# On the Complexity of Recovering Incidence Matrices

**Fedor V. Fomin**
University of Bergen, Norway
fomin@ii.uib.no

**Petr Golovach**
University of Bergen, Norway
Petr.Golovach@ii.uib.no

**Pranabendu Misra**
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
pmisra@mpi-inf.mpg.de

**M. S. Ramanujan**
University of Warwick, Coventry, UK
R.Maadapuzhi-Sridharan@warwick.ac.uk

## Abstract

The incidence matrix of a graph is a fundamental object naturally appearing in many applications, involving graphs such as social networks, communication networks, or transportation networks. Often, the data collected about the incidence relations can have some slight noise. In this paper, we initiate the study of the computational complexity of recovering incidence matrices of graphs from a binary matrix: given a binary matrix $M$ which can be written as the superposition of two binary matrices $L$ and $S$, where $S$ is the incidence matrix of a graph from a specified graph class, and $L$ is a matrix ($i$) of small rank or, ($ii$) of small (Hamming) weight. Further, identify all those graphs whose incidence matrices form part of such a superposition. Here, $L$ represents the noise in the input matrix $M$. Another motivation for this problem comes from the Matroid Minors project of Geelen, Gerards and Whittle, where perturbed graphic and co-graphic matroids play a prominent role. There, it is expected that a perturbed binary matroid (or its dual) is presented as $L + S$ where $L$ is a low rank matrix and $S$ is the incidence matrix of a graph. Here, we address the complexity of constructing such a decomposition.

When $L$ is of small rank, we show that the problem is NP-complete, but it can be decided in time $(mn)^{O(r)}$, where $m, n$ are dimensions of $M$ and $r$ is an upper-bound on the rank of $L$. When $L$ is of small weight, then the problem is solvable in polynomial time $(mn)^{O(1)}$. Furthermore, in many applications it is desirable to have the list of all possible solutions for further analysis. We show that our algorithms naturally extend to enumeration algorithms for the above two problems with delay $(mn)^{O(r)}$ and $(mn)^{O(1)}$, respectively, between consecutive outputs.

**2012 ACM Subject Classification** Theory of computation → Design and analysis of algorithms; Mathematics of computing → Enumeration

**Keywords and phrases** Graph Incidence Matrix, Matrix Recovery, Enumeration Algorithm

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2020.50

## 1 Introduction

Suppose that we are given a large binary data matrix $M$, and we know that this matrix is of the form $M = L + S$, where $S$ is the incidence matrix of an undirected graph and $L$ is a sparse binary matrix. We assume that the sums are taken over GF(2) and thus $1 + 1 = 0$. See Section 2 for a formal definition of incidence matrices. Now, consider the following two computational questions.

**Question 1:** How efficiently can we recover *some* incidence matrix $S$ such that $M = L + S$ as described above?

**Question 2:** How efficiently can we recover *all* possible graphs whose incidence matrices form part of such a superposition $M = L + S$?

Our main motivation for studying these questions comes from the study of perturbed graphic matroids, which naturally arise in many settings. A prominent example is the emerging Matroid Minors Project of Geelen, Gerards, and Whittle [5], mirroring the Graph Minors project. Let us recall that binary matroids are represented by a matrix over $\mathsf{GF}(2)$, and graphic matroids are a sub-class of binary matroids defined by the incidence matrices of graphs. Then, for each proper minor-closed class $\mathcal{M}$ of binary matroids, there exists a non-negative integer $r$ such that every well-connected matroid $M \in \mathcal{M}$ is either a perturbation of a graphic matroid or a co-graphic matroid [5]. In other words, either $M$ or $M^\star$ (the dual matroid) can be decomposed as $L + S$ where $S$ is the incidence matrix of a graph and $L$ is a matrix of rank at most $r$. Matroid Minors and perturbed matroids are expected to have important applications in matroid algorithms, similar to the applications of Graph Minors and $H$-minor free decompositions in graph algorithms. In the applications of perturbed binary matroids [6, 4], it is expected that a decomposition of the perturbed matroid $M$ into $L + S$ is given as a part of the input. A natural computational question that arises here is the construction of such a decomposition for a given matroid.

Such a problem can be seen as the problem of the recovery of incidence matrices, which is a fundamental representation of graphs. Often the data collected about the network incidence relations contains some slight noise or errors. Our objective is to decompose the data matrix $M$ into two sparse matrices $L$ and $S$, where $L$ is a low weight matrix, that is matrix with a small number of ones, representing the error or noise, and $S$ is the incidence matrix of some graph from a specified graph class.

Besides applications in matroid minors theory, the problem of superposing binary matrix $M$ in $L + S$ has strong connections to robust Principal Component Analysis (PCA), a popular approach to robust subspace learning by the decomposition of the data matrix into low rank and sparse components. Here we have as data a matrix $M$, which is the superposition of a low rank component $L$ and a sparse component $S$. In particular, this approach to robust PCA was popularized by Candès et al. [1], Wright et al. [11], and Chandrasekaran et al. [2]. Thus our problem can be seen as a variant of robust PCA for binary matrices, with additional constraint on sparse component $S$ of being incidence matrix of some graph. Other variants of robust PCA when the structure of the sparse matrix $S$ is imposed from the structure of some graph were studied in [10, 12].

In this paper, we initiate the study of the computational complexity of graph recovering problems and identify several settings which imply efficient algorithms for the questions stated above. Moreover, we also provide complexity theoretic lower bounds for certain other settings that preclude polynomial-time solvability of either question. Let us formally describe our contributions and state our results.

**Our contributions.** Our first contribution is the formulation of a pair of generic Graph Recovering problems dealing with two common notions of sparsity for $L$. Let $\mathcal{C}$ be a fixed class of simple graphs.

(MIN-RANK, $\mathcal{C}$)-GRAPH RECOVERY
**Input:** An $n \times m$ binary matrix $M$ and $r \in \mathbb{N}$.
**Question:** Decide whether $M = L + S$, where $L$ is a binary matrix of GF(2)-rank at most $r$, $S$ is the incidence matrix of a graph in $\mathcal{C}$ and the sums are taken over GF(2).

Let $\|L\|$ denote the *(Hamming) weight* of a matrix $L$, i.e. the number of non-zero entries in $L$.

