Parameterized Complexity Classification of Deletion to List Matrix-Partition for Low-Order Matrices

Akanksha Agrawal
Ben-Gurion University of the Negev, Beer-Sheva, Israel
agrawal@post.bgu.ac.il

Sudeshna Kolay
Ben-Gurion University of the Negev, Beer-Sheva, Israel
sudeshna@post.bgu.ac.il

Jayakrishnan Madathil
The Institute of Mathematical Sciences, HBNI, Chennai, India
jayakrishnanm@imsc.res.in

Saket Saurabh
University of Bergen, Bergen, Norway
The Institute of Mathematical Sciences, HBNI, Chennai, India
saket@imsc.res.in

Abstract

Given a symmetric $\ell \times \ell$ matrix $M = (m_{i,j})$ with entries in $\{0, 1, *\}$, a graph G and a function $L : V(G) \to 2^{[\ell]}$ (where $[\ell] = \{1, 2, \ldots, \ell\}$), a list M-partition of G with respect to L is a partition of $V(G)$ into ℓ parts, say, $V_1, V_2, \ldots, V_\ell$ such that for each $i, j \in \{1, 2, \ldots, \ell\}$, (i) if $m_{i,j} = 0$ then for any $u \in V_i$ and $v \in V_j$, $uv \notin E(G)$, (ii) if $m_{i,j} = 1$ then for any (distinct) $u \in V_i$ and $v \in V_j$, $uv \in E(G)$, (iii) for each $v \in V(G)$, if $v \in V_i$ then $i \in L(v)$. We consider the Deletion to List M-Partition problem that takes as input a graph G, a list function $L : V(G) \to 2^{[\ell]}$ and a positive integer k. The aim is to determine whether there is a k-sized set $S \subseteq V(G)$ such that $G - S$ has a list M-partition. Many important problems like Vertex Cover, Odd Cycle Transversal, Split Vertex Deletion, Multiway Cut and Deletion to List Homomorphism are special cases of the Deletion to List M-Partition problem. In this paper, we provide a classification of the parameterized complexity of Deletion to List M-Partition, parameterized by k, (a) when M is of order at most 3, and (b) when M is of order 4 with all diagonal entries belonging to $\{0, 1\}$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Fixed parameter tractability

Keywords and phrases list matrix partitions, parameterized classification, Almost 2-SAT, important separators, iterative compression

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2019.41

Funding *Akanksha Agrawal*: Supported by the PBC Program of Fellowships for Outstanding Post-doctoral Researchers from China and India (No. 5101479000).
Saket Saurabh: Supported by the Horizon 2020 Framework program, ERC Consolidator Grant LOPPRE (No. 819416).

1 Introduction

A large number of problems in algorithmic graph theory are of the following two types. (1) Given a graph G, can the vertices of G be partitioned subject to a set of constraints? And (2) given a graph G and a non-negative integer k, is it possible to delete at most k vertices from G so that the vertices of the resulting graph can be partitioned subject to a set of

constraints? In this paper, we study the parameterized complexity of a family of problems of the second type, (with k, the size of the deletion set, as the parameter). We consider those partitions that can be characterised by a matrix of order at most 4. In this regard, we follow Feder et al. [17], who undertook a similar study of partition problems of the first type.

Let M be a symmetric $\ell \times \ell$ matrix with entries from $\{0, 1, *\}$. For a graph G, an *M-partition* $\mathcal{V}$ of G is a partition of $V(G)$ into ℓ parts $\{V_1, V_2, \ldots, V_\ell\}$ (where some part could be empty) such that for every $i \in [\ell]$, (i) V_i is an independent set if $m_{i,i} = 0$, (ii) $G[V_i]$ is a clique if $m_{i,i} = 1$ (and no restriction on V_i if $m_{i,i} = *$); and for distinct indices $i, j \in [\ell]$, (iii) V_i and V_j are completely adjacent if $m_{i,j} = 1$, (iv) V_i and V_j are completely non-adjacent if $m_{i,j} = 0$ (and no restriction on the edges between V_i and V_j if $m_{i,j} = *$). The M-Partition problem takes as input a graph G, and the objective is to determine if G admits an M-partition. This problem encompasses recognition of many graph classes that can be characterised by a partition of the vertex set satisfying certain constraints. For instance, consider the following matrices:

$$M_1 = \begin{pmatrix} 0 & * \\ * & 0 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 0 & * \\ * & 1 \end{pmatrix} \qquad M_\ell = \begin{pmatrix} 0 & * & \cdots & * \\ * & 0 & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & 0 \end{pmatrix}_{\ell \times \ell}$$

The set of graphs that admit an M_1-partition and M_2-partition are exactly the family of bipartite graphs and split graphs, respectively. Note that both bipartite graphs and split graphs have polynomial time recognition algorithms [3, 22, 25]. The graphs that admit an M_ℓ-partition are exactly the graphs that admit a proper colouring using at most ℓ colours. It is well-known that while 2-colouring is polynomial time solvable [3], ℓ-colouring is NP-hard for every $\ell \geq 3$ [19, 20].

For an $\ell \times \ell$ matrix M, List M-Partition is a generalization of M-Partition. Let M be a symmetric $\ell \times \ell$ matrix over $\{0, 1, *\}$. Given a graph G and a function $L : V(G) \to 2^{[\ell]}$ (L is called a *list function*, and for each $v \in V(G)$, $L(v)$ is called the *list* of v), a *list M-partition of G with respect to L* (or a list M-partition of G that respects L) is an M-partition $\mathcal{V} = \{V_1, V_2, \ldots, V_\ell\}$ of G such that for each $v \in V(G)$, if $v \in V_i$ for some $i \in [\ell]$ then $i \in L(v)$. List M-Partition takes as input a graph G, a list function $L : V(G) \to 2^{[\ell]}$, and the objective is to determine if G admits a list M-partition of G that respects L. Note that when an instance of List M-Partition has the input list function mapping every vertex to the set $[\ell]$, then it is also an instance of M-Partition (where we forget the list function).

Both M-Partition and List M-Partition problems have been extensively studied in the literature, both for restricted matrices and for restricted graph classes (see, for example, [1, 5, 8, 10, 11, 12, 16, 17, 24, 29] and references therein). The most widely known special case of List M-Partition is perhaps the List Colouring problem. The notion of studying the restriction of colouring problems with a list of allowed colours on vertices was introduced and studied (independently) by Vizing [28] and Erdős et al. [13]. Since its advent, the problem has been extensively studied in both graph theory and algorithms [13, 24, 29]. Another important special case of List M-Partition is the List Homomorphism problem, which to the best of our knowledge, was introduced by Feder and Hell [14], and has been extensively studied in the literature [1, 8, 10, 11, 12, 16].

