LIST MATRIX PARTITIONS OF SPECIAL GRAPHS

by

Payam Valadkhan

M.Sc., Simon Fraser University, 2009

a Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

APPROVAL

Name:	Payam Valadkhan
Degree:	Doctor of Philosophy
Title of Thesis:	List matrix partitions of special graphs

Examining Committee: Dr. Ramesh Krishnamurti
Chair

Dr. Pavol Hell, Senior Supervisor

Dr. Gabor Tardos, Co-Supervisor

Dr. Andrei A. Bulatov, Supervisor

Dr. Binay Bhattacharya, SFU Examiner

Dr. Kathie Cameron, External Examiner,
Professor, Mathematics,
Wilfrid Laurier University

Date Approved: 3 June 2013

Abstract

Let M be a symmetric $m \times m$ matrix with entries from the set $\{0, 1, *\}$. The M-partition problem asks whether the vertices of a given graph G can be partitioned into m parts $V_0, V_1 \cdots V_{m-1}$ such that any two distinct vertices in (possibly equal) parts V_i and V_j are adjacent if $M(i,j) = 1$, and non-adjacent if $M(i,j) = 0$. This problem generalizes k-coloring and H-coloring problems, as well as many other well-known graph problems. In its list version, which is called the list M-partition problem, a list is assigned to each vertex to restrict its placement into certain parts. An open problem, called the dichotomy problem, asks whether each (list) M-partition problem is polynomial or NP-complete. The difficulty of this problem led to the study of restrictions on the input graphs. A secondary goal was to identify the well-known graph classes for which all (list) M-partition problems are polynomial. Several graph classes including perfect graphs, chordal graphs, etc. have been studied so far. In this thesis we continue this line of research, focusing mainly on the list version. We identify certain graph classes defined in terms of geometric configurations, and we prove that for these classes all list M-partition problems are polynomial. These classes include such well-known classes as interval and circular arc graphs. We also consider other standard graphs classes including some generalizations of the aforementioned classes, line graphs and their extensions to quasi-line graphs and claw-free graphs, and some special cases of H-free graphs. For these classes we provide a positive answer to the dichotomy problem for certain kinds of matrices M.

Keywords: list M-partition problem; graph partitions; homomorphism; special graph classes; perfect graphs; interval graphs; line graphs; H-free graphs; dichotomy

Subject Terms: Graph Theory; Graph Coloring; Graph Partitions; Graph Algorithms

To my parents, Ali and Najmeh.

Peace

Acknowledgments

I express my gratitude for the help of my supervisors, Pavol Hell and Gabor Tardos, who provided many insightful comments and much practical advice on this research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The M-partition problem was first formulated and introduced by Feder et al. in [129]. The main motivation was the fact that many important combinatorial problems involve partitioning the vertices of a given graph into parts satisfying certain constraints on adjacencies between or among vertices in different parts [48, 129]. For example, consider the classic problem of 3-coloring the vertices of a given graph G. This is equivalent to finding a partition of the set $V(G)$, the vertex set of G, into three parts V_0, V_1, V_2 such that any two distinct vertices in the same part are non-adjacent. As another example, consider the well-known clique cutset problem ([267]), which is equivalent to asking whether $V(G)$ can be partitioned into three non-empty parts V_0, V_1, V_2 such that any two distinct vertices in V_0 are adjacent and any vertex in V_1 is non-adjacent to any vertex in V_2.

To formulate a common generalization for this kind of problem, Feder et al. [129] introduced a general framework as follows: let M be a symmetric $m \times m$ matrix with entries from the set $\{0, 1, *\}$ (the rows and columns are indexed starting at 0), an *M-partition* of a graph G is a partition of the set $V(G)$ into parts $V_0, V_1 \cdots V_{m-1}$ such that any two distinct vertices in (possibly equal) parts V_i and V_j are adjacent if $M(i, j) = 1$, and non-adjacent if $M(i, j) = 0$, while $M(i, j) = *$ signifies no restriction. Note that the case of $i = j$ implies that the part V_i is a clique when $M(i, i) = 1$, a stable set when $M(i, i) = 0$ and an arbitrary set when $M(i, i) = *$. The *M-partition problem* asks whether the input graph G has an M-partition. Based on these definitions, the 3-coloring and the clique cutset problems can be formulated as M-partition problems using the matrices shown in Figure

1.1(a) and (b), respectively. Note that for the clique cutset problem, the corresponding M-partition problem ignores the condition that all parts must be non-empty. We will get back to this point later.

$$\begin{bmatrix} 0 & * & * \\ * & 0 & * \\ * & * & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & * & * \\ * & * & 0 \\ * & 0 & * \end{bmatrix}$$

(a) 3-coloring (b) clique cutset

Figure 1.1: The matrix M for some special problems.

The M-partition problem generalizes the k-coloring and the H-coloring problems, as well as many other well-known graph problems if we ignore possible side conditions related to the maximum or minimum number of vertices in each part (e.g., the non-emptiness of each part in the clique cutset problem). These problems include the split graphs recognition problem ([164]), the (a, b)-graphs recognition problem ([28]), the clique cutset problem ([267, 277]), the stable cutset problem ([266]), the skew cutset problem ([56]), the clique-cross problem ([116]), the homogeneous set problem ([236]) and many types of problems involving joins ([53]). In Chapter 2 we will define many of these problems along with their background and applications. Thus the M-partition problem provides a unifying framework for many important graph problems.

To capture the side conditions ignored by the M-partition problem (e.g., certain parts have to be non-empty, or have a fixed number of vertices), the *list version* of the M-partition problem, called the *list M-partition problem*, is introduced ([129]) as follows: given a graph G with a list $L(v) \subseteq \{0, 1, \cdots m-1\}$ assigned to each vertex $v \in V(G)$, the list M-partition problem asks whether there is an M-partition of G such that $v \in V_i$ only if $i \in L(v)$, for all $v \in V(G)$. In Section 2.6 we explain in detail how we can use the list version to model the typical side conditions (e.g., each part has to be non-empty). We use the term *non-list* or *basic version* to emphasize that an M-partition problem is not in the list version.

One of the main objectives in studying the M-partition problem and its list version is to identify the (list) M-partition problems which can be solved in polynomial time [180]. It can be easily seen that all (list) M-partition problems belong to the class NP (i.e., decision problems solvable in non-deterministic polynomial time). We recall that a problem in the class NP is called NP-complete if any NP problem can be reduced to it in polynomial time. The question of whether an NP-complete problem can be solved in polynomial time is a big challenging open problem in theoretical computer science, with many researchers believing that NP-complete problems cannot be solved in polynomial time. So proving a problem to be NP-complete can be taken as a serious suggestion that there might not be a polynomial-time algorithm to solve this problem. Thus, moving along the main objective, it is relevant to ask whether all (list) M-partition problems can be classified into those which can be solved in polynomial time and those which are NP-complete. This problem is known as the *dichotomy problem* for (list) M-partition problems. It is worthwhile to mention that according to [219], unless $P = NP$, there are problems in NP that are neither polynomial nor NP-complete. Affirmative answers to the dichotomy problem for (list) M-partition problems are known when the matrix M is restricted to certain classes. The most notable example is the class of 1-free matrices (i.e., matrices which have no entry with value 1), for which the dichotomy problems for M-partition problems and for list M-partition problems are both proved to be affirmative [118, 187]. Results of this type are called *complete dichotomy*, as they declare each (list) M-partition problem (for all matrices M in the given class of matrices) to be polynomial or NP-complete. This is in contrast to *partial dichotomy*, where the classification into polynomial and NP-complete cases does not cover all possible choices for the matrix M. These dichotomy results, along with some links to the Feder-Vardi dichotomy conjecture [138, 120], motivated Feder et al. in [129] to ask whether the answer to the dichotomy problem for (list) M-partition problems is affirmative. This question is still open, and marks a direction in studying the M-partition problem. Another approach to the main objective (i.e., identifying the polynomial-time (list) M-partition problems) is through characterizing the *M-partitionable* graphs, i.e., those graphs which have at least one M-partition. Formally, a *minimal obstruction*, or *MO*, for a matrix M is a graph which is not M-partitionable, while any of its proper induced subgraphs is M-partitionable. It is a simple exercise to see that the M-partition problem is polynomial when M has finitely many MOs. Thus, the *characterization problem* aims at identifying all matrices M which have finitely many MOs. Such identification, even when M is restricted to a certain class, is called a *complete characterization.* In Section 2.7 we give a more detailed account of the underlying motivations and the current results for the dichotomy and the characterization problems. For now, we offer a summary of the current main results for these two problems in Table 1.1.

To understand the results in Table 1.1 and other tables, we need to define some key concepts. Define the *block A* (*block B*, respectively) of the matrix M as the sub-matrix (of

M) consisting of the rows and the columns with 0 (1, respectively) on the main diagonal. Define the *block* C of M as the sub-matrix consisting of the intersection of the rows with 0 on the main diagonal and the columns with 1 on the main diagonal. Refer to Figure 1.2 for a graphical display of these blocks. A matrix is said to be 01-diagonal if its main diagonal consists of only 0 and 1 entries (i.e., no * is allowed on the main diagonal).

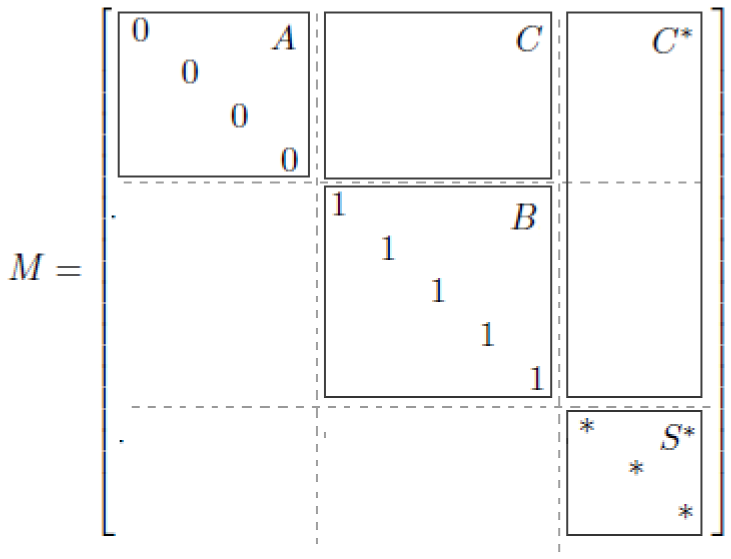

Figure 1.2: The blocks A, B, C, S^* and C^* for a sample matrix M.

From Table 1.1 we can deduce that the dichotomy and characterization problems are both difficult to settle in general. This is evident from the number of restrictions that the author had to impose on the matrix M to make the problems tractable. It is also proved in [121] that proving a complete dichotomy for the M-partition problem will imply the Feder-Vardi dichotomy conjecture (introduced in [138]), which is a prominent open question (see [138, 44, 120, 121]). This link may serve as an additional evidence for the difficulty of answering the dichotomy problem in general. We do not have such evidence for the difficulty of the characterization problem though [180].

As a way to mitigate these difficulties, the dichotomy and the characterization problems were studied under the assumption that the input graph is restricted to certain graph classes such as the class of perfect graphs [119, 121], and its well-known subclasses including chordal graphs [130, 131, 242, 133], cographs [123] and recently, split graphs [134]. The main

#	**Matrix M**	**Characterization Result**	**Complexity Result**	**Ref. Sec.**
1	any	partial characterization for several classes of matrix M [137]	complete quasi-dichotomy (list version), i.e., each problem is quasi-polynomial (of order $2^{O((\log n)^c)}$) or NP-complete [120]	2.7
2	0-free or 1-free	complete characterization [180]	complete dichotomy, for both basic [187] and list [125] versions	2.2,2.7
3	*-free	finitely many MOs [122]	polynomial (list version) [129]	2.7
4	01-diagonal, A and B are both *-free, C is *-free or all *	complete characterization [137]	polynomial (list version) [129]	2.7
5	01-diagonal, A and B are both *-free and do not contain indices $i < i' < i''$ with $M(i,i)$=$M(i,i')$= $M(i',i'')$=$M(i'',i)$	N/A	polynomial (basic version) [137]	2.7
6	size ≤ 4	complete characterization [137]	complete dichotomy (basic and list versions) [129, 48, 78]	2.7
7	size ≤ 5	complete characterization [137]	N/A	2.7

Table 1.1: The main complexity and characterization results in the literature. Recall that by default M is a symmetric matrix with entries from the set $\{0, 1, *\}$. Each result imposes some extra restrictions on M as shown in the table.

motivation for choosing these graph classes was the fact that in several special cases of the M-partition problem, mainly k-coloring and H-coloring problems, the affirmative answer to the dichotomy problem was known when the input graph was restricted to these graph classes. Thus the researchers hoped that the dichotomy and the characterization problems may also be tractable for these graph classes. In Section 2.8 we introduce these graph classes and give a background of M-partition problems restricted to them. For now we just give a summary of the main results in this direction in Table 1.2.

As it is clear from Table 1.2, even restriction to graph classes such as chordal graphs (for which many combinatorial problems are polynomial) did not help finding a complete dichotomy as it was hoped. Due to this failure, studying the dichotomy and characterization problems for some other smaller subclasses are proposed as future work [180]. This is exactly the focus of this thesis, namely finding dichotomy results for special graph classes.

The class of interval graphs (defined in Section 3.1.1) is a subclass of chordal graphs. Studying the M-partition problem for interval graphs (particularly the dichotomy problem) is posed in the preliminary version of [180] as a relevant open problem, with the hope that this restriction could further simplify the dichotomy problem in the case of chordal graphs as evident from Table 1.2, rows 2-10. (This open problem was removed in the final version of [180] as it was settled in this thesis.) We study the list M-partition problem for interval graphs in Chapter 5, and then shift our focus to other related graph classes which are studied in Chapter 5 and other subsequent chapters. Table 1.3 shows a summary of our main dichotomy results for each of these graph classes. Note that in contrast to the current results which mainly consider 01-diagonal matrices (see Tables 1.1 and 1.2), we also consider the cases in which * is allowed on the main diagonal. For this reason, following the model of defining blocks A, B, C, we introduce two additional blocks, namely S^* and C^* (see Figure 1.2), where S^* is the sub-matrix consisting of the rows and the columns with * on the main diagonal (analogous to the blocks A and B), and C^* is the sub-matrix consisting of the intersection of the rows with 0 or 1 on the main diagonal and the columns with * on the main diagonal (analogous to the block C). Our results also include some more technical statements not included in the table. (Refer to the individual chapters for more details.)

In the rest of this chapter, after introducing some basic notation and definitions in Section 1.2 we introduce a very relevant technique in Section 1.3, which helps proving many graph theoretical problems, including the M-partition problem and its list version, to be polynomial when restricted to graph classes having a certain property. In Chapter 2 we give

Graph Class	#	**Matrix** M	**Result(s)**
Perfect Graphs [121]	1	01-diagonal, none of the blocks A,B or C contains both a * entry and an off-diagonal non-* entry	finitely many MOs (⇒ polynomial complexity for the basic version)
Chordal Graphs [131, 133]	2	0-diagonal	linear complexity (list version)
	3	1-diagonal	polynomial complexity (list version)
	4	01-diagonal, A or B is *-free, C is 1-free or 0-free	complete dichotomy (list version)
	5	01-diagonal, A or B is *-free	complete quasi-dichotomy (list version)
	6	01-diagonal, C contains a set of rows and columns without * that together cover all the entries of C different from *	polynomial complexity (list version)
	7	many examples	NP-complete (basic version)
	8	01-diagonal, all off-diagonal entries are *	there is only one MO (⇒ polynomial complexity for the basic version)
	9	size ≤ 4	polynomial complexity (basic version)
	10	size ≤ 3	complete characterization
Cographs [123]	11	any	finitely many MOs, polynomial complexity (list version)
Split Graphs [134]	12	any	finitely many MOs (⇒ polynomial complexity for the basic version), but there are NP-complete matrices M for the list version

Table 1.2: The main complexity and characterization results for special graph classes in the literature.

Graph Class	**Matrix M**	**Complexity Result (list version)**	**Ref. (Theorem or Corollary)**
Interval Graphs	any	polynomial	5.2.4
Circular Arc Graphs	any	polynomial	5.3.3
Permutation Graphs	any	polynomial	5.2.4
Circular Permutation Graphs	any	polynomial	5.3.3
Interval & Interval Containment Bigraphs	any	polynomial	5.2.4
Comparability Graphs	01-diagonal, A is 1-free	complete dichotomy	5.4.6
Circle Graphs	some examples	NP-complete	5.5.1
Line Graphs	01-diagonal	complete dichotomy	6.1.1
	*-diagonal, 1-free	complete dichotomy	6.1.2
	B,S^* and C^* are 0-free, 1-free and all *, resp.	complete dichotomy	6.1.3
	B has size $\geq m-2$	polynomial	6.1.4
	some new examples	NP-complete	6.1.5
Quasi-Line Graphs	B is 0-free, C is 0-free or *-free	complete dichotomy	6.2.4
	1-diagonal, 0-free	polynomial (basic version)	6.2.5
Claw-free Graphs	B is 0-free, C is 0-free or *-free	complete dichotomy	6.3.6
H-free Graphs	0- or 1-diagonal (based on some condition)	complete dichotomy (for all but a few exceptional graphs H)	7.1.1 and 7.1.2
	some examples, except when H is an induced subgraph of P_4	NP-complete	7.1.4
P_5-free Graphs	01-diagonal, B is 0-free, C is *-free or all *	complete dichotomy	7.2.4
$\{P_5, \overline{P_5}\}$-free Graphs	01-diagonal, C is *-free or all *	complete dichotomy	7.2.5
Bull-free Graphs	A is 1-free, B is 0-free	complete dichotomy	7.3.2

Table 1.3: A summary of the main results in this thesis.

a history of the M-partition problem and its related directions and results. In Chapter 3 we introduce the graph classes shown in Table 1.3, and explain why studying the M-partition problem for these graph classes is relevant. In Chapter 4 we introduce some technical tools needed throughout this thesis. The remaining chapters are dedicated to proofs of our main results as outlined in Table 1.3. (Refer to this table for more details on the contents of each chapter.)

Before closing this overview, we would like to justify the restrictions that our results impose on the matrix M (as evident from Table 1.3). In choosing these restrictions, we were guided by the restrictions imposed on M in the existing results as shown in Tables 1.1 and 1.2. The general trends are to restrict certain values for the entries on the main diagonal, or the blocks A, B, C defined above (see Figure 1.2), or the whole matrix. We follow these trends and extend them to the blocks S^* and C^* (see Figure 1.2), as we also consider matrices M with * on the main diagonal.

1.2 Definitions

For an integer $n \geq 0$, denote by $[n]$ the set $\{0, 1, \cdots n-1\}$. Given an arbitrary set X, an *n-tuple* of X is any vector $(x_1, x_2, \cdots x_n)$ with $x_i \in X$, for $i = 1, 2, \cdots n$. A 2-tuple of X is commonly called a *pair* of X. The set of all n-tuples of X is denoted by X^n.

A *digraph* G is a pair (V, E), where V is the set of *vertices* (also called the *vertex set*), and $E \subseteq V^2$ is the set of *arcs*. So each arc $e \in E$ is a pair (u, v) of vertices, and we call the vertices u and v the *endpoints* of the arc e. If $u = v$ then e is called a *loop* at the vertex u. For the sake of clarity, in this thesis we use the notations $V(G)$ and $E(G)$ instead of V and E to emphasize that they belong to the description of the digraph G. In the *graphical representation* of G, we assign a point in the plane to each vertex $v \in V$, and for any arc $(u, v) \in E$, we draw an arc (an arrow) directed from the point u to the point v. Based on this view, we write $u \rightarrow v$ to denote that there is an arc directed from u to v (in G). A $|V(G)| \times |V(G)|$ matrix A_G, known as the *adjacency matrix of* G, is defined as follows: for each vertex $v \in V(G)$, there is exactly one row r_v and one column c_v in A_G corresponding to this vertex, and we define $A_G(r_u, c_v) = 1$ if $(u, v) \in E(G)$, otherwise $A_G(r_u, c_v) = 0$ (for any two not necessarily distinct vertices $u, v \in V(G)$).

An *undirected pair* from a set X consists of two (not necessarily distinct) elements $x, y \in X$, and is denoted by xy. Note that two undirected pairs xy and $x'y'$ are identical whenever the sets $\{x, y\}$ and $\{x', y'\}$ are identical. A *graph* is a pair (V, E), where V is the set of vertices (also called the *vertex set*), and E is a set consisting of undirected pairs of $V(G)$. Each element $uv \in E$ is called an *edge* between u and v (or equivalently, between v and u). In the graphic representation of G, we assign a point in the plane to each vertex $v \in V$, and for any edge $uv \in E$, we draw a segment connecting the points corresponding to the vertices u and v. The adjacency matrix of a graph G is a $|V(G)| \times |V(G)|$ matrix A_G such that for each vertex $v \in V(G)$, there is exactly one row r_v and one column c_v in A_G corresponding to this vertex, and $A_G(r_u, c_v) = 1$ if $uv \in E(G)$, otherwise $A_G(r_u, c_v) = 0$ (for any two not necessarily distinct vertices $u, v \in V(G)$). In this thesis, we assume that the set $V(G)$ (both for graphs and digraphs) is always finite, and all the graphs and digraphs, as the inputs of our algorithms, are encoded using the adjacency matrix.

