The Parameterized Complexity of **Matrix Completion**

Robert Ganian

Joint work with: Eduard Eiben Iyad Kanj

Sebastian Ordyniak Stefan Szeider

ALGORITHMS AND COMPLEXITY GROUP

Matrix Completion

- Input: Matrix over **GF(p)** with **missing entries**

0	0	2	1	2	1
1	4	0	2	*	1
1	4	2	3	4	2
1	*	0	*	3	*
1	4	4	4	*	3

p = 5

  – General Task: Fill in entries to minimize **some measure**
    - Exploits expected similarities between rows of the matrix

Matrix Completion: Basic Measures

- Input: Matrix over **GF(p)** with **missing entries**

0	0	2	1	2	1
1	4	0	2	*	1
1	4	2	3	4	2
1	*	0	*	3	*
1	4	4	4	*	3

p = 5

  - Task 1: Fill in entries to minimize the **rank**
    - Rank Matrix Completion Problem (**RMC**)

Matrix Completion: Basic Measures

- Input: Matrix over **GF(p)** with **missing entries**

0	0	2	1	2	1
1	4	0	2	**2**	1
1	4	2	3	4	2
1	**0**	0	**0**	3	**0**
1	4	4	4	**1**	3

p = 5

  – Task 1: Fill in entries to minimize the **rank**
    - Rank Matrix Completion Problem (**RMC**)

Matrix Completion: Basic Measures

- Input: Matrix over **GF(p)** with **missing entries**

0	0	2	1	2	1
1	4	0	2	*	1
1	4	2	3	4	2
1	*	0	*	3	*
1	4	4	4	*	3

p = 5

  - Task 1: Fill in entries to minimize the **rank**
    - Rank Matrix Completion Problem (**RMC**)
  - Task 2: Fill in entries to minimize the **# of distinct rows**
    - Distinct Row Matrix Completion Problem (**DRMC**)

Matrix Completion: Basic Measures

- Input: Matrix over **GF(p)** with **missing entries**

0	0	2	1	2	1
1	4	0	2	**3**	1
1	4	2	3	4	2
1	**4**	0	**2**	3	**1**
1	4	4	4	**0**	3

p = 5

  – Task 1: Fill in entries to minimize the **rank**
    - Rank Matrix Completion Problem (**RMC**)
  – Task 2: Fill in entries to minimize the **# of distinct rows**
    - Distinct Row Matrix Completion Problem (**DRMC**)

Motivation

- Fundamental problems, well studied
  - Especially in ML and recommender systems

- Example 1: **Netflix Problem**
  - Entries are movie ratings
  - constant-size **p**

p-RMC, p-DRMC

- Example 2: **Triangulation from Incomplete Data**
  - Entries represent distances, large **p**

RMC, DRMC

Aim

- Exact algorithms
- Worst-case complexity
- Runtime guarantees

- Understanding the complexity of **(p-)RMC**, **(p-)DRMC**

  fine-grained

  – What really makes the problems hard?
  – When can they be solved more efficiently?

NP-complete!

Parameterized Complexity?

Considered Parameters

0	0	2	1	2	1
1	4	0	2	*	1
1	4	2	3	4	2
1	*	0	*	3	*
1	4	4	4	*	3

Number of *?

... too restrictive

Considered Parameters

0	0	2	1	2	1
1	4	0	2	*	1
1	4	2	3	4	2
1	*	0	*	3	*
1	4	4	4	*	3

Number of *?
... too restrictive

- Number of **rows** where * occur (**row**)
  - **k** small ↔ a few new users in the **Netflix** setting

Considered Parameters

0	0	2	1	2	1
1	4	0	2	*	1
1	4	2	3	4	2
1	*	0	*	3	*
1	4	4	4	*	3

Number of *?
... too restrictive

- Number of **rows** where * occur (**row**)
  - **k** small ↔ a few new users in the **Netflix** setting
- Number of **columns** where * occur (**col**)
  - **k** small ↔ a few new movies in the **Netflix** setting