Feder et al. [17] studied List M-Partition, and established a complete classification of List M-Partition for small matrices M (into P/NP-hard/quasi polynomial time). Their results form a special case of later results due to Feder and Hell [15, Corollary 3.4] on

constraint satisfaction problems. In this paper, we look at the deletion version of List M-Partition, which we call Deletion to List M-Partition. The problem is formally defined as follows.

Deletion to List M-Partition	**Parameter:** k
Input: A graph G, a list function $L : V(G) \to 2^{[\ell]}$ where ℓ is the order of M, and a non-negative integer k.	
Question: Does there exist $X \subseteq V(G)$ such that $\lvert X\rvert \leq k$ and $G - X$ admits a list M-partition that respects L?	

The Deletion to List M-Partition problem generalises many well-studied classical problems, such as Vertex Cover (VC), Odd Cycle Transversal (OCT), Split Vertex Deletion (SVD) and Multiway Cut, to name a few. These problems (VC, OCT, SVD etc.) have been studied in both classical and parameterized complexity settings and are all NP-hard [30, 9]. Chitnis et al. [7] initiated the study of the deletion version of List Homomorphism to a graph H, called Dl-Hom(H), which is a special case of Deletion to List M-Partition. (In [7], H is considered to be a loopless, simple graph.) They showed that Dl-Hom(H) is FPT (parameterized by k and $|H|$) for any (P_6, C_6)-free bipartite graph H, and conjectured that the problem is FPT for those graphs H for which List Homomorphism (i.e., without deletions) is polynomial-time solvable. Notice that for some of the matrices that do not contain both 1 and 0 and do not have a * as a diagonal entry, the corresponding Deletion to List M-Partition problem is covered by the results in [7].[1] While we study the deletion version of List M-Partition, the "counting version," denoted by #List M-Partition, where given G and L as input, the goal is to determine the number of M-partitions of G that respect L, has been studied by Göbel et al. [21]. They established a complete dichotomy by showing that for any symmetric matrix M over $\{0, 1, *\}$, #List M-Partition is either in FP or #P-complete.

Our Results and Methods

We study the parameterized complexity of Deletion to List M-Partition for different matrices M, and obtain a classification of these problems (into polynomial time solvable, NP-hard and FPT or para-NP-hard) when M is a matrix of order at most 4.

***M* is of order at most 3.** First, we resolve the classical complexity of Deletion to List M-Partition problems when M is of order at most 3, except for one matrix. We extend the study to explore the parameterized complexity of these deletion problems, parameterized by the size k of the deletion set. Specifically, we prove the following theorem.

▶ **Theorem 1** ($\star$[2])**.** *For a* 3×3 *symmetric matrix* M *over* $\{0, 1, *\}$*, the* Deletion to List M-Partition *problem is*

1. *polynomial time solvable if either* M *or* $\overline{M}$ *is equivalent to one of the three matrices*

$$\begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix}, \quad \begin{pmatrix} * & 0 & * \\ 0 & * & * \\ * & * & * \end{pmatrix} \text{ or } \begin{pmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & * \end{pmatrix};$$

[1] The results in [7] only cover a few matrices of the described form because there is an additional constraint of H being a (P_6, C_6)-free bipartite graph.

[2] Due to paucity of space, (full) proofs of statements marked with a $\star$ have been omitted.

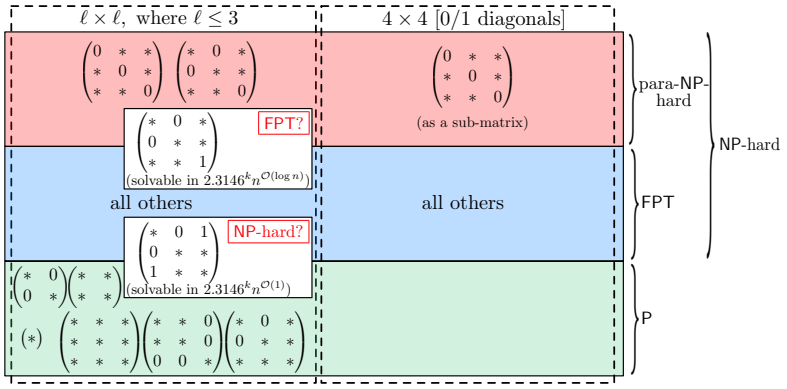

Figure 1 An overview of the results on matrices, up to complementation and equivalence.

2. *para-NP-hard if either M or $\overline{M}$ is equivalent to one of the two matrices*

$$\begin{pmatrix} 0 & * & * \\ * & 0 & * \\ * & * & 0 \end{pmatrix} \text{ or } \begin{pmatrix} * & 0 & * \\ 0 & * & * \\ * & * & 0 \end{pmatrix};$$

3. *solvable by an algorithm running in time $2.3146^k n^{\mathcal{O}(\log n)}$, when M or $\overline{M}$ is equivalent to*

$$\begin{pmatrix} * & 0 & * \\ 0 & * & * \\ * & * & 1 \end{pmatrix};$$

4. *FPT if M is not covered by any of the previous cases.*

Theorem 1 shows that the Deletion to List M-Partition problem has polynomial time algorithms, FPT algorithms or algorithms of the form $c^k n^{\mathcal{O}(\log n)}$, depending on the matrix M (see also Figure 1). We also have two cases for M where the Deletion to List M-Partition problem is para-NP-hard. With such a varied range of complexities, we use several techniques to design the algorithms. The polynomial time algorithms are based on ideas that help to reduce the problem to an equivalent minimum separator problem [18]. For our FPT algorithms, we utilize the notion of important separators, which was introduced by Marx [27]. Another technique we use to design our FPT algorithms for Deletion to List M-Partition is based on reducing the given instance to (polynomially many) instances of Variable Deletion Almost 2-SAT, and then employing the known FPT algorithm for Variable Deletion Almost 2-SAT to resolve the instance. We also use the technique of "iterative compression" for designing FPT algorithm for one of our cases. For the matrix M defined in item 3 of Theorem 1, although the FPT solvability versus W-hardness of Deletion to List M-Partition remains open, we design an algorithm for this problem that runs in time $2.3146^k n^{\mathcal{O}(\log n)}$, where n is the number of vertices in the input graph. We use the technique of separating families introduced by Feder et al. [17] to design this algorithm. These results can be found in Section 3.

