Parameterized Dichotomy 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
UMI ReLax
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 conduct a dichotomy study 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 dichotomy, Almost 2-SAT, important separators, iterative compression

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. [20], 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 $m_{i,j} = *$). M-Partition takes as input a graph G, and the objective is to determine if G admits an M-partition.

The M-Partition 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, 24, 28]. 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 $\ell \geq 3$ [23, 22].

For an $\ell \times \ell$ matrix M, the List M-Partition problem 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 every vertex $v \in V(G)$, if $v \in V_i$ for some $i \in [\ell]$ then $i \in L(v)$. The List M-Partition problem takes as input a graph G, a list function $L : V(G) \to 2^{[\ell]}$, and the objective is to determine whether 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 of $V(G)$ to the entire 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, 7, 10, 13, 14, 15, 19, 20, 27, 34] 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 [33] and Erdős et al. [16]. Since its advent, the problem has been extensively studied in both graph theory and algorithms [16, 27, 34]. 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 [17], and has been extensively studied in the literature [1, 10, 13, 14, 15, 19].

Feder et al. [20] 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 [18, 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.

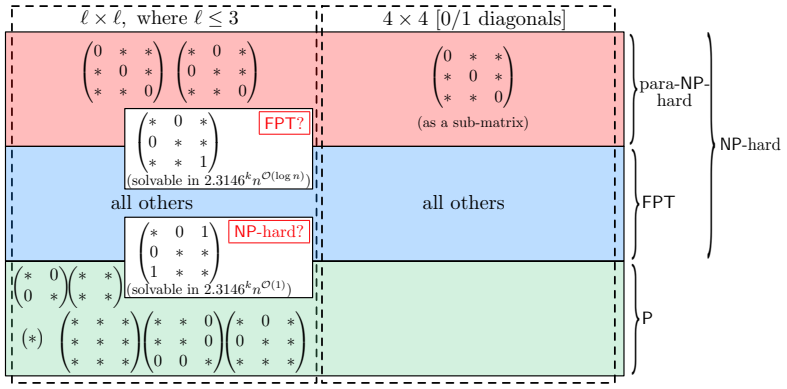

Figure 1 An overview of the results on matrices, up to complementation and equivalence.

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 classical NP-hard problems, such as Vertex Cover, Odd Cycle Transversal (OCT), Split Vertex Deletion (SVD) and Multiway Cut, to name a few. These problems have been studied in both classical and parameterized complexity settings and are all NP-hard [35, 11]. Chitnis et al. [9] 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 [9], 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 [9].[1]

Our Results. We study the parameterized complexity of Deletion to List M-Partition for different matrices M, and make progress towards obtaining 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. Our results show 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 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 results for Deletion to List M-

[1] The results in [9] 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.

Partition when M is of order at most 2 are quite simple (see Appendix C). The techniques required to resolve the problems associated with order 3 matrices are more sophisticated. The polynomial time algorithms for the order 3 matrices are based on ideas that help to reduce the problem to an equivalent minimum separator problem [21]. For our FPT algorithms, we utilize the notion of "important separators", which was introduced by Marx [31]. 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 (symmetric) matrix M' defined by $m'_{1,1} = m'_{2,2} = m'_{1,3} = m'_{2,3} = *$, $m'_{1,2} = 0$ and $m'_{3,3} = 1$, 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. [20] 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 (Please refer Figure 1 and Appendix F for more details). 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.

Due to paucity of space, most of the results and proofs have been moved to the appendix. Results marked with a $\star$ have their (full) proofs in the appendix. In the main sections, we present the more involved algorithmic results and techniques for the cases when M has order 3. These techniques illustrate most of the algorithmic ideas used in this paper.

2 Preliminaries and Basic Tools

Graph Theory. 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 an 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 [26].

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 [30], 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 1. 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 bijective function $\pi : [\ell] \to [\ell]$ such that $m'_{i,j} = m_{\pi(i),\pi(j)}$. And such a bijective function π is called a *witness-bijection* for (M, M'). Notice that if M is equivalent to M', then M' is also equivalent to M with witness-bijection 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 1.** *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 [20], which will be used throughout.

Reducing LIST M-PARTITION to 2-SAT, for a 2×2 matrix M. It was shown in [20] 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 (see Appendix B).

▶ **Proposition 2 ($\star$).** *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. Indeed it is possible to do that, as shown in [20], if the matrix M has a row that contains both a 0 and 1 (see Proposition 2.3 and Corollary 2.4 in [20]).

▶ **Proposition 3** ([20]). *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 4.** *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 5** (Proposition 2.5 [20]). *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 2. The properties of ψ and f (as in Proposition 2), 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 2, 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 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 2 gives us the following result.

▶ **Proposition 6.** Deletion to 2-List M-Partition *is fixed-parameter tractable with running time $2.3146^k n^{\mathcal{O}(1)}$.*

Now using Propositions 3 and 6, we obtain the following result.

▶ **Proposition 7.** *Let M be a 3×3 matrix such that M has a row that contains both a 0 and 1. Then* Deletion to List M-Partition *is fixed-parameter tractable.*

▶ **Proposition 8** ($\star$). 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 9** ($\star$). *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.

Matrices (and witness-bijection $\pi : [3] \to [3]$) $\pi(1)=1, \pi(2)=3, \pi(3)=2$	$\pi(1)=3, \pi(2)=2, \pi(3)=1$	Equivalent matrix	Class	Proof
		$\begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix}$	P	Lemma 13(a)
$\begin{pmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{pmatrix}$	$\begin{pmatrix} * & * & * \\ * & * & 0 \\ * & 0 & * \end{pmatrix}$	$\begin{pmatrix} * & 0 & * \\ 0 & * & * \\ * & * & * \end{pmatrix}$	P	Lemma 13(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 13(c)
		$\begin{pmatrix} * & 0 & 0 \\ 0 & * & 0 \\ 0 & 0 & * \end{pmatrix}$	NP-hard	Lemma 22(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 22(b)

Figure 2 An overview of our NP-hardness vs. P results (not covered by Proposition 8) for matrices of order 3×3, where complement matrices are not shown.

▶ Remark 10. 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 $L(v_i) = \{i\}$.

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. In this 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 (see Appendix E for complete details). The following theorem describes our results of Deletion to List M-Partition for matrices M of order 3.

▶ **Theorem 11.** *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*

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

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. *admits 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.*

Before we go into the proof of Theorem 11, we first address the question of NP-hardness versus polynomial time solvability of these problems. Already we saw in Proposition 8 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 Figure 2 for an overview of these results.

▶ Remark 12. 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 11, we show that Deletion to List M-Partition is FPT, when M or $\overline{M}$ is equivalent to the above matrix.

The remaining NP-hardness results as well as the proof of para-NP-hardness for the cases described in Theorem 11, item 2 can be found in Appendix E. We now briefly discuss the polynomial time solvability of the cases described in Theorem 11, item 1.

▶ **Lemma 13** ($\star$). Deletion to List M-Partition *is polynomial time solvable if M is one of the following matrices: (a) $m_{1,1} = m_{2,2} = m_{3,3} = *$, $m_{1,2} = m_{1,3} = m_{2,3} = *$, (b) $m_{1,1} = m_{2,2} = m_{3,3} = *$, $m_{1,2} = 0, m_{1,3} = m_{2,3} = *$, and (c) $m_{1,1} = m_{2,2} = m_{3,3} = *$, $m_{1,2} = *, m_{1,3} = m_{2,3} = 0$.*

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 9). And recall that for any $X, Y \subseteq V(G)$, an (X, Y)-separator can be found in polynomial time [21].
(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 in a list M-partition, without violating any of the constraints.
(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. In this section, we consider some special cases from Theorem 11 and describe algorithms for these special cases. The rest of the case analysis leading to the proof of Theorem 11 has been deferred to Appendix E.

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

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

Input: (G, L, k).
Output: yes or no.

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

▶ **Definition 15.** *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 16** (Theorem 4.3 [20])**.** *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 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 16, there exists $F \in \mathcal{F}$ such that F contains V_3 and F is disjoint from (at least) one of V_1 or 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 $F \in \mathcal{F}$, Step 2-8 takes $2.3146^k n^{\mathcal{O}(1)}$ time. Therefore, the algorithm runs in time $2.3146^k n^{\mathcal{O}(\log n)}$. We have thus proved Lemma 14.

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.

Algorithm 3.2: Algorithm for the proof of Lemma 17.

Input: (G, L, k).
Output: yes or no.

Define $X = \{u \in V(G) \mid L(u) = \{3\}\}$ and $Y = \{u \in V(G) \mid L(u) \subseteq \{1, 2\}\}$.
Let $\mathcal{F}$ be the family of all important (X, Y)-separators of size at most k.
for *each* $\hat{S} \in \mathcal{F}$ **do**
    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 $v \in \hat{Z}$, either $1 \in L(v)$ or $2 \in L(v)$.)
    For every $v \in \hat{Z}$, if $3 \in L(v)$, let $L(v) \leftarrow L(v) \setminus \{3\}$. (The list of every vertex in $\hat{Z}$ has size at most 2.) Let L' be the new set of lists.
    Set $k' = k - |\hat{S}|$.
    if *Proposition 6 returns yes for the instance* $(G[\hat{Z}], L', k')$ **then**
        return yes
return no

▶ **Lemma 17** (⋆). 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 17 by showing that Algorithm 3.2 solves Deletion to List M-Partition in $4^k n^{\mathcal{O}(1)}$ 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 this. 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 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 other constraints dictated by M are also satisfied). For this, Algorithm 3.2 uses the concept of an important separator, introduced by Marx in [31]. 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 $R_{G-S}(A) \subset R_{G-S'}(A)$. It is known that G contains at most 4^k important (A, B)-separators of size at most k, and that all these important separators can be enumerated in time $\mathcal{O}(4^k(|V(G)| + |E(G)|))$[31, 8].

Next we give a proof sketch of Theorem 11 (for complete details see Appendix E).

Proof sketch of Theorem 11 (⋆). The proof of item 1, item 2 and item 3 of the theorem statement follows from Lemma 13, Observation 3 and Lemma 14, 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. The case when at most one off-diagonal entry in M_U is 0 is in the full proof, given in Appendix E. The following case exhibits 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, then from Proposition 7, 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 5 and 6. So assume that $m_{1,1} = m_{2,2} = 0$. If $m_{3,3} = 0$, then index 1 dominates 2, and by Propositions 5 and 6 the problem is again fixed-parameter tractable. If $m_{3,3} = *$, then an M-partition is a bipartite-star partition, and by Lemma 17, 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 7, 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 MJ Denley, and Roland Häggkvist. *Bipartite graphs and their applications*, volume 131. Cambridge University Press, 1998.
4. Andreas Brandstädt, Feodor F Dragan, Thomas Szymczak, et al. On stable cutsets in graphs. *Discrete Applied Mathematics*, 105(1-3):39–50, 2000.
5. Andreas Brandstädt, Van Bang Le, and Thomas Szymczak. The complexity of some problems related to graph 3-colorability, 1998.
6. Rowland Leonard Brooks. On colouring the nodes of a network. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 37, pages 194–197. Cambridge University Press, 1941.
7. 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.
8. 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.
9. Rajesh Chitnis, László Egri, and Dániel Marx. List H-coloring a graph by removing few vertices. *Algorithmica*, 78(1):110–146, 2017.
10. 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.

406 **11** E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The
407 complexity of multiway cuts (extended abstract). In *ACM Symposium on Theory of Computing*
408 *(STOC)*, pages 241–251, 1992.
409 **12** Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis
410 Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–
411 894, 1994.
412 **13** Josep Díaz, Maria Serna, and Dimitrios M. Thilikos. (H,C,K)-coloring: Fast, easy, and hard
413 cases. In *Mathematical Foundations of Computer Science (MFCS)*, pages 304–315, 2001.
414 **14** Josep Díaz, Maria Serna, and Dimitrios M Thilikos. Recent results on parameterized H-
415 colorings. *Discrete Mathematics and Theoretical Computer Science*, 63:65–86, 2004.
416 **15** László Egri, Andrei Krokhin, Benoit Larose, and Pascal Tesson. The complexity of the list
417 homomorphism problem for graphs. *Theory of Computing Systems*, 51(2):143–178, 2012.
418 **16** Paul Erdős, Arthur L Rubin, and Herbert Taylor. Choosability in graphs. In *Proc. West Coast*
419 *Conf. on Combinatorics, Graph Theory and Computing, Congressus Numerantium*, volume 26,
420 pages 125–157, 1979.
421 **17** Tomas Feder and Pavol Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial*
422 *Theory, Series B*, 72(2):236 – 250, 1998.
423 **18** Tomás Feder and Pavol Hell. Full constraint satisfaction problems. *SIAM J. Comput.*,
424 36(1):230–246, 2006.
425 **19** Tomas Feder, Pavol Hell, and Jing Huang. List homomorphisms and circular arc graphs.
426 *Combinatorica*, 19(4):487–505, Oct 1999.
427 **20** Tomás Feder, Pavol Hell, Sulamita Klein, and Rajeev Motwani. List partitions. *Journal of*
428 *Discrete Mathematics*, 16(3):449–478, 2003.
429 **21** Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian Journal*
430 *of Mathematics*, 8(3):399–404, 1956.
431 **22** Michael R Garey and David S Johnson. Computers and intractability: A guide to the theory
432 of NP-completeness. *Computers and Intractability*, page 340, 1979.
433 **23** Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified NP-complete
434 problems. In *ACM Symposium on Theory of computing (STOC)*, pages 47–63, 1974.
435 **24** Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.
436 **25** Sudeshna Kolay and Fahad Panolan. Parameterized algorithms for deletion to (r, ell)-graphs.
437 In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical*
438 *Computer Science, 2015, December 16-18, 2015, Bangalore, India*, pages 420–433, 2015.
439 **26** Melven R Krom. The decision problem for a class of first-order formulas in which all disjunctions
440 are binary. *Mathematical Logic Quarterly*, 13(1-2):15–20, 1967.
441 **27** Marek Kubale. Some results concerning the complexity of restricted colorings of graphs.
442 *Discrete Applied Mathematics*, 36(1):35–46, 1992.
443 **28** S. Føldes and P. L. Hammer. Split graphs. *South-Eastern Conference on Combinatorics,*
444 *Graph Theory and Computing (SEICCGTC)*, pages 311–315, 1977.
445 **29** John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties
446 is NP-complete. *Journal of Computer and System Sciences*, 20(2):219 – 230, 1980.
447 **30** Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket
448 Saurabh. Faster parameterized algorithms using linear programming. *Transactions on*
449 *Algorithms*, 11(2):15:1–15:31, 2014.
450 **31** Dániel Marx. Parameterized graph separation problems. *Theoretical Computer Science*,
451 351(3):394–406, 2006.
452 **32** San Skulrattanakulchai. δ-list vertex coloring in linear time. *Information Processing Letters*,
453 98(3):101–106, 2006.
454 **33** Vadim G Vizing. Vertex colorings with given colors. *Diskret. Analiz*, 29:3–10, 1976.
455 **34** Margit Voigt. List colourings of planar graphs. *Discrete Mathematics*, 120(1-3):215–219, 1993.
456 **35** Mihalis Yannakakis. Node-and edge-deletion NP-complete problems. In *ACM Symposium on*
457 *Theory of Computing (STOC)*, pages 253–264, 1978.