Let G be a graph. A graph G' is called a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. If $V(G') \neq V(G)$ or $E(G') \neq E(G)$ then G' is called a *proper* subgraph. Additionally, we say that the subgraph G' is *induced* if it has all the possible edges, namely $E(G') =$

$E(G) \cap V(G')^2$. For a subset $D \subseteq V(G)$, the *induced subgraph of* D, denoted by $G[D]$, is the induced subgraph G' in which $V(G') = D$. By *removing a set* D (from G) we mean forming the induced subgraph $G[V(G) - D]$.

Two vertices u and v in a graph G are called *adjacent, or neighbors* whenever $uv \in E(G)$, otherwise they are called *non-adjacent*, or *independent*. Given two non-empty subsets $X, Y \subseteq V(G)$, we say that X is *fully adjacent* (*fully non-adjacent*, respectively) to Y if any vertex in X is adjacent (non-adjacent, respectively) to any vertex in Y. For any vertex u, denote by $N(v)$ the set of its neighbors, i.e., the set of all vertices in $V(G)$ adjacent to v. Similarly, denote by $\overline{N}(v)$ the set of non-neighbors of v, i.e., the set of all vertices in $V(G)$ non-adjacent to v. Note that for any $v \in V(G)$, either $v \in N(v)$ or $v \in \overline{N}(v)$ depending on whether there is a loop at v or not, respectively. The value $|N(v)|$ is called the *degree* of v (in G). Two sets X and Y are called *disjoint* if $X \cap Y = \emptyset$. Given two graphs G and H with disjoint vertex sets, we define the *disjoint union of* G *and* H, denoted by $G + H$, as a graph with the vertex set $V(G) \cup V(H)$ and the edge set $E(G) \cup E(H)$. Two graphs G and H are called *isomorphic* if there exists a bijective mapping $f : V(G) \to V(H)$ such that $uv \in E(G)$ if and only if $f(u)f(v) \in E(H)$.

A *path* with n vertices is a graph with n vertices $v_0, v_1, \cdots v_{n-1}$ and edges v_iv_{i+1}, for $i = 0, 1, \cdots n-2$. If we add the edge $v_{n-1}v_0$ as well then it is called a *cycle* with n vertices. We may use the term "of order n" instead of "with n vertices". A path (cycle, respectively) in a graph G is a subgraph G' of G which is isomorphic to a path (cycle, respectively). Additionally, if G' is an induced subgraph of G then G' is called a *chordless* path (cycle, respectively). To be more specific, suppose a path P in G consists of vertices $v_0, v_1, \cdots v_{t-1}$ (and the edges v_iv_{i+1}, for $i = 0, 1, \cdots t-2$). Then we say that P is a path of order t *from* v_0 *to* v_{t-1}. A graph G is called *connected* if for any two distinct vertices $u, v \in V(G)$, there exists a path in G from u to v. A graph is called *disconnected* if it is not connected. It is well-known in graph theory that any graph G is a disjoint union of several connected graphs $C_1, C_2, \cdots C_r$ known as the *connected components of* G (see [25]). A graph which has no cycle is called a *forest*. Any connected forest is called a *tree*. The smallest value t for which a graph G has a chordless cycle of order t is called the *girth* of G. It is defined to be infinite if G has no cycle.

A graph G is called a *complete graph* if any two distinct vertices in it are adjacent. We denote by K_t the complete graph with t vertices and no loops. A set $D \subseteq V(G)$ is called a *clique* if any two distinct vertices in D are adjacent. A set $D \in V(G)$ is called *stable* *or independent* if any two distinct vertices in D are non-adjacent. The order of the largest clique in G is called the *clique number of* G and is denoted by $\omega(G)$. The order of the largest stable set in G is called the *stability number of* G and is denoted by $\alpha(G)$. The problems of finding the values of $\omega(G)$ and $\alpha(G)$ are called the *maximum clique problem* and the *maximum stable set problem*, respectively.

The *complement of* G, denoted by $\overline{G}$, is the graph with same vertices as G in which two vertices u and v are adjacent if and only if they are not adjacent in G. Note that this definition includes the case of $u = v$ (namely the loops). We use the prefix *co-* before the graph name to denote the complement. For example, co-K_3 represents a graph which is the complement of K_3 (which is a graph with three pairwise non-adjacent vertices with a loop at each vertex), or co-bipartite graphs are the complements of bipartite graphs.

A *coloring* of graph G is a mapping $f : V(G) \to \mathbb{N} \cup \{0\}$, where $\mathbb{N}$ is the set all natural numbers (i.e., $\{1, 2, \cdots\}$). Each value in the range of f is called a *color*. A coloring f (of G) is called *proper* if for any two distinct adjacent vertices $u, v \in V(G)$ we have $f(u) \neq f(v)$. A proper coloring f whose number of colors (i.e., the size of the range of f) is bounded by an integer $k > 0$ is called a *k-coloring* of G. Note that any k-coloring of G corresponds to a partition of $V(G)$ into k stable sets (possibly empty) $V_0, V_1, \cdots V_{k-1}$, where V_i is defined to be the set of vertices u for which $f(u)$ is the i-the color in the range of f. (Recall that our definition of stable set allows loops inside a stable set.) If such partition exists, we say that G *admits a k-coloring* or is *k-colorable*. The minimum value of k for which G is k-colorable is called the *chromatic number* of G and is denoted by $\chi(G)$. We call the problem of finding this number the *minimum coloring* problem. A graph G is called *bipartite* if it is 2-colorable, i.e., $V(G)$ can be partitioned into two stable sets X and Y (possibly empty). If X and Y are fully adjacent then G is called a *complete bipartite* graph.

As a counterpoint of proper coloring, a *clique-covering* of a graph G is a coloring f such that for any two non-adjacent vertices $u, v \in V(G)$ we have $f(u) \neq f(v)$. A clique-covering f whose number of colors is bounded by an integer $l > 0$ is called an *l-clique-covering* of G. As in the case of k-coloring, we can show that any l-clique-covering of G corresps to a partition of $V(G)$ into l cliques (possibly empty) $V_0, V_1, \cdots V_{l-1}$. Note that this partition is indeed covering $V(G)$ with cliques, which explains why it is called a clique-covering. The minimum value of l for which G has an l-clique-covering is called the *clique-covering number* of G and is denoted by $\theta(G)$. Note that any l-clique-covering for G is an l-coloring for $\overline{G}$, and thus we have the relation $\theta(G) = \chi(\overline{G})$.

A *graph class* is a set of graphs (e.g., the class of bipartite graphs). We typically use the capital letters with tilde to denote a graph class (e.g., $\widetilde{G}$), while we use capital letters to denote a graph (e.g., G). We say that a graph class $\widetilde{G'}$ is a *subclass* of a graph class $\widetilde{G}$ if $\widetilde{G'} \subseteq \widetilde{G}$. In this case, we say that $\widetilde{G}$ is a *superclass* or an *extension* of $\widetilde{G'}$. A graph class $\widetilde{G}$ is called *hereditary* if it includes any induced subgraph of any of its members. The *complement of* $\widetilde{G}$ is defined as a graph class whose members are the complements of the graphs in $\widetilde{G}$, e.g., the class of co-bipartite graphs is the complement of the class of bipartite graphs.

In this thesis, we assume that all matrices are symmetric with their entries in the set $\{0, 1, *\}$, unless explicitly stated otherwise. Also, we always denote by m the size of the matrix M, which should be clear from the context. We index the rows and columns of M with integers starting at 0. Denote by $M(i, j)$ the entry of M in the i-th row and the j-column $(0 \leq i, j < m)$. By the *main diagonal* of the matrix M we mean the set of entries $M(i, i)$, for $i = 0, 1, \cdots m - 1$, and by the *off-diagonal* entries we mean the set of all other entries (i.e., those not on the main diagonal). Given a subset $X \subseteq \{0, 1, *\}$, we say that M is X*-free* if it does not contain any entry from X. We also say it is X*-diagonal* if all entries on the main diagonal are from X. To simplify the notation, in writing X-free and X-diagonal, we may drop the brackets and commas related to the set X, e.g., 1-free stands for $\{1\}$-free, and 1*-diagonal stands for $\{1, *\}$-diagonal. We say that M *contains* another $m' \times m'$ matrix M' if M' is a principal sub-matrix of M, i.e., there exists a one to one mapping $f : [m'] \to [m]$ such that $M'(i, j) = M(f(i), f(j))$ for all $(i, j) \in [m']^2$. We construct the *complement of the matrix* M, denoted by $\overline{M}$, by turning 1 entries to 0 and vice versa (leaving the * entries intact). Recall that earlier in Section 1.1, we defined the blocks A, B, C, S^* and C^* of the matrix M (see Figure 1.2). Note that the matrices A, B and S^* are 0-diagonal, 1-diagonal and *-diagonal, respectively. A matrix M is said to be the *corresponding matrix of a graph* G if M is obtained from the adjacency matrix of G by turning all 1 entries to *. In this case, G is said to be the *corresponding graph* of the matrix M. Note that such matrix M is a 1-free matrix, and if G is without loops then M is 0-diagonal as well.

For a graph class $\widetilde{G}$, the (list) M-partition problem *for*, or *restricted to*, $\widetilde{G}$ is defined as the (list) M-partition problem when the input graph is restricted to the class $\widetilde{G}$. The same definition also applies to any other graph problem, e.g., the maximum stable set problem for $\widetilde{G}$.

Given a decision problem which takes a graph as input, by saying that this problem is *polynomial* (of order $O(n^t)$, for some constant $t > 0$) we mean that this problem can be solved (using some algorithm) in polynomial time (of order $O(n^t)$, where n denotes the number of vertices of the input graph). We use the term *pattern* as synonym to matrix. We say that a matrix M is a *polynomial* (*NP-complete*, respectively) pattern, or matrix, for a graph class $\widetilde{G}$ if the decision problem of the list M-partition problem for $\widetilde{G}$ (assuming M to be fixed) is polynomial (NP-complete, respectively). *Quasi-polynomial* is defined as the running time complexity of order $2^{O((\log n)^c)}$, for some fixed constant $c > 0$.

Given a finite set D and an integer $n \geq 1$, an *n-ary relation* R over D is a set of n-tuples of D (i.e., $R \subseteq D^n$). The terms *binary* and *unary* are conventionally used for 2-ary and 1-ary, respectively. We write $R(d_1, d_2, \cdots, d_n)$ to mean that $(d_1, d_2, \cdots, d_n) \in R$. A *vocabulary* τ is a (finite or infinite) set of pairs (R, l) where R is a relation symbol and l is a positive integer. A *structure* H over the vocabulary τ, or in short a τ-structure, consists of a finite set D^H, called the *domain* of H, and a collection of l-ary relations R^H over D^H, corresponding to each pair $(R, l) \in \tau$.

1.3 Clique-Width and Courcelle's Theorem

In this section we introduce an existing technique to develop efficient algorithms to solve a group of graph problems, including (list) M-partition problems, when the input graph is restricted to special graph classes. Note that this technique is very relevant as this thesis is dedicated to studying the (list) M-partition problem for special graph classes (see Table 1.3). The fact that this technique applies to (list) M-partition problems is based on a simple argument. Since this argument does not use any of the techniques that we introduce in this thesis, we include it here in this section.

The concept of *clique-width* was first introduced by Courcelle et al. in [67]. Let us define this concept first. A *labeled graph* is a graph in which any vertex v has an integer label $l(v) > 0$ assigned to it. A labeled graph which uses at most p different labels is called a *p-graph*. We treat any (unlabeled) graph as a 1-graph by assigning label 1 to each of its vertices. Similarly, any 1-graph is treated as a graph by removing all its labels. For this reason, we use the terms graph and 1-graph interchangeably throughout this section. We define the following operations for the labeled graphs:

1. $i(v)$: creating a new labeled graph with only one vertex v which has label i,

2. $G \oplus H$: the disjoint union of two labeled graphs G and H with disjoint vertex sets, which is defined as the labeled graph with the vertex set $V(G) \cup V(H)$ (keeping the labels intact) and the edge set $E(G) \cup E(H)$,

3. $\eta_{i,j}(G)$, where $i \neq j$: drawing an edge between each vertex labeled i and each vertex labeled j in a labeled graph G,

4. $\rho_{i \to j}(G)$: replacing all the labels i with the label j in a labeled graph G.

Any algebraic expression which is made solely by the above operations and uses at most $k > 0$ different labels is called a *k-expression.* Note that any k-expression constructs a labeled graph (by applying the operations as defined above). An example of a 2-expression constructing a complete graph on three vertices v_0, v_1, v_2 (each having label 1) is given below:

$$\rho_{2\to1}(\eta_{1,2}(2(v_0) \oplus \rho_{2\to1}(\eta_{1,2}(2(v_1) \oplus 1(v_2)))))$$

Note that the labels used in a k-expression are not bound to any specific range, the only important thing is that the number of distinct labels should not exceed k.

Given a labeled graph G, the smallest number k for which G can be constructed by a k-expression is called the *clique-width* of G, and denoted by $cwd(G)$. A similar concept existed before the introduction of clique-width, namely *tree width* (see [69, 255]). In general, no efficient algorithm is known to find a k-expression which constructs a given labeled graph G with clique-width upper-bounded by k, even by assuming k to be a fixed constant. However, such algorithms are known for special cases of G as we will explain later in this section. For a constant $k > 0$, we say that a class $\widetilde{G}$ of labeled graphs is of *bounded clique-width* by k if the clique-width of any labeled graph $G \in \widetilde{G}$ is bounded by (i.e., less than or equal to) k. An underlying graph of a p-graph G is a graph G' obtained by removing all the labels from the vertices of G. It is not hard to see that if $cwd(G) \leq k$ then $cwd(G') \leq k$, and if $cwd(G') \leq k$ then $cwd(G) \leq pk$. Furthermore, a k-expression constructing G' can be obtained from a k-expression constructing G in $O(1)$ time. Conversely, a pk-expression constructing G can be obtained from a k-expression constructing G' in linear time (in terms of the number of operations in the k-expression which constructs G'). These facts lead to the following observation:

Observation 1.3.1. *Given a fixed integer $p > 0$, a class $\widetilde{G}$ of p-graphs G is of bounded clique-width if and only if the class $\widetilde{G}'$ of graphs G', in which G' is the underlying graph* *of G, is of bounded clique-width. Furthermore, there is an efficient algorithm to find the k-expressions constructing the graphs in $\widetilde{G}$ if and only if there is an efficient algorithm to find the k'-expressions constructing the graphs in $\widetilde{G}'$ (k and k' are upper-bounds on the clique-widths of the corresponding classes.)*

Courcelle et al. [68] identified a class of graph problems called $LinEMSOL(\tau_{1,p})$, and proved the following result:

Theorem 1.3.2. *([68]) Let $p > 0$ be a fixed integer and $\widetilde{G}$ a class of p-graphs of bounded clique-width by k. Suppose we are given an order $O(f(n))$ algorithm (for some function f) which, for any labeled graph $G \in \widetilde{G}$, outputs a k-expression constructing G. Then any $LinEMSOL(\tau_{1,p})$ problem can be solved in $O(f(n))$ time when restricted to $\widetilde{G}$.*

The class of $LinEMSOL(\tau_{1,p})$ problems includes many well-known decision and optimization problems such as the k-coloring, the minimum clique-covering, the maximum clique and the maximum stable set problems. (Refer to [68] for a more extensive list.) Many of these problems are known to be NP-hard for general graphs. The main relevance of this result to this thesis is the fact that each M-partition problem can be modeled as a $LinEMSOL(\tau_{1,1})$ problem for graphs, and each list M-partition problem can be modeled as a $LinEMSOL(\tau_{1,2^m})$ problem for 2^m-graphs (These facts are also mentioned with less details in [137].) We will prove these statements shortly in this section. Before doing so, let us state their implication (considering Observation 1.3.1):

Corollary 1.3.3. *Let $\widetilde{G}$ be a graph class with bounded clique-width k. Suppose we are given an order $O(f(n))$ algorithm (for some function f) which, for any graph $G \in \widetilde{G}$, outputs a k-expression constructing G. Then the list M-partition problem can be solved in $O(f(n))$ time when restricted to $\widetilde{G}$.*

We should mention that in addition to $LinEMSOL(\tau_{1,p})$ problems, some other graph problems are also proved to be polynomial when restricted to graph classes with bounded clique-width. We refer to [155] for another set of problems of this kind. These results made the concept of clique-width the subject of much study. Many graph classes with bounded clique-width k have been identified and polynomial-time (and often linear) algorithms have been developed to find k-expressions constructing their graphs (see [69, 29, 166] for many examples). Clearly, applying Corollary 1.3.3 yields polynomial-time algorithms to solve the list M-partition problem restricted to these graph classes. The most well-known and perhaps the earliest graph class of this kind is the class of cographs (see Section 2.8.3 for more details on cographs). It has been proved that cographs are exactly those graphs which have clique-width at most 2, and a 2-expression for them can be found in linear time [69].

Thus, the implication of Corollary 1.3.3 for us is that, in studying the M-partition problem and its list version for special graph classes, we only have to focus on those graph classes which either have unbounded clique-width or have bounded clique-width k but no efficient algorithm is known to find k-expressions constructing their graphs (As for other graph classes, we already know that all list M-partition problems are polynomial thanks to Corollary 1.3.3 above.) We note that the boundedness of clique-width and the complexity of finding a k-expression is known for almost all the standard graphs classes (see [97]). In Chapter 3 we will explain that all the graph classes we consider in this thesis (i.e., those in Table 1.3) have unbounded clique-width.

Now let us define $LinEMSOL(\tau_{1,p})$ problems in more detail and explain why they include (list) M-partition problems. We denote by τ_1 the vocabulary consisting of a binary relation E (see the last paragraph of Section 1.2 for the definitions of vocabulary and structures). Any graph can be represented as a τ_1-structure by defining the domain of the structure to be $V(G)$, and the relation E to be representing the adjacencies in G (i.e., $E(u, v)$ if and only if u and v are adjacent in G). Note that the reverse is not true, i.e., not every τ_1-structure corresponds to a graph (for E may not be symmetric). For a fixed integer $p > 0$, the vocabulary $\tau_{1,p}$ is an extension of τ_1, and it consists of a binary relation E and p unary relations $U_1, U_2, \cdots, U_p$. Let G be a p-graph which uses distinct labels $l_1, l_2, \cdots, l_p$ for its vertices. Then G can be represented as a $\tau_{1,p}$-structure by defining the domain of the structure to be $V(G)$, the relation E to be representing the adjacencies in G, and $U_i(u)$ if and only if the vertex u has the label l_i ($i = 1, 2, \cdots, p$). Note that the reverse is not true, i.e., not every $\tau_{1,p}$-structure corresponds to a p-graph (for E may not be symmetric, or U_is may not represent a partition of $V(G)$ into p parts). From now on in this section, we treat graphs and p-graphs as τ_1-structures and $\tau_{1,p}$ structures, respectively.

We assume that the reader is familiar with first-order logic, as is a well-known topic. (We refer to [253, 5] for extensive expositions of this topic.) Here we briefly mention that, given a vocabulary τ, a first-order logic formula over τ consists of the following features: variables ranging over the domain of discourse (shown by small Latin letters), the atomic predicate $u = v$ (returns TRUE if and only if both u and v refer to the same object), the predicates corresponding to the relations in the vocabulary τ, logical connectives (AND $\wedge$, OR $\vee$, NOT $\neg$ and IMPLY $\to$) and quantifiers (for all $\forall$ and there exist(s) $\exists$). Monadic second-order logic, or $MSOL$, is an extension of first-order logic with the extra feature of having variables which range over sets of the objects in the domain of discourse. These variables are called *set variables*, and shown by capital Latin letters. For any set variable X and variable x, the atomic predicate $X(x)$ returns TRUE if and only if the object x belongs to the set X. In any $MSOL$ formula, a *free* variable (free set variable, respectively) is a variable (set variable, respectively) which is not used by any quantifier. We write $\phi(x_1, x_2, \cdots, x_q, X_1, X_2, \cdots X_r)$ to specify that the formula ϕ has free variable $x_1, x_2, \cdots, x_q$ and free set variables $X_1, X_2, \cdots X_r$. An $MSOL$ formula which has no free variable and no free set variable is called *closed*. An example of a closed $MSOL$ formula over τ_1 is $\exists D(\forall v_1, v_2(D(v_1) \wedge \neg D(v_2) \to E(v_1, v_2)))$, which means: "there exists a set D of vertices such that for any two vertices v_1 and v_2, if v_1 is in D and v_2 is not in D then v_1 and v_2 are adjacent" (More informally: "there exists a set of vertices D which is fully adjacent to the rest of the vertices.") We refer to [246, 107, 66] for more details on $MSOL$.

