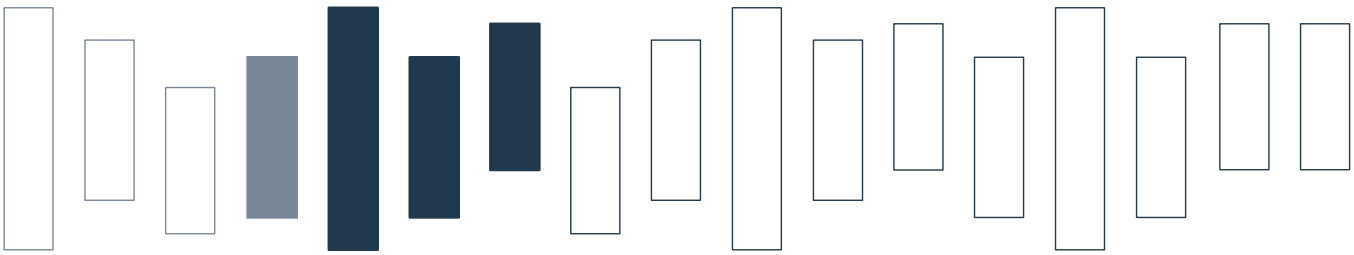
Technical Report AC-TR-18-002

June 2018

Parameterized Algorithms for the Matrix Completion Problem

Robert Ganian, Iyad Kanj, Sebastian Ordyniak, Stefan Szeider



This is the authors' copy of a paper that will appear in the proceedings of the 35th International Conference on Machine Learning (ICML'18), Stockholm, Sweden. Proceedings of Machine Learning Research (PMLR) vol. 80, 2018

www.ac.tuwien.ac.at/tr

Parameterized Algorithms for the Matrix Completion Problem

Robert Ganian [1] **Iyad Kanj** [2] **Sebastian Ordyniak** [3] **Stefan Szeider** [1]

Abstract

We consider two matrix completion problems, in which we are given a matrix with missing entries and the task is to complete the matrix in a way that (1) minimizes the rank, or (2) minimizes the number of distinct rows. We study the parameterized complexity of the two aforementioned problems with respect to several parameters of interest, including the minimum number of matrix rows, columns, and rows plus columns needed to cover all missing entries. We obtain new algorithmic results showing that, for the bounded domain case, both problems are fixed-parameter tractable with respect to all aforementioned parameters. We complement these results with a lower-bound result for the unbounded domain case that rules out fixed-parameter tractability w.r.t. some of the parameters under consideration.

1. Introduction

Problem Definition and Motivation. We consider the matrix completion problem, in which we are given a matrix $\mathbf{M}$ (over some field that we also refer to as the *domain* of the matrix) with missing entries, and the goal is to complete the entries of $\mathbf{M}$ so that to optimize a certain measure. There is a wealth of research on this fundamental problem (Candès & Plan, 2010; Candès & Recht, 2009; Candès & Tao, 2010; Elhamifar & Vidal, 2013; Hardt et al., 2014; Fazel, 2002; Keshavan et al., 2010a;b; Recht, 2011; Saunderson et al., 2016) due to its ubiquitous applications in recommender systems, machine learning, sensing, computer vision, data science, and predictive analytics, among others. In these areas, the matrix completion problem naturally arises after observing a sample from the set of entries of a low-rank matrix, and attempting to recover the missing entries with the goal of optimizing a certain measure. In this paper, we focus our study on matrix completion with respect to two measures (considered separately): (1) minimizing the rank of the completed matrix, and (2) minimizing the number of distinct rows of the completed matrix.

The first problem we consider—matrix completion w.r.t. rank minimization—has been extensively studied, and is often referred to as the low-rank matrix completion problem (Candès & Plan, 2010; Candès & Recht, 2009; Candès & Tao, 2010; Hardt et al., 2014; Fazel, 2002; Keshavan et al., 2010a;b; Recht, 2011; Saunderson et al., 2016). A celebrated application of this problem lies in the recommender systems area, where it is known as the Netflix problem (net). In this user-profiling application, an entry of the input matrix represents the rating of a movie by a user, where some entries could be missing. The goal is to predict the missing entries so that the rank of the complete matrix is minimized.

The low-rank matrix completion problem is known to be **NP**-hard, even when the matrix is over the field $\mathrm{GF}(2)$ (*i.e.*, each entry is 0 or 1), and the goal is to complete the matrix into one of rank 3 (Peeters, 1996). A significant body of work on the low-rank matrix completion problem has centered around proving that, under some feasibility assumptions, the matrix completion problem can be solved efficiently with high probability (Candès & Recht, 2009; Recht, 2011). These feasibility assumptions are: (1) low rank; (2) incoherence; and (3) randomness (Hardt et al., 2014). Hardt *et al.* (2014) argue that feasibility assumption (3), which states that the subset of determined entries in the matrix is selected uniformly at random and has a large (sampling) density, is very demanding. In particular, they justify that in many applications, such as the Netflix problem, it is not possible to arbitrarily choose which matrix entries are determined and which are not, as those may be dictated by outside factors. The low-rank matrix completion problem also has other applications in the area of wireless sensor networks. In one such application, the goal is to reconstruct a low-dimensional geometry describing the locations of the sensors based on local distances sensed by each sensor; this problem is referred to as TRIANGULATION FROM INCOMPLETE DATA (Candès & Recht, 2009). Due to its inherent hardness, the low-rank matrix completion problem has also

*Equal contribution [1]Algorithms and Complexity Group, TU Wien, Vienna, Austria [2]School of Computing, DePaul University, Chicago, USA [3]Algorithms Group, University of Sheffield, Sheffield, UK. Correspondence to: Robert Ganian <rganian@gmail.com>, Iyad Kanj <ikanj@cdm.depaul.edu>, Sebastian Ordyniak <sordyniak@gmail.com>, Stefan Szeider <stefan@szeider.net>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

been studied with respect to various notions of approximation (Candès & Recht, 2009; Candès & Tao, 2010; Frieze et al., 2004; Hardt et al., 2014; Keshavan et al., 2010a;b; Recht, 2011).

The second problem we consider is the matrix completion problem w.r.t. minimizing the number of distinct rows. Although this problem has not received as much attention as low-rank-matrix completion, it certainly warrants studying. In fact, minimizing the number of distinct rows represents a special case of the SPARSE SUBSPACE CLUSTERING problem (Elhamifar & Vidal, 2013), where the goal is to complete a matrix in such a way that its rows can be partitioned into the minimum number of subspaces. The problem we consider corresponds to the special case of SPARSE SUBSPACE CLUSTERING where the matrix is over GF(2) and the desired rank of each subspace is 1. Furthermore, one can see the relevance of this problem to the area of recommender systems; in this context, one seeks to complete the matrix in such a way that the profile of each user is identical to a member of a known (possibly small) group of users.

In this paper, we study the two aforementioned problems through the lens of *parameterized complexity* (Downey & Fellows, 2013). In this paradigm, one measures the complexity of problems not only in terms of their input size n but also by a certain *parameter* $k \in \mathbb{N}$, and seeks—among other things—fixed-parameter algorithms, i.e., algorithms that run in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some function f. Problems admitting such algorithms are said to be *fixed-parameter tractable* (or contained in the parameterized complexity class **FPT**). The motivation is that the parameter of choice—usually describing some structural properties of the instance—can be small in some instances of interest, even when the input size is large. Therefore, by confining the combinatorial explosion to this parameter, one can obtain efficient algorithms for problem instances with a small parameter value for **NP**-hard problems. Problems that are not (or unlikely to be) fixed-parameter tractable can still be solvable in polynomial-time for every fixed parameter value, i.e., they can be solved in time $n^{f(k)}$ for some function f. Problems of this kind are contained in the parameterized complexity class **XP**. We also consider randomized versions of **FPT** and **XP**, denoted by $\mathbf{FPT}_R$ and $\mathbf{XP}_R$, containing all problems that can be solved by a randomized algorithm with a run-time of $f(k)n^{\mathcal{O}(1)}$ and $\mathcal{O}(n^{f(k)})$, respectively, with a constant one-sided error-probability. Finally, problems that remain **NP**-hard for some fixed value of the parameter are hard for the parameterized complexity class **paraNP**. We refer to the respective textbooks for a detailed introduction to parameterized complexity (Downey & Fellows, 2013; Cygan et al., 2015). Parameterized Complexity is a rapidly growing field with various applications in many areas of Computer Science, including Artificial Intelligence (Bäckström et al., 2015; van Bevern et al., 2016; Bessiere et al., 2008; Endriss et al., 2015; Ganian & Ordyniak, 2018; Gaspers & Szeider, 2014; Gottlob & Szeider, 2006).

Parameterizations. The parameters that we consider in this paper are: The number of (matrix) rows that cover all missing entries (row); the number of columns that cover all missing entries (col); and the minimum number of rows and columns which together cover all missing entries (comb). Although we do discuss and provide results for the unbounded domain case, i.e, the case that the domain (field size) is part of the input, we focus on the case when the matrix is over a bounded domain: This case is the most relevant from a practical perspective, and most of the related works focus on this case. It is easy to see that, when stated over any bounded domain, both problems under consideration are in **FPT** when parameterized by the number of missing entries, since an algorithm can brute-force through all possible solutions. On the other hand, parameterizing by row (resp. col) is very interesting from a practical perspective, as rows (resp. columns) with missing entries represent the newly-added elements (*e.g.*, newly-added users/movies/sensors, etc.); here, the above brute-force approach naturally fails, since the number of missing entries is no longer bounded by the parameter alone. Finally, the parameterization by comb is interesting because this parameter subsumes (*i.e.*, is smaller than) the other two parameters (*i.e.*, row and col). In particular, any fixed-parameter algorithm w.r.t. this parameter implies a fixed-parameter algorithm w.r.t. the other two parameters, but can also be efficient in cases where the number of rows or columns with missing entries is large.

Results and Techniques. We start in Section 3 by considering the BOUNDED RANK MATRIX COMPLETION problem over GF(p) (denoted p-RMC), in which the goal is to complete the missing entries in the input matrix so that the rank of the completed matrix is at most t, where $t \in \mathbb{N}$ is given as input. We present a (randomized) fixed-parameter algorithm for this problem parameterized by comb. This result is obtained by applying a branch-and-bound algorithm combined with algebra techniques, allowing us to reduce the problem to a system of quadratic equations in which only few (bounded by some function of the parameter) equations contain non-linear terms. We then use a result by Miura *et al.* (2014) (improving an earlier result by Courtois *et al.* (2002)) in combination with reduction techniques to show that solving such a system of equations is in $\mathbf{FPT}_R$ parameterized by the number of equations containing non-linear terms. In the case where the domain is unbounded, we show that RMC is in **XP** parameterized by either row or col and in $\mathbf{XP}_R$ parameterized by comb.

In Section 4, we turn our attention to the BOUNDED DISTINCT ROW MATRIX COMPLETION problem over both bounded domain (p-DRMC) and unbounded domain (DRMC); here, the goal is to complete the input matrix so

that the number of distinct rows in the completed matrix is at most t. We start by showing that p-DRMC parameterized by comb is fixed-parameter tractable. We obtain this result as a special case of a more general result showing that both DRMC and p-DRMC are fixed-parameter tractable parameterized by the *treewidth* (Robertson & Seymour, 1986; Downey & Fellows, 2013) of the compatibility graph, i.e., the graph having one vertex for every row and an edge between two vertices if the associated rows can be made identical. This result also allows us to show that DRMC is fixed-parameter tractable parameterized by row. Surprisingly, DRMC behaves very differently when parameterized by col, as we show that, for this parameterization, the problem becomes **paraNP**-hard.

	row	col	comb
p-RMC	**FPT**(Th. 3)	**FPT**(Cor. 4)	$\mathbf{FPT}_R$(Th. 7)
p-DRMC	**FPT**(Th. 12)	**FPT**(Th. 12)	**FPT**(Th. 12)
RMC	**XP**(Cor. 5)	**XP**(Cor. 5)	$\mathbf{XP}_R$(Cor. 8)
DRMC	**FPT**(Th. 13)	**paraNP**(Th. 14)	**paraNP**(Th. 14)

Table 1. The parameterized complexity results obtained for the problems p-RMC and p-DRMC and their unbounded domain variants RMC and DRMC w.r.t. the parameters row, col, comb.