(MIN-WEIGHT, $\mathcal{C}$)-GRAPH RECOVERY
**Input:** An $n \times m$ binary matrix $M$ and $r \in \mathbb{N}$.
**Question:** Decide whether $M = L + S$, where $L$ is a binary matrix of weight at most $r$, $S$ is the incidence matrix of a graph in $\mathcal{C}$ and the sums are taken over GF(2).

Our second contribution is establishing the computational complexity of (MIN-RANK, $\mathcal{C}$)-GRAPH RECOVERY and (MIN-WEIGHT, $\mathcal{C}$)-GRAPH RECOVERY for some fundamental graph classes and obtain the following upper and lower bounds.

**(a)** We show that if $\mathcal{C}$ is one of {simple graphs, acyclic graphs, trees, arboricity-$d$ graphs, connected graphs}, then (MIN-RANK, $\mathcal{C}$)-GRAPH RECOVERY can be solved in time $(mn)^{\mathcal{O}(r)}$.
**(b)** We show that if $\mathcal{C}$ is one of {simple graphs, acyclic graphs, trees, connected graphs}, then (MIN-WEIGHT, $\mathcal{C}$)-GRAPH RECOVERY is polynomial-time solvable, i.e., in time $(mn)^{\mathcal{O}(1)}$.
**(c)** We show that the running time in Result (a) cannot be improved to a purely polynomial dependence on $m$ and $n$ (as in Result (b)) unless P=NP. Specifically, we show that if $\mathcal{C}$ is any of {simple graphs, acyclic graphs, trees, arboricity-$d$ graphs, connected graphs}, then the (MIN-RANK, $\mathcal{C}$)-GRAPH RECOVERY problem is NP-complete.

A key feature of the methodology we introduce to obtain our algorithms is that it not only answers Question 1 (i.e., the decision problem) for the settings above, it also naturally extends to an algorithm that addresses the significantly harder Question 2 (i.e., the enumeration of all solutions) in these settings. Specifically, we show that in the case of Result (a) above, we can also *enumerate* all possible solution matrices $S$ (if any exist) with a delay of time $(mn)^{\mathcal{O}(r)}$ between consecutive outputs. Similarly, we show that in the case of Result (b), we can also enumerate all possible solution matrices $S$ (and equivalently, the corresponding graphs) with polynomial delay, i.e., a delay of time $(mn)^{\mathcal{O}(1)}$ between consecutive outputs. The importance of enumeration lies in the fact that a solution $M = S + L$ may not be unique, and further analysis of the list of all solutions is required.

## Roadmap of the paper

Following the introduction of the requisite notation, we first describe the methodology behind our enumeration algorithms, where we reduce the problem of enumerating all solutions to one of *deciding* whether a solution exists for appropriate annotated variants of (MIN-RANK, $\mathcal{C}$)-GRAPH RECOVERY and (MIN-WEIGHT, $\mathcal{C}$)-GRAPH RECOVERY. We then present our decision algorithms for the aforementioned annotated problems. The precise formulation of these problems involves some notation and we do not go into more details here. The algorithms for the annotated variant of (MIN-RANK, $\mathcal{C}$)-GRAPH RECOVERY are centred around a novel application of Matroid Intersection involving carefully chosen matroids. On the other hand, the algorithms for (MIN-WEIGHT, $\mathcal{C}$)-GRAPH RECOVERY are designed through an intricate analysis of the structure of bridges (or cut edges) in graphs and appropriate repeated reassignments of these to columns of our input matrix in a manner reminiscent of the "augmenting path" step in maximum matching algorithms. Finally, we give our NP-completeness result for (MIN-RANK, $\mathcal{C}$)-GRAPH RECOVERY.

## 2 Preliminaries

### Matrices and Linear Algebra

For $\ell \in \mathbb{N}$, we use $[\ell]$ to denote the set $\{1, \ldots, \ell\}$. For a matrix $M$, we denote the set of rows of $M$ by $\text{rows}(M)$ and the set of column of $M$ by $\text{cols}(M)$. For a set $P \subseteq \text{cols}(M)$, we denote by $M[P]$ the submatrix of $M$ induced by the columns in $P$. Consider the set $V = \{v_1, v_2, \ldots, v_k\}$ of vectors over $\mathbb{F}$. The vectors in $V$ are said to be *linearly dependent* if there exist elements $a_1, a_2, \ldots, a_k \in \mathbb{F}$, not all zero, such that $\sum_{i=1}^{k} a_i v_i = 0$. Otherwise these vectors are said to be *linearly independent*. The *rank* of a matrix is the cardinality of a maximum sized set of columns which are linearly independent. The vector space *spanned by $V$* is the set of all linear combinations of vectors in $V$ and is denoted by $\text{span}(V)$. The vector space spanned by the set of columns of a matrix $M$ over the field $\mathbb{F}$ is defined as $\text{span}(\text{cols}(M))$ and is denoted by $\text{col-span}(M)$. We say a matrix $A$ has dimension $n \times m$ (or is an $n \times m$ matrix) if $A$ has $n$ rows and $m$ columns. In this paper we always view the elements of a binary matrix as elements of $\mathsf{GF}(2)$, the Galois field of two elements. Then the $\mathsf{GF}(2)$-rank of a binary $n \times m$ matrix $A$ is the minimum $r$ such that $A = U \cdot V$, where $U$ and $V$ are $n \times r$ and $r \times m$ binary matrices respectively, and arithmetic operations are over $\mathsf{GF}(2)$.

### Graphs

For standard graph theoretic terminology and notation, we refer to [3]. For an undirected graph $G$, we denote by $\text{inc}(G)$, the *incidence matrix* of $G$ which is the $|V(G)| \times |E(G)|$ binary matrix $M$ with a row for each vertex and a column for each edge such that for every $v \in V(G)$ and $e \in E(G)$, $M[v, e] = 1$ if and only if $v$ is an endpoint of $e$. When considering the incidence matrix $M$ of an unspecified graph, we denote by $\text{inc}^{-1}(M)$ the graph $G$ where $V(G) = \text{rows}(M)$, $E(G) = \text{cols}(M)$ and whose incidence matrix is precisely $M$. For $\ell \in \mathbb{N}$, we denote by $K_\ell$ the complete graph on $\ell$ vertices. We assume without loss of generality that the vertices of $K_\ell$ are labelled 1 to $\ell$ and unless otherwise specified, the columns in $\text{inc}(K_n)$ are assumed to be arranged in lexicographically increasing order based on the endpoints of the corresponding edges.

