Learning Popular Ranking Under Constrained Sensing

Srikanth Jagabathula Devavrat Shah

Laboratory for Information & Decision Systems
Massachusetts Institute of Technology

Motivation

Three examples:

One. Elections

- n candidates
- Each voter has a preference list in mind

Goal:

- Learn popular preferences/ranking through polls

Motivation

Three examples:

Two. Revenue maximization

- n products
- Each customer has a preference list in mind

Goal:

- Learn popular preferences/ranking through polls

Motivation

Three examples:

Three. Sports league

- n teams
- Each team has inherent 'quality' PLUS uncertainty
  → distribution over rankings of teams

Goal:

- Learn popular rankings through results of games

Motivation

Three examples:

- Learn popular rankings with sparse support
- Through partial data

Basic questions:

- When is it possible to learn?
- How to learn?
- Partial information:
  - types?
  - how they affect learnability?

Outline

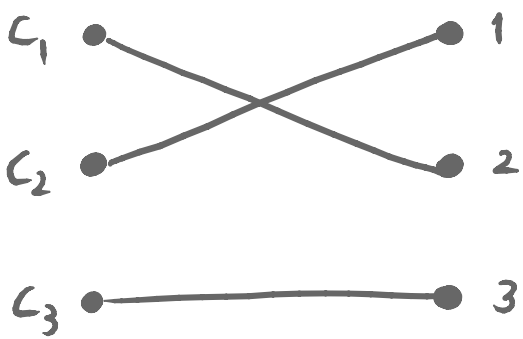
1. Types of partial information

2. Recovery results

3. Algorithm, recoverability condition

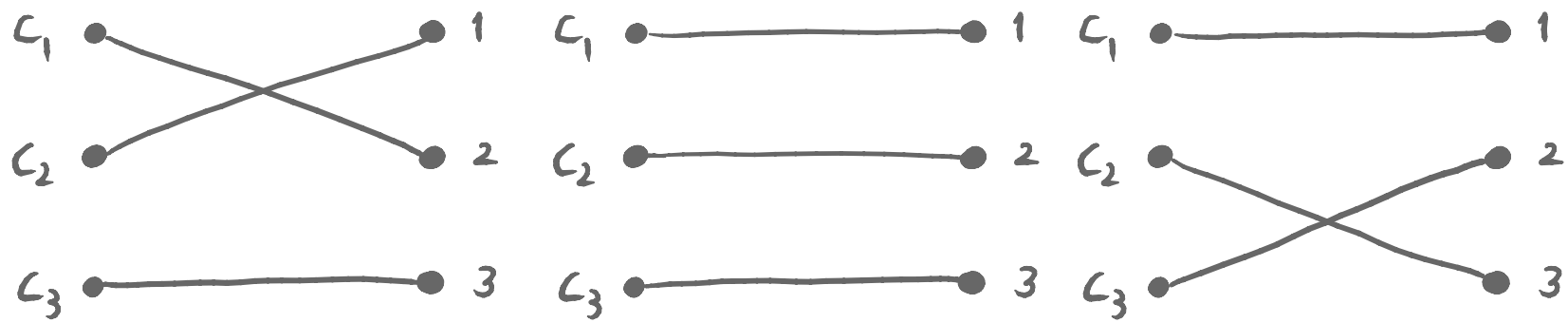
Partial Information Types

First order information: $n = 3$



Partial Information Types

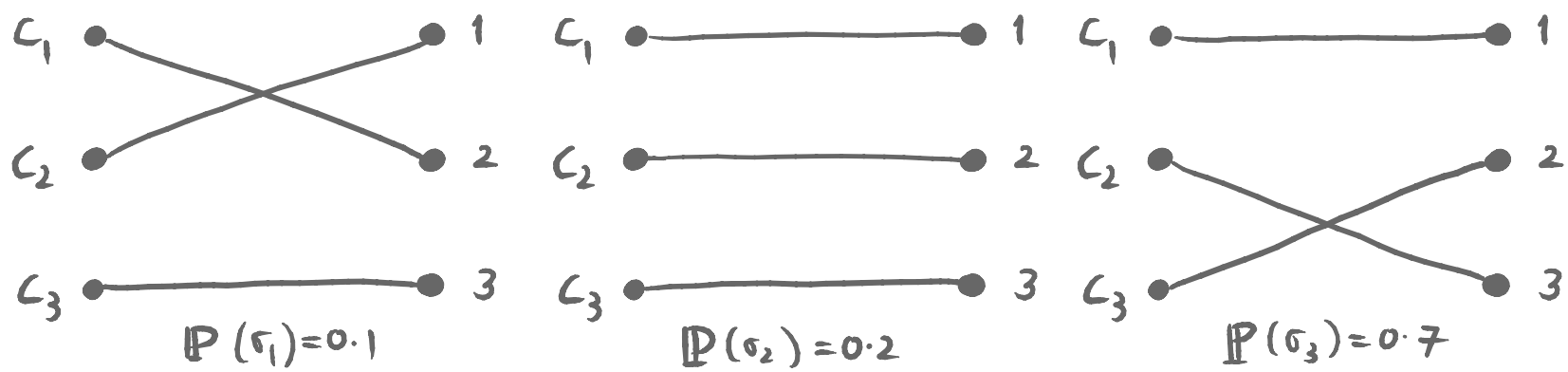
First order information: $n = 3$



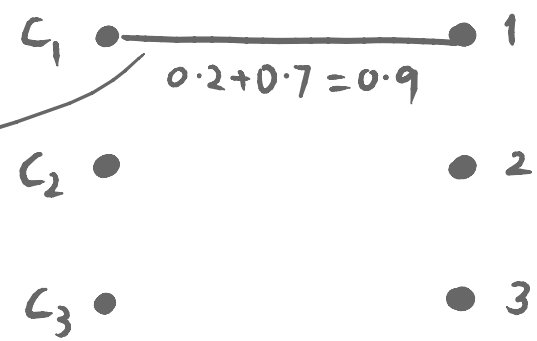
$\mathbb{P}(\sigma_1) = 0.1$ $\mathbb{P}(\sigma_2) = 0.2$ $\mathbb{P}(\sigma_3) = 0.7$