Let $\phi(x_1, x_2, \cdots, x_q, X_1, X_2, \cdots X_r)$ be an $MSOL$ formula based on a vocabulary τ (possibly $q = r = 0$, which means ϕ is closed). Given a τ-structure H and $d_1, d_2, \cdots d_q \in D^H$, $D_1, D_2, \cdots D_r \subseteq D^H$, we write $H \models \phi(d_1, d_2, \cdots d_q, D_1, D_2, \cdots D_r)$ to mean that the formula ϕ returns the value TRUE when the free variables x_i and X_j are replaced by the values d_i and D_j, respectively (for $i = 1, 2, \cdots q$ and $j = 1, 2, \cdots, r$), and each predicate R is defined as the relation R^H (for all relation symbols R in the vocabulary τ).

A decision problem P is said to be a $MSOL(\tau)$ *problem* if there exists a closed $MSOL$ formula ϕ based on the vocabulary τ and a class (i.e., a set) $\widetilde{H}$ of τ-structures such that the problem P can be expressed by the following statement: given a τ-structure $H \in \widetilde{H}$ as input, does $H \models \phi$ hold? We call this statement an $MSOL(\tau)$ statement expressing the problem P. We note that the problem instance consists only of H, whereas ϕ and $\widetilde{H}$ are part of the problem description. In fact, $\widetilde{H}$ is the input set of P, which is supposed to be clear from the description of P. For example, when expressing graph problems such as the 3-coloring problem, $\widetilde{H}$ is the set of those τ_1-structures which correspond to graphs. (Recall that not all τ_1-structures correspond to graphs.) The class of $LinEMSOL(\tau)$ is an extension of $MSOL(\tau)$ problems which includes optimization problems as well. Given a set X and a function $f : X \to \mathbb{R}$, we define $f(X)$ as $\sum_{x \in X} f(x)$. An optimization problem P is said to be a $LinEMSOL(\tau)$ *problem* if there exists an $MSOL$ formula $\phi(X_1, X_2, \cdots, X_r)$ based on the vocabulary τ, a class $\widetilde{H}$ of τ-structures, a fixed integer $t > 0$ and fixed constants a_{ij} $(i = 1, 2, \cdots r,\ j = 1, 2, \cdots t)$ such that the problem P can be expressed by the following statement: given a τ-structure $S \in \widetilde{H}$ and t functions $f_1, f_2, \cdots f_t : D^H \to \mathbb{R}$ as input, find an assignment z of the free set variables of ϕ (i.e., $z : \{X_1, X_2, \cdots, X_r\} \to 2^{D^H}$) which maximizes the value $\sum_{i=1}^{r} \sum_{j=1}^{t} a_{ij} f_j(z(X_i))$, subject to the condition $H \models \phi(z(X_1), z(X_2), \cdots z(X_r))$. We note that the problem instance consists of H and t functions $f_1, f_2, \cdots f_t$, whereas ϕ, $\widetilde{H}$, the constants r, t and a_{ij}s $(i = 1, 2, \cdots r,\ j = 1, 2, \cdots t)$ are part of the problem description.

Having completed the description of $LinEMSOL(\tau)$ problems, now we show that all (list) M-partition problems are $LinEMSOL(\tau_{1,p})$. Given a matrix M, the M-partition problem can be expressed by an $MSOL(\tau_1)$ statement in which $\widetilde{H}$ is the set of all τ_1-structures corresponding to graphs, and the formula ϕ is defined as follows:

$$\phi \equiv \exists V_0, V_1, \cdots V_{m-1}(partition(V_0, V_1, \cdots V_{m-1}) \wedge \bigwedge_{(i,j)\in[m]^2} ad(i, j))$$

where $partition(V_0, V_1, \cdots V_{m-1})$ is defined by

$$\forall u(\bigvee_{i=0}^{m-1} V_i(u)) \wedge \neg\exists v(\bigvee_{i\neq j}(V_i(v) \wedge V_j(v)))$$

and $ad(i, j)$ is defined by

$$ad(i,j) = \begin{cases} \forall u, v(\neg(u = v) \wedge (V_i(u) \wedge V_j(v)) \to \neg E(u, v)) & \text{if } M(i,j) = 0 \\ \forall u, v(\neg(u = v) \wedge (V_i(u) \wedge V_j(v)) \to E(u, v)) & \text{if } M(i,j) = 1 \\ 1 & \text{if } M(i,j) = * \end{cases}$$

This means M-partition problems are $MSOL(\tau_1)$, and thus $LinEMSOL(\tau_{1,p})$ for any integer $p > 0$. As for the list M-partition problem, we need to figure out how to represent an instance $I = (G, L)$ of this problem by a labeled graph. We describe one way to do this. Given a list $L \subseteq [m]$, define $\|L\| = \sum_{i\in L} 2^i$. We represent the instance $I = (G, L)$ by a labeled graph G' which is obtained from G by assigning the label $\|L(v)\|$ to each vertex $v \in V(G)$. This means G' is a 2^m-graph. It is easy to see that I and G' can be uniquely obtained from each other in linear time (in terms of $n = |V(G)|$). Now the list M-partition problem can be expressed by an $MSOL(\tau_{1,2^m})$ statement in which $\widetilde{H}$ is the set of all $\tau_{1,2^m}$-structures corresponding to 2^m-graphs, and the formula ϕ is defined as follows:

$$\begin{aligned}
\phi \quad &\equiv \quad \exists V_0, V_1, \cdots V_{m-1}(partition(V_0, V_1, \cdots V_{m-1}) \wedge \bigwedge_{(i,j)\in[m]^2} ad(i,j) \\
\wedge \quad & \forall u(\bigwedge_{i=0}^{m-1}(V_i(u) \longrightarrow \bigvee_{i\in L\subseteq[m]} U_{\|L\|}(u))))
\end{aligned}$$

This means list M-partition problems are $MSOL(\tau_{1,2^m})$, and thus $LinEMSOL(\tau_{1,2^m})$.

Chapter 2

History of Partition Problems

The main incentive for introducing the M-partition problem stems from the observation that many well-known graph problems can be formulated as M-partition problems [129]. We first introduce some of the most notable problems of this kind in Sections 2.1 to 2.5. These problems arise in different areas of graph theory (prior to the introduction of the M-partition problem), mostly in the study of perfect graphs. The size of the matrix M for these problems rarely exceeds four, and yet it yields non-trivial and important problems. To have a straightforward exposition as to how these problems can be modeled using the M-partition problem, we assume that the input graphs of the problems described in this chapter are without loops. The case in which some vertices of the input graph have loops can still be modeled, though at the expense of some more elaborate (but still easy) arguments involving the list version. We assume that the reader is be able figure this out.

Our aim of introducing these problems is to offer a clear idea with regard to the relevance and significance of the M-partition problem as a common generalization of many important graph problems. After introducing some of these graph problems, we give in Section 2.6 an account regarding the emergence of the M-partition problem and its list version. In Sections 2.7 and 2.8 we introduce the main research directions of studying the M-partition problem relevant to this thesis. We refer to the survey [180] for other directions and variations of the M-partition problem.

2.1 Graph Coloring

Graph coloring is one of the oldest and most studied problems in graph theory [159, 215]. In its general form, which is called the *vertex coloring* problem, a graph G and an integer $k > 0$ are given as inputs, and the question is whether the graph G is k-colorable (see Section 1.2 for the definition of k-colorable.) In many algorithmic settings, the parameter k is considered to be a fixed constant (i.e., not part of the input). In this case, we call the problem *k-coloring.* And the third problem, is the *minimum coloring* problem, which seeks to find the chromatic number of a given graph G (i.e., the minimum value of k for which G is k-colorable). Note that any l-clique-covering of a graph G is also an l-coloring of the complement of G. Thus by applying all these definitions to the complement of the input graphs, we can define the problems of *vertex clique-covering, l-clique-covering and minimum clique-covering.* To keep the terms short, we will use the term clique-covering instead of vertex clique-covering from now on.

While the vertex coloring and the minimum coloring problems cannot be modeled as M-partition problems, for the k-coloring problem we can define M to be the 0-diagonal matrix of size k with * everywhere off-diagonal. Then the (list) M-partition problem is equivalent to the (list) k-coloring problem. The k-coloring problem is a classic example of the M-partition problem. Similarly, we can model l-clique-covering by defining M to be the 1-diagonal matrix of size l with * everywhere off-diagonal.

An early problem related to graph coloring was posed in 1852 by Francis Guthrie asking whether any map of countries can be colored using four colors such that no two neighbor countries have the same color. Using modern terminology, this is equivalent to asking whether any planar graph is 4-colorable. This problem, titled the Four-Color Theorem, was the subject of much early research until its final settlement in 1976 [6].

As a simple observation we have $\chi(G) \geq \omega(G)$. (Recall that $\omega(G)$ is the order of the largest clique in G.) An early research direction involved finding more in-depth relations between the functions χ and ω (see [252] for a survey of these results). This led to the birth of the perfect graph theory which we will detail in Section 2.4.

The list version of graph coloring problems (e.g., the list k-coloring problem) was introduced independently by Vizing [275] and Erdös et al. [113]. These problems were subject of much study and many real-world applications have been found (see [252, 270, 3, 270, 3, 204]).

The coloring problems belong to the first identified NPcomplete problems. In fact, the

k-coloring problem is proved to be NP-complete for $k \geq 3$, and polynomial for $k \leq 2$ [208]. This led many authors to study specific graph classes for which the (list) k-coloring problems (for $k \geq 3$) can be proved to be polynomial. We will give a more detailed discussion of this topic for many graph classes in Chapter 3. For now, we just mention that the coloring problems, in particular the (list) k-coloring problem (as special case of the M-partition problem), have been studied for almost all standard graph classes (see [97]).

2.2 Graph Homomorphisms

Let H be an arbitrary graph. A *(graph) homomorphism* of a graph G to the graph H is a mapping $f : V(G) \to V(H)$ such that $uv \in E(G)$ requires $f(u)f(v) \in E(H)$. Homomorphism have been studied in various contexts [2, 22, 139, 173, 177, 178, 186, 84, 214, 223, 233, 234, 243, 244, 233]. Here we focus on the aspects related to the M-partition problem.

Given a fixed graph H (with possible loops), the *H-coloring problem* asks whether there is a homomorphism from the input graph G to the graph H. The first study of this problem was carried out in [187]. The H-coloring problem is a special case of the M-partition problem: let M be the corresponding matrix of H (obtained from the adjacency matrix of H by replacing 1 entries with *), then it is easy to see that the M-partition problem is equivalent to the H-coloring problem. Note that in this case the matrix M is 1-free. In other words, H-coloring problems are exactly those M-partition problems in which M is 1-free. Also, if H is the complete graph on k vertices (which has no loops) then H-coloring will be equivalent to k-coloring (So the H-coloring problem generalizes the k-coloring problem.)

The complexity of the H-coloring problem was studied by several authors and some partial results were obtained [2, 22, 200, 223, 234, 243]. Hell and Nešetřil [187] unified all these results and filled the gaps to obtain a complete complexity result as follows:

Theorem 2.2.1. *([187]) The H-coloring problem is polynomial if H has a loop or is a bipartite graph, and otherwise it is NP-complete.*

As for the list version of the H-coloring problem, the technical advantage of the lists, which enables one to perform recursions, led to several strong complexity results (see [129]). The case in which every vertex in H has a loop is called *reflexive* list H-coloring. This is equivalent to the list M-partition problem in which M is a *-diagonal 1-free matrix. We have the following complete dichotomy (see Section 3.1.1 for the definition of interval graphs.)

Theorem 2.2.2. *([118]) The reflexive list H-coloring problem is polynomial if H is an interval graph, and otherwise it is NP-complete.*

Next, the case in which H has no loops was considered. This is called *irreflexive* list H-coloring, which is equivalent to the list M-partition problem when M is a 0-diagonal 1-free matrix. As in the previous case, a complete dichotomy is known (see Section 3.1.2 for the definition of circular arc graph.)

Theorem 2.2.3. *([124]) The irreflexive list H-coloring problem is polynomial if H is a bipartite co-circular arc graph, and otherwise it is NP-complete.*

Later on, even a broader dichotomy result was obtained covering all graphs H (equivalently, all 1-free matrices M):

Theorem 2.2.4. *([125]) The list H-coloring problem is polynomial if H is a bi-arc graph, and otherwise it is NP-complete.*

We note that bi-arc graphs form a graph class containing both interval and co-circular arc graphs. We refer to [125] for its formal definition.

The H-coloring problem was also studied for special graph classes. Enright et al. [112] proved that the list H-coloring problem can be solved in polynomial time for interval and permutation graphs (see Chapter 3 for the definition of these graph classes). Several papers studied the H-coloring problem and its list version for bounded-degree graphs (i.e., the graphs in which the degree of each vertex is $\leq c$, for some fixed constant $c \geq 0$) [120, 127, 126, 148, 188].

Several generalizations and variations of H-coloring are known in the literature. The digraph version is the one in which H and input graph G are both digraphs, and the question is whether there exists a homomorphism from G to H, i.e., a mapping $f : V(G) \to V(H)$ such that $(u, v) \in E(G)$ requires that $(f(u), f(v)) \in E(H)$. This problem is called the *digraph H-coloring* problem or the *H-coloring problem for digraphs.* The list version of this problem (defined similarly to the case of graphs) is called the *digraph-list H-coloring* problem or the *list H-coloring problem for digraphs.* Many partial results are obtained for the complexity of the digraph H-coloring problem [10, 11, 9, 12, 15, 179]. The dichotomy for all digraph-list H-coloring problems (i.e., each such problem is polynomial or NP-complete) is proved in [44, 14] . Hell et al. [191] gave the characterizations of those graphs H for which the digraph-list H-coloring problem is polynomial. More importantly, there is a link between the complexity of the digraph H-coloring problem and another deep question in complexity theory, namely the Feder-Vardi conjecture [138]. We will explore this link in more detail in the next section.

We should also mention that the M-partition problem can be seen as H-coloring with H being a more general structure than graphs, known as trigraphs. Such concept are treated in [135, 136, 50].

2.3 Constraint Satisfaction Problems (CSPs)

Constraint Satisfaction Problems, or CSPs, form an important class of problems which generalizes graph homomorphism. Not all M-partition problems are CSPs, nor can every CSP be modeled as an M-partition problem [120, 121, 129]. However, CSPs and M-partition problems share some important problems such as the H-coloring problem. We will explore the link between CSPs and M-partition problems in more detail in Section 2.7, where we will see how this link inspired the study of the dichotomy problem for (list) M-partition problems. For this reason, in this section we introduce and briefly discuss CSPs and their relevant results.

In a CSP, we are given a set of variables and constraints on different subsets of these variables. The goal is to find an assignment of values to the variables which satisfies all the constraints. Formally, let τ be a vocabulary (see the last paragraph of Section 1.2 for the definitions of vocabulary and structures) and H a fixed τ-structure, then *CSP(H)* is the problem asking whether there is a homomorphism from a given τ-structure G (as input) to H, namely a mapping $f : D^G \to D^H$ such that if $(x_1, x_2, \cdots x_l) \in R^G$ then $(f(x_1), f(x_2), \cdots f(x_l)) \in R^H$, for all $(R, l) \in \tau$ and $x_1, x_2, \cdots x_l \in D^G$.

The complexity of the CSP(H), when H is a fixed structure, has been studied for many special classes of H (see [44]). Many of these studied classes exhibited a dichotomy property, i.e., each CSP(H) (for any choice of H within the class) was polynomial or NP-complete. Recall that such dichotomy is not trivial, as there are NP problems that are neither polynomial nor NP-complete, unless $P = NP$ [219]. Some notable results related to this dichotomy property are as follows: Schaefer [260] identified the polynomial cases for Boolean CSPs (namely, CSPs in which D^H is a 2-element set), and proved the rest to be NP-complete. Hell et al. [187] offered a similar dichotomy result for the CSPs corresponding to H-coloring problems (see Theorem 2.2.1). This type of dichotomy result attracted much attention and was proved for several other special cases [44] (see [43, 73, 138]). Feder and Vardi in [138] conjectured that the dichotomy property holds in general, i.e., for each fixed structure H, the CSP(H) is polynomial or NP-complete. After their names, this conjecture is known as the *Feder-Vardi conjecture*.

Additionally, Feder and Vardi identified several classes of CSPs(H) which have the expressive power of the whole set of CSPs(H). In other words, they identified several classes $\widetilde{H}$ of structures H with this property: for an arbitrary structure H, there exists another structure $H' \in \widetilde{H}$ (which can be found in polynomial time) such that CSP(H) and CSP(H') are polynomially equivalent (i.e., each problem can be reduced in polynomial time to the other problem). One such class is those CSPs(H) which are equivalent to digraph H-coloring problems (introduced in the previous section).

Despite much effort, establishing the Feder-Vardi conjecture for general CSP(H) remains as an outstanding open problem until present [120, 129]. However, it was validated in several special cases of H. Bulatov [43] proved the conjecture for the CSP(H) when $|D^H| = 3$. Much progress was made to handle the case of bigger domains [45, 46, 80, 138, 187, 203, 202, 213]. Yet the failure to settle it completely is taken as evidence that this problem could be very hard [44]. Bulatov [44] proved the dichotomy property for a certain class of the CSP(H) called conservative, which generalizes the list H-coloring problem for general graphs and digraphs. A shorter and more digestible proof for this result is presented in [14]. Hell and Rafiey [191] gave a structural characterization for this dichotomy result (analogues to the undirected version as presented in Theorem 2.2.4). Note that the non-list version of this dichotomy result (i.e., the dichotomy property for digraph H-coloring problems) will actually imply the Feder-Vardi conjecture as mentioned earlier. Feder et al. [120] further generalized Bulatov's result to some broader class of structures. A possible criteria for a CSP(H) to be polynomial is suggested in [46]. A short survey of dichotomy results related to CSPs is given in [73].

2.4 Partition Problems in the Study of Perfect Graphs

Perfect graphs are one of the most studied graph classes in the history of graph theory. The study of perfect graphs includes several special M-partition problems. Also, the M-partition problem itself was studied when the input graph is restricted to perfect graphs. For this reason, in this section we introduce perfect graphs and give a brief account of the

M-partition problems appeared in their literature. The other aspect, namely the study of the M-partition problem restricted to perfect graphs, is detailed in Section 2.8.

A graph G is called perfect if the chromatic number of any of its induced subgraph G_0 equals the size of the largest clique of that subgraph (i.e., $\chi(G_0) = \omega(G_0)$ [19]). Perfect graphs contain many important graphs classes, such as bipartite graphs, the line graphs of bipartite graphs (see Section 3.2), chordal graphs (see Section 2.8), split graphs (see Section 2.5), interval graphs (see Section 3.1.1) and comparability graphs (see Section 3.1.3).

The study of perfect graphs was initiated in a 1960 paper of Claude Berge [19]. Since then, many different problems and properties related to perfect graphs were studied. One of the important discoveries was that, a graph is perfect if and only if its complement is perfect. This property, sometimes called the *Perfect Graph Theorem*, was conjectured by Berge and later settled by Lovasz [226]. Berge also made another conjecture, stating that the only minimal non-perfect graphs are cycles with odd number of vertices larger than four (later called *odd holes*) and their complements (later called *odd anti-holes*). An equivalent statement is that, a graph is perfect if and only if it has no induced subgraph isomorphic to an odd hole or an odd anti-hole. A graph with this last condition was later called Berge graph. So, in today's language, Berge's second conjecture stated that perfect graphs are precisely Berge graphs. This conjecture, which was later called the *Strong Perfet Graph Conjecture*, remained open and became one of the most studied directions in graph theory until its final settlement in 2006 by Chudnovsky et al. in [53]. Additionally, in a different paper, Chudnovsky et al. provided a polynomial-time algorithm to recognize perfect graphs [52]. The study of Strong Perfet Graph Conjecture, including the works of Chudnovsky et al. settling the conjecture and finding efficient algorithm to recognize Berge graphs, involved many deep structural results with some of them related to several special cases of the M-partition problem. Now we briefly introduce these M-partition problems in the remainder of this section. Please refer to the cited references for more detail concerning each problem.

A *cutset* of graph G is a non-empty proper subset X of $V(G)$ whose removal makes the graph disconnected (i.e., the graph $G[V(G) - X]$ is disconnected). When X is a stable set, it is called a *stable cutset*. The stable cutset problem asks whether a given graph G has a stable cutset. This problem can be modeled as an M-partition problem for the matrix M shown in Figure 2.1(a), with the extra condition that each part must be non-empty. The importance of this problem in studying perfect graphs was first demonstrated by Tucker in [266]. This problem was shown to be NP-complete [212]. Later on, it was proved to be

NP-complete even when the input graph is restricted to line graphs (see Section 3.2) [30].

$$\begin{bmatrix} 0 & * & * \\ * & * & 0 \\ * & 0 & * \end{bmatrix} \quad \begin{bmatrix} * & * & 0 & * \\ * & 1 & * & * \\ 0 & * & * & * \\ * & * & * & 1 \end{bmatrix} \quad \begin{bmatrix} * & * & 0 & * \\ * & * & * & 1 \\ 0 & * & * & * \\ * & 1 & * & * \end{bmatrix}$$

(a) stable cutset (b) 2-clique cutset (c) skew cutset

Figure 2.1: The matrix M for some special problems.

