List Partitions

Tomas Feder* Pavol Hell† Sulamita Klein‡
Rajeev Motwani§

Abstract

List partitions generalize list colourings and list homomorphisms. Each symmetric matrix M over $0, 1, *$ defines a list partition problem. Different choices of the matrix M lead to many well-known graph theoretic problems including the problem of recognizing split graphs and their generalizations, finding homogeneous sets, joins, clique cutsets, stable cutsets, skew cutsets and so on. We develop tools which allow us to classify the complexity of many list partition problems and, in particular, yield the complete classification for small matrices M. Along the way, we obtain a variety of specific results including: generalizations of Lovász's communication bound on the number of clique-versus-stable-set separators; polynomial-time algorithms to recognize generalized split graphs; a polynomial algorithm for the list version of the Clique Cutset Problem; and the first subexponential algorithm for the Skew Cutset Problem of Chvátal. We also show that the dichotomy (NP-complete versus polynomial-time solvable), conjectured for certain graph homomorphism problems would, if true, imply a slightly weaker dichotomy (NP-complete versus quasipolynomial) for our list partition problems[1].

*E-mail: `tomas@theory.stanford.edu`.

†School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada, V5A1S6. E-mail: `pavol@cs.sfu.ca`. Supported by a Research Grant from the National Sciences and Engineering Research Council.

‡Departamento da Ciência da Computação - I.M., COPPE/Sistemas, Universidade Federal do Rio de Janeiro, RJ, 21945-970, Brasil. E-mail: `sula@cos.ufrj.br`. Supported by CNPq and PRONEX 107/97.

§Department of Computer Science, Stanford University, CA 94305-9045. E-mail: `rajeev@cs.stanford.edu`. Supported by an ARO MURI Grant DAAH04–96–1–0007, NSF Grant IIS-9811904, and NSF Young Investigator Award CCR–9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

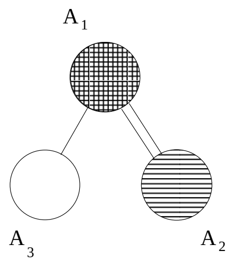
[1]A preliminary version of this paper has appeared in [22].

1 Introduction

Many combinatorial problems seek a partition of the vertices of a given graph into subsets satisfying certain constraints *internally* (a set may be required to be stable, or complete) and *externally* (two sets may be required to be completely nonadjacent – no vertex of one adjacent to any vertex of the other – or completely adjacent – each vertex of one adjacent to each vertex of the other). We may formulate a common generalization of such problems as follows: partition the vertices of an input graph into k parts $A_1, A_2, \ldots, A_k$ with a fixed 'pattern' of requirements as to which A_i's are stable or complete, and which pairs A_i, A_j are completely nonadjacent or completely adjacent. (In some cases, we also deal with a generalization where we replace 'stable' and 'complete' with more general notions of 'sparse' and 'dense'.) These requirements may be conveniently captured by a symmetric k-by-k matrix M in which the diagonal entries $M_{i,i}$ encode the internal restrictions on the sets A_i, and the offdiagonal entries $M(i,j), i \neq j$, encode the restriction on the edges between A_i and A_j.

Specifically, let M be a fixed symmetric k-by-k matrix over $0, 1, *$. An *M-partition* of a graph G is a partition of the vertex set $V(G)$ into k parts $A_1, A_2, \ldots, A_k$, such that A_i is stable (i.e., independent) if $M_{i,i} = 0$, or complete (i.e., a clique) if $M_{i,i} = 1$ (with no restriction if $M_{i,i} = *$), and such that A_i and A_j are completely nonadjacent if $M_{i,j} = 0$, or completely adjacent if $M_{i,j} = 1$ (with no restriction if $M_{i,j} = *$). When k is small, we usually refer to parts $A, B, C, \ldots$ instead of $A_1, A_2, A_3, \ldots$ and write, for example, $A = 0$ to mean $M_{A,A} = 0$ or $AB = 1$ instead of $M_{A,B} = 1$.

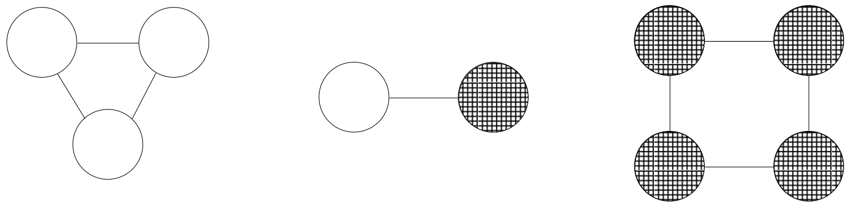
A graph G admits an M-partition if and only if its adjacency matrix $A = A(G)$ can be written, after a suitable simultaneous row and column permutation, in a block form corresponding to M - where 0 denotes an all-zero matrix, 1 an all-one matrix (with $*$'s assumed on the main diagonal), and $*$ any matrix. In Figure 1 we give an example matrix M and illustrate what an adjacency matrix A of graph G with an M-partition might look like. In the same figure, we also introduce a symbolic figure showing a general M-partition. The empty circle depicts a stable set (0 on the main diagonal of M), a shaded circle an arbitrary set (a diagonal $*$ in M), and a doubly shaded circle a clique (a diagonal 1); similarly, two parts are joined by two lines if they are completely adjacent (an off-diagonal 1), joined by a single line if there is no restriction on the edges between them (an off-diagonal $*$), and not joined at all if they are completely nonadjacent (an off-diagonal 0).



$$M = \begin{pmatrix} 1 & 1 & * \\ 1 & * & 0 \\ * & 0 & 0 \end{pmatrix} \qquad A = \left(\begin{array}{ccc|c|cc} * & 1 & 1 & 1 & * & * \\ 1 & * & 1 & 1 & * & * \\ 1 & 1 & * & 1 & * & * \\ \hline 1 & 1 & 1 & * & 0 & 0 \\ \hline * & * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 \end{array}\right)$$

Figure 1. A partition, its matrix M, and example block structure of an adjacency matrix A corresponding to a graph with an M-partition (in the matrix A, each $*$ represents either 0 or 1).

Many graph theoretic concepts can be modeled by M-partitions. Indeed, in Figure 2, we illustrate three such concepts – from the well-known notions of a graph colouring and a split graph [28], to the more recent notion of a clique-cross partition [17].



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

A 3-colouring   A split graph   A clique - cross partition

Figure 2. Three typical partition problems.

All three concepts have natural generalizations which may also be modeled as M-partitions:

A k-colouring of a graph G is an M-partition of G where the matrix M has zeros on the main diagonal and asterisks everywhere else. In other words, M is obtained from the adjacency matrix of the complete k-graph by replacing all ones with asterisks. The natural generalization, an M-partition of G, where M is obtained the same way from the adjacency matrix of an **arbitrary** graph H, is called an *H-colouring* or a *homomorphism* [29, 30]). Thus an H-colouring of G, or a homomorphism of G to H, is a partition of

$V(G)$ into sets $A_h, h \in V(H)$, such that A_h is stable when h is not a loop of H, and $A_h, A_{h'}$ are completely nonadjacent when hh' is not an edge of H. The k-colouring problem is well known to be polynomial-time solvable when $k \leq 2$ and NP-complete otherwise [27]. The H-colouring problem is polynomial-time solvable when H is bipartite or when H contains a loop, and is NP-complete otherwise [30].

A *split graph* is a graph which admits a partition into a stable set and a clique [28], i.e., an M-partition where M is the matrix given in Figure 2, with asterisks off the main diagonal, and exactly one 0 and one 1 on the main diagonal. An (a,b)-*graph* [4] is a natural generalization – a graph whose vertices can be partitioned into a stable sets and b cliques (when $a = b$ they are also called a-*split graphs*); the corresponding M is an $(a+b)$-by-$(a+b)$ matrix having all off-diagonal entries equal to $*$ and with a zeros and b ones on the main diagonal. When $a, b \leq 2$ (this includes split graphs, which have $a = b = 1$), the (a,b)-graphs satisfy the Strong Perfect Graph Conjecture of Berge [31], and can be recognized in polynomial time. (Brandstadt claimed such algorithms in [4], which were in error [5]; more involved polynomial-time algorithms were given in [6], and a new algorithm of complexity $O((n+m)^2)$ in [7]; polynomial time algorithms also follow from our more general results in Section 3.) On the other hand, it is easy to see that when a or b is at least 3 it is NP-complete to recognize (a,b)-graphs [4, 7]. The split graphs $(a = b = 1)$ are a well-known class of perfect graphs, and they admit efficient algorithms for many standard combinatorial optimization problems [28].

A *clique-cross partition* [17] of a graph G is a partition of the vertices of G into four disjoint cliques A, B, C, D such that A, C as well as B, D are completely nonadjacent. This is an M-partition, where M is given in Figure 2; note that M is obtained from the adjacency matrix of the four-cycle by replacing all ones with asterisks and setting all diagonal entries to 1. The more general concept [33] of an H-*clique partition* is the M-partition problem where M is the matrix obtained in the same way from the adjacency matrix of an **arbitrary** graph H. A clique-cross partition can be found in linear time [17]. The more general H-clique partition problem is polynomial-time solvable when H is a triangle-free graph; otherwise it is NP-complete [33].

Several other well-known graph concepts correspond to M-partitions with additional properties. Figure 3 illustrates some of these concepts:

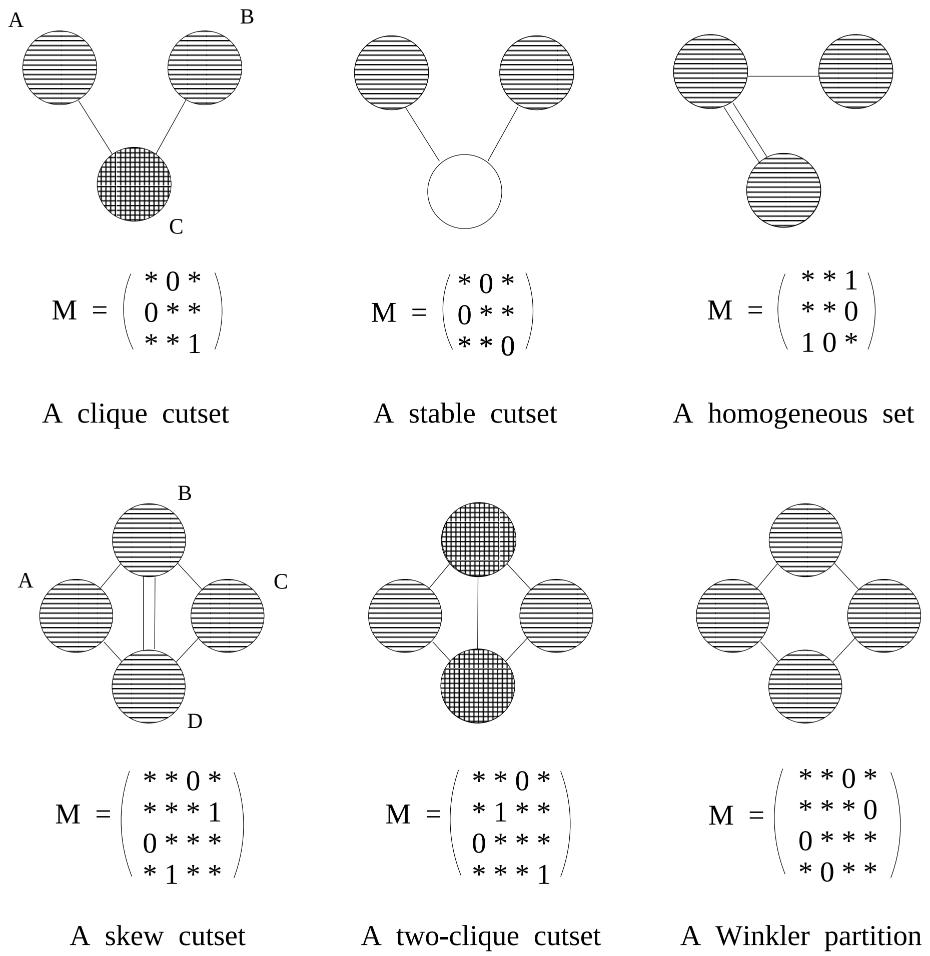


Figure 3. Other well-known partition problems.