### Matroids

We recall relevant definitions related to matroids. For a broader overview on matroids, we refer to [9].

▶ **Definition 1.** *A matroid $M$ is a pair $(E, \mathcal{I})$, where $E$ is a set called the universe or ground set, and $\mathcal{I}$ is a family of subsets of $E$, called independent sets, with the following three properties : (i) $\emptyset \in \mathcal{I}$, (ii) if $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$, and (iii) if $I, J \in \mathcal{I}$ and $|I| < |J|$, then there is $e \in J \setminus I$ such that $I \cup \{e\} \in \mathcal{I}$.*

Any set $F \subseteq E$, $F \notin \mathcal{I}$, is called a *dependent set* and an inclusion-wise maximal set $B$ such that $B \in \mathcal{I}$ is called a *basis*. The cardinality of a basis in a matroid $M$ is called the *rank* of $M$ and is denoted by $\mathsf{rank}(M)$. The rank function of $M$, denoted by $\text{rank}_M()$ (with the reference to $M$ omitted when clear from the context), is a function from $2^E$ to $\mathbb{N} \cup \{0\}$ and is defined as, $\text{rank}(S) = \max_{S' \subseteq S, S' \in \mathcal{I}} |S'|$ for every $S \subseteq E$.

▶ **Definition 2.** *Let $A$ be a matrix over a field $\mathbb{F}$ and $E$ be the set of columns of $A$. The pair $(E, \mathcal{I})$, where $\mathcal{I}$ defined as follows, is a matroid. For every $X \subseteq E$, $X \in \mathcal{I}$ if and only if the columns of $A$ corresponding to $X$ are linearly independent over $\mathbb{F}$. Such matroids are called* linear matroids*. If a matroid $M$ can be defined by a matrix $A$ over a field $\mathbb{F}$, then we say that the matroid is* representable over $\mathbb{F}$ *and $A$ is a linear representation of $M$.*

▶ **Definition 3** (Elongation of matroids). *Let $M = (E, \mathcal{I})$ be a matroid and $k \in \mathbb{N}$. Suppose that* $\text{rank}(M) \leq k \leq |E|$. *A $k$-elongation matroid $M_k$ of $M$ is a matroid with the universe as $E$ and $S \subseteq E$ is a basis of $M_k$ if and only if, it contains a basis of $M$ and $|S| = k$.*

Observe that the rank of the matroid $M_k$ in the definition above is $k$.

▶ **Definition 4** (Direct-sum of matroids). *Consider a set of $\ell$ matroids $\{M_i = (E_i, \mathcal{I}_i) \mid i \in [\ell]\}$, where $E_i \cap E_j = \emptyset$ for every $i \neq j$. The* direct-sum *of these matroids is the matroid $M = (\bigcup_{i \in [\ell]} E_i, \mathcal{I})$ where $I \in \mathcal{I}$ if and only if $I = \bigcup_{i \in [\ell]} I_i$ where $I_i \in \mathcal{I}_i$ for every $i \in [\ell]$.*

Given the representations of $\ell$ linear matroids over $\mathbb{F}$, it is straightforward to obtain a representation for their direct-sum over $\mathbb{F}$ and this can be done in polynomial time.

For a finite field $\mathbb{F}$, $\mathbb{F}[X]$ denotes the *ring of polynomials in $X$ over $\mathbb{F}$* and $\mathbb{F}(X)$ denotes the *field of fractions* of $\mathbb{F}[X]$. A vector $v$ over a field $\mathbb{F}$ is a tuple of elements from $\mathbb{F}$.

The next two propositions follow from [8].

▶ **Proposition 5** ([8]). *Let $M$ be a linear matroid of rank $r$, over a ground set of size $n$, which is representable over a field $\mathbb{F}$. Given $k \geq r$, we can compute a representation of the $k$-elongation of $M$, over the field $\mathbb{F}(X)$ in $\mathcal{O}(nrk)$ field operations over $\mathbb{F}$.*

▶ **Proposition 6** ([8]). *Given a linear representation of the $k$-elongation of $M$ over the field $\mathbb{F}(X)$ and a set of columns in the representation matrix, one can test for linear dependence of this set in polynomial time.*

In the Weighted Matroid Intersection problem, the input is a pair of matroids $M_1 = (E, \mathcal{I}_1), M_2 = (E, \mathcal{I}_2)$ and a weight function $w : E \to \mathbb{N} \cup \{0\}$ and the objective is to find a maximum-weight common independent set in the two matroids. That is, the goal is to compute $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ such that $\Sigma_{e \in I} w(e)$ is maximized. We will use the polynomial-time algorithm for this problem stated in Proposition 7.

▶ **Proposition 7** ([7]). *Given two general matroids $M_1$ and $M_2$ over the element set $E$, one can solve Weighted Matroid Intersection in time $O(\tau W n r^{1.5})$, where $W$ is the largest weight assigned to an element, $n = |E|$, $r = min\{\text{rank}(M_1), \text{rank}(M_2)\}$ and $\tau$ is the time required to test whether a given subset of $E$ is independent in $M_1$ and $M_2$.*

**Graphic Matroids.** These are matroids that arise from graphs in the following way. The graphic matroid $M(G)$ of an undirected graph $G$ has universe $E(G)$ and a set $S \subseteq E(G)$ is independent if the subgraph with vertex set $V(G)$ and edge set $S$ is acyclic. Graphic matroids are representable over every field and a representation of $M(G)$ over $\mathsf{GF}(2)$ can be computed in time polynomial in the size of $G$ [9]. Observe that testing whether a set is independent in the matroid $M(G)$ can be done in polynomial time if one is given either the linear representation of the matroid or the graph $G$.

**Transversal Matroids.** For a bipartite graph $G = (X, Y, E)$, we can define a matroid $M$ with universe $X$, where a set $S \subseteq X$ is independent if there exists a matching in $G$ such that every vertex in $S$ is an endpoint of a matching edge. We denote this matroid by $\text{Tr}(G, X)$.

Observe that if $G$ is given, then one can use a *Maximum Matching* algorithm as a subroutine to determine in polynomial time (in the size of $G$) whether a given set is independent in $\text{Tr}(G, X)$.