***M* is of order 4.** We restrict the matrices M to have only 0s and 1s as their diagonal entries. First, we observe that all these problems are NP-hard. Second, we design FPT algorithms for these problems, unless the matrix M "encompasses" the 3-Colouring problem. Let M_{col} denote the 3×3 matrix with only 0s on the diagonal, and $*$s elsewhere. We prove the following theorem.

▶ **Theorem 2** (⋆). *Consider a* Deletion to List M-Partition *problem, where M is a 4×4 matrix over $\{0, 1, *\}$ that has only 0s and 1s as diagonal entries. If M does not contain an equivalent matrix of M_{col} (or its complement) as a sub-matrix, then the problem is FPT. Otherwise, the problem is para-NP-hard.*

Our FPT algorithms use the technique of iterative compression along with the known FPT algorithm for Variable Deletion Almost 2-SAT. Our use of iterative compression exploits structural properties provided by M in order to design the FPT algorithms. Our results, in particular show that, for a 3×3 matrix (except the matrix defined in item 3 of Theorem 1) or a 4×4 matrix with no $*$s on the diagonal, whenever the List M-Partition is polynomial time solvable, the corresponding Deletion to List M-Partition problem is fixed-parameter tractable, and whenever List M-Partition is NP-hard, Deletion to List M-Partition is para-NP-hard, and thus provide a (partial) parameterized analogue of the results established in [17].

2 Preliminaries and Basic Tools

For a graph G, $V(G)$ and $E(G)$ denote respectively the vertex set and edge set of G. Given a partition $\mathcal{V}$ of $V' \subseteq V(G)$, $V(\mathcal{V}) = V'$, and for $X \subseteq V'$, $\mathcal{V} - X$ denotes the restriction of the partition to $V' \setminus X$. Let G be a graph and $X, Y \subseteq V(G)$. A set of vertices $S \subseteq V(G)$ is said to be an (X, Y)-*separator* if $G - S$ contains no path from X to Y.

Definitions and results for some useful problems. Consider a 2-CNF formula ψ. The variable set of ψ is denoted by $\mathsf{Var}(\psi)$. For a set $Y \subseteq \mathsf{Var}(\psi)$, $\psi - Y$ denotes the 2-CNF formula obtained from ψ by deleting all the clauses that contain a variable from Y. The 2-SAT problem takes as input a 2-CNF formula ψ, and the question is to test if there is a satisfying assignment for ψ. The 2-SAT problem admits a polynomial time algorithm [23]. The Variable Deletion Almost 2-SAT problem takes as input a 2-CNF formula ψ and a non-negative integer k, and the objective is to test if there is a set $X \subseteq \mathsf{Var}(\psi)$ of size at most k such that $\psi - X$ is satisfiable. It is known that Variable Deletion Almost 2-SAT admits an algorithm that runs in time $2.3146^k n^{\mathcal{O}(1)}$, where n is the number of variables [26], and hence is in FPT, when parameterized by k.

Matrices and list partitioning. For an $\ell \times \ell$ matrix M, consider an M-partition $\mathcal{V} = \{V_1, V_2, \ldots, V_\ell\}$ of a graph G. Then the part V_i will be said to have index i. For an instance $(G, L : V(G) \to 2^{[\ell]})$ of List M-Partition, throughout the paper we assume that $L(v) \neq \emptyset$, as otherwise, we can immediately report that (G, L) does not admit an M-partition that respects L. Similarly, for an instance $(G, L : V(G) \to 2^{[\ell]}, k)$ of Deletion to List M-Partition, we assume that $L(v) \neq \emptyset$, as otherwise, we can (safely) delete such a vertex from G and reduce k by 1 (or return that it is a no-instance when $k \leq 0$).

Given a matrix M, the complement $\overline{M}$ is defined as follows: $m_{i,j} = 0 \iff \overline{m}_{i,j} = 1, m_{i,j} = 1 \iff \overline{m}_{i,j} = 0, m_{i,j} = * \iff \overline{m}_{i,j} = *$. The lower triangular submatrix $M_L = (m^L_{i,j})$ of a matrix $M = (m_{i,j})$ is defined as follows: $\forall i \geq j, m^L_{i,j} = m_{i,j}$ and $\forall i < j, m^L_{i,j} = 0$. When the context is clear we drop the superscript from the entry names of the lower triangular matrix M_L and simply use the entry names $m_{i,j}$ of M. Similarly, the upper triangular submatrix $M_U = (m^U_{i,j})$ of a matrix $M = (m_{i,j})$ is defined as follows: $\forall i \leq j, m^U_{i,j} = m_{i,j}$ and $\forall i > j, m^U_{i,j} = 0$. Again, we drop superscripts when the context is clear. The following observation follows from the definition of the complement of a matrix.

▶ **Observation 3.** *A graph G admits a list M-partition with respect to a list function L if and only if $\overline{G}$ admits a list $\overline{M}$-partition with respect to L.*

In this paper, for an $\ell \times \ell$ matrix M, a submatrix M' of order $p \leq \ell$ is defined as follows: there are p distinct indices $\{i_1, i_2, \ldots, i_p\} \in [\ell]$ such that $m'_{a,b} = m_{i_a,i_b}$ for all $a, b \in [p]$. Consider two symmetric $\ell \times \ell$ matrices $M = (m_{i,j})$ and $M' = (m'_{i,j})$ over $\{0, 1, *\}$. We say that M is *equivalent* to M', if there is a permutation $\pi : [\ell] \to [\ell]$ such that $m'_{i,j} = m_{\pi(i),\pi(j)}$. And such a permutation π is called a *witness-permutation* for (M, M'). Notice that if M is equivalent to M', then M' is also equivalent to M with witness-permutation π^{-1}. That is, M and M' are equivalent if they define the same partition up to a re-indexing of the parts. We immediately obtain the following result.

▶ **Proposition 4.** *Let M be a symmetric matrix equivalent to M', with a witness-permutation π. Then, a graph G admits a list M-partition with respect to a list function L if and only if G admits a list M'-partition with respect to the list function L', where $L'(v) = \{\pi(i) \mid i \in L(v)\}$ for every $v \in V(G)$.*

Some Useful Results on List M-Partition

We present a summary of results from [17], which will be used throughout.

Reducing List M-partition to 2-SAT, for a 2×2 matrix M. It was shown in [17] that an instance of the List M-partition problem can be reduced (in polynomial time) to an equivalent instance of 2-SAT, provided that M is a 2×2 matrix. And since 2-SAT is polynomial time solvable [2], so is List M-partition, when M is a 2×2 matrix. What is interesting is that this reduction works even if M is not of order 2, but the size of $L(v)$ is at most 2 for every vertex v of the input graph G. The List M-Partition problem such that the list of every vertex in G has size at most two will also be referred to as the 2-List M-Partition problem. The reduction from 2-List M-Partition to 2-SAT will also be useful while designing algorithms for Deletion to List M-Partition. Next, we state the properties of the reduction from 2-List M-Partition to 2-SAT.