A NP-hardness of Tripartite Vertex Cover

In this Section, we show that Tripartite Vertex Cover is NP-hard. Recall the problem statement.

> Tripartite Vertex Cover
> **Input:** A graph G with a partition of $V(G)$ into three (disjoint) independent sets V_1, V_2, and V_3, and an integer k.
> **Question:** Does there exist a set of vertices $X \subseteq V(G)$ such that $|X| \leq k$ and $G - X$ is a graph with no edges?

A *cubic* graph is a graph in which every vertex has degree exactly three. A *connected* graph is a graph in which there is a path between ever pair of vertices. Cubic Connected Vertex Cover is the Vertex Cover problem where the input graph is a cubic connected graph. It is known that the problem Cubic Connected Vertex Cover is NP-hard [23]. We note that it is a folklore result that Tripartite Vertex Cover is NP-hard, the proof of which can obtained by a reduction from Cubic Connected Vertex Cover. For the sake of completeness, we explicitly give this reduction below.

▶ **Lemma 18.** Tripartite Vertex Cover *is* NP*-hard.*

Proof. Let (G, k) be an instance of Cubic Connected Vertex Cover. We assume that $|V(G)| \geq 5$, such an assumption is valid as otherwise, we can resolve the instance in polynomial time. We construct an instance $(G', V_1', V_2', V_3', k')$ of Tripartite Vertex Cover as follows. By Brooks' Theorem [6], there exists a (proper) colouring $\mathsf{col} : V(G) \to [3]$, such that for each $uv \in E(G)$, we have $\mathsf{col}(u) \neq \mathsf{col}(v)$. We compute a proper colouring col of G with 3 colours using the algorithm in [32]. For $i \in [3]$, we set $V_i' = \{v \in V(G) \mid \mathsf{col}(v) = i\}$. Finally, we set $G' = G$ and $k' = k$. It is easy to see that (G, k) is a Yes instance of Cubic Connected Vertex Cover if and only if $(G', V_1', V_2', V_3', k')$ is a Yes instance of Tripartite Vertex Cover. This concludes the proof. ◀

B Missing Proofs from Section 2

Proof of Proposition 2. Let $(G, L : V(G) \to 2^{[\ell]})$ be an instance of 2-List M-Partition. We describe the construction of 2-SAT formula ψ with variable set $\mathsf{Var}(\psi)$ and the function $f : V(G) \to \mathsf{Var}(\psi)$. For a vertex $v \in V(G)$ with $L(v) = \{i, j\}$, where $i \leq j$, we add a variable $x_v[i]$ to $\mathsf{Var}(\psi)$ and set $f(v) = x_v[i]$. For notational convenience, we let $x_v[j] = \overline{x_v[i]}$, when $i \neq j$. We note that we have added exactly one variable for each vertex in G. Consider a vertex $v \in V(G)$ such that $L(v) = \{i\}$, for some $i \in [\ell]$. To ensure that v belongs to part V_i, we add the clause $(x_v[i] \vee x_v[i])$ to ψ. For a vertex $v \in V(G)$ such that $L(v) = \{i, j\}$, where $i \neq j$, notice that either $x_v[i] = 1$ or $x_v[i] = 0$. The interpretation in the former case is that v belongs to part V_i, and in the later case, we have $x_v[j] = \overline{x_v[i]}$ and thus v belongs to part V_j. For $i \in [\ell]$, let $X_i = \{x_v[i] \mid v \in V(G) \text{ and } i \in L(v)\}$. Consider (not necessarily distinct) integers $i, j \in [\ell]$. If $m_{i,j} = 0$, then vertices which belong to V_i and vertices which belong to V_j must not be adjacent. To achieve this, for (distinct) $x_u[i] \in X_i$ and $x_v[j] \in X_j$ such that $uv \in E(G)$, we add the clause $(\overline{x_u[i]} \vee \overline{x_v[j]})$ to ψ. Similarly, if $m_{i,j} = 1$, then vertices which belong to V_i and vertices which belong to V_j must be adjacent. Thus, for (distinct) $x_u[i] \in X_i$ and $x_v[j] \in X_j$ such that $uv \notin E(G)$, we add the clause $(\overline{x_u[i]} \vee \overline{x_v[j]})$ to ψ.

It is not difficult to see that ψ and f satisfy the desired properties. Since 2-SAT is solvable in polynomial time [26], we conclude that 2-List M-Partition is polynomial time solvable. ◀

Proof of Proposition 8. Consider an $\ell \times \ell$ matrix M which has at least one diagonal entry from $\{0,1\}$. Consider the case when there is $i \in [\ell]$, such that $m_{i,i} = 0$. Notice that from Observation 1, it is enough to consider the above case, as otherwise, we can complement the matrix. We show that Deletion to List M-Partition is NP-hard by a reduction from Vertex Cover. Given an instance (G,k) of Vertex Cover, we construct an instance (G',L,k') of Deletion to List M-Partition by taking $G' = G$, $k' = k$ and L consists of lists $L(v) = \{i\}$ for every $v \in V(G')$. It is straightforward to see that the two instances are equivalent. ◀

Proof of Lemma 9. In the forward direction, let G have a solution set S of size k such that $G - S$ has a list M-partition. By definition of a list M-partition, it must be the case that $v \subseteq S$. By definition of the reduced instance, the set $S' = S - \{v\}$ is a solution for (G',L',k') and (G',L',k') is a yes instance of Deletion to List M-Partition.

In the reverse direction, let G' have a solution set S' of size at most k' such that $G' - S'$ has a list M-partition. Consider the solution set $S = S' \cup \{v\}$ in G. By construction of G', the list M-partition of $G' - S'$ is a list M-partition of $G - S$. Thus, (G,L,k) is a yes instance of Deletion to List M-Partition. ◀

C Classification of matrices of order at most 2

In this section, we classify Deletion to List M-Partition for matrices M of order 1 and 2. The results are mostly consequences of results described in Section 2.

▶ **Lemma 19.** *For a 1×1 (symmetric) matrix M over $\{0,1,*\}$, the* Deletion to List M-Partition *problem is*

(a) polynomial time solvable when $M = (*)$,
(b) NP-hard but FPT for all other cases of M.

Proof. (a) Let $M = (*)$. Consider an instance $(G, L : V(G) \to \{\emptyset, \{1\}\}, k)$. Suppose $v \in V(G)$ has $L(v) = \emptyset$, then v must be deleted and k must be reduced by 1 (Lemma 9). Suppose $k < 0$ then we say no. Otherwise, after applying the above reduction exhaustively, we assume that in the reduced instance (G,L,k) every vertex v has $L(v) = \{1\}$. Then $\{V(G)\}$ is a list M-partition for G.

(b) In any other case, M has a 0 or a 1 as a diagonal entry and by Proposition 8, the Deletion to List M-Partition problem is NP-hard. Also by Proposition 6, all these cases are problems in FPT.

◀

▶ **Lemma 20.** *For a 2×2 symmetric matrix M over $\{0,1,*\}$, the* Deletion to List M-Partition *problem is*

(a) polynomial time solvable in cases when M has both diagonal entries equal to $*$,
(b) NP-hard but FPT for all other cases of M.

Proof. In each of the cases below, let (G,L,k) be an instance of Deletion to List M-Partition. We assume that for each $v \in V(G)$, we have $L(v) \neq \emptyset$ (see Lemma 9).

(a) Let M be a matrix such that both diagonal entries are equal to $*$. By symmetry of M, we have $m_{1,2} = m_{2,1}$. If $m_{1,2} = *$, then any partition $\mathcal{V} = \{V_1, V_2\}$ of $V(G)$ where $\{v \in V(G) \mid L(v) = 1\} \subseteq V_1$ and $\{v \in V(G) \mid L(v) = 2\} \subseteq V_2$ is a list M-partition of G. Next we assume that either $m_{1,2} = m_{2,1} = 0$ or $m_{1,2} = m_{2,1} = 1$. We consider the case

when $m_{1,2} = m_{2,1} = 0$. Note that from Observation 1, the case when $m_{1,2} = m_{2,1} = 1$ can be handled analogously. We now give a polynomial time algorithm for Deletion to List M-Partition for the case we are in. Let $X = \{v \in V(G) \mid L(v) = \{1\}\}$ and $Y = \{v \in V(G) \mid L(v) = \{2\}\}$. Any other vertex $w \in V(G)$ has $L(w) = \{1, 2\}$. Let S be a minimum-sized $X - Y$ separator, that can be found in polynomial time [21]. If $|S| \leq k$, then we return yes, otherwise we return no.

Now we argue the correctness of the algorithm. In one direction let S be any minimum-sized $X - Y$ separator of size at most k. Let X' be the set of vertices in $V(G - S)$ such that for each vertex $w \in X'$ there is a vertex $v \in X$ that has a path to w in $G - S$. Let $Y' = V(G - S) \setminus X'$. Note that $X \setminus S \subseteq X'$, $Y \setminus S \subseteq Y'$ and consequently $\{X', Y'\}$ is a list M-partition for $G - S$. From the above it follows that S is a solution for the Deletion to List M-Partition instance (G, L, k).

On the other hand, let the instance (G, L, k) be a yes instance and let S be a solution for Deletion to List M-Partition. We show that S is an $X - Y$ separator. Let $\{P, Q\}$ be a list M-partition for $G - S$. Then by definition, $X \setminus S \subseteq P$ and $Y \setminus S \subseteq Q$. Therefore, S is an $X - Y$ separator. Since S is of size at most k and an $X - Y$ separator, the size of minimum sized $X - Y$ separator in G is bounded by k.

(b) In any other case, M has a 0 or a 1 as a diagonal entry and by Proposition 8, the Deletion to List M-Partition problem is NP-hard. Also by Proposition 6, all these cases are problems in FPT.

◀

D Framework and Results for Our Iterative Compression Based FPT Algorithms

We define a (compression) version of the problem Deletion to List M-Partition. Later, we will argue how we can use an FPT algorithm for this version to obtain an FPT algorithm for Deletion to List M-Partition. We note that these definitions (and framework) will be used in Section E, F.1 and F.2 to design FPT algorithms for the corresponding Deletion to List M-Partition problems.

Next, we define the (compression) version of the problem, called List M-Compression.

List M-Compression	**Parameter:** k
Input: A graph G, a list function L and a $k+1$ sized vertex subset $S' \subseteq V(G)$ such that $G - S'$ has a list M-partition.	
Output: A vertex subset $S \subseteq V(G)$ of size at most k such that $G - S$ has a list M-partition.	

In the following lemma (Lemma 21), we show that if List M-Compression admits an FPT algorithm, then we can obtain an FPT algorithm for the problem Deletion to List M-Partition. Hence, to design an FPT algorithm for Deletion to List M-Partition, it will be enough to obtain an FPT algorithm for List M-Compression.

▶ **Lemma 21.** *If* List M-Compression *admits an* FPT *algorithm, when parameterized by the solution size, then* Deletion to List M-Partition *admits an* FPT *algorithm, when parameterized by the solution size.*

Proof. Suppose that List M-Compression admits an FPT algorithm, and let (G, L, k) be an input instance of Deletion to List M-Partition. We assume an (arbitrary fixed) ordering $(v_1, \ldots, v_n)$, of vertices in G. For $i \in [n]$, by W_i we denote the set $\{v_1, v_2, \cdots, v_i\}$,

and by G_i we denote the graph $G[W_i]$. By employing the method of iterative compression, we process the graphs iteratively, in the order $G_1, G_2, \cdots, G_n$. We note that if $(G_i, L|_{W_i}, k)$ is a no instance of Deletion to List M-Partition, then (G, L, k) is also a no instance of Deletion to List M-Partition. Whenever we are at iteration i, we assume that we already have computed a solution S_{i-1} to Deletion to List M-Partition for the instance $(G_{i-1}, L|_{W_{i-1}}, k)$. Notice that for $S'_i = S_{i-1} \cup \{v_i\}$, $G_i - S'_i$ admits a list M-partition. If S'_i has size at most k, then we already have a solution to Deletion to List M-Partition for the instance $(G_i, L|_{W_i}, k)$. Otherwise, we compute a solution S_i, to List M-Compression on the instance $(G_i, L|_{W_i}, S'_i)$. Observe that S_i (if it exists) is a solution to Deletion to List M-Partition for $(G_i, L|_{W_i}, k)$ as well. Moreover, if $(G_i, L|_{W_i}, S'_i)$ is a no instance of List M-Compression, then $(G_i, L|_{W_i}, k)$ (and also (G, L, k)) is a no instance of Deletion to List M-Partition as well. The above discussion implies an FPT algorithm for Deletion to List M-Partition. ◀

Next we make a general observation regarding partitions of a graph G where each part is either an independent set or a clique.

▷ Observation 2. For a $k \times k$ matrix M, let G be a graph with a list function $L : V(G) \to [k]$ such that G has a list M-partition $\mathcal{V}$. With respect to $\mathcal{V}$, let $\mathcal{V}_1$ be the collection of independent set parts and $\mathcal{V}_2$ be the collection of clique parts. For another list M-partition $\mathcal{U}$, let $\mathcal{U}_1$ be the collection of independent set parts and $\mathcal{U}_2$ be the collection of clique parts. Then $|V(\mathcal{V}_1) \cap V(\mathcal{U}_2)| \leq |\mathcal{V}_1||\mathcal{U}_2|$ and $|V(\mathcal{V}_2) \cap V(\mathcal{U}_1)| \leq |\mathcal{V}_2||\mathcal{U}_1|$.