**Gammoids.** Let $D$ be a digraph and $S, T \subseteq V(D)$. Then a *gammoid* with respect to $D$ and $S$ on ground set $T$ is a matroid $(T, \mathcal{I})$, where $\mathcal{I}$ is defined as follows. For any $T' \subseteq T$, $T' \in \mathcal{I}$, if and only if there are $|T'|$ vertex disjoint paths which originate in $S$ and end in $T'$. Observe that if $D, S$ are given, then one can use a *Maximum Flow* algorithm as a subroutine to determine in polynomial time (in the size of $D$) whether a given set is independent in this gammoid.

▶ **Definition 8.** *A pair of matroids $M_1 = (E_1, \mathcal{I}_1), M_2 = (E_2, \mathcal{I}_2)$ are said to be* isomorphic *if there is a bijection $\phi : E_1 \to E_2$ such that for every $S \subseteq E_1$, $S \in \mathcal{I}_1$ if and only if $\phi(S) \in \mathcal{I}_2$ where the function $\phi$ is extended in the natural way to subsets of $E_1$. Equivalently, we say that $M_1$ and $M_2$ are isomorphic under the bijection $\phi$.*

## 3 The enumeration algorithms for $(\textsf{min-rank}, \mathcal{C})$-Graph Recovery and $(\textsf{min-weight}, \mathcal{C})$-Graph Recovery

This section is devoted to our enumeration algorithms. We begin by reducing the enumeration task to one of deciding an annotated version of the problem at hand.

### 3.1 Reducing enumeration to decision

Our enumeration strategy is based on first designing an algorithm for the decision version of an "annotated" version of these problems. In this version, certain columns of the input matrix $M$ already have their corresponding columns in the hypothetical solution matrix $S$ (equivalently, in the matrix $L$) identified and the goal is to check whether this partial mapping can be extended to a full solution. We now formally define the annotated versions of these problems.

In the Extended $(\textsf{min-rank}, \mathcal{C})$-Graph Recovery problem, we are given an $n \times m$ binary matrix $M$, number $r \in \mathbb{N}$, a set $C_M \subseteq \text{cols}(M)$ and an injective mapping $\tau : C_M \to \textsf{GF}(2)^n$ and the task is to *decide whether there exist* binary matrices $L$ and $S$ such that $M = L + S$, $S$ is an incidence matrix of a simple graph belonging to the class $\mathcal{C}$, $\text{rank}(L) \leq r$ and moreover, for every $x \in C_M$, $S[\{x\}] = \tau(x)$. Similarly in the Extended $(\textsf{min-weight}, \mathcal{C})$-Graph Recovery problem, the input is the same and the the goal is to decide whether there exist binary matrices $L$ and $S$ such that $M = L + S$, $S$ is an incidence matrix of a simple graph belonging to the class $\mathcal{C}$, $\|L\| \leq r$ and for every $x \in C_M$, $S[\{x\}] = \tau(x)$.

We remark that if, in an instance $(M, r, C_M, \tau)$ of either problem, it holds that $C_M = \emptyset$, then the corresponding mapping $\tau : C_M \to \textsf{GF}(2)^n$ is denoted by $\tau_\emptyset$. Moreover, notice that by setting $C_M = \emptyset$ and $\tau = \tau_\emptyset$, we have that $(\textsc{min-rank}, \mathcal{C})$-Graph Recovery ($(\textsc{min-weight}, \mathcal{C})$-Graph Recovery) is a special case of Extended $(\textsc{min-rank}, \mathcal{C})$-Graph Recovery (respectively, Extended $(\textsc{min-weight}, \mathcal{C})$-Graph Recovery).

Our enumeration algorithms are based on the following pair of algorithms for Extended $(\textsc{min-rank}, \mathcal{C})$-Graph Recovery and Extended $(\textsc{min-weight}, \mathcal{C})$-Graph Recovery.

▶ **Lemma 9.** *For every $\mathcal{C} \in \{$simple graph, forest, connected graph, arboricity-d graph$\}$, there is an algorithm $\mathcal{C}$-$\mathcal{D}$ that runs in time $(mn)^{\mathcal{O}(r)}$ and correctly decides whether or not the given input $(M, r, C_M, \tau)$ is a yes-instance of Extended $(\textsf{min-rank}, \mathcal{C})$-Graph Recovery.*

**Algorithm 1** Algorithm $\mathcal{C}$-$\mathcal{E}$.

```
if $C_M = [m]$ then
    Define the matrix $S$ as $S[\{x\}] = \tau(x)$ for every $x \in [m]$.
    if $\mathrm{inc}^{-1}(S) \in \mathcal{C}$ and $\mathrm{rank}(M + S) \leq r$ then
        Output $S$
    end
end
return
$x \leftarrow min_{p\in[m]}\{p \notin C_M\}$
for each $y \in \mathrm{cols}(\mathrm{inc}(K_n))$ such that $\tau^{-1}(y)$ is undefined do
    if $\mathcal{C}$-$\mathcal{D}((M, r, C_M \cup \{x\}, \tau \cup \{x \longmapsto y\}))$ returns Yes then
        $\mathcal{C}$-$\mathcal{E}((M, r, C_M \cup \{x\}, \tau \cup \{x \longmapsto y\}))$
    end
end
return
```

▶ **Lemma 10.** *For every $\mathcal{C} \in \{$simple graph, forest, connected graph$\}$, there is an algorithm $\mathcal{C}$-$\mathcal{D}'$ that runs in time $(mn)^{\mathcal{O}(1)}$ and correctly decides whether or not the given input $(M, r, C_M, \tau)$ is a yes-instance of Extended $($min-weight$, \mathcal{C})$-Graph Recovery.*

We first assume Lemma 9 and Lemma 10, and present our enumeration algorithms (Theorem 11 and Theorem 12).

