COUNTING LIST MATRIX PARTITIONS OF GRAPHS*

ANDREAS GÖBEL[†], LESLIE ANN GOLDBERG[†], COLIN MCQUILLAN[‡], DAVID RICHERBY[†], AND TOMOYUKI YAMAKAMI[§]

Abstract. Given a symmetric $D \times D$ matrix M over $\{0, 1, *\}$, a list M-partition of a graph G is a partition of the vertices of G into D parts which are associated with the rows of M. The part of each vertex is chosen from a given list in such a way that no edge of G is mapped to a 0 in M and no non-edge of G is mapped to a 1 in M. Many important graph-theoretic structures can be represented as list M-partitions including graph colourings, split graphs and homogeneous sets and pairs, which arise in the proofs of the weak and strong perfect graph conjectures. Thus, there has been quite a bit of work on determining for which matrices M computations involving list M-partitions are tractable. This paper focuses on the problem of counting list M-partitions, given a graph G and given a list for each vertex of G. We identify a certain set of "tractable" matrices M. We give an algorithm that counts list M-partitions in polynomial time for every (fixed) matrix M in this set. The algorithm relies on data structures such as sparse-dense partitions and subcube decompositions to reduce each problem instance to a sequence of problem instances in which the lists have a certain useful structure that restricts access to portions of M in which the interactions of 0s and 1s is controlled. We show how to solve the resulting restricted instances by converting them into particular counting constraint satisfaction problems (#CSPs) which we show how to solve using a constraint satisfaction technique known as "arc-consistency". For every matrix M for which our algorithm fails, we show that the problem of counting list M-partitions is #P-complete. Furthermore, we give an explicit characterisation of the dichotomy theorem — counting list M-partitions is tractable (in FP) if the matrix M has a structure called a derectangularising sequence. If M has no derectangularising sequence, we show that counting list M-partitions is #P-hard. We show that the meta-problem of determining whether a given matrix has a derectangularising sequence is NP-complete. Finally, we show that list M-partitions can be used to encode cardinality restrictions in M-partitions problems and we use this to give a polynomial-time algorithm for counting homogeneous pairs in graphs.

Key words. Counting problems, complexity dichotomy, #P-completeness, graph algorithms, matrix partitions of graphs.

AMS subject classifications. 68Q25, 68Q17, 05C15.

1. Introduction.

A matrix partition of an undirected graph is a partition of its vertices according to a matrix which specifies adjacency and non-adjacency conditions on the vertices, depending on the parts to which they are assigned. For finite sets D and D', the set $\{0, 1, *\}^{D \times D'}$ is the set of matrices with rows indexed by D and columns indexed by D' where each $M_{i,j} \in \{0, 1, *\}$. For any symmetric matrix $M \in \{0, 1, *\}^{D \times D}$, an *$M$-partition* of an undirected graph $G = (V, E)$ is a function $\sigma \colon V \to D$ such that, for distinct vertices u and v,

(i) $M_{\sigma(u),\sigma(v)} \neq 0$ if $(u, v) \in E$ and
(ii) $M_{\sigma(u),\sigma(v)} \neq 1$ if $(u, v) \notin E$.

*A preliminary version of this paper appeared in the proceedings of CCC 2014. The research leading to these results has received funding from the MEXT Grants-in-Aid for Scientific Research and the EPSRC and the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

[†]Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, United Kingdom.

[‡]Department of Computer Science, Ashton Building, University of Liverpool, Liverpool, L69 3BX, United Kingdom.

[§] Department of Information Science, University of Fukui, 3-9-1 Bunkyo, Fukui City, Fukui 910-8507, Japan.

Thus, $M_{i,j} = 0$ means that no edges are allowed between vertices in parts i and j, $M_{i,j} = 1$ means that there must be an edge between every pair of vertices in the two parts and $M_{i,j} = *$ means that any set of edges is allowed between the parts. For entries $M_{i,i}$ on the diagonal of M, the conditions only apply to distinct vertices in part i. Thus, $M_{i,i} = 1$ requires that the vertices in part i form a clique in G and $M_{i,i} = 0$ requires that they form an independent set.

For example, if $D = \{i, c\}$, $M_{i,i} = 0$, $M_{c,c} = 1$ and $M_{c,i} = M_{i,c} = *$, i.e., $M = \left(\begin{smallmatrix} 0 & * \\ * & 1 \end{smallmatrix}\right)$, then an M-partition of a graph is a partition of its vertices into an independent set (whose vertices are mapped to i) and a clique (whose vertices are mapped to c). The independent set and the clique may have arbitrary edges between them. A graph that has such an M-partition is known as a split graph [17].

As Feder, Hell, Klein and Motwani describe [15], many important graph-theoretic structures can be represented as M-partitions, including graph colourings, split graphs, (a, b)-graphs [2], clique-cross partitions [10], and their generalisations. M-partitions also arise as "type partitions" in extremal graph theory [1]. In the special case where M is a $\{0, *\}$-matrix (that is, it has no 1 entries), M-partitions of G correspond to homomorphisms from G to the (potentially looped) graph H whose adjacency matrix is obtained from M by turning every $*$ into a 1. Thus, proper $|D|$-colourings of G are exactly M-partitions for the matrix M which has 0s on the diagonal and $*$s elsewhere.

To represent more complicated graph-theoretic structures, such as homogeneous sets and their generalisations, which arise in the proofs of the weak and strong perfect graph conjectures [5, 20], it is necessary to generalise M-partitions by introducing lists. Details of these applications are given by Feder et al. [15], who define the notion of a list M-partition.

A *list M-partition* is an M-partition σ that is also required to satisfy constraints on the values of each $\sigma(v)$. Let $\mathcal{P}(D)$ denote the powerset of D. We say that σ *respects* a function $L\colon V(G) \to \mathcal{P}(D)$ if $\sigma(v) \in L(v)$ for all $v \in V(G)$. Thus, for each vertex v, $L(v)$ serves as a list of allowable parts for v and a *list M-partition* of G is an M-partition that respects the given list function. We allow empty lists for technical convenience, although there are no M-partitions that respect any list function L where $L(v) = \emptyset$ for some vertex v.

Feder et al. [15] study the computational complexity of the following decision problem, which is parameterised by a symmetric matrix $M \in \{0, 1, *\}^{D \times D}$.

Name. LIST-M-PARTITIONS.
Instance. A pair (G, L) in which G is a graph and L is a function $V(G) \to \mathcal{P}(D)$.
Output. "Yes", if G has an M-partition that respects L; "no", otherwise.

Note that M is a parameter of the problem rather than an input of the problem. Thus, its size is a constant which does not vary with the input.

A series of papers [11, 13, 14] described in [15] presents a complete dichotomy for the special case of homomorphism problems, which are LIST-M-PARTITIONS problems in which M is a $\{0, *\}$-matrix. In particular, Feder, Hell and Huang [14] show that, for every $\{0, *\}$-matrix M (and symmetrically, for every $\{1, *\}$-matrix M), the problem LIST-M-PARTITIONS is either polynomial-time solvable or NP-complete.

It is important to note that both of these special cases of LIST-M-PARTITIONS are constraint satisfaction problems (CSPs) and a famous conjecture of Feder and Vardi [16] is that a P versus NP-complete dichotomy also exists for every CSP. Although general LIST-M-PARTITIONS problems can also be coded as CSPs with restrictions on the input,[1] it is not known how to code them without such restrictions. Since the Feder–Vardi conjecture applies only to CSPs with unrestricted inputs, even if proved, it would not necessarily apply to List-M-partitions.

Given the many applications of List-M-partitions, it is important to know whether there is a dichotomy for this problem. This is part of a major ongoing research effort which has the goal of understanding the boundaries of tractability by identifying classes of problems, as wide as possible, where dichotomy theorems arise and where the precise boundary between tractability and intractability can be specified.

Significant progress has been made on identifying dichotomies for the List-M-partitions problem. Feder et al. [15, Theorem 6.1] give a complete dichotomy for the special case in which M is at most 3×3, by showing that List-M-partitions is polynomial-time solvable or NP-complete for each such matrix. Later, Feder and Hell studied the List-M-partitions problem under the name $\mathrm{CSP}^*_{1,2}(H)$ and showed [12, Corollary 3.4] that, for every M, List-M-partitions is either NP-complete, or is solvable in quasi-polynomial time. In the latter case, they showed that List-M-partitions is solvable in $n^{O(\log n)}$ time, given an n-vertex graph. Feder and Hell refer to this result as a "quasi-dichotomy".

Although the Feder–Vardi conjecture remains open, a complete dichotomy is now known for counting CSPs. In particular, Bulatov [3] (see also [8]) has shown that, for every constraint language Γ, the counting constraint satisfaction problem $\#\mathrm{CSP}(\Gamma)$ is either polynomial-time solvable, or #P-complete. It is natural to ask whether a similar situation arises for counting list M-partition problems. We study the following computational problem, which is parameterised by a finite symmetric matrix $M \in \{0,1,*\}^{D\times D}$.

Name. #List-M-partitions.
Instance. A pair (G, L) in which G is a graph and L is a function $V(G) \to \mathcal{P}(D)$.
Output. The number of M-partitions of G that respect L.

Hell, Hermann and Nevisi [18] have considered the related #M-partitions problem without lists, which can be seen as #List-M-partitions restricted to the case that $L(v) = D$ for every vertex v. This problem is defined as follows.

Name. #M-partitions.
Instance. A graph G.
Output. The number of M-partitions of G.

In each of the problems List-M-partitions, #List-M-partitions and #M-partitions, the matrix M is fixed and its size does not vary with the input.

Hell et al. gave a dichotomy for small matrices M (of size at most 3×3). In particular, [18, Theorem 10] together with the graph-homomorphism dichotomy of Dyer and Greenhill [7] shows that, for every such M, #M-partitions is either polynomial-time

[1] For the reader who is familiar with CSPs, it might be useful to see how a List-M-partitions problem can be coded as a CSP with restrictions on the input. Given a symmetric $M \in \{0,1,*\}^{D\times D}$, let M_0 be the relation on $D \times D$ containing all pairs $(i,j) \in D \times D$ for which $M_{i,j} \neq 1$. Let M_1 be the relation on $D \times D$ containing all pairs $(i,j) \in D \times D$ for which $M_{i,j} \neq 0$. Then a List-M-partitions problem with input G, L can be encoded as a CSP whose constraint language includes the binary relations M_0 and M_1 and also the unary relations corresponding to the sets in the image of L. Each vertex v of G is a variable in the CSP instance with the unary constraint $L(v)$. If (u,v) is an edge of G then it is constrained by M_1. If it is a non-edge of G, it is constrained by M_0. Note that the CSP instance satisfies the restriction that every pair of distinct variables has exactly one constraint, which is either M_0 or M_1. In a general CSP instance, a pair of variables could be constrained by M_0 and M_1 or one of them, or neither. It is not clear how to code such a general CSP instance as a list partitions problem.

solvable or #P-complete. An interesting feature of counting M-partitions, identified by Hell et al. is that, unlike the situation for homomorphism-counting problems, there are tractable M-partition problems with non-trivial counting algorithms. Indeed the main contribution of the present paper, as described below, is to identify a set of "tractable" matrices M and to give a non-trivial algorithm which solves #LIST-M-PARTITIONS for every such M. We combine this with a proof that #LIST-M-PARTITIONS is #P-complete for every other M.

1.1. Dichotomy theorems for counting list M-partitions. Our main theorem is a general dichotomy for the counting list M-partition problem, for symmetric matrices M of all sizes. As noted above, since there is no known coding of list M-partition problems as CSPs without input restrictions, our theorem is not known to be implied by the dichotomy for #CSP.

Recall that FP is the class of functions computed by polynomial-time deterministic Turing machines. #P is the class of functions f for which there is a nondeterministic polynomial-time Turing machine that has exactly $f(X)$ accepting paths for every input X; this class can be thought of as the natural analogue of NP for counting problems. Our main theorem is the following.

THEOREM 1. *For any symmetric matrix $M \in \{0,1,*\}^{D\times D}$, the problem* #LIST-$M$-PARTITIONS *is either in* FP *or* #P*-complete.*

To prove Theorem 1, we investigate the complexity of the more general counting problem #$\mathcal{L}$-M-PARTITIONS, which has two parameters — a matrix $M \in \{0,1,*\}^{D\times D}$ and a (not necessarily proper) subset $\mathcal{L}$ of $\mathcal{P}(D)$. In this problem, we only allow sets in $\mathcal{L}$ to be used as lists.

Name. #$\mathcal{L}$-M-PARTITIONS.
Instance. A pair (G, L) where G is a graph and L is a function $V(G) \to \mathcal{L}$.
Output. The number of M-partitions of G that respect L.

Note that M and $\mathcal{L}$ are fixed parameters of #$\mathcal{L}$-M-PARTITIONS — they are not part of the input instance. The problem #LIST-M-PARTITIONS is just the special case of #$\mathcal{L}$-M-PARTITIONS where $\mathcal{L} = \mathcal{P}(D)$.

We say that a set $\mathcal{L} \subseteq \mathcal{P}(D)$ is *subset-closed* if $A \in \mathcal{L}$ implies that every subset of A is in $\mathcal{L}$. This closure property is referred to as the "inclusive" case in [12].

DEFINITION 2. *Given a set $\mathcal{L} \subseteq \mathcal{P}(D)$, we write $\mathcal{S}(\mathcal{L})$ for its subset-closure, which is the set $\mathcal{S}(\mathcal{L}) = \{X \mid \text{for some } Y \in \mathcal{L},\ X \subseteq Y\}$.*

We prove the following theorem, which immediately implies Theorem 1.

THEOREM 3. *Let M be a symmetric matrix in $\{0,1,*\}^{D\times D}$ and let $\mathcal{L} \subseteq \mathcal{P}(D)$ be subset-closed. The problem* #$\mathcal{L}$-M-PARTITIONS *is either in* FP *or* #P*-complete.*

Note that this does not imply a dichotomy for the counting M-partitions problem without lists. The problem with no lists corresponds to the case where every vertex of the input graph G is assigned the list D, allowing the vertex to be potentially placed in any part. Thus, the problem without lists is equivalent to the problem #$\mathcal{L}$-M-PARTITIONS with $\mathcal{L} = \{D\}$, but Theorem 3 applies only to the case where $\mathcal{L}$ is subset-closed.

1.2. Polynomial-time algorithms and an explicit dichotomy. We now introduce the concepts needed to give an explicit criterion for the dichotomy in Theorem 3 and to provide polynomial-time algorithms for all tractable cases. We use standard definitions of relations and their arities, compositions and inverses.

Definition 4. *For any symmetric $M \in \{0,1,*\}^{D\times D}$ and any sets $X, Y \in \mathcal{P}(D)$, define the binary relation*

$$H^M_{X,Y} = \{(i,j) \in X \times Y \mid M_{i,j} = *\}\,.$$

The intractability condition for the problem #$\mathcal{L}$-M-partitions begins with the following notion of rectangularity, which was introduced by Bulatov and Dalmau [4].