We chart our results in Table 1. Interestingly, in the unbounded domain case, both considered problems exhibit wildly different behaviors: While RMC admits **XP** algorithms regardless of whether we parameterize by row or col, using these two parameterizations for DRMC results in the problem being **FPT** and **paraNP**-hard, respectively. On the other hand, in the (more studied) bounded domain case, we show that both problems are in **FPT** (resp. $\mathbf{FPT}_R$) w.r.t. all parameters under consideration. Finally, we prove that 2-DRMC remains **NP**-hard even if every column and row contains (1) a bounded number of missing entries, or (2) a bounded number of determined entries. This effectively rules out **FPT** algorithms w.r.t. the parameters: maximum number of missing/determined entries per row or column.

2. Preliminaries

For a prime number p, let GF(p) be a field of order p; recall that each such field can be equivalently represented as the set of integers modulo p. For positive integers i and $j > i$, we write $[i]$ for the set $\{1, 2, \ldots, i\}$, and $i : j$ for the set $\{i, i+1, \ldots, j\}$.

For an $m \times n$ matrix $\mathbf{M}$ (*i.e.*, a matrix with m rows and n columns), and for $i \in [m]$ and $j \in [n]$, $\mathbf{M}[i, j]$ denotes the element in the i-th row and j-th column of $\mathbf{M}$. Similarly, for a vector d, we write $d[i]$ for the i-th coordinate of d. We write $\mathbf{M}[*, j]$ for the *column-vector* $(\mathbf{M}[1, j], \mathbf{M}[2, j], \ldots, \mathbf{M}[m, j])$, and $\mathbf{M}[i, *]$ for the *row-vector* $(\mathbf{M}[i, 1], \mathbf{M}[i, 2], \ldots, \mathbf{M}[i, n])$. We will also need to refer to submatrices obtained by omitting certain rows or columns from $\mathbf{M}$. We do so by using sets of indices to specify which rows and columns the matrix contains. For instance, the matrix $\mathbf{M}[[i], *]$ is the matrix consisting of the first i rows and all columns of $\mathbf{M}$, and $\mathbf{M}[2 : m, 1 : n-1]$ is the matrix obtained by omitting the first row and the last column from $\mathbf{M}$.

The *row-rank* (resp. *column-rank*) of a matrix $\mathbf{M}$ is the maximum number of linearly-independent rows (resp. *columns*) in $\mathbf{M}$. It is well known that the row-rank of a matrix is equal to its column-rank, and this number is referred to as the *rank* of the matrix. We let $\mathsf{rk}(\mathbf{M})$ and $\mathsf{dr}(\mathbf{M})$ denote the rank and the number of distinct rows of a matrix $\mathbf{M}$, respectively. If $\mathbf{M}$ is a matrix over GF(p), we call GF(p) the *domain* of $\mathbf{M}$.

An *incomplete matrix* over GF(p) is a matrix which may contain not only elements from GF(p) but also the special symbol $\bullet$. An entry is a *missing* entry if it contains $\bullet$, and is a *determined* entry otherwise. A (possibly incomplete) $m \times n$ matrix $\mathbf{M}'$ is *consistent* with an $m \times n$ matrix $\mathbf{M}$ if and only if, for each $i \in [m]$ and $j \in [n]$, either $\mathbf{M}'[i, j] = \mathbf{M}[i, j]$ or $\mathbf{M}'[i, j] = \bullet$.

2.1. Problem Formulation

We formally define the problems under consideration below.

BOUNDED RANK MATRIX COMPLETION (p-RMC)

Input: An incomplete matrix $\mathbf{M}$ over GF(p) for a fixed prime number p, and an integer t.
Task: Find a matrix $\mathbf{M}'$ consistent with $\mathbf{M}$ such that $\mathsf{rk}(\mathbf{M}') \leq t$.

BOUNDED DISTINCT ROW MATRIX COMPLETION (p-DRMC)

Input: An incomplete matrix $\mathbf{M}$ over GF(p) for a fixed prime number p, and an integer t.
Task: Find a matrix $\mathbf{M}'$ consistent with $\mathbf{M}$ such that $\mathsf{dr}(\mathbf{M}') \leq t$.

Aside from the problem variants where p is a fixed prime number, we also study the case where matrix entries range over a domain that is provided as part of the input. In particular, the problems RMC and DRMC are defined analogously to p-RMC and p-DRMC, respectively, with the sole distinction that the prime number p is provided as part of the input. We note that 2-RMC is **NP**-hard even for $t = 3$ (Peeters, 1996), and the same holds for 2-DRMC (see Theorem 15). Without loss of generality, we assume that the rows of the input matrix are pairwise distinct.

2.2. Parameterized Complexity

The class **FPT** consists of all parameterized problems solvable in time $f(k) \cdot n^{\mathcal{O}(1)}$, where k is the parameter and n is the input size; we refer to such running time as **FPT**-*time*. The class **XP** contains all parameterized problems solvable

in time $\mathcal{O}(n^{f(k)})$. The following relations hold among the parameterized complexity classes: **FPT** $\subseteq$ **W**[1] $\subseteq$ **XP**.

The class **paraNP** is defined as the class of parameterized problems that are solvable by a non-deterministic Turing machine in **FPT**-time. In the **paraNP**-hardness proofs, we will make use of the following characterization of **paraNP**-hardness (Flum & Grohe, 2006): Any parameterized problem that remains **NP**-hard when the parameter is a constant is **paraNP**-hard. For problems in **NP**, it holds that **XP** $\subseteq$ **paraNP**; in particular showing the **paraNP**-hardness of a problem rules out the existence of algorithms running in time $\mathcal{O}(n^{f(k)})$ for the problem. We also consider randomized versions of **FPT** and **XP**, denoted by $\mathbf{FPT}_R$ and $\mathbf{XP}_R$, containing all problems that can be solved by a randomized algorithm with a run-time of $f(k)n^{\mathcal{O}(1)}$ and $\mathcal{O}(n^{f(k)})$, respectively, with a constant one-sided error-probability.

2.3. Treewidth

Treewidth (Robertson & Seymour, 1986) is one of the most prominent decompositional parameters for graphs and has found numerous applications in computer science. A *tree-decomposition* $\mathcal{T}$ of a graph $G = (V, E)$ is a pair (T, χ), where T is a tree and χ is a function that assigns each tree node t a set $\chi(t) \subseteq V$ of vertices such that the following conditions hold:

(TD1) For every edge $uv \in E(G)$ there is a tree node t such that $u, v \in \chi(t)$.

(TD2) For every vertex $v \in V(G)$, the set of tree nodes t with $v \in \chi(t)$ forms a non-empty subtree of T.

The sets $\chi(t)$ are called *bags* of the decomposition $\mathcal{T}$ and $\chi(t)$ is the bag associated with the tree node t. The *width* of a tree-decomposition (T, χ) is the size of a largest bag minus 1. A tree-decomposition of minimum width is called *optimal*. The *treewidth* of a graph G, denoted by $\mathsf{tw}(G)$, is the width of an optimal tree decomposition of G. For the presentation of our dynamic programming algorithms, it is convenient to consider tree decompositions in the following normal form (Kloks, 1994): A tuple (T, χ) is a *nice tree decomposition* of a graph G if (T, χ) is a tree decomposition of G, the tree T is rooted at node r, and each node of T is of one of the following four types:

1. a *leaf node*: a node t having no children and $|\chi(t)| = 1$;
2. a *join node*: a node t having exactly two children t_1, t_2, and $\chi(t) = \chi(t_1) = \chi(t_2)$;
3. an *introduce node*: a node t having exactly one child t', and $\chi(t) = \chi(t') \cup \{v\}$ for a node v of G;
4. a *forget node*: a node t having exactly one child t', and $\chi(t) = \chi(t') \setminus \{v\}$ for a node v of G.

For convenience we will also assume that $\chi(r) = \emptyset$ for the root r of T. For $t \in V(T)$ we denote by T_t the subtree of T rooted at t and we write $\chi(T_t)$ for the set $\bigcup_{t' \in V(T_t)} \chi(t')$.

Proposition 1 ((Kloks, 1994; Bodlaender, 1996; Bodlaender et al., 2016)). *It is possible to compute an optimal (nice) tree-decomposition of an n-vertex graph G with treewidth k in time $k^{\mathcal{O}(k^3)}n$, and to compute a 5-approximate one in time $2^{\mathcal{O}(k)}n$. Moreover, the number of nodes in the obtained tree decompositions is $\mathcal{O}(kn)$.*

2.4. Problem Parameterizations

One advantage of the parameterized complexity paradigm is that it allows us to study the complexity of a problem w.r.t. several parameterizations of interest/relevance. To provide a concise description of the parameters under consideration, we introduce the following terminology: We say that a $\bullet$ entry at position $[i, j]$ in an incomplete matrix $\mathbf{M}$ is *covered* by row i and by column j. In this paper, we study RMC and DRMC w.r.t. the following parameterizations (see Figure 1 for illustration):

- $\circ$ col: The minimum number of columns in the matrix $\mathbf{M}$ covering all occurrences of $\bullet$ in $\mathbf{M}$.
- $\circ$ row: The minimum number of rows in the matrix $\mathbf{M}$ covering all occurrences of $\bullet$ in $\mathbf{M}$.
- $\circ$ comb: The minimum value of $r + c$ such that there exist r rows and c columns in $\mathbf{M}$ with the property that each occurrence of $\bullet$ is covered one of these rows or columns.

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

Figure 1. Illustration of the parameters col, row, and comb in an incomplete matrix. Here col $= 4$, row $= 3$, and comb $= 2$.

For instance, the aforementiond problem TRIANGULATION FROM INCOMPLETE DATA, where a small number of distance-sensors are faulty, would result in matrix completion instances where col and row are both small.

We denote the parameter under consideration in brackets after the problem name (e.g., DRMC[comb]). As mentioned in Section 1, both p-RMC and p-DRMC are trivially in **FPT** when parameterized by the number of missing entries, and hence this parameterization is not discussed further.

Given an incomplete matrix $\mathbf{M}$, computing the parameter values for col and row is trivial. Furthermore, the parameter values satisfy comb $\leq$ row and comb $\leq$ col. We show that the parameter value for comb can also be computed in polynomial time.

Proposition 2. *Given an incomplete matrix* $\mathbf{M}$ *over GF*(p)*, we can compute the parameter value for* comb*, along with sets* R *and* C *of total cardinality* comb *containing the indices of covering rows and columns, respectively, in time* $\mathcal{O}((n \cdot m)^{1.5})$.

Proof. We begin by constructing an auxiliary bipartite graph G from $\mathbf{M}$ as follows. For each row i containing a $\bullet$, we create a vertex v_i in G; similarly, for each column j containing a $\bullet$, we create a vertex w_j. For each $\bullet$ that occurs at position $[i, j]$, we add an edge between v_i and w_j.

We observe that if R contains row indices and C contains column indices which together cover all occurrences of $\bullet$, then $X = \{v_i \mid i \in R\} \cup \{w_j \mid j \in C\}$ is a vertex cover of G. Similarly, for each vertex cover X of G, we can obtain a set $R = \{i \mid v_i \in X\}$ and a set $C = \{j \mid w_j \in X\}$ such that R and C cover all occurrences of $\bullet$ in $\mathbf{M}$. Hence there is a one-to-one correspondence between vertex covers in G and sets R and C which cover all $\bullet$ symbols in $\mathbf{M}$, and in particular, the size of a minimum vertex cover in G is equal to comb. The lemma now follows by König's theorem and Hopcroft-Karp's algorithm, which allow us to compute a minimum vertex cover in a bipartite graph G in time $\mathcal{O}(|E(G)| \cdot \sqrt{|V(G)|})$. □

3. Rank Minimization

In this section we present our results for BOUNDED RANK MATRIX COMPLETION under various parameterizations.