A *clique cutset* [35, 39] of a connected graph G is a complete subgraph C whose removal disconnects G. Clearly G has a clique cutset if and only if it admits a partition of the vertices into three non-empty subsets A, B, C such that C is a clique and A, B are completely nonadjacent (so that the removal of C disconnects A from B), i.e., if and only if it admits an M-partition, where M is the matrix given in Figure 3, such that **all parts are non-empty**. Finding clique cutsets, possible in polynomial time [35, 39, 40], is the basis of a decomposition algorithm [35], which allows efficient solution of many optimization problems for the class of decomposable graphs [35]. A *stable cutset* [36] is defined analogously (B is stable), and also corresponds to an M-partition with all parts non-empty. Stable cutsets are of interest because a result of Tucker [36] asserts that a minimal imperfect graph other than an odd cycle cannot contain a stable cutset; this problem has been proved NP-complete in [24].) The *two-clique cutset problem* is defined similarly as a union of two complete subgraphs that disconnects the input graph. Its matrix is given in Figure 3, and we discuss it further in Section 5, where we give a subexponential algorithm for the problem. Whether or not it admits a polynomial-time algorithm remains an interesting open problem.

A *skew cutset* of a connected graph G is a pair of disjoint non-empty sets B, D in G, such that the removal of $B \cup D$ disconnects the graph, and such that B, D are completely adjacent (the 'skew property'). Once again, this is clearly a partition problem – we wish to partition the vertices of G into four non-empty sets A, B, C, D such that A, C are completely nonadjacent and B, D completely adjacent. This is an M-partition, where M is given in Figure 3, with all parts non-empty. Chvátal conjectured [9] that a minimal imperfect graph cannot contain a skew cutset. He proved this for the special skew cutsets where B consists of a single vertex; for this case he also gave a polynomial time recognition algorithm. (The conjecture has also been established [12] if it is required that B and D are both stable ($B = D = 0$); in this case the recognition problem is NP-complete [24].) Chvátal posed the problem of the complexity of finding a general skew cutset. In [22], we offered the first subexponential time algorithm, strongly suggesting that the problem is not NP-complete. Most recently, one of us (Klein), together with de Figueiredo, Kohayakawa, and Reed, indeed found a polynomial algorithm [25].

Winkler formulated a similar problem, seeking a partition into non-empty sets A, B, C, D where there are no edges between A and C nor between between B and D, but at least one edge between A and B, between B and C, between C and D, and between D and A. Winkler asked for the complexity of this problem; it has been shown NP-complete in [37]. This is an M-partition problem (M is given in Figure 3), where there are not only restrictions on the nonemptiness of the parts, but also on the presence of edges in their connections.

A *homogeneous set* [13] in a graph G is a set C of vertices of G such that each vertex outside of C is adjacent either to all or to none of the vertices in C. It is again easy to see that this is a partition problem – we want to partition the vertices into three subsets A, B, C such that A, C are completely adjacent and B, C completely nonadjacent. To avoid the trivial homogeneous sets consisting of a single vertex or the entire vertex set, we also require that C has at least two vertices and $A \cup B$ is non-empty. Therefore, this is an M-partition (with M given in Figure 3) where there are more complex restrictions on the sizes of the parts. Homogeneous sets also define a decomposition (the 'modular decomposition') which facilitates the recognition of comparability graphs (and other similar classes of graphs) [13, 34]. Homogeneous sets (and modular decompositions) can be found efficiently [34].

Size restrictions are also needed to model the concepts of a *homogeneous pair* [9], *join- and 2-join-decomposition* [16, 11], or the more general *amalgam and 2-amalgam decompositions* [8, 11]. (Below we give the matrices of the homogeneous pair and 2-amalgam problems; the matrices for the join, 2-join, and amalgam problems are all obtained from the matrix for 2-amalgam by deleting some rows and columns.) Chvátal and Sbihi [10] showed that no minimal imperfect graph contains a homogeneous pair and used this result to prove the perfectness of an important new class of graphs (the 'bull-free Berge graphs'). A polynomial time algorithm for the recognition of homogeneous pairs has been given by Everett, Klein, and Reed [18]. These decompositions preserve perfection, and are a tool for the recognition of several classes of perfect graphs [3, 8, 35, 40]; they can also be found in polynomial time [14, 15, 8, 11].

$$\underset{\text{(Homogeneous pair)}}{\begin{pmatrix} * & * & 1 & 0 & 1 & 0 \\ * & * & 1 & 0 & 0 & 1 \\ 1 & 1 & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 1 & 0 & * & * & * & * \\ 0 & 1 & * & * & * & * \end{pmatrix}} \quad \underset{(2-\text{amalgam})}{\begin{pmatrix} * & 1 & * & 0 & * & 0 & 1 \\ 1 & * & 0 & * & 0 & * & 1 \\ * & 0 & * & 1 & * & 0 & 1 \\ 0 & * & 1 & * & 0 & * & 1 \\ * & 0 & * & 0 & * & 0 & * \\ 0 & * & 0 & * & 0 & * & * \\ 1 & 1 & 1 & 1 & * & * & 1 \end{pmatrix}}$$

To capture these additional requirements (that certain parts be non-empty, or have at least a fixed number of vertices, or have at least some edges joining them, etc.), we shall introduce the concept of lists. In the list version of a partition problem, each vertex of the input graph has a list of the parts in which it is allowed to be placed. This gives us a wide variety of options in restricting the contents of the individual parts or of their connections. For instance, in the case of homogeneous sets, we may ensure that C has at least two vertices and $A \cup B$ is non-empty by choosing three vertices x, y, z of the input graph and specifying that the lists of x, y

only consist of C and the list of z consists of A, B. Thus the problem of finding a homogeneous set in a graph with n vertices is reduced to n^3 list partition problems. (A homogeneous set exists if and only if at least one of the n^3 choices of x, y, z has a desired list partition). Analogously, one can ensure that there is at least one edge between parts X and Y by restricting (with the choice of lists) two adjacent vertices x, y to be placed into X, Y respectively, for all possible choices of an edge xy in the input graph.

Concretely, let M be a fixed k-by-k matrix. Given a graph G, and for each vertex $v \in V(G)$ a set ('list') $L(v) \subseteq \{1, 2, \ldots, k\}$, we define a *list M-partition* of G, with respect to the lists L, to be an M-partition $A_1, A_2, \ldots, A_k$ of G in which each $v \in V(G)$ belongs to a part A_i with $i \in L(v)$. The *list M-partition problem* asks whether or not an input graph G with lists L admits a list M-partition.

Both the basic M-partition problem ('Does the input graph admit an M-partition?'), and the problem of the existence of an M-partition with all parts non-empty, admit polynomial-time reductions to the list M-partition problem, as do all of the above problems with more complex constraints.

List partitions generalize list colourings, which have proved very fruitful in the study of graph colourings [1, 26]. They also generalize list homomorphisms (or list H-colourings) which have brought a degree of order to the study of the complexity of graph homomorphisms, cf. below. One reason why lists are useful is that they allow us to solve problems by recursing to subproblems with modified lists. (This was also exploited in the algorithms in [25].)

List homomorphisms (or list H-colourings) [19, 20, 21] are close in spirit to list partitions. A list H-colouring of a graph G is a list M-partition of G, where M is obtained from the adjacency matrix of G by replacing all 1's with $*$'s.

In [19, 20, 21] we attempted to classify the complexity of list H-colourings, i.e., the complexity of list M-partition when M is a $(0, *)$-matrix.

When all the diagonal entries of M are the same (all 0 or all $*$), a complete classification is obtained: when all the diagonal entries are $*$ (H has all loops), the problem is polynomial-time solvable if H is an interval graph and is NP-complete otherwise [19]; when all diagonal entries of M are 0 (H has no loops), the problem is polynomial-time solvable if H is bipartite and its complement $\overline{H}$ is a circular arc graph, and is NP-complete otherwise [20].

For general $(0, *)$-matrices M we have conjectured a classification in [21]. It again relates to a kind of geometric representation of H. The important point is that, if true, this classification would imply that all list M-partition problems for $(0,*)$-matrices M (list H-colouring problems) are polynomial-time solvable or NP-complete. This kind of 'dichotomy' is rare in general, and was conjectured, in the more general context of constraint-satisfaction problems, in [23].

Similar comments apply to M-partitions where M is a $(*,1)$-matrix (cf. Proposition 2.8). This problem corresponds to a homomorphism problem among the complementary graphs (still a constraint satisfaction problem). The appealing feature of the general M-partition problem is that it allows these homomorphism-type (constraint-satisfaction-type) constraints on both edges and nonedges of the graph. In particular, general list M-partition problems are **not** constraint satisfaction problems.

As the above examples illustrate, we are often interested in the complexity of finding the desired partitions. This is the recurring theme of all of the above discussion. In this paper, we shall focus on this aspect, although, of course, list partitions offer other interesting questions.

The organization of the paper is as follows:

In Section 2, we describe some basic techniques.

In Section 3, we introduce sparse-dense partitions. Graphs which admit sparse-dense partitions can be recognized efficiently if sparse and dense graphs can. Many partition problems can be modeled as sparse-dense partitions, including many of our M-partitions, and we obtain polynomial-time algorithms for several such problems.

In Section 4, we investigate separator theorems. Motivated by a result of Lovász, we derive several extensions which will be used later. This technique leads to subexponential, but not necessarily polynomial, algorithms for certain M-partitions.

In Section, 5 we illustrate the use of our tools on some prominent example list M-partition problems: the $(2,1)$- and $(2,2)$-graphs of Brandstadt, the Clique Cutset Problem, the Skew Cutset Problem of Chvátal, and the Two-Clique Cutset problem.

In Section 6, we apply the techniques to classify the complexity of list M-partition problems when the matrix M is small. All these problems are polynomial-time solvable when M is a 2-by-2 matrix. For 3-by-3 matrices we classify the problems as polynomial-time solvable or NP-complete. When M is a 4-by-4 matrix, we are able to classify all these problems as being either NP-complete or 'quasipolynomial'.

In Section 7, we prove that if it is true (as conjectured in [23], cf. also [21]) that all list homomorphism problems are polynomial or NP-complete, then it also follows that all list M-partition problems are 'quasipolynomial' or NP-complete.

We use the term *quasipolynomial* for a function that is bounded by $n^{c\log^t n} = 2^{c\log^{t+1} n}$ for some positive constants c, t. While we, of course, prefer to find polynomial (time) algorithms, we take the existence of a quasipolynomial algorithm as evidence that the problem is not likely to be NP-complete. Indeed, no NP-complete problem is known to be solved by a quasipolynomial algorithm, and, since all NP-complete problems are polynomially equivalent, a quasipolynomial algorithm for any one NP-complete problem would imply the existence of such algorithms for **all** NP-complete problems.

2 Basic Tools

We begin by assembling some basic techniques. For some matrices M, these are sufficient to solve the list M-partition problem in polynomial time. We shall also use them in conjuction with other tools to be described in later sections.

The most basic technique is the 2-satisfiability algorithm of [2]. Suppose first that M is a 2-by-2 matrix seeking to partition the input graph into two parts, say A, B. We can solve the list M-partition problem by introducing a boolean variable x_v for each vertex v of the input graph G; we think of the value of x_v as encoding whether or not the vertex v belongs to the part A of the partition ($x_v = 1$ means $v \in A$, $x_v = 0$ means $v \notin A$). It is then easy to see that all the constraints, and lists, of the list M-partition problem can be stated by polynomially many clauses with at most two literals each. For instance, if A is to be a stable set ($A = 0$), we impose the constraint $\overline{x_u} \wedge \overline{x_v}$ for each edge uv of G. Similarly, if, say, A, B are to be completely adjacent ($AB = 1$), we impose the constraint $x_u \wedge \overline{x_v}$ for each nonedge uv of G. Finally, if the list of v is, say, B, we impose the constraint $\overline{x_v}$. Hence the problem can now be solved by the 2-satisfiability algorithm [2].

The same technique applies any time we have an instance in which every list has size at most two:

Proposition 2.1 *There is a polynomial-time algorithm which solves any list M-partition problem restricted to instances in which the list of every vertex of the input graph has size at most two.* □

One other basic technique occurs in many places – the placing of a vertex.

This is the flexibility that lists offer: suppose the input consists of the graph G with lists L, and let v be a vertex of G. We may decide at some point to place a vertex v into a part X (either because the list of v has only X in it, or because we will consider the other options later). This can be accomplished by removing v from the graph and updating the lists of all the other vertices to take it into account: specifically, for all Y with $XY = 0$, we remove Y from the lists of all neighbours of v (they can no longer be placed in Y), and for all Z with $XZ = 1$, we remove Z from the lists of all non-neighbours of v (for a similar reason). Call the resulting graph G' (=$G \setminus v$) and the resulting lists L'.