Definition 5. *A relation $R \subseteq D \times D'$ is* rectangular *if, for all $i, j \in D$, and $i', j' \in D'$,*

$$(i,i'), (i,j'), (j,i') \in R \implies (j,j') \in R\,.$$

Note that the intersection of two rectangular relations is itself rectangular. However, the composition of two rectangular relations is not necessarily rectangular: for example, $\{(1,1),(1,2),(3,3)\} \circ \{(1,1),(2,3),(3,1)\} = \{(1,1),(1,3),(3,1)\}$.

Our dichotomy criterion will be based on what we call $\mathcal{L}$-M-derectangularising sequences. In order to define these, we introduce the notions of pure matrices and M-purifying sets.

Definition 6. *Given index sets X and Y, a matrix $M \in \{0,1,*\}^{X\times Y}$ is* pure *if it has no 0s or has no 1s.*

Pure matrices correspond to ordinary graph homomorphism problems. As we noted above, M-partitions of G correspond to homomorphisms of G when G is a $\{0,*\}$-matrix. The same is true (by complementation) when G is a $\{1,*\}$-matrix.

Definition 7. *For any $M \in \{0,1,*\}^{D\times D}$, a set $\mathcal{L} \subseteq \mathcal{P}(D)$ is M-*purifying *if, for all $X, Y \in \mathcal{L}$, the X-by-Y submatrix $M|_{X\times Y}$ is pure.*

For example, consider the matrix

$$M = \begin{pmatrix} 1 & * & 0 \\ * & 1 & * \\ 0 & * & 1 \end{pmatrix}\,,$$

with rows and columns indexed by $\{0,1,2\}$ in the obvious way. The matrix M is not pure but for $\mathcal{L} = \{\{0,1\},\{2\}\}$, the set $\mathcal{L}$ is M-purifying and so is the closure $\mathcal{S}(\mathcal{L})$.

Definition 8. *An $\mathcal{L}$-M-*derectangularising sequence *of length k is a sequence $D_1, \ldots, D_k$ with each $D_i \in \mathcal{L}$ such that:*

(i) *$\{D_1, \ldots, D_k\}$ is M-purifying and*

(ii) *the relation $H^M_{D_1,D_2} \circ H^M_{D_2,D_3} \circ \cdots \circ H^M_{D_{k-1},D_k}$ is not rectangular.*

If there is an $i \in \{1, \ldots, k\}$ such that D_i is the empty set then the relation $H = H^M_{D_1,D_2} \circ H^M_{D_2,D_3} \circ \cdots \circ H^M_{D_{k-1},D_k}$ is the empty relation, which is trivially rectangular. If there is an i such that $|D_i| = 1$ then H is a Cartesian product, and is therefore rectangular. It follows that $|D_i| \geq 2$ for each i in a derectangularising sequence.

We can now state our explicit dichotomy theorem, which implies Theorem 3 and, hence, Theorem 1.

Theorem 9. *Let M be a symmetric matrix in $\{0,1,*\}^{D\times D}$ and let $\mathcal{L} \subseteq \mathcal{P}(D)$ be subset-closed. If there is an $\mathcal{L}$-M-derectangularising sequence, then the problem #$\mathcal{L}$-M-partitions is #P-complete. Otherwise, it is in* FP.

Sections 3, 4 and 5 develop a polynomial-time algorithm which solves the problem #$\mathcal{L}$-M-partitions whenever there is no $\mathcal{L}$-M-derectangularising sequence. The algorithm involves several steps.

First, consider the case in which $\mathcal{L}$ is subset-closed and M-purifying. In this case, Proposition 15 presents a polynomial-time transformation from an instance of the problem #$\mathcal{L}$-M-PARTITIONS to an instance of a related counting CSP. Algorithm 3 exploits special properties of the constructed CSP instance so that it can be solved in polynomial time using a CSP technique called arc-consistency. (This is proved in Lemma 18.) This provides a solution to the original #$\mathcal{L}$-M-PARTITIONS problem for the M-purifying case.

The case in which $\mathcal{L}$ is not M-purifying is tackled in Section 5. Section 5.1 gives algorithms for constructing the relevant data structures, which include a special case of sparse-dense partitions and also subcube decompositions. Algorithm 9 uses these data structures (via Algorithms 4, 5, 6, 7 and 8) to reduce the #$\mathcal{L}$-M-PARTITIONS problem to a sequence of problems #$\mathcal{L}_i$-M-PARTITIONS where $\mathcal{L}_i$ is M-purifying. Finally, the polynomial-time algorithm is presented in Algorithms 10 and 11. For every $\mathcal{L}$ and M where there is no $\mathcal{L}$-M-derectangularising sequence, either Algorithm 10 or Algorithm 11 defines a polynomial-time function #$\mathcal{L}$-M-PARTITIONS for solving the #$\mathcal{L}$-M-PARTITIONS problem, given an input (G, L). The function #$\mathcal{L}$-M-PARTITIONS is not recursive. However, its *definition* is recursive in the sense that the function #$\mathcal{L}$-M-PARTITIONS defined in Algorithm 11 calls a function #$\mathcal{L}_i$-M-PARTITIONS where $\mathcal{L}_i$ is a subset of $\mathcal{P}(D)$ whose cardinality is smaller than $\mathcal{L}$. The function #$\mathcal{L}_i$-M-PARTITIONS is, in turn, defined either in Algorithm 10 or in 11.

The proof of Theorem 9 shows that, when Algorithms 10 and 11 fail to solve the problem #$\mathcal{L}$-M-PARTITIONS, the problem is #P-complete.

1.3. Complexity of the dichotomy criterion. Theorem 9 gives a precise criterion under which the problem #$\mathcal{L}$-M-PARTITIONS is in FP or #P-complete, where $\mathcal{L}$ and M are considered to be fixed parameters. In Section 6, we address the computational problem of determining which is the case, now treating $\mathcal{L}$ and M as inputs to this "meta-problem". Dyer and Richerby [8] studied the corresponding problem for the #CSP dichotomy, showing that determining whether a constraint language Γ satisfies the criterion for their #CSP(Γ) dichotomy is reducible to the graph automorphism problem, which is in NP. We are interested in the following computational problem, which we show to be NP-complete.

Name. EXISTSDERECTSEQ.

Instance. An index set D, a symmetric matrix M in $\{0, 1, *\}^{D\times D}$ (represented as an array) and a set $\mathcal{L} \subseteq \mathcal{P}(D)$ (represented as a list of lists).

Output. "Yes", if there is an $\mathcal{S}(\mathcal{L})$-M-derectangularising sequence; "no", otherwise.

THEOREM 10. EXISTSDERECTSEQ *is* NP-*complete under polynomial-time many-one reductions.*

Note that, in the definition of the problem EXISTSDERECTSEQ, the input $\mathcal{L}$ is not necessarily subset-closed. Subset-closedness allows a concise representation of some inputs: for example, $\mathcal{P}(D)$ has exponential size but it can be represented as $\mathcal{S}(\{D\})$, so the corresponding input is just $\mathcal{L} = \{D\}$. In fact, our proof of Theorem 10 uses a set of lists $\mathcal{L}$ where $|X| \leq 3$ for all $X \in \mathcal{L}$. Since there are at most $|D|^3 + 1$ such sets, our NP-completeness proof would still hold if we insisted that the input $\mathcal{L}$ to EXISTSDERECTSEQ must be subset-closed.

Let us return to the original problem #LIST-M-PARTITIONS, which is the special case of the problem #$\mathcal{L}$-M-PARTITIONS where $\mathcal{L} = \mathcal{P}(D)$. This leads us to be interested in the following computational problem.

Name. MATRIXHASDERECTSEQ.

Instance. An index set D and a symmetric matrix M in $\{0, 1, *\}^{D\times D}$ (represented as an array).

Output. "Yes", if there is a $\mathcal{P}(D)$-M-derectangularising sequence; "no", otherwise.

Theorem 10 does not quantify the complexity of MatrixHasDerectSeq because its proof relies on a specific choice of $\mathcal{L}$ which, as we have noted, is not $\mathcal{P}(D)$. Nevertheless, the proof of Theorem 10 has the following corollary.

Corollary 11. MatrixHasDerectSeq *is in* NP.

1.4. Cardinality constraints. Many combinatorial structures can be represented as M-partitions with the addition of cardinality constraints on the parts. For example, it might be required that certain parts be non-empty or, more generally, that they contain at least k vertices for some fixed k.

Feder et al. [15] showed that the problem of determining whether such a structure exists in a given graph can be reduced to a List-M-partitions problem in which the cardinality constraints are expressed using lists. In Section 7, we extend this to counting. We show that any #M-partitions problem with additional cardinality constraints of the form, "part d must contain at least k_d vertices" is polynomial-time Turing reducible to #List-M-partitions. As a corollary, we show that the "homogeneous pairs" introduced by Chvátal and Sbihi [6] can be counted in polynomial time. Homogeneous pairs can be expressed as an M-partitions problem for a certain 6×6 matrix, with cardinality constraints on the parts.

2. Preliminaries. For a positive integer k, we write $[k]$ for the set $\{1, \ldots, k\}$. If $\mathcal{S}$ is a set of sets then we use $\bigcap \mathcal{S}$ to denote the intersection of all sets in $\mathcal{S}$. The vertex set of a graph G is denoted $V(G)$ and its edge set is $E(G)$. We write $\{0, 1, *\}^D$ for the set of all functions $\sigma \colon D \to \{0, 1, *\}$ and $\{0, 1, *\}^{D \times D'}$ for the set of all matrices $M = (M_{i,j})_{i \in D, j \in D'}$, where each $M_{i,j} \in \{0, 1, *\}$.

We always use the term "M-partition" when talking about a partition of the vertices of a graph according to a $\{0, 1, *\}$-matrix M. When we use the term "partition" without referring to a matrix, we mean it in the conventional sense of partitioning a set X into disjoint subsets $X_1, \ldots, X_k$ with $X_1 \cup \cdots \cup X_k = X$.

We view computational counting problems as functions mapping strings over input alphabets to natural numbers. Our model of computation is the standard multitape Turing machine. We say that a counting problem P is polynomial-time Turing-reducible to another counting problem Q if there is a polynomial-time deterministic oracle Turing machine M such that, on every instance x of P, M outputs $P(x)$ by making queries to oracle Q. We say that P is polynomial-time Turing-equivalent to Q if each is polynomial-time Turing-reducible to the other. For decision problems (languages), we use the standard many-one reducibility: language A is many-one reducible to language B if there exists a function f that is computable in polynomial time such that $x \in A$ if and only if $f(x) \in B$.

3. Counting list M-partition problems and counting CSPs. Toward the development of our algorithms and the proof of our dichotomy, we study a special case of the problem #$\mathcal{L}$-M-partitions, in which $\mathcal{L}$ is M-purifying and subset-closed. For such $\mathcal{L}$ and M, we show that the problem #$\mathcal{L}$-M-partitions is polynomial-time Turing-equivalent to a counting constraint satisfaction problem (#CSP). To give the equivalence, we introduce the notation needed to specify #CSPs.

A *constraint language* is a finite set Γ of named relations over some set D. For such a language, we define the counting problem #CSP(Γ) as follows.

Name. #CSP(Γ).

Instance. A set V of variables and a set C of constraints of the form $\langle (v_1, \dots, v_k), R\rangle$, where $(v_1, \dots, v_k) \in V^k$ and R is an arity-k relation in Γ.
Output. The number of assignments $\sigma\colon V \to D$ such that

$$(\sigma(v_1), \dots, \sigma(v_k)) \in R \text{ for all } \langle (v_1, \dots, v_k), R\rangle \in C\,. \tag{1}$$

The tuple of variables $v_1, \dots, v_k$ in a constraint is referred to as the constraint's *scope*. The assignments $\sigma\colon V \to D$ for which (1) holds are called the *satisfying assignments* of the instance (V, C). Note that a unary constraint $\langle v, R\rangle$ has the same effect as a list: it directly restricts the possible values of the variable v. As before, we allow the possibility that $\emptyset \in \Gamma$; any instance that includes a constraint $\langle (v_1, \dots, v_k), \emptyset\rangle$ has no satisfying assignments.

Definition 12. *Let M be a symmetric matrix in $\{0, 1, *\}^{D\times D}$ and let $\mathcal{L}$ be a subset-closed M-purifying set. Define the constraint language*

$$\Gamma'_{\mathcal{L},M} = \{H^M_{X,Y} \mid X, Y \in \mathcal{L}\}$$

and let $\Gamma_{\mathcal{L},M} = \Gamma'_{\mathcal{L},M} \cup \mathcal{P}(D)$, where $\mathcal{P}(D)$ represents the set of all unary relations on D.

The unary constraints in $\Gamma_{\mathcal{L},M}$ will be useful in our study of the complexity of the dichotomy criterion, in Section 6. First, we define a convenient restriction on instances of $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$.

Definition 13. *An instance of $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$ is* simple *if:*

(i) *there is exactly one unary constraint $\langle v, X_v\rangle$ for each variable $v \in V$,*

(ii) *there are no binary constraints $\langle (v, v), R\rangle$, and*

(iii) *each pair u, v of distinct variables appears in at most one constraint of the form $\langle (u, v), R\rangle$ or $\langle (v, u), R\rangle$.*

Lemma 14. *For every instance (V, C) of $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$, there is a simple instance (V, C') such that an assignment $\sigma\colon V \to D$ satisfies (V, C) if and only if it satisfies (V, C'). Further, such an instance can be computed in polynomial time.*

Proof. Observe that the set of binary relations in $\Gamma_{\mathcal{L},M}$ is closed under intersections: $H^M_{X,Y} \cap H^M_{X',Y'} = H^M_{X\cap X',Y\cap Y'}$ and this relation is in $\Gamma_{\mathcal{L},M}$ because $\mathcal{L}$ is subset-closed. The binary part of $\Gamma_{\mathcal{L},M}$ is also closed under relational inverse because M is symmetric, so

$$\left(H^M_{X,Y}\right)^{-1} = \{(b, a) \mid (a, b) \in H^M_{X,Y}\} = H^M_{Y,X} \in \Gamma_{\mathcal{L},M}\,.$$

Since $\mathcal{P}(D) \subseteq \Gamma_{\mathcal{L},M}$, the set of unary relations is also closed under intersections.

We construct C' as follows, starting with C. Any binary constraint $\langle (v, v), R\rangle$ can be replaced by the unary constraint $\langle v, \{d \mid (d, d) \in R\}\rangle$. All the binary constraints between distinct variables u and v can be replaced by the single constraint

$$\left\langle (u, v), \bigcap\{R \mid \langle (u, v), R\rangle \in C \text{ or } \langle (v, u), R^{-1}\rangle \in C\}\right\rangle\,.$$

Let the set of constraints produced so far be C''. For each variable v in turn, if there are no unary constraints applied to v in C'', add the constraint $\langle v, D\rangle$; otherwise, replace all the unary constraints involving v in C'' with the single constraint

$$\left\langle v, \bigcap\{R \mid \langle v, R\rangle \in C''\}\right\rangle\,.$$

C' is the resulting constraint set. The closure properties established above guarantee that (V, C') is a $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$ instance. It is clear that it has the same satisfying assignments as (V, C) and that it can be produced in polynomial time. ∎

Our main result connecting the counting list M-partitions problem with counting CSPs is the following.