3.1. Bounded Domain: Parameterization by row

As our first result, we present an algorithm for solving p-RMC[row]. This will serve as a gentle introduction to the techniques used in the more complex result for p-RMC[comb], and will also be used to give an **XP** algorithm for RMC[row].

Theorem 3. p-RMC[row] *is in* **FPT**.

Proof. Let R be the (minimum) set of rows that cover all occurrences of $\bullet$ in the input matrix $\mathbf{M}$. Since the existence of a solution does not change if we permute the rows of $\mathbf{M}$, we permute the rows of $\mathbf{M}$ so that the rows in R have indices $1, \ldots, k$. We now proceed in three steps.

For the first step, we will define the notion of *signature*: A signature S is a tuple (I, D), where $I \subseteq R$ and D is a mapping from $R \setminus I$ to $(I \to \text{GF}(p))$. Intuitively, a signature S specifies a subset I of R which is expected to be independent in $M[k+1:m,*] \cup I$ (i.e., adding the rows in I to $M[k+1:m,*]$ is expected to increase the rank of $M[k+1:m,*]$ by $|I|$); and for each remaining row of R, S specifies how that row should depend on I. The latter is carried out using D: For each row in $R \setminus I$, D provides a set of coefficients expressing the dependency of that row on the rows in I. Formally, we say that a matrix $\mathbf{M}'$ that is compatible with the incomplete matrix $\mathbf{M}$ *matches* a signature (I, D) if and only if, for each row (*i.e.*, vector) $d \in R \setminus I$, there exist coefficients $a^d_{k+1}, \ldots, a^d_m \in \text{GF}(p)$ such that $d = a^d_{k+1}\mathbf{M}[k+1,*] + \cdots + a^d_m\mathbf{M}[m,*] + \sum_{i \in I} D(d)(i) \cdot i$. The first step of the algorithm branches through all possible signatures S. Clearly, the number of distinct signatures is upper-bounded by $2^k \cdot p^{k^2}$.

For the second step, we fix an enumerated signature S. The algorithm will verify whether S is *valid*, *i.e.*, whether there exists a matrix $\mathbf{M}'$ compatible with $\mathbf{M}$ that matches S. To do so, the algorithm will construct a system of $|R \setminus I|$ equations over vectors of size n, and then transform this into a system Υ_S of $|R \setminus I| \cdot n$ equations over GF(p) (one equation for each vector coordinate). For each $d \in R \setminus I$, Υ_S contains the following variables:

- one variable for each coefficient $a^d_{k+1}, \ldots, a^d_m$, and
- one variable for each occurrence of $\bullet$ in the rows of R.

For instance, the first equation in Υ_S has the following form: $d[1] = a^d_{k+1}\mathbf{M}[k+1,1] + \cdots + a^d_m\mathbf{M}[m,1] + \sum_{i \in I} D(d)(i) \cdot i[1]$, where $a^d_{k+1}, \ldots, a^d_m$ are variables, and $d[1]$ as well as each $i[1]$ in the sum could be a variable or a fixed number. Crucially, Υ_S is a system of at most $(k \cdot n)$ *linear* equations over GF(p) with at most $m + kn$ variables, and can be solved in time $\mathcal{O}((m+kn)^3)$ by Gaussian elimination. Constructing the equations takes time $\mathcal{O}(m \cdot n)$.

During the second step, the algorithm determines whether a signature S is valid or not, and in the end, after going through all signatures, selects an arbitrary valid signature $S = (I, D)$ with minimum $|I|$. For the final third step, the algorithm checks whether $|I| + \mathsf{rk}(\mathbf{M}[k+1:m,*]) \leq t$. We note that computing $\mathsf{rk}(\mathbf{M}[k+1:m,*])$ can be carried out in time $\mathcal{O}(nm^{1.4})$ (Ibarra et al., 1982). If the above inequality does not hold, the algorithm rejects; otherwise it recomputes a solution to Υ_S and outputs the matrix $\mathbf{M}'$ obtained from $\mathbf{M}$ by replacing each occurrence of $\bullet$ at position $[i, j]$ by the value of the variable $i[j]$ in the solution to Υ_S. The total running time is $\mathcal{O}((2^k \cdot p^{k^2}) \cdot ((m+kn)^3 + nm^{1.4})) = \mathcal{O}(2^k p^{k^2} \cdot (m+kn)^3)$.

To argue the correctness of the algorithm, consider a matrix $\mathbf{M}'$ that the algorithm outputs. Obviously, $\mathbf{M}'$ is consistent with $\mathbf{M}$. Furthermore, $\mathbf{M}'$ has rank at most t; indeed, the rank of $\mathbf{M}[k+1:m,*]$ is at most $t - |I|$, and every row $d \in R \setminus I$ can be obtained as a linear combination of rows in $\mathbf{M}[k+1:m,*]$ (using coefficients $a^d_{k+1}, \ldots, a^d_m$) and I (using coefficients $D(d)$).

Conversely, assume that there exists a matrix $\mathbf{M}'$ that is consistent with $\mathbf{M}$ and that has rank at most t. Choose $\mathbf{M}'$

to be of minimum rank over all matrices consistent with $\mathbf{M}$. Consider the signature S obtained ("reverse-engineered") from $\mathbf{M}'$ as follows. First, we choose a row-basis B of $\mathbf{M}'$ such that $|B \cap R|$ is minimized, and we set $I = R \cap B$. Now, each row in $R \setminus I$ can be obtained as a linear combination of rows in B and, in particular, as a linear combination of the rows in $\mathbf{M}'[k+1:m,*]$ and I. This can be expressed as a system of equations Υ', where, for each row $d \in R \setminus I$, we write $d = a^d_{k+1}\mathbf{M}[k+1,*] + \cdots + a^d_m\mathbf{M}'[m,*] + \sum_{i \in I} D(d)(i) \cdot i$ and our variables are: $a^d_{k+1}, \ldots, a^d_m$ and, $\forall i \in I, D(d)(i)$. Let us fix an arbitrary solution to Υ' and use the values assigned to variables $D(d)(i), \forall d \in R \setminus I, i \in I$, to define D. Observe that $S = (I, D)$ was chosen so that Υ_S is guaranteed to have a solution.

Next, we argue that $|I| + \mathsf{rk}(\mathbf{M}[k+1:m,*]) = \mathsf{rk}(\mathbf{M}')$. Indeed, assume for a contradiction that there exists a row $r \in R \cap I$ which can be obtained as a linear combination of the rows in I and in $\mathbf{M}[k+1:m,*]$; then, we could replace r in B by a row r' from $\mathbf{M}[k+1:m,*]$ which would violate the minimality of $|B \cap R|$. So, $|I| + \mathsf{rk}(\mathbf{M}[k+1:m,*]) = \mathsf{rk}(\mathbf{M}')$ which means that our algorithm is guaranteed to set S as a valid branch, and hence, will either output a matrix compatible with $\mathbf{M}$ which matches S, or a matrix compatible with $\mathbf{M}$ which matches a different signature but has the same rank as $\mathbf{M}'$. □

Since the row-rank of a matrix $\mathbf{M}$ is equal to its column-rank, the transpose of $\mathbf{M}$ has the same rank as $\mathbf{M}$. Hence:

Corollary 4. p-RMC[col] *is in* **FPT**.

As a consequence of the running time of the algorithm given in the proof of Theorem 3, we obtain:

Corollary 5. RMC[row] *and* RMC[col] *are in* **XP**.

3.2. Bounded Domain: Parameterization by comb

In this subsection, we present a randomized fixed-parameter algorithm for p-RMC[comb] with constant one-sided error probability. Before we proceed to the algorithm, we need to introduce some basic terminology related to systems of equations. Let Υ be a system of ℓ equations $\mathrm{EQ}_1, \mathrm{EQ}_2, \ldots, \mathrm{EQ}_\ell$ over GF(p); we assume that the equations are simplified as much as possible. In particular, we assume that no equation contains two terms over the same set of variables such that the degree/exponent of each variable in both terms is the same. Let EQ_i be a linear equation in Υ, and let x be a variable which occurs in EQ_i (with a non-zero coefficient). Naturally, EQ_i can be transformed into an equivalent equation $\mathrm{EQ}_{i,x}$, where x is isolated, and we use $\Gamma_{i,x}$ to denote the side of $\mathrm{EQ}_{i,x}$ not containing x, i.e., $\mathrm{EQ}_{i,x}$ is of the form $x = \Gamma_{i,x}$. We say that Υ' is obtained from Υ by *substitution of x in EQ_i* if Υ' is the system of equations obtained by:

1. computing $\mathrm{EQ}_{i,x}$ and in particular $\Gamma_{i,x}$ from EQ_i;
2. setting $\Upsilon' := \Upsilon \setminus \{\mathrm{EQ}_i\}$; and
3. replacing x with $\Gamma_{i,x}$ in every equation in Υ'.

Observe that Υ' has size $\mathcal{O}(n \cdot \ell)$, and can also be computed in time $\mathcal{O}(n \cdot \ell)$, where n is the number of variables occurring in Υ. Furthermore, any solution to Υ' can be transformed into a solution to Υ in linear time, and similarly any solution to Υ can be transformed into a solution to Υ' in linear time (*i.e.*, Υ' and Υ are equivalent). Moreover, Υ' contains at least one fewer variable and one fewer equation than Υ.

The following proposition is crucial for our proof, and is of independent interest.

Proposition 6. *Let Υ be a system of ℓ quadratic equations over GF(p). Then computing a solution for Υ is in* $\mathbf{FPT}_R$ *parameterized by ℓ and p, and in* $\mathbf{XP}_R$ *parameterized only by ℓ.*

Proof. Let n be the number of variables in Υ. We distinguish two cases. If $n \geq \ell(\ell+3)/2$, then Υ can be solved in randomized time $\mathcal{O}(2^\ell n^3 \ell (\log p)^2)$ (Miura et al., 2014). Otherwise, $n < \ell(\ell+3)/2$, and we can solve Υ by a brute-force algorithm which enumerates (all of the) at most $p^n < p^{\ell(\ell+3)/2}$ assignments of values to the variables in Υ. The proposition now follows by observing that the given algorithm runs in time $\mathcal{O}(2^\ell n^3 \ell (\log p)^2 + p^{\ell(\ell+3)/2}\ell^2)$. □

Theorem 7. p-RMC[comb] *is in* $\mathbf{FPT}_R$.

Proof. We begin by using Proposition 2 to compute the sets R and C containing the indices of the covering rows and columns, respectively; let $|R| = r$ and $|C| = c$, and recall that the parameter value is $k = r + c$. Since the existence of a solution for p-RMC does not change if we permute rows and columns of $\mathbf{M}$, we permute the rows of $\mathbf{M}$ so that the rows in R have indices $1, \ldots, r$, and subsequently, we permute the columns of $\mathbf{M}$ so that the columns in C have indices $1, \ldots, c$.

Before we proceed, let us give a high-level overview of our strategy. The core idea is to branch over signatures, which will be defined in a similar way to those in Theorem 3. These signatures will capture information about the dependencies among the rows in R and columns in C; one crucial difference is that for columns, we will focus only on dependencies in the submatrix $\mathbf{M}[r+1:m,*]$ (the reason will become clear later, when we argue correctness). In each branch, we arrive at a system of equations that needs to be solved in order to determine whether the signatures are valid. Unlike Theorem 3, here the obtained system of equations will contain non-linear (but quadratic) terms, and hence solving the system is far from being trivial. Once we determine which signatures are valid, we choose one that minimizes the total rank.