▶ **Theorem 11.** *For every $\mathcal{C} \in \{$simple graph, forest, connected graph, arboricity-d graph$\}$, there is an algorithm $\mathcal{C}$-$\mathcal{E}$ that, on input $(M, r, C_M, \tau)$, outputs precisely those matrices $S$ such that $M = S + L$ for some binary matrix $L$ of GF$(2)$-rank at most $r$, $S$ is an incidence matrix of a simple graph in $\mathcal{C}$ such that for every $x \in C_M$, $S[\{x\}] = \tau(x)$. Moreover, the delay between successive outputs is $(mn)^{O(r)}$ and each such matrix $S$ is output exactly once.*

**Proof.** The algorithm $\mathcal{C}$-$\mathcal{E}$ is described in Algorithm 1. The **for** loop is executed at most $\binom{n}{2}$ times in any single call to the algorithm and the recursive call to $\mathcal{C}$-$\mathcal{E}$ is made precisely when there is at least one solution to be output for a specific extension of $C_M$ and $\tau$. This is witnessed by the positive answer returned by the execution of the decision algorithm $\mathcal{C}$-$\mathcal{D}$. Finally, since the depth of the recursion is bounded by $m$, the first part of the lemma follows. The fact that this algorithm outputs precisely the required matrices (exactly once each) with delay bounded by $(mn)^{O(r)}$ follows from the correctness and running time bound of the algorithm $\mathcal{C}$-$\mathcal{D}$ in Lemma 9. ◀

The enumeration algorithm for (MIN-WEIGHT, $\mathcal{C}$)-GRAPH RECOVERY is analogous to that for (MIN-RANK, $\mathcal{C}$)-GRAPH RECOVERY (building on Lemma 10 instead of Lemma 9) and so we only state the theorem, omitting the proof.

▶ **Theorem 12.** *For every $\mathcal{C} \in \{$simple graph, forest, connected graph, arboricity-d graph$\}$, there is an algorithm $\mathcal{C}$-$\mathcal{E}'$ that, on input $(M, r, C_M, \tau)$, outputs precisely those matrices $S$ such that $M = S + L$ for some binary matrix $L$ such that $\|L\| \leq r$, $S$ is an incidence matrix of a simple graph in $\mathcal{C}$ such that for every $x \in C_M$, $S[\{x\}] = \tau(x)$. Moreover, the delay between successive outputs is $(mn)^{O(1)}$ and each such matrix $S$ is output exactly once.*

Notice that in order to enumerate all possible solutions for an instance $(M, r)$ of $(\textsc{min-rank}, \mathcal{C})$-Graph Recovery ($(\textsc{min-weight}, \mathcal{C})$-Graph Recovery), it is sufficient to execute the enumeration algorithm $\mathcal{C}$-$\mathcal{E}$ (respectively, $\mathcal{C}$-$\mathcal{E}'$) on input $(M, r, \emptyset, \tau_\emptyset)$. We next prove Lemma 9.

## 3.2 Decision algorithms for Extended (min-rank, $\mathcal{C}$)-Graph Recovery

The goal of this subsection is to prove Lemma 9. We always assume that without loss of generality, the input $M$ satisfies: $m, n \geq r$. Note that since we require that for every $x \in C_M$, $S[\{x\}] = \tau(x)$, the range of $\tau$ can be assumed to be $\text{cols}(\text{inc}(K_n))$.

Our algorithms for deciding the Extended $(\textsc{min-rank}, \mathcal{C})$-Graph Recovery problem comprise the following two steps, the first of which only relies on $S$ being an incidence matrix and $L$ being a matrix of rank at most $r$. The second step depends on the class $\mathcal{C}$ under consideration and we will describe it in detail for each application separately.

1. We show that given $M$, one can enumerate in time $(mn)^{\mathcal{O}(r)}$, a set $\mathcal{Q}^r_M$ of sufficiently few sets, each of which contains at most $r$ $n$-dimensional binary vectors such that for every $L, S$ such that $M = L + S$, $\text{rank}(L) \leq r$ and $S$ is an incidence matrix, there is a $Q \in \mathcal{Q}^r_M$ whose elements form a basis for $\text{col-span}(L)$.
2. We then show that for each class $\mathcal{C}$ that we consider, if one is *given* a basis for the vector space spanned by the columns of the hypothetical solution matrix $L$ of rank $\leq r$, then it is possible to determine the existence of the required matrix $S$ in polynomial time (in the size of $M$) using an appropriate Weighted Matroid Intersection algorithm as a subroutine.

We begin by formalizing Step 1 of our algorithms.

▶ **Lemma 13.** *Given $M$ and $r$, one can enumerate in time $(mn)^{O(r)}$, a set $\mathcal{Q}^r_M = \{Q_1, \ldots, Q_\ell\}$ satisfying the following properties: (a) $\ell = (mn)^{O(r)}$, (b) each $Q_i$ is a set of $r$ $n$-dimensional vectors over $\mathsf{GF}(2)$, (c) for every pair of matrices $L, S$ such that $M = L + S$, $S$ is an incidence matrix and $\text{rank}(L) \leq r$, there is an $i \in [\ell]$ such that $Q_i$ is a basis for $\text{col-span}(L)$.*

In the rest of this section, for given $r$ and $M$, we denote by $\mathcal{Q}^r_M$ the set described in Lemma 13.

▶ **Definition 14.** *For an $n \times m$ binary matrix $M$ and a set $Q \in \binom{\mathsf{GF}(2)^n}{\leq r}$, we denote by $G^Q_M$ the bipartite graph $(X^Q_M = [m], Y^Q_M = \text{cols}(\text{inc}(K_n)), E^Q_M)$ where the edge set $E^Q_M$ is defined as all those pairs $(x, y)$ such that $x \in [m], y \in \text{cols}(\text{inc}(K_n))$ and satisfying $\exists q \in \text{span}(Q)$ such that $M[\{x\}] = y + q$.*

Note that in the graph $G^Q_M$, the set $Y^Q_M$ contains all columns of the incidence matrix of *every* simple graph on $n$ vertices.

▶ **Observation 15.** *Given $M$, $r$ and $Q$, the graph $G^Q_M$ can be computed in time $2^r \cdot (m+n)^{O(1)}$ and has $m + \binom{n}{2}$ vertices.*

The time bound in the above observation comes from the fact that for each $x \in X^Q_M$ and $y \in Y^Q_M$, deciding whether $(x, y) \in E^Q_M$ can be done by iterating over all the at most $2^r$ vectors in $\text{span}(Q)$.