Proposition 2.2 *The graph G with lists L admits a solution (to the list M-partition problem) with $v \in X$ if and only if the graph G' with lists L' admits a solution.* $\square$

Suppose a row X of M contains both a 0 and a 1, say $XY = 0$ and $XZ = 1$ (either of Y and Z could be X, i.e., we could have $X = 0, XZ = 1$ or $XY = 0, Z = 1$). In this case we can reduce the list M-partition problem for an n-vertex input graph G (with respect to lists L) to the following (at most $n + 1$) subproblems:

First check whether or not the input graph G has a partition in which no vertex lies in the part X, and then check for each vertex v of G which has X in its list whether or not G has a partition with v in X. The former can clearly be accomplished by removing X from all lists (call the resulting lists L'), the latter can be tested by placing v in X and updating the lists of all other vertices of G as explained above. (Denote the resulting lists for the graph $G \setminus v$ by L^v.) Note that since $XY = 0, XZ = 1$, these updates will result in no list containing both Y and Z.

Proposition 2.3 *Suppose the matrix M has $XY = 0, XZ = 1$. Then the input graph G admits a list M-partition with respect to lists L if and only if G admits a list M-partition with respect to the lists L' or if $G \setminus v$ admits a list M-partiton with respect to the lists L^v, for some vertex v of G.* $\square$

Corollary 2.4 *Suppose the matrix M has $XY = 0, XZ = 1$. Then the list M-partition problem can be reduced to one instance with no list containing X, and at most n instances with no list containing both Y and Z.* $\square$

This is particularly useful when $k = 3$, as in this case all lists become size at most two.

We say that X *dominates* Y in the matrix M, if for each Z (possibly equal to X and Y) we have $XZ = YZ$ or $XZ = *$. If X dominates Y, we can eliminate Y from any list containing X, since any vertex that goes to part Y can be placed to X instead. Thus for an input graph G with lists L we may define the modified lists L' obtained from L by removing Y from any list that contains X. (Note that this also may, in some cases, result in all lists having size at most two.)

Proposition 2.5 *If X dominates Y in the matrix M, then an input graph G admits a list M-partition with respect to lists L if and only if it admits a list M-partition with respect to lists L'.* □

Thus if X dominates Y, we may assume that no list contains both X and Y. In particular, when X dominates all other vertices, we may assume that each list is either just $\{X\}$ or does not contain X. This allows us to drop X by placing all vertices with lists $\{X\}$ as explained above, and reducing the matrix by eliminating the row and column corresponding to X.

We say that M is *disconnected* if it can be written as a 2-by-2 block matrix with diagonal blocks M_1, M_2 and off-diagonal blocks having all entries 0.

Proposition 2.6 *If M is disconnected, with diagonal blocks M_1, M_2, and if both the list M_i-partition problems ($i = 1, 2$) can be solved in polynomial time, then also the list M-partition problem is polynomial-time solvable.*

Proof. This is so, since each component of the input graph G must be placed entirely to parts from M_1 or parts from M_2. □

We say that a k-by-k matrix M *contains* a k'-by-k' matrix M', $k' \leq k$, if M' is a principal submatrix of M. In other words, the parts of the M'-partition problem are a subset of the parts of the M-partition problem, with the same constraints on the parts and their connections.

Proposition 2.7 *If M contains M', and the list M'-partition problem is NP-complete, then so is the list M-partition problem.*

Proof. We reduce the list M'-partition problem to the list M-partition problem as follows: Let G with lists $L(v) \subseteq \{1, 2, \ldots, k'\}$ be any instance of the list M'-partition problem. We may view the same graph G, with the same lists L, as an instance of the list M-partition problem as well, since each $L(v) \subseteq \{1, 2, \ldots, k'\} \subseteq \{1, 2, \ldots, k\}$. Clearly, G with lists L admits a list M'-partition if and only if it admits a list M-partition. $\square$

The *complement* $\overline{M}$ of a matrix M is obtained from M by replacing each 0 by 1 and each 1 by 0 (asterisks remain unchanged).

Proposition 2.8 *A graph G admits a list M-partition, with respect to the lists L, if and only if its complement $\overline{G}$ admits a list $\overline{M}$-partition, with respect to the same lists L.* $\square$

3 Sparse-Dense Partitions

We now introduce a class of problems which will be useful for several M-partition problems, and which are interesting in their own right.

Let $\mathcal{S}$ and $\mathcal{D}$ be two classes of graphs, called *sparse* and *dense* respectively, satisfying the following constraints:

- Both $\mathcal{S}$ and $\mathcal{D}$ are closed under taking induced subgraphs.
- There exists a constant c such that the intersection $S \cap D$ has at most c vertices, for any $S \in \mathcal{S}$ and $D \in \mathcal{D}$.

In a given graph G, we say that a set of vertices is sparse (dense) if the subgraph of G they induce is sparse (respectively dense).

A *sparse-dense partition* of a graph G, with respect to the classes $\mathcal{S}$ and $\mathcal{D}$, is a partition of $V(G)$ into two parts $V(G) = S \cup D$, such that $S \in \mathcal{S}$ (S is sparse) and $D \in \mathcal{D}$ (D is dense).

Sparse-dense partitions are inspired by split graphs. Indeed, we may take $\mathcal{S}$ to consist of all edgeless graphs (stable sets) and $\mathcal{D}$ to consist of all complete graphs (cliques). It is clear that both $\mathcal{D}$ and $\mathcal{S}$ are closed under taking induced subgraphs, and as an $S \in \mathcal{S}$ and a $D \in \mathcal{D}$ have at most one vertex in common, we can take $c = 1$. A graph has a sparse-dense partition with respect to this choice of $\mathcal{S}, \mathcal{D}$ if and only if it can be partitioned into a stable set and a clique, i.e., if and only if it is a split graph.

There are a number of other situations conveniently modeled by sparse-dense partitions. Several are described at the end of this section. Let us just mention the following typical examples:

(a, b)**-graphs:** Let $\mathcal{S}$ consist of all a-colourable graphs, and $\mathcal{D}$ of all graphs whose complements are b-colourable; we can take $c = ab$.

Partitions into a graph with clique-size at most a and a graph with stable-set-size at most b: Let $\mathcal{S}$ consist of all graphs without cliques of size $a+1$, and $\mathcal{D}$ of all graphs without stable sets of size $b+1$. The constant c can be taken to be the Ramsey number $R(a+1, b+1)$, as is explained in Proposition 3.3 below.

Partitions into a planar graph and a clique: Just to illustrate the range of possibilities, we may define $\mathcal{S}$ to consist of all planar graphs, and $\mathcal{D}$ of all complete graphs. The four-colour theorem implies that we can take $c = 4$.

(In [7] the authors consider a number of partition problems with similar flavour: e.g., into a stable set and a tree (NP-complete), a stable set and a trivially perfect graph (NP-complete), or a stable set and a threshold graph (polynomial-time solvable). The latter satisfies the conditions for a sparse-dense partition and can in fact be solved by our technique, cf. [7].)

In most of our examples, the classes $\mathcal{S}, \mathcal{D}$ are recognizable in polynomial time. (Of the above examples, only the (a, b)-graphs with $a \geq 3$ or $b \geq 3$ are an exception.) It turns out that in such a case the existence of a sparse-dense partition can be decided in polynomial time. In fact, in such a case *all* sparse-dense partitions can be found in polynomial time:

Theorem 3.1 *Consider any classes of sparse and dense graphs satisfying the above conditions.*

A graph on n vertices has at most n^{2c} different sparse-dense partitions.

Furthermore, all these partitions can be found in time proportional to $n^{2c+2}T(n)$, where $T(n)$ is the time for recognizing sparse and dense graphs.

PROOF. Let $V(G) = S \cup D$ be a particular sparse-dense partition. Then any other sparse-dense partition $V(G) = S' \cup D'$ has $|S' \cap D| \leq c$ and $|S \cap D'| \leq c$, so S' is obtained from S by deleting at most c vertices and inserting at most c new vertices. In fact, if we allow ourselves to insert back a vertex that has just been deleted, we can say that we make exactly c deletions and exactly c insertions. Each of these at most $2c$ operations can be made in at most n ways. This observation proves the first assertion and allows us to find *all* sparse-dense partitions if one such partition is known. It amounts to a $2c$-local search (the current S is changed in at most $2c$ vertices), and can be performed in time n^{2c}.

It remains to explain how to find the first sparse-dense partition. The algorithm proceeds in two phases. The first phase attempts to find as large a sparse set as possible. This is based on the observation that if $V(G) = S \cup D$

is a sparse-dense partition and S' a sparse set smaller than S, then $S' \cap D$ has at most c vertices, and hence, as above, S' can be enlarged by removing some c vertices and inserting some $c+1$ new vertices (recall that subsets of sparse sets are sparse). Thus, starting with any sparse set (for instance the empty set), we can increase its size by performing a $(2c+1)$-local search (making all possible c deletions and $c+1$ insertions and testing if the result is sparse) in time $n^{2c+1}T(n)$. After performing this operation at most n times, we reach a situation where the current sparse set can no longer be enlarged in this way. Clearly, at this point our current sparse set S' has the same size as the (unknown) set S.

The second phase of the algorithm attempts to change S', without changing its size, until $V(G)-S$ is dense. This is accomplished by a $2c$-local search, based on a very similar principle – namely, if $V(G) = S \cup D$ is a sparse-dense partition and $|S| = |S'|$, then S is obtained from S' by a deletion of c vertices and the insertion of c other vertices. Thus we can test all n^{2c} possible new sets S' for sparseness and the corresponding $V(G) - S'$ for denseness, and if no sparse-dense partition is found we can be sure none exists.

The most time-consuming operation is the first phase of the algorithm, finding one sparse-dense partition – taking time $n^{2c+2}T(n)$. $\square$

In cases where computing $T(n)$ is hard (such as (a,b)-graphs with a or b at least 3), it also turns out to be hard to decide if a sparse-dense partition exists:

Proposition 3.2 *Suppose that the disjoint union of sparse graphs is also sparse. If testing for sparse graphs is NP-complete, then the partition problem into sparse and dense graphs is also NP-complete.*

Proof. Suppose we wish to test whether G is sparse. We can construct G' by taking the disjoint union of $c+1$ copies of G. Then G' has a sparse-dense partition if and only if G is sparse. Indeed, if G is sparse, then G' is sparse by the assumption; on the other hand, if G' admits a partition, then our algorithm will find a pair (S, D) where D has at most c vertices. Therefore, one of the copies of G is contained in S, so G is sparse. $\square$

We have defined sparse and dense subgraphs with respect to each other, since the definition depends on the existence of a constant c bounding their intersections. The next result shows that we can define sparse and dense graphs independently, under the assumption that stable sets are always sparse and cliques are always dense.

Proposition 3.3 *Suppose $\mathcal{S}$ is a class of graphs sparse with respect to the cliques (with a constant a), and $\mathcal{D}$ a class of graphs dense with respect to the stable sets (with a constant b). Then $\mathcal{S}, \mathcal{D}$ are sparse and dense with respect to each other, with some associated constant c.*

PROOF. The intersection of $S \in \mathcal{S}$ and $D \in \mathcal{D}$ is sparse and hence cannot contain a $(a+1)$-clique, and is also dense and so cannot contain a $(b+1)$-stable set. Such a graph has its number of vertices bounded by the Ramsey number $c = R(a+1, b+1)$ (cf. [38]). □

This result makes it easy to find additional examples of sparse and dense classes. Examples of sparse classes (with respect to cliques) are stable sets, bipartite graphs, $(c+1)$-clique-free graphs, planar graphs, and c-colourable graphs. (The last one is NP-complete for $c \geq 3$, the remaining are polynomial-time solvable.) Examples of dense classes (with respect to stable sets) can be obtained by taking complements, e.g., cliques, cobipartite graphs, graphs without $(c+1)$-stable sets, complements of planar graphs, and complements of c-colourable graphs. Combining any one of the former with any one of the latter produces a sparse-dense pair of families. In particular, this shows that the earlier example where dense sets have clique-size at most a and sparse sets have stable-set-size at most b satisfies the requirements.

4 Separators

Some partition problems on G can be solved by considering all maximal cliques of G: for example, to decide if a graph is a split graph we can test the complements of all maximal cliques (and all maximal cliques with one vertex deleted) to see if any are stable. Indeed, if C is a clique and S a stable set, some maximal clique (or maximal clique with one vertex deleted) of G always 'separates' C from S in the sense that it contains C and is disjoint from S. Unfortunately, in general the number of maximal cliques is exponential. (The graph $K_{2n} \setminus nK_2$ has 2^n maximal cliques.) The following result of Lovász asserts that there always exists a *subexponential* family of sets that separate cliques and stable sets. Such separators turn out to be surprisingly useful for list M-partitions.

Let G be a graph. A family $\mathcal{E}$ of subsets of $V(G)$ is said to *separate* cliques and stable sets if for any pair of disjoint sets C, S, such that C is a clique and S is a stable set in G, some $E \in \mathcal{E}$ contains C and is disjoint from S.