PROPOSITION 15. *For any symmetric $M \in \{0,1,*\}^{D\times D}$ and any subset-closed, M-purifying set $\mathcal{L}$, the problem #$\mathcal{L}$-M-PARTITIONS is polynomial-time Turing-equivalent to #CSP($\Gamma_{\mathcal{L},M}$).*

Because of its length, we split the proof of the proposition into two lemmas.

LEMMA 16. *For any symmetric matrix $M \in \{0,1,*\}^{D\times D}$ and any subset-closed, M-purifying set $\mathcal{L}$, the problem #CSP($\Gamma_{\mathcal{L},M}$) is polynomial-time Turing-reducible to #$\mathcal{L}$-M-PARTITIONS.*

Proof. Consider an input (V, C) to #CSP($\Gamma_{\mathcal{L},M}$), which we may assume to be simple. Each variable appears in exactly one unary constraint, $\langle v, X_v\rangle \in C$. Any variable v that is not used in a binary constraint can take any value in X_v so just introduces a multiplicative factor of $|X_v|$ to the output of the counting CSP. Thus, we will assume without loss of generality that every variable is used in at least one constraint with a relation from $\Gamma'_{\mathcal{L},M}$ and, by simplicity, there are no constraints of the form $\langle (v, v), R\rangle$.

We now define a corresponding instance (G, L) of #$\mathcal{L}$-M-PARTITIONS. The vertices of G are the variables V of the #CSP instance. For each variable $v \in V$, set

$$L(v) = X_v \cap \bigcap \big\{X \mid \text{for some } u \text{ and } Y,\ \langle (v, u), H^M_{X,Y}\rangle \in C \text{ or } \langle (u, v), H^M_{Y,X}\rangle \in C\big\}\,.$$

The edges $E(G)$ of our instance are the unordered pairs $\{u, v\}$ that satisfy one of the following conditions:

(i) there is a constraint between u and v in C and $M|_{L(u)\times L(v)}$ has a 0 entry, or

(ii) there is no constraint between u and v in C and $M|_{L(u)\times L(v)}$ has a 1 entry.

Since every vertex v is used in at least one constraint with a relation $H^M_{X,Y}$ where, by definition, X and Y are in $\mathcal{L}$, every set $L(v)$ is a subset of some set $W \in \mathcal{L}$. $\mathcal{L}$ is subset-closed so $L(v) \in \mathcal{L}$ for all $v \in V$, as required.

We claim that a function $\sigma\colon V \to D$ is a satisfying assignment of (V, C) if and only if it is an M-partition of G that respects L. Note that, since $\mathcal{L}$ is M-purifying, no submatrix $M|_{X\times Y}$ $(X, Y \in \mathcal{L})$ contains both 0s and 1s.

First, suppose that σ is a satisfying assignment of (V, C). For each variable v, σ satisfies all the constraints $\langle v, X_v\rangle$, $\langle (v, u), H^M_{X,Y}\rangle$ and $\langle (u, v), H^M_{Y,X}\rangle$ containing v. Therefore, $\sigma(v) \in X_v$ and $\sigma(v) \in X$ for each binary constraint $\langle (v, u), H^M_{X,Y}\rangle$ or $\langle (u, v), H^M_{Y,X}\rangle$, so σ satisfies all the list requirements.

To show that σ is an M-partition of G, consider any pair of distinct vertices $u, v \in V$. If there is a constraint $\langle (u, v), H^M_{X,Y}\rangle \in C$, then σ satisfies this constraint so $M_{\sigma(u),\sigma(v)} = *$ and u and v cannot stop σ being an M-partition. Conversely, suppose there is no constraint between u and v in C. If $M|_{L(u)\times L(v)}$ contains a 0, there is no edge $(u, v) \in E(G)$ by construction; otherwise, if $M|_{L(u)\times L(v)}$ contains a 1, there is an edge $(u, v) \in E(G)$ by construction; otherwise, $M_{x,y} = *$ for all $x \in L(u)$, $y \in L(v)$. In all three cases, the assignment to u and v is consistent with σ being an M-partition.

Conversely, suppose that σ is not a satisfying assignment of (V, C). If σ does not satisfy some unary constraint $\langle v, X\rangle$ then $\sigma(v) \notin L(v)$ so σ does not respect $\mathcal{L}$. If σ does not satisfy some binary constraint $\langle (u, v), H^M_{X,Y}\rangle$ where u and v are distinct then, by definition of the relation $H^M_{X,Y}$, $M_{\sigma(u),\sigma(v)} \neq *$. If $M_{\sigma(u),\sigma(v)} = 0$, there is an edge $(u, v) \in E(G)$ by construction, which is forbidden in M-partitions; if $M_{\sigma(u),\sigma(v)} = 1$, there is no edge $(u, v) \in E(G)$ but this edge is required in M-partitions. Hence, σ is not an M-partition. □

LEMMA 17. *For any symmetric $M \in \{0,1,*\}^{D\times D}$ and any subset-closed, M-purifying set $\mathcal{L}$, the problem #$\mathcal{L}$-M-PARTITIONS is polynomial-time Turing-reducible to #CSP($\Gamma_{\mathcal{L},M}$).*

Proof. We now essentially reverse the construction of the previous lemma to give a reduction from #$\mathcal{L}$-M-PARTITIONS to #CSP($\Gamma_{\mathcal{L},M}$). For any instance (G,L) of #$\mathcal{L}$-M-PARTITIONS, we construct a corresponding instance (V,C) of #CSP($\Gamma_{\mathcal{L},M}$) as follows. The set of variables V is $V(G)$. The set of constraints C consists of a constraint $\langle v, L(v)\rangle$ for each vertex $v \in V(G)$ and a constraint $\langle (u,v), H^M_{L(u),L(v)}\rangle$ for every pair of distinct vertices u, v such that:

(i) $(u,v) \in E(G)$ and $M|_{L(u)\times L(v)}$ has a 0 entry, or
(ii) $(u,v) \notin E(G)$ and $M|_{L(u)\times L(v)}$ has a 1 entry.

We show that a function $\sigma\colon V \to D$ is a satisfying assignment of (V,C) if and only if it is an M-partition of G that respects L. It is clear that σ satisfies the unary constraints if and only if it respects L.

If σ satisfies (V,C), then consider any pair of distinct vertices $u, v \in V$. If there is a binary constraint involving u and v, then $M_{\sigma(u),\sigma(v)} = M_{\sigma(v),\sigma(u)} = *$ so the existence or non-existence of the edge (u,v) of G does not affect whether σ is an M-partition. If there is no binary constraint involving u and v, then either there is an edge $(u,v) \in E(G)$ and $M_{\sigma(u),\sigma(v)} \neq 0$ or there is no edge (u,v) and $M_{\sigma(u),\sigma(v)} \neq 1$. In all three cases, σ maps u and v consistently with it being an M-partition.

Conversely, if σ does not satisfy (V,C), either it fails to satisfy a unary constraint, in which case it does not respect L, or it satisfies all unary constraints (so it respects L), but it fails to satisfy a binary constraint $\langle (u,v), H^M_{L(u),L(v)}\rangle$. In the latter case, by construction, $M_{\sigma(u),\sigma(v)} \neq *$ so either $M_{\sigma(u),\sigma(v)} = 0$ but there is an edge $(u,v) \in E(G)$, or $M_{\sigma(u),\sigma(v)} = 1$ and there is no edge $(u,v) \in E(G)$. In either case, σ is not an M-partition of G. □

4. An arc-consistency based algorithm for #CSP($\Gamma_{\mathcal{L},M}$). In the previous section, we showed that a class of #$\mathcal{L}$-M-PARTITIONS problems is equivalent to a certain class of counting CSPs, where the constraint language consists of binary relations and all unary relations over the domain D. We now investigate the complexity of such #CSPs.

Arc-consistency is a standard solution technique for constraint satisfaction problems [19]. It is, essentially, a local search method which initially assumes that each variable may take any value in the domain and iteratively reduces the range of values that can be assigned to each variable, based on the constraints applied to it and the values that can be taken by other variables in the scopes of those constraints.

For any simple #CSP($\Gamma_{\mathcal{L},M}$) instance (V,C), define the vector of *arc-consistent domains* $(D_v)_{v\in V}$ by the procedure in Algorithm 1. At no point in the execution of the algorithm can any domain D_v increase in size so, for fixed D, the running time of the algorithm is at most a polynomial in $|V| + |C|$.

It is clear that, if $(D_v)_{v\in V}$ is the vector of arc-consistent domains for a simple #CSP($\Gamma_{\mathcal{L},M}$) instance (V,C), then every satisfying assignment σ for that instance must have $\sigma(v) \in D_v$ for each variable v. In particular, if some $D_v = \emptyset$, then the instance is unsatisfiable. (Note, though, that the converse does not hold. If $D = \{0,1\}$ and $R = \{(0,1),(1,0)\}$, the instance with constraints $\langle x, D\rangle$, $\langle y, D\rangle$, $\langle z, D\rangle$, $\langle (x,y), R\rangle$, $\langle (y,z), R\rangle$ and $\langle (z,x), R\rangle$ is unsatisfiable but arc-consistency assigns $D_x = D_y = D_z = \{0,1\}$.)

The arc-consistent domains computed for a simple instance (V,C) can yield fur-

Algorithm 1 The algorithm for computing arc-consistent domains for a simple $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$ instance (V, C) where, for each $v \in V$, $\langle v, X_v \rangle \in C$ is the unary constraint involving v.

for $v \in V$ **do**
  $D_v \leftarrow X_v$
repeat
  for $v \in V$ **do**
    $D'_v \leftarrow D_v$
  for $\langle (u, v), R \rangle \in C$ **do**
    $D_u \leftarrow \{d \in D_u \mid \text{for some } d' \in D_v,\ (d, d') \in R\}$
    $D_v \leftarrow \{d \in D_v \mid \text{for some } d' \in D_u,\ (d', d) \in R\}$
until $\forall v \in V,\ D_v = D'_v$
return $(D_v)_{v \in V}$

Algorithm 2 The algorithm for factoring a simple $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$ instance (V, C) with respect to a vector $(D_v)_{v \in V}$ of arc-consistent domains. F is the set of factored constraints.

$F \leftarrow C$
for $\langle (u, v), R \rangle \in C$ **do**
  if $R \cap (D_u \times D_v)$ is a Cartesian product $D'_u \times D'_v$ **then**
    Let $\langle u, X_u \rangle$ and $\langle v, X_v \rangle$ be the unary constraints involving u and v in F.
    $F \leftarrow (F \cup \{\langle u, X_u \cap D'_u \rangle, \langle v, X_u \cap D'_v \rangle\}) \setminus \{\langle (u, v), R \rangle, \langle u, X_u \rangle, \langle v, X_v \rangle\}$
return F

ther simplification of the constraint structure, which we refer to as *factoring*. The factoring applies when the arc-consistent domains restrict a binary relation to a Cartesian product. In this case, the binary relation can be replaced with corresponding unary relations. Algorithm 2 factors a simple instance with respect to a vector $(D_v)_{v \in V}$ of arc-consistent domains, producing a set F of factored constraints. Recall that there is at most one constraint in C between distinct variables and there are no binary constraints $\langle (v, v), R \rangle$ because the instance is simple. Note also that, if $|D_u| \leq 1$ or $|D_v| \leq 1$, then $R \cap (D_u \times D_v)$ is necessarily a Cartesian product. It is easy to see that the result of factoring a simple instance is simple, that Algorithm 2 runs in polynomial time and that the instance (V, F) has the same satisfying assignments as (V, C).

The *constraint graph* of a CSP instance (V, C) (in any constraint language) is the undirected graph with vertex set V that contains an edge between every pair of distinct variables that appear together in the scope of some constraint.

Algorithm 3 uses arc-consistency to count the satisfying assignments of simple $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$ instances. It is straightforward to see that the algorithm terminates, since each recursive call is either on an instance with strictly fewer variables or on one in which at least one variable has had its unary constraint reduced to a singleton and no variable's unary constraint has increased. For general inputs, the algorithm may take exponential time to run but, in Lemma 18 we show that the running time is polynomial for the inputs we are interested in.

We first argue that the algorithm is correct. By Lemma 14, we may assume that the given instance (V, C) is simple. Every satisfying assignment $\sigma \colon V \to D$ satisfies $\sigma(v) \in D_v$ for all $v \in V$ so restricting our attention to arc-consistent domains does

Algorithm 3 The arc-consistency based algorithm for counting satisfying assignments to simple instances of $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$. The input is a simple instance (V,C) of $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$.

```
function AC(variable set V, constraint set C)
    Use Algorithm 1 to compute the vector of arc-consistent domains (D_v)_{v∈V}
    Use Algorithm 2 to construct the set F of factored constraints
    if D_v = ∅ for some v ∈ V then
        return 0
    Compute the constraint graph H of (V, F)
    Let H_1, ..., H_κ be the components of H with V_i = V(H_i)
    Let F_i be the set of constraints in F involving variables in V_i
    for i ∈ [κ] do
        if |D_w| = 1 for some w ∈ V_i then
            Z_i ← 1
        else
            Choose w_i ∈ V_i
            Let θ_i be the unary constraint involving w_i in F_i
            for d ∈ D_{w_i} do
                F'_{i,d} ← (F_i ∪ {⟨w_i, {d}⟩}) \ {θ_i}
            Z_i ← Σ_{d∈D_{w_i}} AC(V_i, F'_{i,d})
    return ∏_{i=1}^{κ} Z_i
```

not alter the output. Factoring the constraints also does not change the number of satisfying assignments: it merely replaces some binary constraints with equivalent unary ones. The constraints are factored, so any variable v with $|D_v| = 1$ must, in fact, be an isolated vertex in the constraint graph because, as noted above, any binary constraint involving it has been replaced by unary constraints. Therefore, if a component H_i contains a variable v with $|D_v| = 1$, that component is the single vertex v, which is constrained to take a single value, so the number of satisfying assignments for this component, which we denote Z_i, is equal to 1. (So we have now shown that the if branch in the for loop is correct.) For components that contain more than one variable, it is clear that we can choose one of those variables, w_i, and group the set of M-partitions σ according to the value of $\sigma(w_i)$. (So we have now shown that the else branch is correct.) Because there are no constraints between variables in different components of the constraint graph, the number of satisfying assignments factorises as $\prod_{i=1}^{\kappa} Z_i$.

For a binary relation R, we write

$$\pi_1(R) = \{a \mid (a,b) \in R \text{ for some } b\}$$
$$\pi_2(R) = \{b \mid (a,b) \in R \text{ for some } a\}\,.$$

For the following proof, we will also need the observation of Dyer and Richerby [8, Lemma 1] that any rectangular relation $R \subseteq \pi_1(R) \times \pi_2(R)$ can be written as $(A_1 \times B_1) \cup \cdots \cup (A_\lambda \times B_\lambda)$, where the A_i and B_i partition $\pi_1(R)$ and $\pi_2(R)$, respectively. The subrelations $A_i \times B_i$ are referred to as *blocks*. A rectangular relation $R \neq \pi_1(R) \times \pi_2(R)$ must have at least two blocks.

LEMMA 18. *Suppose that $\mathcal{L}$ is subset-closed and M-purifying. If there is no $\mathcal{L}$-M-derectangularising sequence, then Algorithm 3 runs in polynomial time.*

Proof. We will argue that the number of recursive calls made by the function AC in Algorithm 3 is bounded above by a polynomial in $|V|$. This suffices, since every other step of the procedure is obviously polynomial.