Proof. For an independent set $I \in \mathcal{V}_1$ and a clique $C \in \mathcal{U}_2$, we have $|I \cap C| \leq 1$. Hence, follows the claim. ◀

E Missing Results and Proofs from Section 3

In this section, we complete the proof of Theorem 11 and the classical complexity dichotomy for Deletion to List M-Partition when M is of order 3.

In the following lemma, we describe some remaining cases which are NP-hard.

▶ **Lemma 22.** Deletion to List M-Partition *is* NP*-hard if M is one of the following matrices: (a) $m_{1,1} = m_{2,2} = m_{3,3} = *$, $m_{1,2} = m_{1,3} = m_{2,3} = 0$, (b) $m_{1,1} = m_{2,2} = m_{3,3} = *$, $m_{1,3} = m_{2,3} = 0, m_{1,2} = 1$.*

Proof. **(a)** In this case all diagonal entries of M are *s and all off-diagonal entries are 0s. We show that this problem is NP-hard by a reduction from the Multiway Cut problem, defined as follows.

Multiway Cut
Input: A graph G, a set of "terminals" $T \subseteq V(G)$ and a non-negative integer k.
Question: Does there exist a set of vertices $X \subseteq V(G) \setminus T$ such that $\lvert X\rvert \leq k$, and $G - X$ contains no $t - t'$ path for every distinct $t, t' \in T$?

It is known that Multiway Cut is NP-hard even when $|T| = 3$ [12]. Let an instance $(G, T = \{t_1, t_2, t_3\}, k)$ of Multiway Cut be given. We construct an instance (G', L, k') of Deletion to List M-Partition as follows. To construct G', first, add all vertices and edges of G to G'. Thus, $V(G) \subseteq V(G')$ and $E(G) \subseteq E(G')$. Now add k additional copies of each terminal to G'. That is, for each $t_i \in T$, add k new vertices $t_{i1}, t_{i2}, \ldots, t_{ik}$ to $V(G')$; and for each edge $t_i x$, add edges $t_{i1}x, t_{i2}x, \ldots, t_{ik}x$. (Thus, each t_{ij} is a "copy" of t_i.) For $i \in [3]$, let $T_i = \{t_i\} \cup \{t_{ij} \mid j \in [k]\}$. And let $T' = T_1 \cup T_2 \cup T_3$. Now define the list of each

vertex of G' as follows. For every vertex $v \in T_1$, set $L(v) = \{1\}$. For every vertex $v \in T_2$, set $L(v) = \{2\}$ and for every vertex $v \in T_3$, set $L(v) = \{3\}$. And for every vertex $v \in V(G') \setminus T'$, set $L(v) = \{1, 2, 3\}$. Finally, set $k' = k$.

Now, if S is a solution to the Multiway Cut instance (G, T, k), we show that S is also a solution to the Deletion to List M-Partition instance (G', L, k). To prove this, we construct a partition $\mathcal{V} = \{V_1, V_2, V_3\}$ of $G' - S$, and show that it is a list M-partition of $G' - S$ that respects L. Let $\mathcal{C} = \{C_1, C_2, \cdots, C_t\}$ be the connected components of $G - S$. Notice that for no distinct $i, j \in [3]$, t_i and t_j are in the same connected component in $\mathcal{C}$. Without loss of generality, we assume that $t_i \in C_i$, for $i \in [3]$. We set $V_1 = V(C_1) \cup T_1$, $V_2 = V(C_2) \cup T_2$ and $V_3 = V(G' - S) \setminus (V_1 \cup V_2)$. By the construction of $\mathcal{V}$ it is immediate that it is a list M-partition of $G' - S$ that respects L.

To see the reverse direction, suppose that S' is a solution for the Deletion to List M-Partition instance (G', L, k'). Since $|S'| \leq k$, we can assume that S' does not contain any vertex of T_i for every $i \in [3]$, as $|T_i| = k + 1$ and each vertex of T_i is a copy of the same vertex. Then, $S' \subseteq V(G) \setminus \{t_1, t_2, t_3\}$. Now, for distinct vertices $t_i, t_{i'} \in T$, $L(t_i) \cap L(t_{i'}) = \emptyset$. Therefore, t_i and $t_{i'}$ belong to different components of $G' - S'$. Hence, t_i and $t_{i'}$ belong to different components of $G - S'$ as well. That is, S' is a solution to the Multiway Cut instance (G, T, k).

(b). In this case all diagonal entries of M are *s while $m_{1,3} = m_{2,3} = 0, m_{1,2} = 1$. We show that this problem is NP-hard by a reduction from the Tripartite Vertex Cover problem, defined as follows.

> Tripartite Vertex Cover
> **Input:** A graph G with a partition of $V(G)$ into three (disjoint) independent sets Z_1, Z_2, and Z_3, and a non-negative integer k.
> **Question:** Does there exist a set of vertices $X \subseteq V(G)$ such that $|X| \leq k$ and $G - X$ is a graph with no edges?

It is a folklore result that Tripartite Vertex Cover is NP-hard (For the sake of completeness, we exhibit the NP-hardness of Tripartite Vertex Cover in Appendix A).

Let (G, Z_1, Z_2, Z_3, k) be an instance of the Tripartite Vertex Cover problem. We create an instance (G', L, k') of Deletion to List M-Partition as follows. We let $V(G') = V(G)$, $E(G') = \{uv \mid u \in Z_1, v \in Z_2, \text{ and } uv \notin E(G)\} \cup \{uv \mid u \in Z_1, v \in Z_3, \text{ and } uv \in E(G)\} \cup \{uv \mid u \in Z_2, v \in Z_3, \text{ and } uv \in E(G)\}$, and $k' = k$. Finally, for each $i \in [3]$ and $u \in Z_i$, we set $L(u) = \{i\}$.

Now we prove the correctness of our NP-hardness reduction. In the forward direction, let S be a vertex cover of size at most k in G. We show that S is a solution for the Deletion to List M-Partition instance (G', L, k'). Let $V_1 = Z_1 \setminus S$, $V_2 = Z_2 \setminus S$, and $V_3 = Z_3 \setminus S$. Firstly, note that for each $i \in [3]$, and each vertex in $u \in V_i \setminus S$, we have $L(v) = i$. Thus, it remains to argue that for each $u \in V_1$ and $v \in V_2$, we have $uv \in E(G')$, and for each $u \in V_1 \cup V_2$ and $v \in V_3$, we have $uv \notin E(G)$. First, consider the case when there are vertices $u \in V_1$ and $v \in V_2$, such that $uv \notin E(G')$. In this case, by the construction of G', we have $uv \in E(G)$, and $u, v \notin S$. This contradicts that S is a vertex cover of G. Next, consider the case when there are vertices $u \in V_1 \cup V_2$ and $v \in V_3$, such that $uv \in E(G')$. Again, by the construction of G', we have $uv \in E(G)$ and $u, v \notin E(G)$, contradicting that S is a vertex cover in G. This concludes the proof of forward direction.

In the reverse direction, let S be a solution to Deletion to List M-Partition for the instance (G', L, k'), and $\mathcal{V} = \{V_1, V_2, V_3\}$ a list M-partition of $G' - S$ that respects L, where for $i \in [3]$ and $v \in V_i$, we have $i \in L(v)$. Note that $V_1 \subseteq Z_1$, $V_2 \subseteq Z_2$, and $V_3 \subseteq Z_3$. We will show that S is solution to Tripartite Vertex Cover for the instance (G, Z_1, Z_2, Z_3, k).

Suppose not, then consider an edge $uv \in E(G - S)$. If $u \in Z_1$ and $v \in Z_2$, then $uv \notin E(G')$, $u \in V_1$, and $v \in V_2$. This contradict that V_1 and V_2 are completely adjacent. If $u \in Z_1$ and $v \in Z_3$, then $uv \in E(G')$, $u \in V_1$, and $v \in V_3$. This contradict that no vertex in V_1 is adjacent to a vertex in V_3. The case when $u \in Z_2$ and $v \in Z_3$ can be argued analogously. This concludes the proof. ◀

Proof of Lemma 13. 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 9).

(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 in a list M-partition, without violating any of the constraints.

(b) In this case, index 3 dominates both 1 and 2. So we can assume that there is no vertex $v \in V(G)$ such that $\{1, 3\} \subseteq L(v)$ or $\{1, 2\} \subseteq L(v)$. Let $X = \{v \in V(G) \mid L(v) = \{1\}\}$, $Y = \{v \in V(G) \mid L(v) = \{2\}\}$ and $Z = \{v \in V(G) \mid 3 \in L(v)\}$. Consider a solution S to the Deletion to List M-Partition instance (G, L, k). Observe the following two facts. First, if a vertex $v \in S \cap Z$, then $S \setminus \{v\}$ is also a solution. (For a list M-partition $\mathcal{V}$ of $G - (S \setminus \{v\})$, we can place v in V_3 without violating any of the constraints.) Secondly, S is an (X, Y)-separator in $G - Z$. Conversely, note that any (X, Y)-separator S' in $G - Z$ is a solution for the Deletion to List M-Partition instance (G, L, k). In particular, note that as there is no path in $G - (Z \cup S')$ from $X \setminus S'$ to $Y \setminus S'$, every vertex $v \in V(G) - S'$ with $L(v) = \{1, 2\}$ is reachable from at most one of the sets $X \setminus S'$ or $Y \setminus S'$. Therefore, we can obtain a list M-partition $\mathcal{V} = \{V_1, V_2, V_3\}$ of $G - S'$ as follows: for $v \in V(G) \setminus S'$, place v in V_i if $L(v) = \{i\}$, and if $L(v) = \{1, 2\}$, then place v in V_1 if v is reachable from $X \setminus S'$, and place v in V_2 otherwise. Therefore, in order to solve Deletion to List M-Partition on (G, L, k), all we need to do is find the minimum (X, Y)-separator in $G - Z$, which can be done in polynomial time.

(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. Now consider the sets $X = \{v \in V(G) \mid L(v) = \{3\}\}$ 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. A minimum sized (X, Y)-separator can be found in polynomial time [21]. ◀

Next, we will prove Observation 3 which will cover the cases in Theorem 11 when Deletion to List M-Partition is para-NP-hard. This observation is based on the following fact. For an $\ell \times \ell$ matrix M, testing whether a given graph G admits an M-partition (i.e., the recognition version) is the same as solving Deletion to List M-Partition (i.e., the deletion version) with parameter $k = 0$, and $L(v) = [\ell]$ for $v \in V(G)$. Therefore, for a matrix M, if the recognition version itself is NP-hard, then Deletion to List M-Partition is para-NP-hard. From the above discussions, we immediately obtain the following observation.

▷ Observation 3. The problem Deletion to List M-Partition is para-NP-hard, when M is one of the following matrices.

1. $m_{1,1} = m_{2,2} = m_{3,3} = 0$ and $m_{1,2} = m_{1,3} = m_{2,3} = *$, or
2. $m_{1,1} = m_{2,2} = *$, $m_{3,3} = 0$, $m_{1,3} = m_{2,3} = *$ and $m_{1,2} = 0$.

Proof of Observation 3. The proof of item 1 follows from the fact that such an M-partition corresponds to a 3-colouring. Therefore, List M-Partition corresponds to the 3-Colouring problem, which is NP-hard [22]. Similarly, the proof of item 2 follows from the fact that such an M-partition corresponds to a stable cutset partition. Therefore, List M-Partition corresponds to the Stable Cutset problem, which is NP-hard [4]. ◀

Proof of Lemma 17. We prove Lemma 17 by proving that Algorithm 3.2 is correct. And to prove that, it is enough to prove the following. Let S be an optimal solution to the list partition problem, and $S' \subseteq S$ be a minimal (X, Y)-separator, where X and Y are as defined in Step 1 of the algorithm. 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. ◀

Algorithm for the Deletion to List $(2, 1)$-Graph**.** We now design an FPT algorithm for the Deletion to List M-Partition problem, where M is the (symmetric) matrix such that $m_{1,1} = m_{2,2} = 0, m_{3,3} = 1$ and $m_{1,2} = m_{1,3} = m_{2,3} = *$. We note that the List M-Partition for this case is same as the recognition problem of (r, ℓ)-graphs, for $r = 2, \ell = 1$. The family of (r, ℓ)-graphs was introduced by Brandstadt et al. [5]. A graph that has a list M-partition will be referred to as a list $(2, 1)$-graph. A list M-partition will also be referred to as a list $(2, 1)$-partition. An FPT algorithm for Deletion to $(2, 1)$-Graph, the non-list version of the problem, was given in [25]. The algorithm for Deletion to List $(2, 1)$-Graph is very similar, which is presented below.

▶ **Lemma 23.** Deletion to List $(2, 1)$-Graph *admits an* FPT *algorithm.*

We use iterative compression and the FPT algorithm for Deletion to 2-List M-Partition to obtain an algorithm for Deletion to List $(2, 1)$-Graph. Note that from Lemma 21 it is enough to design an FPT algorithm for the compression version of the problem. We call the compression version of Deletion to List $(2, 1)$-Graph as the List $(2, 1)$-Compression problem (see Section D). The following lemma shows that List $(2, 1)$-Compression is in FPT.

▶ **Lemma 24.** List $(2, 1)$-Compression *admits an* FPT *algorithm.*

Proof. We design an algorithm for List $(2, 1)$-Compression. Let (G, L, S') be an instance of the problem. We compute a list $(2, 1)$-partition $\mathcal{V} = \{V_1, V_2, V_3\}$ of $G - S'$ using the algorithm in [5]. We note that the algorithm of Brandstadt et al. [5] can compute all $(2, 1)$-partitions for the given graph in polynomial time, and since $G - S'$ admits a list $(2, 1)$-partition, one of the partitions returned by the algorithm in [5] is a valid list $(2, 1)$-partition of $G - S'$.

Let $\mathcal{V}_1 = \{V_1, V_2\}$ be the independent parts and $\mathcal{V}_2 = \{V_3\}$ be the clique part. Let S be a *hypothetical solution* of size at most k for the problem, which the algorithm is supposed to compute. Also let $\mathcal{U}$ be the list $(2,1)$-partition of $G - S$, where $\mathcal{U}_1 = \{U_1, U_2\}$ are the independent parts and $\mathcal{U}_2 = \{U_3\}$ is the clique part. The algorithm first guesses a partition (Y, N) of S' such that $Y = S' \cap S$ and $N = S' \setminus S$. Note that since N is not part of the solution S, $G[N]$ has a list $(2,1)$-partition. There are at most $3^{|N|} \leq 3^k$ list $(2,1)$-partitions of $G[N]$ and we guess a partition (and we try all possible partitions). After this guess, for each $v \in N$, we only keep in $L(v)$ the index of the part that v belongs to in the current guessed list $(2,1)$-partition of $G[N]$.