Lovász's subexponential bound turns out to be quasipolynomial, as do our generalizations of it. It is not known whether or not the bound can be improved to a polynomial.

Theorem 4.1 *[32] Every graph with n vertices has a family of $n^{\frac{1}{2}\log n}$ sets that separate cliques and stable sets.*

Moreover, such a family can be found in time $n^{\frac{1}{2}\log n}$ times a polynomial in n.

PROOF. The bound actually given in [32] is $2^{\binom{1+\log(n+1)}{2}} \leq n^{\frac{1}{2}\log n}$. It is couched in terms of communication complexity (as a communication game), cf. also [22]. Here we describe a more combinatorial view of the proof. For simplicity we will only prove a weaker bound of $n^{\log n}$ sets (found in time $n^{\log n}$ times a polynomial in n). The bound as given in [32], and as claimed in the Theorem, is a direct corollary of our Theorem 4.7, obtained by setting $t = 2$. We choose to give the proof (of the weaker bound) in detail because it will allow us to explain how the proof needs to be modified to obtain our generalizations.

The idea of the proof is to obtain a family of sets E which are sufficient to separate cliques and stable sets, and which can be described 'concisely' – and hence are not too numerous.

Suppose C is a clique and S a disjoint stable set, in G. A *valid encoding* of the pair C, S will be a sequence $v_1, v_2, \ldots, v_k$ of vertices of G obtained as follows:

Let $G_1 = G$. At any stage, n_i will denote the number of vertices of the graph G_i.

Suppose G_{i-1}, v_{i-1} have already been defined. Then we define v_i and G_i by either of the following two pairs of rules:

- v_i is a vertex of S whose degree in G_{i-1} is greater than $n_{i-1}/2$,
- and G_i is the graph obtained from G_{i-1} by deleting v_i and all its neighbours,

or

- v_i is a vertex of C whose degree in G_{i-1} is smaller than or equal to $n_{i-1}/2$,

- and G_i is the graph obtained from G_{i-1} by deleting v_i and all vertices which are **not** neighbours of v_i.

Since at each step we remove more than half of the vertices, G_i becomes empty for $i > \log n$, and we may assume that $k \leq \log n$. (All logarithms in this paper are base two.)

We now claim that a valid encoding of a pair C, S determines a set E which contains C and is disjoint from S. Equivalently, we will find two complementary sets $E = C^+$ and $\overline{E} = S^+$ such that C^+ contains C and S^+ contains S. To obtain C^+, S^+ we *decode* the sequence $v_1, v_2, \ldots, v_k$ as follows:

Let $C_1^+ = \emptyset, S_1^+ = \emptyset$. At any stage, C_i^+, S_i^+ will be disjoint, G_i will be the graph obtained from G by deleting C_i^+ and S_i^+, and n_i will denote the number of vertices of G_i.

If C_{i-1}^+, S_{i-1}^+ have already been defined, we consider the degree d of v_i in G_i.

- If $d > n_i/2$, then we add v_i to S_{i-1}^+ and all its neighbours to C_{i-1}^+, thus forming new C_i^+, S_i^+,
- otherwise ($d \leq n_i/2$), we add v_i to C_{i-1}^+ and all the vertices that are not its neighbours to S_{i-1}^+, creating in this way new C_i^+, S_i^+.

Once all v_i have been processed, we form C^+ by adding to C_k^+ all the remaining vertices of G_k of high degree in G_k, that is degree in G_k greater than $n_k/2$, and form S^+ by adding to S_k^+ all the other vertices (of degree at most $n_k/2$ in G_k). Note that S^+ is the complement of C^+.

Since the decoding process reverses the steps of the encoding, the resulting set C^+ contains C, and the resulting set S^+ contains S. Indeed, if $v = v_i \in C$ for some i, it was chosen as v_i since its degree in G_{i-1} was high, and hence is placed in C^+ in the decoding process. Similarly, for $v = v_i \in S$. Otherwise, the degree of $v \in C$ in G_k is low, and the degree of $w \in S$ is high, so once again, they are correctly placed in C^+, S^+, respectively.

Let $\mathcal{E}$ denote the set of all sets C^+ produced by this decoding process from all possible sequences $v_1, v_2, \ldots, v_k, k = \lceil \log n \rceil$. Then $\mathcal{E}$ separates cliques and stable sets, since for each clique C and stable set S some sequence is the encoding of C, S. Moreover, $\mathcal{E}$ has at most $n^k = n^{\log n}$ elements. $\square$

We remark that to obtain the better bound ($2^{\binom{1+\log(n+1)}{2}} \leq n^{\frac{1}{2}\log n}$) we would describe the separators by binary sequences, as is explained in the proof of Theorem 4.7. (Recall that the better bound actually follows from Theorem 4.7 by letting $t = 2$.)

We have several generalizations of the theorem, which we will use to solve certain M-partition problems.

Let G be a graph. A *clique-pair* (or a *skew set*) in G is a pair of disjoint sets A, B of vertices of G such that each $a \in A$ is adjacent in G to each $b \in B$. An *stable-pair* (or a *disconnected set*) in G is a pair of disjoint sets A, B such that no $a \in A$ is adjacent in G to any $b \in B$. Note that when $A = A_i, B = A_j$ are parts of an M-partition then a clique-pair A_i, A_j corresponds to $m_{i,j} = 1$ and a stable pair A_i, A_j to $m_{i,j} = 0$. Thus cliques and stable sets are 1's and 0's (respectively) **on the diagonal** of M, and clique-pairs and stable-pairs are 1's and 0's (respectively) **off the diagonal** of M.

Let G be a graph. We say that a family $\mathcal{E}$ of subsets of $V(G)$ *separates cliques and stable-pairs*, if for any pair $C, (A, B)$, where C is a clique and (A, B) a stable-pair, such that C and $A \cup B$ are disjoint, some $E \in \mathcal{E}$ contains C, and is disjoint from A **or** disjoint from B. Similarly, we say that $\mathcal{E}$ *separates clique-pairs and stable-pairs*, if, for any pair $(A, B), (C, D)$, where (A, B) is a stable-pair and (C, D) a clique-pair, such that A, C are disjoint, and B, D are disjoint, some $E \in \mathcal{E}$ contains C and is disjoint from A, or contains D and is disjoint from B.

Theorem 4.2 *Every graph with n vertices has a family of $n^{\log n}$ sets that separate clique-pairs and stable-pairs.*

Moreover, such a family can be found in time $n^{\log n}$ times a polynomial in n.

Proof: Suppose A, B is a stable pair, and C, D a clique pair, in a graph G. We again define a valid encoding. Having seen the complete details above, we make the description here more concise. Thus a valid encoding of the pair $(A, B), (C, D)$ will be a sequence $v_1, v_2, \ldots, v_k$ of vertices of G obtained as follows: There will be two auxiliary sets U, W of vertices, initially both equal to $V(G)$. At each stage i, we define the vertex $v_i \in U$ to be either

- a vertex of A of *high degree* in W, i.e., adjacent to more than one half of the vertices in W,
- and remove from U the vertex v_i, and remove from W all the neighbours of v_i,

or

- a vertex of C of *low degree* in W, i.e., adjacent to at most one half of the vertices in W,
- and remove from U the vertex v_i, and remove from W all its non-neighbours.

Since the size of the set W is halved at each stage, we may again assume that $k < \log n$.

The decoding process is a little different. We shall be building two complementary pairs of sets, $A^+, B^+ = \overline{A^+}$ and $C^+, D^+ = \overline{C^+}$, such that $A \subseteq A^+, C \subseteq C^+$ or $B \subseteq B^+, D \subseteq D^+$. Initially all four sets A^+, B^+, C^+, D^+ are empty. We also have auxiliary sets U, W, both initially equal to $V(G)$, similar to the ones from the encoding procedure. We process the vertices $v_1, v_2, \ldots, v_k$ in this order. Once v_{i-1} has been processed, we consider the degree of v_i with respect to W.

- If v_i has high degree in W, we place it in A^+ and put all its neighbours in D^+, removing v_i from U and its neighbours from W.
- If v_i is of low degree, we place it in C^+ and put all its non-neighbours in B^+, removing v_i from U and its non-neighbours from W.

Note that A^+, C^+ are disjoint, and so are B^+, D^+ (but A^+ could have common elements with either B^+ or D^+).

Once all v_i have been processed, one of two things can happen: Either W has become empty – which means that every vertex is either in B^+ or in D^+, and so we have a pair of complementary sets B^+, D^+ with $B \subseteq B^+, D \subseteq D^+$, or W is still non-empty. In the latter case we know that we can place all vertices of high degree in W into the set C^+, and all vertices of low degree in W into the set A^+ – thus every vertex belonging to either C^+ or A^+, and so we have a pair of complementary sets A^+, C^+ with $A \subseteq A^+, C \subseteq C^+$, as claimed.

Let $\mathcal{E}$ be the family of all sets C^+ and D^+ obtained from all sequences $v_1, v_2, \ldots, v_k$. It follows that $\mathcal{E}$ separates clique-pairs and stable-pairs. □

An argument similar to that given for Theorem 4.2 will show:

Theorem 4.3 *Every graph with n vertices has a family of $n^{\log n}$ sets that separate cliques and stable-pairs.*

Moreover, such a family can be found in time $n^{\log n}$ times a polynomial in n. □

There is an important special case of this last theorem. For cliques and stable-pairs that **partition** the vertices of G, there is a polynomial separating family:

Theorem 4.4 *For every graph G with n vertices there exists a family of n sets, which separates all cliques C and all stable-pairs (A, B) with the property that A, B, C partition $V(G)$.*

Moreover, such a family of separators can be found in polynomial time.

Proof: Let G' be a minimal chordal extension of G, and let $v_1, v_2, \ldots, v_n$ be a perfect elimination ordering of G'. Minimal chordal extension of an arbitrary graph, and a perfect elimination ordering of a chordal graph, can be found in polynomial time [28]. It follows from the definition of a perfect elimination ordering that, for each $i = 1, 2, \ldots, n$, the set E_i consisting of v_i and all $v_j, j \geq i$ adjacent to v_i induces a clique in G'. Moreover, each clique of G' is contained in one of the cliques E_i - namely one with i being the first subscript such that v_i is present in the clique. We claim that the family $\mathcal{E} = \{E_1, E_2, \ldots, E_n\}$ satisfies the statement of the theorem.

Thus suppose C is a clique in G and (A, B) a stable-pair in G, such that A, B, C partition $V(G)$. Since G' is a minimal chordal extension of G, it will only contain an edge not in G if it is a chord in a chordless cycle of G. In particular, G' cannot have an edge joining a vertex of A to a vertex of B, since any cycle of G which contains both a vertex of A and a vertex of B must contain two vertices of C and thus a chord. Thus (A, B) is also a stable pair in G', and, of course, C is also a clique in G'. Thus some E_i contains C and is disjoint from A or from B. □

This result illustrates that it is sometimes possible to find **polynomial** separating families.

Here is how we can use these results to reduce certain complex list partition problems to simpler ones:

Corollary 4.5 *Suppose M has $XZ = 0$ and $YW = 1$. Then the list M-partition problem reduces to $n^{\log n}$ instances each of which has no list containing $\{X, Y\}$, or no list containing $\{Z, W\}$.*

In the special case $Z = X, W = Y$, i.e., $X = 0$ and $Y = 1$, the number of instances can be reduced to $n^{\frac{1}{2}\log n}$ (Theorem 4.1).

In the special case $X = Y$, i.e., $XZ = 0$ and $XW = 1$, the number of instances can be reduced just $n+1$ – with one instance having no list containing X, and n instances having no list containing both Y and Z (Corollary 2.4).

Proof. Suppose first that X, Z, Y, W are all different. Then any M-partition of an input graph G contains the clique-pair (Y, W), and the disjoint stable-pair (X, Z). According to Theorem 4.2 there is a family of $n^{\log n}$ sets E that separate all clique-pairs from all stable-pairs. For each set E we obtaine two instances – in one we remove X from all vertices in E and Y from all vertices not in E, and in the other we remove Z from all vertices in E and W from all vertices not in E. The other cases are treated similarly. □

We may define separators also in the sparse-dense model: A family $\mathcal{E}$ *separates dense sets and sparse sets* if for any pair of disjoint sets D (dense) and S (sparse) there is a set $E \in \mathcal{E}$ which contains D and is disjoint from S.

The next result concerns the case when sparse subgraphs are the a-colourable subgraphs and dense subgraphs the complements of b-colourable subgraphs:

Theorem 4.6 *Every graph with n vertices has a family of $n^{\frac{1}{2}ab \log n}$ sets that separate the a-colourable subgraphs and the complements of b-colourable subgraphs.*

Moreover, such a family can be found in time $n^{\frac{1}{2}ab \log n}$ times a polynomial in n.