Partial Information Types

First order information: $n = 3$

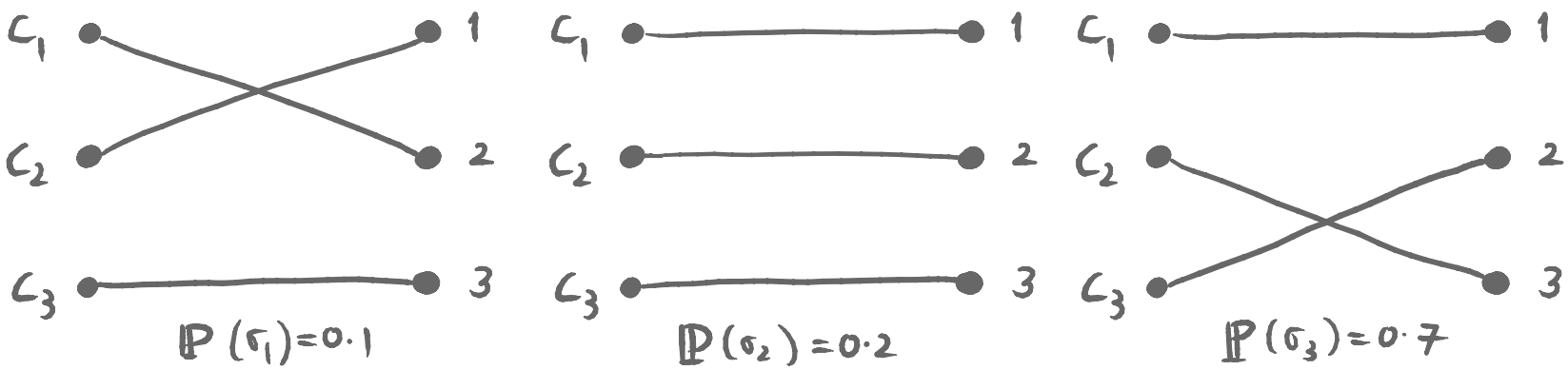


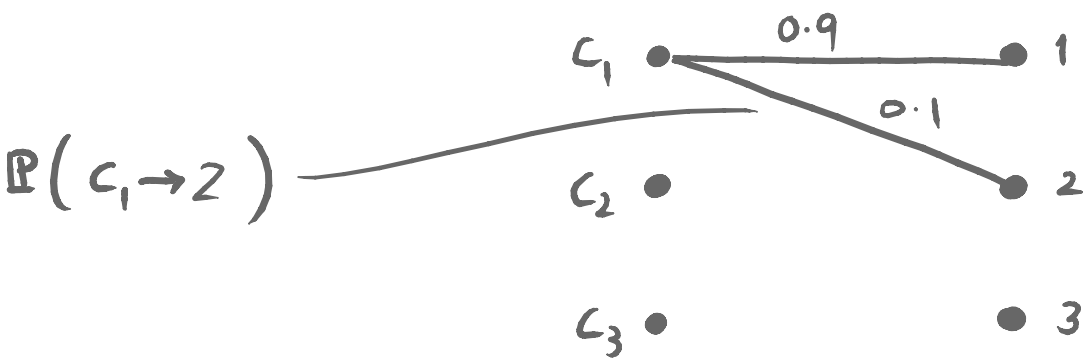
$\mathbb{P}(c_1 \to 1)$



Partial Information Types

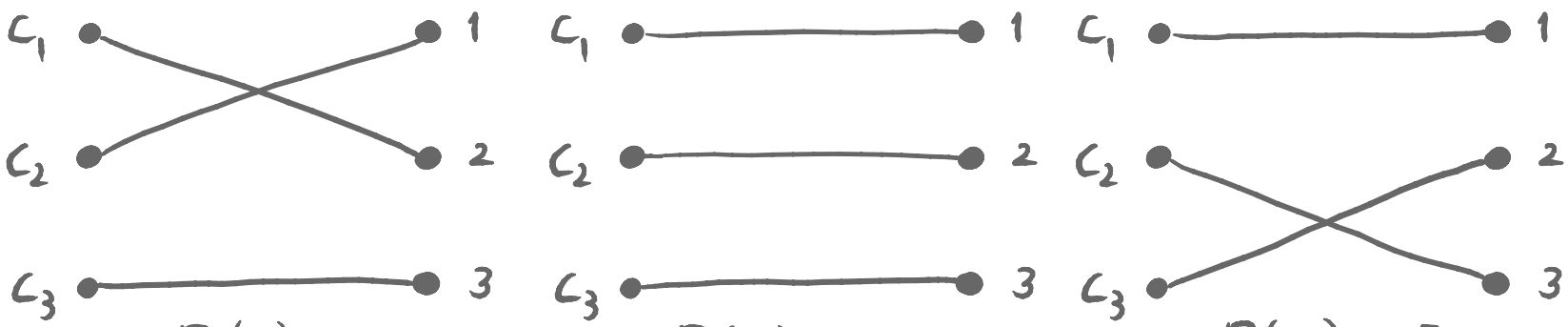
First order information: $n = 3$





Partial Information Types

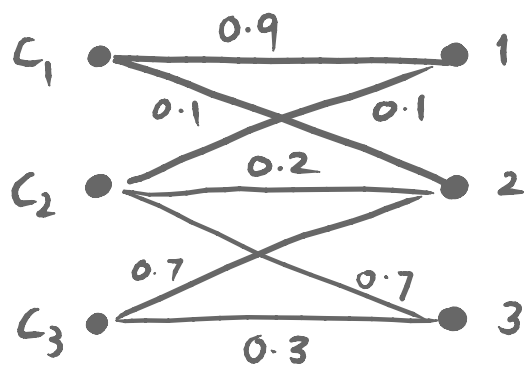
First order information: $n = 3$



$\mathbb{P}(\sigma_1) = 0.1$ $\quad$ $\mathbb{P}(\sigma_2) = 0.2$ $\quad$ $\mathbb{P}(\sigma_3) = 0.7$

$$Q = [Q_{ij}] \equiv$$

$\mathbb{P}(c_i \to j)$

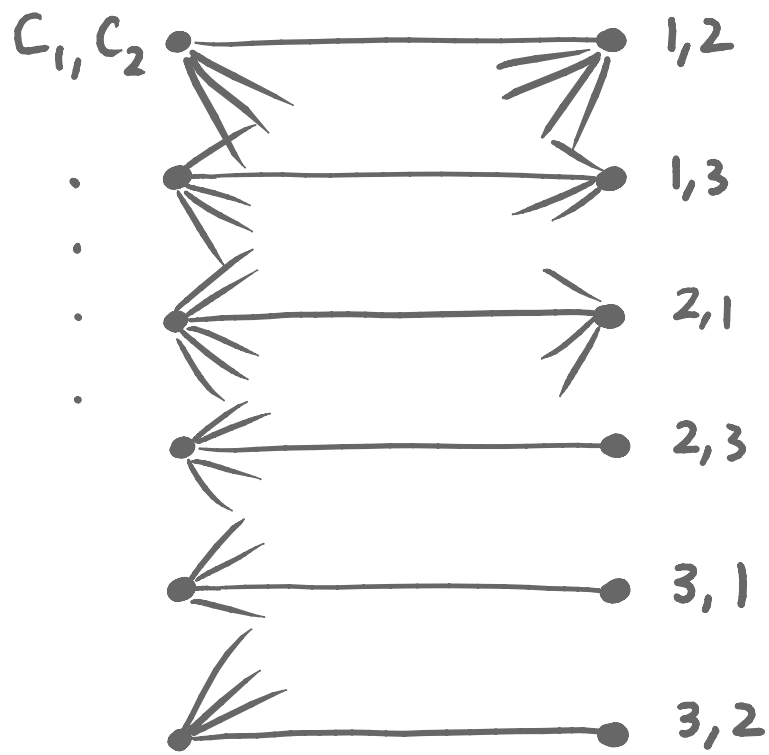


Partial Information Types

Second order information: $n = 3$

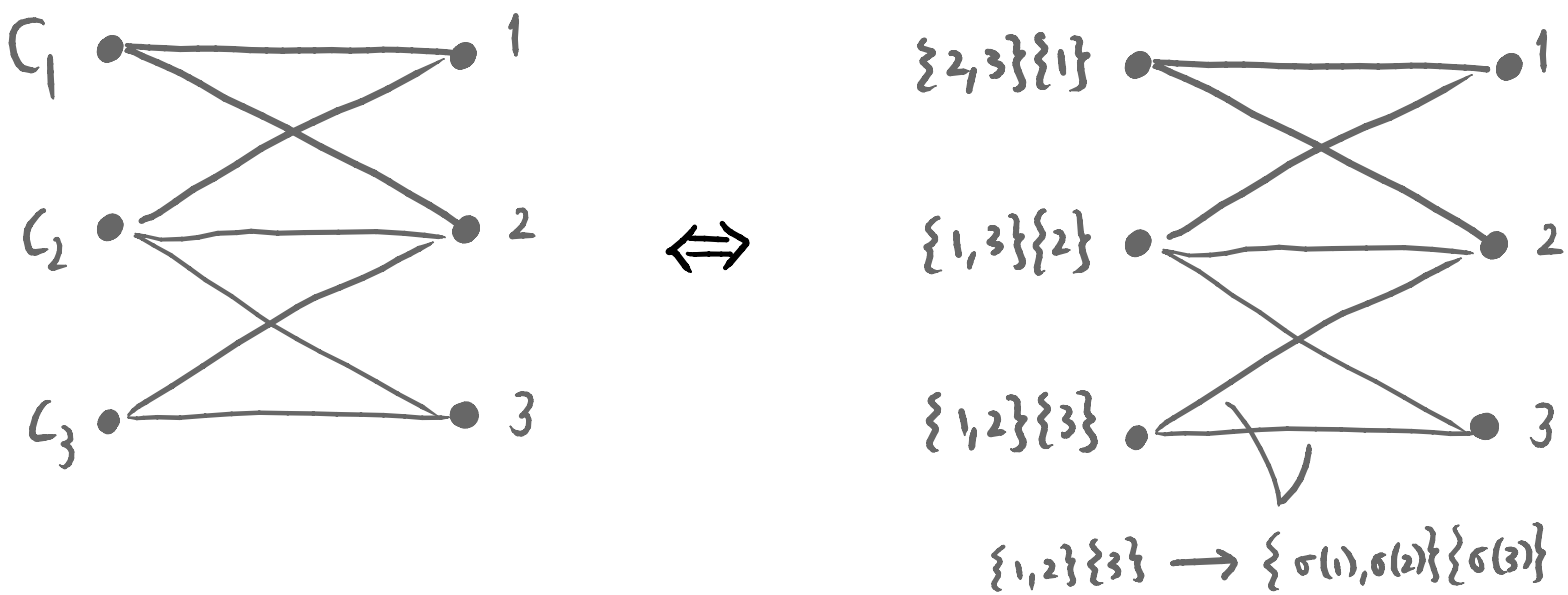
$$Q = \left[Q_{ii';jj'} \right] \equiv$$