A cutset which is a clique is called a *clique cutset*. We will discuss this concept in Section 2.5. A *2-clique cutset* is a cutset that is the union of two cliques (i.e., the complement of the graph induced by the cutset is a bipartite graph). The 2-clique cutset problem asks whether a given graph G has a 2-clique cutset. This problem can be modeled as an M-partition problem for the matrix M shown in Figure 2.1(b), with the extra condition that the parts V_0,V_2 and the union of parts V_1 and V_3 must be non-empty. The application of this problem in the study of perfect graphs was demonstrated in [176, 1]. The first sub-exponential algorithm to solve this problem was introduced in [129]. A polynomial-time algorithm was later found in [48].

A *skew cutset* is a cutset which can be partitioned into two non-empty parts X_0 and X_1 which are fully adjacent (i.e., any vertex in X_0 is adjacent to any vertex in X_1). The skew cutset problem asks whether a given graph G has a skew cutset. This problem is equivalent to an M-partition problem for the matrix M shown in Figure 2.1(c), with the extra condition that each part must be non-empty. Skew cutsets was first introduced by Chvátal [56] in his study of perfect graphs, in which analyzing the complexity of the skew cutset problem was posed as an open problem. Determining the complexity of the skew cutset problem (as an M-partition problem) was open for many years, until the first sub-exponential time algorithm to solve the list version of this problem was provided in [129], followed by a polynomial-time algorithm in [85]. A more efficient solution, for the basic version, was later presented in [209]. Skew cutsets also played an important role in proving the Strong Perfect Graph conjecture by Chudnovsky et al. [53].

A non-empty proper subset $X \subseteq V(G)$ is called a *homogeneous set*, or a *module*, if any vertex $v \in V(G) - X$ is either adjacent to all vertices in X or is adjacent to none of them (i.e., $N(v) \cap X$ is either X or $\emptyset$). For a graph G, a pair (X_0, X_1) of disjoint non-empty subsets of $V(G)$ is called a *homogeneous pair* if: 1) for every $v \notin X_0 \cup X_1$, $N(v) \cap X_0$ is either empty or equal to X_0 and $N(v) \cap X_1$ is either empty or equal to X_1, 2) $|X_0| \geq 2$ or $|X_1| \geq 2$, and 3) $|V(G) - X_0 \cup X_1| \geq 2$. The homogeneous pair problem asks whether a given graph G has a homogeneous pair. This problem can be modeled as an M-partition problem for the matrix M shown in Figure 2.2(a), with the extra condition that the part X_0 or X_1 has at least two vertices, and the union of other parts also has at least two vertices. An application of this problem was first described by Chvátal and Sbihi [59] in validating a special case of the Strong Perfect Graph conjecture. A homogeneous pair in which the parts X_0 and X_1 are both cliques is called a *homogeneous pair of cliques.* The problem of asking whether a given graph G has a homogeneous pair of cliques can be modeled as an M-partition problem for the matrix M shown in Figure 2.2(b), with the same extra conditions as the homogeneous pair problem. As we will see in Section 6.2, this problem has a role in describing the structure of claw-free graphs. The homogeneous pair problem is shown to be polynomial in [115]. The homogeneous pair of cliques problem is also proved to be polynomial in [115]. A faster algorithm to solve some special cases is presented in [210].

$$\begin{bmatrix} \ast & \ast & 1 & 0 & 1 & 0 \\ \ast & \ast & 1 & 0 & 0 & 1 \\ 1 & 1 & \ast & \ast & \ast & \ast \\ 0 & 0 & \ast & \ast & \ast & \ast \\ 1 & 0 & \ast & \ast & \ast & \ast \\ 0 & 1 & \ast & \ast & \ast & \ast \end{bmatrix}$$

(a) homogeneous pair

$$\begin{bmatrix} 1 & \ast & 1 & 0 & 1 & 0 \\ \ast & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & \ast & \ast & \ast & \ast \\ 0 & 0 & \ast & \ast & \ast & \ast \\ 1 & 0 & \ast & \ast & \ast & \ast \\ 0 & 1 & \ast & \ast & \ast & \ast \end{bmatrix}$$

(b) homogeneous pair of cliques

$$\begin{bmatrix} \ast & 1 & \ast & 0 & \ast & 0 & 1 \\ 1 & \ast & 0 & \ast & 0 & \ast & 1 \\ \ast & 0 & \ast & 1 & \ast & 0 & 1 \\ 0 & \ast & 1 & \ast & 0 & \ast & 1 \\ \ast & 0 & \ast & 0 & \ast & 0 & \ast \\ 0 & \ast & 0 & \ast & 0 & \ast & \ast \\ 1 & 1 & 1 & 1 & \ast & \ast & 1 \end{bmatrix}$$

(c) 2-amalgam

Figure 2.2: The matrix M for some special problems.

Several other partitions (or decompositions, as there are typically called) appeared in the study of perfect graphs. They include *join-decomposition, 2-join-decomposition* [77, 64], and their more general cases known as *amalgam* and *2-amalgam decompositions.* The problems of deciding whether a graph has these decompositions can be modeled as M-partition problems. The matrix M corresponding to the 2-amalgam decomposition is shown in Figure 2.2(c). The matrices corresponding to other decompositions can be obtained from this matrix by removing certain rows and columns [129]. All these M-partition problems can be solved in polynomial time [75, 76, 47, 64]. We refer to [21, 47, 267, 278] for the applications of these problems.

2.5 Other Partition Problems

In this section we describe some other special cases of the M-partition problem studied in the literature of graph theory. A graph G is a *split graph* if $V(G)$ can be partitioned into a stable set and a clique [164]. Split graphs are a well-known subclass of perfect graphs with many interesting properties (see [164, 32]). The recognition problem of split graphs (deciding whether a given graph is a split graph or not) can be modeled as an M-partition problem with the matrix M shown in Figure 2.3(a). It is known that split graphs can be characterized by finitely many forbidden subgraphs [143]. More precisely, a graph is a split graph if and only if it does not contain any induced subgraph isomorphic to $\overline{C_4}, C_4$ or C_5 (C_t is a cycle with t vertices). This easily yields a polynomial-time algorithm to recognize split graphs (which is an M-partition problem). In fact, even a linear time algorithm is known for solving this problem [143, 185].

$$\begin{bmatrix} 0 & * \\ * & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & * & * & * \\ * & 0 & * & * \\ * & * & 1 & * \\ * & * & * & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & * & * \\ 1 & 0 & * & * \\ * & * & 1 & 0 \\ * & * & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & * & * & * & * \\ * & 0 & 1 & 0 & 0 \\ * & 1 & 0 & 0 & 0 \\ * & 0 & 0 & 0 & 1 \\ * & 0 & 0 & 1 & 0 \end{bmatrix}$$

(a) split graph (b) $(2,2)$-graph (c) $(2,2)$-polar graph (d) 2-bisplit

Figure 2.3: The matrix M for some special problems.

There are several generalizations and variations of split graphs which lead to similar M-partition problems. One natural generalization is the concept of (k,l)-*graph*, defined as any graph G whose vertices can be partitioned into $k > 0$ stable sets and $l > 0$ cliques [28]. The recognition problem of (k,l)-graphs is an M-partition problem in which M is a $(k+l)\times(k+l)$ matrix with k 0s and l 1s on the main diagonal, and * everywhere off-diagonal. An example of such matrix for $k = l = 2$ is given in Figure 2.3(b). When $k, l \leq 2$, the recognition problem is shown to be polynomial [27]. Some new and more efficient algorithms were demonstrated later [37, 129]. On the other hand, if at least one of the parameters k or l is at least three then the recognition problem is proved to be NP-complete [28, 37]. Converting these results into the framework of the M-partition problem, we can deduce the following dichotomy result for 01-diagonal matrices M with * everywhere off-diagonal: the M-partition problem is polynomial if the number of 0s and the number of 1s on the main diagonal are both less than three, and otherwise it is NP-complete. The (k,l)-graph recognition problem has also been studied when the input graph is restricted to cographs (see Section 2.8.3) and chordal graphs (see Section 2.8.2).

Another generalization of split graphs is (k, l)-*polar graphs*, introduced in [49]. A graph G is a (k, l)-polar graph if $V(G)$ can be partitioned into two parts V_r and V_b such that the graphs $\overline{G[V_r]}$ and $G[V_b]$ are the disjoint union of $k > 0$ and $l > 0$ cliques, respectively. In other words, it is equivalent to partitioning $V(G)$ into a complete k-partite graph and a disjoint union of l cliques. The recognition problem, which is introduced in [49], can be modeled as an M-partition problem where the matrix M is a 01-digonal matrix of size $k+l$ with its $k \times k$ block A having 1 everywhere off-diagonal, its $l \times l$ block B having 0 everywhere off-diagonal and its block C being all *. An example of the matrix M for $k = l = 2$ is given in Figure 2.3(c). This problem is proved to be polynomial in [49]. It has also been studied for some special graph classes including cographs [111] and chordal graphs[110], for which efficient polynomial-time algorithms were demonstrated.

Brandstädt et al. [32] introduced *bisplit graphs*, as those graphs whose vertex set can be partitioned into a stable set and a complete bipartite graph. They also introduced a generalized version, namely k-bisplit graphs, defined as any graph G whose vertices can be partitioned into a stable set and a disjoint union of $k > 0$ complete bipartite graphs. The recognition problem of k-bisplit graphs can be modeled as an M-partition problem in which M is a 0-diagonal matrix of size $2k+1$ with the 0-th row corresponding to the stable set and the rows $2i-1$ and $2i$ corresponding to one of the complete bipartite graphs ($i = 1, 2, \cdots k$). The entries are defined as $M(2i-1, 2i) = 1$, for $i = 1, 2, \cdots k$ (to indicate that the biparite graphs are complete), and $M(i, j) = 0$, for $1 \leq i < j \leq 2k$ and $j > i+1$ (to indicate the disjointness of bipartite graphs), and all other entries are *. An example of such matrix for $k = 2$ is given in Figure 2.3(d). When the parameter k is fixed, the recognition problem is proved to be polynomial in [32]. A similar polynomial result is also obtained in [132].

The clique cutset problem asks whether a given graph G has a clique cutset [267]. This problem can be modeled as an M-partition problem for the matrix M shown in Figure 2.4(a), with the extra condition that each part must be non-empty. This problem has applications in designing efficient algorithms to solve many optimization problems (see [267]). Many efficient algorithms are provided to solve this problem [164, 267, 277, 278].

The problem of asking whether a given graph G has a module (see Section 2.4) can be modeled as an M-partition problem for the matrix M shown in Figure 2.4(b), with the extra condition that the 0-th part and the union of the other two parts must be non-empty.

$$\begin{bmatrix} 1 & * & * \\ * & * & 0 \\ * & 0 & * \end{bmatrix} \quad \begin{bmatrix} * & 0 & 1 \\ 0 & * & * \\ 1 & * & * \end{bmatrix} \quad \begin{bmatrix} 1 & * & 0 & * \\ * & 1 & * & 0 \\ 0 & * & 1 & * \\ * & 0 & * & 1 \end{bmatrix} \quad \begin{bmatrix} * & * & 0 & * \\ * & * & * & 0 \\ 0 & * & * & * \\ * & 0 & * & * \end{bmatrix}$$

(a) clique cutset (b) homogeneous set (c) clique-cross (d) Winkler's partition

Figure 2.4: The matrix M for some special problems.

This problem is part of a well-known decomposition, known as the *modular decomposition* (see [70, 164, 237, 238]). Several efficient algorithms are developed to solve this problem [70, 164, 227].

A *clique-cross* partition is a partition of $V(G)$ into four cliques V_0, V_1, V_2, V_3 such that V_0 and V_2, as well as V_1 and V_3 are fully non-adjacent. The clique-cross problem asks whether a given graph G has a clique-cross partition. This problem can be modeled as an M-partition problem for the matrix M shown in Figure 2.4(c), and it can be solved in linear time [116]. Note that the matrix corresponding to the clique-cross problem can be obtained from the adjacency matrix of C_4 (the cycle of order 4) by setting all 1 entries to * and setting all the entries on the main diagonal to 1. This suggests a generalization to the so-called *H-clique* problem by replacing C_4 with an arbitrary graph H (So it is an M-partition problem where M is obtained from the adjacency matrix of H by setting all 1 entries to * and setting all the entries on the main diagonal to 1.) The H-clique problem (which is also an M-partition problem) is studied in [229] for which a complete dichotomy is obtained as follows: the H-clique problem is polynomial if H has no clique of size 3, and otherwise it is NP-complete.

A similar problem is posed by Peter Winkler asking whether $V(G)$ can be partitioned into four non-empty parts V_0, V_1, V_2, V_3 such that V_0 and V_2, as well as V_1 and V_3 are completely non-adjacent, and there is at least one edge between V_0 and V_1, between V_1 and V_2, between V_2 and V_3, and between V_3 and V_0. This is an M-partition problem with some extra conditions on the number of vertices and edge between the parts. Its corresponding matrix M is shown in Figure 2.4(d). We refer to [180] for the applications of this partition problem.

2.6 The M-Partition Problem and its Variations

As we saw in previous sections, many prominent problems in combinatorics and graph theory are special cases of the M-partition problem. The study of the M-partition problem in its general form began by Feder el al. in [129]. Their aim was to unify all such problems into a single framework.

They noticed that many of these problems contain requirements not addressed in the formulation of the M-partition problem. Take the example of the clique cutset problem (see Section 2.5), which requires all parts to be non-empty, or consider Winkler's partition problem (see Section 2.5) which, in addition to the non-emptiness of each part, requires at least one edge between some certain parts. Other problems described in previous sections, implicitly or explicitly, contain similar requirements. To capture these requirements, Feder et al. also introduced the *list version* of the M-partition problem (see Section 1.1 for its formal definition).

Let us explain how we can model the clique cutset problem (namely its extra condition for the non-emptiness of each part) using the list version. Since each part V_i, $i = 0, 1, 2, 3$ has to be non-empty, in any solution of the problem there must be distinct vertices $v_i \in V_i$ $(i = 0, 1, 2, 3)$. Note that total number of choices of v_is is at most $4!\binom{n}{4}$. We consider all these choices. For a given choice, we need to place v_i in the part V_i. We can model this requirement by defining the list $L(v_i) = \{i\}$ for $i = 0, 1, 2, 3$. We define the list of other vertices as containing all the parts (i.e., the list $[m]$). Thus, for each choice of v_is, we created an instance of the list M-partition problem (where M is the matrix corresponding to the clique cutset problem). It is easy to see that any solution of the original clique cutset problem is a solution of one of these instances and vice versa. Thus, solving the original problem is equivalent to solving at most $4!\binom{n}{4}$ (that is polynomially many) instances of the list M-partition problem . In the case of Winkler's partition problem, note that in any solution of the problem there must be edges $e_i = u_i v_i$ between the part V_i and V_{i+1} (where $u_i \in V_i$ and $v_i \in V_{i+1}$, $i = 0, 1, 2, 3$, all index calculations are modulo 4). The total number of choices for the vertices u_i and v_i $(i = 0, 1, 2, 3)$ is at most $O(n^8)$. Again, we consider all such choices and fix a choice for the edges $e_i = u_i v_i$ $(i = 0, 1, 2, 3)$. This requires the vertices u_i and v_{i-1} to be placed in the part V_i, which can be modeled by defining their lists to be $\{i\}$. We define the list of other vertices as containing all the parts. Now, as in the case of the clique cutset problem, it is easy to see that solving the original problem is equivalent to solving $O(n^8)$ (that is polynomially many) instances of the list M-partition problem (where M is the matrix corresponding to Winkler's partition problem). Using these examples as a guide, the reader should be able to see how different restrictions on vertices and edges inside or between the parts can be modeled using the list version.

We should mention that other than the basic version (i.e., the original version without lists) and the list version, there are other variations of the M-partition problem, which have been studied for special cases the matrix M. These variations include: 1) the *retraction version* ([118, 128, 192]): a special case of the list version in which each list consists of either one single part or all the parts, and 2) the *surjective version* ([23, 82, 161, 232, 274]): same as the non-list version, with the extra restriction that each part must be non-empty. Using a similar approach as in the case of list version, one can convert these variants into (polynomially many) instances of the list version. This makes the list version the dominant variant, particularly in studying the M-partition problem. We follow the same trend in this thesis by remaining focused on the list version. Another variation is the *digraph version* defined as follows: given a (not necessarily symmetric) matrix M with entries from the set $\{0, 1, *\}$), an M-partition of a digraph G is a partition of the set $V(G)$ into parts $V_0, V_1 \cdots V_{m-1}$ such that for all $0 \leq i, j < m$ and any two distinct vertices $u_i \in V_i$ and $u_j \in V_j$, we have $(u_i, u_j) \in E(G)$ if $M(i, j) = 1$, and $(u_i, u_j) \notin E(G)$ if $M(i, j) = 0$. The problem which asks whether or not the input digraph G has an M-partition is called the *digraph M-partition problem* or the *M-partition problem for digraphs.* The list version of this problem (defined similarly to the case of graphs) is called the *digraph-list M-partition problem* or the *list M-partition problem for digraphs.* We refer to [136, 180] for more details on this variation. Here we just mention that the digraph (-list) H-coloring problem is equivalent to the digraph (-list) M-partition problems for 1-free (not necessarily symmetric) matrix M.

2.7 The Dichotomy and Characterization Problems

After introducing the M-partition problem, Feder el al. (in [129]) dedicated the rest of their paper to analyzing its computational complexity. The main problem they considered was to classify matrices M into those for which the M-partition problem was polynomial and those for which it is NP-complete [48]. As mentioned earlier in Section 1.1, we call this problem the dichotomy problem for M-partition problems. This problem is motivated by several special cases of the M-partition problem, most notably the H-coloring problem, which exhibited such dichotomy property. Recall from Section 2.2 that the H-coloring problem is equivalent to the M-partition problem when M is 1-free, for which several complete dichotomy results are known, both for the basic and the list version (see Theorems 2.2.1 and 2.2.4). Also, a similar dichotomy result can be derived for the complement of the H-coloring problem, namely when M is 0-free. More precisely, we make the following simple observation:

Observation 2.7.1. *([180]) Any M-partition of G is an $\overline{M}$-partition of $\overline{G}$.*

By applying this observation, we can deduce from Theorems 2.2.1 and 2.2.4 that a complete dichotomy holds for the M-partition problem when M is 1-diagonal 0-free, and for the list M-partition problem when M is 0-free.

Both the H-coloring problem and its complement (i.e., when M is 1-free and 0-free, respectively), in addition to being special cases of the M-partition problem, are also special cases of CSP(H). We already gave a detailed account on CSPs in Section 2.3, where we explained the Feder-Vardi conjecture regarding the dichotomy property of all the CSP(H). Inspired by the dichotomy results for the H-coloring problem and its complement, Feder et al. posed the dichotomy problem for M-partition problems, as well as their list version, as a question analogous to that of the Feder-Vardi conjecture for CSPs. Validating this problem became a direction in studying the M-partition problem.

Later on, the link to the Feder-Vardi conjecture became more fortified and meaningful by the following result:

Theorem 2.7.2. *([121]) For every digraph H, there exists a matrix M such that the digraph H-coloring problem is polynomially equivalent to the M-partition problem for perfect graphs.*

Recall from Section 2.3 that any CSP(H) can be reduced in polynomial time to a digraph H'-coloring problem for some digraph H'. Thus proving the dichotomy problem for M-partition problems, even when the input graph is restricted to perfect graphs, will actually yield the Feder-Vardi conjecture. This link, on one hand further motivates the study of the dichotomy problem for M-partition problems, but on the other hand is evidence that it may not be easy to solve this problem [180]. It was based on this difficulty that the study of the M-partition problem (particularly the dichotomy problem) was carried out for the cases in which the input graph was restricted to special graph classes. The details about this direction are given in Section 2.8.

Now we give a brief description of the current results related to the complexity and the dichotomy of M-partition problems. We already mentioned the dichotomy results when M is 1-free or 0-free. The next natural case is when M is *-free, for which Feder et al. (in [129]) proved that any list M-partition problem can be solved in polynomial time. They conjectured the quasi-dichotomy property for all list M-partition problems (i.e., each problem is quasi-polynomial or NP-complete, see Section 1.2). This conjecture was later proved in [120]. Additionally, they studied the complexity of the list version when M has size no more than four. This study was completed by several subsequent papers, which together yielded the following complete dichotomy result:

Theorem 2.7.3. *([129, 48, 78]) Suppose M is a matrix with size ≤ 4. Then the M-partition problem is NP-complete if M has no * on the main diagonal and contains 3-coloring or its complement, otherwise it is polynomial. Also, the list M-partition problem is polynomial if M does not contain 3-coloring, stable cutset, reflexive four-cycle, stable cutset pair, or any of their complements (shown in Figure 2.5), and otherwise it is NP-complete.*

$$\begin{bmatrix} 0 & * & * \\ * & 0 & * \\ * & * & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & * & * \\ * & 1 & * \\ * & * & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & * & * \\ * & * & 0 \\ * & 0 & * \end{bmatrix}$$

3-coloring 3-coloring's complement stable cutset

$$\begin{bmatrix} * & * & 0 & * \\ * & * & * & 0 \\ 0 & * & * & * \\ * & 0 & * & * \end{bmatrix} \qquad \begin{bmatrix} * & * & 0 & 0 \\ * & 0 & * & 0 \\ 0 & * & 0 & * \\ 0 & 0 & * & * \end{bmatrix}$$

reflexive four-cycle stable cutset pair

Figure 2.5: The matrix M for some special problems.