▶ **Proposition 5** ($\star$)**.** *Let $(G, L : V(G) \to 2^{[\ell]})$ be an instance of* 2-List M-Partition*. In polynomial time we can output an instance ψ of* 2-SAT *and a bijective function $f : V(G) \to \mathsf{Var}(\psi)$, such that for every $X \subseteq V(G)$: i) $\psi - f(X)$ is a yes instance of* 2-SAT *if and only if $(G - X, L|_{V(G-X)} : V(G - X) \to 2^{[\ell]})$ is a yes instance of* 2-List M-Partition*, and ii) a set $A \subseteq \mathsf{Var}(\psi - f(X))$ is a satisfying assignment for $\psi - f(X)$ if and only if for each $v \in \{f^{-1}(a) \mid a \in A\}$ with $L|_{V(G-X)}(v) = \{i, j\}$, where $i \leq j$, we have $v \in V_i$. Here, $\mathcal{V} = \{V_1, V_2, \ldots, V_\ell\}$ is an M-partition of $G - X$ (if it exists).*

From the above proposition, we can conclude that 2-List M-Partition admits a polynomial time algorithm. It also suggests a strategy for solving List M-partition: try to reduce the size of every list. It is indeed possible to do that, as shown in [17], if the matrix M has a row that contains both a 0 and a 1 (see Proposition 2.3 and Corollary 2.4 in [17]).

▶ **Proposition 6** ([17])**.** *Suppose the matrix M has $m_{i_1,i_2} = 0$ and $m_{i_1,i_3} = 1$ (one of i_3 or i_2 can be equal to i_1). Then an instance (G, L) of the* List M-Partition *problem can be reduced (in polynomial time) to (i) one instance with no list containing i_1 and (ii) at most $|V(G)|$ instances with no list containing both i_2 and i_3, such that (G, L) is a yes instance if and only if at least one of the $|V(G)| + 1$ instances is a yes instance.*

The above proposition comes handy when M is a 3×3 matrix, as in this case the problem reduces to $|V(G)| + 1$ instances that have only lists of size at most two. Another notion that will be useful in our algorithm is "domination", defined below.

▶ **Definition 7.** *For a symmetric matrix M of order ℓ over $\{0, 1, *\}$, and rows $i_1, i_2 \in [\ell]$, i_1 dominates i_2 in M if for each column $i_3 \in [\ell]$, either $m_{i_1 i_3} = m_{i_2 i_3}$ or $m_{i_1 i_3} = *$.*

▶ **Proposition 8** (Proposition 2.5 [17])**.** *Consider a matrix M of order ℓ, where the row i_1 dominates the row i_2, and an instance (G, L) of* List M-Partition*. Let L' be the list function such that for each $v \in V(G)$, (i) $L'(v) = L(v)$ if $|L(v) \cap \{i_1, i_2\}| \leq 1$, and (ii) $L'(v) = L(v) \setminus \{i_2\}$ otherwise. The graph G admits a list M-partition that respects L if and only if it admits a list M-partition that respects L'.*

Basic tools for Deletion to List M-Partition. We show how the results presented earlier for List M-Partition can be used for solving Deletion to List M-Partition. Consider an instance (G, L, k) of Deletion to List M-Partition. Suppose $|L(v)| \leq 2$ for every $v \in V(G)$. Let ψ be the 2-CNF formula and $f : V(G) \to \mathsf{Var}(\psi)$ be the bijective function returned by Proposition 5. The properties of ψ and f (as in Proposition 5), ensures that deleting a set $X \subseteq V(G)$ of vertices from G is equivalent to deleting the variables $f(X)$ from ψ. Let Deletion to 2-List M-Partition be the special case of Deletion to List M-Partition where the list of each vertex has size at most two. By Proposition 5, Deletion to 2-List M-Partition is equivalent to testing whether k variables can be deleted from ψ to make it satisfiable. This is exactly the same as the Variable Deletion Almost 2-SAT problem, which admits an FPT algorithm running in time $2.3146^{k'} n'^{\mathcal{O}(1)}$, where k' is the size of the deletion set and n' is the number of variables. This together with Proposition 5 gives us the following result.

▶ **Proposition 9.** Deletion to 2-List M-Partition *is fixed-parameter tractable with running time $2.3146^k n^{\mathcal{O}(1)}$.*

Now using Propositions 6 and 9, we obtain the following result.

▶ **Proposition 10.** *Let M be a 3×3 matrix such that M has a row that contains both a 0 and a 1. Then* Deletion to List M-Partition *is fixed-parameter tractable.*

▶ **Proposition 11** ($\star$)**.** Deletion to List M-Partition *is NP-hard if at least one of the diagonal entries of M is 0 or 1.*

The reduction rule stated in the following lemma will be useful in our algorithms.

▶ **Lemma 12** ($\star$)**.** *For a matrix M, let (G, L, k) be an instance of* Deletion to List M-Partition*, with a vertex $v \in V(G)$, such that $L(v) = \emptyset$. Then, (G, L, k) and $(G - \{v\}, L|_{V(G)\setminus\{v\}}, k - 1)$ are equivalent instances of* Deletion to List M-Partition*.*

▶ Remark 13. We note that for any Deletion to List M-Partition problem, we assume that we are looking for a list M-partition where each part is non-empty. First, if there is a part that is empty in the list M-partition, then our current instance can be resolved as an instance of Deletion to List M'-Partition, where M' is a matrix of order strictly less than M. Otherwise, for no list M-partition is any part empty. In such a case, in a polynomial time preprocessing step, we can guess one vertex v_i per part i of the hypothetical list M-partition $\mathcal{V}$ and appropriately reduce $L(v_i) = \{i\}$.

Table 1 An overview of our NP-hardness vs. P results (not covered by Proposition 11) for matrices of order 3×3, where complement matrices are not shown.