Considered Parameters

0	0	2	1	2	1
1	4	0	2	*	1
1	4	2	3	4	2
1	*	0	*	3	*
1	4	4	4	*	3

Number of *?
... too restrictive

- Number of **rows** where * occur (**row**)
  - **k** small ⟷ a few new users in the **Netflix** setting
- Number of **columns** where * occur (**col**)
  - **k** small ⟷ a few new movies in the **Netflix** setting
- Number of **columns** and **rows** covering all * (**comb**)
  - Better than **col** and **row**

Results

- Rank Minimization vs. Distinct Row Minimization
  - **Opinion poll**: Which is harder?

	row	col	comb
p-RMC			
p-DRMC			

Results

- **Rank Minimization** vs. **Distinct Row Minimization**
  - **Opinion poll**: Which is harder?

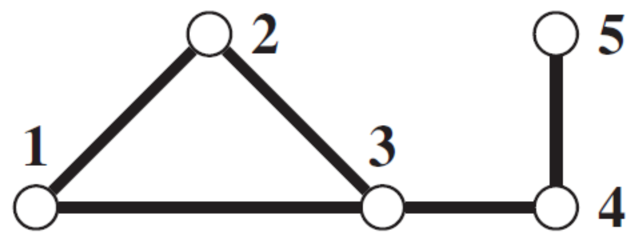
	row	col	comb
p-RMC	FPT*	FPT	FPT$^\star_R$
p-DRMC	FPT	FPT	FPT*

- $\star$ – explicitly proven results (others follow)
- ***R*** – randomized
- Also works when ***p*** is considered a parameter

Proof Technique: DRMC

- Graph representation of **compatibilities** between rows in **(p-)DRMC** instances

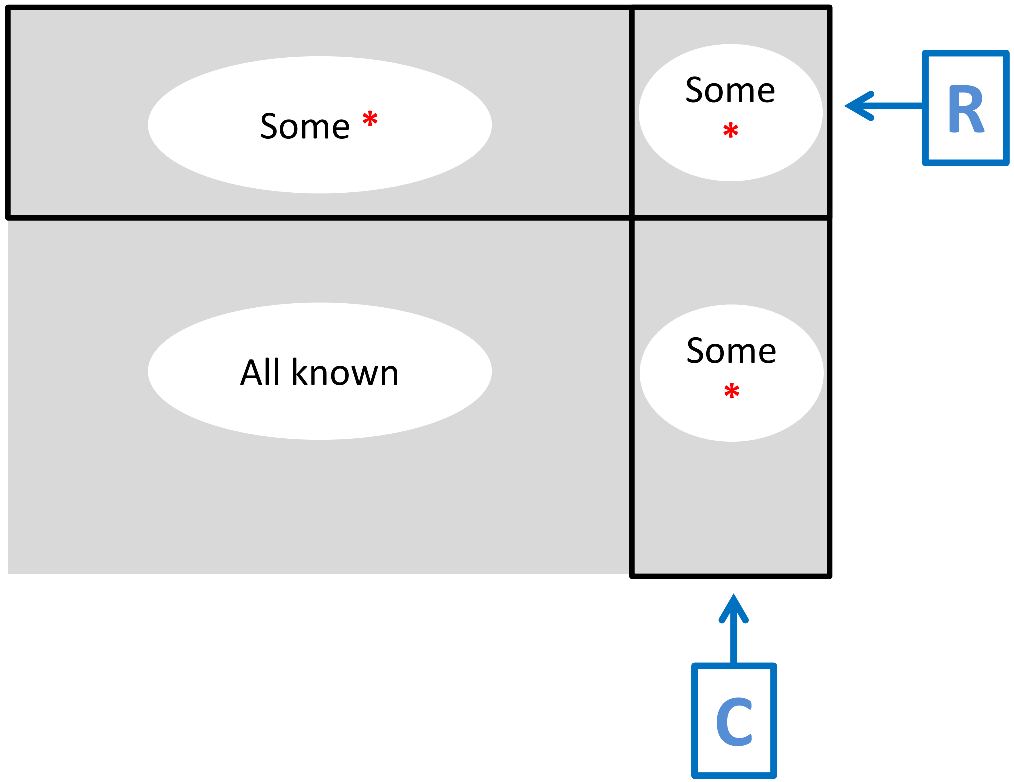
$$\begin{pmatrix} 1 & \bullet & 0 & \bullet & \bullet & 1 \\ 1 & 0 & 0 & 1 & \bullet & \bullet \\ 1 & 0 & \bullet & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & \bullet \\ 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$