In another paper, Feder et al. [137] focused on another approach to identify the matrices M for which the M-partition problem was polynomial. They proposed the following sufficient (but not necessary) condition: the M-partition problem is polynomial if all M-partitionable graphs can be characterized by a finite set of forbidden induced subgraphs, i.e., M has finitely many minimal obstructions. (Recall that a minimal obstruction is a graph which is not M-partitionable, but any of its proper induced subgraphs is M-partitionable, see Section 1.1.) So their aim was to determine which matrices M have finitely many minimal obstructions and which matrices do not. This problem is known as the *characterization problem* [180], which serves as a tool helping us to find polynomial cases for the M-partition problem. Note that, having infinitely many minimal obstruction does necessarily dismiss the possibility of a polynomial complexity. For example, the 2-coloring problem (as an M-partition problem) has infinitely many minimal obstructions (namely, odd cycles), while it can solved in polynomial time. We will also see that having finitely many minimal obstructions will not necessarily lead to a polynomial complexity for the list version (e.g., see Section 2.8.4).

The work of Feder et al., as well as some subsequent works, solved the characterization problem for many different classes of matrices. We refer to Tables 1.1 and 1.2 for a summary of these results and the survey [180] for more details.

In addition to the finiteness of the number of minimal obstructions, Feder et al. (in [137]) also concerned themselves with estimating the size of the largest minimal obstruction in terms of the size of the matrix M (and its blocks A and B). Similar attempts have also been made in [122]. From the complexity point of view, the implication of upper-bounding the size of the minimal obstructions was its effect in upper-bounding the running time of solving the M-partition problem. More precisely, let d be an upper-bound on the size of minimal obstructions. Then solving the M-partition problem, using the characterization property, can be performed in $O(n^{d+2})$ time. Thus, by upper-bounding the parameter d, we upper-bound the degree of the polynomial bound on the running time. In some later works (e.g., [121]), the author made distinctions between those matrices for which d can be polynomially upper-bounded in terms of m and those for which we have but an exponential upper-bound (in terms of m). One interpretation of this distinction could be that, when d is exponentially upper-bounded (in terms of m), then the running time $O(n^{d+2})$, even though polynomial, may not be considered as very practical, while having d polynomially bounded (in terms of m) is more acceptable. This also agrees with the common sense. Thus, in our study in this thesis, we insist on offering polynomial running times whose degrees are polynomially bounded by the size of M. For this reason, we included the running time of our algorithms in the theorem statements of this thesis, as evidence for the fulfillment of this promise.

2.8 Restriction to the Subclasses of Perfect Graphs

In the previous sections we already mentioned evidence suggesting the difficulty of solving the dichotomy and the characterization problems in general. This motivates the study of these problems in special cases in which the input graph is restricted to some certain well-behaved graphs, hoping that the problem may become more tractable.

2.8.1 Perfect Graphs

The study of the M-partition problem for perfect graphs was initiated in [119] and its revised version [121]. We already gave the definition and some background for perfect graphs in Section 2.4, where we mentioned an important line of research dedicated to proving the Strong Perfect Graph conjecture which led to several special cases of the M-partition problem. In this section, we mention some other research directions relevant to the M-partition problem restricted to perfect graphs.

Some special cases of the M-partition problem, namely the k-coloring, the l-clique-covering and the H-coloring problems are well-studied for the class of perfect graphs and many of its subclasses. Recall that the definition of perfect graphs implies that a perfect graph G is k-colorable if and only if it does not contain a clique of size $k+1$. This solves the characterization problem for the k-coloring problem when restricted to perfect graphs. It means we can solve the k-coloring problem for perfect graphs in polynomial time by considering all possible combinations of $k+1$ distinct vertices of the input graph and testing whether they form a clique or not (which can be accompolished in $O(n^{k+3})$ time.) Note that this algorithm does not provide a k-coloring of G (if one exists), but only states whether one exists or not. Some polynomial-time algorithms to find a k-coloring (or deciding that none exists), as well as for solving some other standard graph problems including the vertex coloring, the clique-covering, the maximum clique and the maximum stable set problems for perfect graphs are provided by Grötschel et al. [168]. All these algorithms are based on the ellipsoid method for linear programming. As noted by Grötschel et al. and other researchers (see for example [222, 198]), these algorithms are impractical and of theoretical interest only, thus finding efficient combinatorial algorithms for these problems, in particular the vertex coloring problem, is left as an open problem. Such algorithms are developed for many subclasses of perfect graphs though (e.g., chordal graphs [152]). We will discuss some of them in the subsequent sub-sections. We refer to the survey [230] for a more comprehensive treatment of this subject. The above arguments (about the characterization and complexity problems) also apply to the l-clique-covering problem, as it is equivalent to l-coloring the complement of the input graph, which is again a perfect graph (according to the Perfect Graph Theorem, see Section 2.4). As for the H-coloring problem, there is a similar single minimal obstruction characterization: a perfect graph G is H-colorable if and only if it does not contain a clique of size $\omega(H) + 1$ [180].

These results motivate the study of the M-partition problem in its general form for perfect graphs, which was first carried out in [121]. Their main result is the identification of a group of 01-diagonal matrices M called normal matrices, for which the number of perfect minimal obstructions (i.e., the minimal obstructions which are perfect graphs) is proved to be finite (see Table 1.2). They also proved Theorem 2.7.2 which we already stated in Section 2.3, and discussed how this result serves as evidence suggesting that the dichotomy problem may not be easy to solve for M-partition problems, even when restricted to perfect graphs. This dispels the hope that restricting to perfect graphs could simplify proving a complete dichotomy for the M-partition problem. (This is in contrast to the special cases of k-coloring and H-coloring problems [180].) Thus to further simplify the problem, several subclasses of perfect graphs including chordal, cographs and split graphs were considered. We shall discuss these classes in the following sections.

2.8.2 Chordal Graphs

A graph is chordal [104] if it has no chordless cycle of order larger than three. Chordal graphs are a well-known subclass of perfect graphs (see [164]). Chordal graphs have a very tractable structure and can be recognized in linear time (see [256, 170, 146]). Using these structural properties, Gavril [152] developed efficient algorithms to solve some graph problems for chordal graphs including several M-partition problems such as the k-coloring and the l-clique-covering problems. Additionally, the complexities of several other M-partition problems are known when restricted to chordal graphs. The H-coloring problem can be solved in polynomial time for chordal graphs (using the same characterization property as in the case of perfect graphs). Also, the following characterization result is proved for the (k, l)-graph recognition problem (another special case of the M-partition problem, see Section 2.5): a chordal graph is a (k, l)-graph if and only if it does not contain any induced subgraph isomorphic to a specific graph H, which consists of the disjoint union of $k + 1$ copies of the complete graph on $l + 1$ vertices [184]. Note that since H is a fixed graph, we can check in polynomial time whether a given graph G contains an induced subgraph isomorphic to H or not (by iterating over all order $|V(H)|$ subsets of $V(G)$ and check whether it is isomorphic to H or not). This implies a polynomial-time algorithm to solve the (k, l)-graph recognition problem for chordal graphs. All these results inspired Feder et al. to initiate the study of the M-partition problem for chordal graphs [131], hoping that solving the complexity (and the characterization) problem could be doable for these graphs. However, they failed to find a complete answer, and rather offered several partial dichotomy and characterization results (see Table 1.2). This suggested the difficulty of these problems even when restricted to chordal graphs. Thus the study of other subclasses of perfect graphs or chordal graphs was taken up as we will see in the subsequent sections.

Chordal graphs contain several important subclasses, including interval graphs (see Section 3.1.1) and split graphs (see Section 2.5). Indeed, one early motivation to study chordal graphs was due to interval graphs [152]. The study of the M-partition problem for interval graphs is handled in Chapter 5 of this thesis. As for split graphs, they are exactly those chordal graphs whose complement graphs are also chordal. Additionally, as the size of the graph goes to infinity, the fraction of the chordal but non split graphs goes to zero [17]. In other words, almost all chordal graphs are split graphs. The study of the M-partition problem for split graphs is already carried out in [134] (see Section 2.8.4). We refer to [88, 34] for a more comprehensive list of other important subclasses of chordal graphs.

2.8.3 Cographs

Another well-known subclass of perfect graphs, beside chordal graphs, is the class of cographs [123, 62, 63, 164, 261]. These graphs have been discovered independently by several authors in 1970s, including Jung [205], Lerchs [221], Seinsche [261], and Sumner [265] under different titles.

A *cograph* is a graph that can be generated from a single vertex by complementation and disjoint union. More precisely, we can construct cographs using these rules: 1) any graph with a single vertex is a cograph, 2) if G is a cograph, so is its complement $\overline{G}$, and 3) if G_1 and G_2 are cographs, so is their disjoint union $G = G_1 + G_2$ (i.e., $V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$). Additionally, there are several other ways to define this graph class. For example, cographs are equivalent to P_4-free graphs (i.e., graphs which do not have a chordless path of order four) [261]. See [91] for a more comprehensive list of other alternative definitions. Cographs have a very tractable structure (see [62, 63, 171, 158]), which allows many standard graph problems including the k-coloring and the l-clique-covering problems to be solvable in linear time (see [91]). This fact motivated Feder et al. [123] to study the more general case of the coloring problems, namely the M-partition problem, for cographs.

Unlike chordal graphs, cographs happened to be very tractable when it comes to the M-partition problem. It is proved that all 01-diagonal matrices M have finitely many cograph minimal obstructions [81, 123]. This implies that all M-partition problems can be solved in polynomial time for cographs. We already mentioned in Section 1.3 that cographs are exactly those graph which have clique-width at most two, and a linear algorithm is known to generate a 2-expression constructing any cograph [69]. Thus applying Corollary 1.3.3 yields that the list M-partition problem is linear for all matrices M. The same complexity result is also proved in [123], where a linear time algorithm is presented.

2.8.4 Split Graphs

As we mentioned earlier, split graphs form an important subclass of chordal graphs (and hence perfect graphs). We already introduced these graphs and mentioned their importance (see Section 2.5). As a subclass of chordal graphs, studying the M-partition problem for split graphs may help solve the M-partition problem for chordal graphs. Such an attempt was carried out in a recent study of [134]. It is proved that all 01-diagonal matrices M have finitely many split minimal obstructions. Thus all M-partition problems can be solved in polynomial time for split graphs. However, for the list version, only some NP-complete cases are introduced. This suggests that the difficulty of obtaining a complete dichotomy result for the list version still persists in this subclass.

Chapter 3

The Graph Classes Considered in this Thesis

In Section 2.8 we gave the history of the M-partition problem for some special graph classes. As we mentioned in Section 1.1, in this thesis we study the (list) M-partition problem for some other graph classes shown in Table 1.3. For this reason, in this chapter we introduce these graphs classes and explain why studying the M-partition problem for these graph classes is relevant.

As a general idea, restricting the classic graph problems (e.g., the k-coloring problem, the maximum stable set problem, etc) to special graph classes has a long history in graph theory. Many of these classic problems are NP-complete for general graphs. This motivated many authors to impose certain restrictions on the input graph to make it possible to solve these problems in polynomial time (see [34]). The main sources of inspiration was the class of cographs and perfect graphs (see for example [7, 144]). We already mentioned these graph classes in Section 2.8. Recall that several standard graph problems such as the k-coloring, the l-clique-covering, the maximum clique and the maximum stable set problems are all polynomial for perfect graphs (while they are NP-complete for general graphs). For cographs, these problems can be solved even more efficiently, sometimes in linear time. This caused much interest for finding more graph classes similar to cographs and perfect graphs. Following this line of research, many graph classes were proposed and different graph problems (mainly the maximum stable set and the minimum coloring problems) were studied for them. Describing all such classes and graph problems will be beyond the scope of this thesis. We refer to the website [97] and the survey [34] for more comprehensive references. Instead, in this chapter we introduce the graph classes which we consider in this thesis (i.e., those in Table 1.3).

Our interest in these graph classes is based on a general trend in the literature of the M-partition problem, in which knowing the complexity of several special cases of the (list) M-partition problem restricted to a certain graph class is taken as a suggestion that solving the dichotomy problem for the (list) M-partition problem (for general matrix M) restricted to this graph class could be doable. Based on this criterion, for each graph class that we introduce in this chapter, a brief history for the existing results (particularly complexity results) related to special cases of the (list) M-partition problem restricted to that graph class will be given (to justify our interest for this class). As an additional justification, recall that in Section 2.1 we mentioned that the k-coloring problem (as a special case of the M-partition problem) is studied for virtually all graph classes (see [97]). This implies that studying the M-partition problem for any graph class can be seen as an extension of the current results regarding the k-coloring problem for this graph class. In some cases, existing results for other special cases of the M-partition problem (e.g., the stable cutset problem for line graphs, see Section 3.2) may provide additional justification for studying the M-partition problem for these graph classes.

Before proceeding further, we should mention one important fact. Recall from Section 1.3 that the list M-partition problem is polynomial for any class $\widetilde{G}$ of graphs with bounded clique-width k, for which we can efficiently find a k-expression constructing each graph in $\widetilde{G}$. Note that except cographs, all other graph classes studied so far for the M-partition problem (namely perfect, chordal and split graphs) have unbounded clique-width (see [97] for the relevant references). We will also see in this chapter that all the graph classes in Table 1.3 have unbounded clique-width too. Thus our results cannot be deduced from Corollary 1.3.3.

3.1 Intersection and the Containment Graphs of Geometric Objects

The *intersection graph* of a non-empty family $\widetilde{F}$ of sets is defined by assigning a vertex to each set in $\widetilde{F}$ and drawing an edge between two vertices if and only if their corresponding sets have non-empty intersection. Also, the *containment graph* of $\widetilde{F}$ is defined by assigning a vertex to each set in $\widetilde{F}$ and drawing an edge between two vertices if and only if one of the corresponding sets is a subset of the other.

In Section 5.1 we introduce tools to develop efficient algorithms for solving list M-partition restricted to graph classes which have certain conditions. We note that these conditions hold in several graph classes defined by geometric configurations, particularly the intersection and containment graphs of intervals on a line and arcs on a circle. Thus we dedicate Chapter 5 to these graph classes, namely interval graphs (intersection graphs of intervals on a line), circular arc graphs (intersection graphs of arcs on a circle), interval containment graphs (containment graphs of intervals on a line, equivalent to permutation graphs as we will see later) and circular arc containment graphs (containment graphs of arcs on a circle, equivalent to circular permutation graphs as we will see later). Additionally, in Chapter 5, we consider the bipartite version of these graph classes and some of their extensions, including comparability graphs (which are general containment graphs). In the following sub-sections, we give a brief introduction for each of these graphs classes, where we will see that all these classes are subclasses of perfect graphs, except for circular arc graphs. Thus our choice of these classes is consistent with the existing direction of studying the (list) M-partition problem restricted to perfect graphs.

Now we explain why studying the M-partition problem for these graphs classes is interesting. As mentioned in Section 2.8.2, interval graphs are one of the important subclasses of chordal graphs, and the difficulty of proving complete dichotomy for chordal graphs motivated the study of its subclasses. The class of split graphs, another important subclass of chordal graphs, is already studied in [134] (see Section 2.8). This leaves the case of interval graphs, which is posed as an open problem in the preliminary version of [180]. Circular arc graphs form a well-known superclass of interval graphs [95]. So studying the M-partition problem for circular arc graphs can be seen as a natural extension of our interest for interval graphs. As we will see in the following sub-sections, the k-coloring problem (as a special case of the M-partition problem) is studied for all these graph classes and efficient algorithms are known for solving this problem. In addition to the k-coloring problem, Enright et al. [112] recently proved that the list H-coloring problem (another special case of the list M-partition problem which generalizes the k-coloring problem) is polynomial for interval and permutation graphs. These results further motivate our study of the M-partition problem for these graph classes which is carried out in Chapter 5.

3.1.1 Interval Graphs

An *interval graph* is the intersection graph of a set of intervals on a fixed line. The concept of interval graph was developed to model some real world situations in different areas of science. Cohen [60] initially used interval graphs in population biology models. Later on, these graphs were used for other purposes such as modeling DNA mappings in genetics [282], resource allocation in operations research and scheduling theory [13] and temporal reasoning [167].

We already mentioned that interval graphs form a subclass of chordal graphs, and chordal graphs form a subclass of perfect graphs. As a characterization, interval graphs are chordal graphs whose complements are comparability graphs (see Section 3.1.3) [140, 164]. Other well-known generalizations of interval graphs are circular arc graphs and trapezoid graphs [95].

Booth et al. [26] introduced a linear time algorithm to recognize interval graphs using a complex data structure called PQ tree. A simpler algorithm is introduced by Habib et al. [26] using lexicographic breadth-first search and the fact that interval graphs are exactly the chordal graphs whose complements are comparability graphs (as mentioned earlier). For interval graphs, as a subclass of chordal graphs, linear time algorithm is known to solve several standard graph problems including k-coloring and l-clique-covering (as special cases of the M-partition problem). The clique-width of interval graphs is known to be unbounded [165].

3.1.2 Circular Arc Graphs

A *circular arc graph* is the intersection graph of a set of arcs on a fixed circle. Circular arc graphs are first studied in 1960s by Hadwiger et al. [172] and Klee [211]. These graphs later gained importance due to their structural properties and similarity to interval graphs, as well as their applications in modeling many real-world systems such as genetics, traffic control, resource allocation problems in operations research and many others. (Refer to [225] for a list of references for these applications.)

Circular arc graphs form a natural extension of the class of interval graphs [95]. However, not all circular arc graphs are perfect, for example all odd cycles and their complements (which are minimal non-perfect graphs) are circular arc graphs. In general, solving graph problems for circular arc graphs is more difficult than for interval graphs, partially because interval graphs satisfy the Helly property (i.e., a set of pairwise intersecting intervals share a common point), while circular arc graphs in general do not satisfy this property [225].

The first polynomial-time algorithm to recognize circular arc graphs was introduced by Tucker [268]. Later on, a linear time algorithm (of order $O(|V(G)| + |E(G)|)$) was discovered by McConnell [235]. The k-coloring problem is proved to be polynomial for circular arc graphs, while the vertex coloring problem (where k is part of the input) is NP-complete [151]. Other standard graph problems including l-clique-covering are proved to be polynomial for these graphs in [153]. More efficient algorithms to solve these problems are introduced in [169]. The clique-width of circular arc graphs is known to be unbounded, as they contain interval graphs.

3.1.3 Comparability Graphs

A graph G is a *comparability graph* if there exists a partial ordering $<$ on $V(G)$ such that vertices $u, v \in V(G)$ are adjacent if and only if $u < v$ or $v < u$. By orienting each any edge $uv \in E(G)$ from u to v whenever $u < v$, we obtained a digraph with this property that, if $u \to v$ (meaning there is an arc oriented from u to v) and $v \to w$ then $u \to w$ (for all vertices $u, v, w \in V(G)$). Any orientation (of the edges of G) with this property is called a *transitive orientation*. This offers another alternative definition for comparability graphs: a graph is comparability if and only if it admits a transitive orientation of its edges.

It is easy to see that the containment graph of any set family is a comparability graph. The reverse is also true: any comparability graph is a containment graph of some set family. To see this, assign to each vertex v of a comparability graph the set of all vertices u for which $u \to v$ (using the transitive orientation). Thus comparability graphs and containment graphs are the same (see [271]). In other words, comparability graphs form the extension of any graph class consisting of containment graphs, e.g., the class of permutation graphs, the class of circular containment graphs, etc. Other notable subclasses of comparability graphs include complete graphs, bipartite graphs and the complements of interval graphs. Comparability graphs form a subclass of perfect graphs [164]. This means any containment graph is perfect.

Several polynomial-time algorithms have been introduced to recognize comparability graphs. Golumbic [164] offered a $O(\Delta \cdot |E(G)|)$ time algorithm (where Δ denotes the largest degree of a vertex in the input graph G), and Spinrad [267] introduced a $O(n^2)$ time algorithm for this purpose.

Many standard graph problems are shown to be polynomial when restricted to comparability graphs. These problems include special cases of the M-partition problem such as the k-coloring and the l-clique-covering problems (see [195, 163]). A transitive orientation of a comparability graph G can be found in linear time [34, 239]. An ordering $v_1, v_2, \cdots v_n$ of $V(G)$ is called a *topological ordering* if for no two indiced $1 \leq i < j \leq n$ we have $v_j \to v_i$. It is shown that running the greedy coloring algorithm over the topological ordering yields an optimal coloring (in polynomial time) [230]. This implies that comparability graphs are contained in a well-known subclass of perfect graphs called perfectly orderable graphs (see [98]). However, note that the list k-coloring problem is NP-complete for the class of comparability graphs (when $k \geq 3$), as it contains bipartite graphs. The clique-width of comparability graphs is known to be unbounded, since they contain permutation graphs as subclass (see Sub-section 3.1.4).