Matrices (and witness-permutation $\pi : [3] \to [3]$) $\pi(1) = 1, \pi(2) = 3, \pi(3) = 2$	$\pi(1) = 3, \pi(2) = 2, \pi(3) = 1$	Equivalent matrix	Class	Proof
		$\begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix}$	P	Lemma 16(a)
$\begin{pmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{pmatrix}$	$\begin{pmatrix} * & * & * \\ * & * & 0 \\ * & 0 & * \end{pmatrix}$	$\begin{pmatrix} * & 0 & * \\ 0 & * & * \\ * & * & * \end{pmatrix}$	P	Lemma 16(b)
$\begin{pmatrix} * & 0 & * \\ 0 & * & 0 \\ * & 0 & * \end{pmatrix}$	$\begin{pmatrix} * & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$	$\begin{pmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & * \end{pmatrix}$	P	Lemma 16(c)
		$\begin{pmatrix} * & 0 & 0 \\ 0 & * & 0 \\ 0 & 0 & * \end{pmatrix}$	NP-hard	Lemma 14(a)
$\begin{pmatrix} * & 0 & 1 \\ 0 & * & 0 \\ 1 & 0 & * \end{pmatrix}$	$\begin{pmatrix} * & 0 & 0 \\ 0 & * & 1 \\ 0 & 1 & * \end{pmatrix}$	$\begin{pmatrix} * & 1 & 0 \\ 1 & * & 0 \\ 0 & 0 & * \end{pmatrix}$	NP-hard	Lemma 14(b)

3 Classification of 3×3 matrices

In this section, our main objective is to classify Deletion to List M-Partition for matrices M of order 3, both in classical complexity as well as parameterized complexity. Throughout the section, $M = (m_{i,j})$ denotes a symmetric 3×3 matrix over $\{0, 1, *\}$. We prove some of the main algorithmic results that we obtain for Deletion to List M-Partition for matrices M of order 3, and thus present a partial proof of Theorem 1.

Before we go into the proof of Theorem 1, we first address the question of NP-hardness versus polynomial time solvability of these problems. Already we saw in Proposition 11 that Deletion to List M-Partition is NP-hard if at least one of the diagonal entries of M is 0 or 1. So we now need to consider only those matrices M for which $m_{1,1} = m_{2,2} = m_{3,3} = *$. And up to complementation and equivalence of matrices, we are left with only six such matrices. We resolve five of these cases; see Table 1 for an overview of these results.

▶ **Lemma 14** ($\star$). Deletion to List M-Partition *is NP-hard if M is one of the following matrices: (a) $m_{1,1} = m_{2,2} = m_{3,3} = *$, $m_{1,2} = m_{1,3} = m_{2,3} = 0$, and (b) $m_{1,1} = m_{2,2} = m_{3,3} = *$, $m_{1,3} = m_{2,3} = 0, m_{1,2} = 1$.*

▶ Remark 15. We do not know whether Deletion to List M-Partition is NP-hard or not when M is described as $m_{1,1} = m_{2,2} = m_{3,3} = *$, $m_{1,2} = 0, m_{1,3} = 1, m_{2,3} = *$. However, in the course of the proof of Theorem 1, we show that Deletion to List M-Partition is FPT, when M or $\overline{M}$ is equivalent to the above matrix.

We now briefly discuss the polynomial time solvability of the cases described in Theorem 1, item 1.

▶ **Lemma 16** ($\star$). Deletion to List M-Partition *is polynomial time solvable if M is one of the following matrices:*

(a) $\begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix}$, (b) $\begin{pmatrix} * & 0 & * \\ 0 & * & * \\ * & * & * \end{pmatrix}$, (c) $\begin{pmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & * \end{pmatrix}$.

Proof Sketch. In each of the cases below, (G, L, k) denotes an input instance of Deletion to List M-Partition. We assume that for each $v \in V(G)$, we have $L(v) \neq \emptyset$ (see Lemma 12). And recall that for any $X, Y \subseteq V(G)$, an (X, Y)-separator can be found in polynomial time [18].

(a) In this case, all entries of the matrix M are $*$s. Notice that a vertex $v \in V(G)$ can go to any one of the parts V_i in a list M-partition as long as $i \in L(v)$.

(b) Let $X = \{v \in V(G) \mid L(v) = \{1\}\}$, $Y = \{v \in V(G) \mid L(v) = \{2\}\}$ and $Z = \{v \in V(G) \mid 3 \in L(v)\}$. We can show that $S \subseteq V(G)$ is a solution for the Deletion to List M-Partition instance (G, L, k) if and only if S is an (X, Y)-separator in $G - Z$.

(c) In this case, indices 1 and 2 dominate each other. Thus we can assume that the list of no vertex contains both 1 and 2. Let $X = \{v \in V(G) \mid L(v) = \{3\}\}$ and $Y = \{x \in V(G) \mid L(v) \subseteq \{1, 2\}\}$. Notice that a set $S \subseteq V(G)$ is a solution to the Deletion to List M-Partition instance (G, L, k) if and only if S is an (X, Y)-separator. ◀

Now we turn our attention to the parameterized complexity of Deletion to List M-Partition when M is of order 3. First, we consider the matrices defined in item 2 of Theorem 1.

▶ **Observation 17.** *The problem* Deletion to List M-Partition *is para-NP-hard, when M is one of the two matrices matrices define in item 2 of Theorem 1. The first matrix in item 2 corresponds to a 3-colouring and the second matrix corresponds to a stable-cutset partition. Therefore, the corresponding* List M-Partition *problems correspond to the problems* 3-Colouring *and* Stable Cutset*, respectively, both of which are NP-hard [19, 4].*

In the remaining part of this section, we consider some special cases from Theorem 1 and describe algorithms for these special cases. The rest of the case analysis leading to the proof of Theorem 1 has been omitted due to space constraints.

Algorithm for the Deletion to List Clique-Cutset Partition. The problem is Deletion to List M-Partition, where M is the matrix such that an M-partition is a clique-cutset partition, i.e., M is such that $m_{1,1} = m_{2,2} = *$, $m_{3,3} = 1$ and $m_{1,3} = m_{2,3} = *$. $m_{1,2} = 0$. Consider a clique-cutset partition $\mathcal{V} = \{V_1, V_2, V_3\}$ of a graph G. The subgraph $G[V_3]$ is a clique, and V_3 is also a cutset between the parts V_1 and V_2. Hence the name clique-cutset partition. We now prove the following lemma, the proof of which relies on a family of vertex subsets that "separate cliques and stable-pairs," defined below.