Consider a run of the algorithm on instance (V,C) which, by Lemma 14, we may assume to be simple. Suppose the run makes a recursive call with input $(V_i, F'_{i,d})$. For each $v \in V_i$, let D'_v denote the arc-consistent domain for v that is computed during the recursive call. We will show below that $D'_v \subset D_v$ for every variable $v \in V_i$. This implies that the recursion depth is at most $|D|$. As a crude bound, it follows that the number of recursive calls is at most $(|V| \cdot |D|)^{|D|}$, since each recursive call that is made is nested below a sequence of at most $|D|$ previous calls, each of which chose a vertex $v \in V$ and "pinned" it to a domain element $d \in D$ (i.e., introduced the constraint $\langle v, \{d\}\rangle$).

Towards showing that the domains of all variables decrease at each recursive call, suppose that we are computing $\mathrm{AC}(V,C)$ and the arc-consistent domains are $(D_v)_{v\in V}$. As observed above, for any component H_i of the constraint graph on which a recursive call is made, we must have $|D_v| > 1$ for every $v \in V_i$. Fix such a component and, for each $v \in V_i$, let D'_v be the arc-consistent domain calculated for v in the recursive call on H_i. It is clear that $D'_v \subseteq D_v$; we will show that $D'_v \subset D_v$.

Consider a path $v_1 \dots v_\ell$ in H_i, where $v_1 = w_i$ and $v_\ell = v$. For each $j \in [\ell-1]$, there is exactly one binary constraint in F_i involving v_j and v_{j+1}. This is either $\langle (v_j, v_{j+1}), R_j\rangle$ or $\langle (v_{j+1}, v_j), R_j^{-1}\rangle$ and, without loss of generality, we may assume that it is the former. For $j \in [\ell-1]$, let $R'_j = R_j \cap (D_{v_j} \times D_{v_{j+1}}) = H^M_{D_{v_j}, D_{v_{j+1}}}$. The relation R'_j is pure because D_{v_j} and $D_{v_{j+1}}$ are in the subset-closed set $\mathcal{L}$ and, since $\mathcal{L}$ is M-purifying, so is $\{D_{v_j}, D_{v_{j+1}}\}$. These two domains do not form a derectangularising sequence by the hypothesis of the lemma, so $H^M_{D_{v_j}, D_{v_{j+1}}}$ is rectangular. If some $R_j = \emptyset$ then $D_{v_j} = D_{v_{j+1}} = \emptyset$ by arc-consistency, contradicting the fact that $|D_v| > 1$ for all $v \in V_i$. If some R'_j has just one block, $R_j \cap (D_{v_j} \times D_{v_{j+1}})$ is a Cartesian product, contradicting the fact that F is a factored set of constraints. Thus, every R'_j has at least two blocks.

For $j \in [\ell-1]$, let $\Phi_j = R'_1 \circ \cdots \circ R'_j$. As above, note that $\{D_{v_1}, \dots, D_{v_{j+1}}\}$ is M-purifying and the sequence $D_{v_1}, \dots, D_{v_{j+1}}$ is not derectangularising, so Φ_j is rectangular. We will show by induction on j that $\pi_1(\Phi_j) = D_{v_1}$, $\pi_2(\Phi_j) = D_{v_{j+1}}$ and Φ_j has at least two blocks. Therefore, since the recursive call constrains $\sigma(w_i)$ to be d and $d \in A$ for some block $A \times B \subset \Phi_\ell$, we have $D'_v \subseteq B \subset D_v$, which is what we set out to prove.

For the base case of the induction, take $j = 1$ so $\Phi_1 = R'_1$. We showed above that R'_1 has at least two blocks and that $R'_1 = H^M_{D_{v_1}, D_{v_2}}$. By arc-consistency, $\pi_1(R'_1) = D_{v_1}$ and $\pi_2(R'_1) = D_{v_2}$.

For the inductive step, take $j \in [\ell-2]$. Suppose that $\pi_1(\Phi_j) = D_{v_1}$, $\pi_2(\Phi_j) = D_{v_{j+1}}$ and $\Phi_j = \bigcup_{s=1}^{\lambda}(A_s \times A'_s)$ has at least two blocks. We have $\Phi_{j+1} = \Phi_j \circ R'_{j+1}$ and $R'_{j+1} = \bigcup_{t=1}^{\mu}(B_t \times B'_t)$ for some $\mu \geq 2$.

For every $d \in D_{v_1}$, there is a $d' \in D_{v_{j+1}}$ such that $(d, d') \in \Phi_j$ by the inductive hypothesis, and a $d'' \in D_{v_{j+1}}$ such that $(d', d'') \in D_{v_{j+2}}$, by arc-consistency. Therefore, $\pi_1(\Phi_{j+1}) = D_{v_1}$; a similar argument shows that $\pi_2(\Phi_{j+1}) = D_{v_{j+2}}$.

Suppose, towards a contradiction, that $\Phi_{j+1} = D_{v_1} \times D_{v_{j+2}}$. For this to be the case, we must have $A'_s \cap B_t \neq \emptyset$ for every $s \in \{1,2\}$ and $t \in [\mu]$. Now, let $D^*_{v_{j+1}} = D_{v_{j+1}} \setminus (A'_2 \cap B_2)$ and consider the relation

$$R = \{(d_1, d_3) \mid \text{for some } d_2 \in D^*_{v_{j+1}},\ (d_1, d_2) \in \Phi_j \text{ and } (d_2, d_3) \in R'_{j+1}\ \}.$$

Since $A_1' \subseteq D^*_{v_{j+1}}$ the non-empty sets $A_1' \cap B_1$ and $A_1' \cap B_2$ are both subsets of $D^*_{v_{j+1}}$ so $A_1 \times B_1' \subseteq R$ and $A_1 \times B_2' \subseteq R$. Similarly, $B_1 \subseteq D^*_{v_{j+1}}$, so $A_2' \cap B_1 \subseteq D^*_{v_{j+1}}$ so $A_2 \times B_1' \subseteq R$. However, $(A_2 \times B_2') \cap R = \emptyset$, so R is not rectangular. We will now derive a contradiction by showing that R is rectangular. Note that

$$R = H^M_{D_{v_1}, D_{v_2}} \circ \cdots \circ H^M_{D_{v_{j-1}}, D_{v_j}} \circ H^M_{D_{v_j}, D^*_{v_{j+1}}} \circ H^M_{D^*_{v_{j+1}}, D_{v_{j+2}}}$$

but this relation is rectangular because the hypothesis of the lemma guarantees that the sequence

$$D_{v_1}, \ldots, D_{v_j}, D^*_{v_{j+1}}, D_{v_{j+2}}$$

is not an $\mathcal{L}$-M-derectangularising sequence and all of the elements of this sequence are in $\mathcal{L}$, and $\{D_{v_1}, \ldots, D_{v_j}, D^*_{v_{j+1}}, D_{v_{j+2}}\}$ is M-purifying. ∎

5. Polynomial-time algorithms and the dichotomy theorem. Bulatov [3] showed that every problem of the form #CSP(Γ) is either in FP or #P-complete. Together with Proposition 15, his result immediately shows that a similar dichotomy exists for the special case of the problem #$\mathcal{L}$-M-PARTITIONS in which $\mathcal{L}$ is M-purifying and is closed under subsets. Our algorithmic work in Section 4 can be combined with Dyer and Richerby's explicit dichotomy for #CSP to obtain an explicit dichotomy for this special case of #$\mathcal{L}$-M-PARTITIONS. In particular, Lemma 18 gives a polynomial-time algorithm for the case in which there is no $\mathcal{L}$-M-derectangularising sequence. When there is such a sequence, $\Gamma_{\mathcal{L},M}$ is not "strongly rectangular" in the sense of [8]. It follows immediately that #CSP($\Gamma_{\mathcal{L},M}$) is #P-complete [8, Lemma 24] so #$\mathcal{L}$-M-PARTITIONS is also #P-complete by Proposition 15. In fact, the dichotomy for this special case does not require the full generality of Dyer and Richerby's dichotomy. If there is an $\mathcal{L}$-M-derectangularising sequence then it follows immediately from work of Bulatov and Dalmau [4, Theorem 2 and Corollary 3] that #CSP($\Gamma_{\mathcal{L},M}$) is #P-complete.

In this section we will move beyond the case in which $\mathcal{L}$ is M-purifying to provide a full dichotomy for the problem #$\mathcal{L}$-M-PARTITIONS. We will use two data structures: *sparse-dense partitions* and a representation of the set of *splits* of a bipartite graph. Similar data structures were used by Hell et al. [18] in their dichotomy for the #M-PARTITIONS problem for matrices of size at most 3-by-3.

5.1. Data Structures. We use two types of graph partition. The first is a special case of a sparse-dense partition [15] which is also called an (a,b)-graph with $a = b = 2$.

DEFINITION 19. *A bipartite–cobipartite partition of a graph G is a partition (B,C) of $V(G)$ such that B induces a bipartite graph and C induces the complement of a bipartite graph.*

LEMMA 20. *[15, Theorem 3.1; see also the remarks on (a,b)-graphs.] There is a polynomial-time algorithm for finding all bipartite–cobipartite partitions of a graph G.*

The second decomposition is based on certain sub-hypercubes called subcubes. For any finite set U, a *subcube* of $\{0,1\}^U$ is a subset of $\{0,1\}^U$ that is a Cartesian product of the form $\prod_{u\in U} S_u$ where $S_u \in \{\{0\},\{1\},\{0,1\}\}$ for each $u \in U$. We can also associate a subcube $\prod_{u\in U} S_u$ with the set of assignments $\sigma\colon U \to \{0,1\}$ such that $\sigma(u) \in S_u$ for all $u \in U$. Subcubes can be represented efficiently by listing the projections S_u.

DEFINITION 21. *Let $G = (U, U', E)$ be a bipartite graph, where U and U' are disjoint vertex sets, and $E \subseteq U \times U'$. A* subcube decomposition *of G is a list $U_1, \ldots, U_k$ of* *subcubes of* $\{0,1\}^U$ *and a list* $U_1', \dots, U_k'$ *of subcubes of* $\{0,1\}^{U'}$ *such that the following hold.*

(i) *The union* $(U_1 \times U_1') \cup \dots \cup (U_k \times U_k')$ *is the set of assignments* $\sigma \colon U \cup U' \to \{0,1\}$ *such that:*

$$\text{no edge } (u,u') \in E \text{ has } \sigma(u) = \sigma(u') = 0 \text{ and} \tag{2}$$

$$\text{no pair } (u,u') \in (U \times U') \setminus E \text{ has } \sigma(u) = \sigma(u') = 1. \tag{3}$$

(ii) *For distinct* $i,j \in [k]$, $U_i \times U_i'$ *and* $U_j \times U_j'$ *are disjoint.*

(iii) *For each* $i \in [k]$, *either* $|U_i| = 1$ *or* $|U_i'| = 1$ *(or both).*

Note that, although we require $U_i \times U_i'$ and $U_j \times U_j'$ to be disjoint for distinct $i,j \in [k]$, we allow $U_i \cap U_j \neq \emptyset$ as long as U_i' and U_j' are disjoint, and vice-versa. It is even possible that $U_i = U_j$, and indeed this will happen in our constructions below.

Lemma 22. *A subcube decomposition of a bipartite graph* $G = (U, U', E)$ *can be computed in polynomial time, with the subcubes represented by their projections.*

Proof. For a vertex x in a bipartite graph, let $\Gamma(x)$ be its set of neighbours and let $\overline{\Gamma}(x)$ be its set of non-neighbours on the other side of the graph. Thus, for $x \in U$, $\overline{\Gamma}(x) = U' \setminus \Gamma(x)$ and, for $x \in U'$, $\overline{\Gamma}(x) = U \setminus \Gamma(x)$.

Observe that we can write $\{0,1\}^n \setminus \{0\}^n$ as the disjoint union of n subcubes $\{0\}^{k-1} \times \{1\}^1 \times \{0,1\}^{n-k}$ with $1 \leq k \leq n$, and similarly for any other cube minus a single point.

We first deal with two base cases. If G has no edges, then the set of assignments $\sigma \colon U \cup U' \to \{0,1\}$ satisfying (2) and (3) is the disjoint union of

$$\{0\}^U \times \{0\}^{U'}, \quad (\{0,1\}^U \setminus \{0\}^U) \times \{0\}^{U'}, \quad \text{and} \quad \{0\}^U \times (\{0,1\}^{U'} \setminus \{0\}^{U'}).$$

The second and third terms can be decomposed into subcubes as described above to produce the output. Similarly, if G is a complete bipartite graph, then the set of assignments satisfying (2) and (3) is the disjoint union of

$$\{1\}^U \times \{1\}^{U'}, \quad (\{0,1\}^U \setminus \{1\}^U) \times \{1\}^{U'}, \quad \text{and} \quad \{1\}^U \times (\{0,1\}^{U'} \setminus \{1\}^{U'}).$$

If neither of these cases occurs then there is a vertex x such that neither $\Gamma(x)$ nor $\overline{\Gamma}(x)$ is empty. If possible, choose $x \in U$; otherwise, choose $x \in U'$. To simplify the description of the algorithm, we assume that $x \in U$; the other case is symmetric. We consider separately the assignments where $\sigma(x) = 0$ and those where $\sigma(x) = 1$. Note that, for any assignment, if $\sigma(y) = 0$ for some vertex y, then $\sigma(z) = 1$ for all $z \in \Gamma(y)$ and, if $\sigma(y) = 1$, then $\sigma(z) = 0$ for all $z \in \overline{\Gamma}(y)$. Applying this iteratively, setting $\sigma(x) = c$ for $c \in \{0,1\}$ also determines the value of σ on some set $S_{x=c} \subseteq U \cup U'$ of vertices.

Thus, we can compute a subcube decomposition for G recursively. First, compute $S_{x=0}$ and $S_{x=1}$. Then, recursively compute subcube decompositions of $G - S_{x=0}$ (the graph formed from G by deleting the vertices in $S_{x=0}$) and $G - S_{x=1}$. Translate these subcube decompositions into a subcube decomposition of G by extending each subcube $(U_i \times U_i')$ of $G - S_{x=c}$ to a subcube $(V_i \times V_i')$ of G whose restriction to $G - S_{x=c}$ is $(U_i \times U_i')$ and whose restriction to $S_{x=c}$ is an assignment σ with $\sigma(x) = c$ (in fact, all assignments that set x to c agree on the set $S_{x=c}$, by construction).

It remains to show that the algorithm runs in polynomial time. The base cases are clearly computable in polynomial time, as are the individual steps in the recursive cases, so we only need to show that the number of recursive calls is polynomially bounded. At the recursive step, we only choose $x \in U'$ when $E(G) = U'' \times U'$ for some proper subset $\emptyset \subset U'' \subset U$ and, in this case, the two recursive calls are to base cases. Since each recursive call when $x \in U$ splits U' into disjoint subsets, there can be at most $|U'| - 1$ such recursive calls, so the total number of recursive calls is linear in $|V(G)|$. □

5.2. Reduction to a problem with M-purifying lists. Our algorithm for counting list M-partitions uses the data structures from Section 5.1 to reduce problems where $\mathcal{L}$ is not M-purifying to problems where it is (which we already know how to solve from Sections 3 and 4). The algorithm is defined recursively on the set $\mathcal{L}$ of allowed lists. The algorithm for parameters $\mathcal{L}$ and M calls the algorithm for $\mathcal{L}_i$ and M where $\mathcal{L}_i$ is a subset of $\mathcal{L}$. The base case arises when $\mathcal{L}_i$ is M-purifying.