$$\mathbb{P}\left(c_i \to j \,;\, c_{i'} \to j' \right)$$



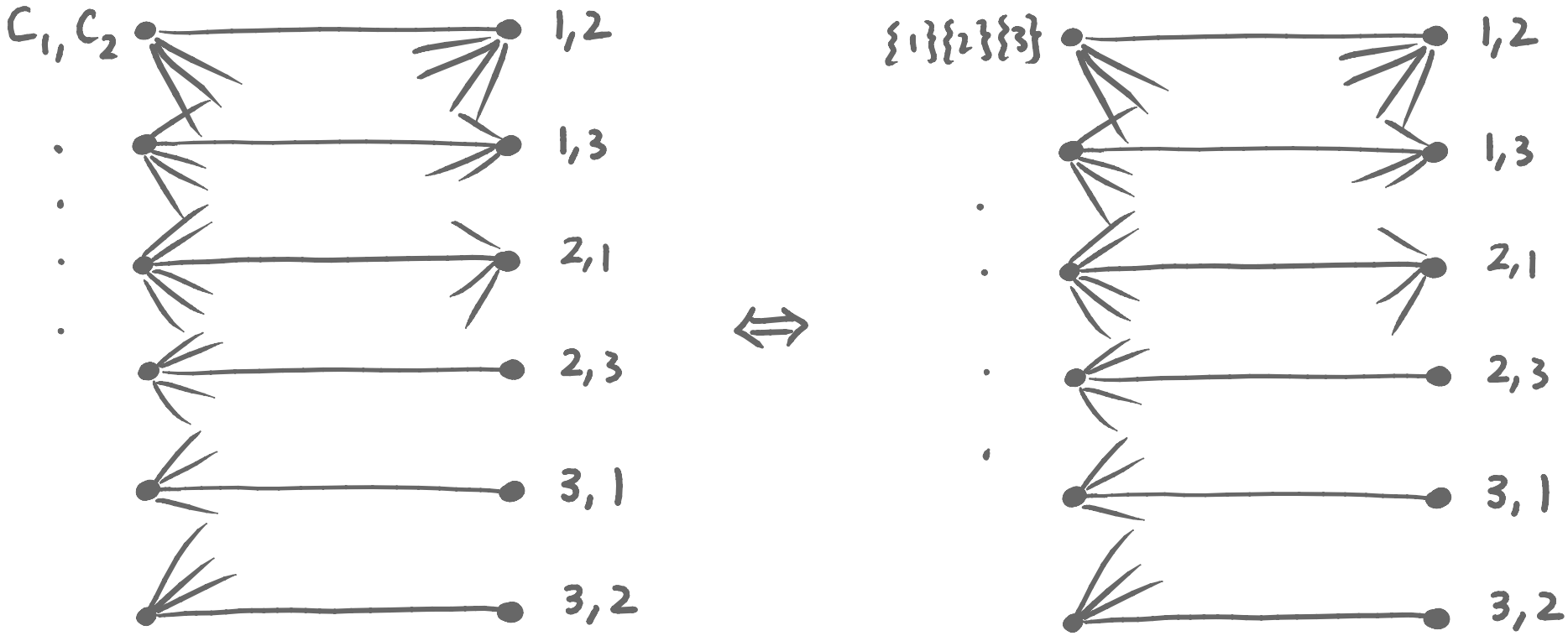
Partial Information Types

First order information: $n = 3$



Partial Information Types

Second order information: $n = 3$



Partial Information Types

Partition λ of n:

- $\lambda = (\lambda_1, \dots, \lambda_r)$, $\lambda_1 \geq \dots \geq \lambda_r \geq 1$, $\lambda_1 + \dots + \lambda_r = n$
- D_λ distinct ways to partition $\{1, \dots, n\}$ as per λ

$$D_\lambda = \frac{n!}{\prod_{i=1}^{r} \lambda_i!}$$

- Examples:
  - $\lambda = (n-1, 1)$: first order
  - $\lambda = (n-2, 1, 1)$: second order
  - $\lambda = (1, \dots, 1)$: complete information

Partial Information Types

Partition λ of n:

- $\lambda = (\lambda_1, \dots, \lambda_r)$, $\lambda_1 \geq \dots \geq \lambda_r \geq 1$, $\lambda_1 + \dots + \lambda_r = n$
- D_λ distinct ways to partition $\{1, \dots, n\}$ as per λ

λ-order partial information:

- $D_\lambda \times D_\lambda$ matrix $Q = [Q_{tt'}]$

$$Q_{tt'} = \mathbb{P}\begin{pmatrix} \lambda\text{-partition } t \\ \text{mapped } t' \end{pmatrix}$$

$$= \sum_{\sigma \in S_n} \mathbb{P}(\sigma) \cdot \mathbb{1}\{\sigma(t) = t'\}$$

Partial Information Types

Partition λ of n:

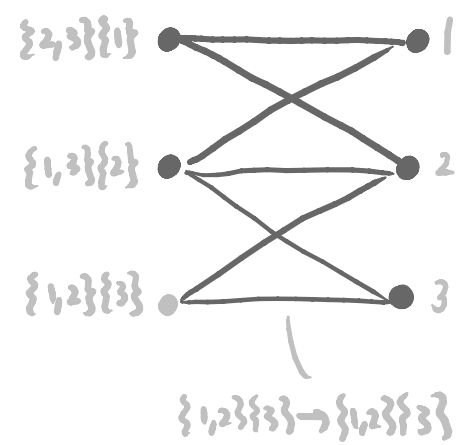
- $\lambda = (\lambda_1, \dots, \lambda_r)$, $\lambda_1 \geq \dots \geq \lambda_r \geq 1$, $\lambda_1 + \dots + \lambda_r = n$
- D_λ distinct ways to partition $\{1, \dots, n\}$ as per λ

λ-order partial information:

- $D_\lambda \times D_\lambda$ matrix $Q = [Q_{tt'}]$

$$Q_{tt'} = \mathbb{P}\begin{pmatrix} \lambda\text{-partition } t \\ \text{mapped } t' \end{pmatrix}$$

$$= \sum_{\sigma \in S_n} \mathbb{P}(\sigma) \cdot \mathbb{1}\{\sigma(t) = t'\}$$



Question

Partition λ of n:

- $\lambda = (\lambda_1, \dots, \lambda_r)$, $\lambda_1 \geq \dots \geq \lambda_r \geq 1$, $\lambda_1 + \dots + \lambda_r = n$

Given: λ-order partial information

- $D_\lambda \times D_\lambda$ matrix $Q = [Q_{tt'}]$

Task: recover the distribution over permutations, i.e.