▶ **Lemma 18.** Deletion to List Clique-Cutset Partition *is solvable in* $2.3146^k n^{\mathcal{O}(\log n)}$ *time.*

▶ **Definition 19.** *For a graph G, a pair of disjoint sets $A, B \subseteq V(G)$ is a* stable-pair *if for every $a \in A$ and $b \in B$, $ab \notin E(G)$. A family $\mathcal{F} \subseteq \{F \mid F \subseteq V(G)\}$* separates cliques and stable-pairs *if for every $A, B, C \subseteq V(G)$ such that A, B is a stable-pair, $G[C]$ is a clique and $C \cap (A \cup B) = \emptyset$, there is $F \in \mathcal{F}$ such that $C \subseteq F$ and either $F \cap A = \emptyset$ or $F \cap B = \emptyset$.*

▶ **Proposition 20** (Theorem 4.3 [17])**.** *Every graph on n vertices has a family of size $n^{\log n}$ that separate cliques and stable-pairs. Moreover, such a family can be found in time $n^{\mathcal{O}(\log n)}$.*

Let us see the implication of this proposition to our problem. Consider an instance (G, L, k) of Deletion to List Clique-Cutset Partition, and its solution $S \subseteq V(G)$ (if it exists). Let $\mathcal{F}$ be a family of sets that separates cliques and stable-pairs in G. And consider the list

Algorithm 3.1 Algorithm for Deletion to List Clique-Cutset Partition.

Input: (G, L, k).
Output: yes or no.
1 Construct a family $\mathcal{F}$ of size $n^{\log n}$ that separates cliques and stable-pairs in G.
2 **for** *each* $F \in \mathcal{F}$ **do**
3 Define $L_1 : V(G) \to 2^{[3]}$ as follows. For every $v \in F$, set $L_1(v) = L(v) \setminus \{1\}$. For every $v \notin F$, set $L_1(v) = L(v) \setminus \{3\}$.
4 **if** *Proposition 9 returns yes for the instance* (G, L_1, k) **then**
5 **return** yes
6 Define $L_2 : V(G) \to 2^{[3]}$ as follows. For every $v \in F$, set $L_2(v) = L(v) \setminus \{2\}$. For every $v \notin F$, set $L_2(v) = L(v) \setminus \{3\}$.
7 **if** *Proposition 9 returns yes for the instance* (G, L_2, k) **then**
8 **return** yes
9 **return** no

clique-cutset partition (V_1, V_2, V_3) of $G - S$. Note that $G[V_3]$ is a clique, V_1, V_2 is a stable-pair, and V_3 is disjoint from $V_1 \cup V_2$. Then, by Proposition 20, there exists $F \in \mathcal{F}$ such that F contains V_3 and F is disjoint from at least one of V_1 and V_2. Consider the set of lists L_1 obtained from L by removing 1 from $L(v)$ for every $v \in F$ and 3 from $L(v)$ for every $v \notin F$. Observe that if $F \cap V_1 = \emptyset$, then S is a solution for the Deletion to List M-Partition instance (G, L_1, k) as well. Similarly, let L_2 be the set of lists obtained from L by removing 2 from $L(v)$ for every $v \in F$ and 3 from $L(v)$ for every $v \notin F$. Then if $F \cap V_2 = \emptyset$, then S is a solution for the Deletion to List M-Partition instance (G, L_2, k) as well. But we know that either $F \cap V_1 = \emptyset$ or $F \cap V_2 = \emptyset$. Therefore, S is a solution to either (G, L_1, k) or (G, L_2, k). These observations lead us to our algorithm (see Algorithm 3.1).

The correctness of Algorithm 3.1 is apparent from our previous discussions. As for the running time, Step 1 takes $n^{\mathcal{O}(\log n)}$ time to construct $\mathcal{F}$, and we have $|\mathcal{F}| \leq n^{\log n}$. For each $F \in \mathcal{F}$, Steps 2–8 take $2.3146^k n^{\mathcal{O}(1)}$ time. Therefore, the algorithm runs in time $2.3146^k n^{\mathcal{O}(\log n)}$. We have thus proved Lemma 18.

Algorithm for Deletion to List M-Partition, where M is the bipartite-star matrix or the three-stars matrix. We are now going to design an FPT algorithm that works for two different partitions. The first of these is an M-partition when M is defined by $m_{1,1} = m_{2,2} = 0$, $m_{3,3} = *$, $m_{1,2} = *$, $m_{1,3} = m_{2,3} = 0$. We call M the bipartite-star matrix and an M-partition a bipartite-star partition. The second partition is an M-partition for M defined by $m_{1,1} = m_{2,2} = m_{3,3} = *$, $m_{1,2} = m_{1,3} = m_{2,3} = 0$. We call M the three-stars matrix and an M-partition a three-stars partition. We shall prove the following lemma.

▶ **Lemma 21.** Deletion to List M-Partition, *where M is either the bipartite-star matrix or the three-stars matrix is fixed-parameter tractable.*

We prove Lemma 21 by showing that Algorithm 3.2 solves Deletion to List M-Partition in $16^k n^{\mathcal{O}(1)}$ time, when M is either the bipartite-star matrix or the three-stars matrix. Let (G, L, k) be an instance of Deletion to List M-Partition. The idea behind our algorithm is as follows. Assume that (G, L, k) is a yes-instance. Let $S \subseteq V(G)$ be an optimal solution for the problem, and $\mathcal{V} = \{V_1, V_2, V_3\}$ be a list M-partition of $G - S$. Then, S contains an (X, Y)-separator where $X = \{u \in V(G) \mid L(u) = \{3\}\}$ and $Y = \{u \in V(G) \mid L(u) \subseteq \{1, 2\}\}$. And for a vertex u with $3 \in L(u)$, note that u can be placed in V_3 (because $m_{3,3} = *$) if u

Algorithm 3.2 Algorithm for the proof of Lemma 21.

Input: (G, L, k).
Output: yes or no.
1 Define $X = \{u \in V(G) \mid L(u) = \{3\}\}$ and $Y = \{u \in V(G) \mid L(u) \subseteq \{1, 2\}\}$.
2 Let $\mathcal{F}$ be the family of all important (X, Y)-separators of size at most k.
3 **for** *each* $\hat{S} \in \mathcal{F}$ **do**
4 Let $\hat{Z}$ be the reachability set of $Y \setminus \hat{S}$ in $G - \hat{S}$. (Note that for $v \in V(G) \setminus (\hat{S} \cup \hat{Z})$, we have $3 \in L(v)$, and for $v \in \hat{Z}$, $L(v) \cap \{1, 2\} \neq \emptyset$.)
5 Define a new list function L' for the vertices in $\hat{Z}$. For every $v \in \hat{Z}$, let $L'(v) = L(v) \setminus \{3\}$. (Then $|L'(v)| \leq 2$ for every $v \in \hat{Z}$.)
6 Set $k' = k - |\hat{S}|$.
7 **if** *Proposition 9 returns yes for the instance* $(G[\hat{Z}], L', k')$ **then**
8 **return** yes
9 **return** no

is not reachable from Y in $G - S$. Hence we try to place as many vertices as possible in V_3 by choosing an (X, Y)-separator that is "farthest from X," (and by augmenting such a separator to ensure that the other constraints dictated by M are also satisfied). For this, Algorithm 3.2 uses the concept of an important separator, introduced by Marx in [27]. For a graph G and $A \subseteq V(G)$, $R_G(A) = \{v \in V(G) \mid$ there is a path from x to v in G for some $x \in A\}$. And the set $R_G(A)$ is called the *reachability set of A in G*. For $A, B \subseteq V(G)$, a minimal (A, B)-separator $S \subseteq V(G)$ is said to be an *important (A, B)-separator* if there exists no (A, B)-separator S' such that $|S'| \leq |S|$ and $R_{G-S}(A) \subset R_{G-S'}(A)$.