▶ **Definition 16.** *Consider an $n \times m$ binary matrix $M$, $Q \in \binom{\mathsf{GF}(2)^n}{\leq r}$, $C_M \subseteq \text{cols}(M)$ and an injective mapping $\tau : C_M \to \mathsf{GF}(2)^n$. If $Z = \{(x, \tau(x)) \mid x \in C_M\}$ is a matching in the graph $G^Q_M$, then we denote by $\widetilde{G}^Q_M$ the graph obtained from $G^Q_M$ by deleting all edges incident* *on the set $V(Z)$ except the edges in $Z$. Otherwise, we denote by $\widetilde{G}^Q_M$ the graph obtained from $G^Q_M$ by deleting every edge. Then, $\mathrm{Tr}(G^Q_M, Y^Q_M, \langle C_M, \tau\rangle)$ denotes the transversal matroid $\mathrm{Tr}(\widetilde{G}^Q_M, Y^Q_M)$.*

▶ **Definition 17.** *Let $\phi : Y^Q_M \to E(K_n)$ be the trivial bijection which maps $y \in Y^Q_M$ to the corresponding edge of $K_n$ (recall that $Y^Q_M$ is precisely the set of columns of the incidence matrix of $K_n$). We denote by $\mathcal{M}(K_n, Y^Q_M)$ the matroid over the element set $Y^Q_M$ which is isomorphic to the graphic matroid $M(K_n)$ under the bijection $\phi$. We assume that $\phi$ is extended in the natural way to subsets of vertices. That is, for every $Y \subseteq Y^Q_M$, $\phi(Y) = \bigcup_{y\in Y} \phi(y)$.*

▶ **Observation 18.** *Let $\phi$ and $\mathcal{M}(K_n, Y^Q_M)$ be as in Definition 17. For every $n \times m$ incidence matrix $S$, the graph $\mathrm{inc}^{-1}(S)$ is isomorphic to the subgraph of $K_n$ with edge set $\phi(\mathrm{cols}(S))$.*

We are now ready to give our algorithms for each choice of $\mathcal{C}$.

### 3.2.1 Recovering simple graphs

▶ **Lemma 19.** *$(M, r, C_M, \tau)$ is a yes-instance of* Extended-(min-rank, simple)-*Graph Recovery if and only if there is a $Q \in \mathcal{Q}^r_M$ such that the graph $G^Q_M$ has a matching saturating $X^Q_M$ and containing the edges $\{(x, \tau(x)) \mid x \in C_M\}$.*

**Proof.** In the forward direction, suppose that $M = S + L$ where $\mathrm{inc}^{-1}(S)$ is a simple graph, $\mathrm{rank}(L) \le r$ and for every $x \in C_M$, $S[\{x\}] = \tau(x)$. By Lemma 13, we know that some $Q \in \mathcal{Q}^r_M$ is a basis for col-span$(L)$. Consequently, by Definition 14, for each $x \in [m]$, there is an edge in $G^Q_M$ between $x$ and $S[\{x\}]$. Since no 2 columns in $S$ are identical ($\mathrm{inc}^{-1}(S)$ is a simple graph), this implies a matching saturating $X^Q_M$ in $G^Q_M$ and extending the set $\{(x, \tau(x)) \mid x \in C_M\}$ as required.

Conversely, suppose that there is a matching $C$ saturating $X^Q_M$ in $G^Q_M$ and extending the set $\{(x, \tau(x)) \mid x \in C_M\}$. For each $x \in X^Q_M = [m]$, we denote by $x_C$ the partner of $x$ in $C$. By definition, for every $x \in C_M$, $x_C = \tau(x)$. We define $S$ and $L$ as follows. For each $x \in [m]$, set $S[\{x\}] = x_C$ and $L[\{x\}] = M[\{x\}] + x_C$. Since $S$ contains only distinct columns and these are all contained in $\mathrm{cols}(\mathrm{inc}(K_n))$, it follows that $S$ is the incidence matrix of a simple graph. On the other hand, Definition 14 implies that for every $x \in [m]$, since $(x, x_C) \in E^Q_M$, it must be the case that $L[\{x\}] = M[\{x\}] + x_C \in \mathrm{span}(Q)$. Consequently we have that col-span$(L) \subseteq \mathrm{span}(Q)$ and since $|Q| \le r$, the lemma follows. ◀

Lemma 13 and Lemma 19 imply our algorithm for Extended-(min-rank, simple)-Graph Recovery.

▶ **Lemma 20.** Extended-(min-rank, simple)-*Graph Recovery can be solved in time $(mn)^{O(r)}$.*

Observe that based on Lemma 19, we may conclude that if the given instance is positive, then there is a solution $M = S + L$ where the columns of $S$ form an independent set of size $m$ in the transversal matroid $\mathrm{Tr}(G^Q_M, Y^Q_M)$ for some $Q \in \mathcal{Q}^r_M$. Moreover, there must be such an independent set saturated by a matching extending $\{(x, \tau(x)) \mid x \in C_M\}$. Consequently, we have the following observation which forms a critical part of all our algorithms.

▶ **Observation 21.** *$(M, r, C_M, \tau)$ is a yes-instance of* Extended-(min-rank, simple)-*Graph Recovery if and only if there is a $Q \in \mathcal{Q}^r_M$ such that there is an independent set of size $m$ in the transversal matroid $\mathrm{Tr}(G^Q_M, Y^Q_M, \langle C_M, \tau\rangle)$.*

### 3.2.2 Recovering acyclic graphs

▶ **Lemma 22.** *$(M, r, C_M, \tau)$ is a yes-instance of* Extended (min-rank, acyclic)-Graph Recovery *if and only if there is a $Q \in \mathcal{Q}^r_M$ such that there is a common independent set of size $m$ in the transversal matroid $\mathrm{Tr}(G^Q_M, Y^Q_M, \langle C_M, \tau\rangle)$ and the matroid $\mathcal{M}(K_n, Y^Q_M)$.*

**Proof.** In the forward direction, suppose that $M = S + L$ where $\mathrm{inc}^{-1}(S)$ is a forest, for every $x \in C_M$, $S[\{x\}] = \tau(x)$ and $\mathrm{rank}(L) \leq r$. By Lemma 13, we know that some $Q \in \mathcal{Q}^r_M$ is a basis for col-span$(L)$. Moreover, we know that the columns of $S$ form an independent set in $\mathrm{Tr}(G^Q_M, Y^Q_M, \langle C_M, \tau\rangle)$ (by Observation 21). Hence, it is sufficient to show that the set of columns of $S$ form an independent set of the matroid $\mathcal{M}(K_n, Y^Q_M)$. But this follows from the fact that $\mathrm{inc}^{-1}(S)$ is a forest and $\mathcal{M}(K_n, Y^Q_M)$ is isomorphic to $M(K_n)$ under $\phi$ (see Definition 17 and Observation 18).