3.1.4 Interval Containment (Permutation) Graphs

An *interval containment graph* is the containment graph of a set of intervals on a fixed line. A *permutation graph* is the intersection graph of line segments between two fixed parallel lines in the plane. It is known that any interval containment graph is a permutation graph and vice versa [106].

Thus, permutation graphs are both intersection and containment graphs. For this reason, they form a subclass of comparability graphs, and thus are perfect graphs. As a characterization, permutation graphs are exactly those comparability graphs whose complement graphs are also comparability graphs [106]. Using this property, a polynomial-time algorithm to recognize permutation graphs can be developed (by applying a comparability graph recognition algorithm to the input graph and its complement). Permutation graphs contain cographs (defined in Section 2.8) as subclass [99].

Permutation graphs were introduced in [114, 247], where their useful characterizations and many applications are shown. In fact, they form an important subclass of perfect graphs, as many important graph problems become polynomial when restricted to permutation graphs [33]. These problems include the standard problems of k-coloring and l-clique-covering (as special cases of the M-partition problem). We refer to [33] for a comprehensive list of references on these problems, and to [164] for a survey of permutation graphs properties. The clique-width of permutation graphs is known to be unbounded [165].

3.1.5 Circular Arc Containment (Circular Permutation) Graphs

A *circular arc containment graph* is the containment graph of a set of arcs on a fixed circle. A *circular permutation graph* is the intersection graph of of paths between two fixed concentric circles in the plane such that no two paths intersect more than once [93]. It is known that any circular arc containment graph is a circular permutation graph and vice versa [257]. Circular arc containment graphs are a natural generalization of interval containment (permutation) graphs introduced in the previous sub-section. Thus, as in the case of permutation graphs, circular permutation graphs are both intersection and containment graphs. So, they form a subclass of comparability graphs, and thus are perfect graphs.

Circular permutation graphs, which generalize the well-studied class of permutation graphs, are first introduced and studied in [257], where a $O(\Delta \cdot |E(G)|)$ time algorithm was developed to recognize them. Discovering a new characterization of circular permutation graphs in [263] led to a more efficient recognition algorithm in $O(|V(G)| + |E(G)|)$ time.

Many standard graph problems, including k-coloring, l-clique-covering are studied for circular permutation graphs and efficient algorithms are developed (see [245]). The clique-width of circular permutation graphs is known to be unbounded, as they contain permutation graphs.

3.1.6 Interval and Interval Containment Bigraphs

Both interval graphs and interval containment graphs have bipartite versions. A graph G is an *interval bigraph* (*interval containment bigraph*, respectively) if it is a bipartite graph with parts X and Y such that the adjacency between the vertices of X and Y follows the rule of intersection graphs (containment graphs, respectively) of intervals, i.e., a vertex $x \in X$ is adjacent to a vertex $y \in Y$ if and only if the corresponding interval of x intersects with (contains or is contained in, respectively) the corresponding interval of y.

Interval bigraphs are first introduced and studied in [174]. An equivalent definition of these graphs, using digraphs, is presented in [83]. Müller [241] introduced a polynomial recognition algorithm for interval bigraphs and made a conjecture for characterization of these graphs in terms of forbidden subgraphs. This conjecture was proved in a special case in which the interval bigraphs are trees [42]. Further characterizations of interval bigraphs were found in [183], where it is shown that the complement of interval bigraphs are exactly those circular arc graphs which have the clique-covering number ≤ 2 and contain no two arcs covering the whole circle. The clique-width of interval bigraphs is known to be unbounded [35]. We refer to [34, 42, 41, 40, 224, 259, 83, 276, 182] for more characterizations of interval bigraphs.

The class of interval containment bigraphs contains interval bigraphs as subclass, and is contained in the class of comparability graphs [94]. It is conjectured that these two classes (namely the class of interval containment bigraphs and the class of interval bigraphs) are possibly equal (see [94]). The clique-width of interval containment bigraphs is known to be unbounded, as they contain interval bigraphs. Note that any bipartite graph is a perfect graph, and thus both graph classes are subclasses of perfect graphs.

3.2 Line, Quasi-line and Claw-free Graphs

In Chapter 6 we will study the M-partition problem for line graphs and its extension to quasi-line and claw-free graphs. In this section we give the definitions and some background for these graph classes.

Given a graph G, the *line graph* of G, denoted by $L(G)$, is defined as a graph whose vertices correspond to the edges of G and two distinct vertices are adjacent if and only if their corresponding edges share one endpoint in G. Throughout this thesis, by *line graph* we mean the line graph of some graph. Line graphs were used as early as 1932 in several works including [279] and [217] (see [193]) under different names. The name "line graph" first appeared in 1960 in [175].

Several works [16, 217, 272] studied the structural properties of line graphs including the characterization using nine forbidden subgraphs (see [16] for all such results). These results became the basis for a linear recognition algorithm for line graphs [258].

Many standard graph problems have been studied for line graphs (see [96] for a comprehensive list of these problems and their results). As far as the M-partition problem is concerned, the classic problem of 3-coloring is shown to be NP-complete for line graphs [197]. Another M-partition problem, namely the stable cutset problem (defined in Section 2.4) is studied in [30] for line graphs and is proved to be NP-complete. Additionally, from M-partition's point of view, restricting the input graph to line graphs is equivalent to M-partition the edges of general graphs. In other words, it leads to the edge version of the M-partition problem, which increases our interest for studying this class.

A graph is *claw-free* if it does not contain any induced subgraph isomorphic to a certain graph called *claw*, which is composed of three independent vertices fully adjacent to a fourth vertex (i.e., a complete bipartite graph in which one part has one vertex and the other part has three vertices). Note that line graphs are claw-free. This simple observation formed the initial motivation for studying claw-free graphs as an interesting generalization of line graphs [54, 117]. Later on, various properties of claw-free graphs were discovered (see [108, 109, 117]), which drew much attention to these graphs. Our study of the M-partition problem for claw-free graphs can be seen as an extension of our study for line graphs. We refer to [117] for a comprehensive survey of important results for claw-free graphs.

We already mentioned in Section 2.4 that the Strong Perfect Graph conjecture was settled by Chudnovsky et al. in [53]. Their proof is based on many decomposition techniques. Applying similar techniques to claw-free graphs led to a complete structural characterization of these graphs by Chudnovsky et al. in [54]. Unfortunately, their structural theorem is long and complicated for general claw-free graphs. For this reason, they introduced an intermediate graph class between the class of line and the class of claw-free graphs (i.e., a class that contains the former class and is contained in the latter class), which is called *quasi-line* graphs. They are defined as the graphs in which the set of neighborhood of each vertex can be partitioned into two cliques. For quasi-line graphs, the structural theorem becomes simpler and more digestible. So we included quasi-line graphs along with line and claw-free graphs in our study. Using this structural theorem, we derive some dichotomy result for quasi-line graphs in Section 6.2 (Theorem 6.2.5).

3.3 H-free Graphs

Given a family $\widetilde{H}$ of graphs, a graph G is said to be $\widetilde{H}$*-free* if it contains no member of $\widetilde{H}$ as induced subgraph. When $\widetilde{H}$ has only one member H, we may use the notation H-*free* instead of $\widetilde{H}$-free. To avoid confusion, we assume that all the graphs in this section are without loops. More precisely, we assume that all the graphs in the family $\widetilde{H}$ are without loops, and we only consider the $\widetilde{H}$-free graphs which are without loops. One early motivation of studying $\widetilde{H}$-free graphs stemmed from the class of cographs (see [29, 144]). As we mentioned earlier in Section 2.8.3, efficient (and even linear) algorithms have been developed to solve many standard graph problems when restricted to cographs. Recall that cographs are the same as P_4-free graphs. (Recall that P_4 is the path with four vertices, see Section 2.8.) This fact made it interesting to consider the class of H-free graphs when H is a *one-vertex extension of* P_4, i.e., a graph with five vertices which contains P_4 as induced subgraph [29, 36]. All such graphs are shown in Figure 3.1, along with the titles used for them in the literature. More generally, the class of H-free graphs and $\widetilde{H}$-free graphs were also considered in the literature, when H and $\widetilde{H}$ are a general graph and a general graph family, respectively [228, 215]. Studying the graph problems for H-free graphs, both for general graphs H and one-vertex extensions of P_4, were subject to much study targeting several standard graph problems. We now give a brief overview for each direction with the main focus on the coloring and other graph problems related to the M-partition problem.

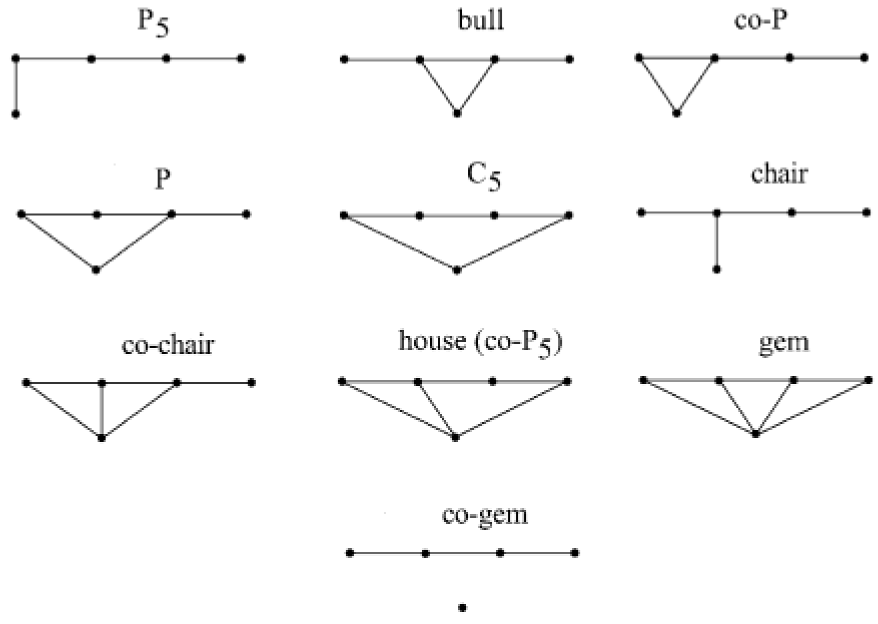

Figure 3.1: One vertex extensions of P_4.

For any graph H, Král et al. [215] determined the complexity of the vertex coloring problem restricted to H-free graphs. They show that it is polynomial when H is an induced subgraph of P_4 or $P_1 + P_3$ (i.e., disjoint union of a vertex and the path P_3 on three vertices), and NP-complete otherwise. Recall that in the vertex coloring problem we have a graph G and an integer $k > 0$ as input, and our aim is to decide whether G is k-colorable or not. Thus, the vertex coloring problem cannot be modeled as an M-partition problem in a straightforward way as the k-coloring problem.

This result inspired two directions: first, studying the vertex coloring problem for $\widetilde{H}$-free graphs, where $\widetilde{H}$ is a family containing more than one graph [162, 160], and second, studying the k-coloring problem for H-free graphs. Some partial results are obtained for the first question. We refer the reader to [249, 162, 160] for a complete list of known results for this question.

Now we focus on the second problem, namely studying the k-coloring problem (which is a special case of the M-partition problem) for H-free graphs. Note that our complexity results for the M-partition problem restricted to H-free graphs in Section 7.1 (Theorems 7.1.1, 7.1.2, and Corollary 7.1.4) can be seen as along this direction. In contrast to the vertex coloring problem, analyzing the k-coloring problem for H-free graphs is an open problem [160, 71]. The k-coloring problem is easily solved in polynomial time when $k \leq 2$, as it is equivalent to deciding if the input graph is bipartite. So from now on we assume $k \geq 3$. For any fixed constant $g \geq 3$, the 3-coloring problem is known to be NP-complete for the class of all graphs whose girth is larger than g [206]. This result can be extended easily to the k-coloring problem as well (assuming $k \geq 3$). Note that if H contains a chordless cycle of order $g \geq 3$ then any graph with girth larger than g is H-free. This implies that the k-coloring problem is NP-complete for H-free graphs when H contains a cycle. This brings down our task to studying H-free graphs when H has no cycle, namely when H is a forest. In this case we use the notation F instead of H, to indicate its being a forest.

The problem of k-coloring for F-free graphs has been studied in many papers [79, 196, 206, 207, 215, 216, 220, 251, 270, 280]. We mentioned in Section 3.2 that the 3-coloring problem is proved to be NP-complete for claw-free graphs. The implication of this result is that if F contains a vertex with degree at least three then F-free graphs will contain claw-free graphs as subclass, and thus the k-coloring problem becomes NP-complete for F-free graphs. This further brings down our task to studying forests F which have no vertex of degree ≥ 3. It is easy to see that this condition implies that F is a union of disjoint paths, which is called a *linear forest*.

In [38], the 3-coloring problem is proved to be polynomial for F-free graphs when F is a linear forest with at most six vertices, except when $F = 2P_3$ (i.e., the disjoint union of two paths on three vertices). This last case is later proved to be also polynomial in [39]. These results yield a complete dichotomy in the case that H has at most six vertices as follows: the 3-coloring problem for H-free graphs is polynomial if H is a linear forest, and otherwise it is NP-complete. A similar complexity result is proved in [162] for the 4-coloring problem and graphs H which have at most five vertices. Some other special types of larger linear forest are also considered. We refer to [72] (Theorem 3) for an extensive list of linear forests for which the complexity of the k-coloring problem (for certain values of $k \geq 3$) is known. Denote by P_t the path with $t > 0$ vertices. Note that P_t (for any $t > 0$) is a special linear forest. Several graph problems have been studied for P_t-free graphs, particularly determining the complexity of the k-coloring problem (and some of its variations including the list and the retraction versions) for P_t-free graphs has been the subject of extensive investigations, and it is known for many pairs of k and t (see [71] for a comprehensive list of such results). Other pairs of k and t are still under investigation (see [199] for a very recent result). Our study of the list M-partition problem for P_5-free graphs (in Section 7.2) can be seen as an extension of these works. We refer to [250] for a survey of the 3-coloring problem when restricted to $\widetilde{H}$-free graphs.

Now we focus on the case in which the set of forbidden subgraphs (i.e., the set $\widetilde{H}$) consists of the one-vertex extensions of P_4 (shown in Figure 3.1). Brandstädt et al. [29], along the line of several previous related works, classified the clique-width of many such graph classes into bounded and unbounded. Using this classification, we may conclude that all list M-partition problems can be solved in polynomial time for those graph classes which are proved to have bounded clique-width and to have an efficient algorithm to generate the k-expression for every graph belonging to them (see Corollary 1.3.3). For other classes, much effort has been made to determine the complexity of several graph problems including the k-coloring problem (see [228, 36, 156, 24, 154]). In this thesis we focus on two particular one-vertex extensions of P_4, namely P_5 and the bull (as shown in Figure 3.1). Now we give some background for each class.

The class of P_5-free graphs and its subclasses have been the subject of much study in the last two decades (see [196, 7, 199]). The subclasses considered so far are $\widetilde{H}$-free graphs where $P_5 \in \widetilde{H}$. In most cases, $\widetilde{H}$ is a finite set consisting of some one-vertex extensions of P_4. Fouquet et al. [145] characterized the structure of $\{P_5, \overline{P_5}\}$-free graphs (i.e., the P_5-free graphs whose complements are also P_5-free), and offered a polynomial-time algorithm to solve the vertex coloring problem restricted to these graphs. The k-coloring problem for P_5-free graphs has been studied by several authors as part of the more general direction of studying the k-coloring problem for P_t-free graphs (as mentioned earlier). Finally, Hoàng et al. [196] provided the first polynomial-time algorithm to this problem (for any $k > 0$). Our dichotomy result of the list M-partition problem for P_5-free and $\{P_5, \overline{P_5}\}$-free graphs (Theorems 7.2.4 and 7.2.5) can be seen as extension of these results. Many other subclasses of P_5-free graphs have been studied for different graph problems, e.g., the vertex coloring and clique-covering problems for $\{P_5, \overline{P_5}, fork\}$-free graphs [144], and $\{P_5, diamond\}$-free graphs [7].

As for bull-free graphs, note that the bull contains a triangle (i.e., a clique of size 3). Thus bull-free graphs contain the class of triangle-free graphs as subclass, for which the 3-coloring problem is known to be NP-complete [231]. This implies that the k-coloring problem for bull-free graphs is NP-complete ($k \geq 3$). In addition to the motivations we stated earlier (that the bull is a one-vertex extension of P_4), bull-free graphs received much attention due to their connection with perfect graphs (see [51]). Recall that in Section 2.8 we mentioned that finding a polynomial time combinatorial algorithm to solve the k-coloring problem for perfect graphs is an open problem, and bull-free perfect graphs are one of the subclasses (of perfect graphs) for which some success was made in this direction [86, 87, 222]. The structure of bull-free graphs is studied and completely characterized in [51].

Chapter 4

Tools

In this chapter we introduce some general definitions, assumptions and tools which will be used frequently throughout this thesis. After introducing some general notation in Section 4.1, we introduce two main tools in Section 4.2, which include breaking an instance of the (list) M-partition problem and applying the separation property. These tools allow us to reduce a list M-partition problem into smaller sub-problems which have simpler structures than the original one in terms of the matrix and/or the input graph. In the next section we provide some techniques which allow us to apply these tools. The last section is dedicated to a very useful auxiliary result related to classifying the complexity of the list M-partition problem for 0-diagonal matrices M. This result will be used again and again throughout this thesis.

4.1 Definitions and Assumptions

To prove several NP-complete results in this thesis, we make use of a problem known as *not-all-equal 3-satisfiability*, denoted by *NAE 3-SAT*. We only consider the NAE 3-SAT problem without negated variables ([150]), defined as follows: an instance of this problem consists of variables $x_1, x_2, \cdots x_n$ and clauses $Cl_1, Cl_2, \cdots Cl_r$. Each clause is a set consisting of three distinct variables called the variables or *literals* of that clause. The problem asks whether there is an assignment of TRUE or FALSE to each variable such that each clause does not have the same value assigned to all its literals. This problem is known to be NP-complete [150]. An M-partition $V_0, V_1, \cdots V_{m-1}$ of a graph G corresponds to a coloring f of G defined as $f(v) = i$ if $v \in V_i$. Based on this, we call each index $i \in [m]$ a *color*. We typically identify an M-partition with its corresponding coloring functions. For example, the statement that "G has an M-partition f" means: "G has an M-partition whose corresponding coloring function is f." Using this coloring representation, we identify the i-th row (and column) of the matrix M with the color i ($i \in [m]$). As an example, we may say "the color i in the matrix M," which refers to the i-th row (or column) of M. Also, by saying that a vertex *u is colored with i* (in some M-partition), we mean that u is placed in the part V_i (in that M-partition). A color c is called *0-color* if it has entry 0 on the main diagonal in M (i.e., $M(c, c) = 0$). Similarly, we define the notations of *1-color* and **-color*.

Given a matrix M, a *list function* of a graph G is a function $L : V(G) \rightarrow 2^{[m]}$ which assigns a list $L(u) \subseteq [m]$ (of the colors of M) to each vertex $u \in V(G)$. Recall that we defined the list M-partition problem in Section 1.1. An *instance* $I = (G, L)$ of the list M-partition problem consists of a graph G and a list function L of G. A *solution of I* is any M-partition f of G which *respects the list* L, i.e., $f(u) \in L(u)$ for each $u \in V(G)$. Again, we would identify any solution of I with its corresponding coloring f (e.g., we would say: "a solution f of I ..."). An instance $I' = (G', L')$ is called a *sub-instance* of $I = (G, L)$ (both using the same matrix) when G' is an induced subgraph of G and $L'(v) \subseteq L(v)$ for any $v \in V(G')$.

When no matrix is mentioned, by default we assume I is an instance of the list M-partition problem, where M should be clear from the context. We always denote by m the size of the matrix M. Since the case of $m = 1$ is trivial, throughout this thesis we always assume that $m \geq 2$. In some cases, we may construct instances with some lists being empty (i.e., $L(v) = \emptyset$ for some $v \in V(G)$). We call such instance *invalid*, and they should be treated as instances which have no solution (since no coloring option exists for those vertices having empty list). To express the running time of any algorithm whose input consists of a graph G or an instance $I = (G, L)$, the parameter n always denotes the number of vertices of G. By saying that the running time, or complexity, of an algorithm is polynomial, we mean that the running time is polynomial in n.