▶ **Proposition 22** ([27, 6]). *Given a graph G, sets $A, B \subseteq V(G)$ and an integer k, G contains at most 4^k important (A, B)-separators of size at most k. Moreover, all these important separators can be enumerated in time $\mathcal{O}(4^k(|V(G)| + |E(G)|))$.*

▶ **Lemma 23.** *Algorithm 3.2 is correct.*

Proof. In light of Remark 13, we assume that $X, Y \neq \emptyset$, where X and Y are as defined in Step 1 of Algorithm 3.2.

Notice first that if Algorithm 3.2 returns yes, then (G, L, k) is indeed a yes-instance. Assume that the algorithm returns yes. Then there exists $\hat{S} \in \mathcal{F}$ such that $\hat{S}$ is an important (X, Y)-separator, and $(G[\hat{Z}], L', k')$ is a yes-instance, where $\hat{Z} = R_{(G-\hat{S})}(Y \setminus \hat{S})$, (and $X, Y, \mathcal{F}, L'$ and k' are as defined in the algorithm). Let $U \subseteq \hat{Z}$ be an optimal solution for the instance $(G[\hat{Z}], L', k')$. Then, $|U| \leq k' = k - |\hat{S}|$. Let (V_1, V_2, V_3) be a partition of $Z \setminus U$ that respects L'. Since $3 \notin L'(v)$ for every $v \in Z$, we have $V_3 = \emptyset$. It is not difficult to see that $(V_1, V_2, V(G) \setminus (\hat{Z} \cup \hat{S} \cup U))$ is a list partition of $V(G) \setminus (\hat{S} \cup U)$ that respects L. That is, $\hat{S} \cup U$ is a solution for the instance (G, L, k) and $|\hat{S} \cup U| \leq k$.

Now, we prove that if (G, L, k) is a yes-instance, then Algorithm 3.2 returns yes. Assume that (G, L, k) is a yes-instance, and let S be an optimal solution to the list partition instance (G, L, k), and $S' \subseteq S$ be a minimal (X, Y)-separator. Let $\hat{S}$ be an important (X, Y)-separator that dominates S', i.e., $|\hat{S}| \leq |S'|$ and $R_{G-S'}(X) \subset R_{G-\hat{S}}(X)$. We will show that $\tilde{S} = (S \setminus S') \cup \hat{S}$ is also an optimal solution.

We first show that $\hat{S}$ is an $(S' \setminus \hat{S}, Y)$-separator. Suppose not, then there is an s–y path P in $G - \hat{S}$ for some $s \in S' \setminus \hat{S}$ and $y \in Y \setminus \hat{S}$. As S' is a minimal (X, Y)- separator, there is an X–Y path, say P', that intersects S' only in s. Consider the X–s subpath of

P' and call it P''. None of the vertices of P'' can belong to $\hat{S}$, as for each $v \in V(P'')$, $v \in R_{G-S'}(X) \subseteq R_{G-\hat{S}}(X)$. Thus, P'' is a path in $G - \hat{S}$. Then P'' followed by the path P is an X–Y path in $G - \hat{S}$, which contradicts the fact that $\hat{S}$ is an (X, Y)-separator. Thus, $\hat{S}$ is an $(S' \setminus \hat{S}, Y)$-separator.

Now, let Z' be the reachability set of $Y \setminus S'$ in $G - S'$, and $\hat{Z}$ the reachability set of $Y \setminus \hat{S}$ in $G - \hat{S}$. (Notice that for every vertex $v \notin \hat{Z}$, we have $3 \in L(v)$, and therefore, v can be placed in V_3 without violating any of the constraints on $m_{3,3}$ or $m_{1,3}$ or $m_{2,3}$. And every vertex $v \in \hat{Z}$ has either 1 or 2 on its list.) We claim that $\hat{Z} \subseteq Z'$. Consider $z \in \hat{Z}$. Let Q be a w–z path in $G - \hat{S}$ for some $w \in Y$. Suppose $z \notin Z'$. Then Q must pass through S', i.e., $V(Q) \cap S' \neq \emptyset$. Let $u \in V(Q) \cap S' \neq \emptyset$. But then the u–w subpath of Q is an $(S' \setminus \hat{S}) - Y$ path in $G - \hat{S}$, which contradicts the fact that $\hat{S}$ is an $(S' \setminus \hat{S}, Y)$-separator. Thus, $\hat{Z} \subseteq Z'$, and $G[\hat{Z}]$ is an induced subgraph of $G[Z']$. Observe that $(S \setminus S')$ is a solution of size at most $k - |S'|$ for the list partition instance on $G[Z']$. Hence, $(S \setminus S')$ is a solution for the list partition instance on $G[\hat{Z}]$ as well. ◀

▶ **Lemma 24** ($\star$). *Algorithm 3.2 runs in time* $16^k n^{\mathcal{O}(1)}$.

Lemmas 23 and 24 together prove Lemma 21. Next we give a proof sketch of Theorem 1.

Proof sketch of Theorem 1. The proofs of item 1, item 2 and item 3 of the theorem statement follow from Lemma 16, Observation 17 and Lemma 18, respectively. Therefore, we next focus on the proof of item 4 of the theorem statement. We only consider matrices that are not covered by any of the previous three cases. Let us first classify matrices M depending on the number of off-diagonal entries that are 0s. In fact, we will only make distinctions based on the off-diagonal entries in the upper triangular submatrix M_U of M, since M is a symmetric matrix. Below, we deal with one of the classes, which is illustrative of the techniques used to resolve these problems.