Next, consider the sets $V_C = (V(\mathcal{V}_1) \cap V(\mathcal{U}_2)) \setminus (S \cup S')$ and $V_I = (V(\mathcal{V}_2) \cap V(\mathcal{U}_1)) \setminus (S \cup S')$. Each set has size at most 2 and we guess the vertices in V_C and V_I (see Observation 2). After the guess of V_I and V_C, any vertex in $V(\mathcal{V}_1) \setminus V_C$ either belongs to $V(\mathcal{U}_1)$ or to the solution S. Consider a vertex $v \in V(\mathcal{V}_1) \setminus V_C$. We can safely delete the index 3 from $L(v)$, thereby reducing the size of its list to at most 2. Similarly, any vertex in $V(\mathcal{V}_2) \setminus V_I$ either belongs to $V(\mathcal{U}_2)$ or to the solution. Consider a vertex $v \in V(\mathcal{V}_2) \setminus V_I$. We can safely delete the indices 1 and 2 from $L(v)$, thereby reducing the size of its list to at most 2. Thus, we have reduced our instance to an instance of List $(2,1)$-Compression. Since there are at most $4^k n^4$ guesses, we have at most $4^k n^4$ many instances of List $(2,1)$-Compression such that our original instance is a yes instance of List $(2,1)$-Compression if and only if there is one of the created instances of Deletion to 2-List M-Partition is a yes instance. By Proposition 6, List $(2,1)$-Compression is in FPT. ◀

Equivalently, FPT algorithms can be obtained for Deletion to List $(1,2)$-Graph. Here, the problem Deletion to List $(1,2)$-Graph is the Deletion to List M-Partition problem when M is the (symmetric) matrix such that $m_{1,1} = 0, m_{2,2} = m_{3,3} = 1$ and $m_{1,2} = m_{1,3} = m_{2,3} = *$.

▶ **Corollary 25.** Deletion to List $(1,2)$-Graph *admits an* FPT *algorithm.*

Finally, we describe the full proof of Theorem 11.

Proof of Theorem 11. The proof of item 1, item 2 and item 3 of the theorem statement follows from Lemma 13, Observation 3 and Lemma 14, 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 each class separately.

Case 1: At most one off-diagonal entry in M_U is 0. If a row of M has both a 0 and a 1, then by Proposition 7, the problem is fixed-parameter tractable. Therefore, we can assume that no row has both a 0 and 1. Since we are under the assumption that at most one off-diagonal entry of M_U is 0, by complementation, we can assume that at most one off-diagonal entry of M_U is 1 as well. (Notice that if at least two off-diagonal entries of M_U were 1, then at least two off-diagonal entries of $\overline{M}_U$ would be 0, and we will consider this in our later cases.) Again, because M is a symmetric matrix, observe that whenever there is exactly one 0 and exactly one 1 as off-diagonal entries of M_U then there is a row containing both a 0 and a 1. In such a case, we apply Proposition 7. Therefore, we are left with the case when M_U has two off-diagonal entries as $*$s and the other off-diagonal entry is from $\{0, 1, *\}$. We assume for rest of the sub-cases that $m_{1,3} = m_{2,3} = *$. (All other cases of M such that

M_U has two off-diagonal entries as $*$s and the other off-diagonal entry is from $\{0, 1, *\}$ can be symmetrically argued.)

Case 1.1. Consider the case when $m_{1,3} = m_{2,3} = *$ and $m_{3,3} = *$, then index 3 dominates 1 and 2 (since M is symmetric). Hence, if a list contains both 1 and 3, remove 1; and if a list contains both 2 and 3, remove 2. (This is safe, by Proposition 5.) Now every list has size at most two, and by Proposition 6, the problem is fixed-parameter tractable. Therefore, for the rest of the analysis of this case, we assume that $m_{3,3} \in \{0, 1\}$.

Case 1.2. Consider the case when $m_{1,1}, m_{2,2}, m_{3,3} \in \{0, 1\}$, but not all three diagonal entries are of the same value. (Recall that $m_{1,3} = m_{2,3} = *, m_{3,3} \in \{0, 1\}$.) Suppose $m_{1,2} = 1$. Then, if $m_{1,1} = 0$ or $m_{2,2} = 0$, then we have a row with both a 0 and a 1. And by Proposition 7, the problem is fixed parameter tractable. So assume $m_{1,1} = m_{2,2} = 1$. Then, 1 and 2 dominate each other. Therefore, we can assume that no list contains both 1 and 2. By Propositions 5 and 6, the problem is fixed-parameter tractable. By complementation, the case when $m_{1,2} = 0$ is also fixed-parameter tractable. If $m_{1,2} = *$, then note that in the current case of $m_{1,1}, m_{2,2}, m_{3,3} \in \{0, 1\}$ where all of them are not equal, two of $m_{1,1}, m_{2,2}, m_{3,3}$ are equal by pigeonhole principle. This case is dealt with using Lemma 23 or Corollary 25.

Case 1.3. We further consider sub-cases based on the entry $m_{1,2} \in \{0, 1, *\}$. (Again, we have $m_{1,3} = m_{2,3} = *, m_{3,3} \in \{0, 1\}$.)

Case 1.3.1. First, suppose that $m_{1,2} = *$. Then, if $m_{1,1} = *$, then 1 dominates 2. If $m_{2,2} = *$, then index 2 dominates 1. In either case, by Propositions 5 and 6 the problem is fixed-parameter tractable. Therefore, assume that $m_{1,2} = *$ and $m_{1,1}, m_{2,2} \neq *$. Then, $m_{1,1}, m_{2,2} \in \{0, 1\}$. If $m_{1,1} = m_{2,2} = m_{3,3}$, then M is the matrix corresponding to 3-Colouring, which is not possible. Otherwise not all three of $m_{1,1}, m_{2,2}, m_{3,3}$ are the same. But we already dealt with this case in the preceding paragraph.

Case 1.3.2. Now, assume that $m_{1,2} = 0$. If $m_{1,1}$ or $m_{2,2}$ is 1, then we have a row with both a 0 and a 1. And by Proposition 7, the problem is fixed-parameter tractable. Thus $m_{1,1}, m_{2,2} \in \{0, *\}$. If $m_{1,1} = m_{2,2} = 0$, then indices 1 and 2 dominate each other. If $m_{1,1} = *$ and $m_{2,2} = 0$, then index 1 dominates 2. If $m_{1,1} = 0$ and $m_{2,2} = *$, then index 2 dominates 1. In all the cases, by Propositions 5 and 6 the problem is fixed-parameter tractable.

So assume that $m_{1,1} = m_{2,2} = *$. Recall that we are under the assumption that $m_{3,3} \in \{0, 1\}$. If $m_{3,3} = 0$, then an M-partition is a stable cutset partition, and if $m_{3,3} = 1$, then an M-partition is a clique cutset partition, neither of which is possible (see item 2 and 3 of the theorem statement).

To end our argument, note that a matrix $\overline{M}$ with $\overline{m}_{1,2} = 0, \overline{m}_{1,3} = \overline{m}_{2,3} = *$ is the complement of a matrix M with $m_{1,2} = 1, m_{1,3} = m_{2,3} = *$. Thus if all cases where $m_{1,2} = 0, m_{1,3} = m_{2,3} = *$ are covered, then so are all cases where $m_{1,2} = 1, m_{1,3} = m_{2,3} = *$.

Case 2: 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. Then, by Proposition 7, the problem is fixed-parameter tractable. So assume that $m_{1,1}, m_{2,2}, m_{3,3} \in \{0, *\}$. Now, we will consider each possibility of $m_{1,2}$ for our analysis. Note first that if $m_{1,2} = 1$, then M has a row containing both a 0 and a 1, and by Proposition 7, the problem is fixed-parameter tractable. So, assume that $m_{1,2} = *$.

Case 2.1. 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 5 and 6.

Case 2.2. So assume that $m_{1,1} = m_{2,2} = 0$. If $m_{3,3} = 0$, then index 1 dominates 2, and by Propositions 5 and 6 the problem is again fixed-parameter tractable.

If $m_{3,3} = *$, then an M-partition is a bipartite-star partition, and by Lemma 17, the problem is fixed-parameter tractable.

Case 3: All off-diagonal entries of M_U are 0s. If any one of $m_{1,1}, m_{2,2}$ or $m_{3,3}$ is 1, then M has a row containing both a 0 and a 1. And by Proposition 7, the problem is fixed-parameter tractable. So assume that $m_{1,1}, m_{2,2}, m_{3,3} \in \{0, *\}$.

Case 3.1. If $m_{1,1} = 0$ and $m_{2,2} = *$, then index 2 dominates 1. If $m_{1,1} = *$ and $m_{2,2} = 0$, then index 1 dominates 2. In either case, the problem is fixed-parameter tractable, by Propositions 5 and 6.

Case 3.2. So assume that $m_{1,1} = m_{2,2}$. If $m_{1,1} = m_{2,2} = 0$, then index 1 dominates 2. If $m_{1,1} = m_{2,2} = *$ and $m_{3,3} = 0$, then index 1 dominates 3. In either case, by Propositions 5 and 6 the problem is fixed-parameter tractable.

So assume that $m_{1,1} = m_{2,2} = m_{3,3} = *$. Then M is the three-stars matrix, and by Lemma 17, the problem is fixed parameter tractable. ◀

F Classification of 4×4 matrices where no diagonal entry is a $*$

In this section, we obtain a complete dichotomy result on the parameterized complexity of Deletion to List M-Partition, when M is a 4×4 matrix whose all diagonal entries belong to the set $\{0, 1\}$. To this end, we start by defining the following matrix, which we call the 3-Colouring matrix.

$$M_{\mathsf{col}} = \begin{pmatrix} 0 & * & * \\ * & 0 & * \\ * & * & 0 \end{pmatrix}$$

We will establish our dichotomy result by proving the following theorem.

▶ **Theorem 26.** *Consider a* Deletion to List M-Partition *problem, where the matrix M has only* 0*s and* 1*s 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.*

We note that, the NP-hardness of all Deletion to List M-Partition problems, with the added assumption that no diagonal entry of M is a $*$, follow from Lewis and Yannakakis' result [29] on vertex deletion to non-trivial, hereditary graph classes.

As in the case of 3×3 matrices, we solve a number of these problems by reducing the given instance to Deletion to 2-List M-Partition. But there are several cases in which this fails, and we are forced to apply different techniques, using structural properties of the matrix M.

In the following lemma, we show that when M contains an equivalent matrix of M_{col} (or its complement) as a sub-matrix, then the problem is the problem is para-NP-hard.

▶ **Lemma 27.** *Consider a* Deletion to List M-Partition *problem, where the matrix M has only* 0*s and* 1*s as diagonal entries and M contains an equivalent matrix of M_{col} (or its complement) as a sub-matrix. Then the* Deletion to List M-Partition *problem is para-*NP*-hard.*

Proof. To prove the para-NP-hardness result, notice that it is enough to show that for $k = 0$, the problem is NP-hard. And when $k = 0$, the Deletion to List M-Partition problem

is same as the List M-Partition problem. If M contains a 3×3 submatrix which is equivalent to the complement of M_{col}, then $\overline{M}$ contains a 3×3 submatrix which is equivalent to M_{col}. Thus, from Observation 1 it is enough to consider the case when M has a 3×3 submatrix M' which is equivalent to M_{col}. Let $\{i_1, i_2, i_3\} \subseteq [4]$ be the submatrix index of M_s for M, where $i_1 \leq i_2 \leq i_3$. If for $c \in [3]$, $m'_{c,c} = *$, then $m_{i_c,i_c} = *$, contradicting that M has all diagonal entries from $\{0, 1\}$. Thus, we have that $m'_{i,i} = 0$, for each $i \in [3]$. The above implies that $M' = M_{\mathsf{col}}$, i.e. the bijection-witness is the identity function.

In the following we will establish the para-NP-hardness of Deletion to List M-Partition. Let G be an instance of 3-Colouring. We create an instance (G', L, k) of Deletion to List M-Partition as follows. We set $G' = G$ and $k = 0$. For each $v \in V(G)$, we set $L(v) = \{i_1, i_2, i_3\}$. Now notice that G is a yes instance of 3-Colouring if and only if (G', L, k) is a yes instance of Deletion to List M-Partition. This completes the proof. ◀

In the following, we state a result regarding Deletion to List M-Partition, for a specific matrix. We note that this result follows from a result of Chitnis et al. [9].

▷ Observation 4 ([9]). Deletion to List $\hat{M}$-partition (also called 2-Disconnected Bipartition or 2-DB) is fixed parameter tractable parameterized by the solution size, when $\hat{M}$ (called the 2-DB matrix) is the following matrix.

$$M_{\mathsf{2db}} = \begin{pmatrix} 0 & * & 0 & 0 \\ * & 0 & 0 & 0 \\ 0 & 0 & 0 & * \\ 0 & 0 & * & 0 \end{pmatrix}$$

Hereafter, we will only consider those 4×4 matrices which are not equivalent to the matrix (or its complement) in Observation 4. For integers $i, j \in \{0, 1, 2, 3, 4\}$, such that $i + j = 4$, whenever we write a matrix subscripted by (the ordered pair) ij, for example M_{ij}, then we will mean that there are exactly i and j diagonal entries in M_{ij} that are 0s and 1s, respectively. We will prove our result by considering cases on the types of diagonal entries the matrix M has. In Section F.1, Section F.2 and Section F.3 we give FPT algorithms for Deletion to List M_{22}-Partition, Deletion to List M_{31}-Partition, Deletion to List M_{40}-Partition, respectively.

In the rest of this section, we will give the proof of Theorem 26, assuming the results of Section F.1 to F.3.