In an instance I, we say that a color $c_1 \in L(u)$ *conflicts* with a color $c_2 \in L(v)$ if coloring u and v with the colors c_1 and c_2, respectively, violates the definition of the M-partition problem, i.e., either $M(c_1, c_2) = 1$ and $uv \notin E(G)$, or $M(c_1, c_2) = 0$ and $uv \in E(G)$. A list $L(v)$ is said to be *singleton* if it contains only one color (i.e., $|L(v)| = 1$). Let V_0 be a subset of $V(G)$ consisting only of vertices with singleton list. A procedure called *updating the lists based on* V_0 acts as follows: for any vertex $v_0 \in V_0$ and $v \in V(G)$, any color in $L(v)$ which conflicts with the only color in $L(v_0)$ is eliminated. Let L' be the new list function (once all these eliminations have taken place). Note that this procedure neutralizes the effect of the set V_0 in the final solution of I. In other words, we can color each vertex $v \in V_0$ with the unique color in $L(v)$, and to color the rest of vertices we have to solve the sub-instance $(G[V(G) - V_0], L')$.

By *pre-coloring* a vertex $v \in V(G)$ with a color $c \in L(v)$ we mean replacing the list $L(v)$ with the singleton list $\{c\}$. This creates a new sub-instance of I which is different from I only for the list of the vertex v. For technical convenience, we also define the pre-coloring when $c \notin L(v)$. In this case, we replace the list $L(v)$ with the empty set (which leads to an invalid sub-instance). We can extend the definition of pre-coloring to any subset $V_0 \subseteq V(G)$ as follows: let f_0 be an M-partition of the subgraph $G[V_0]$. Then by *pre-coloring* V_0 *based on* f_0 we mean replacing each list $L(v)$ with the singleton list $\{f_0(v)\}$ or the empty set, depending on whether $f_0(v) \in L(v)$ or not (for all $v \in V_0$).

4.2 Breaking an Instance and the Separation Property

We say that an instance $I = (G, L)$ of the list M-partition problem *is broken* into sub-instances $I_J = (G, L_J)$ of I, for $J \in \widetilde{J}$ where $\widetilde{J}$ is some index set, if any solution of I is a solution of some sub-instance I_J. Note that since the sub-instances I_J use the same graph G, any solution of any sub-instance I_J is also a solution of I. For the sake of completeness, we define breaking I into *zero* instances (when $\widetilde{J} = \emptyset$) as equivalent to saying that I has no solution. By *breaking an instance* I into sub-instances $I_J = (G, L_J)$ of I, for $J \in \widetilde{J}$, we mean generating (using some algorithm) the sub-instance $I_J = (G, L_J)$ of I $(J \in \widetilde{J})$ into which I is broken.

Throughout this thesis, the following break-up scheme is used frequently: we find some polynomial-time algorithm to break an instance I into (polynomially many) sub-instances, and then we repeat this process for each of these sub-instances. We continue this repeated breaking until we arrive at sub-instances which are easier to solve. During this process, we will make sure that the total number of sub-instances remains polynomial. Similar techniques of repeatedly breaking an instance are used in [131, 196] as well.

Given a set $\widetilde{I}$ of instances of the list M-partition problem, by having *oracle access* for these instances we mean the assumption that we can solve any instance $I \in \widetilde{I}$ in $O(1)$ time. In this definition, solving means either returning a solution for I or declaring that no solution exists. Let M be an arbitrary 01-diagonal matrix. An instance $I = (G, L)$ of the list M-partition problem is said to be *homogeneous* if the set $\cup_{v \in V(G)} L(v)$ consists entirely either of 0-colors or of 1-colors. Let $\widetilde{G}$ be an arbitrary hereditary graph class. We say that a matrix M has the *separation property* for the class $\widetilde{G}$ if there exists a polynomial-time algorithm ALG to solve any instance $I = (G, L)$ of the list M-partition problem with $G \in \widetilde{G}$, provided that we have oracle access for any homogeneous sub-instance of I. The algorithm ALG is said to be *giving the separation property.*

Observation 4.2.1. *The separation property is hereditary, i.e., if M has the separation property for a hereditary class $\widetilde{G}$ then it has the separation property for any hereditary subclass $\widetilde{H}$ of $\widetilde{G}$. Furthermore, any algorithm giving the separation property for $\widetilde{G}$ also gives the separation property for $\widetilde{H}$.*

Proof. Let $I = (G, L)$ be an instance of the list M-partition problem for $\widetilde{H}$. Suppose we have oracle access for any homogeneous sub-instance of I. Note that I can be considered as an instance for the class $\widetilde{G}$ as well, and the fact that M has the separation property for $\widetilde{G}$ implies that I can be solved in polynomial time using the oracle access. □

A conventional application of the separation property used throughout this thesis is given in the following observation

Observation 4.2.2. *Let M be a 01-diagonal matrix with blocks A and B (see Figure 1.2). Suppose M has the separation property for a hereditary graph class $\widetilde{G}$. Assume the list A-partition and the list B-partition problems, restricted to $\widetilde{G}$, are both polynomial. Then the list M-partition problem, restricted to $\widetilde{G}$, is also polynomial. Moreover, suppose there exists an $O(n^d)$ time algorithm ALG_1 to solve the list A-partition or the list B-partition problems when restricted to $\widetilde{G}$, and an $O(n^t)$ time algorithm ALG_2 giving the separation property for $\widetilde{G}$. Then the list M-partition problem, restricted to $\widetilde{G}$, can be solved in $O(n^{td})$ time.*

Proof. Let $I = (G, L)$ be an instance of the list M-partition problem with $G \in \widetilde{G}$. Since M is 01-diagonal, any homogeneous sub-instance of I uses either only 0-colors or only 1-colors in its lists. Thus, such sub-instance is an instance of either the list A-partition problem or the list B-partition problem, and so it can be solved in $O(n^d)$ time using the algorithm ALG_1. Now by using ALG_1 instead of the oracles in the algorithm ALG_2, we get an algorithm (without oracles) to solve I in $O(n^{td})$ time. □

Thus, proving the separation property reduces the task of studying the list M-partition problem for 01-diagonal matrix M to studying the list A-partition problem and the list B-partition problem, which are supposed to be simpler. This is one of our the main approaches in this thesis, particularly in Chapter 7. In the next section we will provide some sufficient conditions for M to have the separation property.

Before closing this section, we have to mention that the running time analysis in Observation 4.2.2 is rather rough: the bound td in the observation can be replaced by remarkably smaller values. However, as we stated earlier (see Section 2.7), in this thesis the main focus is on polynomial vs. NP-complete complexities, and not optimizing the algorithms. Also for polynomial-time algorithms, our only purpose is to keep the degree (of the polynomial bound of the running time) bounded by a polynomial in terms of m. The running time analysis in Observation 4.2.2 is good enough for this purpose. A more accurate analysis of the running times and further optimizing the polynomial-time algorithms are left as future research direction.

4.3 Some Techniques to Prove the Separation Property

In an instance $I = (G, L)$, two distinct vertices $u, v \in V(G)$ are called *separated* if no color in $L(u)$ conflicts with any color in $L(v)$. We say that two disjoint non-empty subsets $U, V \subseteq V(G)$ are separated if any $u \in U$ and $v \in V$ are separated. Note that if we break I into sub-instances, and the sets U and V are separated in I, then they remain separated in each sub-instance as well.

The concept of *sparse-dense partition* is defined in [129] as follows: given two graph classes $\widetilde{S}$ and $\widetilde{D}$, we call $(\widetilde{S}, \widetilde{D})$ a *sparse-dense pair* with respect to a constant $d > 0$ if no two graphs $G \in \widetilde{S}$ and $H \in \widetilde{D}$ can share more than d vertices (i.e., there is no induced subgraph G' of G with $d + 1$ vertices which is isomorphic to an induced subgraph H' of H). In this case, we call the members of $\widetilde{S}$ and $\widetilde{D}$ sparse and dense graphs, respectively. Although this definition is symmetric, in most applications the first class (i.e., $\widetilde{S}$) consists of graphs much sparser than the second class (i.e., $\widetilde{D}$), and thus we call them sparse, while we call the members of the second family dense graphs. A *sparse-dense partition* (VA, VB) of a graph G is a partition of its vertices into sets VA and VB which induce sparse and dense subgraphs, respectively.

Theorem 4.3.1. *([129]) Given a sparse-dense pair of graph classes with respect to a constant $d > 0$, any graph G on n vertices has at most n^{2d} different sparse-dense partitions. Furthermore, all these partitions can be found in $O(n^{2d+2})$ time, provided that we have oracle access for deciding whether a subgraph of G is sparse or dense.*

Let M be an arbitrary 01-diagonal matrix. An instance $I = (G, L)$ of the list M-partition problem is called *list-homogeneous* if each list $L(v)$ consists entirely either of 0-colors or of 1-colors. In other words, the instance I is list-homogeneous if there is a partition (VA, VB) of $V(G)$ such that the instances $(G[VA], L)$ and $(G[VB], L)$ are both homogeneous instances. We call such partition the *homogeneous partition of I*. The following useful tool can be derived from Theorem 4.3.1:

Lemma 4.3.2. *Suppose M is a 01-diagonal matrix with blocks A and B of size k and l, respectively (see Figure 1.2). Then any instance of the list M-partition problem can be broken into n^{2kl} many list-homogeneous sub-instances. Furthermore, we can generate all these sub-instances in $O(n^{2kl+3})$ time, provided that we have oracle access for solving any homogeneous sub-instance of I.*

Proof. Let $I = (G, L)$ be an instance of the list M-partition problem. Define $\widetilde{S}$ and $\widetilde{D}$ as the set of A-partitionable and B-partitionable graphs, respectively. Note that any graph in $\widetilde{S}$ is k-colorable and any graph in $\widetilde{D}$ can be partitioned into l cliques. Thus no graph with more than kl vertices can be isomorphic to a member of both sets. This means $(\widetilde{S}, \widetilde{D})$ is a sparse-dense pair with respect to the constant kl. Thus by Theorem 4.3.1, we may conclude that we have at most n^{2kl} many sparse-dense partitions $P = (VA, VB)$ of G, which can be found in $O(n^{2kl+2})$ time. For each sparse-dense partition $P = (VA, VB)$ of G, obtain the sub-instance I_P (of I) by applying the following changes to I: remove any 1-color from the list of any vertex in VA, and remove any 0-color from the list of any vertex in VB. These changes can be performed in $O(n)$ time. It is easy to see that I_P is a list-homogeneous sub-instance, with (VA, VB) as its homogeneous partition. Now we claim that I is broken into the sub-instances I_P, for all sparse-dense partitions P of G. Let f be a solution of I. Define the set VA_f (VB_f, respectively) as the set of all vertices having a 0-color (1-color, respectively) in the solution f. Note that $P_f = (VA_f, VB_f)$ is a sparse-dense partition of the graph G, since $G[VA_A]$ and $G[VA_B]$ are A-partitionable and B-partitionable, respectively. This means f is also a solution of the sub-instance I_{P_f}, which satisfies the definition of breaking. $\square$

Now we use the above lemma to derive two important tools related to the separation property:

Theorem 4.3.3. *Suppose M is a 01-diagonal matrix in which the block C is either *-free or all *. Then M has the separation property for the class of all graphs. Moreover, there exists an algorithm giving the separation property with running time $O(n^{2kl+4})$, where k and l are the sizes of the blocks A and B of M, respectively.*

Proof. Let $I = (G, L)$ be an instance of the list M-partition problem. Suppose we have oracle access for any homogeneous sub-instance of I. By using this oracle access and applying Lemma 4.3.2, we can break I into list-homogeneous sub-instances in $O(n^{2kl+3})$ time. Now we show how to solve each of these sub-instances. Let $I' = (G, L')$ be one of these sub-instances and let (VA, VB) be its homogeneous partition. Suppose we can break I' in polynomial time into sub-instances in which the sets V_A and V_B are separated. Then solving each sub-instance $I'' = (G, L'')$ of these sub-instances will consist of solving the list M-partition problem for each set V_A and V_B independently. In other words, we shall solve the instances $(G[V_A], L'')$ and $(G[V_B], L'')$ independently. Note that these instances are both homogeneous sub-instances of I, and thus can be solved in $O(1)$ time using oracle access. So it is sufficient to show how to break each sub-instance I' in such a manner.

In the case of block C being entirely made of *, the sets VA and VB are already separated and we are done. The case of C being *-free can be handled by an approach similar to that of Corollary 7.1 in [129]. We explain the approach in full detail here. Let $VA_1, VA_2, \cdots VA_{k'}$ ($VB_1, VB_2, \cdots VB_{l'}$, respectively) be a partition of VA (VB, respectively) into non-empty classes such that two vertices belong to the same class if and only if they have the same set of neighbourhood in VB (VA, respectively). Let us call these classes the neighborhood classes of VA (VB, respectively). An important property of these partitions is that for any pair $(i, j) \in \{1, 2, \cdots k'\} \times \{1, 2, \cdots l'\}$, the classes VA_i and VB_j are either fully adjacent (i.e., any two vertices $u \in VA_i$ and $u \in VB_j$ are adjacent) or fully non-adjacent (i.e., any two vertices $u \in VA_i$ and $u \in VB_j$ are non-adjacent). Now we define several parameters for any solution f of I' as follows: define SA_f (SB_f, respectively) as a function in which for $i = 1, 2, \cdots k'$ ($j = 1, 2, \cdots l'$, respectively), $SA_f(i)$ ($SB_f(j)$, respectively) is the set of colors used by f for at least one vertex in the class VA_i (VB_j, respectively). Let J_f be the pair (SA_f, SB_f) of functions. The fact that the block C is *-free implies that the sets $SA_f(i)$, $i = 1, 2, \cdots k'$, are pairwise disjoint. To prove this, suppose to the contrary, a color c

belongs to two sets $SA_f(i)$ and $SA_f(j)$ for some indices $1 \leq i \neq j \leq k'$. Let $u \in SA_f(i)$ and $v \in SA_f(j)$ be vertices colored with c (by f). Since u and v belong to different neighborhood classes, there must be vertex $w \in VB$ which is adjacent to exactly one of the vertices u and v. This implies that the entry $M(c, f(w))$ is * which is a contradiction, as this entry is in the block C. A similar argument shows the sets $SB_f(j)$, $j = 1, 2, \cdots l'$, are also pairwise disjoint. Another important property is that, for any pair $(i, j) \in \{1, 2, \cdots k'\} \times \{1, 2, \cdots l'\}$ of indices and any color $c \in SA_f(i)$ and $c' \in SB_f(j)$, the entry $M(c, c')$ is 1 or 0 depending on whether the class VA_i is fully adjacent or fully non-adjacent to the class VB_j. So let $\widetilde{J}$ be the set of all pairs (SA, SB) in which:

1. $SA : \{1, 2, \cdots k'\} \to 2^{[m]}$ ($SB : \{1, 2, \cdots l'\} \to 2^{[m]}$, respectively) is a function in which sets $SA(i)$, $i = 1, 2, \cdots k'$, ($SB(j)$, $j = 1, 2, \cdots l'$, resp) are pairwise disjoint and non-empty,

2. For any pair $(i, j) \in \{1, 2, \cdots k'\} \times \{1, 2, \cdots l'\}$ of indices and any color $c \in SA(i)$ and $c' \in SB(j)$, the entry $M(c, c')$ is 1 or 0 depending on whether the class VA_i is fully adjacent or fully non-adjacent to the class VB_j.

Note that $\widetilde{J}$ contains the pairs J_f for all possible solutions f of I (However, it may contain tuples not corresponding to any solution.) If $k' > k$ or $l' > l$ then $\widetilde{J} = \emptyset$. Otherwise, for each 0-color (1-color, respectively) we have $k' + 1$ ($l' + 1$, respectively) choices to be placed in one of the classes VA_i (VB_j, respectively) or none of them. This implies that we have at most $(k' + 1)^k \cdot (l' + 1)^l = O(1)$ many choices for the pairs in $\widetilde{J}$. So, the total number of elements in $\widetilde{J}$ is $O(1)$.

For each pair $J = (SA, SB) \in \widetilde{J}$, define I'_J as a sub-instance of I' which is obtained from I' by removing from the list of any vertex in VA_i (VB_j, respectively) any color which is not in $SA(i)$, for $i = 1, 2, \cdots k'$ ($SB(j)$ $j = 1, 2, \cdots l'$, respectively). Due to the properties of the functions SA and SB, it is easy to see that in each sub-instance I'_J, the sets VA and VB are separated. Also, it is easy to see that any solution f of I' is a solution of the sub-instance I'_{J_f} as well. This implies that I' is broken into sub-instances I'_J, for $J \in \widetilde{J}$. Moreover, we can construct each sub-instance I'_J in $O(n)$ time, as either $k' \leq k$ and $l' \leq l$ or $\widetilde{J} = \emptyset$. Thus all such sub-instances can be generated in $O(n)$ time and we are done. □

Now we introduce another tool to prove the separation property, which constitutes the main tool used in this thesis. Given an instance $I = (G, L)$ of the list M-partition problem, a pair (V_Q, V_S) of two disjoint non-empty subsets of $V(G)$ is said to be a *split pair* in I if V_Q (V_S, respectively) is a clique (stable set, respectively) and the list of any vertex in V_Q (V_S, respectively) consists entirely of 1-colors (0-colors, respectively). Note that this definition implies that $G[V_Q \cup V_S]$ must be a split graph (see Section 2.5) which explains why we use the term split for such pairs.

Theorem 4.3.4. *Suppose M is a 01-diagonal matrix and $\widetilde{G}$ is a hereditary graph class. Suppose there exists a polynomial-time algorithm ALG which takes as input an instance $I = (G, L)$ of the list M-partition problem with $G \in \widetilde{G}$ and a split pair (V_Q, V_S) in I, and breaks I into sub-instances in which the sets V_Q and V_S are separated. Then M has the separation property for $\widetilde{G}$. Moreover, assuming that the running time of the algorithm ALG is upper-bounded by $O(n^d)$ ($d \geq 1$), there exists an algorithm giving the separation property with running time $O(n^{6kld})$, where k and l are the sizes of the blocks A and B of M, respectively.*

Proof. Let $I = (G, L)$ be an instance of the list M-partition problem with $G \in \widetilde{G}$. Suppose we have oracle access for any homogeneous sub-instance of I. By using this oracle access and applying Lemma 4.3.2, we can break I in $O(n^{2kl+3})$ time into list-homogeneous sub-instances. Now we show how to solve each of these sub-instances. Let $I' = (G, L')$ be one of these sub-instances and let (VA, VB) be its homogeneous partition. Suppose we can break I', in polynomial time, into sub-instances in which the sets VA and VB are separated. Then solving each sub-instance $I'' = (G, L'')$ of these sub-instances will consist of solving the list M-partition problem for each set VA and VB independently. In other words, we have to solve the instances $(G[VA], L'')$ and $(G[VB], L'')$ independently. Note that both of these instances are homogeneous sub-instances of I, and thus can be solved in $O(1)$ time using oracle access. So it is sufficient to show how to break each sub-instance I' in such a manner. A necessary condition for I' to have a solution is that the graphs $G[VA]$ and $G[VB]$ must be A-partitionable and B-partitionable, respectively. This can be checked in $O(1)$ time using oracle access. Suppose this necessary condition holds (otherwise I' will have no solution). Let $VA = X_1 \cup \cdots \cup X_k$ and $VB = Y_1 \cup \cdots \cup Y_l$ be the A-partition and B-partition for graphs $G[VA]$ and $G[VB]$, respectively, given by oracle access. Let $P_1, P_2, \cdots P_r$ be an arbitrary ordering of all the pairs (X_i, Y_j), $i = 1, 2, \cdots k, j = 1, 2, \cdots l$, for which both sets X_i and Y_j are non-empty. Note that each pair P_i is indeed a split pair. So by repeatedly applying the algorithm ALG to each pair P_i, $i = 1, 2, \cdots r$, we can break I' into sub-instances in which the sets VA and VB are separated. More precisely, we perform the following procedure to break I':

1. Initially define $\widetilde{I} = \{I'\}$,
2. For $i = 1, 2, \cdots r$ do as follows: for each instance $I'' \in \widetilde{I}$, use the algorithm ALG to break I'' into sub-instances in which the sets in the pair P_i are separated, and replace I'' with these sub-instances (in the set $\widetilde{I}$).

It is easy to see that after this procedure, for any sub-instance in $\widetilde{I}$, the sets VA and VB are separated. Also note that I' has been broken into the sub-instance in $\widetilde{I}$ (as a result of repeatedly breaking each member in the above procedure).

To analyze the running time of breaking I', let a_i be the number of sub-instances in $\widetilde{I}$ in the i-th iteration (i.e., when we are considering the pair P_i). We have $a_{i+1} = O(n^d \cdot a_i)$, for $i = 1, 2, \cdots r-1$, which implies that $a_i = O(n^{d.(i-1)})$, and thus the running time of breaking I' is $O(n^{rd}) = O(n^{kld})$. □

4.4 A General Tool Related to 0-diagonal 1-free Matrices

Theorem 4.4.1. *Suppose $\widetilde{G}$ is a graph class which contains all the bipartite graphs. Let M be a 0-diagonal 1-free matrix. Then the list M-partition problem, restricted to $\widetilde{G}$, can be solved in polynomial time (of order $O(n^2)$) if M corresponds to a bipartite co-circular arc graph, and otherwise it is NP-complete.*

Proof. Let H be the graph corresponding to M. Recall that the list M-partition problem is equivalent to the list H-coloring problem. If H contains an odd cycle then we claim that the list H-coloring problem for bipartite graphs is NP-complete. To prove this, let t be the smallest value for which H contains an odd cycle of order t, which we denote by C_{2t+1}. This implies that C_{2t+1} must be actually a chordless cycle (i.e., an induced subgraph of H). Now we introduce a reduction from the list 3-coloring problem for bipartite graphs (known to be NP-complete) to the list C_{2t+1}-coloring problem for bipartite graphs as follows: given an instance of the former problem with the bipartite graph G and lists $L(v) \subseteq [3]$ assigned to each of its vertices, replace each edge of G with a path of order $2t-1$ and assign the list $[2t+1]$ to each new vertex (if there is any). Assume, without loss of generality, that the colors $0, 1$ and 2 represent three successive vertices in the cycle C_{2t+1}. Then it is easy to see that the new graph G has a list C_{2t+1}-coloring if and only if the original graph G has a list 3-coloring.

