_{N \setminus S}$ in expectation

Theorem

With constant probability, RDCs of $\mathcal{M}$, $\mathcal{M}|_S$, and $\mathcal{M}|_{N \setminus S}$ are close to each other.

Theorem

There is an algorithm that, given a good approximate RDC of $\mathcal{M}$, returns $\Omega(\mathrm{OPT})$ weight in expectation.

Our Approach and Technical Contributions

- Sample constant fraction of elements $\rightarrow$ set S
- Use RDC of $\mathcal{M}|_S$ to approximate RDC of $\mathcal{M}|_{N\setminus S}$
- Use approximate RDC to retrieve $\Omega(\mathrm{OPT})$ weight from $\mathcal{M}|_{N\setminus S}$ in expectation
- Conclude that we return $\Omega(\mathrm{OPT})$ weight from $\mathcal{M}$ in expectation

Theorem

With constant probability, RDCs of $\mathcal{M}$, $\mathcal{M}|_S$, and $\mathcal{M}|_{N\setminus S}$ are close to each other.

Theorem

There is an algorithm that, given a good approximate RDC of $\mathcal{M}$, returns $\Omega(\mathrm{OPT})$ weight in expectation.

Conclusion

- **Main result:** Constant-competitive algorithm for RAMSP when matroid not given upfront

Conclusion

- **Main result:** Constant-competitive algorithm for RAMSP when matroid not given upfront
  - Solves open question from [Babaioff, Immorlica, Kleinberg 2007]

Conclusion

- **Main result:** Constant-competitive algorithm for RAMSP when matroid not given upfront
  - Solves open question from [Babaioff, Immorlica, Kleinberg 2007]
  - First constant-competitive algorithm for "matroid unknown" setting

Conclusion

- **Main result:** Constant-competitive algorithm for RAMSP when matroid not given upfront
  - Solves open question from [Babaioff, Immorlica, Kleinberg 2007]
  - First constant-competitive algorithm for "matroid unknown" setting
  - Bonus: works in "random sample + adversarial order" setting

Conclusion

- **Main result:** Constant-competitive algorithm for RAMSP when matroid not given upfront
  - Solves open question from [Babaioff, Immorlica, Kleinberg 2007]
  - First constant-competitive algorithm for "matroid unknown" setting
  - Bonus: works in "random sample + adversarial order" setting
- **Open questions:**

Conclusion

- **Main result:** Constant-competitive algorithm for RAMSP when matroid not given upfront
  - Solves open question from [Babaioff, Immorlica, Kleinberg 2007]
  - First constant-competitive algorithm for "matroid unknown" setting
  - Bonus: works in "random sample + adversarial order" setting
- **Open questions:**
  - Improve competitive ratio?

Conclusion

- **Main result:** Constant-competitive algorithm for RAMSP when matroid not given upfront
  - Solves open question from [Babaioff, Immorlica, Kleinberg 2007]
  - First constant-competitive algorithm for "matroid unknown" setting
  - Bonus: works in "random sample + adversarial order" setting
- **Open questions:**
  - Improve competitive ratio?
  - Utilize rank-density curves or densities in general MSP?

Conclusion

- **Main result:** Constant-competitive algorithm for RAMSP when matroid not given upfront
  - Solves open question from [Babaioff, Immorlica, Kleinberg 2007]
  - First constant-competitive algorithm for "matroid unknown" setting
  - Bonus: works in "random sample + adversarial order" setting
- **Open questions:**
  - Improve competitive ratio?
  - Utilize rank-density curves or densities in general MSP?
  - Resolve general MSP conjecture from [Babaioff, Immorlica, Kleinberg 2007]?