- $\mathbb{P}(\sigma)$, $\sigma \in S_n$; possibly $n!$ distinct values.
- Likely, when support $\mathbb{P}$ is sparse, i.e.

$$K = \left| \{ \sigma : \mathbb{P}(\sigma) > 0 \} \right| \text{ is small } \left(\text{certainly} \leq D_\lambda^2 \right).$$

Question

Partition λ of n:

- $\lambda = (\lambda_1, \dots, \lambda_r)$, $\lambda_1 \geq \dots \geq \lambda_r \geq 1$, $\lambda_1 + \dots + \lambda_r = n$

Given: λ-order partial information

- $D_\lambda \times D_\lambda$ matrix $Q = [Q_{tt'}]$

Task: recover the distribution over permutations, i.e.

- $\mathbb{P}(\sigma)$, $\sigma \in S_n$; possibly $n!$ distinct values.
- Likely, when support $\mathbb{P}$ is sparse, i.e.

$$K = \left| \{ \sigma : \mathbb{P}(\sigma) > 0 \} \right| \text{ is small } \left(\text{certainly} \leq D_\lambda^2 \right).$$

Question

Recover $x \in \mathbb{R}_+^{n!}$ from

$$y = Ax$$

$\mathbb{R}_+^{D_\lambda^2}$ columns correspond to action of permutation

Occam' razor: the sparsest consistent solution.

$$\ell_0\text{-opt} \equiv \begin{cases} \min \|z\|_0 = |\{z_\sigma : z_\sigma \neq 0\}| & \text{over } z \in \mathbb{R}_+^{n!} \\ \text{subject to} \quad Az = y. \end{cases}$$

Question

$$\ell_0\text{-opt} \equiv \begin{cases} \min \ \|Z\|_0 = |\{Z_\sigma : Z_\sigma \neq 0\}| & \text{over } Z \in \mathbb{R}_+^{n!} \\ \text{subject to} \quad AZ = Y. \end{cases}$$

Recent work on when ℓ_0-opt provides correct answer

- Streaming algorithm (cf. Book by Muthukrishnan '05)
- Sparse Graph Codes (cf. Gallager '63, Sipser-Spielman '96 ....)
- Compressive Sensing (Candès, Donoho, Tao, ...(all not listed)...)

Question

$$\ell_0\text{-opt} \equiv \begin{cases} \min \ \|z\|_0 = |\{z_\sigma : z_\sigma \neq 0\}| & \text{over } z \in \mathbb{R}_+^n! \\ \text{subject to} \quad Az = y. \end{cases}$$

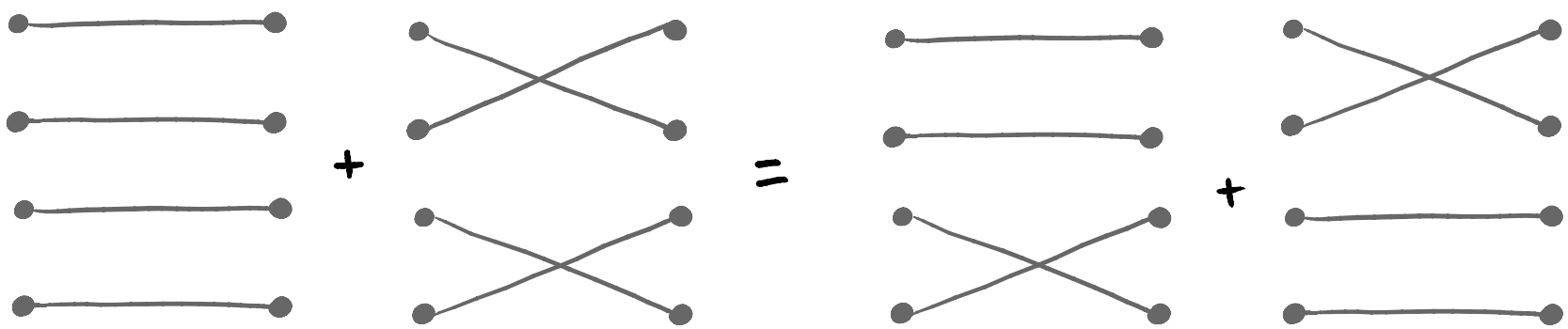
Recent work on when ℓ_0-opt provides correct answer

- Primary focus: design of A with minimal meas.
  - For us, A is given
- Key insight: if A satisfies k-RIP then
  - Then x can be recovered as long as $\|x\|_0 < k/2$

Question

$$\ell_0\text{-opt} \equiv \begin{cases} \min \ \|z\|_0 = \left|\{z_\sigma : z_\sigma \neq 0\}\right| & \text{over } z \in \mathbb{R}_+^{n!} \\ \text{subject to} \quad Az = y. \end{cases}$$

Counter-example : lack of k-RIP, $k=4$.



Question

$$\ell_0\text{-opt} \equiv \begin{cases} \min \ \|z\|_0 = \left|\{z_\sigma : z_\sigma \neq 0\}\right| & \text{over } z \in \mathbb{R}_+^{n!} \\ \text{subject to} \quad Az = y. \end{cases}$$

Questions:

- when does ℓ_0-opt recover distribution?
- How 'well' does it perform?
- Is there a simple algorithm to recover, or solve ℓ_0-opt?

Results

Random model : is counter-example very special ?

- Choose K permutations uniformly at random
- To each assign probability at random, e.g.
  - choose $q_i \in [0,1]$ uniformly, $1 \le i \le K$
  - set probabilities : $p_i = q_i / \sum_{j=1}^{K} q_j$, $1 \le i \le K$

Results

Random model : is counter-example very special ?

- Choose K permutations uniformly at random
- To each assign probability at random, e.g.
  - choose $q_i \in [0,1]$ uniformly, $1 \le i \le K$
  - set probabilities : $p_i = q_i / \sum_{j=1}^{K} q_j$, $1 \le i \le K$

Revised question : given λ-order information,

- Upto what K can distribution be recovered with probability $1-o(1)$.

Results

Theorem 1. $\lambda = (n-1, 1)$. Let, $K \leq (1-\varepsilon) n \log n$ for any $\varepsilon > 0$. Then, with probability $1 - o(1)$, ℓ_0-opt has a unique solution. Further, a simple algorithm recovers it.

Results

Theorem 1. $\lambda = (n-1, 1)$. Let, $K \leq (1-\varepsilon) n \log n$ for any $\varepsilon > 0$. Then, with probability $1 - o(1)$, ℓ_0-opt has a unique solution. Further, a simple algorithm recovers it.

$$\ell_1\text{-opt} \equiv \begin{cases} \min \ \|z\|_1 = \sum z_\sigma & \text{over } z \in \mathbb{R}_+^{n!} \\ \text{subject to} \quad Az = y. \end{cases}$$

Proposition 2. $\lambda = (n-1, 1)$. The ℓ_1-opt will admit multiple solutions with probability $1 - o(1)$.

Results

Theorem 1. $\lambda = (n-1, 1)$. Let, $K \leq (1-\varepsilon) n \log n$ for any $\varepsilon > 0$. Then, with probability $1-o(1)$, ℓ_0-opt has a unique solution. Further, a simple algorithm recovers it.

Theorem 1^c. $\lambda = (n-1, 1)$. Let, p_i take values in $\left\{ \frac{k}{s} : 0 \leq k \leq s \right\}$. Then, no algorithm can recover with probability $1-o(1)$, if

$$K \geq 3n \frac{\log(T+n)}{\log n}$$

$\rightarrow T \sim 2^{\log^2 n}$ then $K \sim n \log n$.

Results

Theorem 2. $\lambda = (n-m, m)$. With probability $1-o(1)$, ℓ_0-opt and our algorithm will recover distribution as long as

$$K \leq \frac{(1-\varepsilon)}{m!} n^m \log n, \quad \text{for any } \varepsilon > 0.$$

Here m is a constant.