Small treewidth → **(p-)DRMC** can be solved efficiently

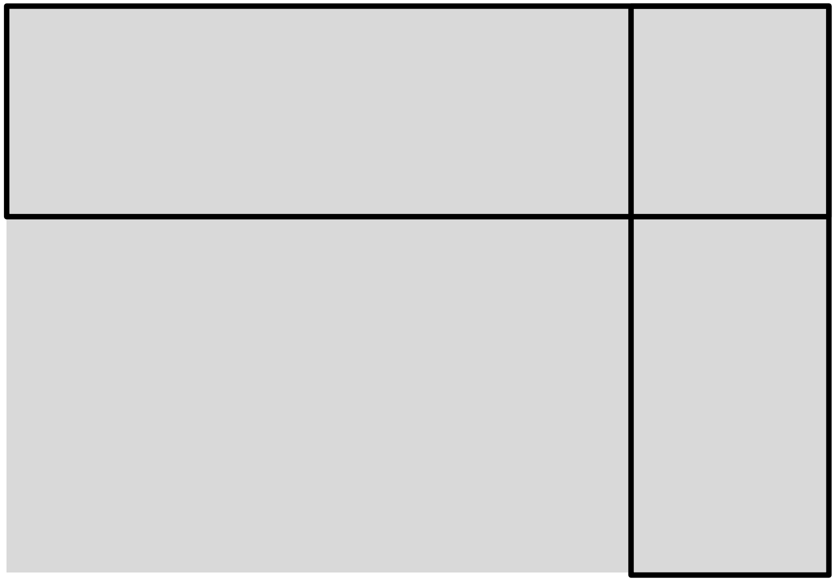
- DRMC solution ↔ Minimum **Clique-Cover** in graph
- **row**, **col** and **comb** → bounded treewidth $(k + pk)$