For the first step, let us define the notion of signature that will be used in this proof. A *signature* S is a tuple (I_R, D_R, I_C, D_C) where:

1. $I_R \subseteq R$;
2. D_R is a mapping from $R \setminus I_R$ to $(I_R \to \text{GF}(p))$;
3. $I_C \subseteq C$; and
4. D_C is a mapping from $C \setminus I_C$ to $(I_C \to \text{GF}(p))$.

We say that a matrix $\mathbf{M}'$ compatible with the incomplete matrix $\mathbf{M}$ *matches* a signature (I_R, D_R, I_C, D_C) if:

- for each row $d \in R \setminus I_R$, there exist coefficients $a^d_{r+1}, \ldots, a^d_m \in \text{GF}(p)$ such that $d = a^d_{r+1}\mathbf{M}'[r+1, *] + \cdots + a^d_m \mathbf{M}'[m, *] + \sum_{i \in I_R} D_R(d)(i) \cdot i$; and
- for each column $h \in C \setminus I_c$, there exist coefficients $b^h_{c+1}, \ldots, b^h_n \in \text{GF}(p)$ such that $h[r+1 : m] = b^h_{c+1}\mathbf{M}'[r+1 : m, c] + \cdots + b^h_n \mathbf{M}'[r+1 : m, n] + \sum_{i \in I_C} D_C(h)(i) \cdot i[r+1 : m]$.

The number of distinct signatures is upper-bounded by $2^r \cdot p^{r^2} \cdot 2^c \cdot p^{c^2} \leq 2^k \cdot p^{k^2}$, and the first step of the algorithm branches over all possible signatures S. For the second step, fix an enumerated signature S. The algorithm will verify whether S is *valid*, *i.e.*, whether there exists a matrix $\mathbf{M}'$, compatible with the incomplete $\mathbf{M}$, that matches S. To do so, the algorithm will construct a system of $|R \setminus I_R|$ equations over vectors of size n and of $|C \setminus I_C|$ equations over vectors of size $m - r$, and then transform this into a system Υ_S of $|R \setminus I_R| \cdot n + |C \setminus I_C| \cdot (m - r)$ equations over GF(p) (one equation for each vector coordinate). For each $d \in R \setminus I_R$, Υ_S contains the following variables:

- one variable for each $a^d_{r+1}, \ldots, a^d_m$, and
- one variable for each occurrence of $\bullet$.

For instance, the first equation in Υ_S for some $d \in R \setminus I_R$ has the following form: $d[1] = a^d_{r+1}\mathbf{M}[r+1, 1] + \cdots + a^d_m \mathbf{M}[m, 1] + \sum_{i \in I_R} D_R(d)(i) \cdot i[1]$, where $a^d_{r+1}, \ldots, a^d_m$ are variables, $D_R(d)(i)$ is a number, and all other occurrences are either variables or numbers. Crucially, for all $j > c$, the equations for $d[j]$ defined above contain only linear terms; however, for $j \in [c]$ these equations may also contain non-linear terms (in particular, $a^d_{r+1}, \ldots, a^d_m$ are variables and $\mathbf{M}[r+1, j], \ldots, \mathbf{M}[m, j]$ can contain $\bullet$ symbols, which correspond to variables in the equations). For $z \in [m]$ and $y \in [n]$, if an element $\mathbf{M}[z, y]$ contains $\bullet$, then we will denote the corresponding variable used in the equations as $x_{z,y}$.

Next, for each $h \in C \setminus I_C$, Υ_S contains the following variables:

- one variable for each $b^h_{c+1}, \ldots, b^h_n$, and
- one variable for each occurrence of $\bullet$.

For instance, the second equation in Υ_S for some $h \in C \setminus I_C$ has the following form: $h[r+2] = b^h_{c+1}\mathbf{M}[r+2, c+1] + \cdots + b^h_n \mathbf{M}[r+2, n] + \sum_{i \in I_C} D_C(d)(i) \cdot i[r+2]$, where $b^h_{c+1}, \ldots, b^h_n$ are variables, $D_C(d)(i)$ is a number, and all other occurrences are either variables or numbers. Observe that all of these equations for h are linear, since the submatrix $\mathbf{M}[r+1 : m, c+1 : n]$ contains no $\bullet$ symbols.

This completes the definition of our system of equations Υ_S. Recall that the only equations in Υ_S that may contain non-linear terms are those for $d[j]$ when $j \leq c$, and in particular Υ_S contains at most k^2 equations with non-linear terms (k equations for at most k vectors d in $R \setminus I_R$). We will now use substitutions to simplify Υ_S by removing all linear equations; specifically, at each step we select an arbitrary linear equation EQ_i containing a variable x, apply substitution of x in EQ_i to construct a new system of equations with one fewer equation, and simplify all equations in the new system. If at any point we reach a system of equations which contains an invalid equation (*e.g.*, 2=5), then Υ_S does not have a solution, and we discard the corresponding branch. Otherwise, after at most $|R \setminus I_R| \cdot n + |C \setminus I_C| \cdot (m - r) \in \mathcal{O}(kn + km)$ substitutions we obtain a system of at most k^2 quadratic equations Ψ_S such that any solution to Ψ_S can be transformed into a solution to Υ_S in time at most $\mathcal{O}(kn + km)$. We can now apply Proposition 6 to solve Ψ_S and mark S as a valid signature if Ψ_S has a solution.

After all signatures have been processed, in the third—and final—step we select a valid signature $S = (I, D)$ that has the minimum value of $|I_R| + |I_C|$. The algorithm will then check whether $|I_R| + |I_C| + \mathsf{rk}(\mathbf{M}[r+1 : m, c : 1+n]) \leq t$. If this not the case, the algorithm rejects the instance. Otherwise, the algorithm recomputes a solution to Υ_S, and outputs the matrix $\mathbf{M}'$ obtained from $\mathbf{M}$ by replacing each occurrence of $\bullet$ at position $[i, j]$ in $\mathbf{M}'$ by $x_{i,j}$.

We now proceed to proving the correctness of the algorithm. We do so by proving the following two claims:

Claim 1. *If there exists a signature $S = (I_R, D_R, I_C, D_C)$ for $\mathbf{M}$ such that $|I_R| + |I_C| + \mathsf{rk}(\mathbf{M}[r+1 : m, c : 1+n]) \leq t$, then there exists a matrix $\mathbf{M}'$ compatible with $\mathbf{M}$ such that $\mathsf{rk}(\mathbf{M}') \leq |I_R| + |I_C| + \mathsf{rk}(\mathbf{M}[r+1 : m, c : 1+n])$. In particular, if S is marked as valid by the algorithm, then the algorithm outputs a matrix $\mathbf{M}'$ satisfying the above.*

Proof of Claim. Since S is valid, the system of equations Υ_S has a solution; fix one such solution. Consider the matrix $\mathbf{M}'$ obtained from $\mathbf{M}$ by replacing each occurrence of $\bullet$ at position $[i, j]$ by the value of $x_{i,j}$ from the selected solution

to Υ_S. Then the solution to Υ_S guarantees that each row in $R \setminus I_R$ can be obtained as a linear combination of rows in $\mathbf{M}' \setminus (R \setminus I_R)$, and hence deleting all rows in $R \setminus I_R$ will result in a matrix $\mathbf{M}'_1$ such that $\mathsf{rk}(\mathbf{M}') = \mathsf{rk}(\mathbf{M}'_1)$.

Next, consider the matrix $\mathbf{M}'[r+1 : m, *]$ which is obtained by removing all rows in I_R from $\mathbf{M}'_1$; clearly, this operation decreases the rank by at most $|I_R|$, and hence $\mathsf{rk}(\mathbf{M}'[r+1 : m, *]) \leq \mathsf{rk}(\mathbf{M}'_1) \leq \mathsf{rk}(\mathbf{M}'[r+1 : m, *]) + |I_R|$.

Third, consider the matrix $\mathbf{M}'_2$ obtained from $\mathbf{M}'[r+1 : m, *]$ by removing all columns in $C \setminus I_C$. The solution to Υ_S guarantees that each removed column can be obtained as a linear combination of columns in $\mathbf{M}'[r+1 : m, *] \setminus (C \setminus I_C)$, and hence, $\mathsf{rk}(\mathbf{M}'[r+1 : m, *) = \mathsf{rk}(\mathbf{M}'_2)$. Finally, we consider the matrix $\mathbf{M}'[r+1 : m, c+1 : n] = \mathbf{M}[r+1 : m, c+1 : n]$ which is obtained by removing all columns in I_C from $\mathbf{M}'_2$. Clearly, removing $|I_C|$ columns decreases the rank by at most $|I_C|$, and hence $\mathsf{rk}(\mathbf{M}[r+1 : m, c+1 : n]) \leq \mathsf{rk}(\mathbf{M}'_2) \leq \mathsf{rk}(\mathbf{M}[r+1 : m, c+1 : n]) + |I_C|$. Putting the above inequalities together, we get $\mathsf{rk}(\mathbf{M}') \leq \mathsf{rk}(\mathbf{M}'[r+1 : m, *]) + |I_R| \leq \mathsf{rk}(\mathbf{M}[r+1 : m, c+1 : n]) + |I_C| + |I_R|$. ■

Claim 2. *If there exists a matrix $\mathbf{M}'$ compatible with $\mathbf{M}$ such that $\mathsf{rk}(\mathbf{M}') \leq t$, then there exists a valid signature $S = (I_R, D_R, I_C, D_C)$ such that $|I_R| + |I_C| + \mathsf{rk}(\mathbf{M}[r+1 : m, c : 1+n]) \leq t$.*

Proof of Claim. Consider the following iterative procedure that creates a set I_R from the hypothetical matrix $\mathbf{M}'$. Check, for each row $r \in R$, whether R can be obtained as a linear combination of all other rows in $\mathbf{M}'$, which can be done by solving a system of linear equations; if this is the case, remove r from $\mathbf{M}'$ and restart from any row in R that remains in $\mathbf{M}'$. In the end, we obtain a submatrix $\mathbf{M}'_R$ of $\mathbf{M}'$ which only contains those rows in R that cannot be obtained as a linear combination of all other rows in $\mathbf{M}'_R$; let I_R be the set of rows in R that remain in $\mathbf{M}'_R$. Furthermore, since each row $r' \in R \setminus I_R$ can be obtained as a linear combination of rows in $\mathbf{M}'_R$, for each such r' we compute a set of coefficients $\tau_{r'}$ that can be used to obtain r' and store those coefficients corresponding to I_R in D_R. For instance, if row $r' \in R \setminus I_R$ can be obtained by an additive term containing 1 times row $u \in I_R$, then we set $D_R(r') = (u \mapsto 1)$.

At this point, we have identified I_R and D_R. Next, we turn our attention to the submatrix $\mathbf{M}'[r+1 : m, *]$, where we proceed similarly but for columns. In particular, for each column $c \in C$ restricted to $\mathbf{M}'[r+1 : m, *]$, we check whether c can be obtained as a linear combination of all other columns in $\mathbf{M}'[r+1 : m, *]$, and if the answer is positive then we remove c from $\mathbf{M}'[r+1 : m, *]$ and restart from any column in C that remains in $\mathbf{M}'[r+1 : m, *]$. This results in a new submatrix $\mathbf{M}'_C$ of $\mathbf{M}'[r+1 : m, *]$, and those columns of C that remain in $\mathbf{M}'_C$ are stored in I_C. Then, for each column in $c' \in C \setminus I_C$, we compute a set of coefficients $\tau_{c'}$ that can be used to obtain that column and store the values of the coefficients that correspond to I_C in D_C, analogously as we did for the rows.

At this point, we have obtained a signature S. The validity of S follows from its construction. Indeed, to solve Υ_S, we can set each variable $x_{i,j}$ representing the value of a $\bullet$ symbol at $\mathbf{M}[i, j]$ to $\mathbf{M}'[i, j]$, and all other variables will capture the coefficients that were stored in $\tau_{r'}$ and $\tau_{c'}$ for a row r' or a column c', respectively.