Proof. We have already observed that a sparse graph can meet a dense graph in at most $c = ab$ vertices.

We know that $n^{\frac{1}{2}\log n}$ separators E are sufficient to separate each stable set from each clique. If we separate each of the a stable sets from a sparse subgraph and each of the b cliques from a dense subgraph, we obtain $c = ab$ such sets E. We can then construct a separator E' for the sparse and dense subgraphs by taking, for each of the b cliques, the intersection of the a separators E corresponding to the a stable sets, and then letting E' be the union of the b intersections corresponding to the b cliques. Since there are at most $n^{\frac{1}{2}\log n}$ separators E, and the separator E' is constructed from $c = ab$ such separators, the $n^{\frac{1}{2}c \log n}$ bound follows. □

Our last generalization concerns the case when sparse subgraphs are the $(a+1)$-clique-free subgraphs, and dense subgraphs are the $(b+1)$-stable-set-free subgraphs. Note that if we know that all stable sets are sparse and all cliques are dense, then sparse graphs are $(c+1)$-clique-free, and dense graphs are $(c+1)$-stable-set-free. Thus the following result can be used in all such situations; in particular it can be used when $a = b = 1$, i.e., for separating cliques and stable sets. We take this opportunity also to refine the arguments to obtain the better bounds.

Instead of using sequences of vertices to describe the separators, we shall be using binary sequences – we simply represent each vertex by a binary sequence. The number of such sequences is then 2 power the length of the sequence.

The bound in the Theorem is less than $2^{[\log^{(a+b)} n]/[(a+b)!]}$, which for the case $a = b = 1$ equals $2^{[\log^2 n]/2} = n^{\frac{1}{2}\log n}$.

Theorem 4.7 *Every graph with n vertices has a family of $2^{C(a+b,n)}$ sets that separate the $(a+1)$-clique-free subgraphs and the $(b+1)$-stable-set-free subgraphs, where*

$$C(t,n) \leq \binom{t+\log(n+1)}{t} - 1 - \log(n+1)$$

is the solution of the recurrence

$$C(t,0) = 0,$$

$$C(1,n) = 0,$$

$$C(t,n) = \log(n+1) + \max_{n/2<d<n} C(t-1,d) + C(t,n-d-1).$$

Moreover, such a family can be found in time $2^{C(a+b,n)}$ times a polynomial in n.

Proof. We shall encode the sparse-dense pairs by binary sequences. Equivalently, we may talk of sequences of vertices as before, but count each vertex as having a certain length. In fact, for this proof the sequences will use one additional special symbol, %. Thus, together with the n vertices of the input graph, we will have $n+1$ different symbols, and will encode these by giving each symbol a different binary sequence of length $\log(n+1)$. With this measure of length, we shall show how to represent separators by sequences of length $C(a+b,n)$.

The description is as follows. Suppose G is a given graph, and S, D a disjoint pair of sets, where S is $(a+1)$-clique-free, and D $(b+1)$-stable-set-free.

Suppose first that there is a vertex $v \in S$ of degree $d > n/2$. We shall describe the separator by first giving the binary sequence for v, followed by two binary sequences, one describing the pair $S' = S \cap N, D' = D \cap N$, where N is the set of neighbours of v, and the other describing the pair $S'' = S \setminus S', D'' = D \setminus D'$. In the decoding process, we will be able to tell that $v \in S$, and recursively decode the two subsequences to produce a correct separator for S, D in G. Note that the first sequence has length at most $C(a-1+b, d) \leq C(a-1+b, n)$, since S' must be a-clique-free (being a subset of S and completely adjacent to $v \in S$). On the other hand, the second sequence has length at most $C(a+b, n-d-1) \leq C(a+b, \lfloor n/2 \rfloor)$.

Similarly, if there is a vertex $w \in D$ of degree $d \leq n/2$, then the description will start with the binary sequence for w, followed by two sequences, one for the $n-d-1$ non-neighbours of w and the other one for the d neighbours of w. The lengths of these sequences are again $C(a+b-1, n-d-1) \leq C(a+b-1, n)$ and $C(a+b, d) \leq C(a+b, \lfloor n/2 \rfloor)$. Here we have used the fact that if w is not adjacent to any of the vertices in D, then removing it decreases the size of the largest stable set by one.

If neither v or w can be found, then we can define the separator to consist of all the vertices of degree greater than $n/2$. It is easy to see that this set contains D and is disjoint from S. We shall use the special symbol % to indicate in the sequence that this is the case. (We need such an indication when recursively decoding the sequence.)

If $a+b=1$, then $a=0$ and the sparse graph is empty, or $b=0$ and the dense graph is empty.

Thus, we have the recurrence stated in the theorem, which can be bounded by

$$C(t, n) \leq \log(n+1) + C(t-1, n) + C(t, \lfloor n/2 \rfloor).$$

The bound on $C(t, n)$ follows by induction, with base cases

$$\begin{aligned} \binom{t + \log 1}{t} - 1 - \log 1 &= 0, \\ \binom{1 + \log(n+1)}{1} - 1 - \log(n+1) &= 0, \end{aligned}$$

and inductive case

$$\binom{t+\log(n+1)}{t} = \binom{t-1+\log(n+1)}{t-1} + \binom{t+\log(\frac{n+1}{2})}{t}$$

and $\log(n+1) = 1 + \log(\frac{n+1}{2})$. $\Box$

5 Example Applications

In this section we shall illustrate the general techniques of the preceding sections on some important examples. In the following section we treat **all** the remaining partition problems with at most four parts. We give the proofs in this section in full detail, allowing us to abbreviate the similar proofs given in the next section.

5.1 The List Version of Generalized Split Graphs

We first return to the case of generalized split graphs. Recall that G is an (a,b)-graph if its vertices can be partitioned into a stable sets $A_1, A_2, \ldots, A_a$ and b cliques $A_{a+1}, A_{a+2}, \ldots, A_{a+b}$, i.e., if and only if G has an M-partition where M is an $(a+b)$ by $(a+b)$ matrix with all off-diagonal entries equal to $*$ and with the first a diagonal entries equal to 0 and the last b diagonal entries equal to 1. We shall show how Theorem 3.1 implies a polynomial-time algorithm to recognize (a,b)-graphs when $a, b \leq 2$. In fact, we shall solve the list version of these problems:

Corollary 5.1 *If both $a \leq 2$ and $b \leq 2$, then the list M-partition problem is polynomial-time solvable. Otherwise it is NP-complete.*

Proof. First we note that if $a \geq 3$ then the list M-partition problem is NP-complete, since we can decide whether or not an input graph G is 3-colourable by endowing all its vertices with the list $\{1,2,3\}$ and then ask whether or not it has a list M-partition. (If $b \geq 3$ the proof is similar.)

Thus assume that both $a \leq 2$ and $b \leq 2$. Let $\mathcal{S}$ be the class of all a-colourable graphs, and $\mathcal{D}$ the class of all graphs with b-colourable complements. Note that both classes can be recognized in polynomial time. According to Theorem 3.1 we can generate, in polynomial time, all sparse-dense partitions of any input graph G.

Suppose G with lists L is an instance of the list M-partition problem. For each sparse-dense partition of G, we update the lists of the vertices as follows: If v belongs to the sparse part $(A_1 \cup A_2 \cup \ldots \cup A_a)$ we remove all elements of $\{a+1, a+2, \ldots, a+b\}$ from $L(v)$ (if present). If v belongs to the dense part $(A_{a+1} \cup \ldots \cup A_{a+b})$ we remove all elements of $\{1, 2, \ldots, a\}$ from $L(v)$ (if present). The resulting instance has all lists of size at most two, and hence can be solved by 2-satisfiability (see Proposition 2.1). (Note that it is possible that some lists have become empty.) It is clear that G has a list M-partition with respect to the original lists L if and only if it has a list M-partition with respect to at least one of the modified lists. $\Box$

The corollary yields algorithms for all the polynomial generalized split graph recognition problems [4, 7]. Specifically, it gives polynomial time algorithms for the recognition of split graphs, $(2,1)$-graphs, $(1,2)$-graphs, and $(2,2)$-graphs. All other (a,b)-graph recognition problems are NP-complete [4, 7].

5.2 The List Clique Cutset Problem

The best known polynomial time solvable three-part partition problem is the Clique Cutset Problem, i.e., $C = 1, AB = 0$, all others $*$. In [22] we have shown that the list version of this problem can be reduced, in polynomial time, to the list-free version solved by polynomial time algorithms of Tarjan [35] and Whitesides [39, 40]. Here we give a direct polynomial time algorithm for the List Clique Cutset Problem. It is motivated by the original algorithms [35, 39, 40], and it follows from one of our separator theorems – Theorem 4.4.

Corollary 5.2 *There is a polynomial time algorithm for the List Clique Cutset Problem.*

Proof: The theorem yields a polynomial size family $\mathcal{E}$ such that whenever an input graph G has a partition A, B, C with $C = 1, AB = 0$, some $E \in \mathcal{E}$ contains C and is disjoint from A or from B. Thus for each E from the family we will do two tests: In both tests, we remove C from all the lists of the vertices that do not belong to E. For the vertices that belong to E, we remove A in the first test, and B in the second test. This ensures that if a partition exists, one of the tests will succeed. Each test can be performed in polynomial time by Proposition 2.1. $\Box$

5.3 The List Skew Cutset Problem

The best known four-part partition problem is the Skew Cutset Problem, i.e., $AC = 0$, $BD = 1$, and all others $*$. The complexity of this problem was a well-known open problem in the theory of perfect graphs [9]. In [22] we presented the first subexponential algorithm for the problem, strongly suggesting that it was not NP-complete. Since then, a polynomial algorithm has been found by one of us (Klein), with de Figueiredo, Kohayakawa, and Reed [25]. It is worth noting that although the algorithm uses a different technique from the ones given here, it still very much uses the flexibility of recursively reducing the problem by modifying the lists. Lists in partition problems have turned out to be very useful. Below we show how a quasipolynomial algorithm for the list skew cutset problem follows from Theorem 4.2.

Corollary 5.3 *The List Skew Cutset Problem can be solved in time $n^{\log n}$ times a polynomial in n.*

Proof. Since (B, D) is a clique-pair, and (A, C) a disjoint stable-pair, we can apply Theorem 4.2. For each E from the family $\mathcal{E}$ generated by Theorem we perform two tests: The first test assumes that E contains B and is disjoint from A – thus deleting A from all the lists of the vertices that belong to E, and deleting B from the vertices that do not belong to E. The second test assumes that E contains D and is disjoint from C, also updating the lists accordingly. During the first test, no list has both A and B. If at any point a vertex has a list of size one, we eliminate it and restrict its neighbours and non-neighbours accordingly. Thus we arrive at a situation that every vertex has a list of size at least two, but never contains both A and B. This means that every list contains C or D. But $C = D = CD = *$, so we can freely choose to put all vertices to either C or D, according to their lists. The second test is done similarly. □

5.4 The List Two-Clique Cutset Problem

As an application of our Theorem 4.3 we give a quasipolynomial bound for the *two-clique cutset problem*, that is, $AC = 0, B = D = 1$, all others equal to $*$. This example is also interesting because it illustrates how we can use a separator theorems twice. (This is a recurring theme in the general classification of matrices with $k = 4$.)

Corollary 5.4 *The List Two-Clique Cutset Problem can be solved in time $n^{2\log n}$ times a polynomial in n.*

PROOF. Let $\mathcal{E}$ be a family of $n^{\log n}$ sets that separates cliques and stable pairs, as guaranteed by Theorem 4.3. If the input graph G admits a partition as specified above, then some E_1 will contain B and be disjoint from A or C, and some E_2 will contain D and be disjoint from A or C. For **any two** elements E_1, E_2 of $\mathcal{E}$, we shall make four tests: In the first test we assume that E_1 contains B and is disjoint from A, while E_2 contains D and is also disjoint from A; in the second test we assume E_1 contains B and is disjoint from A, while E_2 contains D and is disjoint from C. The third and fourth tests are defined similarly (assuming that E_1 is disjoint from C). As before, the tests are performed by correspondingly modifying the lists of the elements inside or outside of E_1, E_2 respectively. The second test results in all lists of size two, as does one of the last two tests; these can be solved again by Proposition 2.1. Consider the first test (the remaining test is done similarly). No list has both A, B and no list has both A, D. If there are any lists of size three, they must be $\{B, C, D\}$. We can again assume that there are no lists of size one. We are now in the following situation: If a list has no C then it must be $\{B, D\}$. Since $C = BC = BD = CD = *$, we can complete the test by placing all vertices that have C in their list to C, and checking that those vertices with lists $\{B, D\}$ can be partitioned into two cliques, i.e., that the graph they induce has a bipartite complement. □

We note that there is no polynomial time algorithm known for this problem.