More generally, for $\lambda = (\lambda_1, \ldots, \lambda_r)$ with $\lambda_1 = n - O(1)$,

$$K \sim D_\lambda \log D_\lambda.$$

Results

Theorem 3. $\lambda = (\lambda_1, \dots, \lambda_r)$: $\lambda_1 = n - o(n^{2/3})$. With prob. $1-o(1)$, ℓ_0-opt and our algorithm will recover distribution as long as

$$K \leq (1-\varepsilon) D_\lambda \log\log D_\lambda, \text{ for any } \varepsilon > 0.$$

Results

Theorem 4. $\lambda = (\lambda_1, \dots, \lambda_r)$: **general case.** With prob. $1-o(1)$, ℓ_0-opt and our algorithm will recover distribution as long as

$$K \leq D_\lambda^{\gamma(\lambda)}.$$

Definition: $\gamma(\lambda)$: $\alpha_i = \lambda_i / n$, $1 \leq i \leq r$.

$$H(\alpha) = -\sum_{i=1}^{r} \alpha_i \log \alpha_i \quad ; \quad H'(\alpha) = -\sum_{i=2}^{r} \alpha_i \log \alpha_i$$

$$\gamma(\lambda) = (1-o(1)) \frac{M H'(\alpha)}{(M+1) H(\alpha)} \quad ; \quad M = \left\lceil \frac{1}{1-\alpha_1} \right\rceil$$

Results

Theorem 4. $\lambda = (\lambda_1, \dots, \lambda_r)$: $K \leq D_\lambda^{\gamma(\lambda)}$.

$\alpha_i = \lambda_i / n$, $H(\alpha) = -\sum_{i=1}^{r} \alpha_i \log \alpha_i$; $H'(\alpha) = -\sum_{i=2}^{r} \alpha_i \log \alpha_i$

$$\gamma(\lambda) = (1 - o(1)) \frac{M H'(\alpha)}{(M+1) H(\alpha)} \quad ; \quad M = \left\lceil \frac{1}{1-\alpha_1} \right\rceil$$

Thms 1-3 : $\alpha_1 \to 1^-$: can be shown that $\gamma(\lambda) \to 1$.

That is, $K \sim D_\lambda$.

Results

Theorem 4. $\lambda = (\lambda_1, \dots, \lambda_r)$: $K \leq D_\lambda^{\gamma(\lambda)}$.

$\alpha_i = \lambda_i / n$, $H(\alpha) = -\sum_{i=1}^{r} \alpha_i \log \alpha_i$; $H'(\alpha) = -\sum_{i=2}^{r} \alpha_i \log \alpha_i$

$$\gamma(\lambda) = (1 - o(1)) \frac{M H'(\alpha)}{(M+1) H(\alpha)} \quad ; \quad M = \left\lceil \frac{1}{1-\alpha_1} \right\rceil$$

Other extreme : $\alpha_1 \to 0$: can be shown that $\gamma(\lambda) \to \frac{1}{2}$.

That is, $K \sim \sqrt{D_\lambda}$.

In compliance with Birthday Paradox : $\lambda = (1, \dots, 1)$.

Algorithm

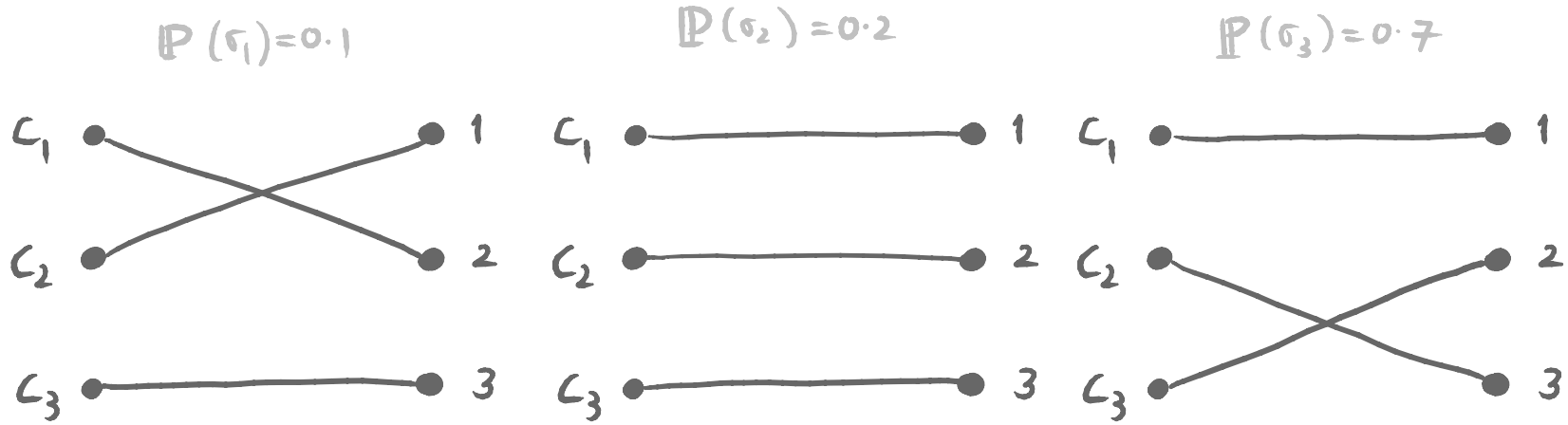
Sufficiency Conditions:

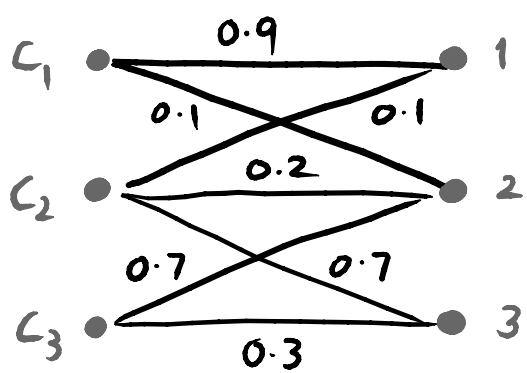
1. **Unique witness:** for any σ with $p_\sigma > 0$, there exists (t, t') s.t. $\sigma(t) = t'$ but for all other σ' with $p_{\sigma'} > 0$, $\sigma'(t) \neq t'$.

2. **Linear independence:** given K permutations with probabilities p_i, $1 \leq i \leq K$,

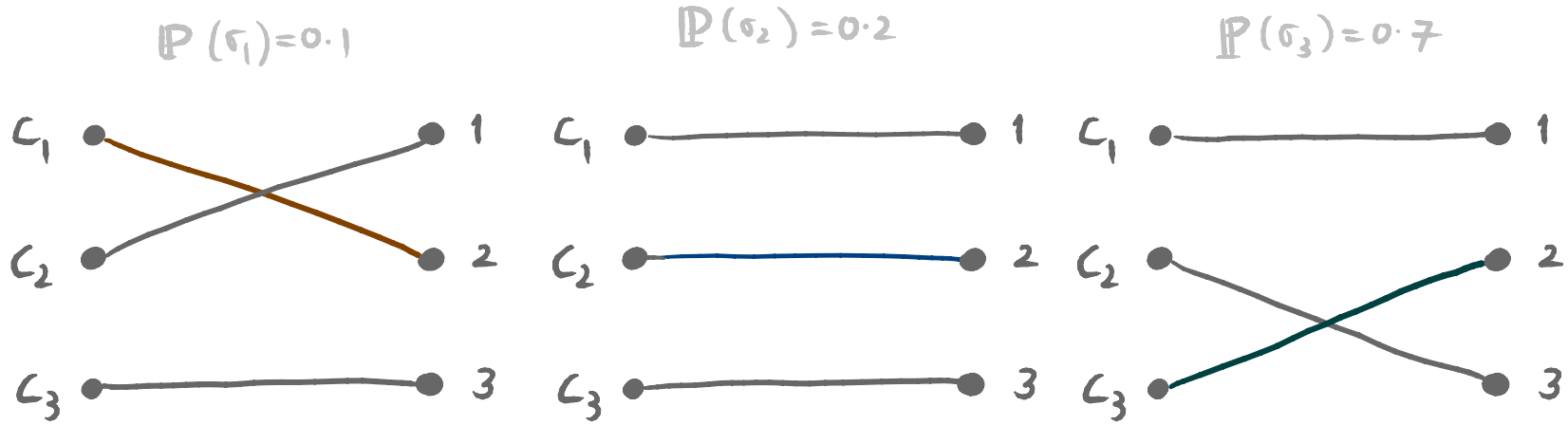
$$\sum c_i p_i \neq 0 \text{ for any } c_i \in \{-K, \ldots, K\}, 1 \leq i \leq K.$$