Finally, we argue that $|I_R| + |I_C| + \mathsf{rk}(\mathbf{M}[r+1 : m, c : 1+n]) \leq t$. Since $\mathbf{M}'_R$ was obtained from $\mathbf{M}'$ only by deleting linearly dependent rows, $\mathsf{rk}(\mathbf{M}') = \mathsf{rk}(\mathbf{M}'_R)$. Furthermore, since $\mathbf{M}'[r+1 : m, *]$ can be obtained by deleting $|I_R|$ rows from $\mathbf{M}'_R$, and all of these deleted rows are linearly independent of all other rows in $\mathbf{M}'_R$, we obtain $\mathsf{rk}(\mathbf{M}'[r+1 : m, *]) = \mathsf{rk}(\mathbf{M}'_R) - |I_R|$. By repeating the above arguments, we see that $\mathsf{rk}(\mathbf{M}'[r+1 : m, *]) = \mathsf{rk}(\mathbf{M}'_C)$ and $\mathsf{rk}(\mathbf{M}'[r+1 : m, c+1 : n]) = \mathsf{rk}(\mathbf{M}'_C) - |I_C|$. Recall that $\mathbf{M}'[r+1 : m, c+1 : n] = \mathbf{M}[r+1 : m, c+1 : n]$. Putting the above together, we obtain $\mathsf{rk}(\mathbf{M}'[r+1 : m, c+1 : n]) + |I_C| = \mathsf{rk}(\mathbf{M}'[r+1 : m, *])$, and $\mathsf{rk}(\mathbf{M}'[r+1 : m, c+1 : n]) + |I_C| + |I_R| = \mathsf{rk}(\mathbf{M}') \leq t$. ■

Finally, the total running time of the algorithm is obtained by combining the branching factor of branching over all signatures ($\mathcal{O}(2^k \cdot p^{k^2})$) with the run-time of Proposition 6 for k^2 many quadratic equations ($\mathcal{O}(3^{k^2} n^3 (\log p)^2 + p^{k^4})$). We obtain a running time of $\mathcal{O}(3^{k^2} \cdot p^{k^4} \cdot n^3)$. □

As a consequence of the running time of the algorithm given in the proof of Theorem 7, we obtain:

Corollary 8. RMC[comb] *is in* $\mathbf{XP}_R$.

4. Bounded Distinct Row Matrix Completion

Let $(p, \mathbf{M}, t)$ be an instance of DRMC. We say that two rows of $\mathbf{M}$ are *compatible* if whenever the two rows differ at some entry then one of the rows has a $\bullet$ at that entry. The *compatibility graph* of $\mathbf{M}$, denoted by $G(\mathbf{M})$, is the undirected graph whose vertices correspond to the row indices of $\mathbf{M}$ and in which there is an edge between two vertices if and only if their two corresponding rows are compatible. See Figure 2 for an illustration.

We start by showing that DRMC (and therefore p-DRMC) can be reduced to the Clique Cover problem, which is defined as follows.

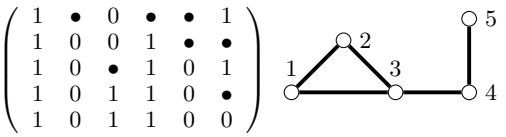


Figure 2. Illustration of a matrix and its compatibility graph. The vertex label indicates the corresponding row number.

CLIQUE COVER (CC)

Input: An undirected graph G and an integer k.
Task: Find a partition of $V(G)$ into at most k cliques, or output that no such partition exists.

Lemma 9. *An instance $\mathcal{I} = (p, \mathbf{M}, t)$ of* DRMC *has a solution if and only if the instance $\mathcal{I}' = (G(\mathbf{M}), t)$ of* CC *does. Moreover, a solution for $\mathcal{I}'$ can obtained in polynomial-time from a solution for $\mathcal{I}$ and vice versa.*

Proof. Let $\mathbf{M}'$ be a solution for $\mathcal{I}$ and let $\mathcal{P}$ be the partition of the indices of the rows of $\mathbf{M}'$ such that two row-indices r and r' belong to the same set in $\mathcal{P}$ if and only if $\mathbf{M}'[r, *] = \mathbf{M}'[r', *]$. Then $\mathcal{P}$ is also a solution for $\mathcal{I}'$, since $G[P]$ is a clique for every $P \in \mathcal{P}$.

Conversely, let $\mathcal{P}$ be a solution for $\mathcal{I}'$. We claim that there is a solution $\mathbf{M}'$ for $\mathcal{I}$ such that $\mathbf{M}'[r, *] = \mathbf{M}'[r', *]$ if and only if r and r' are contained in the same set of $\mathcal{P}$. Towards showing this, consider a set $P \in \mathcal{P}$ and a column index c of $\mathbf{M}$, and let $\mathbf{E}(\mathbf{M}[P, c])$ be the set of all values occurring in $\mathbf{M}[P, c]$. Then $|\mathbf{E}(\mathbf{M}[P, c]) \setminus \{\bullet\}| \leq 1$, that is, all entries of $\mathbf{M}[P, c]$ that are not $\bullet$ are equal; otherwise, $G[P]$ would not be a clique. Consequently, by replacing every $\bullet$ occurring in $\mathbf{M}[P, c]$ with the unique value in $\mathbf{E}(\mathbf{M}[P, c]) \setminus \{\bullet\}$ if $\mathbf{E}(\mathbf{M}[P, c]) \setminus \{\bullet\} \neq \emptyset$, and with an arbitrary value otherwise, and by doing so for every column index c and every $P \in \mathcal{P}$, we obtain the desired solution $\mathbf{M}'$ for $\mathcal{I}$. □

Theorem 10. CC *is in* **FPT** *when parameterized by the treewidth of the input graph.*

Proof. Let $\mathcal{I} = (G, k)$ be an instance of CC. We will show the Theorem using a standard dynamic programming algorithm on a tree-decomposition of G. Because of Proposition 1 we can assume that we are given a nice tree-decomposition (T, χ) of G of width ω. For every node $t \in V(T)$ we will compute the set $\mathcal{R}(t)$ of records containing all pairs $(\mathcal{P}, c)$, where $\mathcal{P}$ is a partition of $\chi(t)$ into cliques, i.e., for every $P \in \mathcal{P}$ the graph $G[P]$ is a clique, and c is the minimum integer such that $G[\chi(T_t)]$ has a partition $\mathcal{P}'$ into c cliques with $\mathcal{P} = \{ P' \cap \chi(t) \mid P' \in \mathcal{P}' \} \setminus \{\emptyset\}$. Note that $\mathcal{I}$ has a solution if and only if $\mathcal{R}(r)$ contains a record $(\emptyset, c)$ with $c \leq k$, where r is the root of (T, χ). It hence suffices to show how to compute the set of records for the four different types of nodes of a nice tree-decomposition.

Let l be a leaf node of T with $\chi(l) = \{v\}$. Then $\mathcal{R}(l) := \{(\{\{v\}\}, 1)\}$. Note the $\mathcal{R}(l)$ can be computed in constant time.

Let t be an introduce node of T with child t' and $\chi(t) = \chi(t') \cup \{v\}$. Then $\mathcal{R}(t)$ can be obtained from $\mathcal{R}(t')$ as follows. For every $(\mathcal{P}', c') \in \mathcal{R}(t')$ and every $P' \in \mathcal{P}'$ such that $G[P' \cup \{v\}]$ is a clique, we add the record $((\mathcal{P}' \setminus \{P'\}) \cup \{P' \cup \{v\}\}, c')$ to $\mathcal{R}(t)$. Moreover, for every $(\mathcal{P}', c') \in \mathcal{R}(t')$, we add the record $(\mathcal{P}' \cup \{\{v\}\}, c' + 1)$ to $\mathcal{R}(t)$. Note that $\mathcal{R}(t)$ can be computed in time $\mathcal{O}(|\mathcal{R}(t')|\omega^2)$.

Let t be a forget node of T with child t' and $\chi(t) \cup \{v\} = \chi(t')$. Then $\mathcal{R}(t)$ consists of all records $(\mathcal{P}, c)$ such that c is the minimum integer such that there is a record $(\mathcal{P}', c) \in \mathcal{R}(t')$ and a set $P' \in \mathcal{P}'$ with $v \in P'$ and $(\mathcal{P}' \setminus P') \cup (P' \setminus \{v\}) = \mathcal{P}$; if no such record exists $(\mathcal{P}, c)$ is not in $\mathcal{R}(t)$. Note that $\mathcal{R}(t)$ can be computed in time $\mathcal{O}(|\mathcal{R}(t')|\omega^2)$.

Let t be a join node with children t_1 and t_2. Then $\mathcal{R}(t)$ contains all records $(\mathcal{P}, c)$ such that there are integers c_1 and c_2 with $c_1 + c_2 - |\mathcal{P}| = c$ and $(\mathcal{P}, c_1) \in \mathcal{R}(t_1)$ and $(\mathcal{P}, c_2) \in \mathcal{R}(t_2)$. Note that $\mathcal{R}(t)$ can be computed in time $\mathcal{O}((|\mathcal{R}(t_1)| + |\mathcal{R}(t_2)|)\omega)$ (assuming that the records are kept in an ordered manner).

The total run-time of the algorithm is then the number of nodes of T, i.e., $\mathcal{O}(\omega|V(G)|)$, times the maximum time required at any of the four types of nodes, i.e., $\mathcal{O}(|\mathcal{R}(t)|\omega^2)$, which because $|\mathcal{R}(t)| \leq \omega!$ is at most $\mathcal{O}(\omega!\omega^3|V(G)|)$. □

Note that the above theorem also implies that the well-known COLORING problem is **FPT** parameterized by the treewidth of the complement of the input graph. The theorem below follows immediately from Lemmas 9 and 10.

Theorem 11. DRMC *and* p-DRMC *are in* **FPT** *when parameterized by the treewidth of the compatibility graph.*

4.1. p-DRMC

Theorem 12. p-DRMC[comb] *is in* **FPT**.

Proof. Let $(\mathbf{M}, t)$ be an instance of p-DRMC, and let k be the parameter comb. By Proposition 2, we can compute a set $R_\bullet$ of rows and a set $C_\bullet$ of columns, where $|R_\bullet \cup C_\bullet| \leq k$, and such that every occurrence of $\bullet$ in $\mathbf{M}$ is either contained in a row or column in $R_\bullet \cup C_\bullet$. Let R and C be the set of rows and columns of $\mathbf{M}$, respectively. Let $\mathcal{P}$ be the unique partition of $R \setminus R_\bullet$ such that two rows r and r' belong to the same set in $\mathcal{P}$ if and only if they are identical on all columns in $C \setminus C_\bullet$. Then $|P| \leq (p+1)^k$, for every $P \in \mathcal{P}$, since two rows in P can differ

on at most $|C_\bullet| \leq k$ entries, each having $(p+1)$ values to be chosen from. Moreover, any two rows in $R \setminus R_\bullet$ that are not contained in the same set in $\mathcal{P}$ are not compatible, which implies that they appear in different components of $G(\mathbf{M}) \setminus R_\bullet$ and hence the set of vertices in every component of $G(\mathbf{M}) \setminus R_\bullet$ is a subset of P, for some $P \in \mathcal{P}$. It is now straightforward to show that $\mathsf{tw}(G(\mathbf{M})) \leq k+(p+1)^k$, and hence, $\mathsf{tw}(G(\mathbf{M}))$ is bounded by a function of the parameter k. Towards showing this consider the tree-decomposition (T, χ) for $G(\mathbf{M})$, where T is a path containing one node t_P with $\chi(t_P) = R_\bullet \cup P$ for every $P \in \mathcal{P}$. Then (T, χ) is tree-decomposition of width $k + (p+1)^k - 1$ for $G(\mathbf{M})$. The theorem now follows from Theorem 11. □

4.2. DRMC

The proof of the following theorem is very similar to the proof of Theorem 12, i.e., we mainly use the observation that the parameter row is also a bound on the treewidth of the compatibility graph and then apply Theorem 11.