6 Classifications for Small Matrices M

Recall that most of our motivating examples of M-partitions dealt with small values of k. Split graphs have $k = 2$; stable set partition, clique partition, and homogeneous set have $k = 3$; skew partition and the problem of Winkler have $k = 4$; and, so on (cf. Figures 2,3,4). This suggests a systematic investigation of M-partition problems with small k. Here, we focus on the case $k \leq 4$.

When $k = 2$, all list M-partition problems are polynomial-time solvable, by Proposition 2.1.

Theorem 6.1 *Suppose the size of M is $k = 3$. Then the list M-partition problem is NP-complete when M or its complement is the matrix of 3-colouring or the stable cutset problems (Figure 3), and is polynomial-time solvable otherwise.*

PROOF. The NP-complete cases are standard results [27, 24].

Consider a matrix M with rows A, B, and C and connections AB, AC, and BC, which is different from the four exceptional matrices described in the theorem. We may assume that M is connected, thus at most one of the connections is 0, and, by complementation, at most one of the connections is 1. We may also assume that no row has both a 0 and a 1, cf. Corollary 2.4 and Proposition 2.1. In particular, we do not have both a connection of type 0 and a connection of type 1. Thus without loss of generality we may assume that $AC = BC = *$. In that case we may also assume that $C \neq *$, otherwise C dominates all other rows, and we can eliminate it as explained after Proposition 2.5.

If each part A, B, and C is of type either 0 or 1, and not all are the same type, then the problem is polynomial-time solvable by Theorem 3.1, since after we have decided which vertices go to parts of type 0 and which to parts of type 1, we are left with each list of size one or two.

If $AB = *$ as well, then we also have $A, B \neq *$ and so we may assume that all three values A, B, and C are the same, either 0 or 1. This is impossible, as M is matrix different from the matrix of 3-colouring and its complement.

Up to complementation we may assume that $AB = 0$. We may also assume that $A, B \neq 1$, else we have a row with both a 0 and a 1. If one of A and B dominates the other, we reduce all lists to size at most two. The only other possibility is that $A = B = *$. Since M is not the matrix of the stable-cutset problem we conclude that $C = 1$, i.e., it is the matrix of the clique-cutset problem solved in the previous section. □

We now proceed to discuss matrices of size $k = 4$. It will be easier to deal first with matrices M which have no $*$'s on the main diagonal:

Theorem 6.2 *Suppose M is of size $k = 4$ and assume it does not contain any $*$'s on the main diagonal.*

- *If M contains the matrix corresponding to 3-colouring, or its complement, then the list M-partition problem is NP-complete;*
- *otherwise, the list M-partition problem is polynomial-time solvable.*

PROOF. The first statement follows from Propositions 2.7 and 2.8. Thus assume that M does not contain the matrix corresponding to 3-colouring, or its complement.

If there are two parts of type 0 and two of type 1, we proceed as the case of $k = 3$, solving, for each sparse-dense partition, the remaining problem by 2-satisfiability. If there are three parts of type 0 and one part of type 1 (a similar argument applies when there are three 1's and one 0), then we proceed the same way, since the three 0 parts do not yield 3-colouring and all other three-part problems without a diagonal 1 are polynomial time solvable. Once the vertices that go to the part of type 1 are known, we can remove them, modifying the remaining lists, and solve the three-part subproblem.

By complementation, we may now assume that all four parts are of type 0.

Suppose there are two disjoint connections of type 1, say $AB = CD = 1$. Then we may try to place two vertices — v into A and w into C: for vertices adjacent to both v and w we can remove A and C from the lists, for those nonadjacent to both we can remove B and D, for those adjacent to v but not to w we can remove A and D, and for those adjacent to w but not v we can remove B and C. Thus we can reduce the problem to the following polynomial-time solvable instances: One instance with lists without A, one instance with lists without C, and at most n^2 instances with lists of size at most two (corresponding to all possible choices of v, w).

If there are three connections of type 1 incident with one part, then M is disconnected, and we can solve the problem by solving each connected part separately, since M does not contain 3-colouring. On the other hand, if there are three connection of type 1 which form a triangle, say $AB = BC = AC = 1$, then we can reduce the problem to at most $n^2 + 2$ polynomial-time solvable cases by trying to place one vertex in A and one in B.

Thus it remains to consider the case of at most two connections of type 1. If there are two such connections then we may assume that they both touch on A, thus, say, $AB = AC = 1$. If $AD = 0$ then trying to place a vertex in A leads to $n + 1$ polynomial-time solvable instances. Thus we may assume that $AB = AC = 1, AD = *$. In this case D dominates B or C unless $BD = CD = 1$ or $BC = 1, BD = CD$. Thus we may assume that no list contains both B and D or no list contains both C and D. Now we can reduce each of these problems to $n + 1$ polynomial-time solvable cases by trying to place a vertex in A. The same technique (trying to place a vertex to A) also works when $BD = CD = 1, BC = 0$, since then C dominates

B. Note that we cannot have $BD = CD = BC = 1$, since M does not contain 3-colouring. Thus we are left with the case $BC = 1, BD = CD = 0$ (and $AB = AC = 1, AD = *$). In this case we can reduce the problem to one instance of lists without A, one instance of lists without B, and n^2 instances of a vertex v placed into A and a vertex w into B. The former two problems are polynomial-time solvable; the latter problem can also be solved in polynomial time, since vertices adjacent to w and nonadjacent to v must map to A, while no other vertex can map to A – hence we can remove the vertices that map to A and solve the three-part subproblem.

Suppose there is only one connection of type 1, namely $AB = 1$. If $AX = BY = 0$ for some (possibly equal) X, Y, then we can again reduce the problem to n^2+2 polynomial-time solvable instances by trying to place a vertex in A and a vertex in B. Thus we may assume that, say, $AB = 1, AC = AD = *$. Since M does not contain 3-colouring, we must have $CD = 0$, and it is easy to see that in the remaining cases one of C, D dominates the other; if no list contains both C and D, then the problem can be reduced to $n+1$ instances of polynomial-time solvable instances by trying to place a vertex in A or B. We note that a linear-time algorithm for (the complement of) the problem $AB = 1, AC = AD = 1, BC = BD = 1, CD = 0$ (which is one of the problems considered in this paragraph) was given by Everett, Klein, and Reed [17].

If there are no connections of type 1, then we have a list homomorphism problem, solvable in polynomial time (see Lemma 6.4.1 below, or [20]). □

Note that the M-partition problem (without lists) is trivial if there is a part of type $*$ (all vertices can be placed in it). Thus the Theorem allows a complete classification of the M-partition problem without lists:

Corollary 6.3 *If the size of M is $k = 4$, then the M-partition problem (without lists) is*

- *NP-complete when M contains the matrix of 3-colouring or its complement, and no diagonal entry is $*$;*
- *and is polynomial-time solvable otherwise.*

Proof. The polynomial algorithms follow from the Theorem and the above remark.

Suppose M contains the matrix of 3-colouring; say, M is the matrix

$$\begin{pmatrix} 0 & * & * & x_1 \\ * & 0 & * & x_2 \\ * & * & 0 & x_3 \\ x_1 & x_2 & x_3 & y \end{pmatrix}$$

(When M contains the complement of the matrix of 3-colouring, the argument is similar.) By assumption, $y \neq *$.

Suppose first that $y = 1$. We prove the NP-completeness of M-partition by reducing to it the problem of 3-colourability. Thus suppose that G is a graph we would like to 3-colour, and let G' consist of two disjoint copies of G. We claim that G is 3-colourable if and only if G' admits an M-partition. Indeed, if G is 3-colourable, then G' is also 3-colourable, and hence admits an M-partition (with all vertices in the first three parts). On the other hand, an M-partition of G' cannot place two vertices from different copies of G in the fourth part, since the fourth part is a clique. Thus all vertices of one copy of G are placed in the first three parts, i.e., G is 3-colourable.

Now suppose that $y = 0$. If any other $x_i = 0$, then the union of the i-th part and the fourth part is a stable set, and a graph admits an M-partition if and only if it is 3-colourable. If any $x_i = 1$, then it is again the case that G is 3-colourable if and only if G' (from above) admits an M-partition. Indeed, an M-partition of G' cannot place both a vertex from the first copy of G to the i-th part and a vertex from the second copy in the fourth part, since those parts are completely adjacent. Therefore at least one copy of G is 3-coloured. □

We now turn to the general list M-partition problem, where M may have parts of type $*$. We are no longer able to classify the problems as NP-complete or polynomial, but we do give a classification as NP-complete or quasipolynomial.

Theorem 6.4 *Suppose the size of M is $k = 4$. The list M-partition problems are quasipolynomial (sometimes polynomial) or NP-complete.*

Proof. If M has no 1 (or equivalently, by complementation, no 0), then the $0, *$ matrix M is obtained from the adjaceny matrix of a graph H by replacing all 1's with $*$'s, and the list M-partition problem is the previously studied list H-colouring problem:

Lemma 6.4.1 *If M is a $(0, *)-$ matrix of size $k = 4$, then the list M-partition problem is polynomial or NP-complete.*

PROOF. Suppose first that the size k of M is arbitrary, and assume (by a components argument) that the graph H is connected.

If all diagonal entries are $*$, that is, if all vertices of H have a loop, then it is shown in [19] that the problem is polynomial if H is an interval graph, and NP-complete otherwise.

If all diagonal entries are 0, that is, if no vertex of H has a loop, then it is the problem is NP-complete if H is not bipartite, by [30]. In fact, it is shown in [20] that the problem is also NP-complete when the complement of H is not a circular arc graph, and is polynomial-time solvable in the remaining case of a bipartite H whose complement is a circular arc graph.

The remaining case, at least one 0 and one $*$ on the main diagonal, that is, some vertex of H has a loop and some vertex does not, is studied in [21]. A condition is given there (a kind of geometric representability of H, similar to having a complement representable by circular arcs), and it is shown that the list H-colouring problem is NP-complete unless H satisfies this condition. It is not known whether the problem is otherwise polynomial; however, this is shown in the case where H is a tree [21].

We are thus left with the case where H is not a tree and has at least one loop (L) and one nonloop (N). We now assume again that the size of M is $k = 4$. The problem is NP-complete if it contains one of the two NP-complete cases on three vertices: 3-colouring corresponds to H being a triangle with no loops, and stable cutset corresponds to H being an induced path LNL. We show that the remaining cases are polynomial.

Since H is not a tree, the longest cycle in H is either a triangle or a fourcycle. If it is a triangle ABC, then D is connected only, say, to A. At least one of A, B, C must have a loop. If it is A, then A dominates all parts. By Proposition 2.5 we may assume all vertices have lists which are $\{A\}$ or a subset of $\{B, C, D\}$. The former vertices can be placed to A using Corollary 2.4, and the remaining problem on $\{B, C, D\}$ is polynomial (by Theorem 6.1). If A has no loop, then say B has a loop, and then D cannot have a loop because DAB would be an LNL path. Then D is dominated by both B and C, so we can assume all lists are contained in $\{A, D\}$ or in $\{A, B, C\}$. Furthermore, regardless of whether or not C has a loop, B dominates C, so we can assume no list contains both B and C, so all lists are of size at most two, and the problem can be solved by Proposition 2.1.

If the longest cycle in H is a fourcycle $ABCD$, possibly with one or two chords, we proceed as follows. If both chords AC and BD are present, then H is a clique with at least one loop, say at A. But then A dominates all parts, and we obtain a problem on $\{B, C, D\}$ as before. If only one chord, say AC, is present, then if either A or C has a loop, it dominates all parts and we proceed as before. If neither A nor C has a loop, then both B and D must have loops (else there is a triangle without loops), but then DAB is an LNL path, the other NP-complete case.

In the remaining case, H is a fourcycle $ABCD$ without chords, at least one loop and one nonloop. To avoid an LNL path, opposite vertices on the cycle cannot both have loops, so there is either a single loop or two adjacent loops. If there is a loop at B, then A and C dominate each other, and can be replaced by a single part, thus obtaining a polynomial three-part problem. If there are two loops at B and C, then B dominates D, and C dominates A, so we can assume no list contains $\{B, D\}$ or $\{A, C\}$, and the problem can be solved by Proposition 2.1. □

By the preceding lemma, we can assume that M has at least one 0 and at least one 1. It turns out then that the only NP-complete problems are those mentioned earlier, namely for those matrices containing the matrix of 3-colouring or of stable cutset, or their complements.

The next two lemmas cover the cases where M has an off-diagonal 0 and an off-diagonal 1.

