2014 IEEE 29th Conference on Computational Complexity

Counting list matrix partitions of graphs

Andreas Göbel
University of Oxford
andreas.goebel@cs.ox.ac.uk

Leslie Ann Goldberg
University of Oxford
leslie.goldberg@cs.ox.ac.uk

Colin McQuillan
University of Liverpool
Colin.McQuillan@liverpool.ac.uk

David Richerby
University of Oxford
david.richerby@cs.ox.ac.uk

Tomoyuki Yamakami
University of Fukui
tomoyukiyamakami@gmail.com

***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, 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 lists for each vertex of G. We give an algorithm that solves this problem in polynomial time for every (fixed) matrix M for which the problem is tractable. 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 and only if the matrix M has a structure called a derectangularising sequence. Finally, we show that the meta-problem of determining whether a given matrix has a derectangularising sequence is NP-complete.**

I. 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 any symmetric matrix $M \in \{0,1,*\}^{D\times D}$ indexed by a finite set D, an *M-partition* of an undirected graph $G=(V,E)$ is a function $\sigma\colon V \to D$ such that, for every edge $(u,v)\in E$, $M_{\sigma(u),\sigma(v)} \neq 0$. Also, for every pair (u,v) of distinct vertices that is not in E, $M_{\sigma(u),\sigma(v)} \neq 1$. 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 distinct 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 [14], many important graph-theoretic structures can be represented as M-partitions, including graph colourings, split graphs, (a,b)-graphs [2], clique-cross partitions [9], 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

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.

The full version of this paper is available as [16].

is necessary to generalise M-partitions by introducing lists. Details of these applications are given by Feder et al. [14], who define the notion of a list M-partition. In particular, constraints that certain parts must have at least some constant number of vertices can be implemented using lists.

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. [14] 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 [10], [12], [13] described in [14] presents a 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 [13] 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 [15] 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, 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 LIST-M-PARTITIONS. Feder et al. [14, Theorem 6.1] give a 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 CSP$^*_{1,2}(H)$ and showed [11, 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 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 $\#$CSP(Γ) 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 problem #M-PARTITIONS 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 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, the paper [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 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 this paper, as described below, is the development of an algorithm which solves every tractable case of the problem #LIST-M-PARTITIONS, together with a proof that all other cases are #P-complete. Thus, we obtain the following theorem.

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

Since there is no known coding of list M-partition problems as CSPs without input restrictions, Theorem 1 does not seem to be implied by the dichotomy for #CSP. To develop our algorithms and prove the theorem, 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 [11].

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 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.

A. 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 $i', j' \in D'$, the fact that (i, i'), (i, j') and (j, i') are in R implies that (j, j') is in R.

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. Pure matrices are significant because matrix partition problems associated with pure matrices correspond to graph homomorphism problems.

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.

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

Definition 8. An *$\mathcal{L}$-M-derectangularising sequence* for M of length k is a sequence $D_1, \ldots, D_k$ with each $D_i \in \mathcal{L}$ such that: $\{D_1, \ldots, D_k\}$ is M-purifying and 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, where $\circ$ denotes composition of relations.

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 III, IV and V 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 to solve it 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 V. We first give algorithms for constructing the relevant data structures: a special case of sparse-dense partitions and also subcube decompositions. Algorithm 9 uses these data structures (via Algorithms 4 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 subset-closed $\mathcal{L}$ and every M where there is no $\mathcal{L}$-M-derectangularising sequence, 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 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.

All proofs are included in the full version of this paper, which appears in preliminary form as [16]. Some algorithms are described informally in this extended abstract, but they are described more formally (as pseudocode) in the full version. To assist the reader, we have used the numbering from the full version in the extended abstract (so there are gaps in the numbering sequence here).

In Section 7 of the full paper [16], we show how to use lists to implement simple cardinality constraints. More formally, we show that, for any $D \times D$ matrix M, the following problem is polynomial-time Turing reducible to #LIST-M-PARTITIONS: for a fixed function $C\colon D \to \mathbb{Z}_{\geq 0}$, how many M-partitions of the input graph G map at least $C(d)$ vertices to part d for all $d \in D$? As a corollary, we show that there is a polynomial-time algorithm for counting the "homogeneous pairs" (defined in [6]) in any graph.