We will use the following computational problem to reduce #$\mathcal{L}$-M-PARTITIONS to a collection of problems #$\mathcal{L}'$-M-PARTITIONS that are, in a sense, disjoint.
Name. #$\mathcal{L}$-M-PURIFY.
Instance. A graph G and a function $L\colon V(G) \to \mathcal{L}$.
Output. Functions $L_1, \dots, L_t\colon V(G) \to \mathcal{L}$ such that

(i) for each $i \in [t]$, the set $\{L_i(v) \mid v \in V(G)\}$ is M-purifying,
(ii) for each $i \in [t]$ and $v \in V(G)$, $L_i(v) \subseteq L(v)$, and
(iii) each M-partition of G that respects L respects exactly one of the functions $L_1, \dots, L_t$.

We will give an algorithm for solving the problem #$\mathcal{L}$-M-PURIFY in polynomial time when there is no $\mathcal{L}$-M-derectangularising sequence of length exactly 2. The following computational problem will be central to the inductive step.
Name. #$\mathcal{L}$-M-PURIFY-STEP.
Instance. A graph G and a function $L\colon V(G) \to \mathcal{L}$.
Output. Functions $L_1, \dots, L_k\colon V(G) \to \mathcal{L}$ such that

(i) for each $i \in [k]$ and $v \in V(G)$, $L_i(v) \subseteq L(v)$,
(ii) every M-partition of G that respects L respects exactly one of $L_1, \dots, L_k$, and
(iii) for each $i \in [k]$, there is a $W \in \mathcal{L}$ which is inclusion-maximal in $\mathcal{L}$ but does not occur in the image of L_i.

Note that we can trivially produce a solution to the problem #$\mathcal{L}$-M-PURIFY-STEP by letting $L_1, \dots, L_k$ be an enumeration of all possible functions such that all lists $L_i(v)$ have size 1 and satisfy $L_i(v) \subseteq L(v)$. Such a function L_i corresponds to an assignment of vertices to parts so there is either exactly one L_i-respecting M-partition or none, which means that every L-respecting M-partition is L_i-respecting for exactly one i. However, this solution is exponentially large in $|V(G)|$ and we are interested in solutions that can be produced in polynomial time. Also, if $L(v) = \emptyset$ for some vertex v, the algorithm is entitled to output an empty list, since no M-partition respects L.

The following definition extends rectangularity to $\{0, 1, *\}$-matrices and is used in our proof.

DEFINITION 23. *A matrix $M \in \{0, 1, *\}^{X \times Y}$ is* $*$-rectangular *if the relation $H^M_{X,Y}$ is rectangular.* Thus, M is $*$-rectangular if and only if $M_{x,y} = M_{x',y} = M_{x,y'} = *$ implies that $M_{x',y'} = *$ for all $x, x' \in X'$ and all $y, y' \in Y''$.

We will show in Lemma 24 that the function #$\mathcal{L}$-M-PURIFY-STEP from Algorithm 4 is a polynomial-time algorithm for the problem #$\mathcal{L}$-M-PURIFY-STEP whenever $\mathcal{L}$ is not M-purifying and there is no length-2 $\mathcal{L}$-M-derectangularising sequence. Note that a length-2 $\mathcal{L}$-M-derectangularising sequence is a pair $X, Y \in \mathcal{L}$ such

Algorithm 4 A polynomial-time algorithm for the problem #$\mathcal{L}$-M-PURIFY-STEP when $\mathcal{L} \subseteq \mathcal{P}(D)$ is subset-closed, $\mathcal{L}$ is not M-purifying and there is no length-2 $\mathcal{L}$-M-derectangularising sequence. The input is a pair (G, L) with $V(G) = \{v_1, \ldots, v_n\}$.

```
function #L-M-PURIFY-STEP(G,L)
    if there is a v_i ∈ V(G) with L(v_i) = ∅ then return the empty sequence
    else if  there are X, Y ∈ L, a, b ∈ X, and d ∈ Y
                such that M_{a,d} = 0 and M_{b,d} = 1 then
        Run Algorithm 5    /* Case 1 */
    else if there is an X ∈ L such that M|_{X×X} is not pure then
        Run Algorithm 6    /* Case 2 */
    else
        Run Algorithm 7    /* Case 3 */
```

Algorithm 5 Case 1 in Algorithm 4.

```
Choose X, Y ∈ L, a, b ∈ X, and d ∈ Y
such that M_{a,d} = 0, M_{b,d} = 1 and X and Y are inclusion-maximal in L
for i ∈ [n] do
    L_i(v_i) ← L(v_i) ∩ {d}
    for j < i do
        if (v_i, v_j) ∈ E(G) then
            L_i(v_j) ← {d' ∈ L(v_j) | d' ≠ d and M_{d,d'} ≠ 0}
        else
            L_i(v_j) ← {d' ∈ L(v_j) | d' ≠ d and M_{d,d'} ≠ 1}
    for j > i do
        if (v_i, v_j) ∈ E(G) then
            L_i(v_j) ← {d' ∈ L(v_j) | M_{d,d'} ≠ 0}
        else
            L_i(v_j) ← {d' ∈ L(v_j) | M_{d,d'} ≠ 1}
    L_{n+1}(v_i) ← L(v_i) \ {d}
return L_1, ..., L_{n+1} (of course, if we have L_i(v) = ∅ for any i and v then L_i can
be omitted from the output)
```

that $M|_{X\times Y}$, $M|_{X\times X}$ and $M|_{Y\times Y}$ are pure and $M|_{X\times Y}$ is not $*$-rectangular. If $\mathcal{L} \neq \mathcal{P}(D)$, it is possible that a matrix that is not $*$-rectangular has no length-2 $\mathcal{L}$-M-derectangularising sequence. For example, let $D = \{1, 2, 3\}$ and $\mathcal{L} = \mathcal{P}(\{1, 2\})$ and let $M_{3,3} = 0$ and $M_{i,j} = *$ for every other pair $(i, j) \in D^2$. M is not $*$-rectangular but this fact is not witnessed by any submatrix $M|_{X\times Y}$ for $X, Y \in \mathcal{L}$.

LEMMA 24. *Let M be a symmetric matrix in $\{0, 1, *\}^{D\times D}$ and let $\mathcal{L} \subseteq \mathcal{P}(D)$ be subset-closed. If $\mathcal{L}$ is not M-purifying and there is no length-2 $\mathcal{L}$-M-derectangularising sequence, then Algorithm 4 is a polynomial-time algorithm for the problem #$\mathcal{L}$-M-PURIFY-STEP.*

Proof. We consider an instance (G, L) of the problem #$\mathcal{L}$-M-PURIFY-STEP with $V(G) = \{v_1, \ldots, v_n\}$. If there is a $v_i \in V(G)$ with $L(v_i) = \emptyset$ then no M-partition of G respects L, so the output is correct. Otherwise, we consider the three cases that can occur in the execution of the algorithm.

Case 1. In this case column d of $M|_{X\times Y}$ contains both a zero and a one. Equivalently, row d of $M|_{Y\times X}$ does. Algorithm 5 groups the set of M-partitions of G that

Algorithm 6 Case 2 in Algorithm 4.

Choose $X \in \mathcal{L}$ such that $M|_{X\times X}$ is not pure and X is inclusion-maximal in $\mathcal{L}$
Let $X_0 \subseteq X$ be the set of rows of $M|_{X\times X}$ that contain a 0
$X_1 \leftarrow X \setminus X_0$
$V_X \leftarrow \{v_j \in V(G) \mid L(v_j) = X\}$
if $V_X = \emptyset$ **then return** L
else
  Use the algorithm promised in Lemma 20 to compute the list
    $(B_1, C_1), \ldots, (B_k, C_k)$ of all bipartite–cobipartite partitions of $G[V_X]$
  for $i \in [k], j \in [n]$ **do**
    if $v_j \notin V_X$ **then**
      $L_i(v_j) \leftarrow L(v_j)$
    else if $v_j \in B_i$ **then**
      $L_i(v_j) \leftarrow X_0$
    else /* $v_j \in C_i$ */
      $L_i(v_j) \leftarrow X_1$
  return $L_1, \ldots, L_k$

respect L, based on the first vertex that is placed in part d. For $i \in [n]$, L_i requires that v_i is placed in part d and $v_1, \ldots, v_{i-1}$ are not in part d; L_{n+1} requires that part d is empty. Thus, no M-partition can respect more than one of the L_i. Now consider an L-respecting M-partition $\sigma\colon V(G) \to D$ and suppose that i is minimal such that $\sigma(v_i) = d$. We claim that σ respects L_i. We have $\sigma(v_i) = d$, as required. For $j \neq i$, we must have $\sigma(v_j) \in L(v_j)$ since σ respects L and we must have $M_{d,\sigma(v_j)} \neq 1$ if $(v_i, v_j) \notin E(G)$ and $M_{d,\sigma(v_j)} \neq 0$ if $(v_i, v_j) \in E(G)$, since σ is an M-partition. In addition, by construction, $\sigma(v_j) \neq d$ if $j < i$. Therefore, σ respects L_i. A similar argument shows that σ respects L_{n+1} if $\sigma(v) \neq d$ for all $v \in V(G)$. Hence, any M-partition that respects L respects exactly one of the L_i.

Finally, we show that, for each $i \in [n+1]$, there is a set W which is inclusion-maximal in $\mathcal{L}$ and is not in the image of L_i. For $i \in [n]$, we cannot have both a and b in $L_i(v_j)$ for any v_j, so X is not in the image of L_i. Y contains d, so Y is not in the image of L_{n+1}.

Case 2. In this case, every row of $M|_{X_0\times X}$ contains a 0, while every row of $M|_{X_1\times X}$ fails to contain a zero. Since $M|_{X\times X}$ is not pure, but no row of $M|_{X\times X}$ contains both a zero and a one (since we are not in Case 1), X_0 and X_1 are non-empty. Note that $M|_{X_0\times X_0}$ and $M|_{X_1\times X_1}$ are both pure, but every entry of $M|_{X_0\times X_1}$ is a $*$.

If $V_X = \emptyset$ then X is an inclusion-maximal member of $\mathcal{L}$ that is not in the image of L, so the output of Algorithm 6 is correct. Otherwise, $(B_1, C_1), \ldots, (B_k, C_k)$ is the list containing all partitions (B, C) of V_X such that B induces a bipartite graph in G and C induces the complement of a bipartite graph. The algorithm returns $L_1, \ldots, L_k$. X is not in the image of any L_i so, to show that $\{L_1, \ldots, L_k\}$ is a correct output for the problem #$\mathcal{L}$-M-PURIFY-STEP, we just need to show that every M-partition of G that respects L respects exactly one of $L_1, \ldots, L_k$. For $i \neq i'$, $(B_i, C_i) \neq (B_{i'}, C_{i'})$ so there is at least one vertex v_j such that $L_i(v_j) = X_0$ and $L_{i'}(v_j) = X_1$ or vice-versa. Since X_0 and X_1 are disjoint, no M-partition can simultaneously respect L_i and $L_{i'}$. It remains to show that every M-partition respects at least one of $L_1, \ldots, L_k$. To do this, we deduce two structural properties of $M|_{X\times X}$.

First, we show that $M|_{X\times X}$ has no $*$ on its diagonal. Suppose towards a contra-

Algorithm 7 Case 3 in Algorithm 4.

Choose inclusion-maximal X and Y in $\mathcal{L}$ so that $M|_{X\times Y}$ is not pure
Let $X_0 \subseteq X$ be the set of rows of $M|_{X\times Y}$ that contain a 0
$X_1 \leftarrow X \setminus X_0$
Let $Y_0 \subseteq Y$ be the set of columns of $M|_{X\times Y}$ that contain a 0
$Y_1 \leftarrow Y \setminus Y_0$
$V_X \leftarrow \{v_j \in V(G) \mid L(v_j) = X\}$
$V_Y \leftarrow \{v_j \in V(G) \mid L(v_j) = Y\}$
if $V_X = \emptyset$ or $V_Y = \emptyset$ **then return** L
else
  Let E be the set of edges of G between V_X and V_Y
  Use the algorithm promised in Lemma 22 to produce a subcube decomposition $(U_1, U_1'), \dots, (U_k, U_k')$ of (V_X, V_Y, E)
  for $i \in [k], j \in [n]$ **do**
    if $v_j \in V_X$ and the projection of U_i on v_j is $\{0\}$ **then**
      $L_i(v_j) \leftarrow X_0$
    else if $v_j \in V_X$ and the projection of U_i on v_j is $\{1\}$ **then**
      $L_i(v_j) \leftarrow X_1$
    else if $v_j \in V_Y$ and the projection of U_i' on v_j is $\{0\}$ **then**
      $L_i(v_j) \leftarrow Y_0$
    else if $v_j \in V_Y$ and the projection of U_i' on v_j is $\{1\}$ **then**
      $L_i(v_j) \leftarrow Y_1$
    else
      $L_i(v_j) \leftarrow L(v_j)$
return $L_1, \dots, L_k$

diction that $M_{d,d} = *$ for some $d \in X$. If $d \in X_0$, then, for each $d' \in X_1$, $M_{d,d'} = M_{d',d} = *$ because, as noted above, every entry of $M|_{X_0\times X_1}$ is a $*$. Therefore, the 2×2 matrix $M' = M|_{\{d,d'\}\times\{d,d'\}}$ contains at least three $*$s so it is pure. $\{d,d'\} \subseteq X \in \mathcal{L}$ so, by the hypothesis of the lemma, the length-2 sequence $\{d,d'\},\{d,d'\}$ is not $\mathcal{L}$-M-derectangularising, so M' must be $*$-rectangular, so $M_{d',d'} = *$ for all $d' \in X_1$. Similarly, if $M_{d',d'} = *$ for some $d' \in X_1$, then $M_{d,d} = *$ for all $d \in X_0$. Therefore, if $M|_{X\times X}$ has a $*$ on its diagonal, every entry on the diagonal is $*$. But M contains a 0, say $M_{i,j} = 0$ with $i,j \in X_0$. For any $k \in X_1$,

$$M|_{\{i,j\}\times\{j,k\}} = \begin{pmatrix} 0 & * \\ * & * \end{pmatrix},$$

so the length-2 sequence $\{i,j\},\{j,k\}$ is $\mathcal{L}$-M-derectangularising, contradicting the hypothesis of the lemma (note that $\{i,j\},\{j,k\} \subseteq X \in \mathcal{L}$).