In the converse direction, suppose that for some $Q \in \mathcal{Q}^r_M$, there is a common independent set $I$ of size $m$ in the transversal matroid $\mathrm{Tr}(G^Q_M, Y^Q_M, \langle C_M, \tau\rangle)$ and the matroid $\mathcal{M}(K_n, Y^Q_M)$. We construct $S$ and $L$ as follows. Pick a matching $C$ in $\widetilde{G}^Q_M$ (see Definition 16) saturating $I$ and $X^Q_M$. Since $|I| = m$, such a matching exists. For each $x \in X^Q_M = [m]$, we denote by $x_C$ the partner of $x$ in $C$. Notice that by the definition of $\widetilde{G}^Q_M$, every vertex $x \in C_M$ has a unique neighbor, which must be $\tau(x)$.

We now define $S$ and $L$ as follows. For each $x \in [m]$, set $S[\{x\}] = x_C$ and $L[\{x\}] = M[\{x\}] + x_C$. Since $S$ contains only distinct columns and these are all contained in $\mathrm{cols}(\mathrm{inc}(K_n))$, it follows that $S$ is the incidence matrix of a simple graph. In addition, for every $x \in C_M$, $S[\{x\}] = \tau(x)$ by the definition of $S$. Moreover, from Observation 18 and the fact that the set $I = \{x_C | x \in [m]\}$ is an independent set in $\mathcal{M}(K_n, Y^Q_M)$, we have that $\mathrm{inc}^{-1}(S)$ is isomorphic to the subgraph of $K_n$ induced by the edge set $\phi(I)$, which is a forest.

Finally, by the definition of $G^Q_M$, we have that for every $x \in [m]$, $M[\{x\}] + x_C \in \mathrm{span}(Q)$. Consequently we have that col-span$(L) \subseteq \mathrm{span}(Q)$ and since $|Q| \leq r$, and the lemma follows. ◀

▶ **Lemma 23.** Extended (min-rank, acyclic)-Graph Recovery *can be solved in time $(mn)^{O(r)}$.*

### 3.2.3 Recovering Graphs of Fixed Arboricity

In the Arboricity-$d$ Graph Recovery problem, the input is the pair $(M, r)$ and the goal is to decide whether $M = S + L$, where $H = \mathrm{inc}^{-1}(S)$ has arboricity at most $d$ and $\mathrm{rank}(L) \leq r$. Observe that Extended Arboricity-$d$ Graph Recovery is a generalization of Extended Acyclic Graph Recovery (set $d = 1$).

▶ **Definition 24.** *For every $d \in \mathbb{N}$, we define the gammoid $\mathrm{Gam}^d(G^Q_M, Y^Q_M, \langle C_M, \tau\rangle)$ as follows. Consider the digraph $D_1$ obtained from $\widetilde{G}^Q_M$ by orienting all edges from $X^Q_M$ to $Y^Q_M$. For each $y \in Y^Q_M$, construct a directed path $D^y_2 = (y, 1), \ldots, (y, d)$. We define the digraph $D_3$ as the graph obtained by identifying each $y \in Y^Q_M$ in $D_1$ with $(y, 1)$ in $D^y_2$. The vertex set of $D_3$ is now $X^Q_M \cup (Y^Q_M \times [d])$. Then, $\mathrm{Gam}^d(G^Q_M, Y^Q_M, \langle C_M, \tau\rangle)$ is defined as the gammoid with respect to $D_3$ and $X^Q_M$ on ground set $Y^Q_M \times [d]$.*

▶ **Definition 25.** *For each $i \in [d]$, define the matroid $\mathcal{J}^i(K_n, Y^Q_M)$ as the matroid over element set $Y^Q_M \times \{i\}$ which is isomorphic to $\mathcal{M}(K_n, Y^Q_M)$ under the bijection $\psi_i$ where $\psi_i(y, i) = y$, for each $y \in Y^Q_M$. We denote by $\mathcal{M}^d(K_n, Y^Q_M)$ the direct-sum of the $d$ matroids $\{\mathcal{J}^i(K_n, Y^Q_M) \mid i \in [d]\}$. We also extend each $\psi_i$ to sets in the following way: for every $Z_1 \subseteq Y^Q_M$, and $Z_2 = Z_1 \times \{i\}$, $\psi_i(Z_2) = Z_1$.*

The following lemma generalizes Lemma 22.

▶ **Lemma 26.** *$(M, r, C_M, \tau)$ is a yes-instance of* Extended Arboricity-$d$ *Graph Recovery if and only if there is a $Q \in \mathcal{Q}^r_M$ such that there is a common independent set of size $m$ in the gammoid $\mathrm{Gam}^d(G^Q_M, Y^Q_M, \langle C_M, \tau\rangle)$ and the matroid $\mathcal{M}^d(K_n, Y^Q_M)$.*

▶ **Lemma 27.** *For every fixed constant $d$,* Extended (min-rank, $d$-arboricity)-*Graph Recovery can be solved in time $(mn)^{O(r)}$ on input $(M, r, C_M, \tau)$.*

**Proof.** (Sketch) The crux of this lemma is a proof that $(M, r, C_M, \tau)$ is a yes-instance of Extended (min-rank, $d$-arboricity)-Graph Recovery if and only if there is a $Q \in \mathcal{Q}^r_M$ such that there is a common independent set of size $m$ in an appropriate defined gammoid and the matroid $\mathcal{M}^d(K_n, Y^Q_M)$. ◀

## 3.2.4 Recovering connected graphs

Here, we prove an analogous version of Lemma 23 for Extended (min-rank, connected)-Graph Recovery.

▶ **Lemma 28.** *Extended* (min-rank, connected)-*Graph Recovery can be solved in time $(mn)^{O(r)}$.*

**Proof.** (Sketch) The crux of this lemma is a proof that $(M, r, C_M, \tau)$ is a yes-instance of Extended (min-rank, connected)-Graph Recovery if and only if there is a $Q \in \mathcal{Q}^r_M$ such that there is a common independent set of size $m$ in the transversal matroid $\mathrm{Tr}(G^Q_M, Y^Q_M, \langle C_M, \tau\rangle)$ and the $m$-elongation of the matroid $\mathcal{M}(K_n, Y^Q_M)$ (assuming that $m \geq n-1$, otherwise we have a no-instance). The details of this argument follow along the same lines as the proof of Lemma 22. The crucial part of this lemma is the choice of the $m$-elongation of $\mathcal{M}(K_n, Y^Q_M)$ as the matroid that we intersect with $\mathrm{Tr}(G^Q_M, Y^Q_M, \langle C_M, \tau\rangle)$. ◀

Lemma 9 is now a straightforward consequence of Lemma 20, Lemma 23, Lemma 28, and Lemma 27.