Proof of Theorem 26. Consider a matrix M that has only 0s and 1s as diagonal entries. Also assume that the matrix M is not equivalent to M_{2db} (or its complement). First, if the matrix contains an equivalent matrix of M_{col} (or its complement) as a sub-matrix, then from Lemma 27, the problem is para-NP-hard. Therefore, we are in the case where M does not contain an equivalent matrix of M_{col} or M_{2db} (or their complements) as a sub-matrix. Then the following cases arise:

- M has two diagonal entries as 0 and two diagonal entries as 1. Then the statement of the theorem follows from Lemma 28 (Section F.1).
- M has three diagonal entries as 0 and one diagonal entry as 1. Then the statement of the theorem follows from Lemma 32 (Section F.2).
- M has three diagonal entries as 1 and one diagonal entry as 0. Let $\overline{M}$ be the complement of M. If a graph G realizes a list $\overline{M}$-partition, then the graph $\overline{G}$ will realize a list

M-partition. Thus, by considering the complement of the input graph, the statement of the theorem follows from Lemma 32 (Section F.2).

- If M has all diagonal entries as 0, then the statement of the theorem follows from Lemma 33. On the other hand, if M has all diagonal entries as 1, then by taking the complement of the input graph G and the complement of the matrix M, we obtain the statement of the theorem from Lemma 33 (Section F.3).

This completes the proof. ◀

F.1 FPT Algorithm for Deletion to List M_{22}-Partition

In this section, we design an FPT algorithm for the problem Deletion to List M_{22}-Partition. The focus of this section will be to design FPT algorithm for List M_{22}-Compression. And this together with Lemma 21 will imply that Deletion to List M_{22}-Partition admits an FPT algorithm.

To design our FPT algorithm for List M_{22}-Compression, we will design a polynomial time algorithm for List M_{22}-Partition. Firstly, we give an algorithm for List M_{22}-Compression assuming a polynomial time algorithm for List M_{22}-Partition. Then, we design a polynomial time algorithm for the List M_{22}-Partition problem.

In the following lemma, we show that List M_{22}-Compression is FPT.

▶ **Lemma 28.** List M_{22}-Compression *admits an* FPT *algorithm.*

Proof. Let (G, L, S') be the instance of the problem. Then $G - S'$ has a list M_{22}-partition. With respect to this partition, let $\mathcal{V}_1$ be the collection of parts that are independent sets, and $\mathcal{V}_2$ be the collection of parts that are cliques. Let S be a *hypothetical solution* of size k for the problem, which the algorithm is supposed to compute. Again, $G - S$ has a list M_{22}-partition. Let $\mathcal{U}_1$ be the collection of parts that are independent sets, and $\mathcal{U}_2$ be the collection of parts that are cliques. The algorithm first guesses a partition (Y, N) of S' such that $Y = S' \cap S$ and $N = S' \setminus S$. After this guess, the objective is to compute a set Z of size at most $k' = k - |Y|$ such that $G - (Z \cup Y)$ has a list M_{22}-partition. Also note that since N is not part of the solution S, $G[N]$ must have a list M_{22}-partition. Consider the restricted pairs $(\mathcal{V}_1 - (S \cup S'), \mathcal{V}_2 - (S \cup S'))$ and $(\mathcal{U}_1 - (S \cup S'), \mathcal{U}_2 - (S \cup S'))$ of $G - (S \cup S')$. By Observation 2 we know that the cardinalities of each of the sets $(V(\mathcal{V}_1) \cap V(\mathcal{U}_2)) \setminus (S \cup S')$ and $(V(\mathcal{V}_2) \cap V(\mathcal{U}_1)) \setminus (S \cup S')$ are bounded by 4. So now the algorithm guesses the set $V_C = (V(\mathcal{V}_1) \cap V(\mathcal{U}_2)) \setminus (S \cup S')$ and $V_I = (V(\mathcal{V}_2) \cap V(\mathcal{U}_1)) \setminus (S \cup S')$, each of them having size at most 4. After the guess of V_I and V_C, any vertex in $V(\mathcal{V}_2) \setminus V_I$ either belongs to $V(\mathcal{U}_2)$ or belongs to the hypothetical solution S. Similarly any vertex in $V(\mathcal{V}_1) \setminus V_C$ either belongs to $V(\mathcal{U}_1)$ or belongs to the hypothetical solution S. There are at most $4^{|N|} \leq 4^{k+1}$ many list M_{22}-partitions of $G[N]$. The algorithm now guesses a list M_{22}-partition of $G[N]$ that is the restriction of the final M_{22}-partition of $G - S$, to $G[N]$. Since the list M_{22}-partition is fixed, for each vertex $v \in N$, if it is guessed to belong to part i in the fixed list M_{22}-partition of $G[N]$ and if $i \notin L(v)$ then we discard the guess. Otherwise, we delete from $L(v)$ all indices but i. Consider vertices in $(V(\mathcal{V}_1) \setminus V_C) \cup V_I$. By the above arguments, these vertices will only belong to independent set parts in the final list M_{22}-partition of $G - S$, or they can be deleted. Therefore, for any such vertex v, we can delete from $L(v)$ the indices of all the clique parts. In case v ends up with an empty list, then v must be deleted (Lemma 9). If $k = 0$ then we do not have the budget to delete v and therefore we discard the current guess. Otherwise, we delete v and reduce the parameter k by 1 (Lemma 9). After this reduction all these vertices have lists of size at most 2. Similarly, we can reduce the sizes of lists of vertices

in $(V(\mathcal{V}_2) \setminus V_I) \cup V_C$ to 2, or either delete a vertex and decrement the parameter k or discard the current guess. Therefore, on fixing guesses for V_C, V_I and a list M_{22}-partition of $G[N]$ we reduce our List M_{22}-Compression problem to a Deletion to 2-List M_{22}-Partition problem, which by Proposition 6 is FPT. Since there are polynomially many guesses that are being made, the List M_{22}-Compression problem is FPT.

Finally, we analyse the correctness of the algorithm. In one direction, suppose for a reduced instance (G', L', k') we obtain a k'-sized solution set S^* such that $G' - S^*$ list M_{22}-partition $\mathcal{U}'$. Note that G' is an induced subgraph of G and for each vertex $u \in V(G')$, $L'(u) = L(u)$. Consider the set $S = S^* \cup \{V(G) \setminus V(G')\}$. By construction of the reduced instances, the size of S is at most k. Therefore, $\mathcal{U}'$ is a list M_{22}-partition of G. On the other hand, suppose that for the original instance (G, L, S', k) of List M_{22}-Compression there is a k-sized solution S such that $G - S$ has a list M_{22}-partition $\mathcal{U}$. Recall that $\mathcal{V}$ is a list M_{22}-partition of $G - S'$. Let $Y = S' \cap S$, $N = S' \setminus S$, $V_C = (V(\mathcal{V}_1) \cap V(\mathcal{U}_2)) \setminus (S \cup S')$ and $V_I = (V(\mathcal{V}_2) \cap V(\mathcal{U}_1)) \setminus (S \cup S')$ and let $\mathcal{U}_N$ be the list M_{22}-partition of $G[N]$ when $\mathcal{U}$ is restricted to $G[N]$. Let $k' = k - |Y|$. For each vertex $u \in N$ let $L'(u)$ be the sublist of $L(u)$ that contains only the index of the part in $\mathcal{U}_N$ to which it belongs. For each vertex $u \in (V(\mathcal{V}_1) \setminus V_C) \cup V_I$, let $L'(u)$ be the sublist of $L(u)$ that contains only the indices of the independent set parts. Similarly, for each vertex $u \in (V(\mathcal{V}_2) \setminus V_I) \cup V_C$, let $L'(u)$ be the sublist of $L(u)$ that contains only the indices of the clique parts. Then for the corresponding set of guesses our algorithm is going to output yes for the reduced instance $(G - Y, L', k')$ of 2-List M-Partition. This concludes the proof of correctness of our algorithm. ◀

In the following lemma, we give a polynomial time algorithm for List M_{22}-Partition.

▶ **Lemma 29.** *The* List M_{22}-Partition *problem admits a polynomial time algorithm.*

Proof. We will describe an iterative algorithm. Let (G, L) be the input of the List M_{22}-Partition problem, and let $V(G) = \{v_1, \ldots, v_n\}$. For every $i \in [n]$, we define the subgraph $G_i = G[\{v_1, \ldots, v_i\}]$. We iterate through the instances $I_i = (G_i, L)$ starting with $i = 5$. When we consider the ith instance and a known list M_{22}-partition for the instance I_{i-1}, our objective is to obtain a list M_{22}-partition of the instance I_i. We formally define this iteration problem as follows.

List M_{22}-Iteration
Input: A graph G, a list function L, a vertex $v \in V(G)$ and a list M_{22}-partition of $G - \{v\}$.
Output: A list M_{22}-partition of G.

We reduce the List M_{22}-Partition problem to $n - 4$ instances of the List M_{22}-Iteration problem in the following manner. Let (G, L) be an instance of List M_{22}-Partition, where $V(G) = \{v_1, v_2, \ldots, v_n\}$. For $i \in [n]$, we let $X_i = \{v_j \mid j \in [i]\}$. When $i \leq 4$, there is a trivial brute-force way of finding a list M_{22}-partition $\mathcal{V}_i$ for G_i. Let $I_i = (G_i, L|_{X_i}, v_i, \mathcal{V}_{i-1})$ be the i^{th} instance of List M_{22}-Iteration. Hence, we start the iteration with the instance $I_5 = (G_5, L|_{X_i}, v_5, \mathcal{V}_4)$ and try to obtain a list M_{22}-partition for G_i. If such a partition $\mathcal{V}_i$ exists, we go to the next iteration. If during any iteration, the corresponding instance does not have a list M_{22}-partition, it implies that the original instance (G, L) is a no instance for List M_{22}-Partition. If the input instance (G, L) is a yes instance, then $\mathcal{V}_n$ is a list M_{22}-partition for G, where $n = |V(G)|$. Since there are at most n iterations, the total time taken by the algorithm to solve List M_{22}-Partition is at most n times the time taken to solve List M_{22}-Iteration.

The following claim shows that List M_{22}-Iteration can be solved in polynomial time.

▷ Claim 30. List M_{22}-Iteration can be solved in polynomial time.

Proof. Let $(G, L, v, \mathcal{V})$ be the instance of the problem. Then $\mathcal{V}$ is a list M_{22}-partition of $G - \{v\}$. With respect to this partition, let $\mathcal{V}_1$ be the collection of parts that are independent sets, and $\mathcal{V}_2$ be the collection of parts that are cliques. Let $\mathcal{U}$ be a hypothetical list M_{22}-partition for G. Let $\mathcal{U}_1$ be the collection of parts that are independent sets, and $\mathcal{U}_2$ be the collection of parts that are cliques. The algorithm first guesses to which part of $\mathcal{U}$ the vertex v will belong: If the index of that part is not in $L(v)$ then we discard the guess, and otherwise we delete all other indices in the list $L(v)$ and keep only the index of that part. After this guess, consider the restricted pairs $(\mathcal{V}_1, \mathcal{V}_2)$ and $(\mathcal{U}_1 - \{v\}, \mathcal{U}_2 - \{v\})$ of $G - v$. By Observation 2 we know that the cardinality of each of the set $(V(\mathcal{V}_1) \cap V(\mathcal{U}_2)) \setminus \{v\}$ and $(V(\mathcal{V}_2) \cap V(\mathcal{U}_1)) \setminus \{v\}$ are bounded by 4. So now the algorithm guesses the set $V_C = (V(\mathcal{V}_1) \cap V(\mathcal{U}_2)) \setminus \{v\}$ and $V_I = (V(\mathcal{V}_2) \cap V(\mathcal{U}_1)) \setminus \{v\}$, each of them having size at most 4. We also guess to which part of $\mathcal{U}$ the vertices of $V_I \cup V_C$ will belong, and subsequently reduce the list sizes to 1 or reject the current guess. After the guess of V_I and V_C, any vertex in $V(\mathcal{V}_2) - V_I$ belongs to $V(\mathcal{U}_2)$. Similarly any vertex in $V(\mathcal{V}_1) - V_C$ belongs to $V(\mathcal{U}_1)$. Consider vertices in $(V(\mathcal{V}_1) - V_C)$. By the above arguments, these vertices will only belong to independent set parts in the final list M_{22}-partition of $G - S$. Therefore, for such a vertex v, we delete all indices corresponding to independent parts and only keep the indices of the clique parts. In the process, if a vertex has an empty list, then we reject the current guess. After this reduction all these vertices have lists of size at most 2. Similarly, we can reduce the sizes of lists of vertices in $(V(\mathcal{V}_2) - V_I)$ to 2, or reject the current guess due to resulting empty lists. Therefore, on fixing guesses for V_C, V_I we reduce our List M_{22}-Iteration problem to a 2-List M_{22}-Partition, which as stated in Proposition 2, is a special case of 2-SAT. Moreover, due to self-reducibility of 2-SAT, a satisfying solution can be obtained. This results in an M_{22}-partition of G [20].

Finally, we analyse the correctness of the algorithm. In one direction, suppose we obtain a list M_{22}-partition $\mathcal{U}$ for G in one of our reduced problems. Notice that the reduced problems are of the form (G, L') where for each $w \in V(G)$, $L'(w) \subseteq L(w)$. Therefore, this is also a list M_{22}-partition of (G, L). On the other hand, suppose there is a list M_{22}-partition $\mathcal{U}$ for the original instance (G, L). Recall that $\mathcal{V}$ is a list M_{22}-partition of $G \setminus \{v\}$. Let $V_C = (V(\mathcal{V}_1) \cap V(\mathcal{U}_2)) \setminus \{v\}$ and $V_I = (V(\mathcal{V}_2) \cap V(\mathcal{U}_1)) \setminus \{v\}$ and let v belong to part i in $\mathcal{U}$. Then for the corresponding set of guesses our algorithm is going to output yes for the corresponding reduced instance of 2-List M_{22}-Partition. This concludes the proof of correctness of our algorithm. ◀

◀

F.2 FPT Algorithm for Deletion to List M_{31}-Partition

In this section, we consider the problem Deletion to List M_{31}-Partition, which is the Deletion to List M-Partition problem, when M has three diagonal entries as 0 and one diagonal entry as 1. Again, we will focus on designing an FPT algorithm for List M_{31}-Compression. And this together with Lemma 21 will imply that Deletion to List M_{31}-Partition admits an FPT algorithm.

Similar to the Section F.2, to design our FPT algorithm for List M_{31}-Compression, we will design a polynomial time algorithm for List M_{31}-Partition. Then, we give an algorithm for List M_{31}-Compression using the polynomial time algorithm for List M_{31}-Partition.