Second, we show that there is no sequence $d_1, \dots, d_\ell \in X_0$ of odd length such that

$$M_{d_1,d_2} = M_{d_2,d_3} = \cdots = M_{d_{\ell-1},d_\ell} = M_{d_\ell,d_1} = *.$$

Suppose for a contradiction that such a sequence exists. Note that $M|_{X_0\times X_0}$ is $*$-rectangular since X_0, X_0 is not an $\mathcal{L}$-M-derectangularising sequence and $M|_{X_0\times X_0}$ is pure since Case 1 does not apply. We will show by induction that for every non-negative integer $\kappa \le (\ell-3)/2$, $M_{d_1,d_{\ell-2\kappa-2}} = *$. This gives a contradiction by taking $\kappa = (\ell-3)/2$ since $M_{d_1,d_1} = *$ and we have already shown that $M|_{X_0\times X_0}$ has no $*$

on its diagonal. For every κ, the argument follows by considering the matrix $M_\kappa = M|_{\{d_1,d_{\ell-2\kappa-1}\}\times\{d_{\ell-2\kappa-2},d_{\ell-2\kappa}\}}$. The definition of the sequence $d_1,\ldots,d_\ell$ together with the symmetry of M guarantees that both entries in row $d_{\ell-2\kappa-1}$ of M_κ are $*$. It is also true that $M_{d_1,d_{\ell-2\kappa}} = *$: If $\kappa = 0$ then this follows from the definition of the sequence; otherwise it follows by induction. The fact that $M_{d_1,d_{\ell-2\kappa-2}} = *$ then follows by $*$-rectangularity.

This second structural property implies that, for any $M|_{X\times X}$-partition of $G[V_X]$, the graph induced by vertices assigned to X_0 has no odd cycles, and is therefore bipartite. Similarly, the vertices assigned to X_1 induce the complement of a bipartite graph. Therefore, any M-partition of G that respects L must respect at least one of the $L_1,\ldots,L_k$, so it respects exactly one of them, as required.

Case 3. Since Cases 1 and 2 do not apply and $\mathcal{L}$ is not M-purifying, there are distinct $X, Y \in \mathcal{L}$ such that X and Y are inclusion-maximal in $\mathcal{L}$ and $M|_{X\times Y}$ is not pure. As in the previous case, the sets X_0, X_1, Y_0 and Y_1 are all non-empty.

If either V_X or V_Y is empty then either X or Y is an inclusion-maximal set in $\mathcal{L}$ that is not in the image of L so the output of Algorithm 7 is correct. Otherwise, $(U_1, U'_1), \ldots, (U_k, U'_k)$ is a subcube decomposition of the bipartite subgraph (V_X, V_Y, E). The U_is are subcubes of $\{0,1\}^{V_X}$ and the U'_is are subcubes of $\{0,1\}^{V_Y}$. The algorithm returns $L_1,\ldots,L_k$.

Note that if $|U'_i| = 1$ then Y is not in the image of L_i. Similarly, if $|U'_i| > 1$ but $|U_i| = 1$ then X is not in the image of L_i. The definition of subcube decompositions guarantees that, for every i, at least one of these is the case. To show this definition of $L_1,\ldots,L_k$ is a correct output for the problem #$\mathcal{L}$-M-PURIFY-STEP, we must show that any M-partition of G that respects L also respects exactly one L_i. Since the sets in $\{U_i \times U'_i \mid i \in [k]\}$ are disjoint subsets of $\{0,1\}^{V_X\cup V_Y}$, any M-partition of G that respects L respects at most one L_i so it remains to show that every M-partition of G respects at least one L_i. To do this, we deduce two structural properties of $M|_{X\times Y}$.

First, we show that every entry of $M|_{X_0\times Y_0}$ is 0. The definition of X_0 guarantees that every row of $M|_{X_0\times Y_0}$ contains a 0. Since Case 1 does not apply, and M is symmetric, every entry of $M|_{X_0\times Y_0}$ is either 0 or $*$. Suppose for a contradiction that $M_{i,j} = *$ for some $(i,j) \in X_0 \times Y_0$. Pick $i' \in X_1$. For any $j' \in Y_0 \setminus \{j\}$ we have $M_{i,j} = M_{i',j} = M_{i',j'} = *$, so by $*$-rectangularity of $M|_{X\times Y_0}$ we have $M_{i,j'} = *$. Thus, every entry of $M|_{\{i\}\times Y_0}$ is $*$, so there is a $*$ in every Y_0-indexed column of M. By the same argument, swapping the roles of X and Y, every entry in $M|_{X_0\times Y_0}$ is $*$, contradicting the fact that $M|_{X\times Y}$ contains a 0 since $M|_{X\times Y}$ is not pure.

Second, a similar argument shows that every entry of $M|_{X_1\times Y_1}$ is 1.

Thus for all M-partitions σ of G respecting L, for all $x \in V_X$ and $y \in V_Y$, if $(x,y) \in E$ then $(\sigma(x),\sigma(y)) \notin X_0 \times Y_0$ while if $(x,y) \notin E$ then $(\sigma(x),\sigma(y)) \notin X_1 \times Y_1$. Using the definition of subcube decompositions, this shows that any M-partition of G respecting L respects some L_i. □

We can now give an algorithm for the problem #$\mathcal{L}$-M-PURIFY. The algorithm consists of the function #$\mathcal{L}$-M-PURIFY, which is defined in Algorithm 8 for the trivial case in which $\mathcal{L}$ is M-purifying and in Algorithm 9 for the case in which it is not. Note that for any fixed $\mathcal{L}$ and M the algorithm is defined either in Algorithm 8 or in Algorithm 9 and the function #$\mathcal{L}$-M-PURIFY is not recursive. However, the *definition* is recursive, so the function #$\mathcal{L}$-M-PURIFY defined in Algorithm 9 does make a call to a function #$\mathcal{L}_i$-M-PURIFY for some $\mathcal{L}_i$ which is smaller than $\mathcal{L}$. The function #$\mathcal{L}_i$-M-PURIFY is in turn defined in Algorithm 8 or Algorithm 9. The correctness of the algorithm follows from the definition of the problem. The following lemma

Algorithm 8 A trivial algorithm for the problem $\#\mathcal{L}$-M-PURIFY for the case in which $\mathcal{L}$ is M-purifying.

function $\#\mathcal{L}$-M-PURIFY(G,L) **return** L

Algorithm 9 A polynomial-time algorithm for the problem $\#\mathcal{L}$-M-PURIFY when $\mathcal{L} \subseteq \mathcal{P}(D)$ is subset-closed and is not M-purifying and there is no length-2 $\mathcal{L}$-M-derectangularising sequence. This algorithm calls the function $\#\mathcal{L}$-M-PURIFY-STEP from Algorithm 4. It also calls the function $\#\mathcal{L}_i$-M-PURIFY for various lists $\mathcal{L}_i$ which are shorter than $\mathcal{L}$. These functions are defined inductively in Algorithm 8 and here.

function $\#\mathcal{L}$-M-PURIFY(G,L)
  // $\emptyset \in \mathcal{L}$ since $\mathcal{L}$ is subset-closed. Since $\mathcal{L}$ is not M-purifying, $\mathcal{L} \neq \{\emptyset\}$,
  // hence $|\mathcal{L}| > 1$.
  Let B be the empty sequence of list functions
  $L_1, \ldots, L_k \leftarrow \#\mathcal{L}$-$M$-PURIFY-STEP$(G, L)$
  for $i \in [k]$ **do**
    $\mathcal{L}_i \leftarrow \bigcup_{v \in V(G)} \mathcal{P}(L_i(v))$
    $L'_1, \ldots, L'_j \leftarrow \#\mathcal{L}_i$-$M$-PURIFY$(G, L_i)$
    Add $L'_1, \ldots, L'_j$ to B
  return B

bounds the running time.

LEMMA 25. *Let $M \in \{0, 1, *\}^{D \times D}$ be a symmetric matrix and let $\mathcal{L} \subseteq \mathcal{P}(D)$ be subset-closed. If there is no length-2 $\mathcal{L}$-M-derectangularising sequence, then the function $\#\mathcal{L}$-M-PURIFY as defined in Algorithms 8 and 9 is a polynomial-time algorithm for the problem $\#\mathcal{L}$-M-PURIFY.*

Proof. Note that $\mathcal{L}$ is a fixed parameter of the problem $\#\mathcal{L}$-M-PURIFY — it is not part of the input. The proof is by induction on $|\mathcal{L}|$. If $|\mathcal{L}| = 1$ then $\mathcal{L} = \{\emptyset\}$ so it is M-purifying. In this case, function $\#\mathcal{L}$-M-PURIFY is defined in Algorithm 8. It is clear that it is a polynomial-time algorithm for the problem $\#\mathcal{L}$-M-PURIFY.

For the inductive step suppose that $|\mathcal{L}| > 1$. If $\mathcal{L}$ is M-purifying then function $\#\mathcal{L}$-M-PURIFY is defined in Algorithm 8 and again the result is trivial. Otherwise, function $\#\mathcal{L}$-M-PURIFY is defined in Algorithm 9. Note that $\mathcal{L} \subseteq \mathcal{P}(D)$ is subset-closed and there is no length-2 $\mathcal{L}$-M-derectangularising sequence. From this, we can conclude that, for any subset-closed subset $\mathcal{L}'$ of $\mathcal{L}$, there is no length-2 $\mathcal{L}'$-M-derectangularising sequence. So we can assume by the inductive hypothesis that for all subset-closed $\mathcal{L}' \subset \mathcal{L}$, the function $\#\mathcal{L}'$-M-PURIFY runs in polynomial time.

The result now follows from the fact that the function $\#\mathcal{L}$-M-PURIFY-STEP runs in polynomial time (as guaranteed by Lemma 24) and from the fact that each $\mathcal{L}_i$ is a strict subset of $\mathcal{L}$, which follows from the definition of problem $\#\mathcal{L}$-M-PURIFY-STEP. Each M-partition that respects L respects exactly one of $L_1, \ldots, L_k$ and, hence, it respects exactly one of the list functions that is returned. □

5.3. Algorithm for $\#\mathcal{L}$-M-PARTITIONS and proof of the dichotomy. We can now present our algorithm for the problem $\#\mathcal{L}$-M-PARTITIONS. The algorithm consists of the function $\#\mathcal{L}$-M-PARTITIONS which is defined in Algorithm 10 for the case in which $\mathcal{L}$ is M-purifying and in Algorithm 11 when it is not.

LEMMA 26. *Let $M \in \{0, 1, *\}^{D \times D}$ be a symmetric matrix and let $\mathcal{L} \subseteq \mathcal{P}(D)$ be subset-closed. If there is no $\mathcal{L}$-M-derectangularising sequence, then the function $\#\mathcal{L}$-*

Algorithm 10 A polynomial-time algorithm for the problem $\#\mathcal{L}$-M-PARTITIONS when $\mathcal{L}$ is subset-closed and M-purifying and there is no $\mathcal{L}$-M-derectangularising sequence.

```
function #L-M-PARTITIONS(G,L)
    (V,C) ← the instance of #CSP(Γ_{L,M}) obtained by applying the polynomial-
      time Turing reduction from Proposition 15 to the input (G,L)
    return AC(V,C) where AC is the function from Algorithm 3
```

Algorithm 11 A polynomial-time algorithm for the problem $\#\mathcal{L}$-M-PARTITIONS when $\mathcal{L}$ is subset-closed and not M-purifying and there is no $\mathcal{L}$-M-derectangularising sequence. The algorithm calls the function $\#\mathcal{L}$-M-PURIFY(G, L) from Algorithm 9.

```
function #L-M-PARTITIONS(G,L)
    L_1, ..., L_t ← #L-M-PURIFY(G,L)
    Z ← 0
    for i ∈ [t] do
        L_i ← ⋃_{v∈V(G)} P(L_i(v))
        (V,C_i) ← the instance of #CSP(Γ_{L_i,M}) obtained by applying the
          polynomial-time Turing reduction from Proposition 15 to the input (G,L_i)
        Z_i ← AC(V,C_i) where AC is the function from Algorithm 3
        Z ← Z + Z_i
    return Z
```

M-PARTITIONS as defined in Algorithms 10 and 11 is a polynomial-time algorithm for the problem $\#\mathcal{L}$-M-PARTITIONS.

Proof. If $\mathcal{L}$ is M-purifying then the function $\#\mathcal{L}$-M-PARTITIONS is defined in Algorithm 10. Proposition 15 shows that the reduction in Algorithm 10 to a CSP instance is correct and takes polynomial time. The CSP instance can be solved by the function AC in Algorithm 3, whose running time is shown to be polynomial in Lemma 18.

If $\mathcal{L}$ is not M-purifying then the function $\#\mathcal{L}$-M-PARTITIONS is defined in Algorithm 11. Lemma 25 guarantees that the function $\#\mathcal{L}$-M-PURIFY is a polynomial-time algorithm for the problem $\#\mathcal{L}$-M-PURIFY. If the list $L_1, \ldots, L_t$ is empty then there is no M-partition of G that respects L so it is correct that the function $\#\mathcal{L}$-M-PARTITIONS returns 0. Otherwise, we know from the definition of the problem $\#\mathcal{L}$-M-PURIFY that

(i) functions $L_1, \ldots, L_t$ are from $V(G)$ to $\mathcal{L}$,
(ii) for each $i \in [t]$, the set $\{L_i(v) \mid v \in V(G)\}$ is M-purifying,
(iii) for each $i \in [t]$ and $v \in V(G)$, $L_i(v) \subseteq L(v)$, and
(iv) each M-partition of G that respects L respects exactly one of $L_1, \ldots, L_t$.

The desired result is now the sum, over all $i \in [t]$, of the number of M-partitions of G that respect L_i. Since the list $L_1, \ldots, L_t$ is generated in polynomial time, t is bounded by some polynomial in $|V(G)|$.

Now, for each $i \in [t]$, $\mathcal{L}_i$ is a subset-closed subset of $\mathcal{L}$. Since there is no $\mathcal{L}$-M-derectangularising sequence, there is also no $\mathcal{L}_i$-M-derectangularising sequence. Also, $\mathcal{L}_i$ is M-purifying. Thus, the argument that we gave for the purifying case shows that Z_i is the desired quantity. □

We can now combine our results to establish our dichotomy for the problem $\#\mathcal{L}$-

M-PARTITIONS.

THEOREM 9. *Let M be a symmetric matrix in $\{0,1,*\}^{D\times D}$ and let $\mathcal{L}\subseteq\mathcal{P}(D)$ be subset-closed. If there is an $\mathcal{L}$-M-derectangularising sequence, then the problem #$\mathcal{L}$-M-PARTITIONS is #P-complete. Otherwise, it is in* FP.

Proof. Suppose that there is an $\mathcal{L}$-M-derectangularising sequence $D_1,\dots,D_k$. Recall (from Definition 2) the definition of the subset-closure $\mathcal{S}(\mathcal{L}'')$ of a set $\mathcal{L}''\subseteq\mathcal{P}(D)$. Let

$$\mathcal{L}'=\mathcal{S}(\{D_1,\dots,D_k\})\,.$$

Since $\{D_1,\dots,D_k\}$ is M-purifying, so is $\mathcal{L}'$, which is also subset-closed. It follows that $\Gamma_{\mathcal{L}',M}$ is well defined (see Definition 12) and contains each of the relations $H^M_{D_1,D_2},\dots,H^M_{D_{k-1},D_k}$ (and possibly others). Since $H^M_{D_1,D_2}\circ H^M_{D_2,D_3}\circ\cdots\circ H^M_{D_{k-1},D_k}$ is not rectangular, #CSP($\Gamma_{\mathcal{L}',M}$) is #P-complete [4, Theorem 2 and Corollary 3] (see also [8, Lemma 24]). By Proposition 15, the problem #$\mathcal{L}'$-M-PARTITIONS is #P-complete so the more general problem #$\mathcal{L}$-M-PARTITIONS is also #P-complete. On the other hand, if there is no $\mathcal{L}$-M-derectangularising sequence, then the result follows from Lemma 26. □

6. Complexity of the dichotomy criterion. The dichotomy established in Theorem 9 is that, if there is an $\mathcal{L}$-M-derectangularising sequence, then the problem #$\mathcal{L}$-M-PARTITIONS is #P-complete; otherwise, it is in FP. This section addresses the computational problem of determining which is the case, given $\mathcal{L}$ and M.