Lemma 6.4.2 *Suppose $AC = 0$, $BD = 1$. Then list M-partition is quasipolynomial or NP-complete.*

Proof. Note that this case includes the list skew cutset problem solved in quasipolynomial time in Corollary 5.3. The proof below is written in an abbreviated style; the full details could be written out in a manner similar to the proof of Corollary 5.3.

Since $AC = 0$ and $BD = 1$, we can assume that no list contains $\{A, B\}$ or no list contains $\{C, D\}$ ($n^{\log n}$ cases); also, we can assume no list contains $\{A, D\}$ or no list contains $\{B, C\}$ ($n^{\log n}$ cases). (These are obtained by applying Theorem 4.2 in two ways.) The four possibilities are similar, so say there is no $\{A, B\}$ and no $\{A, D\}$. That is, all lists are contained in $\{A, C\}$ or in $\{B, C, D\}$. We than have the following:

$C \neq 1$, else place vertex in C, no $\{A, C\}$ (Corollary 2.4 applies, since $C = 1$ and $AC = 0$), hence drop A and solve the polynomial problem with the three parts B, C, D.

$C \neq 0$, else no $\{B, C\}$ or no $\{C, D\}$ ($n^{\log n}$ cases from Theorem 4.3, which applies as $C = 0$ and $BD = 1$), and we can solve these problems by 2-satisfiability, cf. Proposition 2.1.

Therefore we must have $C = *$. We also have the following:

$A \neq 1$, else place vertex in A, no $\{A, C\}$ (as $A = 1$ and $AC = 0$), drop A and solve the polynomial $\{B, C, D\}$ problem.

$B \neq 0$, else place vertex in B, no $\{B, D\}$ (as $B = 0$ and $BD = 1$), solve the 2-satisfiability problem.

$D \neq 0$, else place vertex in D, no $\{B, D\}$ (as $D = 0$ and $BD = 1$), solve the 2-satisfiability problem.

$AB \neq 1$, else place vertex in A, no $\{B, C\}$ (as $AB = 1$ and $AC = 0$), solve the 2-satisfiability problem.

$AD \neq 1$, else place vertex in A, no $\{C, D\}$ (as $AC = 0$ and $AD = 1$), solve the 2-satisfiability problem.

$BC \neq 0$, else place vertex in B, no $\{C, D\}$ (as $BC = 0$ and $BD = 1$), solve the 2-satisfiability problem.

$CD \neq 0$, else place vertex in D, no $\{B, C\}$ (as $BD = 1$ and $CD = 0$), solve the 2-satisfiability problem.

So far, we have $AC = 0$, $BD = 1$, $C = *$, $A \neq 1$, $B \neq 0$, $D \neq 0$, $AB \neq 1$, $AD \neq 1$, $BC \neq 0$, $CD \neq 0$.

In addition we have not both $BC = CD = *$, otherwise C dominates A, no $\{A, C\}$, drop A and solve the polynomial $\{B, C, D\}$ problem.

We may assume $BC = 1$ (by symmetry between B and D).

Now, we have no $\{A, C\}$ or no $\{B, C\}$ ($n^{\log n}$ cases as $AC = 0$ and $BC = 1$), so either drop A to get a $\{B, C, D\}$ problem, or get a 2-satisfiability problem. □

Lemma 6.4.3 *Suppose $AB = 0$, $AD = 1$. Then list M-partition is quasipolynomial or NP-complete.*

Proof. Place vertex in A, no $\{B, D\}$. The we have

$BC \neq 0$ and $CD \neq 1$, as the case of disjoint connections with value 0 and 1 was covered by the previous lemma.

$BC \neq 1$, else place vertex in B, no $\{A, C\}$ (as $AB = 0$ and $BC = 1$), solve the 2-satisfiability problem. So $BC = *$.

$CD \neq 0$, else place vertex in D, no $\{A, C\}$ (as $AD = *$ and $CD = 0$), solve the 2-satisfiability problem. So $CD = *$.

So far, we have $AB = 0$, $AD = 1$, $BC = *$, $CD = *$.

Suppose $C = *$. Then $AC \neq *$, else C dominates all and could be dropped. By symmetry under complementation, we can assume $AC = 0$. Also C dominates B, so no $\{B, C\}$. Place vertex in A, no $\{C, D\}$ (as $AC = 0$ and $AD = 1$). So a list can contain only one of $\{B, C, D\}$, solve the 2-satisfiability problem.

For the other case, $C \neq *$, by symmetry under complementation, we can assume $C = 0$. We also have

No $\{A, C\}$ or no $\{C, D\}$ ($n^{\log n}$ cases as $C = 0$ and $AD = 1$). If no $\{A, C\}$, since no $\{B, D\}$, solve the 2-satisfiability problem. So no $\{C, D\}$. As a result, all lists are contained in $\{A, D\}$ or in $\{A, B, C\}$.

Now, no $\{A, D\}$ or no $\{A, B\}$ ($n^{\log n}$ cases as $AB = 0$, $AD = 1$), so either drop D to get a $\{A, B, C\}$ problem, or get a 2-satisfiability problem. $\square$

By these two lemmas, we can assume that 1 (or equivalently 0, by complementation) occurs only on the diagonal, so that all off-diagonal entries are $0, *$.

We first consider the case where there are at least two 1s (on the main diagonal). For this case, we can assume that there is at least one 0 not on the main diagonal. Otherwise, if all off-diagonal entries are $*$, then if say $A = *$, then A dominates all other parts, and we obtain a size three problem on $\{B, C, D\}$; if none of A, B, C, D is $*$, then either they are two 0's and two 1's (polynomial by the sparse-dense technique), or we get an NP-complete problem by 3-colourability.

The next three lemmas consider the possible placements of the 0 connection with respect to the (at least two) 1 parts. Either the 0 connection is not incident on any of the two 1's, or it is incident on one of them, or it is incident on both of them.

Lemma 6.4.4 *Suppose all off-diagonal entries are* $0, *$, *and* $B = D = 1$, $AC = 0$. *Then list* M*-partition is quasipolynomial or* NP*-complete.*

PROOF. There is no $\{A, B\}$ or no $\{B, C\}$ ($n^{\log n}$ cases as $B = 1$, $AC = 0$).

There is no $\{A, D\}$ or no $\{C, D\}$ ($n^{\log n}$ cases as $D = 1$, $AC = 0$).

If there is no list of size three, solve 2-satisfiability. A list of size three can only be $\{A, B, D\}$ or $\{B, C, D\}$. By symmetry, assume it is $\{B, C, D\}$. So all lists are contained in $\{A, C\}$ or in $\{B, C, D\}$.

We have

$C \neq 1$, else place vertex in C, no $\{A, C\}$ (as $C = 1$ and $AC = 0$), drop A and get a three-part problem on $\{B, C, D\}$.

$C \neq 0$, else no $\{B, C\}$ ($n^{\frac{1}{2}\log n}$ cases as $C = 0$ and $B = 1$), solve 2-satisfiability. So $C = *$.

$BC \neq 0$, else place vertex in B, no $\{B, C\}$ (as $B = 1$ and $BC = 0$), solve 2-satisfiability. So $BC = *$.

$CD \neq 0$, else place vertex in D, no $\{C, D\}$ (as $D = 1$ and $CD = 0$), solve 2-satisfiability. So $CD = *$.

Now all $\{A, C\}$ vertices can be put in C without loss of generality, so drop A and get a three-part problem on $\{B, C, D\}$. $\square$

Lemma 6.4.5 *Suppose all off-diagonal entries are* $0, *$, *and* $A = B = 1$, $AC = 0$. *Then list* M*-partition is quasipolynomial or* NP*-complete.*

PROOF. We can assume $D \neq 1$ and $CD \neq 0$ by the previous lemma, so $CD = *$.

Place a vertex in A, no $\{A, C\}$ (as $A = 1$ and $AC = 0$).

Also no $\{A, B\}$ or no $\{B, C\}$ ($n^{\log n}$ cases as $B = 1$, $AC = 0$).

Suppose first no $\{A, B\}$. All lists are contained in $\{A, D\}$ or in $\{B, C, D\}$.

We have

$D \neq 0$, else no $\{A, D\}$ ($n^{\frac{1}{2}\log n}$ cases as $D = 0$, $A = 1$), so drop A and get a three-part $\{B, C, D\}$ problem. So $D = *$.

$AD \neq 0$, else place vertex in A, no $\{A, D\}$ (as $A = 1$ and $AD = 0$), so drop A and get a three-part $\{B, C, D\}$ problem. So $AD = *$.

$BD \neq 0$, else place vertex in B, no $\{B, D\}$ (as $B = 1$ and $BD = 0$), solve 2-satisfiability problem. So $BD = *$.

But now, D dominates all vertices, so drop D and get a three-part $\{A, B, C\}$ problem.

Suppose instead no $\{B, C\}$. All lists are contained in $\{C, D\}$ or in $\{A, B, D\}$.

We have

$D \neq 0$, else no $\{A, D\}$ ($n^{\frac{1}{2}\log n}$ cases as $D = 0$, $A = 1$), solve 2-satisfiability. So $D = *$.

$AD \neq 0$, else place vertex in A, no $\{A, D\}$ (as $A = 1$ and $AD = 0$), solve 2-satisfiability. So $AD = *$.

$BD \neq 0$, else place vertex in B, no $\{B, D\}$ (as $B = 1$ and $BD = 0$), solve 2-satisfiability. So $BD = *$.

But now, D dominates all vertices, so drop D and get a three-part $\{A, B, C\}$ problem. $\square$

Lemma 6.4.6 *Suppose all off-diagonal entries are* $0, *$, *and* $A = C = 1$, $AC = 0$. *Then list* M*-partition is quasipolynomial or* NP*-complete.*

PROOF. By the last two lemmas, we can assume $AB = BC = AD = BD = CD = *$, and also $B \neq 1$, $D \neq 1$. If $B = *$, then B dominates all vertices, so drop B and solve $\{A, C, D\}$ problem. Similarly, if $D = *$, then D dominates all vertices, so drop D and solve $\{A, B, C\}$ problem. So $B = D = 0$. The problem is then the problem of recognizing $(2,2)$-graphs, which is solved in polynomial time by Corollary 5.1. □

We are now left with the case where M has a single 1, and this 1 is on the diagonal, say $D = 1$.

For convenience, we define the *separating statement* for $x = A, B, C$ to be $xD = 0$ or $x = 0$. If this statement holds for x, then there is no $\{x, D\}$, either by placing a vertex in D (as $D = 1$ and $xD = 0$), or $n^{\frac{1}{2}\log n}$ cases for $D = 1$ and $x = 0$.

If all three separating statements hold, then there is no $\{x, D\}$ for $x = A, B, C$, so drop D and solve the three-part $\{A, B, C\}$ problem.

Suppose next exactly two separating statements hold, say for A, B. So all lists are contained in $\{C, D\}$ or in $\{A, B, C\}$, and $C = CD = *$. The three possible cases are covered by the next three lemmas.

Lemma 6.4.7 *Suppose there is a single* 1 *at* $D = 1$*, and* $AD = BD = 0$*,* $C = CD = 1$*. Then list* M*-partition is quasipolynomial or* NP*-complete.*

Proof. We may assume that not both $AC = BC = *$, else C dominates all vertices, so we can drop C and get a three-part $\{A, B, D\}$ problem.

Say $AC = 0$.Then $BC \neq 0$, else we have two components $\{A, B\}$ and $\{C, D\}$. So $BC = *$.

We have

$A \neq 0$, else C dominates A, no $\{A, C\}$, solve 2-satisfiability problem. (So $A = *$.)

$AB \neq 0$, else C dominates B, no $\{B, C\}$, solve 2-satisfiability problem. (So $AB = *$.)

$B \neq *$, else B dominates A, no $\{A, B\}$, solve 2-satisfiability problem. (So $B = 0$.)

The remaining problem on $\{A, B, C\}$ is the stable cutset problem, which is NP-complete. □

Lemma 6.4.8 *Suppose there is a single* 1 *at* $D = 1$*, and* $A = B = 0$*,* $C = CD = *$*. Then list* M*-partition is quasipolynomial or* NP*-complete.*

Proof. We may asume that not both $AC = BC = *$, else C dominates all vertices, drop C and get a three-part $\{A, B, D\}$ problem.

Say $AC = 0$. Then $AB = *$ and $BC = 0$, else C dominates A, no $\{A, C\}$, solve 2-satisfiability problem.

Each connected component of the subgraph induced by the vertices with lists included in $\{A, B, C\}$ can go to $\{A, B\}$ or to $\{C\}$, but it can always be put in $\{C\}$ if C is in the lists. Solve 2-satisfiability problem. □

Lemma 6.4.9 *Suppose there is a single* 1 *at* $D = 1$*, and* $AD = B = 0$*,* $A = BD = *$*,* $C = CD = *$*. Then list* M*-partition is quasipolynomial or* NP*-complete.*

Proof. $AC = 0$ and $AB = *$, else C dominates B, no $\{B, C\}$, solve 2-satisfiability problem.