Theorem 13. DRMC[row] *is in* **FPT**.

Proof. Let $(p, \mathbf{M}, t)$ be an instance of DRMC, let k be the parameter row and let $R_?$ be a set of rows with $|R_?| \leq k$ covering all occurrences of $\bullet$ in $\mathbf{M}$. Then $G(\mathbf{M}) \setminus R_?$ is an independent set since any two distinct rows without $\bullet$ are not compatible. It is now straightforward to show that $\mathsf{tw}(G(\mathbf{M})) \leq k$ and hence bounded by a function of our parameter k. Towards showing this the following tree-decomposition (T, χ) for $G(\mathbf{M})$, where T is a path containing one node t_r with $\chi(t_r) = R_? \cup \{r\}$ for every $r \in R \setminus R_?$. Then (T, χ) is tree-decomposition of width k for $G(\mathbf{M})$. The theorem now follows from Theorem 11. □

For our remaining hardness proofs we will make use of the following problem.

PARTITIONING INTO TRIANGLES (PIT)	
Input:	A graph G.
Task:	Is there a partition $\mathcal{P}$ of $V(G)$ into triangles, i.e., $G[P]$ is a triangle for every $P \in \mathcal{P}$?

We will often use the following easy observation.

Observation 1. *A graph G that does not contain a clique with four vertices has a partition into triangles if and only if it has a partition into at most $|V(G)|/3$ cliques.*

Theorem 14. DRMC[col] *is* **paraNP**-*hard.*

Proof. We will reduce from the following variant of 3-SAT, which is **NP**-complete (Berman et al., 2003).

3-SATISFIABILITY-2 (3-SAT-2)	
Input:	A propositional formula ϕ in conjunctive normal form such that (1) every clause of ϕ has exactly three distinct literals and (2) every literal occurs in exactly two clauses.
Task:	Is ϕ satisfiable?

To make our reduction easier to follow, we will divide the reduction into two steps. Given an instance (formula) ϕ of 3-SAT-2, we will first construct an equivalent instance G of PIT with the additional property that G does not contain a clique on four vertices. We note that similar reductions from variants of the satisfiability problem to PIT are known (and hence our first step does not show anything new for PIT); however, our reduction is specifically designed to simplify the second step, in which we will construct an instance $(\mathbf{M}, |V(G)|/3)$ of DRMC such that $G(\mathbf{M})$ is isomorphic to G and $\mathbf{M}$ has only seven columns. By Observation 1 and Lemma 9, this proves the theorem since $(\mathbf{M}, |V(G)|/3)$ has a solution if and only if ϕ does.

Let ϕ be an instance of 3-SAT-2 with variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$. We first construct the instance G of PIT such that G does not contain a clique of size four. For every variable x_i of ϕ, let $G(x_i)$ be the graph with vertices x_i^1, x_i^2, $\bar{x}_i^1$, $\bar{x}_i^2$, x_i and edges forming a triangle on the vertices x_i^1, x_i^2, and x_i as well as a triangle on the vertices $\bar{x}_i^1$, $\bar{x}_i^2$, and x_i. Moreover, for every clause C_j with literals $l_{j,1}$, $l_{j,2}$, and $l_{j,3}$, let $G(C_j)$ be the graph with vertices $l_{j,1}^1$, $l_{j,1}^2$, $l_{j,2}^1$, $l_{j,2}^3$, $l_{j,3}^1$, $l_{j,3}^2$, h_j^1, and h_j^2 and edges between $l_{j,r}^1$ and $l_{j,r}^2$ for every $r \in \{1,2,3\}$ as well as edges forming a complete bipartite graph between $\{h_j^1, h_j^2\}$ and all other vertices of $G(C_j)$. Let $f : [m] \times [3] \to \{ x_i^o, \bar{x}_i^o \mid 1 \leq i \leq n \wedge 1 \leq o \leq 2 \}$ be any bijective function such that for every j and r with $1 \leq j \leq m$ and $1 \leq r \leq 3$, it holds that: If $f(j,r) = x_i^o$ (for some i and o), then x_i is the r-th literal of C_j; and if $f(j,r) = \bar{x}_i^o$, then $\bar{x}_i$ is the r-th literal of C_j. Figures 3 and 4 illustrate the gadgets $G(x_i)$ and $G(C_j)$.

The graph G is obtained from the disjoint union of the graphs $G(x_1), \ldots, G(x_n), G(C_1), \ldots, G(C_m)$ after applying the following modifications:

- For every j and r with $1 \leq j \leq m$ and $1 \leq r \leq 3$ add edges forming a triangle on the vertices $l_{j,r}^1$, $l_{j,r}^2$, $f(j,r)$.
- for every i with $1 \leq i \leq 2n - m$, add the vertices g_i^1, g_i^2 and an edge between g_i^1 and g_i^2. Finally we add edges forming a complete bipartite graph between all vertices in $\{ g_i^o \mid 1 \leq i \leq 2n - m \wedge 1 \leq o \leq 2 \}$ and all vertices in $\{ h_i^o \mid 1 \leq i \leq n \wedge 1 \leq o \leq 2 \}$.

This completes the construction of G. The following claim concludes the first step of our reduction.

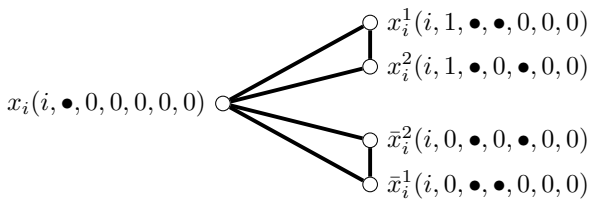


Figure 3. An illustration of the gadget $G(x_i)$ introduced in the reduction of Theorem 14. The label of each vertex v indicates the row vector $R(v)$.

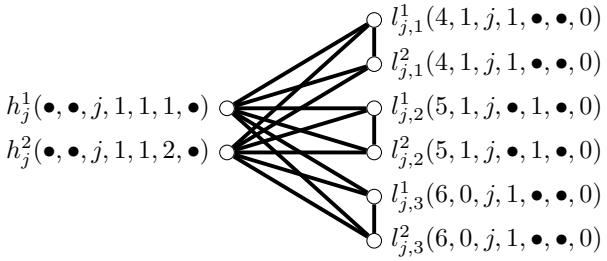


Figure 4. An illustration of the gadget $G(C_j)$ introduced in the reduction of Theorem 14. The label of each vertex v indicates the row vector $R(v)$; here we assume that $f(j,1) = x_4^1$, $f(j,2) = x_5^2$, and $f(j,3) = \bar{x}_6^1$.

Claim 3. *ϕ is satisfiable if and only if G has a partition into triangles. Moreover, G does not contain a clique of size four.*

Proof. We first show that G does not contain a clique of size four by showing that the neighborhood of any vertex in G does not contain a triangle.

- If $v = x_i$ for some i with $1 \leq i \leq n$, then $N_G(v) = \{x_i^1, x_i^2, \bar{x}_i^1, \bar{x}_i^2\}$ and does not contain a triangle.
- If $v = x_i^o$ for some i and o with $1 \leq i \leq n$ and $1 \leq o \leq 2$, then $N_G(v) = \{x_i, x_i^{o-1 \mod 2+1}, l_{j,r}^1, l_{j,r}^2\}$, where $f^{-1}(x_i^o) = (j, r)$, and does not contain a triangle.
- The case for $v = \bar{x}_i^o$ for i and o as above is analogous.
- If $v = l_{j,r}^o$ for some j, r, and o with $1 \leq j \leq m$, $1 \leq r \leq 3$, and $1 \leq o \leq 2$, then $N_G(v) = \{l_{j,r}^{o-1 \mod 2+1}, f(j,r), h_j^1, h_j^2\}$ and does not contain a triangle.
- If $v = h_j^o$ for some j and o with $1 \leq j \leq m$ and $1 \leq o \leq 2$, then $N_G(v) = \{\, l_{j,r}^o \mid 1 \leq r \leq 3 \wedge 1 \leq o \leq 2 \,\} \cup \{\, g_{j'}^1, g_{j'}^2 \mid 1 \leq j' \leq 2n - m \,\}$ and does not contain a triangle.
- If $v = g_j^o$ for some j and o with $1 \leq j \leq 2n - m$ and $1 \leq o \leq 2$, then $N_G(v) = \{g_j^{o-1 \mod 2+1}\} \cup \{\, h_{j'}^{o'} \mid 1 \leq j' \leq m \wedge 1 \leq o' \leq 2 \,\}$ and does not contain a triangle.

We now show that ϕ is satisfiable if and only if G has a partition into triangles. Towards showing the forward direction let τ be a satisfying assignment for ϕ. Then a partition $\mathcal{P}$ of G into triangles contains the following triangles:

(1) for every i with $1 \leq i \leq n$ the triangle x_i, x_i^1, x_i^2 if $\tau(x_i) = 0$ and the triangle x_i, $\bar{x}_i^1$, $\bar{x}_i^2$, otherwise,

(2) for every j with $1 \leq j \leq m$ and every r with $1 \leq r \leq 3$ such that the r-th literal of C_j is satisfied by τ, the triangle $l_{j,r}^1, l_{j,r}^2, f(j,r)$,

(3) for every j with $1 \leq j \leq m$ and every r with $1 \leq r \leq 3$ such that the r-th literal of C_j is not satisfied by τ, the triangle $l_{j,r}^1, l_{j,r}^2, h_j^o$, where $o \in \{1, 2\}$ and the r-th literal of C_j is the o-th literal in C_j that is not satisfied by τ; note that this is always possible because C_j has at most two literals that are not satisfied by τ,

(4) Let A be the subset of $\{\, h_i^o \mid 1 \leq i \leq n \wedge 1 \leq o \leq 2 \,\}$ containing all h_i^o that are not yet part of a triangle, i.e., that are not part of a triangle added in (3). Then $|A| = 2n - m$ and it is hence possible to add the following triangles, i.e., for every $v \in A$ a triangle containing v and the two vertices g_p^1 and g_p^2 for some p with $1 \leq p \leq 2n - m$.

Towards showing the reverse direction let $\mathcal{P}$ be a partition of $V(G)$ into $|V(G)|/3$ triangles. Then $\mathcal{P}$ satisfies:

(A1) For every i with $1 \leq i \leq n$, $\mathcal{P}$ either contains the triangle $\{x_i, x_i^1, x_i^2\}$ or the triangle $\{x_i, \bar{x}_i^1, \bar{x}_i^2\}$.

(A2) For every j with $1 \leq j \leq m$, $\mathcal{P}$ there is an r with $1 \leq r \leq 3$ such that $\mathcal{P}$ contains the triangle $\{l_{j,r}^1, l_{j,r}^2, f(j,r)\}$.

(A1) follows because these are the only two triangles in G containing x_i for every i with $1 \leq i \leq n$. Moreover, (A2) follows because for every j and r with $1 \leq j \leq m$ and $1 \leq r \leq 3$ there are only three triangles containing one of the vertices $l_{j,r}^1$ and $l_{j,r}^2$, i.e., the triangles $\{l_{j,r}^1, l_{j,r}^2, h_j^1\}$, $\{l_{j,r}^1, l_{j,r}^2, h_j^2\}$, and $\{l_{j,r}^1, l_{j,r}^2, f(j,r)\}$. (A2) now follows because $\mathcal{P}$ can contain at most two triangles containing one of h_i^1 and h_i^2. But then the assignment τ setting $\tau(x_i) = 1$ for every i with $1 \leq i \leq n$ if and only if $\mathcal{P}$ contains the triangle $\{x_i, \bar{x}_i^1, \bar{x}_i^2\}$ is a satisfying assignment for ϕ, because of (A2). □

We will now proceed to the second (and final) step of our reduction, i.e., we will construct an instance $(\mathbf{M}, |V(G)|/3)$ of DRMC such that: $G(\mathbf{M})$ is isomorphic to G and $\mathbf{M}$

has only seven columns. Because of Observation 1 and Lemma 9, this shows that DRMC is **NP**-hard already for only 7 columns and concludes the proof of the theorem.