The following lemma will allow us to show that the problem EXISTSDERECTSEQ (the problem of determining whether there is an $\mathcal{S}(\mathcal{L})$-M-derectangularising sequence, given $\mathcal{L}$ and M) and the related problem MATRIXHASDERECTSEQ (the problem of determining whether there is a $\mathcal{P}(D)$-M-derectangularising sequence, given M) are both in NP. Note that, for this "meta-problem", $\mathcal{L}$ and M are the inputs whereas, previously, we have regarded them as fixed parameters.

LEMMA 28. *Let $M\in\{0,1,*\}^{D\times D}$ be symmetric, and let $\mathcal{L}\subseteq\mathcal{P}(D)$ be subset-closed. If there is an $\mathcal{L}$-M-derectangularising sequence, then there is one of length at most* $512(|D|^3+1)$.

Proof. Pick an $\mathcal{L}$-M-derectangularising sequence $D_1,\dots,D_k$ with k minimal; we will show that $k\le 512(|D|^3+1)$. Define

$$R=H^M_{D_1,D_2}\circ H^M_{D_2,D_3}\circ\cdots\circ H^M_{D_{k-1},D_k}.$$

Note that $R\subseteq D_1\times D_k$. By the definition of derectangularising sequence, there are $a,a'\in D_1$ and $b,b'\in D_k$ such that (a,b), (a',b) and (a,b') are all in R but $(a',b')\notin R$. So there exist

$$(x_1,\dots,x_k),(y_1,\dots,y_k),(z_1,\dots,z_k)\in D_1\times\cdots\times D_k$$

with $(x_1,x_k)=(a,b)$, $(y_1,y_k)=(a',b)$ and $(z_1,z_k)=(a,b')$ such that $M_{x_i,x_{i+1}}=M_{y_i,y_{i+1}}=M_{z_i,z_{i+1}}=*$ for every $i\in[k-1]$ but, for any $(w_1,\dots,w_k)\in D_1\times\cdots\times D_k$ with $(w_1,w_k)=(a',b')$, there is an $i\in[k-1]$ such that $M_{w_i,w_{i+1}}\ne *$.

Setting $D'_i=\{x_i,y_i,z_i\}$ for each i gives an $\mathcal{L}$-M-derectangularising sequence $D'_1,\dots,D'_k$ with $|D'_i|\le 3$ for each $1\le i\le k$. (Note that any submatrix of a pure matrix is pure.) For all $1\le s<t\le k$ define

$$R_{s,t}=H^M_{D'_s,D'_{s+1}}\circ H^M_{D'_{s+1},D'_{s+2}}\circ\cdots\circ H^M_{D'_{t-1},D'_t}.$$

Since $D'_1, \ldots, D'_k$ is $\mathcal{L}$-M-derectangularising, $R_{1,k}$ is not rectangular but, by the minimality of k, every other $R_{s,t}$ is rectangular. Note also that no $R_{s,t} = \emptyset$ since, if that were the case, we would have $R_{1,k} = \emptyset$, which is rectangular.

Suppose for a contradiction that $k > 512(|D|^3 + 1)$. There are at most $|D|^3 + 1$ subsets of D with size at most three, so there are indices $1 \leq i_0 < i_1 < i_2 < \cdots < i_{512} \leq k$ such that $D'_{i_0} = \cdots = D'_{i_{512}}$. There are at most $2^{|D'_{i_0}|^2} - 1 \leq 2^9 - 1 = 511$ non-empty binary relations on D'_{i_0}, so $R_{i_0,i_m} = R_{i_0,i_n}$ for some $1 \leq m < n \leq 512$. Since $R_{1,k}$ is not rectangular,

$$R_{1,k} = R_{1,i_0} \circ R_{i_0,i_n} \circ R_{i_n,k} = R_{1,i_0} \circ R_{i_0,i_m} \circ R_{i_n,k} = R_{1,i_m} \circ R_{i_n,k}$$

is not rectangular. Therefore, $D'_1, D'_2, \ldots, D'_{i_m}, D'_{1+i_n}, D'_{2+i_n}, \ldots, D'_k$ is an $\mathcal{L}$-M-derectangularising sequence of length less than k, which contradicts the minimality of k. □

Now that we have membership in NP, we can prove completeness.

THEOREM 10. EXISTSDERECTSEQ *is* NP*-complete under polynomial-time many-one reductions.*

Proof. We first show that EXISTSDERECTSEQ is in NP. Given D, a symmetric matrix $M \in \{0,1,*\}^{D\times D}$ and $\mathcal{L} \subseteq \mathcal{P}(D)$, a non-deterministic polynomial time algorithm for EXISTSDERECTSEQ first "guesses" an $\mathcal{S}(\mathcal{L})$-M-derectangularising sequence $D_1, \ldots, D_k$ with $k \leq 512(|D|^3 + 1)$. Lemma 28 guarantees that such a sequence exists if the output should be "yes". The algorithm then verifies that each D_i is a subset of a set in $\mathcal{L}$, that $\{D_1, \ldots, D_k\}$ is M-purifying, and that the relation $H^M_{D_1,D_2} \circ H^M_{D_2,D_3} \circ \cdots \circ H^M_{D_{k-1},D_k}$ is not rectangular. All of these can be checked in polynomial time without explicitly constructing $\mathcal{S}(\mathcal{L})$.

To show that EXISTSDERECTSEQ is NP-hard, we give a polynomial-time reduction from the well-known NP-hard problem of determining whether a graph G has an independent set of size k.

Let G and k be an input to the independent set problem. Let $V(G) = [n]$ and assume without loss of generality that $k \in [n]$. Setting $D = [n]\times[k]\times[3]$, we construct a $D\times D$ matrix M and a set $\mathcal{L}$ of lists such that there is an $\mathcal{S}(\mathcal{L})$-M-derectangularising sequence if and only if G has an independent set of size k.

M will be a block matrix, constructed using the following 3×3 symmetric matrices. Note that each is pure, apart from Id.

$$M_{\text{start}} = \begin{pmatrix} * & * & 0 \\ * & * & 0 \\ 0 & 0 & * \end{pmatrix} \qquad M_{\text{end}} = \begin{pmatrix} * & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{pmatrix} \qquad M_{\text{bij}} = \begin{pmatrix} * & 0 & 0 \\ 0 & * & 0 \\ 0 & 0 & * \end{pmatrix}$$

$$\mathbf{0} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \text{Id} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

For $v \in [n]$ and $j \in [k]$, let $D[v,j] = \{(v,j,c) \mid c \in [3]\}$. Below, when we say that $M|_{D[v,j]\times D[v',j']} = N$ for some 3×3 matrix N, we mean more specifically that $M_{(v,j,c),(v',j',c')} = N_{c,c'}$ for all $c, c' \in [3]$. M is constructed as follows.

(i) For all $v \in [n]$, $M|_{D[v,1]\times D[v,1]} = M_{\text{start}}$ and $M|_{D[v,k]\times D[v,k]} = M_{\text{end}}$.

(ii) For all $v \in [n]$ and all $j \in \{2, \ldots, k-1\}$, $M|_{D[v,j]\times D[v,j]} = M_{\text{bij}}$.

(iii) If $v \neq v'$, $(v,v') \notin E(G)$ and $j < k$, then

1. $M|_{D[v,j]\times D[v',j+1]} = M|_{D[v',j+1]\times D[v,j]} = M_{\text{bij}}$ and
2. $M|_{D[v,j]\times D[v',j']} = M|_{D[v',j']\times D[v,j]} = \mathbf{0}$ for all $j' > j+1$.

(iv) For all $v, v' \in [n]$ and $j, j' \in [k]$ not covered above, $M|_{D[v,j]\times D[v',j']} = \mathrm{Id}$.
To complete the construction, let $\mathcal{L} = \{D[v,j] \mid v \in [n], j \in [k]\}$. We will show that G has an independent set of size k if and only if there is an $\mathcal{S}(\mathcal{L})$-M-derectangularising sequence.

For the forward direction of the proof, suppose that G has an independent set $I = \{v_1, \ldots, v_k\}$ of size k. We will show that

$$D[v_1,1], D[v_1,1], D[v_2,2], D[v_3,3], \ldots, D[v_{k-1},k-1], D[v_k,k], D[v_k,k]$$

(where the first and last elements are repeated and the others are not) is $\mathcal{S}(\mathcal{L})$-M-derectangularising. Since there is no edge $(v_i, v_{i'}) \in E(G)$ for $i, i' \in [k]$, the matrix $M|_{D[v_i,i]\times D[v_{i'},i']}$ is always one of M_{start}, M_{end}, M_{bij} and $\mathbf{0}$, so it is always pure. Therefore, $\{D[v_1,1], \ldots, D[v_k,k]\}$ is M-purifying. It remains to show that the relation

$$R = H^M_{D[v_1,1],D[v_1,1]} \circ H^M_{D[v_1,1],D[v_2,2]} \circ \cdots \circ H^M_{D[v_{k-1},k-1],D[v_k,k]} \circ H^M_{D[v_k,k],D[v_k,k]}$$

is not rectangular.

Consider $i \in [k-1]$. Since $(v_i, v_{i+1}) \notin E(G)$, $M|_{D[v_i,i]\times D[v_{i+1},i+1]} = M_{\mathrm{bij}}$ so $H^M_{D[v_i,i],D[v_{i+1},i+1]}$ is the bijection that associates (v_i, i, c) with $(v_{i+1}, i+1, c)$ for each $c \in [3]$. Therefore,

$$H^M_{D[v_1,1],D[v_1,2]} \circ \cdots \circ H^M_{D[v_{k-1},k-1],D[v_k,k]}$$

is the bijection that associates $(v_1, 1, c)$ with (v_k, k, c) for each $c \in [3]$. We have $M|_{D[v_1,1]\times D[v_1,1]} = M_{\mathrm{start}}$ and $M|_{D[v_k,k]\times D[v_k,k]} = M_{\mathrm{end}}$ so

$$\begin{aligned} H^M_{D[v_1,1],D[v_1,1]} &= \{((v_1,1,c),(v_1,1,c')) \mid c,c' \in [2]\} \cup \{((v_1,1,3),(v_1,1,3))\} \\ H^M_{D[v_k,k],D[v_k,k]} &= \{((v_k,k,1),(v_k,k,1))\} \cup \{((v_k,k,c),(v_k,k,c')) \mid c,c' \in \{2,3\}\}, \end{aligned}$$

and, therefore,

$$R = \{((v_1,1,c),(v_k,k,c')) \mid c,c' \in [3]\} \setminus \{((v_1,1,3),(v_k,k,1))\},$$

which is not rectangular, as required.

For the reverse direction of the proof, suppose that there is an $\mathcal{S}(\mathcal{L})$-M-derectangularising sequence $D_1, \ldots, D_m$. The fact that the sequence is derectangularising implies that $|D_i| \geq 2$ for each $i \in [m]$ — see the remarks following Definition 8. Each set in the sequence is a subset of some $D[v,j]$ in $\mathcal{L}$ so for every $i \in [m]$ let v_i denote the vertex in $[n]$ and let j_i denote the index in $[k]$ such that $D_i \subseteq D[v_i, j_i]$. Clearly, it is possible to have $(v_i, j_i) = (v_{i'}, j_{i'})$ for distinct i and i' in $[m]$.

We will finish the proof by showing that G has a size-k independent set. Let

$$R = H^M_{D_1,D_2} \circ \cdots \circ H^M_{D_{m-1},D_m},$$

which is not rectangular because the sequence is $\mathcal{S}(\mathcal{L})$-M-derectangularising. Since $\{D_1, \ldots, D_m\}$ is M-purifying, and any submatrix of Id with at least two rows and at least two columns is impure, every pair $(i,i') \in [m]^2$ satisfies $M|_{D[v_i,j_i]\times D[v_{i'},j_{i'}]} \neq \mathrm{Id}$. This means that we cannot have $(v_i, v_{i'}) \in E(G)$ for any pair $(i,i') \in [m]^2$ so the set $I = \{v_1, \ldots, v_m\}$ is independent in G. It remains to show that $|I| \geq k$.

Observe that, if $v_i = v_{i'}$, we must have $j_i = j_{i'}$ since, otherwise, the construction ensures that

$$M|_{D[v_i,j_i]\times D[v_{i'},j_{i'}]} = M|_{D[v_i,j_i]\times D[v_i,j_{i'}]} = \mathrm{Id},$$

which we already ruled out. Therefore, $|I| \geq |\{j_1, \dots, j_m\}|$.

We must have $|j_i - j_{i+1}| \leq 1$ for each $i \in [m-1]$ as, otherwise, we would have $M|_{D[v_i,j_i]\times D[v_{i+1},j_{i+1}]} = \mathbf{0}$, which implies that $R = \emptyset$, which is rectangular. There must be at least one $i \in [m-1]$ such that $v_i = v_{i+1}$ and $j_i = j_{i+1} = 1$, so $M|_{D[v_i,j_i]\times D[v_{i+1},j_{i+1}]} = M_{\text{start}}$. If not, R is a composition of relations corresponding to M_{bij} and M_{end} and any such relation is either a bijection, or of the form of M_{end}, so it is rectangular. Similarly, there must be at least one i such that $v_i = v_{i+1}$ and $j_i = j_{i+1} = k$, giving $M|_{D[v_i,j_i]\times D[v_{i+1},j_{i+1}]} = M_{\text{end}}$. Therefore, the sequence $j_1, \dots, j_m$ contains 1 and k. Since $|j_i - j_{i+1}| \leq 1$ for all $i \in [m-1]$, it follows that $[k] \subseteq \{j_1, \dots, j_m\}$, so $|I| \geq k$, as required. In fact, $\{j_1, \dots, j_m\} = [k]$ since each $j_i \in [k]$ by construction. ∎

We defined the problem ExistsDerectSeq using a concise input representation: $\mathcal{S}(\mathcal{L})$ does not need to be written out in full. Instead, the instance is a subset $\mathcal{L}$ containing the maximal elements of $\mathcal{S}(\mathcal{L})$. For example, when the instance is $\mathcal{L} = \{D\}$, we have $\mathcal{S}(\mathcal{L}) = \mathcal{P}(D)$. It is important to note that the NP-completeness of ExistsDerectSeq is not an artifact of this concise input coding. The elements of the list $\mathcal{L}$ constructed in the NP-hardness proof have length at most three, so the list $\mathcal{S}(\mathcal{L})$ could also be constructed explicitly in polynomial time.

Lemma 28 has the following immediate corollary for the complexity of the dichotomy criterion of the general #List-M-partitions problem. Recall that, in this version of the meta-problem, the input is just the matrix M.