▶ **Lemma 31.** *Consider a* List M_{31}-Partition *problem, where the input matrix does not contain the* 3-Colouring *matrix as a sub-matrix. Then there is a polynomial time algorithm that finds a required partition.*

Matrix		Class	Proof
	Contains the 3-Coloring matrix as a sub matrix	para-NP-hard	Lemma 27 (Also see Lemma 32-Case 2)
	$\begin{pmatrix}0&1&-&-\\1&0&-&-\\-&-&0&-\\-&-&-&1\end{pmatrix}, \begin{pmatrix}0&-&1&-\\-&0&-&-\\1&-&0&-\\-&-&-&1\end{pmatrix}, \begin{pmatrix}0&-&-&-\\-&0&1&-\\-&1&0&-\\-&-&-&1\end{pmatrix}$	FPT	Lemma 32-Case 1
	$\begin{pmatrix}0&0&0&*\\0&0&0&-\\0&0&0&-\\*&-&-&1\end{pmatrix}, \begin{pmatrix}0&0&0&1\\0&0&0&1\\0&0&0&0/1\\1&1&0/1&1\end{pmatrix}, \begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0/1\\0&0&0/1&1\end{pmatrix}$	FPT	Lemma 32-Case 3
	$\begin{pmatrix}0&*&0&*\\*&0&0&-\\0&0&0&-\\*&-&-&1\end{pmatrix}, \begin{pmatrix}0&*&0&-\\*&0&0&*\\0&0&0&-\\-&*&-&1\end{pmatrix}$	FPT	Lemma 32-Case 4.1
$\begin{pmatrix}0&*&0&-\\*&0&0&-\\0&0&0&*\\-&-&*&1\end{pmatrix}$	$\begin{pmatrix}0&*&0&1\\*&0&0&0\\0&0&0&*\\1&0&*&1\end{pmatrix}$	FPT	Lemma 32-Case 4.2.1
	$\begin{pmatrix}0&*&0&0\\*&0&0&0\\0&0&0&*\\0&0&*&1\end{pmatrix}$	FPT	Lemma 32-Case 4.2.2
	$\begin{pmatrix}0&*&0&1\\*&0&0&1\\0&0&0&*\\1&1&*&1\end{pmatrix}$	FPT	Lemma 32-Case 4.2.3
$\begin{pmatrix}0&*&0&0/1\\*&0&0&0/1\\0&0&0&0/1\\0/1&0/1&0/1&1\end{pmatrix}$	$\begin{pmatrix}0&*&0&0\\*&0&0&0/1\\0&0&0&0\\0&0/1&0&1\end{pmatrix}, \begin{pmatrix}0&*&0&1\\*&0&0&0/1\\0&0&0&1\\1&0/1&1&1\end{pmatrix}, \begin{pmatrix}0&*&0&0/1\\*&0&0&0\\0&0&0&0\\0/1&0&0&1\end{pmatrix}, \begin{pmatrix}0&*&0&0/1\\*&0&0&1\\0&0&0&1\\0/1&1&1&1\end{pmatrix}$	FPT	Lemma 32-Case 4.3.1
	$\begin{pmatrix}0&*&0&1\\*&0&0&1\\0&0&0&0\\1&1&0&1\end{pmatrix}$	FPT	Lemma 32-Case 4.3.2
	$\begin{pmatrix}0&*&0&0\\*&0&0&0\\0&0&0&1\\0&0&1&1\end{pmatrix}$	FPT	Lemma 32-Case 4.3.3
$\begin{pmatrix}0&*&0&-\\*&0&*&-\\0&*&0&-\\-&-&-&1\end{pmatrix}$	$\begin{pmatrix}0&*&0&*\\*&0&*&-\\0&*&0&-\\*&-&-&1\end{pmatrix}, \begin{pmatrix}0&*&0&-\\*&0&*&-\\0&*&0&*\\-&-&*&1\end{pmatrix}, \begin{pmatrix}0&*&0&=\\*&0&*&-\\0&*&0&=\\=&-&=&1\end{pmatrix}$	FPT	Lemma 32-Case 4.4.1
	$\begin{pmatrix}0&*&0&0\\*&0&*&*\\0&*&0&1\\0&*&1&1\end{pmatrix}$	FPT	Lemma 32-Case 4.4.2
	$\begin{pmatrix}0&*&0&0\\*&0&*&0\\0&*&0&1\\0&0&1&1\end{pmatrix}$	FPT	Lemma 32-Case 4.4.3
	$\begin{pmatrix}0&*&0&0\\*&0&*&1\\0&*&0&1\\0&1&1&1\end{pmatrix}$	FPT	Lemma 32-Case 4.4.4

Figure 3 An overview of our results for matrices of order 4×4, when the diagonal has exactly three 0s and a 1. (Complement or equivalent matrices are not shown.) Each row of the table only considers matrices that are not covered by the previous rows. In a matrix, all entries marked by '=' are of the same value. That is, if both m_{ij} and $m_{p,q}$ are marked by '=', then $m_{ij} = m_{pq}$.

The proof of Lemma 31 can be obtained in the same way as that of Lemma 29 (and using the case analysis in the proof of Lemma 32, to be given shortly).

Before moving on to the algorithm for List M_{31}-Compression, we make an assumption regarding the matrix M_{31} and explain why such an assumption is valid. Recall that in the matrix M_{31}, exactly three of the diagonal entries are 0 and the remaining diagonal entry is 1. We assume that in the matrix M_{31} (where entries are denoted by $m_{i,j}$), $m_{1,1} = m_{2,2} = m_{3,3} = 0$ and $m_{4,4} = 1$. Notice that if M_{31} were not a matrix which satisfies the above condition on the diagonal entries, then there would be a matrix equivalent to M_{31} which satisfies the condition. From the above argument together with Proposition 1 it is enough to consider the case when M_{31} indeed satisfies the previously stated condition on the diagonal entries. We now move to our algorithm for List M_{31}-Compression, in the following lemma.

▶ **Lemma 32.** *Consider a* List M_{31}-Compression *problem, where the input matrix does not contain the* 3-Colouring *matrix as a sub-matrix. Then the problem is* FPT*. Otherwise,*

*the problem is para-*NP*-hard.*

Proof. Let (G, L, S') be the instance of the List M_{31}-Compression problem. Then $G - S'$ has a list M_{31}-partition $\mathcal{V}$. With respect to this partition, let $\mathcal{V}_1$ be the collection of parts that are independent sets, and $\mathcal{V}_2$ be the collection of parts that are cliques.

Let S be a *hypothetical solution* of size k for the problem, which the algorithm is supposed to compute. Again, $G-S$ has a list M_{31}-partition $\mathcal{U}$. Let $\mathcal{U}_1$ be the collection of parts that are independent sets, and $\mathcal{U}_2$ be the collection of parts that are cliques. The algorithm first guesses a partition (Y, N) of S' such that $Y = S' \cap S$ and $N = S' \setminus S$. After this guess, the objective is to compute a set Z of size at most $k' = k - |Y|$ such that $G-(Z \cup Y)$ has a list M_{31}-partition. Also note that since N is not part of the solution S, $G[N]$ must have a list M_{31}-partition. Consider the pairs $(\mathcal{V}_1 - (S \cup S'), \mathcal{V}_2 - (S \cup S'))$ and $(\mathcal{U}_1 - (S \cup S'), \mathcal{U}_2 - (S \cup S'))$ of $G - (S \cup S')$. By Observation 2 we know that the cardinalities of each of the sets $(V(\mathcal{V}_1) \cap V(\mathcal{U}_2)) \setminus (S \cup S')$ and $(V(\mathcal{V}_2) \cap V(\mathcal{U}_1)) \setminus (S \cup S')$ are bounded by 3. So now the algorithm guesses the set $V_C = (V(\mathcal{V}_1) \cap V(\mathcal{U}_2)) \setminus (S \cup S')$ and $V_I = (V(\mathcal{V}_2) \cap V(\mathcal{U}_1)) \setminus (S \cup S')$, each of them having size at most 3. We also guess which part in $\mathcal{U}$ each vertex of $V_I \cup V_C$ will belong to: Note that these vertices are not meant to be deleted. Suppose in our current guess the vertex $v \in V_I \cup V_C$ goes to the part i of $\mathcal{U}$. If $i \notin L(v)$, we discard our current guess. Otherwise, we delete from $L(v)$ all indices except for i. After the guess of V_I and V_C, any vertex in $V(\mathcal{V}_2) \setminus V_I$ either belongs to $V(\mathcal{U}_2)$ or belongs to the hypothetical solution S. Similarly any vertex in $V(\mathcal{V}_1) \setminus V_C$ either belongs to $V(\mathcal{U}_1)$ or belongs to the hypothetical solution S. There are at most $4^{|N|} \leq 4^{k+1}$ list M_{31}-partitions of $G[N]$. The algorithm now guesses a list M_{31}-partition of $G[N]$ that is the restriction of the final list M_{31}-partition of $G - S$, to $G[N]$. Since the list M_{31}-partition is fixed, for each vertex $v \in N$, if it is guessed to belong to part U_i in the fixed list M_{31}-partition of $G[N]$ and if $i \notin L(v)$ then we discard the guess. Otherwise, we delete from $L(v)$ all indices but i. Consider vertices in $V(\mathcal{V}_1) \setminus V_C$. By the above arguments, these vertices will only belong to independent set parts in the final list M_{31}-partition of $G - S$, or they can be deleted. Therefore, for each such vertex v we delete from $L(v)$ the index of the clique part and only keep indices of the independent set parts. Suppose this results in $L(v)$ becoming empty, and suppose $k = 0$, then we must discard the current guess. Otherwise, if $L(v)$ is empty and $k > 0$ then we must delete v and reduce k by 1 (Lemma 9). After this reduction all these vertices have lists of size at most 3. On the other hand, vertices in $V(\mathcal{V}_2) \setminus V_I$ can only belong to the clique part in the final list M_{31}-partition of $G - S$, or can be deleted. Therefore we can modify the lists of these vertices to have only the indices of the clique part. If this results in a vertex $v \in V(\mathcal{V}_2) \setminus V_I$ to have an empty list, then as before we discard the current guess if $k = 0$ or else delete v and reduce k by 1 (Lemma 9). Thus, these lists are of size at most 1.

The next step will be to try and reduce the list sizes of the vertices in $(V(\mathcal{V}_1) \setminus V_C)$ by at least one, since this will result in an instance of Deletion to 2-List M_{31}-Partition. By Proposition 6, we know that Deletion to 2-List M_{31}-Partition is FPT. We let $\mathcal{V}_1 = \{V_1, V_2, V_3\}, \mathcal{V}_2 = \{V_4\}, \mathcal{U}_1 = \{U_1, U_2, U_3\}, \mathcal{U}_2 = \{U_4\}$. Recall that we know the parts V_1, V_2, V_3, V_4 but not the parts U_1, U_2, U_3, U_4. In the current scenario, for a vertex v if $L(v) \cap \{1, 2, 3\} \neq \emptyset$ then $4 \notin L(v)$. Similarly, if $4 \in L(v)$ then $L(v) = \{4\}$. Recall that we are under the assumption that $m_{1,1} = m_{2,2} = m_{3,3} = 0$ and $m_{4,4} = 1$. We do a case analysis over the possibilities of M (see Figure 3 for an overview of these results):

Case 1: Suppose one of $m_{1,2}, m_{1,3}, m_{2,3}$ is 1. We will argue for the case when $m_{1,2} = 1$, and the other cases have symmetric arguments. We guess a vertex $v \in U_1$. Note that for a neighbour w of v, w cannot belong to U_1 and therefore we delete the index 1 from $L(w)$. Similarly, for a non-neighbour w of v, we delete the index 2 from $L(w)$. Thus, for each guess

of v, we have reduced our problem to an instance of DELETION TO 2-LIST M_{31}-PARTITION. This implies that our problem is a yes instance if and only if one of the polynomially many reduced instances is a yes instance of DELETION TO 2-LIST M_{31}-PARTITION.

Case 2: If $m_{1,2} = m_{2,3} = m_{1,3} = *$, then M_{31} contains the 3-COLOURING matrix as a sub-matrix, which is a contradiction.

Case 3: Suppose $m_{1,2} = m_{2,3} = m_{1,3} = 0$. First consider the case when one of $m_{1,4}, m_{2,4}$, $m_{3,4}$ is $*$. We argue for the case when $m_{1,4} = *$ (other cases are symmetric). Then index 1 dominates index 2. This implies that we have reduced our problem to an instance of DELETION TO 2-LIST M_{31}-PARTITION. Otherwise, $m_{1,4} \neq *, m_{2,4} \neq *, m_{3,4} \neq *$. Then at least 2 entries amongst $m_{1,4}, m_{2,4}, m_{3,4}$ are the same. Without loss of generality, let $m_{1,4} = m_{2,4}$. Then index 1 dominates index 2. This implies that we have reduced our problem to an instance of DELETION TO 2-LIST M_{31}-PARTITION.

Case 4: In this case, at least one of $m_{1,2}, m_{2,3}, m_{1,3}$ is 0 and at least one is $*$. Without loss of generality, let $m_{1,2} = *, m_{1,3} = 0$.

Case 4.1: First suppose $m_{2,3} = 0$ and at least one of $m_{2,4} = *$ or $m_{1,4} = *$ holds. If $m_{2,4} = *$ then index 2 dominates index 3 and we have reduced our problem to an instance of DELETION TO 2-LIST M_{31}-PARTITION. If $m_{1,4} = *$ then index 1 dominates index 3 and we have reduced our problem to an instance of DELETION TO 2-LIST M_{31}-PARTITION.

Case 4.2: Now suppose $m_{2,3} = 0$ and $m_{3,4} = *$. We will consider further cases depending on the entries $m_{1,4}$ and $m_{2,4}$. We recall that none of $m_{1,4}$ or $m_{2,4}$ is a $*$, otherwise, it would be covered by the previous cases. Thus, we have the following four cases depending on whether $m_{1,4}$ and $m_{2,4}$ are 0 or 1.