If $BC = *$, the problem on $\{A, B, C\}$ is the stable cutset problem, which is NP-complete.

If $BC = 0$, then each connected component of the subgraph induced by the vertices with lists included in $\{A, B, C\}$ can go to $\{A, B\}$ or to $\{C\}$, but it can always be put in $\{C\}$ if C is in the lists. Solve the 2-satisfiability problem. □

The remaining case with a single 1 at $D = 1$ has at most one separating statement holding, say for A.

Lemma 6.4.10 *Suppose there is a single* 1 *at* $D = 1$*,* $B = BD = C = CD = *$*. Then list* M*-partition is quasipolynomial or* NP*-complete.*

Proof. If $BC = *$, then one of B, C dominates all vertices and can be dropped, to obtain a three-part problem, unless $AB = AC = 0$ (and $A = *$), in which case the rows of B and C are identical, so B and C can be collapsed to a single part.

So $BC = 0$. We consider various cases of the values of (A, AB, AC):

$(0, *, *)$ is the stable cutset problem, which is NP-complete.

$(*, *, *)$: if also $AD = *$, A dominates all vertices and can be dropped, to obtain a three-part problem on $\{B, C, D\}$. If $AD = 0$, then place a vertex in D, no $\{A, D\}$ (as $D = 1$ and $AD = 0$). Also, we have no $\{B, D\}$ or no $\{C, D\}$ ($n^{\log n}$ cases, as $D = 1$ and $BC = 0$). Say, there is no $\{B, D\}$. Then all lists are contained in $\{C, D\}$ or in $\{A, B, C\}$, and can be assumed to be of size at last two. Now place all vertices with the list $\{C, D\}$ in C, and place all vertices with lists contained in $\{A, B, C\}$ in either A or C. Since $A = C = AC = 1$, this is a solution.

$(*, 0, *)$ with $AD = 0$, $(0, 0, 0)$, and $(0, 0, *)$: all three have no $\{A, D\}$ (place a vertex in D in the first case as $AD = 0$ and $D = 1$, in the other two cases we get $n^{\frac{1}{2}\log n}$ cases as $A = 0$ and $D = 1$). Also no $\{B, D\}$ or no $\{C, D\}$ ($n^{\log n}$ cases as $D = 1$ and $BC = 0$). So $\{A, B, C\}$ is the only 3-element list, but C dominates A in all three cases, no $\{A, C\}$, and we can solve the 2-satisfiability problem.

$(*, 0, *)$ with $AD = *$ has the rows of A and C identical, so A and C can be collapsed to a single part.

$(*, 0, 0)$ has $n^{\log n}$ cases each of no $\{A, D\}$ or no $\{B, D\}$ (as $D = 1$ and $AB = 0$), no $\{A, D\}$ or no $\{C, D\}$ (as $D = 1$ and $AC = 0$), no $\{B, D\}$ or no $\{C, D\}$ (as $D = 1$ and $BC = 0$). So there is at most one of $\{A, D\}$, $\{B, D\}$, $\{C, D\}$, and all other lists contained in $\{A, B, C\}$. For lists contained in $\{A, B, C\}$, the connected components go to a single one of A, B, or C, and C can always be preferred if possible. Solve the resulting 2-satisfiability problem. □

This completes the proof of the theorem. □

7 General k

Some results apply to arbitrarily sized matrices. For instance, recall that we have given in [19, 20, 21] a near-complete classification of the complexity of list M-partition when M is a $(0, *)$-matrix. By complementation, we obtain similar results for $(*, 1)$-matrices (cf. Proposition 2.8). When M is a $(0, 1)$-matrix, we can apply our Theorem 3.1:

Corollary 7.1 *If M is a $(0, 1)$-matrix, then the list M-partition problem is polynomial-time solvable.*

Proof. Once we know from Theorem 3.1 which vertices are placed in parts of type 0 (stable sets) and which in parts of type 1 (cliques), we can find the classes of the equivalence in which two vertices that are both in parts of type 0 are equivalent if they have the same open neighbourhood, and two vertices that are both in parts of type 1 are equivalent if they have the same closed neighbourhood. Having these equivalence classes in hand, we can easily check if a list partition exists. (Recall that the size k of the matrix M is fixed.) □

The next result shows that the classification of M-partition problems as quasipolynomial or NP-complete might extend to an arbitrary size k. Let

M' be obtained from the matrix M by replacing all 1's by 0's. Now M' is a $(0,*)$-matrix, and hence corresponds to the adjacency matrix of a graph H (when $*$'s are replaced by 1's); the list M'-partition problem is precisely the list H-colouring problem.

Theorem 7.2 *Each list M-partition problem can be reduced to $n^{c\log n}$ instances of the list H-colouring problem (each with possible additional restrictions on the allowed lists).*

PROOF. Suppose $AC = 0, BD = 1$ in M. According to Corollary 4.5, we can reduce the list M-partition problem for an input graph with arbitrary lists L to $n^{\log n}$ (or fewer) instances each of which contains no list containing both A and B or no list containing both C and D. Each of these can in turn be reduced to further $n^{\log n}$ (or fewer) instances in which there is no list containing both A' and B' or no list containing both C' and D', for some other pair $A'C' = 0, B'D' = 1$ in M. Since M is fixed, we obtain at most $(n^{\log n})^c = n^{c\log n}$ instances in which no pair of vertices v, v' has lists $L(v) \supseteq \{X,Y\}$ and $L(v') \supseteq \{Z,W\}$ where X,Y,Z,W are any parts such that $XZ = 0, YW = 1$ in M.

Consider one such instance, a graph G with lists L. For any pair of vertices v, v' of G the pairs $X \in L(v), Y \in L(v')$ all have $XY = 0,*$, or all have $XY = *,1$. For vertices v, v' for which they are $0,*$, leave the edge or nonedge between v and v' unchanged. For those v, v' for which they are $*,1$, revert an edge between v and v' to a nonedge, and a nonedge to an edge. It is easy to see that the original graph G has a list M-partition if and only if the modified graph has a list M'-partition, i.e., a list H-homomorphism. Note that the homomorphism problems inherit some restrictions on which pairs of parts (or parts) can appear in the lists. $\square$

Corollary 7.3 *If, as conjectured by Feder and Vardi [23], all list H-colouring problems are polynomial or NP-complete, then all list M-partition problems are quasipolynomial or NP-complete.*

PROOF. If the above instances of list H-colouring problems, with list restrictions, can be solved in polynomial time, we obtain a quasipolynomial algorithm for list M-partition. If a list H-colouring problem (with list restrictions) is NP-complete, then the original problem is also NP-complete. $\square$

We are grateful to Donald Knuth and Jan Kratochvil for valuable suggestions.

References

[1] N. Alon and M. Tarsi, Colorings and orientations of graphs, *Combinatorica* 12 (1992) 125–134.

[2] B. Aspvall, F. Plass and R.E. Tarjan, A linear time algorithm for testing the truth of certain quantified Boolean formulas, *Information Processing Letters* 8 (1979) 121–123.

[3] R.E. Bixby, A composition for perfect graphs, *Annals of Discrete Mathematics* 21 (1984) 221–224.

[4] A. Brandstadt, Partitions of graphs into one or two stable sets and cliques, *Discrete Mathematics* 152 (1996) 47–54.

[5] A. Brandstadt, Corrigendum, *Discrete Mathematics* 186 (1998) 295.

[6] A. Brandstadt, Partitions of graphs into one or two stable sets and cliques, Informatik-Berichte Nr. 105, 1/1991, FernUniversitat Hagen Technical Report 1991.

[7] A. Brandstadt, V.B. Le, T. Szymczak, The complexity of some problems related to graph 3-colourability, *Discrete Applied Mathematics* 89 (1998) 59-73.

[8] M. Burlet and J. Fonlupt, A polynomial algorithm to recognize a Meyniel graph, *Annals of Discrete Mathematics* 21 (1984) 225-252.

[9] V. Chvátal, Star-cutsets and perfect graphs, *Journal of Combinatorial Theory, Series B* 39 (1985) 189–199.

[10] V. Chvátal and N. Sbihi, Bull-free Berge graphs are perfect, *Graphs and Combinatorics* 3 (1987) 127–139.

[11] G. Conruéjols and W. H. Cunningham, Compositions for perfect graphs, *Discrete Math.* 55 (1985) 245–254.

[12] G. Conruéjols and B. Reed, Complete multi-partite cutsets in minimal imperfect graphs, *Journal of Combinatorial Theory, Series B* 59 (1993) 191–198.

[13] A. Cournier and M. Habib, A new linear algorithm for modular decomposition, Sophie Tison, ed., *CAAP'94: International Colloquium, Lectures Notes in Computer Science*, Edinburgh, UK (1994), pp. 68–82.

[14] W. H. Cunningham, A combinatorial decomposition theory, Ph.D. Thesis, University of Waterloo, Ontario, (Canada) 1973.

[15] W. H. Cunningham, Decomposition of directed graphs, *SIAM Journal of Algebraic Discrete Methods* 3 (1982) 214–228.

[16] W. H. Cunningham and J.Edmonds, A combinatorial decomposition theory, *Canad. J. Math.* 32 (1980) 734–765.

[17] H. Everett, S. Klein, and B. Reed, An algorithm for finding clique-cross partitions, *Congressus Numerantium* 135 (1998) 171–177.

[18] H. Everett, S. Klein, and B. Reed, An algorithm for finding homogeneous pairs, *Discrete Applied Mathematics*, 72 (1977) 209–218.

[19] T. Feder and P. Hell, List homomorphisms to reflexive graphs, *Journal of Combinatorial Theory, Series B*, 72 (1998) 236–250.

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

[21] T. Feder, P. Hell, and J. Huang, List homomorphisms to general graphs, Manuscript, 2000.

[22] T. Feder, P. Hell, S. Klein, and R. Motwani, Complexity of graph partition problems, *Proc. 31st Annual ACM Symposium on Theory of Computing*, Atlanta GA, May 1999, pp. 464-472.

[23] T. Feder and M. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction: a study through datalog and group theory, *SIAM Journal on Computing* 28 (1999) 57–104.

[24] C. M. H. de Figueiredo and S. Klein, The NP-completeness of multipartite cutset testing, *Congressus Numerantium* 119 (1996) 217–222.

[25] C. M. H. de Figueiredo, S. Klein, Y. Kohayakawa and B. Reed, Finding skew partitions efficiently, Technical Report **ES-503/99**, COPPE/UFRJ, Brazil, to appear in LATIN 2000.

[26] H. Fleischner and M. Stiebitz, A solution of a colouring problem of P. Erdos, *Discrete Mathematics* 101 (1992) 39–48.

[27] M.R. Garey and D.S. Johnson, **Computers and Intractability**, W.H. Freeman and Company, San Francisco, 1979.

[28] M.C. Golumbic, **Algorithmic Graph Theory and Perfect Graphs**, Academic Press, New York, 1980.

[29] W. Gutjahr, E. Welzl and G. Woeginger, Polynomial graph-colorings, *Discrete Applied Mathematics* 35 (1992) 29–45.

[30] P. Hell and J. Nešetřil, On the complexity of H-colouring, *Journal of Combinatorial Theory, Series B* 48 (1990) 92–110.

[31] C.T. Hoang and V.B. Le, Recognizing perfect 2-split graphs, *SIAM J. on Discrete Math.* 13 (2000) 48 - 55.

[32] L. Lovász, Communication complexity: a survey, In: **Paths, flows, and VLSI-Layout**, (B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, eds.), Springer, pp. 235–265.

[33] G. MacGillivray and M.L. Yu, Generalized Partitions of Graphs, *Discrete Applied Mathematics* 91 (1999) 143–153.

[34] R.M. McConnel and J.P. Spinrad, Linear-time modular decomposition and efficient transitive orientation of comparability graphs, *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 1994, pp. 536–545.

[35] R.E. Tarjan, Decomposition by clique separators, *Discrete Mathematics* 55 (1985) 221–232.

[36] A. Tucker, Coloring graphs with stable sets, *Journal of Combinatorial Theory, Series B* 34 (1983) 258–267.

[37] N. Vikas, Computational complexity of graph compaction, *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, SODA 1999.

[38] D.G. West, **Introduction to Graph Theory**, Prentice Hall, 1996.

[39] S. Whitesides, An algorithm for finding clique cutsets, *Information Processing Letters* 12 (1981) 31–32.

[40] S. Whitesides, A method for solving certain graph recognition and optimization problems, with applications to perfect graphs, *Annals of Discrete Mathematics* 21 (1984) 281–297.