Proof Technique: RMC

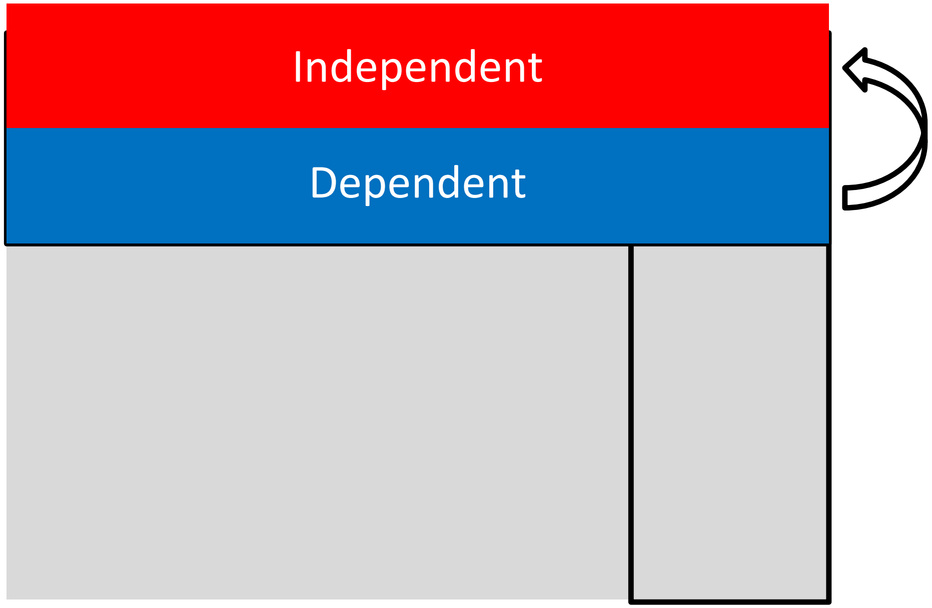


- Can permute rows and columns as above

Proof Technique: RMC

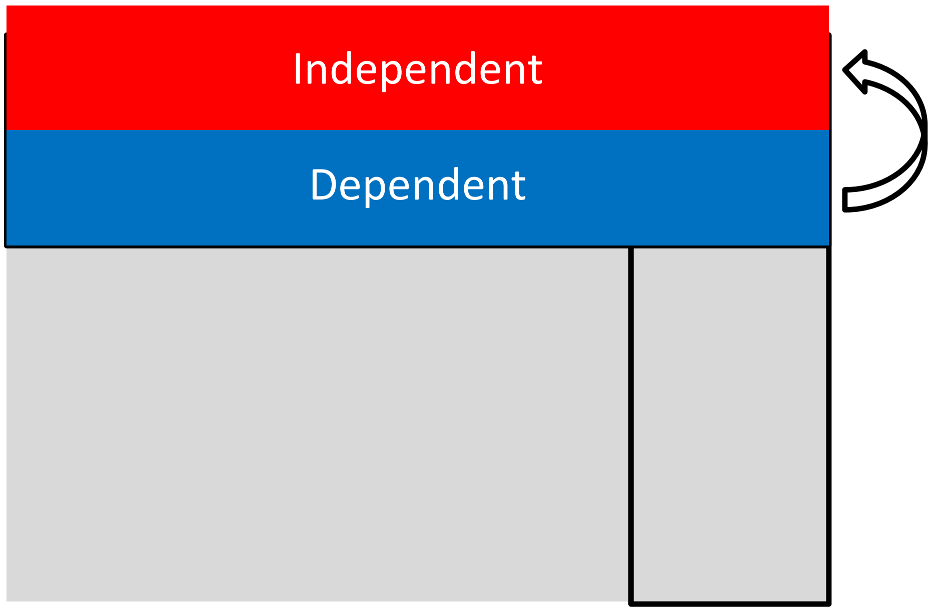


Proof Technique: RMC



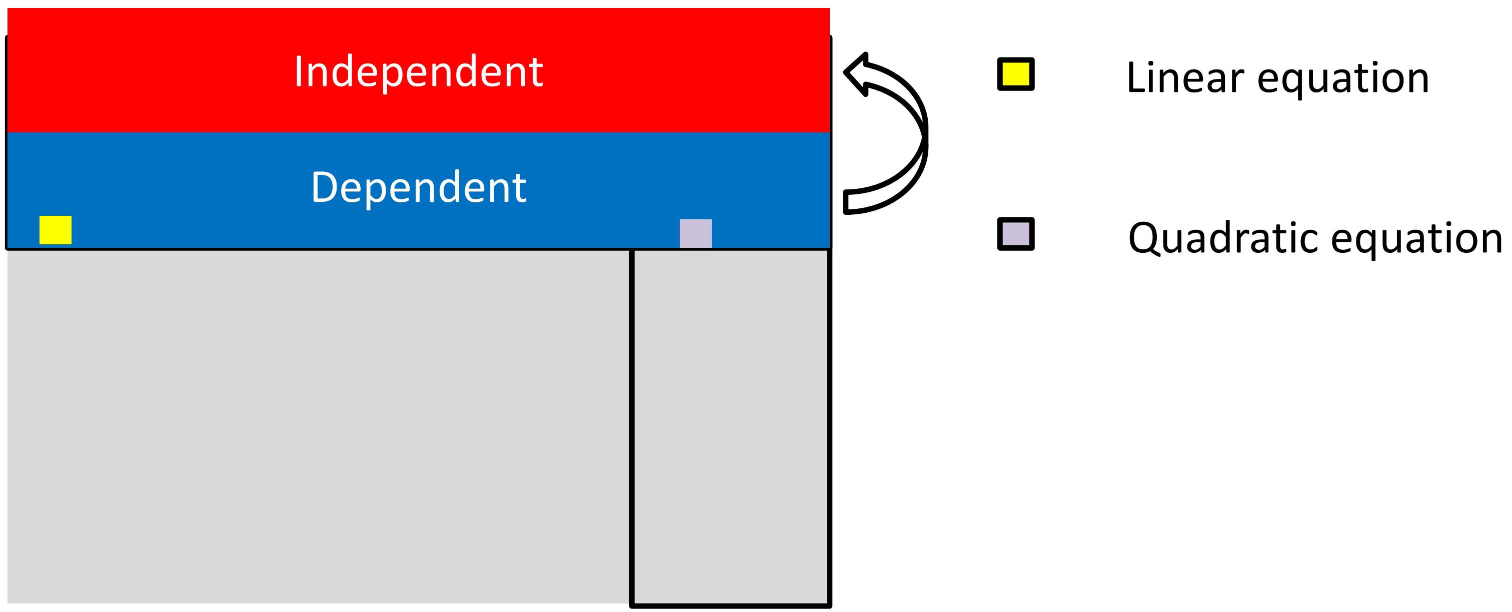
- **Step 1**: Branch into (in)dependent rows in **R**
  - Also branch to determine dependency factors in **R**
  - Same for **C**

Proof Technique: RMC



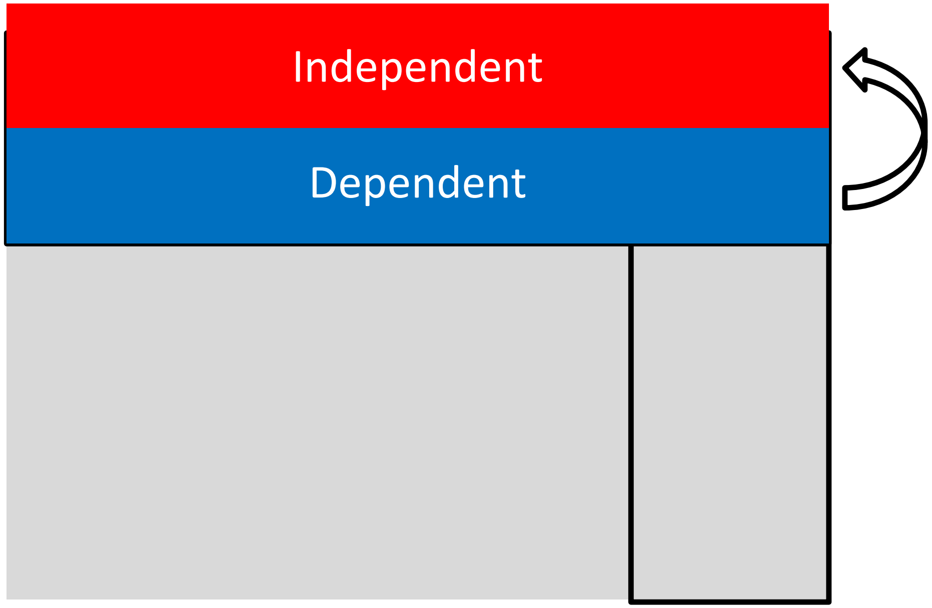
- **Step 2**: Verify branch (are dependent rows ok?)

Proof Technique: RMC



- **Step 2**: Verify branch (are dependent rows ok?)
  - Solving a set of linear/quadratic equations
  - Linear equations: Preprocess to remove
  - Quadratic equations: Only few, admit p^{k^2} algorithm

Proof Technique: RMC



- **Step 3**: Output branch with the least independent rows/columns among **C** and **R**

What about higher domains (p)?