Case 4.2.1: Suppose $m_{1,4} = 1, m_{2,4} = 0$. A symmetric argument can be given for when $m_{2,4} = 1, m_{1,4} = 0$. In this case, consider the vertices of $V_4 \setminus V_I$. At this point of the algorithm, these vertices only have the index 4 in their list, and these vertices either get deleted or will be in U_4. Suppose $|V_4 \setminus V_I| > k$. Then, we know that there is at least one vertex $v \in V_4 \cap U_4$, and we guess this vertex v (there are at most n such guesses). Now consider a neighbour w of v such that $L(w) \subseteq \{1, 2, 3\}$. Such a neighbour cannot be part of U_2 as $m_{2,4} = 0$. Thus, for all such neighbours w of v, we delete the index 2 from $L(w)$ and reduce the list size to at most 2. The remaining vertices w with $L(w) \subseteq \{1, 2, 3\}$ must be non-neighbours of v. Since $m_{1,4} = 1$ it must be the case that w cannot belong to U_1. Therefore, for all such non-neighbours w of v, we delete the index 1 from $L(w)$ and reduce the list size to at most 2. Thus, we have reduced our problem to an instance of DELETION TO 2-LIST M_{31}-PARTITION. On the other hand, suppose $|V_4 \setminus V_I| \leq k$. Then, we guess for each vertex $v \in V_4 \setminus V_I$, whether v belongs to U_4 or is deleted. For our current guess, if we delete j vertices from $V_4 \setminus V_I$ then we decrease our parameter k by j. There are $2^{|V_4 \setminus V_I|} \leq 2^k$ guesses for the fate of the vertices of $V_4 \setminus V_I$. For a fixed guess, we know the vertices of U_4. Since $m_{1,4} = 1, m_{2,4} = 0$ and $m_{3,4} = *$, all common neighbours of U_4 can only be in $U_1 \cup U_3$ and therefore the index 2 can be deleted from their lists. Similarly, all common non-neighbours of U_4 can only be in $U_2 \cup U_3$ and therefore the index 1 can be deleted from their lists. All remaining vertices have a neighbour and a non-neighbour in U_4. Such vertices can only be in U_3 and therefore the indices 1 and 2 can be deleted from their lists. Thus, after this reduction, we have an instance of DELETION TO 2-LIST M_{31}-PARTITION. In total, we have at most $5^k n^{\mathcal{O}(1)}$ many reduced instance of DELETION TO 2-LIST M_{31}-PARTITION such that our original instance is a yes instance of LIST M_{31}-COMPRESSION if and only if there is one reduced instance that is a yes instance of DELETION TO 2-LIST M_{31}-PARTITION.

Case 4.2.2: Suppose $m_{1,4} = m_{2,4} = 0$. Let $Z_4 = \{v \in V(G) \mid L(v) = \{4\}\}$. Note that in

the current scenario, (i) a vertex v that finally belongs to U_4 has $L(v) = \{4\}$, and (ii) any vertex v with the index $4 \in L(v)$ must have $L(v) = \{4\}$ and such a vertex either goes into U_4 or gets deleted. Now let H be an auxiliary graph defined as follows: $V(H) = V(G)$ while $E(H) = \{uv \mid uv \notin E(G), u, v \in Z_4\} \cup \{uv \mid uv \in E(G), |\{u, v\} \cap Z_4| \leq 1\}$. In other words, the edges in H remain the same as in G except for when both endpoints lie in Z_4; in that case the edges in $G[Z_4]$ are deleted and the non-edges of $G[Z_4]$ are introduced to construct H. Note that the current instance of Deletion to List M-Partition is a yes instance if and only if (H, L, k) is a yes instance of 2-DB. Thus, by Observation 4 our problem Deletion to List M-Partition is FPT.

Case 4.2.3: Suppose $m_{1,4} = m_{2,4} = 1$. Let $Z_4 = \{v \in V(G) \mid L(v) = \{4\}\}$. Again, in the current scenario, (i) a vertex v that finally belongs to U_4 has $L(v) = \{4\}$, and (ii) any vertex v with the index $4 \in L(v)$ must have $L(v) = \{4\}$ and such a vertex either goes into U_4 or gets deleted. Now let H be an auxiliary graph defined as follows: $V(H) = V(G)$ while $E(H) = \{uv \mid uv \notin E(G), |\{u, v\} \cap Z_4| \geq 1\} \cup \{uv \mid uv \in E(G), |\{u, v\} \cap Z_4| = 0\}$. In other words, the edges in H remain the same as in G except for when at least one endpoint lies in Z_4; in that case all edges in G incident to Z_4 are deleted and all non-edges of G incident to Z_4 are introduced to construct H. Note that the current instance of Deletion to List M-Partition is a yes instance if and only if (H, L, k) is a yes instance of 2-DB. Thus, by Observation 4 our problem Deletion to List M-Partition is FPT.

Otherwise, $m_{1,4}, m_{2,4}, m_{3,4}$ are not $*$, and at least two of these entries are the same. We further consider the cases based on the values of $m_{1,4}, m_{2,4}$ and $m_{3,4}$.

Case 4.3.1: Suppose $m_{2,3} = 0$ and one of $m_{1,4} = m_{3,4}$ or $m_{2,4} = m_{3,4}$ holds. We only consider the case when $m_{1,4} = m_{3,4}$ (other case can be handled analogously). In this case the index 1 dominates the index 3. And, thus in this case, we can obtain an FPT algorithm by reducing it to an instance of Deletion to 2-List M_{31}-Partition.

From the above we can assume that when $m_{2,3} = 0$, none of $m_{1,4} = m_{3,4}$ or $m_{2,4} = m_{3,4}$ holds. Moreover, since $m_{1,4}, m_{2,4}, m_{3,4} \in \{0, 1\}$, at least two of them must be equal. From the above we can conclude that $m_{1,4} = m_{2,4}$ and $m_{1,4} \neq m_{3,4}$. Now we consider cases based on whether $m_{1,4}$ is 1 or 0.

Case 4.3.2: Suppose $m_{2,3} = 0$, $m_{1,4} = m_{2,4} = 1, m_{3,4} = 0$. Recall that we are in the case when $m_{1,2} = *, m_{1,3} = 0$. Consider the vertices of $V_4 \setminus V_I$. At this point of the algorithm, these vertices only have the index 4 in their list, and these vertices either get deleted or will be in U_4. Suppose $|V_4 \setminus V_I| > k$. Then, we know that there is at least one vertex $v \in V_4 \cap U_4$. Guess such a vertex v and do as follows. Note that there are at most n possible guesses for v. Since $m_{1,4} = 1$, $m_{3,4} = 0$ and since $v \in V_4 \cap U_4$, a vertex in $V_3 \setminus V_C$ cannot belong to U_1. Therefore, we delete the index 1 from the list of such a vertex. By similar arguments, a vertex in $V_1 \setminus V_C$ cannot belong to U_3 and we delete 3 from the list of such a vertex. The same arguments can be made for vertices of $V_3 \setminus V_C$ belonging to U_2, and for vertices of $V_2 \setminus V_C$ belonging to U_3. Thus, we have reduced our problem to an instance of Deletion to 2-List M_{31}-Partition. On the other hand, suppose $|V_4 \setminus V_I| \leq k$. Then, we guess for each vertex $v \in V_4 \setminus V_I$, whether v belongs to U_4 or is deleted. For our current guess, if we delete j vertices from $V_4 \setminus V_I$ then we decrease our parameter k by j. There are $2^{|V_4 \setminus V_I|} \leq 2^k$ guesses for the fate of the vertices of $V_4 \setminus V_I$. For a fixed guess, we know the vertices of U_4. In light of Remark 10, we can assume that $U_4 \neq \emptyset$, for otherwise the problem reduces to Deletion to List M'-partition, where M' is the 3×3 matrix obtained from our M_{31} matrix by deleting its fourth row and column. Since $m_{1,4} = m_{2,4} = 1$ and $m_{3,4} = 0$, all common neighbours of U_4 can only be in $U_1 \cup U_2$ and therefore the index 3 can be deleted from their lists. Similarly, all common non-neighbours of U_4 can only be in U_3 and

therefore the indices 1 and 2 can be deleted from their lists. All remaining vertices have a neighbour and a non-neighbour in U_4 and therefore must be deleted and for each such deleted vertex the parameter k is decreased by 1. Thus, after this reduction, we have an instance of DELETION TO 2-LIST M_{31}-PARTITION. In total, we have at most $5^k n^{\mathcal{O}(1)}$ reduced instances of DELETION TO 2-LIST M_{31}-PARTITION such that our original instance is a yes instance of LIST M_{31}-COMPRESSION if and only if there is one reduced instance that is a yes instance of DELETION TO 2-LIST M_{31}-PARTITION.

Case 4.3.3: Suppose $m_{2,3} = 0$, $m_{1,4} = m_{2,4} = 0, m_{3,4} = 1$. Consider the vertices of $V_4 \setminus V_I$. These vertices either get deleted or will be in U_4. Suppose $|V_4 \setminus V_I| > k$. Then, we know that there should be at least one vertex $v \in V_4 \cap U_4$. Guess such a vertex v. There are at most n possible guesses, and for each guess do as follows. Since $m_{1,4} = 0$, $m_{3,4} = 1$ and since $v \in V_4 \cap U_4$, a vertex in $V_1 \setminus V_C$ cannot belong to U_3. Therefore, we can delete the index 3 from the list of such a vertex. By similar arguments, a vertex in $V_3 \setminus V_C$ cannot belong to U_1 and we can delete the index 1 from the list of such a vertex. The same arguments can be made for vertices of $V_3 \setminus V_C$ belonging to U_2, and for vertices of $V_2 \setminus V_C$ belonging to U_3. Thus, we have reduced our problem to an instance of DELETION TO 2-LIST M_{31}-PARTITION. On the other hand, suppose $|V_4 \setminus V_I| \leq k$. Then, we guess for each vertex $v \in V_4 \setminus V_I$, whether v belongs to U_4 or is deleted. For our current guess, if we delete j vertices from $V_4 \setminus V_I$ then we decrease our parameter k by j. There are $2^{|V_4 \setminus V_I|} \leq 2^k$ guesses for the fate of the vertices of $V_4 \setminus V_I$. For a fixed guess, we know the vertices of U_4. As in the previous case, we can assume that $U_4 \neq \emptyset$, because of Remark 10. Since $m_{1,4} = m_{2,4} = 0$ and $m_{3,4} = 1$, all common neighbours of U_4 can only be in U_3 and therefore the indices $1, 2$ can be deleted from their lists. Similarly, all common non-neighbours of U_4 can only be in $U_1 \cup U_2$ and therefore the index 3 can be deleted from their lists. All remaining vertices have a neighbour and a non-neighbour in U_4. Therefore each such vertex must be deleted and the parameter k decreased by 1 for each such vertex. Thus, after this reduction, we have an instance of DELETION TO 2-LIST M_{31}-PARTITION. Since we are making at most $5^k n^{\mathcal{O}(1)}$ guesses, we have at most $5^k n^{\mathcal{O}(1)}$ reduced instances of DELETION TO 2-LIST M_{31}-PARTITION such that our original instance is a yes instance of LIST M_{31}-COMPRESSION if and only if there is one reduced instance that is a yes instance of DELETION TO 2-LIST M_{31}-PARTITION.

Now we consider cases where $m_{2,3} = *$.

Case 4.4.1: Suppose $m_{2,3} = *$ and one of $m_{1,4} = *$, $m_{3,4} = *$ or $m_{1,4} = m_{3,4}$ holds. If $m_{1,4} = *$ then index 1 dominates index 3, and if $m_{3,4} = *$ then index 3 dominates index 1. In both cases, we have reduced our problem to an instance of DELETION TO 2-LIST M_{31}-PARTITION. Otherwise, if $m_{1,4} = m_{3,4}$, then index 1 dominates index 3 and again we have reduced our problem to an instance of DELETION TO 2-LIST M_{31}-PARTITION.

In the following cases we can assume without loss of generality that $m_{1,4} = 0$ and $m_{3,4} = 1$. Now we do case analysis on the value of $m_{2,4}$.

Case 4.4.2: Let $m_{2,3} = *, m_{1,4} = 0, m_{3,4} = 1, m_{2,4} = *$. Recall that we are still under the assumption that $m_{1,2} = *, m_{1,3} = 0$. Consider the vertices of $V_4 \setminus V_I$. These vertices either get deleted or will be in U_4. Suppose $|V_4 \setminus V_I| > k$. Then, we know that there is at least one vertex $\hat{v} \in V_4 \cap U_4$. We guess this vertex $\hat{v}$. There are at most n such guesses. Note that at this point of the algorithm, $L(\hat{v}) = \{4\}$. Since $m_{1,4} = 0$, $m_{3,4} = 1$ and since $\hat{v} \in V_4 \cap U_4$, a vertex in $V_1 \setminus V_C$ cannot belong to U_3. Therefore, we can delete the index 3 from the list of such a vertex. By similar arguments, a vertex in $V_3 \setminus V_C$ cannot belong to U_1 and we can delete the index 1 from the list of such a vertex. After this, only vertices of $V_2 \setminus V_C$ can have lists of size 3. Let A be the subset of vertices in $V_3 \setminus V_C$ that are neighbours to $\hat{v}$ and B be the set of vertices in $V_3 \setminus V_C$ that are non-neighbours to $\hat{v}$. As $m_{1,4} = 0$, a vertex from A

cannot belong to U_1 and therefore, the index 1 can be deleted from its list. Similarly, since $m_{3,4} = 1$, a vertex from B cannot belong to U_3 and the index 3 can be deleted from its list. Thus, we have reduced our problem to an instance of Deletion to 2-List M_{31}-Partition. Since there are at most n guesses for the vertex $\hat{v} \in V_4 \cap U_4$, we create at most n reduced instances of Deletion to 2-List M_{31}-Partition such that our original instance is a yes instance of List M_{31}-Compression if and only if there is one reduced instance that is a yes instance of Deletion to 2-List M_{31}-Partition. On the other hand, suppose $|V_4 \setminus V_I| \leq k$. Then, we guess for each vertex $v \in V_4 \setminus V_I$, whether v belongs to U_4 or is deleted. For our current guess, if we delete j vertices from $V_4 \setminus V_I$ then we decrease our parameter k by j. There are $2^{|V_4 \setminus V_I|} \leq 2^k$ guesses for the fate of the vertices of $V_4 \setminus V_I$. Again, we assume that $U_4 \neq \emptyset$ because of Remark 10. For a fixed guess, we know the vertices of U_4. Since $m_{1,4} = 0, m_{3,4} = 1$ and $m_{2,4} = *$, all vertices with a neighbour in U_4 can only be in $U_2 \cup U_3$ and therefore the index 1 can be deleted from their lists. Similarly, a vertex that has no neighbours in U_4 can only be in $U_1 \cup U_2$ and therefore the index 3 can be deleted from their lists. Thus, we have an instance of Deletion to 2-List M_{31}-Partition. In total, we have $5^k n^{\mathcal{O}(1)}$ many reduced instances of Deletion to 2-List M_{31}-Partition such that our original instance is a yes instance of List M_{31}-Compression if and only if there is one reduced instance that is a yes instance of Deletion to 2-List M_{31}-Partition.