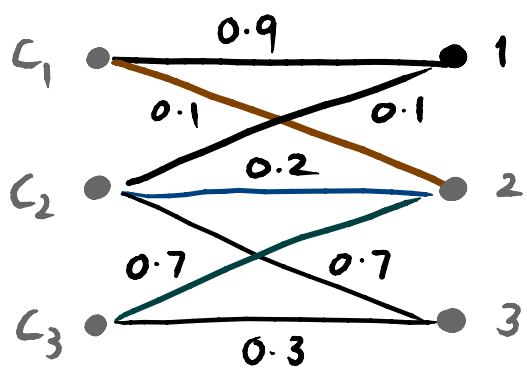
Algorithm : $\lambda = (n-1, 1)$



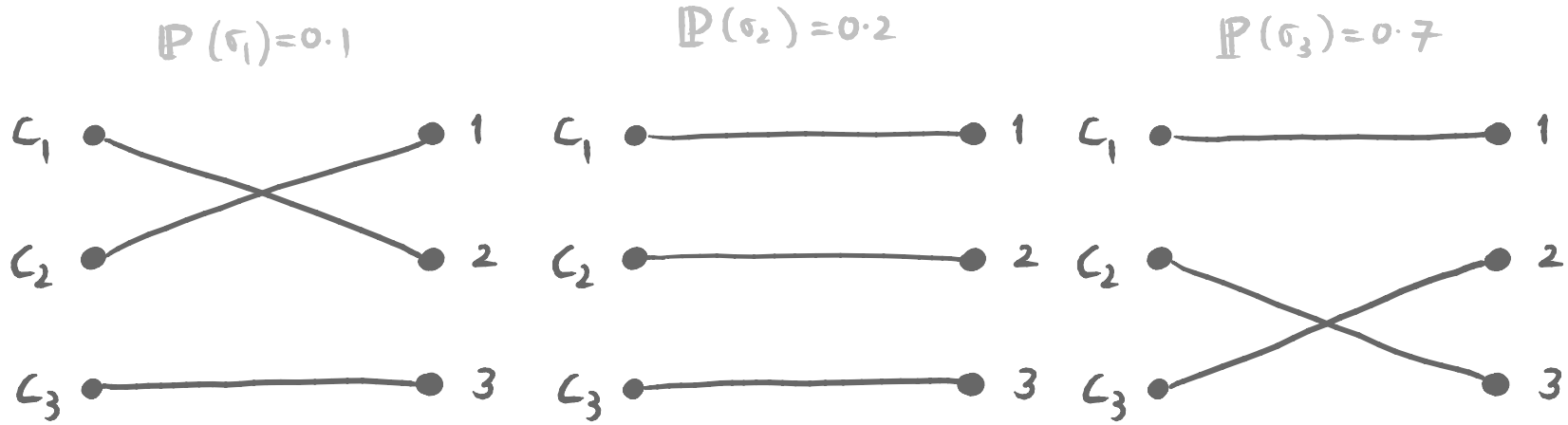


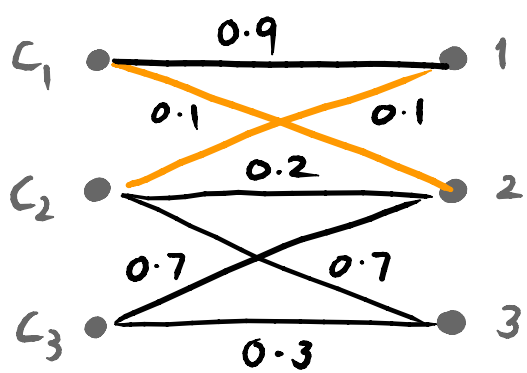
Algorithm : $\lambda = (n-1, 1)$





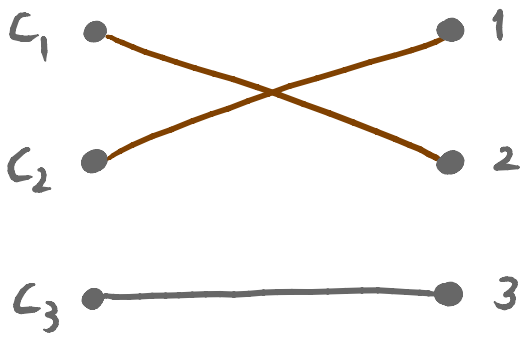
Algorithm : $\lambda = (n-1, 1)$



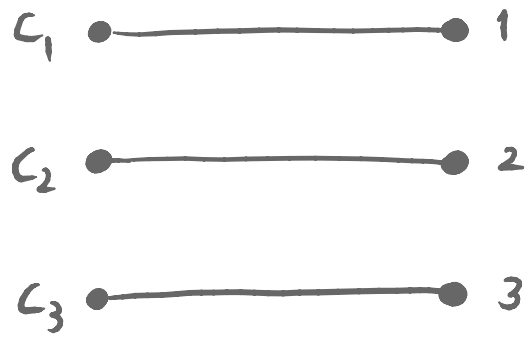


Algorithm : $\lambda = (n-1, 1)$

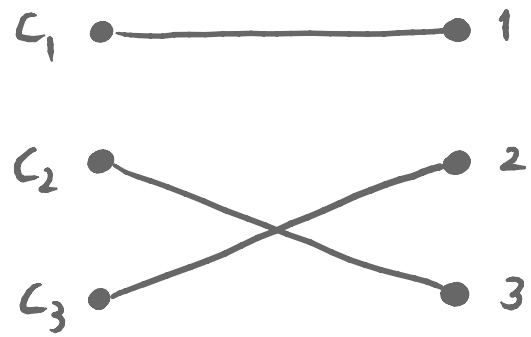
$\mathbb{P}(\sigma_1) = 0.1$

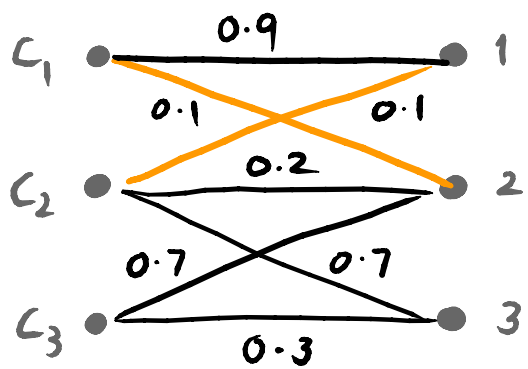


$\mathbb{P}(\sigma_2) = 0.2$



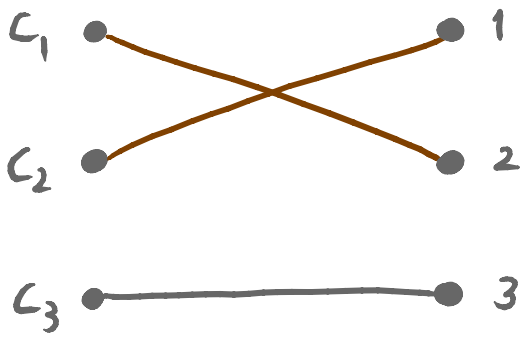
$\mathbb{P}(\sigma_3) = 0.7$



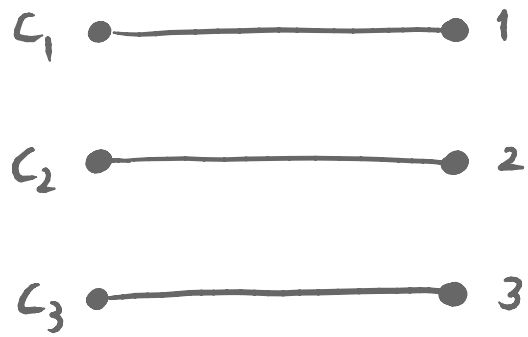


Algorithm : $\lambda = (n-1, 1)$

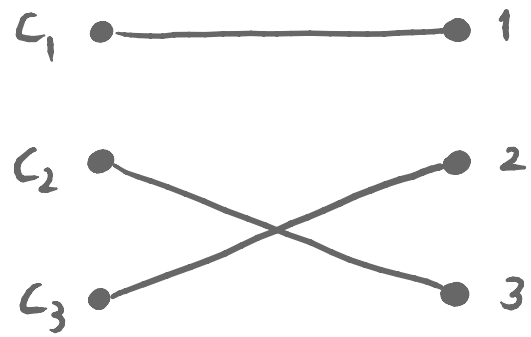
$\mathbb{P}(\sigma_1) = 0.1$

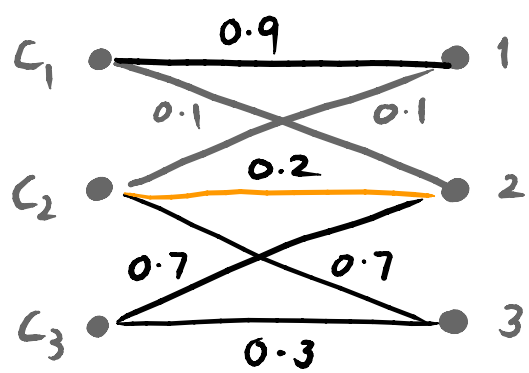


$\mathbb{P}(\sigma_2) = 0.2$



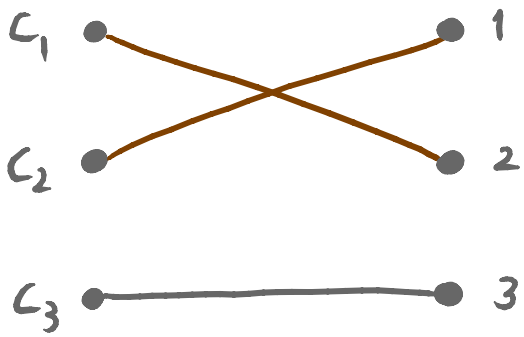
$\mathbb{P}(\sigma_3) = 0.7$



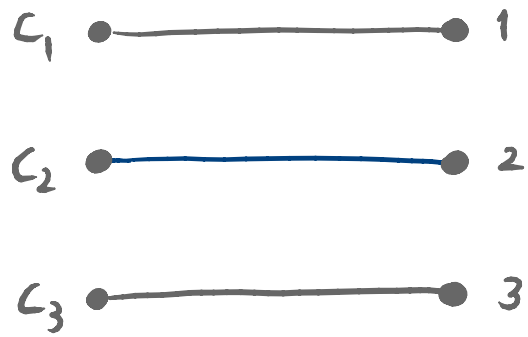


Algorithm : $\lambda = (n-1, 1)$

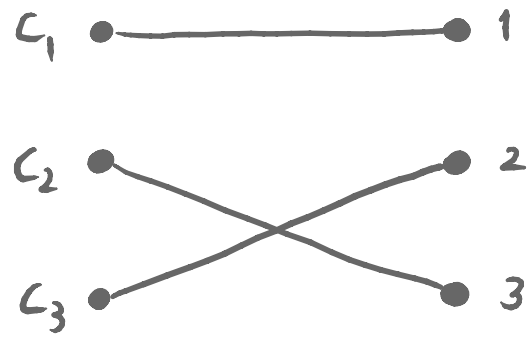
$\mathbb{P}(\sigma_1) = 0.1$

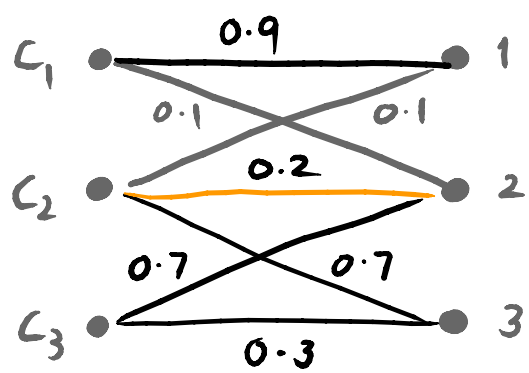