- **Rank Minimization** vs. **Distinct Row Minimization**
  - **Opinion poll**: Which is harder?

	row	col	comb
p-RMC	FPT*	FPT	FPT$^\star_R$
p-DRMC	FPT	FPT	FPT*
RMC	XP*		
DRMC	FPT*		

What about higher domains (p)?

- Rank Minimization vs. Distinct Row Minimization
  - **Opinion poll**: Which is harder?

	row	col	comb
p-RMC	FPT*	FPT	FPT$^\star_R$
p-DRMC	FPT	FPT	FPT*
RMC	XP*	XP	XP$^\star_R$
DRMC	FPT*	paraNP-h*	paraNP-h

MC: Advanced Measures

- Example:

0	1	1	0	0	1
0	1	1	1	0	0
1	1	1	1	0	1
0	*	*	*	1	1

  – 1 means user (row) likes an item (column)
  – How would you complete the missing entries?

MC: Advanced Measures

- Example:

0	1	1	0	0	1
0	1	1	1	0	0
1	1	1	1	0	1
0	**1**	**1**	**1**	1	1

  - 1 means user (row) likes an item (column)
  - How would you complete the missing entries?
    - For **DRMC** and **RMC** it doesn't matter…
    - To capture this intuition, we need *clustering*
      - Complete matrix so as to get only "a few, similar" clusters

Matrix Completion: Clustering

- Input:
  - Boolean Matrix **M** (can be lifted to fixed domain)
  - number of clusters **k**
  - Hamming (or arithmetic) distance within cluster **r**
  - **comb** (or **row** or **col**)
- Actually 3 problems (based on Clustering variant)
  - **IN-Clustering**: Partition rows into **k** clusters, each made of rows with distance **≤r** from a center (a row in **M**)
  - **ANY-Clustering**: Same, but centers need not be in **M**
  - **PAIR-Clustering**: No centers, **r** bounds pairwise distance

Matrix Clustering

- Unlike **DRMC** and **RMC**, all 3 clustering variants are **NP-hard** even if all entries are known
  - Luckily, both **k** (desired # of clusters) and **r** (distances) are well-motivated parameters

Parameter:	k	r
In-Clustering	W[2]-c	paraNP-c
Any/Pair-Clustering	paraNP-c	paraNP-c

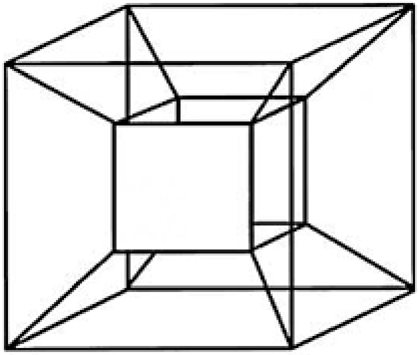
Matrix Clustering

- Unlike **DRMC** and **RMC**, all 3 clustering variants are **NP-hard** even if all entries are known
  - Luckily, both **k** (desired # of clusters) and **r** (distances) are well-motivated parameters

Parameter:	k	r	$k+r$
IN-CLUSTERING	W[2]-c	paraNP-c	FPT
ANY/PAIR-CLUSTERING	paraNP-c	paraNP-c	FPT

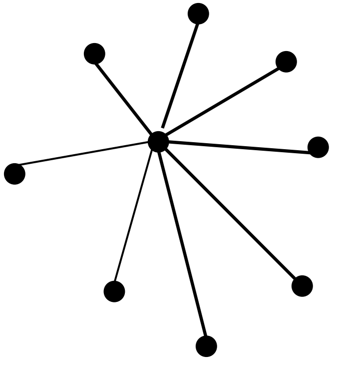
Matrix Clustering[r+k]