B. 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 the full version [16], 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 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. This 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 to EXISTSDERECTSEQ must be subset-closed. Let us now 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 fully 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.*

II. MORE DETAILS

The remainder of this extended abstract gives more details about our results, which are established in the full paper [16].

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

Name. $\#\mathrm{CSP}(\Gamma)$.
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.

Definition 12. For a subset-closed, M-purifying set $\mathcal{L}$, 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.

In the full version of the paper [16], we prove the following proposition.

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 $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$.*

The reduction from $\#\mathcal{L}$-M-PARTITIONS to $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$ starts with an instance (G, L) of $\#\mathcal{L}$-M-PARTITIONS and constructs an instance (V, C) of $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$ with $V = V(G)$. The constraint set C contains a unary constraint for each variable (capturing the role of the list function L). For every edge (u, v) such that $M|_{L(u)\times L(v)}$ has a 0 entry, there is a binary constraint $\langle (u, v), H^M_{L(u),L(v)}\rangle$. Similarly, there is such a constraint for every non-edge (u, v) such that $M|_{L(u)\times L(v)}$ has a 1 entry. These constraints ensure a bijection between satisfying assignments of (V, C) and M-partitions of G that respect L. The construction in the other direction is essentially the reverse of this construction, except that first we must simplify the CSP instance, ensuring that there is exactly one unary constraint for each variable, that there are no binary constraints of the form $\langle (v, v), R\rangle$ (where the variable v is repeated), and that every pair of distinct variables appears in at most one constraint. Such an instance will be called a *simple* instance below.

IV. An arc-consistency based algorithm for $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$

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 v may take any value in the domain and iteratively reduces D_v, the range of values that can be assigned to v, based on the constraints applied to it and on the values that can be taken by other variables in the scopes of those constraints. The detailed definition of the vector $(D_v)_{v\in V}$ of arc-consistent domains is given in the full version [16], along with a (standard) polynomial-time algorithm for computing it. The important point is that every satisfying assignment σ must have $\sigma(v) \in D_v$ for each variable v.

Since the $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$ instances that we deal with are simple (as defined above), the arc-consistent domains can yield further 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. A polynomial-time algorithm is given in the full version [16] which 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. The instance (V, F) is simple and it has the same satisfying assignments as (V, C).

The *constraint graph* of a CSP instance (V, C) 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. This is used in the following algorithm.