## 3.3 Decision algorithms for Extended (min-weight, $\mathcal{C}$)-Graph Recovery

The goal of this subsection is to prove Lemma 10. The proof is involved and has several stages. In the following, we provide a high level overview of our proof strategy.

Recall that an instance of (min-weight, $\mathcal{C}$)-Graph Recovery is of the form $(M, k, C_M, \tau)$ where $C_M \subseteq \mathrm{cols}(M)$ and $\tau$ fixes the mapping of the columns in $C_M$ in the solution. When we just want $G$ to be a simple graph (i.e., $\mathcal{C}$ is the class of all simple graphs), a solution $M = L + S$ minimizing the cost, $\|L\|$, can be computed in polynomial time using an algorithm for Minimum Weight Bipartite Matching, in an auxiliary bipartite graph which represents the costs of mapping each column of $M$ to a specific edge. The design of this bipartite graph is similar in spirit to that of the graph in Definition 14. We refer to a solution $M = L + S$ minimizing the cost $\|L\|$, as a *minimum cost solution.*

To obtain connected graphs and forests, we build upon this algorithm. We first describe an algorithm that computes a minimum cost solution $M = L_0 + S_0$ along with the graph $G_0$ and try to reassign columns of $M$ associated with non-bridge edges in $G_0$ to reduce the number of connected components without increasing the cost of the solution at hand. That is, we move from one solution to another without increasing the cost but while decreasing the number of connected components in the graph corresponding to the incidence matrix in the solution. Note that, we may need to perform a number of reassignments in a sequence before the number of connected components is reduced. To determine this sequence of reassignments, we associate them with paths in an auxiliary digraph, where the nodes are the edges and non-edges of the current graph, and directed edges indicate feasible reassignments, or reassignments that convert a bridge in the current graph to a non-bridge. We show that the current solution minimizes the number of connected components if and only if the auxiliary digraph has no paths starting from a non-bridge edge and ending at a non-edge. This requires an intricate analysis of the structure of a solution and how reassignments affect them. This leads to an iterative algorithm that "remaps" the edges in the graph associated with the current solution using paths in the auxiliary digraph. When this algorithm terminates, we obtain a minimum cost solution that minimizes the number of connected components in the corresponding graph. We then use this algorithm as the starting point for the algorithms to recover connected graphs and forests. To recover a connected graph, we observe that any further reassignments to reduce the number of connected components must increase the cost of the solution. Therefore, we design an iterative algorithm that reassigns non-bridge edges until every edge is a bridge, or the graph becomes connected. This algorithm can be used to recover a connected graph, or a forest satisfying the required properties.

### 3.4 NP-completeness of (min-rank, $\mathcal{C}$)-Graph Recovery

▶ **Theorem 29.** (min-rank, $\mathcal{C}$)-*Graph Recovery is* NP-*complete for* $\mathcal{C} \in \{$*simple graphs, acyclic graphs, connected graphs, arboricity-d graphs*$\}$.

## 4 Concluding remarks and future work

In this paper, we have initiated the study of a family of graph recovery problems. In these problems, the aim is to recover the incidence matrix of a graph (contained in a specific graph class) from a given binary matrix. The input matrix is assumed to be a perturbation of an incidence matrix by either a small rank or low weight matrix. We have demonstrated the rich combinatorial structure possessed by these problems by designing decision and enumeration algorithms using classic concepts such as matchings and matroids. We leave open the following concrete questions.

1. Is (min-rank, $\mathcal{C}$)-Graph Recovery fixed-parameter tractable parameterized by $r$ for the classes we consider? That is, can it be solved in time $f(r)(mn)^{\mathcal{O}(1)}$ for some computable function $f$?
2. Identify classes $\mathcal{C}$ for which (min-rank, $\mathcal{C}$)-Graph Recovery is polynomial-time solvable.
3. Characterize those $\mathcal{C}$ for which (min-weight, $\mathcal{C}$)-Graph Recovery is polynomial-time solvable.

### References

1 Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, 2011. doi:10.1145/1970392.1970395.

2 Venkat Chandrasekaran, Sujay Sanghavi, Pablo A. Parrilo, and Alan S. Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011. doi:10.1137/090761793.

3 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 3rd edition, 2005.

4 Fedor V Fomin, Petr A Golovach, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Covering vectors by spaces in perturbed graphic matroids and their duals. *arXiv preprint*, 2019. arXiv:1902.06957.

5 Jim Geelen, Bert Gerards, and Geoff Whittle. On rota's conjecture and excluded minors containing large projective geometries. *Journal of Combinatorial Theory, Series B*, 96(3):405–425, 2006.

6 Jim Geelen and Rohan Kapadia. Computing girth and cogirth in perturbed graphic matroids. *Combinatorica*, 38(1):167–191, 2018.

7 Chien-Chung Huang, Naonori Kakimura, and Naoyuki Kamiyama. Exact and approximation algorithms for weighted matroid intersection. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 430–444, 2016. doi:10.1137/1.9781611974331.ch32.

8 Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. Deterministic truncation of linear matroids. *ACM Trans. Algorithms*, 14(2):14:1–14:20, 2018. doi:10.1145/3170444.

9 James G. Oxley. *Matroid theory*, volume 21 of *Oxford Graduate Texts in Mathematics*. Oxford University Press, 2nd edition, 2010.

10 Nauman Shahid, Nathanael Perraudin, Vassilis Kalofolias, Gilles Puy, and Pierre Vandergheynst. Fast robust pca on graphs. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):740–756, 2016.

11 John Wright, Arvind Ganesh, Shankar R. Rao, YiGang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Proceedings of 23rd Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2080–2088. Curran Associates, Inc., 2009. URL: http://papers.nips.cc/paper/3704-robust-principal-component-analysis-exact-recovery-of-corrupted-low-rank-matrices-via-convex-optimization.

12 Mengnan Zhao, M Devrim Kaba, René Vidal, Daniel P Robinson, and Enrique Mallada. Sparse recovery over graph incidence matrices. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 364–371. IEEE, 2018.