$\mathbf{M}$ will contain one row $R(u)$ for every $u \in V(G)$, which is defined as follows.

- if $u = x_i$ for some i with $1 \leq i \leq n$, we set $R(u) = (i, \bullet, 0, 0, 0, 0, 0)$,
- if $u = x_i^o$ for some i and o with $1 \leq i \leq n$ and $1 \leq o \leq 2$, we set either:
  - $R(u) = (i, 1, \bullet, \bullet, 0, 0, 0)$, if $o = 1$ or
  - $R(u) = (i, 1, \bullet, 0, \bullet, 0, 0)$, otherwise
- if $u = \bar{x}_i^o$ for some i and o with $1 \leq i \leq n$ and $1 \leq o \leq 2$, we set either:
  - $R(u) = (i, 0, \bullet, \bullet, 0, 0, 0)$, if $o = 1$ or
  - $R(u) = (i, 0, \bullet, 0, \bullet, 0, 0)$, otherwise
- if $u = l_{j,r}^o$ for some j, r, and o with $1 \leq j \leq n$, $1 \leq r \leq 3$, and $1 \leq o \leq 2$, then either:
  - if x_i is the r-th literal of C_j (for some i with $1 \leq i \leq n$), then either:
    - if $f(j, r) = x_i^1$, we set $R(u) = (i, 1, j, 1, \bullet, \bullet, 0)$,
    - otherwise, i.e., if $f(j, r) = x_i^2$, we set $R(u) = (i, 1, j, \bullet, 1, \bullet, 0)$,
  - if $\bar{x}_i$ is the r-th literal of C_j (for some i with $1 \leq i \leq n$), then either:
    - if $f(j, r) = \bar{x}_i^1$, we set $R(u) = (i, 0, j, 1, \bullet, \bullet, 0)$,
    - otherwise, i.e., if $f(j, r) = \bar{x}_i^2$, we set $R(u) = (i, 0, j, \bullet, 1, \bullet, 0)$,
- if $u = h_j^o$ for some j and o with $1 \leq j \leq m$ and $1 \leq o \leq 2$, we set $R(u) = (\bullet, \bullet, j, 1, 1, o, \bullet)$,
- if $u = g_j^o$ for some j and o with $1 \leq j \leq 2n - m$ and $1 \leq o \leq 2$, we set $R(u) = (\bullet, \bullet, \bullet, \bullet, \bullet, \bullet, j)$,

Figures 3 and 4 illustrate the row vectors assigned to the vertices in the gadgets $G(x_i)$ and $G(C_j)$. It remains to show that $G(\mathbf{M})$ is indeed isomorphic to G. Let $u \in V(G)$, we distinguish the following cases:

- if $u = x_i$ for some i with $1 \leq i \leq n$, we need to show that $N_{G(\mathbf{M})}(u) = N_G(u) = \{x_i^1, x_i^2, \bar{x}_i^1, \bar{x}_i^2\}$. Since $R(x_i) = (i, \bullet, 0, 0, 0, 0, 0)$ is compatible with $R(x_i^1) = (i, 1, \bullet, \bullet, 0, 0, 0)$, $R(x_i^2) = (i, 1, \bullet, 0, \bullet, 0, 0)$, $R(\bar{x}_i^1) = (i, 0, \bullet, \bullet, 0, 0, 0)$, and $R(\bar{x}_i^2) = (i, 0, \bullet, 0, \bullet, 0, 0)$, we already have that $N_G(u) \subseteq N_{G(\mathbf{M})}(u)$. Moreover, $R(x_i) = (i, \bullet, 0, 0, 0, 0, 0)$ is not compatible with:
  - $R(x_{i'})$, $R(x_{i'}^o)$, or $R(\bar{x}_{i'}^o)$ for any i' and o with $1 \leq i' \leq n$, $i' \neq i$, and $1 \leq o \leq 2$ because the first column of these rows is equal to i' and $i' \neq i$.
  - $R(l_{j,r}^o)$ for any j, r, and o with $1 \leq j \leq m$, $1 \leq r \leq 3$, and $1 \leq o \leq 2$ because the third column of $R(l_{j,r}^o)$ is equal to j, and hence not equal to the corresponding column of $R(u)$, which is 0.
  - $R(h_j^o)$ for any j and o with $1 \leq j \leq m$ and $1 \leq o \leq 2$, because the third column of $R(h_j^o)$ is equal to j and $j \neq 0$.
  - $R(g_j^o)$ for any j and o with $1 \leq j \leq 2n - m$ and $1 \leq o \leq 2$, because the seventh column of $R(g_j^o)$ is equal to j and $j \neq 0$.

  This shows that $N_{G(\mathbf{M})}(u) \subseteq N_G(u)$ and hence $N_{G(\mathbf{M})}(u) = N_G(u)$, as required.
- if $u = x_i^1$ for some i with $1 \leq i \leq n$, we need to show that $N_{G(\mathbf{M})}(u) = N_G(u) = \{x_i, x_i^2, l_{j,r}^1, l_{j,r}^2\}$, where j and r are such that $f^{-1}(x_i^1) = (j, r)$. Since $R(x_i^1) = (i, 1, \bullet, \bullet, 0, 0, 0)$ is compatible with $R(x_i) = (i, \bullet, 0, 0, 0, 0, 0)$, $R(x_i^2) = (i, 1, \bullet, 0, \bullet, 0, 0)$, $R(l_{j,r}^1) = R(l_{j,r}^2) = (i, 1, j, 1, \bullet, \bullet, 0)$, we already have that $N_G(u) \subseteq N_{G(\mathbf{M})}(u)$. Moreover, $R(x_i^1) = (i, 1, \bullet, \bullet, 0, 0, 0)$ is not compatible with:
  - $R(x_{i'})$, $R(x_{i'}^o)$, or $R(\bar{x}_{i'}^o)$ for any i' and o with $1 \leq i' \leq n$, $i' \neq i$, and $1 \leq o \leq 2$ because the first column of these rows is equal to i' and $i' \neq i$.
  - $R(\bar{x}_i^o)$ for any o with $1 \leq o \leq 2$ because the second column of $R(\bar{x}_i^o)$ is equal to 0 and $0 \neq 1$.
  - $R(l_{j',r'}^o)$ for any j', r', and o with $1 \leq j' \leq m$, $1 \leq r' \leq 3$, $1 \leq o \leq 2$, and $(j', r') \neq (j, r)$ because either:
    - the r' literal of C_j is not x_i, in which case either the first or second column of $R(u)$ and $R(l_{j',r'}^o)$ differ or
    - the r' literal of C_j is x_i, in which case $f(j', r') = x_i^2$ and the fifth column of $R(l_{j',r'}^o)$ is equal to 1, and hence not equal to the fifth column of $R(u)$, which is 0.
  - $R(h_j^o)$ for any j and o with $1 \leq j \leq m$ and $1 \leq o \leq 2$, because the fifth column of $R(h_j^o)$ is equal to 1.
  - $R(g_j^o)$ for any j and o with $1 \leq j \leq 2n - m$ and $1 \leq o \leq 2$, because the seventh column of $R(g_j^o)$ is equal to j and $j \neq 0$.

  This shows that $N_{G(\mathbf{M})}(u) \subseteq N_G(u)$ and hence $N_{G(\mathbf{M})}(u) = N_G(u)$, as required.
- the cases for $u \in \{x_i^2, \bar{x}_i^1, \bar{x}_i^2\}$ are analogous to the previous case,

- if $u = l^1_{j,r}$ for some j and r with $1 \leq j \leq m$ and $1 \leq r \leq 3$, we have to show that $N_{G(\mathbf{M})}(u) = N_G(u) = \{l^2_{j,r}, f(j,r), h^1_j, h^2_j\}$. Assume w.l.o.g. that $f(j,r) = x^1_i$ for some i with $1 \leq i \leq n$. Then since $R(l^1_{j,r}) = (i,1,j,1,\bullet,\bullet,0)$ is compatible with $R(l^2_{j,r}) = R(u)$, $R(x^1_i) = (i,1,\bullet,\bullet,0,0,0)$, $R(h^1_j) = (\bullet,\bullet,j,1,1,1,\bullet)$, and $R(h^2_j) = (\bullet,\bullet,j,1,1,2,\bullet)$, we already have that $N_G(u) \subseteq N_{G(\mathbf{M})}(u)$. Moreover, $R(l^1_{j,r}) = (i,1,j,1,\bullet,\bullet,0)$ is not compatible with:
  - $R(x_i) = (i,\bullet,0,0,0,0,0)$ and $R(x^2_i) = (i,1,\bullet,0,\bullet,0,0)$ because of the fourth column,
  - $R(\bar{x}^o_i)$ for any o with $1 \leq o \leq 2$ because of the second column,
  - $R(x_{i'})$, $R(x^o_{i'})$, or $R(\bar{x}^o_{i'})$ for any i' and o with $1 \leq i' \leq n$, $i' \neq i$, and $1 \leq o \leq 2$ because the first column of these rows is equal to i' and $i' \neq i$.
  - $R(l^o_{j',r'})$ for any j', r', and o with $1 \leq j' \leq m$, $1 \leq r' \leq 3$, $(j',r') \neq (j,r)$, and $1 \leq o \leq 2$ because either:
    - $j' \neq j$ and they hence differ in the third column, or
    - $j' = j$ but $r' \neq r$ in which case they differ in the first or second column because the clause C_j cannot contain the same literal (x_i) twice.
  - $R(h^o_{j'})$ for any j' and o with $1 \leq j' \leq m$, $j' \neq j$, and $1 \leq o \leq 2$, because the third column of $R(h^o_{j'})$ is equal to j' and $j' \neq j$.
  - $R(g^o_{j'})$ for any j' and o with $1 \leq j' \leq 2n-m$ and $1 \leq o \leq 2$, because the seventh column of $R(g^o_{j'})$ is equal to j' and $j' \neq 0$.
- if $u = h^o_j$ for some j and o with $1 \leq j \leq m$ and $1 \leq o \leq 2$, we have to show that $N_{G(\mathbf{M})}(u) = N_G(u) = \{l^1_{j,r}, l^2_{j,r} \mid 1 \leq r \leq 3\{ \cup \mid g^1_{j'}, g^2_{j'} \mid 1 \leq j' \leq 2n-m \}$. Then since $R(h^o_j) = (\bullet,\bullet,j,1,1,o,\bullet)$ is compatible with $R(l^{o'}_{j,r})$ for any r and o' with $1 \leq r \leq 3$ and $1 \leq o' \leq 2$ – this is because the third column of $R(l^o_{j,r})$ is always equal to j, the fourth and fifth columns are always either $\bullet$ or 1, and the sixth column is always equal to $\bullet$ – and $R(h^o_j)$ is also clearly compatible with $R(g^1_{j'}) = R(g^2_{j'}) = (\bullet,\bullet,\bullet,\bullet,\bullet,\bullet,j')$ (for every j' with $1 \leq j' \leq 2m-n$), we already have that $N_G(u) \subseteq N_{G(\mathbf{M})}(u)$. Moreover, $R(h^o_j) = (\bullet,\bullet,j,1,1,o,\bullet)$ is not compatible with:
  - $R(x_i)$, $R(x^{o'}_i)$, and $R(\bar{x}^{o'}_i)$ for any i and o' with $1 \leq i \leq n$ and $1 \leq o' \leq 2$ because either the fourth or the fifth column of all these rows is equal to 0 and $0 \neq 1$.
  - $R(l^{o'}_{j',r'})$ for any j', r', and o' with $1 \leq j' \leq m$, $j' \neq j$, $1 \leq r' \leq 3$ and $1 \leq o' \leq 2$ because the third column of all these rows is j' and $j' \neq j$,
  - $R(h^{o-1 \mod 2+1}_j)$ because of the sixth column,
  - $R(h^{o'}_{j'})$ for any j' and o' with $1 \leq j' \leq m$ and $1 \leq o' \leq 2$ because the third column of all these vectors is j' and $j' \neq j$,
- if $u = g^o_j$ for some j and o with $1 \leq j \leq 2n-m$ and $1 \leq o \leq 2$, we have to show that $N_{G(\mathbf{M})}(u) = N_G(u) = \{g^{o-1 \mod 2+1}_j\} \cup \{ h^1_{j'}, h^2_{j'} \mid 1 \leq j' \leq m \}$. We have already shown in the previous case that g^o_j is adjacent to all vertices in $\{ h^1_{j'}, h^2_{j'} \mid 1 \leq j' \leq m \}$. Moreover, since $R(g^o_j) = R(g^{o-1 \mod 2+1}_j) = (\bullet,\bullet,\bullet,\bullet,\bullet,\bullet,j)$ we obtain that $N_G(u) \subseteq N_{G(\mathbf{M})}(u)$. Moreover, $R(g^o_j) = (\bullet,\bullet,\bullet,\bullet,\bullet,\bullet,j)$ is not compatible with any other row because the seventh column of any other row is not equal to j. □

We conclude this section with a hardness result showing that 2-DRMC remains **NP**-hard when the number of missing or known entries in each column/row is bounded.