Case 4.4.3: Let $m_{2,3} = *, m_{1,4} = 0, m_{3,4} = 1, m_{2,4} = 0$. Recall that we are still under the assumption that $m_{1,2} = *, m_{1,3} = 0$. Consider the vertices of $V_4 \setminus V_I$. These vertices either get deleted or will be in U_4. Suppose $|V_4 \setminus V_I| > k$. Then, we know that there is at least one vertex $v \in V_4 \cap U_4$. We guess that vertex v. There are at most n guesses. Since $m_{1,4} = 0$, $m_{3,4} = 1$ and since $v \in V_4 \cap U_4$, a vertex in $V_1 \setminus V_C$ cannot belong to U_3. Therefore, we can remove the index 3 from the list of such a vertex. By similar arguments, a vertex in $V_3 \setminus V_C$ cannot belong to U_1 and we can remove the index 1 from the list of such a vertex. The same arguments can be made for vertices of $V_3 \setminus V_C$ belonging to U_2, and for vertices of $V_2 \setminus V_C$ belonging to U_3. Thus, we have reduced our problem to an instance of Deletion to 2-List M_{31}-Partition. On the other hand, suppose $|V_4 \setminus V_I| \leq k$. Then, we guess for each vertex $v \in V_4 \setminus V_I$, whether v belongs to U_4 or is deleted. For our current guess, if we delete j vertices from $V_4 \setminus V_I$ then we decrease our parameter k by j. There are $2^{|V_4 \setminus V_I|} \leq 2^k$ guesses for the fate of the vertices of $V_4 \setminus V_I$. For a fixed guess, we know the vertices of U_4. Again, we assume that $U_4 \neq \emptyset$. Since $m_{1,4} = m_{2,4} = 0$ and $m_{3,4} = 1$, all common neighbours of U_4 can only be in U_3 and therefore the indices $1, 2$ can be deleted from their lists. Similarly, all common non-neighbours of U_4 can only be in $U_1 \cup U_2$ and therefore the index 3 can be deleted from their lists. For any remaining vertex w, w has a neighbour and a non-neighbour in U_4. Therefore w must be deleted and the parameter k must be decreased by 1. Thus, after this reduction, we have an instance of Deletion to 2-List M_{31}-Partition. In total, we have at most $5^k n^{\mathcal{O}(1)}$ many reduced instances of Deletion to 2-List M_{31}-Partition such that our original instance is a yes instance of List M_{31}-Compression if and only if there is one reduced instance that is a yes instance of Deletion to 2-List M_{31}-Partition.

Case 4.4.4: Let $m_{2,3} = *, m_{1,4} = 0, m_{3,4} = 1, m_{2,4} = 1$. Recall that we are still under the assumption that $m_{1,2} = *, m_{1,3} = 0$. Consider the vertices of $V_4 \setminus V_I$. These vertices either get deleted or will be in U_4. Suppose $|V_4 \setminus V_I| > k$. Then, we know that there is at least one vertex $v \in V_4 \cap U_4$. As in the previous cases, we guess such a vertex v. Since $m_{1,4} = 0$, $m_{3,4} = 1$ and since $v \in V_4 \cap U_4$, a vertex in $V_1 \setminus V_C$ cannot belong to U_3. Therefore, we can delete the index 3 from the list of such a vertex. By similar arguments, a vertex in $V_3 \setminus V_C$ cannot belong to U_1 and we can delete the index 1 from the list of such a vertex. The same arguments can be made for vertices of $V_1 \setminus V_C$ belonging to U_2, and for vertices of $V_2 \setminus V_C$

Description/(Representative) Matrix	Class	Proof
Contains the 3-Coloring matrix as a sub matrix	para-NP-hard	Lemma 27 (Also see Lemma 33-Case 1)
At least one off-diagonal entry is 1	FPT	Proposition 6
$\begin{pmatrix} 0 & 0 & * & * \\ 0 & 0 & * & * \\ * & * & 0 & 0 \\ * & * & 0 & 0 \end{pmatrix}$	FPT	Lemma 33-Case 2
$\begin{pmatrix} 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \\ * & * & * & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ * & 0 & 0 & 0 \\ * & * & 0 & 0 \end{pmatrix}$	FPT	Lemma 33-Case 3
$M_{\mathsf{2db}} = \begin{pmatrix} 0 & * & 0 & 0 \\ * & 0 & 0 & 0 \\ 0 & 0 & 0 & * \\ 0 & 0 & * & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & * & * \\ 0 & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ * & 0 & 0 & 0 \end{pmatrix}$	FPT	Lemma 33-Case 4 (and Observation 4)
$\begin{pmatrix} 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 \\ * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	FPT	Lemma 33-Case 5
$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	FPT	Lemma 33-Case 6

Figure 4 An overview of our results for matrices of order 4×4 for the case when all diagonal entires are 0s. (Complement or equivalent matrices are not shown.) Each row of the table only considers matrices that are not covered by the previous rows.

belonging to U_1. Thus, we have reduced our problem to an instance of Deletion to 2-List M_{31}-Partition. On the other hand, suppose $|V_4 \setminus V_I| \leq k$. Then, we guess for each vertex $v \in V_4 \setminus V_I$, whether v belongs to U_4 or is deleted. For our current guess, if we delete j vertices from $V_4 \setminus V_I$ then we decrease our parameter k by j. There are $2^{|V_4 \setminus V_I|} \leq 2^k$ guesses for the fate of the vertices of $V_4 \setminus V_I$. Again, we assume that $U_4 \neq \emptyset$ because of Remark 10. For a fixed guess, we know the vertices of U_4. Since $m_{1,4} = 0, m_{2,4} = m_{3,4} = 1$, all common neighbours of U_4 can only be in $U_3 \cup U_4$ and therefore the index 1 can be deleted from their lists. Similarly, all common non-neighbours of U_4 can only be in U_1 and therefore the indices $3, 4$ can be deleted from their lists. Consider a remaining vertex w. The vertex w has a neighbour and a non-neighbour in U_4. Therefore, w must be deleted and the parameter k must be decreased by 1. Thus, after this reduction, we have an instance of Deletion to 2-List M_{31}-Partition. In total, we have $5^k n^{\mathcal{O}(1)}$ reduced instances of Deletion to 2-List M_{31}-Partition such that our original instance is a yes instance of List M_{31}-Compression if and only if there is one reduced instance that is a yes instance of Deletion to 2-List M_{31}-Partition.

From the above case analysis, we conclude that on fixing guesses for V_C, V_I and an M_{31}-partition of $G[N]$ we reduce our List M_{31}-Compression problem to at most $5^k n^{\mathcal{O}(1)}$ instances of Deletion to 2-List M_{31}-Partition, such that our original List M_{31}-Compression instance is a yes instance if and only if one of the reduced instances is a yes instance of Deletion to 2-List M_{31}-Partition. Since by Proposition 6, Deletion to 2-List M_{31}-Partition is FPT and since there are polynomially many guesses that are being made, the List M_{31}-Compression problem is FPT. ◀

F.3 FPT Algorithm for Deletion to List M_{40}-Partition

Finally, we consider the Deletion to List M_{40}-Partition problem, where M_{40} is a matrix that has all four diagonal entries as 0. In Figure 4, we give an overview of these results.

▶ **Lemma 33.** *For a* Deletion to List M_{40}-Partition *problem, if M_{40} 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.*

Proof. For a matrix M_{40}, graph G and positive integer k, we want to determine if there is a vertex set S of size at most k such that $G - S$ has a list M_{40}-partition $\{V_1, V_2, V_3, V_4\}$. We also know that in order to be a list M_{40}-partition, the partition must have each part V_i, $1 \leq i \leq 4$, as an independent set.

It is enough to consider problems where the input matrix does not contain the 3-Colouring matrix as a sub-matrix. In the remainder of the proof, we assume that M does not contain an equivalent matrix of M_{col} (or its complement) as a sub-matrix. When all 4 parts are independent sets, then the proof is very similar to that of Theorem 6.2 of [20]. In the theorem, all cases where the input matrix has at least one off-diagonal entry as 1, can be reduced to polynomially many instances of Deletion to 2-List M_{40}-Partition problems. By Proposition 6, the Deletion to 2-List M_{40}-Partition is FPT. Therefore all Deletion to List M_{40}-Partition problems, where the input matrix M_{40} has at least one off-diagonal entry as 1 and does not have the 3-Colouring matrix as a sub-matrix, are FPT. The remaining cases are on Deletion to List M_{40}-Partition problems where all off-diagonal entries are either 0 or $*$. Since the input matrix M_{40} is a symmetric matrix, we will only be distinguishing cases according to the lower triangular sub-matrix M_L of M_{40}. In most of the cases our strategy is to reduce the list sizes using Proposition 5; if the list sizes of each vertex become at most 2 then we apply Proposition 6 to show that in the considered case the problem is FPT.

Case 1. If at most one off-diagonal entry of M_L is 0, the 3-Colouring matrix is a sub-matrix of M and therefore this case is para-NP-hard.

Case 2. Suppose exactly two off-diagonal entries of M_L are 0s. Let m_{i_1,j_1} and m_{i_2,j_2} be these two off-diagonal 0 entries. First, since these are off-diagonal entries, we have $i_i \neq j_1$ and $i_2 \neq j_2$. Now, if the four indices i_1, i_2, j_1, j_2 are not pairwise distinct, then M contains the 3-Colouring matrix as a sub-matrix. (To see this, if two of the indices are equal, say, if $i_1 = j_2$, then the sub-matrix of M obtained by removing the i_1th row and column is the 3-Colouring matrix.) Thus, i_1, i_2, j_1, j_2 are all distinct. Then index i_1 dominates j_1 and index i_2 dominates j_2. Therefore, by Proposition 5, this Deletion to List M_{40}-Partition problem becomes a Deletion to 2-List M_{40}-Partition problem and hence is FPT.

Case 3. Suppose exactly three off-diagonal entries of M_L that are 0s. Notice that in this case we have exactly three off-diagonal entries as $*$s and exactly three off-diagonal entries are 0s. We will analyse this case by considering K_4 (clique on four vertices), where each vertex denotes a part and the edges are annotated with an element from $\{0, *\}$, which represents the corresponding off-diagonal entry. That is, we let $V(K_4) = \{z_1, z_2, z_3, z_4\}$, where z_i represents the part i, for $i \in [4]$, and $E(K_4) = \{z_i z_j \mid i, j \in [4], i \neq j\}$. For (distinct) $z_i, z_j \in V(K_4)$, we set $\mathsf{ann}(z_i z_j) = m_{i,j}$. For $i \in [4]$, we let $x^i_{\mathsf{ann}} = |\{j \in [4] \setminus \{i\} \mid \mathsf{ann}(z_i z_j) = *\}|$. Note that for every $i \in [4]$, $x^i_{\mathsf{ann}} \leq 3$. Firstly, we consider the case when there is $i \in [4]$ such that $x^i_{\mathsf{ann}} = 3$. Let $\hat{I} = [4] \setminus \{i\}$. For every distinct $j, j' \in \hat{I}$, we have that j dominates j'. Now we apply Proposition 5 to reduce the list size for each vertex to at most 2. We can now obtain an FPT algorithm for the case using Proposition 6. Hereafter, we assume that for each $i \in [4]$,

we have $x^i_{\mathsf{ann}} < 3$. If there is a subset $X \subseteq V(K_4)$ of size 3 such that for each $zz' \in K_4[X]$, we have $\mathsf{ann}(zz') = *$, then notice that M has a matrix equivalent to M_{col} as a submatrix. Thus, we assume that there is no such X. We now consider the remaining case, which is not covered by any of the previous cases. Notice that the maximum matching size of K_4 is exactly two and we are in the case when there are exactly three off-diagonal entries which are $*$s. Thus, there exists $i \in [4]$, such that $x^i_{\mathsf{ann}} = 2$. Let j_1, j_2, j_3 be distinct integers in $[4] \setminus \{i\}$ such that $\mathsf{ann}(z_i z_{j_1}) = \mathsf{ann}(z_i z_{j_2}) = *$ and $\mathsf{ann}(z_i z_{j_3}) = 0$. Note that $\mathsf{ann}(z_{j_1} z_{j_2}) = 0$ (otherwise, we have a triangle with all edges annotated with $*$s). Without loss of generality we assume that $\mathsf{ann}(z_{j_1} z_{j_3}) = *$ and $\mathsf{ann}(z_{j_2} z_{j_3}) = 0$. Note that in the above case i dominates j_3 and j_1 dominates j_2. Again we reduce the list size for each vertex to at most 2 by using Proposition 5 and obtain an FPT algorithm by using Proposition 6.

Case 4. Suppose exactly four off-diagonal entries of M_L are 0s. If the remaining two $*$ entries of M_L do not have a common row or a common column, then M_{40} is equivalent to the 2-DB matrix. And then by Observation 4, the problem is FPT. Otherwise, without loss of generality, the two $*$ entries have a common row i_1. Let the respective columns be i_2 and i_3. Let the remaining index be i_4. Then index i_1 dominates i_4 and index i_2 dominates i_3. Therefore, the problem becomes a Deletion to 2-List M_{40}-Partition.

Case 5. At most one off-diagonal entry of M_L is $*$. Let i_1 and i_2 be the row and column of this entry, respectively. Let the other two row indices be i_3 and i_4. Then index i_1 dominates i_3 and index i_2 dominates i_4. Therefore, the problem becomes a Deletion to 2-List M_{40}-Partition and is FPT.

Case 6. All the off-diagonal entries of M_L are 0. Let index 1 dominates index 2 and index 3 dominates index 4. Therefore, the problem becomes a Deletion to 2-List M_{40}-Partition problem and is FPT.

Thus, all Deletion to List M_{40}-Partition problems, where the input matrix M has all diagonal entries as 0 and does not have a sub-matrix equivalent to the 3-Colouring matrix (or its complement), are FPT. ◀