Algorithm 3. This algorithm uses arc-consistency to count 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(V,C)
    Compute the vector of arc-consistent domains
        $(D_v)_{v\in V}$
    Construct the set $F$ of factored constraints
    if $D_v = \emptyset$ for some $v \in V$ then
        return 0
    Compute the constraint graph $H$ of $(V, F)$
    Let the components of $H$ be $H_1, \dots, H_\kappa$ with
        $V_i = V(H_i)$
    for $i \in \{1, \dots, \kappa\}$ do
        Let $F_i$ be the set of constraints in $F$
            involving variables in $V_i$
        if $|D_w| = 1$ for some $w \in V_i$ then
            // $w$ is an isolated vertex in $H$. This
            // component has (exactly) 1 satisfying
            // assignment.
            $Z_i \leftarrow 1$
    else
        Choose $w_i \in V_i$
        Let $\theta_i$ be the unary constraint involving
          $w_i$ in $F_i$
        for $d \in D_{w_i}$ do
            $F'_{i,d} \leftarrow (F_i \cup \{\langle w_i, \{d\}\rangle\}) \setminus \{\theta_i\}$
            // Variable $w_i$ is 'pinned' to value $d$.
        $Z_i \leftarrow \sum_{d \in D_{w_i}} \mathrm{AC}(V_i, F'_{i,d})$
        // The number of satisfying assignments
        // factorises as $\prod_i Z_i$ because there are
        // no constraints between variables in
        // different components.
return $\prod_{i=1}^{\kappa} Z_i$
```

In the full version [16], we prove that Algorithm 3 terminates with the correct output — this follows from properties of the factoring algorithm which are hinted at in the comments of the algorithm. For general inputs, the algorithm may take exponential time to run, but we prove the following lemma which shows that the running time is polynomial for the inputs that interest us.

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.*

The key to bounding the running time is showing that the recursion depth of the algorithm is at most $|D|$. To do this, we show that the sizes of the domains of *all* variables decrease with each recursive call. The proof is based on an analysis of the constraints in a component H_i of the constraint graph, using the facts that $\mathcal{L}$ is subset-closed and M-purifying and that there is no $\mathcal{L}$-M-derectangularising sequence.

V. Polynomial-time algorithms and the dichotomy theorem

Bulatov [3] showed that every problem of the form $\#\mathrm{CSP}(\Gamma)$ 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 IV 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 Dyer and Richerby [8]. It follows immediately that $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$ 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 $\#\mathrm{CSP}(\Gamma_{\mathcal{L},M})$ 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 based on graph partitions. The first is a special case of a sparse-dense partition [14]. The second is 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.

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. *[14, 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 data structure is based on sub-hypercubes. 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 $U_1, \dots, U_k$ 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. The union $(U_1 \times U'_1) \cup \cdots \cup (U_k \times U'_k)$ is the set of assignments $\sigma\colon U \cup U' \to \{0,1\}$ such that no edge $(u, u') \in E$ has $\sigma(u) = \sigma(u') = 0$ and no pair $(u, u') \in (U \times U') \setminus E$ has $\sigma(u) = \sigma(u') = 1$. Also, for distinct $i, j \in [k]$, $U_i \times U'_i$ and $U_j \times U'_j$ are disjoint and for each $i \in [k]$, either $|U_i| = 1$ or $|U'_i| = 1$ (or both).

In the full version [16], we prove the following lemma.

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.*

Our algorithm for counting list M-partitions uses the data structures to reduce problems where $\mathcal{L}$ is not M-purifying to problems where it is (which we already know how to solve from Sections III and IV). 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 #$\mathcal{L}_i$-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 $L_1, \ldots, L_t\colon V(G) \to \mathcal{L}$ such that

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

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 $L_1, \ldots, L_k\colon V(G) \to \mathcal{L}$ such that every M-partition of G that respects L respects exactly one of $L_1, \ldots, L_k$, and for each $i \in [k]$, there is a $W \in \mathcal{L}$ which is inclusion-maximal in $\mathcal{L}$ and which 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 $L_1, \ldots, L_k$ be an enumeration of all possible functions $V(G) \to \{\{d\} \mid d \in \bigcup_{v \in V(G)} L(v)\}$. 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. If an algorithm for problem #$\mathcal{L}$-M-PURIFY-STEP is given an input with $L(v) = \emptyset$ for some vertex v, then the algorithm is entitled to output an empty list, since no M-partition respects L.

Our algorithm for the problem #$\mathcal{L}$-M-PURIFY-STEP is function #$\mathcal{L}$-M-PURIFY-STEP which is specified in Algorithm 4 at the end of this extended abstract. We show the following.

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.*

The proof of Lemma 24 considers each of the cases that arise in the execution of the algorithm — refer to Algorithm 4 at the end of this extended abstract. We give the flavour of the argument here. In Case 1, column d of $M|_{X\times Y}$ contains both a zero and a one. Equivalently, row d of $M|_{Y\times X}$ does. The algorithm groups the set of M-partitions of G that 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}.

The full version [16] contains arguments for the other two cases. To deal with these cases, we prove structural properties of the matrix M. In Case 2, we find that $M|_{X_0\times X_0}$ contains no 1s and $M_{X_1\times X_1}$ contains no 0s. Also, $M_{X_0\times X_1}$ and $M_{X_1\times X_0}$ contain only $*$s. $M|_{X\times X}$ has no $*$ on its diagonal and it has no sequence $d_1, \ldots, d_\ell \in X_0$ of odd length such that $M_{d_1,d_2} = M_{d_2,d_3} = \cdots = M_{d_{\ell-1},d_\ell} = M_{d_\ell,d_1} = *$. This 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. This ensures that the output, which is constructed using the list $(B_1, C_1), \ldots, (B_k, C_k)$ of all bipartite–cobipartite partitions of $G[V_X]$, is correct. In Case 3 (described in detail in the full version [16]) it is always true that there are distinct $X, Y \in \mathcal{L}$ such that $M|_{X\times Y}$ is not pure. The algorithm uses a subcube decomposition of a certain subgraph of G to produce the desired list functions. The proof that the algorithm is correct is given in the full version [16].
Algorithms 8 and 9. Algorithm 8 is a trivial algorithm for the problem #$\mathcal{L}$-M-PURIFY for the case in which $\mathcal{L}$ is M-purifying. It defines a function #$\mathcal{L}$-M-PURIFY which takes input (G, L) and returns L. Algorithm 9 is an algorithm for the same problem when $\mathcal{L}$ is not M-purifying (but it is still subset-closed). The algorithm defines a function as follows.