Theorem 15. *The restriction of* 2-DRMC *to instances in which each row and each column contains exactly three missing entries is* **NP***-hard. The same holds for the restriction of* 2-DRMC *to instances in which each row and each column contains at most 4 determined entries.*

Proof. Consider the problem of (properly) coloring a graph on n vertices, having minimum degree $n-4$ and no independent set of size 4, by $n/3$ colors, where n is divisible by 3; denote this problem as $(n/3)$-COLORING$^{\delta=n-4}$.

We first show that this problem is **NP**-hard via a reduction from the PARTITION INTO TRIANGLES problem on K_4-free cubic graphs. The **NP**-hardness of the aforementioned problems follows from the **NP**-hardness of the PARTITION INTO TRIANGLES problem on planar cubic graphs (Cerioli et al., 2008), since a K_4 in a cubic graph must be isolated, and hence can be removed from the start.

Next, observe that PARTITION INTO TRIANGLES on K_4-free cubic graphs is polynomial-time reducible to $(n/3)$-COLORING$^{\delta=n-4}$, via the simple reduction that complements the edges of the graph. Given a K_4-free cubic graph G, the (edge) complement of G, $\overline{G}$, has minimum degree $n-4$, and contains no independent set of size more than 3. This can be seen as follows. Clearly, if G can be partitioned into triangles then $\overline{G}$ has an $(n/3)$-coloring. Conversely, if $\overline{G}$ has an $(n/3)$-coloring, then since $\overline{G}$ has no independent set of size more than 3, each of the $n/3$ color classes in $\overline{G}$ contains exactly 3 vertices. Therefore, G can be partitioned into triangles.

Finally, we reduce from $(n/3)$-COLORING$^{\delta=n-4}$ to p-DRMC by mimicking a standard reduction from 3-coloring to rank minimization (Peeters, 1996). Given an instance G of $(n/3)$-COLORING$^{\delta=n-4}$, we construct an $n \times n$ matrix

$\mathbf{M}$ whose rows and columns correspond to the vertices in G, as follows. The diagonal entries of $\mathbf{M}$ are all ones. For any entry at row i and column j, where $i \neq j$, $\mathbf{M}[i,j] = 0$ if $ij \in E(G)$, and is $\bullet$ otherwise. Finally, we set $r = n/3$. Observe that since each vertex in G has $n-4$ neighbors, the number of missing entries in any row of $\mathbf{M}$ is 3. Since $\mathbf{M}$ is symmetric, the number of missing entries in any column of $\mathbf{M}$ is 3 as well.

If G is a yes-instance of $(n/3)$-COLORING$^{\delta=n-4}$, then G can be partitioned into $n/3$ independent sets, each of size exactly 3. We claim that the missing entries in $\mathbf{M}$ can be filled so that the total number of identical rows is at most $n/3$. To shows this, it suffices to show that the three rows corresponding to the vertices in any of the $n/3$ independent sets in G can be filled so that to produce identical rows. To do so, consider such an independent set $\{u, v, w\}$ in G, and for convenience, denote their corresponding rows in $\mathbf{M}$ by u, v, w, respectively. Since u, v, w is an independent set, the two entries $\mathbf{M}[v, u]$ and $\mathbf{M}[w, u]$ are $\bullet$. Similarly, we have $\mathbf{M}[u, v] = \mathbf{M}[w, v] = \bullet$ and $\mathbf{M}[u, w] = \mathbf{M}[v, w] = \bullet$. (That is, the only entry containing "1" in any of the three rows has corresponding entries in the other two rows that are $\bullet$.) Therefore, if we replace all these six missing entries with 1, and all other missing entries with 0, we make the three rows u, v, w identical.

To prove the converse, suppose that the missing entries in $\mathbf{M}$ can be completed so to obtain at most $n/3$ identical rows. Note that, for any two adjacent vertices u, v, their rows cannot be completed into identical rows since u has 0 at column v, whereas v has 1. Therefore, all rows that are completed into the same identical row correspond to an independent set in G. Since G has no independent set of size more than 3, and since the number of identical rows in the completed matrix is at most $n/3$, it follows that the number of rows that have been completed into the same row is exactly 3, and those correspond to an independent set of size 3 in G. Therefore, G is a yes-instance of $(n/3)$-COLORING$^{\delta=n-4}$.

The second statement in the theorem follows via a reduction from 3-COLORING on graphs of maximum degree at most 4, which is known to be **NP**-hard (Garey et al., 1976), using similar arguments. □

5. Conclusion

We studied the parameterized complexity of two fundamental matrix completion problems under several parameterizations. For the bounded domain case, we painted a positive picture by showing that the two problems are in **FPT** (resp. $\mathbf{FPT}_R$) w.r.t. all considered parameters. For the unbounded domain case, we characterized the parameterized complexity of DRMC by showing that it is in **FPT** parameterized by row, and **paraNP**-hard parameterized by col (and hence by comb). For RMC, we could show its membership in **XP** (resp. $\mathbf{XP}_R$) w.r.t. all considered parameters. Three immediate open questions ensue:

- Is it possible to obtain a deterministic algorithm for p-RMC and RMC parameterized by comb?
- Can we improve our **XP** (resp. $\mathbf{XP}_R$) results for RMC to **FPT** (resp. $\mathbf{FPT}_R$) or show that the problems are **W**[1]-hard?
- Does a hardness result, similar to the one given in Theorem 15 for p-DRMC, hold for p-RMC?

Acknowledgments. Robert Ganian is also affiliated with FI MU, Czech Republic.

References

Matrix completion and the Netflix challenge. See https://en.wikipedia.org/wiki/Matrix_completion.

Bäckström, C., Jonsson, P., Ordyniak, S., and Szeider, S. A complete parameterized complexity analysis of bounded planning. *J. of Computer and System Sciences*, 81(7): 1311–1332, 2015.

Berman, P., Karpinski, M., and Scott, A. D. Approximation hardness of short symmetric instances of MAX-3SAT. *Electronic Colloquium on Computational Complexity (ECCC)*, (049), 2003.

Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Quimper, C., and Walsh, T. The parameterized complexity of global constraints. In Fox, D. and Gomes, C. P. (eds.), *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pp. 235–240. AAAI Press, 2008.

Bodlaender, H. L. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25 (6):1305–1317, 1996.

Bodlaender, H. L., Drange, P. G., Dregi, M. S., Fomin, F. V., Lokshtanov, D., and Pilipczuk, M. A $O(c^k n)$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.

Candès, E. J. and Plan, Y. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

Candès, E. J. and Recht, B. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

Candès, E. J. and Tao, T. The power of convex relaxation: near-optimal matrix completion. *IEEE Trans. Information Theory*, 56(5):2053–2080, 2010.

Cerioli, M. R., Faria, L., Ferreira, T. O., Martinhon, C. A. J., Protti, F., and Reed, B. A. Partition into cliques for cubic graphs: Planar case, complexity and approximation. *Discrete Applied Mathematics*, 156(12):2270–2278, 2008.

Courtois, N., Goubin, L., Meier, W., and Tacier, J. Solving underdefined systems of multivariate quadratic equations. In *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pp. 211–227. Springer, 2002.

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S. *Parameterized Algorithms*. Springer, 2015.

Downey, R. G. and Fellows, M. R. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer Verlag, 2013. ISBN 978-1-4471-5558-4, 978-1-4471-5559-1.

Elhamifar, E. and Vidal, R. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2765–2781, 2013.

Endriss, U., de Haan, R., and Szeider, S. Parameterized complexity results for agenda safety in judgment aggregation. In Weiss, G., Yolum, P., Bordini, R. H., and Elkind, E. (eds.), *Proceedings of AAMAS 2015, the 14th International Conference on Autonomous Agents and Multiagent Systems*, pp. 127–136. IFAAMAS/ACM, 2015.

Fazel, M. *Matrix rank minimization with applications*. PhD thesis, Stanford University, 2002.

Flum, J. and Grohe, M. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer Verlag, Berlin, 2006.

Frieze, A. M., Kannan, R., and Vempala, S. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004.

Ganian, R. and Ordyniak, S. The complexity landscape of decompositional parameters for ILP. *Artificial Intelligence*, 257:61 – 71, 2018.

Garey, M. R., Johnson, D. S., and Stockmeyer, L. J. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

Gaspers, S. and Szeider, S. Guarantees and limits of preprocessing in constraint satisfaction and reasoning. *Artificial Intelligence*, 216:1–19, 2014. doi: 10.1016/j.artint.2014.06.006.

Gottlob, G. and Szeider, S. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems. *The Computer Journal*, 51(3):303–325, 2006. Survey paper.

Hardt, M., Meka, R., Raghavendra, P., and Weitz, B. Computational limits for matrix completion. In *Proceedings of The 27th Conference on Learning Theory*, volume 35 of *JMLR Workshop and Conference Proceedings*, pp. 703–725. JMLR.org, 2014.

Ibarra, O. H., Moran, S., and Hui, R. A generalization of the fast LUP matrix decomposition algorithm and applications. *J. Algorithms*, 3(1):45–56, 1982.

Keshavan, R. H., Montanari, A., and Oh, S. Matrix completion from a few entries. *IEEE Trans. Information Theory*, 56(6):2980–2998, 2010a.

Keshavan, R. H., Montanari, A., and Oh, S. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11:2057–2078, 2010b.

Kloks, T. *Treewidth: Computations and Approximations*. Springer Verlag, Berlin, 1994.

Miura, H., Hashimoto, Y., and Takagi, T. Extended algorithm for solving underdefined multivariate quadratic equations. *IEICE Transactions*, 97-A(6):1418–1425, 2014.

Peeters, R. Orthogonal representations over finite fields and the chromatic number of graphs. *Combinatorica*, 16(3): 417–431, 1996.

Recht, B. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12:3413–3430, 2011.

Robertson, N. and Seymour, P. D. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

Saunderson, J., Fazel, M., and Hassibi, B. Simple algorithms and guarantees for low rank matrix completion over F_2. In *Proceedings of The IEEE International Symposium on Information Theory*, pp. 86–90. IEEE, 2016.

van Bevern, R., Komusiewicz, C., Niedermeier, R., Sorge, M., and Walsh, T. H-index manipulation by merging articles: Models, theory, and experiments. *Artif. Intell.*, 240:19–35, 2016.