Corollary 11. MatrixHasDerectSeq *is in* NP.

Proof. Take $\mathcal{L} = \{D\}$ in Lemma 28. ∎

7. Cardinality constraints. Finally, we show how lists can be used to implement cardinality constraints of the kind that often appear in counting problems in combinatorics.

Feder, Hell, Klein and Motwani [15] point out that lists can be used to determine whether there are M-partitions that obey simple cardinality constraints. For example, it is natural to require some or all of the parts to be non-empty or, more generally, to contain at least some constant number of vertices. Given a $D \times D$ matrix M, we represent such cardinality constraints as a function $C\colon D \to \mathbb{Z}_{\geq 0}$. We say that an M-partition σ of a graph G *satisfies* the constraint if, for each $d \in D$, $|\{v \in V(G) \mid \sigma(v) = d\}| \geq C(d)$. Given a cardinality constraint C, we write $|C| = \sum_{d\in D} C(d)$.

We can determine whether there is an M-partition of $G = (V, E)$ that satisfies the cardinality constraint C by making at most $|V|^{|C|}$ queries to an oracle for the list M-partitions problem, as follows. Let L_C be the set of list functions $L\colon V \to \mathcal{P}(D)$ such that:

(i) for all $v \in V$, either $L(v) = D$ or $|L(v)| = 1$, and
(ii) for all $d \in D$, there are exactly $C(d)$ vertices v with $L(v) = \{d\}$.

There are at most $|V|^{|C|}$ such list functions and it is clear that G has an M-partition satisfying C if, and only if, it has a list M-partition that respects at least one $L \in L_C$. The number of queries is polynomial in $|V|$ as long as the cardinality constraint C is independent of G.

For counting, the situation is a little more complicated, as we must avoid double-counting. The solution is to count all M-partitions of the input graph and subtract off those that fail to satisfy the cardinality constraint. We formally define the problem #C-M-partitions as follows, parameterized by a $D \times D$ matrix M and a cardinality constraint function $C\colon D \to \mathbb{Z}_{\geq 0}$.

Name. #C-M-partitions.

Instance. A graph G.
Output. The number of M-partitions of G that satisfy C.

Proposition 31. *$\#C$-M-partitions is polynomial-time Turing reducible to $\#$List-M-partitions.*

Proof. Given the cardinality constraint function C, let $R = \{d \in D \mid C(d) > 0\}$: that is, R is the set of parts that have a non-trivial cardinality constraint. For any set $P \subseteq R$, say that an M-partition σ of a graph $G = (V, E)$ *fails on* P if $|\{v \in V \mid \sigma(v) = d\}| < C(d)$ for all $d \in P$. That is, if σ violates the cardinality constraints on all parts in P (and possibly others, too). Let Σ be the set of all M-partitions of our given input graph G. For $i \in R$, let $A_i = \{\sigma \in \Sigma \mid \sigma \text{ fails on } \{i\}\}$ and let $A = \bigcup_{i \in R} A_i$. By inclusion-exclusion,

$$\begin{aligned} |A| &= - \sum_{\emptyset \subset P \subseteq R} (-1)^{|P|} \Big| \bigcap_{i \in P} A_i \Big| \\ &= - \sum_{\emptyset \subset P \subseteq R} (-1)^{|P|} \big| \{\sigma \in \Sigma \mid \sigma \text{ fails on } P\} \big| . \end{aligned}$$

We wish to compute

$$\begin{aligned} \big| \{\sigma \in \Sigma \mid \sigma \text{ satisfies } C\} \big| &= \big|\Sigma\big| - |A| \\ &= \big|\Sigma\big| + \sum_{\emptyset \subset P \subseteq R} (-1)^{|P|} \big| \{\sigma \in \Sigma \mid \sigma \text{ fails on } P\} \big| . \end{aligned}$$

Therefore, it suffices to show that we can use lists to count the M-partitions that fail on each non-empty $P \subseteq R$. For such a set P, let L_P be the set of list functions L such that

(i) for all $v \in V$, either $L(v) = D \setminus P$ or $L(v) = \{p\}$ for some $p \in P$, and
(ii) for all $p \in P$, $\big| \{v \in V \mid L(v) = \{p\}\} \big| < C(p)$.

Thus, the set of M-partitions that respect some $L \in L_P$ is precisely the set of M-partitions that fail on P. Also, for distinct L and L' in L_P, the set of M-partitions that respect L is disjoint from the set of M-partitions that respect L'. So we can compute $\big| \{\sigma \in \Sigma \mid \sigma \text{ fails on } P\} \big|$ by making $|L_P|$ calls to $\#$List-M-partitions, noting that $|L_P| \le |V|^{|C|}$. □

As an example of a combinatorial structure that can be represented as an M-partition problem with cardinality constraints, consider the *homogeneous pairs* introduced by Chvátal and Sbihi [6]. A homogeneous pair in a graph $G = (V, E)$ is a partition of V into sets U, W_1 and W_2 such that:

(i) $|U| \ge 2$;
(ii) $|W_1| \ge 2$ or $|W_2| \ge 2$ (or both);
(iii) for every vertex $v \in U$, v is either adjacent to every vertex in W_1 or to none of them; and
(iv) for every vertex $v \in U$, v is either adjacent to every vertex in W_2 or to none of them.

Feder et al. [15] observe that the problem of determining whether a graph has a homogeneous pair can be represented as the problem of determining whether it has an M_{hp}-partition satisfying certain constraints, where $D = \{1, \ldots, 6\}$ and

$$M_{\mathrm{hp}} = \begin{pmatrix} * & * & 1 & 0 & 1 & 0 \\ * & * & 1 & 1 & 0 & 0 \\ 1 & 1 & * & * & * & * \\ 0 & 1 & * & * & * & * \\ 1 & 0 & * & * & * & * \\ 0 & 0 & * & * & * & * \end{pmatrix}.$$

W_1 corresponds to the set of vertices mapped to part 1 (row 1 of M_{hp}), W_2 corresponds to the set of vertices mapped to part 2 (row 2 of M_{hp}), and U corresponds to the set of vertices mapped to parts 3–6.

In fact, there is a one-to-one correspondence between the homogeneous pairs of G in which W_1 and W_2 are non-empty and the M_{hp}-partitions σ of G that satisfy the following additional constraints. For $d \in D$, let $N_\sigma(d) = |\{v \in V(G) \mid \sigma(v) = d\}|$ be the number of vertices that σ maps to part d. We require that

(i) $N_\sigma(3) + N_\sigma(4) + N_\sigma(5) + N_\sigma(6) \geq 2$,
(ii) $N_\sigma(1) > 0$ and $N_\sigma(2) > 0$, and
(iii) at least one $N_\sigma(1)$ and $N_\sigma(2)$ is at least 2.

To see this, consider a homogeneous pair (U, W_1, W_2) in which W_1 and W_2 are non-empty. Note that there is exactly one M_{hp}-partition of G in which vertices in W_1 are mapped to part 1 and vertices in W_2 are mapped to part 2 and vertices in U are mapped to parts 3–6. There is exactly one part available to each $v \in U$ since v has an edge or non-edge to W_1 (but not both!) ruling out exactly two parts and v has an edge or non-edge to W_2 ruling out an additional part. Going the other way, an M_{hp}-partition that satisfies the constraints includes a homogeneous pair.

Now let

$$M_{\mathrm{hs}} = \begin{pmatrix} * & 0 & 1 \\ 0 & * & * \\ 1 & * & * \end{pmatrix}.$$

There is a one-to-one correspondence between the homogeneous pairs of G in which W_2 is empty and the M_{hs}-partitions of G that satisfy the following additional constraints.

(i) At least two vertices are mapped to parts 2–3 (vertices in these parts are in U).

(ii) At least two vertices are mapped to part 1 (vertices in this part are in W_1).

Symmetrically, there is also a one-to-one correspondence between the homogeneous pairs of G in which W_1 is empty and the M_{hs}-partitions of G that satisfy the above constraints. (Partitions according to M_{hs} correspond to so-called "homogeneous sets" but we do not need the details of these.)

It is known from [9] that, in deterministic polynomial time, it is possible to determine whether a graph contains a homogeneous pair and, if so, to find one. We show that the homogeneous pairs in a graph can also be counted in polynomial time. We start by considering the relevant list-partition counting problems.

THEOREM 32. *There are polynomial-time algorithms for* #LIST-M_{hp}-PARTITIONS *and* #LIST-M_{hs}-PARTITIONS.

Proof. We first show that there is a polynomial-time algorithm for #LIST-M_{hp}-PARTITIONS. The most natural way to do this would be to show that there is no $\mathcal{P}(D)$-M_{hp}-derectangularising sequence and then apply Theorem 9. In theory, we could show that there is no $\mathcal{P}(D)$-M_{hp}-derectangularising sequence by brute force since $|D| = 6$, but the number of possibilities is too large to make this feasible. Instead, we argue non-constructively.

First, if there is no $\mathcal{P}(D)$-M_{hp}-derectangularising sequence, the result follows from Theorem 9.

Conversely, suppose that $D_1, \ldots, D_k$ is a $\mathcal{P}(D)$-M_{hp}-derectangularising sequence. Let M be the matrix such that $M_{i,j} = 0$ if $(M_{\mathrm{hp}})_{i,j} = 1$ and $M_{i,j} = (M_{\mathrm{hp}})_{i,j}$, otherwise. $D_1, \ldots, D_k$ is also a $\mathcal{P}(D)$-M-derectangularising sequence, since $H^{M}_{X,Y} = H^{M_{\mathrm{hp}}}_{X,Y}$ for any $X, Y \subseteq D$ and any sequence $D_1, \ldots, D_k$ is M-purifying because M is already pure. Therefore, by Theorem 9, counting list M-partitions is #P-complete.

However, counting the list M-partitions of a graph G corresponds to counting list homomorphisms from G to the 6-vertex graph H whose two components are an edge and a 4-clique, and which has loops on all six vertices. There is a very straightforward polynomial-time algorithm for this problem (a simple modification of the version without lists in [7]). Thus, #P = FP so, in particular, there is a polynomial-time algorithm for counting list M_{hp}-partitions.

The proof that there is a polynomial-time algorithm for #List-M_{hs}-partitions is similar. $\square$

Corollary 33. *There is a polynomial-time algorithm for counting the homogeneous pairs in a graph.*

Proof. We are given a graph $G = (V, E)$ and we wish to compute the number of homogeneous pairs that it contains. By the one-to-one correspondence given earlier, it suffices to show how to count M_{hp}-partitions and M_{hs}-partitions of G satisfying additional constraints. We start with the first of these. Recall the constraints on the M_{hp}-partitions σ that we wish to count:

(i) $N_\sigma(3) + N_\sigma(4) + N_\sigma(5) + N_\sigma(6) \geq 2$,
(ii) $N_\sigma(1) > 0$ and $N_\sigma(2) > 0$, and
(iii) at least one $N_\sigma(1)$ and $N_\sigma(2)$ is at least 2.

Define three subsets Σ_1, Σ_2 and $\Sigma_{1,2}$ of the set of M_{hp}-partitions of G that satisfy the constraints. In the definition of each of Σ_1, Σ_2 and $\Sigma_{1,2}$, we will require that parts 1 and 2 are non-empty and parts 3–6 contain a total of at least two vertices. In Σ_1, part 1 must contain at least two vertices; in Σ_2, part 2 must contain at least two vertices; in $\Sigma_{1,2}$, both parts 1 and 2 must contain at least two vertices. The number of suitable M_{hp}-partitions of G is $|\Sigma_1| + |\Sigma_2| - |\Sigma_{1,2}|$.

Each of $|\Sigma_1|$, $|\Sigma_2|$ and $|\Sigma_{1,2}|$ can be computed by counting the M_{hp}-partitions of G that satisfy appropriate cardinality constraints. Parts 1 and 2 are trivially dealt with. The requirement that parts 3–6 must contain at least two vertices between them is equivalent to saying that at least one of them must contain at least two vertices or at least two must contain at least one vertex. This can be expressed with a sequence of cardinality constraint functions and using inclusion–exclusion to eliminate double-counting.

Counting constrained M_{hs}-partitions of G is similar (but simpler). $\square$

REFERENCES

[1] B. Bollobás and A. Thomason, *The structure of hereditary properties and colourings of random graphs*, Combinatorica, 20 (2000), pp. 173–202.
[2] A. Brandstädt, *Partitions of graphs into one or two independent stable sets and cliques*, Discrete Math., 152 (1996), pp. 47–54.

[3] A. Bulatov, *The complexity of the counting constraint satisfaction problem*, in Proc. 35th International Colloquium on Automata, Languages and Programming (ICALP 2008), vol. 5125 of LNCS, Springer, 2008, pp. 646–661.

[4] A. Bulatov and V. Dalmau, *Towards a dichotomy theorem for the counting constraint satisfaction problem*, Inform. Comput., 205 (2007), pp. 651–678.

[5] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas, *The strong perfect graph theorem*, Ann. Math. (2), 164 (2006), pp. 51–229.

[6] V. Chvátal and N. Sbihi, *Bull-free Berge graphs are perfect*, Graph. Combinator., 3 (1987), pp. 127–139.

[7] M. Dyer and C. Greenhill, *The complexity of counting graph homomorphisms*, Random Struct. Algorithms, 17 (2000), pp. 260–289.

[8] M. Dyer and D. Richerby, *An effective dichotomy for the counting constraint satisfaction problem*, SIAM J. Comput, 42 (2013), pp. 1245–1274.

[9] H. Everett, S. Klein, and B. Reed, *An algorithm for finding homogeneous pairs*, Discrete Appl. Math., 72 (1997), pp. 209–218.

[10] ———, *An optimal algorithm for finding clique-cross partitions*, in Proc. 29th Southeastern International Conference on Combinatorics, Graph Theory and Computing, vol. 135, 1998, pp. 171–177.

[11] T. Feder and P. Hell, *List homomorphisms to reflexive graphs*, J. Combin. Theory Ser. B, 72 (1998), pp. 236–250.

[12] ———, *Full constraint satisfaction problems*, SIAM J. Comput., 36 (2006), pp. 230–246.

[13] T. Feder, P. Hell, and J. Huang, *List homomorphisms and circular arc graphs*, Combinatorica, 19 (1999), pp. 487–505.

[14] ———, *Bi-arc graphs and the complexity of list homomorphisms*, J. Graph Theory, 42 (2003), pp. 61–80.

[15] T. Feder, P. Hell, S. Klein, and R. Motwani, *List partitions*, SIAM J. Discrete Math., 16 (2003), pp. 449–478.

[16] T. Feder and M. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory*, SIAM J. Comput., 28 (1999), pp. 57–104.

[17] M. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Elsevier Science, second ed., 2004.

[18] P. Hell, M. Hermann, and M. Nevisi, *Counting partitions of graphs*, in Proc. 23rd International Symposium on Algorithms and Computation (ISAAC 2012), vol. 7676 of LNCS, Springer, 2012, pp. 227–236.

[19] C. Lecoutre, *Constraint Networks: Techniques and Algorithms*, Wiley–IEEE Press, 2009.

[20] L. Lovász, *Normal hypergraphs and the perfect graph conjecture*, Discrete Math., 2 (1972), pp. 253–267.