Exactly two off-diagonal entries of M_U are 0s. Assume that $m_{1,3} = m_{2,3} = 0$ and $m_{1,2} \in \{1, *\}$. All other cases of two off-diagonal entries of M_U being 0 can be symmetrically argued. If any one of $m_{1,1}, m_{2,2}$ or $m_{3,3}$ is 1, then M has a row that contains both a 0 and a 1, and then by Proposition 10, the problem is fixed-parameter tractable. So assume that $m_{1,1}, m_{2,2}, m_{3,3} \in \{0, *\}$. We consider each possibility of $m_{1,2}$ for our analysis. First, suppose $m_{1,2} = *$. If $m_{1,1} = *$, then index 1 dominates 2. If $m_{2,2} = *$, then index 2 dominates 1. In either case, the problem is fixed-parameter tractable, by Propositions 8 and 9. So assume that $m_{1,1} = m_{2,2} = 0$. If $m_{3,3} = 0$, then index 1 dominates 3, and by Propositions 8 and 9 the problem is again fixed-parameter tractable. If $m_{3,3} = *$, then an M-partition is a bipartite-star partition, and by Lemma 21, the problem is fixed-parameter tractable. The other possibility for $m_{1,2}$ is that $m_{1,2} = 1$. Then M has a row containing both a 0 and a 1, and by Proposition 10, the problem is fixed-parameter tractable. ◀

4 Conclusion

We almost complete the parameterized classification of Deletion to List M-Partition when the matrix M is of order ≤ 3, or when it is of order 4 and has its diagonal entries from $\{0, 1\}$. We do not know whether the Deletion to Clique Cutset problem is FPT– we obtain an algorithm with running time $2.3146^k n^{\mathcal{O}(\log n)}$, where k is the solution size. Also, the NP-hardness of the Deletion to List M-Partition problem when $m_{1,1} = m_{2,2} = m_{3,3} = *$, $m_{1,2} = 0, m_{1,3} = 1, m_{2,3} = *$ is open, although we give an FPT algorithm, parameterized by

the solution size. It would be interesting to complete the classification of these problems, as well as the paramaterized dichotomy of Deletion to List M-Partition for all matrices of order 4. We are also interested in optimising the running time of our algorithms, and studying the kernelization complexity of these problems in the future.

References

1 Noga Alon and Michael Tarsi. Colorings and orientations of graphs. *Combinatorica*, 12(2):125–134, 1992.

2 Bengt Aspvall, Michael F. Plass, and Robert E. Tarjan. A Linear-Time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas. *Information Processing Letters*, 8(3):121–123, 1979.

3 Armen S. Asratian, Tristan M. J. Denley, and Roland Häggkvist. *Bipartite graphs and their applications*, volume 131. Cambridge University Press, 1998.

4 Andreas Brandstädt, Feodor F. Dragan, Van Bang Le, and Thomas Szymczak. On stable cutsets in graphs. *Discrete Applied Mathematics*, 105(1-3):39–50, 2000. doi:10.1016/S0166-218X(00)00197-9.

5 Kathie Cameron, Elaine M. Eschen, Chính T. Hoàng, and R. Sritharan. The Complexity of the List Partition Problem for Graphs. *SIAM J. Discrete Math.*, 21(4):900–929, 2007.

6 Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM*, 55(5):21:1–21:19, 2008.

7 Rajesh Chitnis, László Egri, and Dániel Marx. List H-Coloring a Graph by Removing Few Vertices. *Algorithmica*, 78(1):110–146, 2017.

8 Marek Cygan, Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, Ivan Mihajlin, Jakub Pachocki, and Arkadiusz Socała. Tight Bounds for Graph Homomorphism and Subgraph Isomorphism. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1643–1649, 2016.

9 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The Complexity of Multiway Cuts (Extended Abstract). In *ACM Symposium on Theory of Computing (STOC)*, pages 241–251, 1992.

10 Josep Díaz, Maria Serna, and Dimitrios M. Thilikos. (H,C,K)-coloring: Fast, easy, and hard cases. In *Mathematical Foundations of Computer Science (MFCS)*, pages 304–315, 2001.

11 Josep Díaz, Maria Serna, and Dimitrios M Thilikos. Recent results on parameterized H-colorings. *Discrete Mathematics and Theoretical Computer Science*, 63:65–86, 2004.

12 László Egri, Andrei Krokhin, Benoit Larose, and Pascal Tesson. The complexity of the list homomorphism problem for graphs. *Theory of Computing Systems*, 51(2):143–178, 2012.

13 Paul Erdős, Arthur L. Rubin, and Herbert Taylor. Choosability in graphs. In *Proc. West Coast Conf. on Combinatorics, Graph Theory and Computing, Congressus Numerantium*, volume 26, pages 125–157, 1979.

14 Tomás Feder and Pavol Hell. List Homomorphisms to Reflexive Graphs. *Journal of Combinatorial Theory, Series B*, 72(2):236–250, 1998.

15 Tomás Feder and Pavol Hell. Full Constraint Satisfaction Problems. *SIAM J. Comput.*, 36(1):230–246, 2006. doi:10.1137/S0097539703427197.

16 Tomás Feder, Pavol Hell, and Jing Huang. List Homomorphisms and Circular Arc Graphs. *Combinatorica*, 19(4):487–505, October 1999.

17 Tomás Feder, Pavol Hell, Sulamita Klein, and Rajeev Motwani. List Partitions. *Journal of Discrete Mathematics*, 16(3):449–478, 2003.

18 Lester R. Ford and Delbert R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.

19 Michael R. Garey and David S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. *Computers and Intractability*, page 340, 1979.

20 Michael. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some Simplified NP-Complete Graph Problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1.

21 Andreas Göbel, Leslie Ann Goldberg, Colin McQuillan, David Richerby, and Tomoyuki Yamakami. Counting List Matrix Partitions of Graphs. *SIAM J. Comput.*, 44(4):1089–1118, 2015. doi:10.1137/140963029.

22 Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

23 Melven R. Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(1-2):15–20, 1967.

24 Marek Kubale. Some results concerning the complexity of restricted colorings of graphs. *Discrete Applied Mathematics*, 36(1):35–46, 1992.

25 S. Føldes and P. L. Hammer. Split graphs. *South-Eastern Conference on Combinatorics, Graph Theory and Computing (SEICCGTC)*, pages 311–315, 1977.

26 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster Parameterized Algorithms Using Linear Programming. *Transactions on Algorithms*, 11(2):15:1–15:31, 2014.

27 Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*, 351(3):394–406, 2006.

28 Vadim G. Vizing. Vertex colorings with given colors. *Diskret. Analiz*, 29:3–10, 1976.

29 Margit Voigt. List colourings of planar graphs. *Discrete Mathematics*, 120(1-3):215–219, 1993.

30 Mihalis Yannakakis. Node-and Edge-deletion NP-complete Problems. In *ACM Symposium on Theory of Computing (STOC)*, pages 253–264, 1978.