```
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
    $L_1, \dots, L_k \leftarrow$ #$\mathcal{L}$-$M$-PURIFY-STEP$(G, L)$
    for $i \in [k]$ do
        $\mathcal{L}_i \leftarrow \mathcal{P}(\bigcup_{v \in V(G)} L_i(v))$
        $L'_1, \dots, L'_j \leftarrow$ #$\mathcal{L}_i$-$M$-PURIFY$(G, L_i)$
        Add $L'_1, \dots, L'_j$ to $B$
    return $B$
```

For any fixed $\mathcal{L}$ and M the function #$\mathcal{L}$-M-PURIFY is defined either in Algorithm 8 or in Algorithm 9. The function is not recursive, but its definition is recursive in the sense that the function #$\mathcal{L}$-M-PURIFY defined in Algorithm 9 makes 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 9. The correctness of the algorithm follows from the definition of the problem. The following lemma, proved in the full version [16], 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 function #$\mathcal{L}$-M-*PURIFY *is a polynomial-time algorithm for the problem* #$\mathcal{L}$-M-PURIFY.

Finally, we present our algorithm for the problem #$\mathcal{L}$-M-PARTITIONS.

Algorithms 10 and 11. Algorithm 10 defines a function #$\mathcal{L}$-M-PARTITIONS which solves the problem #$\mathcal{L}$-M-PARTITIONS when $\mathcal{L}$ is subset-closed and M-purifying and there is no $\mathcal{L}$-M-derectangularising sequence. It uses the polynomial-time transformation from #$\mathcal{L}$-M-PARTITIONS to $\#\text{CSP}(\Gamma_{\mathcal{L},M})$ from Proposition 15 and the function AC from Algorithm 3. Algorithm 11 defines a function #$\mathcal{L}$-M-PARTITIONS for the same problem when $\mathcal{L}$ is not M-purifying but it is subset-closed and there is no $\mathcal{L}$-M-derectangularising sequence. It is defined as follows. The definition is recursive, even though the function #$\mathcal{L}$-M-PARTITIONS is not recursive.