- Much harder than the previous two algorithms
- Here: just a brief, high-level sketch showing the ideas
- Equivalent to graph problems on powers of (induced subgraphs of) **hypercubes**
- Technique: **Kernelization**



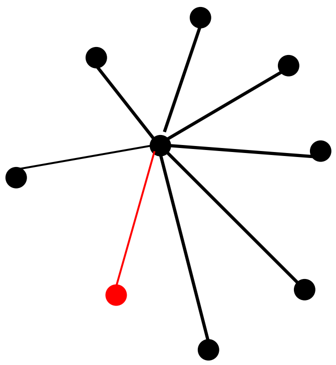
Matrix Clustering[r+k]

- **Step 1**: Reduce degree
  - – Irrelevant “vertex” technique



Matrix Clustering[r+k]

- **Step 1**: Reduce degree
  - Irrelevant “vertex” technique
  - **Sunflower Lemma**



- **Outcome**: each row has at most **f**(**r**+**k**)-many rows at distance **≤r**
  - For **IN-Clustering**: Red-Blue Dominating Set

Matrix Clustering[r+k]

- **Step 2**:
  - If #**rows** is too large, reject (because of **Step 1**)
  - If #**rows** is parameter-bounded… consider:

0	0	0	0	0	0	0	0	0	0	0	0	. . .
1	1	1	1	1	1	1	1	1	1	1	1	

  - Because of *connectivity*, two rows cannot differ in many coordinates
    - Stronger claim: the # of “important coordinates” is bounded
- **Outcome**: (exponential) kernel

Matrix Completion to Clustering

Parameter:	k	r	$k+r$
In-Clustering	W[2]-c	paraNP-c	FPT
Any/Pair-Clustering	paraNP-c	paraNP-c	FPT

Matrix Completion to Clustering

- By extending these techniques, we get:

Parameter:	k	r	$k + r$
IN-CLUSTERING	W[2]-c	paraNP-c	FPT
ANY/PAIR-CLUSTERING	paraNP-c	paraNP-c	FPT

Matrix Completion to Clustering

- By extending these techniques, we get:

Parameter:	k	r	$k + r$	$k + r + \texttt{cover}$
IN-CLUSTERING	W[2]-c	paraNP-c	FPT	N/A
ANY/PAIR-CLUSTERING	paraNP-c	paraNP-c	FPT	N/A
IN/ANY/PAIR-CLUSTERING$^{\square}$	paraNP-c	paraNP-c	paraNP-c	FPT

Concluding Notes

- **Matrix Completion** is very well-studied in other fields
  - Google hits:

Matrix Completion: ± 273,000
Vertex Cover: ± 261,000
Hamiltonian cycle: ± 177,000

- Would be interesting to see some practical work on **MC**
  - Lots done on finding/approximating the "*right measure*"
  - But how about efficiently solving the problem for simple measures?
    - **Low-rank Matrix Completion** well studied, but others…?

Concluding Notes

	row	col	comb
p-RMC	$\mathbf{FPT}^\star$	**FPT**	$\mathbf{FPT}^\star_R$
p-DRMC	**FPT**	**FPT**	$\mathbf{FPT}^\star$
RMC	$\mathbf{XP}^\star$	**XP**	$\mathbf{XP}^\star_R$
DRMC	$\mathbf{FPT}^\star$	**paraNP**-h*	**paraNP**-h

- No lower bounds for **RMC**
- Can we derandomize?
  - Requires a deterministic algorithms for **k** quadratic equations over many variables…

Thank you for your attention

Questions?

	row	col	comb
p-RMC	$\mathsf{FPT}^\star$	FPT	$\mathsf{FPT}^\star_R$
p-DRMC	FPT	FPT	$\mathsf{FPT}^\star$
RMC	$\mathsf{XP}^\star$	XP	$\mathsf{XP}^\star_R$
DRMC	$\mathsf{FPT}^\star$	paraNP-h*	paraNP-h