$\mathbb{P}(\sigma_2) = 0.2$



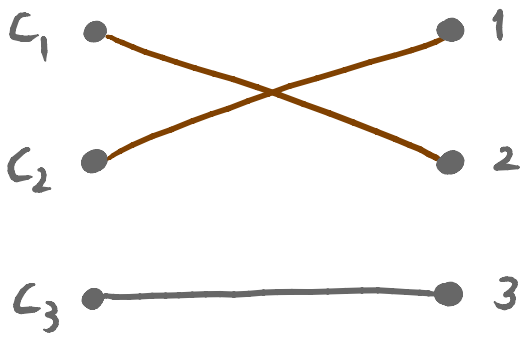
$\mathbb{P}(\sigma_3) = 0.7$



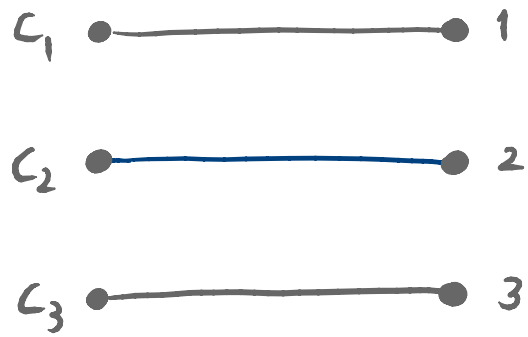


Algorithm : $\lambda = (n-1, 1)$

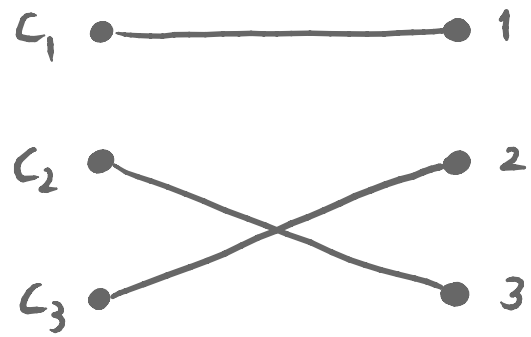
$\mathbb{P}(\sigma_1) = 0.1$

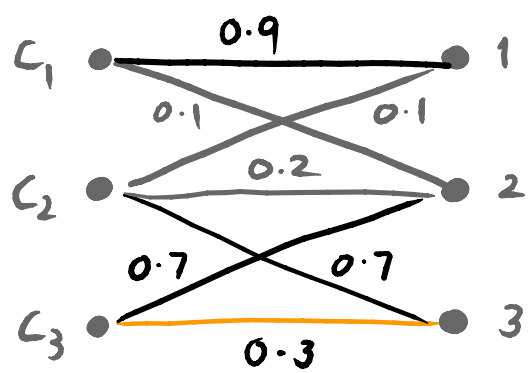


$\mathbb{P}(\sigma_2) = 0.2$



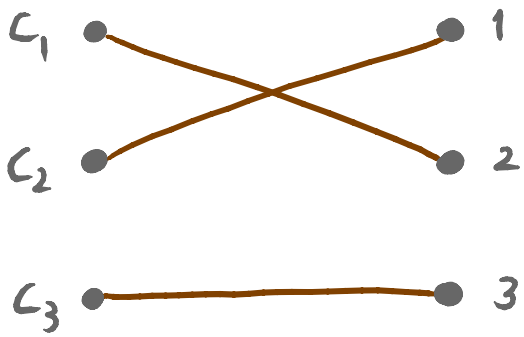
$\mathbb{P}(\sigma_3) = 0.7$



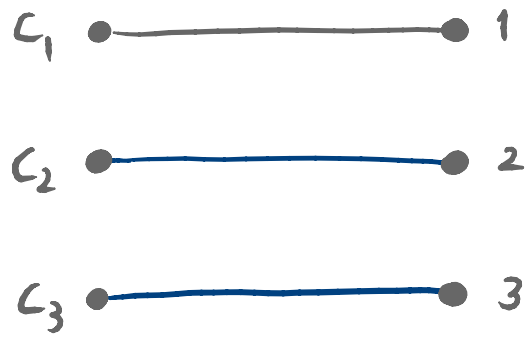


Algorithm : $\lambda = (n-1, 1)$

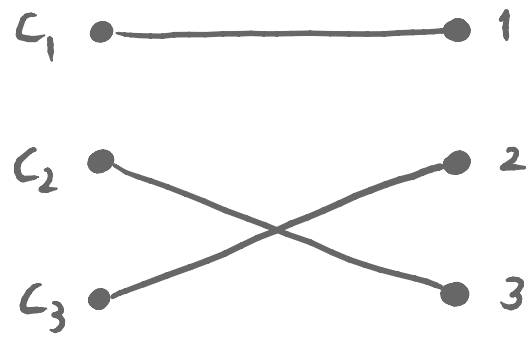
$\mathbb{P}(\sigma_1) = 0.1$

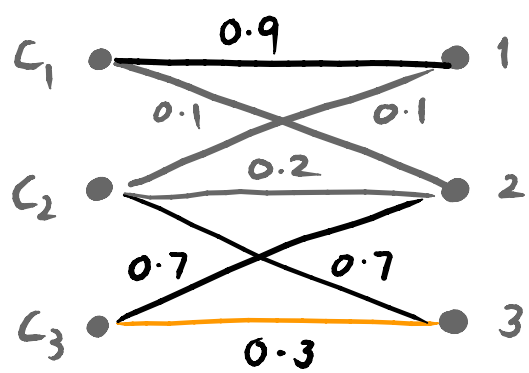


$\mathbb{P}(\sigma_2) = 0.2$



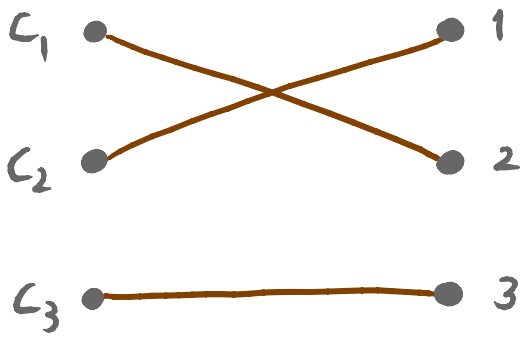
$\mathbb{P}(\sigma_3) = 0.7$



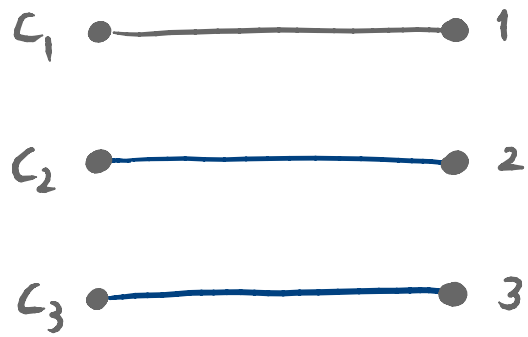


Algorithm : $\lambda = (n-1, 1)$

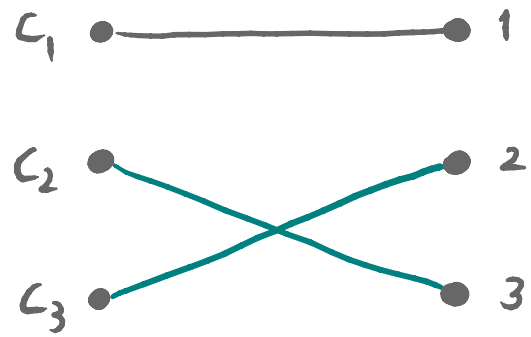
$\mathbb{P}(\sigma_1) = 0.1$

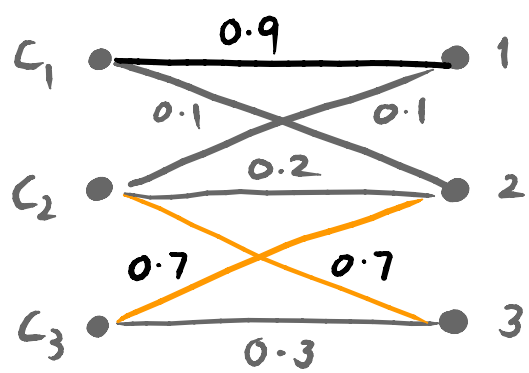


$\mathbb{P}(\sigma_2) = 0.2$



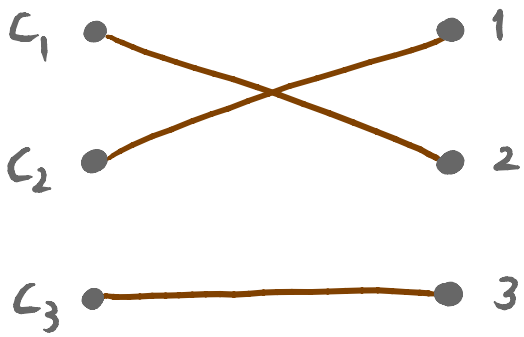
$\mathbb{P}(\sigma_3) = 0.7$



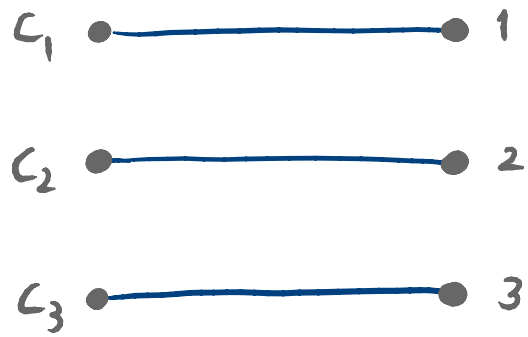