```
function #$\mathcal{L}$-$M$-PARTITIONS($G$,$L$)
    $L_1, \dots, L_t \leftarrow$ #$\mathcal{L}$-$M$-PURIFY$(G, L)$
    $Z \leftarrow 0$
    for $i \in [t]$ do
        $\mathcal{L}_i \leftarrow \bigcup_{v \in V(G)} \mathcal{P}(L_i(v))$
        $Z \leftarrow Z +$ #$\mathcal{L}_i$-$M$-PARTITIONS$(G, L_i)$
    return $Z$
```

In the full version [16], we prove the following lemma.

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

Theorem 9 now follows from Lemma 26 (the positive case), from Proposition 15 (the connection to $\#\text{CSP}(\Gamma_{\mathcal{L},M})$) and from [4, Theorem 2 and Corollary 3] (#P-hardness for certain #CSP problems, see also [8, Lemma 24]). The details are in the full version [16].

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 #$\mathcal{L}$-$M$-PURIFY-STEP($G$,$L$)
    if there is a $v_i \in V(G)$ with $L(v_i) = \emptyset$ then
        /* No $M$-partition of $G$ respects $L$ */
        return the empty sequence
    else if there are $X, Y \in \mathcal{L}$, $a, b \in X$, and $d \in Y$ such that $M_{a,d} = 0$ and $M_{b,d} = 1$ then
        /* Case 1 */
        Choose such $X$, $Y$, $a$, $b$ and $d$ so that $X$ and $Y$ are inclusion-maximal in $\mathcal{L}$
        for $i \in [n]$ do
            $L_i(v_i) \leftarrow \{d\}$
            for $j < i$ do
                if $(v_i, v_j) \in E(G)$ then
                    $L_i(v_j) \leftarrow \{d' \in L(v_j) \mid d' \neq d$ and $M_{d,d'} \neq 0\}$
                else
                    $L_i(v_j) \leftarrow \{d' \in L(v_j) \mid d' \neq d$ and $M_{d,d'} \neq 1\}$
            for $j > i$ do
                if $(v_i, v_j) \in E(G)$ then
                    $L_i(v_j) \leftarrow \{d' \in L(v_j) \mid M_{d,d'} \neq 0\}$
                else
                    $L_i(v_j) \leftarrow \{d' \in L(v_j) \mid M_{d,d'} \neq 1\}$
            $L_{n+1}(v_i) \leftarrow L(v_i) \setminus \{d\}$
        return $L_1, \ldots, L_{n+1}$
    else if there is an $X \in \mathcal{L}$ such that $M|_{X \times X}$ is not pure then
        /* Case 2 */
        Choose such an $X$ that 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 $L_1, \ldots, L_k$
    else
        // We omit the description of Case 3 in the extended abstract. In this case, there is a pair
        // $X, Y \in \mathcal{L}$ such that $M|_{X \times Y}$ is not pure. Each $L_i$ is derived from a pair $(U_i, U_i')$ in a
        // subcube decomposition of a subgraph of $G$. The singleton projections in $U_i$ and $U_i'$
        // restrict the lists of vertices $v_j$ with $L(v_j) \in \{X, Y\}$.
```

References

[1] B. Bollobás and A. Thomason. The structure of hereditary properties and colourings of random graphs. *Combinatorica*, 20:173–202, 2000.

[2] A. Brandstädt. Partitions of graphs into one or two independent stable sets and cliques. *Discrete Math.*, 152:47–54, 1996.

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

[4] A. Bulatov and V. Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. *Inform. Comput.*, 205(5):651–678, 2007.

[5] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Ann. Math. (2)*, 164(1):51–229, 2006.

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

[7] M. Dyer and C. Greenhill. The complexity of counting graph homomorphisms. *Random Struct. Algorithms*, 17(3–4):260–289, 2000.

[8] M. Dyer and D. Richerby. An effective dichotomy for the counting constraint satisfaction problem. *SIAM J. Comput*, 42(3):1245–1274, 2013.

[9] H. Everett, S. Klein, and B. Reed. An optimal algorithm for finding clique-cross partitions. In *Proc. 29th Southeastern International Conference on Combinatorics, Graph Theory and Computing*, volume 135, pages 171–177, 1998.

[10] T. Feder and P. Hell. List homomorphisms to reflexive graphs. *J. Combin. Theory Ser. B*, 72(2):236–250, 1998.

[11] T. Feder and P. Hell. Full constraint satisfaction problems. *SIAM J. Comput.*, 36(1):230–246, 2006.

[12] T. Feder, P. Hell, and J. Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19(4):487–505, 1999.

[13] T. Feder, P. Hell, and J. Huang. Bi-arc graphs and the complexity of list homomorphisms. *J. Graph Theory*, 42(1):61–80, 2003.

[14] T. Feder, P. Hell, S. Klein, and R. Motwani. List partitions. *SIAM J. Discrete Math.*, 16(3):449–478, 2003.

[15] 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(1):57–104, 1999.

[16] Andreas Göbel, Leslie Ann Goldberg, Colin McQuillan, David Richerby, and Tomoyuki Yamakami. Counting list matrix partitions of graphs. *CoRR*, abs/1306.5176, 2013.

[17] M. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Discrete Mathematics. Elsevier Science, second edition, 2004.

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

[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(3):253–267, 1972.