Algorithm : $\lambda = (n-1, 1)$

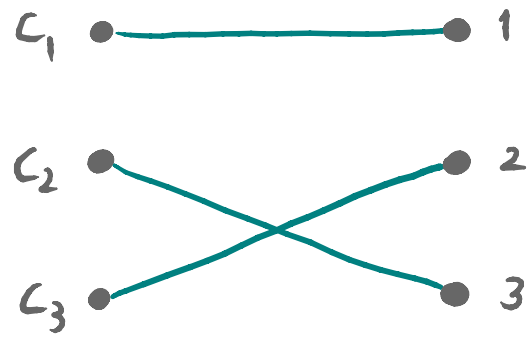
$\mathbb{P}(\sigma_1) = 0.1$

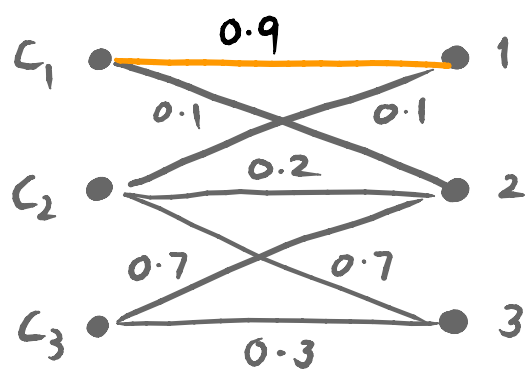


$\mathbb{P}(\sigma_2) = 0.2$



$\mathbb{P}(\sigma_3) = 0.7$





Discussion

It is possible to recover popular rankings

- when their support is reasonably sparse
- And, recoverability gracefully depends on the type of partial information.

Robustness

- A natural optimization formulation seem to work surprisingly well for learning Customer Choices
- More work needs to done (data ?)

Discussion

It is possible to recover popular rankings

- when their support is reasonably sparse
- And, recoverability gracefully depends on the type of partial information.

Other types of data : e.g.

- comparison : $K \sim \log n$
- comparison plus favorite : $K \sim \sqrt{n} \log n$!

Discussion

It is possible to recover popular rankings

- when their support is reasonably sparse
- And, recoverability gracefully depends on the type of partial information.

Exciting set of questions:

- And, lots of applications.