Now assume H has no odd cycle, and thus is a bipartite graph. Given an instance $I = (G, L)$ of the list H-coloring problem for an arbitrary graph G, if G is not bipartite then clearly there is no solution for I, otherwise G has to be a bipartite graph. This suggest a polynomial reduction from the list H-coloring problem for general graphs to the list H-coloring problem for bipartite graphs. Feder et al. in [124] proved that the former problem (i.e., the list H-coloring problem for general graphs) is NP-complete when H is not a co-circular arc graph, and otherwise it is polynomial. The polynomial case can be inferred (from their proof) to be square in degree. Thus the theorem follows. $\square$

Recall that when the matrix M is 0-diagonal 1-free, the list M-partition problem is equivalent to the H-coloring problem. Thus the above theorem can be equivalently formulated as follows: suppose $\widetilde{G}$ is a graph class which contains all the bipartite graphs, and H is a graph without loops. Then the list H-coloring problem for $\widetilde{G}$ can be solved in polynomial time (of order $O(n^2)$) if H is a bipartite co-circular arc graph, and otherwise it is NP-complete.

Chapter 5

Graphs Representing Geometric Configurations

Certain properties of graphs representing geometric structures (e.g., interval graphs, circular arc graphs, permutation graphs, etc) can be exploited to solve the M-partition problem efficiently in polynomial time. In this chapter we introduce some properties of this type and some tools to benefit from them. These tools are introduced in Section 5.1. In the subsequent sections we show how they can be applied to a variety of graph classes, mostly those classes consisting of intersection or containment graphs of certain geometric objects, in particular intervals on a line and arcs on a circle.

In Section 5.2 we focus on the intersection and containment graphs of intervals (namely interval and permutation graphs), as well as their bipartite versions (namely interval and interval containment bigraphs). Using our tools, we show that all list M-partition problems are polynomial for these classes (Theorem 5.2.4).

Next, we consider the natural extension of these classes by using arcs on a circle instead of intervals on a line. Intersection graphs of arcs on a circle (namely circular arc graphs) and containment graphs of arcs on a circle (namely circular arc containment graphs) are both studied in Section 5.3. Using the same techniques as the previous section, but in a more elaborate way, we show that all list M-partition problems for these classes can be solved in polynomial time (Theorem 5.3.3). We also take a brief look at several natural extensions of these classes, and for each of them we introduce some NP-complete list M-partition problem(s).

In Section 5.4 we consider comparability graphs. Recall (from Section 3.1.3) that this class not only is a natural extension of the class of circular arc containment graphs, it is also an extension of any graph class consisting of containment graphs. For this class, we prove the dichotomy (for the list version) in several special cases of the matrix M (particularly in Theorem 5.4.6).

In the last section, we briefly discuss other classes of intersection graphs. They include the class of circle graphs (the intersection graphs of chords in a circle), which have a structure similar to permutation graphs and are tightly connected to circular arc and circular arc containment graphs. However, unlike these graphs, we find an NP-complete list M-partition problem (which is actually a list H-coloring problem) for circle graphs (Theorem 5.5.1).

We already gave detailed accounts (see Chapter 3) for each of the graph classes considered in this chapter, and we explained that all these classes have polynomial-time recognition algorithms. Thus, in this chapter we always assume that the intervals or arcs (or any other geometric objects) corresponding to the vertices of the input graph, or the transitive orientation of the edges (in the case of comparability graphs), are all given as part of the input. Also, without loss of generality, using a simple argument, we may assume that all interval and arcs (used in intersection and containment graphs) are closed (i.e., the endpoints are part of the interval/arc). When the context is clear, we would identify a vertex with its corresponding geometric object. For example, we may say "the arc v," which means "the arc corresponding to the vertex v."

5.1 The Main Techniques

Suppose an instance $I = (G, L)$ of the list M-partition problem and an arbitrary subset $D \subseteq V(G)$ are given. In general, the number of solutions for the instance $(G[D], L)$ may be exponential in $n = |V(G)|$. We introduce an alternate way to represent these solutions for which we have polynomially many choices (in terms of n) in many cases of G and D. Given an M-partition f of the graph $G[D]$ (the subgraph of G induced by D), we introduce a function $NC_D^f : V(G) - D \to 2^{[m]}$ by defining $NC_D^f(v)$ to be the set of colors which occur, in the coloring f, in the neighborhood of v in D (i.e., $NC_D^f(v) = \{f(w) | w \in D, vw \in E(G)\}$). Similarly, we introduce a function $\overline{NC_D}^f : V(G) - D \to 2^{[m]}$ by defining $\overline{NC_D}^f(v)$ to be the set of colors which occur, in the coloring f, in the non-neighborhood of v in D (i.e, $\overline{NC_D}^f(v) = \{f(w) | w \in D, vw \notin E(G)\}$). For both functions, when the context is clear, we may drop the superscript f and the subscript D. The pair $P = (NC_D^f, \overline{NC_D}^f)$ of functions is called the *profile of the M-partition* f of $G[D]$. We say that a color $c \in [m]$ for a vertex $v \in V(G) - D$ *conforms* with the profile P if $M(c, c') \neq 0$ for each color $c' \in NC_D(v)$ and $M(c, c') \neq 1$ for each color $c' \in \overline{NC_D}(v)$. In the case of $D = \emptyset$, we define $NC_D(v)$ and $\overline{NC_D}(v)$ to be the empty set for all vertices $v \in V(G)$, and we call such a profile an *empty profile*.

Note that the profile P completely represents the effect of pre-coloring the set D (based on the coloring f) on all vertices in $V(G) - D$. In other words, given an instance $I = (G, L)$ of the list M-partition problem and a vertex $v \in V(G) - D$, a color $c \in L(v)$ does not conform with the profile P if and only if there exists a vertex $u \in D$ such that the color $c \in L(v)$ conflicts with the color $f(u) \in L(u)$. This implies that, instead of storing the full details of the coloring f, we may store only its profile without the loss of necessary information needed by the M-partition problem. The advantage of this representation is that, as we shall see later, for certain graph classes the total number of profiles (for certain subsets D) is polynomially bounded (in terms of $n = |V(G)|$), while the number of M-partitions of $G[D]$ could be exponential in n. This is the main ingredient for designing polynomial-time algorithms in this chapter. The following result demonstrates how to use the profile representation. Throughout this chapter, for an ordering $v_1, v_2, \cdots v_n$ of $V(G)$ (which is clear from the context), we denote by D_i $(0 \leq i \leq n)$ the set $\{v_1, v_2, \cdots v_i\}$ (which implies that $D_0 = \emptyset$).

Theorem 5.1.1. *There exists an algorithm to solve any instance $I = (G, L)$ of the list M-partition problem, provided that an ordering $v_1, v_2, \cdots v_n$ of $V(G)$ is provided as part of input. Furthermore, the algorithm runs in $O(n \cdot \sum_{i=0}^{n} n_i^2)$ time, where n_i, $0 \leq i \leq n$, is the number of distinct profiles of all the solutions of the instance $(G[D_i], L)$ of the list M-partition problem. (Recall $D_i = \{v_1, v_2, \cdots, v_i\}$.)*

Proof. For each $0 \leq i \leq n$, we want to build a set $\widetilde{P_i}$ consisting of the (distinct) profiles of all the solutions of the instance $(G[D_i], L)$. For $i = 0$, the set D_i is empty, and thus $\widetilde{P_0}$ contains only one empty profile. Suppose the set $\widetilde{P_{i-1}}$ is already constructed for some $1 \leq i \leq n$. We now show how to construct the set $\widetilde{P_i}$. We iterate over all profiles $P \in \widetilde{P_{i-1}}$. Suppose P is the profile of a solution f of $(G[D_{i-1}], L)$. The solution f can be extended to a solution of the instance $(G[D_i], L)$ by defining $f(v_i) = c$ only if the color c for v_i conforms with P. This can be checked in $O(1)$ time. The total choices for c is also $O(1)$. Having fixed a color c for

$f(v_i)$, the profile P' of the solution f (of the instance $(G[D_i], L)$) can be derived from the profile $P = (NC, \overline{NC})$ by the following procedure: for each vertex $v \in V(G) - D_i$, add the color c to the set $NC(v)$ if v is adjacent to v_i, and to the set $\overline{NC}(v)$ if v is non-adjacent to v_i. Having constructed P', the next step is to add this profile to the set $\widetilde{P_i}$. To do so, we need to check whether P' is already in this set, and add it if it is not so. This can be checked in $O(n \cdot n_i)$ time by simply comparing P' to any existing profile in $\widetilde{P_i}$ (comparing two profiles can be accomplished in $O(n)$ time). We note that by a more elaborate algorithm which uses sorted lists, we can do this checking much faster. However, we adhere to the simple algorithm as in this thesis we are mainly concern about our algorithms to be polynomial rather than optimal. Thus $\widetilde{P_i}$ can be constructed in $O(n \cdot n_{i-1} \cdot n_i)$ time, and the whole process (until $i = n$) can be accomplished in $O(n \cdot \sum_{i=0}^{n} n_{i-1} \cdot n_i)$, which is $O(n \cdot \sum_{i=0}^{n} n_i^2)$. If $\widetilde{P_i} = \emptyset$, for some $0 < i \leq n$ then it means there is no solution for the instance I, otherwise there is a solution. Furthermore, to generate a solution of I (if one exists), we may keep track of the values $f(v_i)$ and the path which leads us from the empty profile in $\widetilde{P_0}$ to the final profile in $\widetilde{P_n}$. □

An ordering $v_1, v_2, \cdots v_n$ of $V(G)$ is called an *M-profile-bounding ordering of degree t* if the total number of (distinct) profiles of all the M-partitions of $G[D_i]$ is upper-bounded by n^t, for $i = 1, 2, \cdots n$. Using this ordering as input in the algorithm of Theorem 5.1.1 yields the following corollary:

Corollary 5.1.2. *Suppose an arbitrary matrix M and a constant $t > 0$ are given. There exists a polynomial-time algorithm (of order $O(n^{2t+2})$) to solve any instance $I = (G, L)$ of the list M-partition problem, provided that G has an M-profile-bounding ordering of degree t which is given as part of the input.*

Using this corollary as the main tool, in this chapter our general approach to solve the list M-partition problem is to find an M-profile-bounding ordering of some constant degree for the input graph G. Now we introduce some tools for finding such an ordering. A sequence $S_1, S_2, \cdots S_n$ of sets is called *inclusive* if $S_i \subseteq S_{i+1}$ for $i = 1, 2, \cdots n-1$. A key (but simple) observation is as follows:

Observation 5.1.3. *The number of inclusive sequences $S_1, S_2, \cdots S_n$ in which $S_i \subseteq [m]$ is a set of colors ($i = 1, 2, \cdots n$), is exactly $(n+1)^m$.*

Proof. Any inclusive sequence $S_1, S_2, \cdots S_n$ can be uniquely characterized by a function $f : [m] \to \{1, 2, \cdots, n+1\}$ where $f(c)$ is defined as the smallest index $1 \leq i \leq n+1$ for which $c \in S_i$, and define $i = n+1$ if c does not belong to any set in the sequence. Note that we have $n+1$ choices for each index $f(c)$, and consequently $(n+1)^m$ many different choices for the function f. □

Our general approach in this chapter is based on the fact that for many graph classes (especially those representing some geometric configuration), the profiles related to set D_i can be broken into several inclusive sequences (for which we have polynomially many choices). This implies that we have polynomially many choices for the profiles and it leads to the fact that we have an M-profile-bounding ordering. To describe this approach more formally, we need to define some technical concepts. Given a graph G and a constant $t > 0$, we say that an ordering $v_1, v_2, \cdots v_n$ of $V(G)$ is *t-bounding* if for any $1 \leq i \leq n$, the set $V(G) - D_i$ can be partitioned into at most t parts such that for any two vertices u, v in the same part, one of the two sets $N(u) \cap D_i$ and $N(v) \cap D_i$ is a subset of the other. Such a partition of $V(G) - D_i$ is called a *t-bounding partition.*

Theorem 5.1.4. *Given a matrix M and a graph G, any t-bounding ordering of $V(G)$ is an M-profile-bounding ordering of degree $2mt$.*

Proof. Let $v_1, v_2, \cdots v_n$ be a t-bounding ordering of $V(G)$. Given $1 \leq i \leq n$, let $U_1, U_2, \cdots U_t$ be a t-bounding partition of $V(G) - D_i$ (some parts may be empty). Being a t-bounding ordering implies that there exists an ordering $w_1, w_2, \cdots w_{|U_j|}$ of U_j (for $j = 1, 2, \cdots t$) such that $N(w_p) \cap D_i \subseteq N(w_{p+1}) \cap D_i$ $(1 \leq p < |U_j|)$. This implies that, given an M-partition f of $G[D_i]$, the sequence $\{NC^f_{D_i}(w_p)\}_{p=1}^{|U_j|}$ is inclusive. Thus according to Observation 5.1.3, we have at most $(|U_j|+1)^m$ many choices for this sequence, which is representing the restriction of the function NC_{D_i} to the set U_j. By considering all indices $1 \leq j \leq t$, we may conclude that the total number of choices for the function NC_{D_i} is at most $\prod_{j=1}^{t}(|U_j|+1)^m$, which is upper-bounded by n^{mt}. A similar argument, by reversing the ordering of w_is, implies that the total number of choices for the function $\overline{NC}_{D_i}$ is also upper-bounded by n^{mt}. Thus the total number of choices for the profiles of all the M-partitions of $G[D_i]$ is upper-bounded by n^{2mt}. □

Note that Corollary 5.1.2 and Theorem 5.1.4 together introduce the following chain of implications: being a t-bounding ordering implies being an M-profile-bounding ordering, and the latter implies a polynomial-time algorithm to solve all list M-partition problems. Based on this relation, our main approach in this chapter is to find a t-bounding ordering, for some constant $t > 0$, for the graphs we are studying. In the next section we will start applying this technique to interval graphs and other similar graph classes.

5.2 The Intersection and Containment Graphs of Intervals

We recall that we assume all intervals to be closed. For a closed interval v on a line, we denote by $r(v)$ and $l(v)$ its right and left endpoints, respectively. Also, we may treat the points on a line as real numbers. Thus, for two points p_1 and p_2 on the line, the statement $p_1 \leq p_2$ means the point p_1 does not lie on the right side of the point p_2.

Lemma 5.2.1. *There exists a $O(n \log n)$ time algorithm to find a 1-bounding ordering for any interval graph G.*

Proof. Let $v_1, v_2, \cdots v_n$ be an arrangement of $V(G)$ based on non-decreasing order of $r(v_i)$, $i = 1, 2, \cdots n$. Such an ordering can be found in $O(n \log n)$ time using any standard sorting algorithm (e.g., the Heap Sort). We claim that this is a 1-bounding ordering. Given $1 \leq i \leq n$, assume $u, v \in V(G) - D_i$ are two distinct vertices. Without loss of generality, suppose $l(v) \leq l(u)$. Assume an interval $w \in D_i$ intersects the interval u, but not the interval v. The fact that $r(w) \leq r(v)$ (due to the non-decreasing order of $r(v_i)$s) implies that $r(w) < l(v)$. Combined with the assumption $l(v) \leq l(u)$, we obtain $r(w) < l(u)$ which contradicts the assumption that w intersects u. This means $N(u) \cap D_i \subseteq N(v) \cap D_i$, which proves our claim. ☐

Recall that interval containment graphs are identical with permutation graphs (see Section 3.1.4 for details).

Lemma 5.2.2. *There exists a $O(n \log n)$ time algorithm to find a 1-bounding ordering for any interval containment (or permutation) graph.*

Proof. Let $v_1, v_2, \cdots v_n$ be an arrangement of $V(G)$ based on non-decreasing order of $r(v_i)$, $i = 1, 2, \cdots n$ (which can be found in $O(n \log n)$). Using a similar argument as in the proof of Lemma 5.2.1 above, one can prove that this ordering is a 1-bounding ordering. ☐

Now we handle the case of interval and interval containment bigraphs, which are the bipartite version of interval and interval containment graphs, respectively (see Section 3.1.6):

Lemma 5.2.3. *There exists a $O(n \log n)$ time algorithm to find a 2-bounding ordering for any interval or interval containment bigraph G.*

Proof. The definition of interval or interval containment bigraph requires that G is a bipartite graph with parts V_1 and V_2. Let $v_1, v_2, \cdots v_n$ be an arrangement of $V(G)$ based on non-decreasing order of $r(v_i)$, $i = 1, 2, \cdots n$. Such an ordering can be found in $O(n \log n)$ time. We claim that this is a 2-bounding ordering. Given $1 \leq i \leq n$, partition the set $V(G) - D_i$ into two parts $X_1 = (V(G) - D_i) \cap V_1$ and $X_2 = (V(G) - D_i) \cap V_2$. By a similar argument as in the proof of Lemma 5.2.1 above, one can show that this partition is a 2-bounding partition. □

Combining all these lemmas along with Corollary 5.1.2 and Theorem 5.1.4, we obtain the following result:

Theorem 5.2.4. *For any matrix M, the list M-partition problem can be solved in polynomial time when restricted to the following graph classes: interval graphs (in $O(n^{4m+2})$ time), interval containment (permutation) graphs (in $O(n^{4m+2})$ time), interval bigraphs (in $O(n^{8m+2})$ time) and interval containment bigraphs (in $O(n^{8m+2})$ time).*

We close this section by mentioning that recently Enright et al. [112] showed that the list H-coloring problem (a special case of the list M-partition problem) can be solved in polynomial time for interval and permutation graphs. This result is a special case of the above theorem. Moreover, our result offers a running time of $O(n^d)$ with $d = 4m + 2$, whereas the algorithm in [112] has $d = O(m^4)$. However, their approach may work for other graph classes.

5.3 Circular Arc and Circular Arc Containment Graphs

We recall that we assume all arcs to be closed. For an arc v on a circle, define the *right* and *left* endpoints of v with respect to the counter-clockwise direction, and denote them by $r(v)$ and $l(v)$, respectively. In the proofs given in this section, we fix a reference point p on the circle. We say that a point p_1 is on the right (left, respectively) side of another point p_2 if moving in the counter-clockwise direction (the clockwise direction, respectively) starting from p, we encounter p_2 before we encounter p_1. Given two points p_1, p_2, denote by $[p_1, p_2]$ the closed arc between p_1 and p_2 in the counter-clockwise direction.

Lemma 5.3.1. *There exists a $O(n \log n)$ time algorithm to find a 2-bounding ordering for any circular arc graph G.*

Proof. All the points and arcs in this proof are assumed to be on the circle containing the arcs of G. We fix a reference point p as an arbitrary point on the circle which is not the endpoint of any arc in $V(G)$. For each vertex $v \in V(G)$, define $q(v)$ as the right endpoint or the left endpoint of v, depending on whether v contains the point p or not, respectively. Let $v_1, v_2, \cdots v_n$ be an ordering of $V(G)$ in which $q(v_i)$ is not on the right side of $q(v_{i+1})$ $(1 \leq i < n)$. Such an ordering can be found in $O(n \log n)$ time. We claim that this is a 2-bounding ordering. Fix a value for $1 \leq i \leq n$. We partition the set $V(G) - D_i$ into parts V_1 and V_2 consisting of the arcs which contain the point p and the arcs which do not contain the point p, respectively. Let $u, v \in V(G) - D_i$ be two distinct vertices. We consider two cases:

1. Suppose $u, v \in V_1$. Without loss of generality, assume that the point $r(u)$ is not on the right side of $r(v)$. This implies that the arc $[p, r(u)]$ is contained in the arc $[p, r(v)]$. Assume $x \in D_i$ is adjacent to u. We show that it has to be adjacent to v too. This is obvious if x contains p. So assume that it does not. Since $l(u) \in [q(v_i), p]$ and $r(x) \notin [q(v_i), p]$, the only way that the arc x could intersect the arc u is that x intersects the arc $[p, r(u)]$. Recalling that $[p, r(v)]$ contains $[p, r(u)]$, we may conclude that x also intersects v. This implies that $N(u) \cap D_i \subseteq N(v) \cap D_i$.

2. Suppose $u, v \in V_2$. Without loss of generality, assume that the point $l(u)$ is not on the left side of $l(v)$. We claim that $N(u) \cap D_i \subseteq N(v) \cap D_i$. To prove so, note that any arc $x \in D_i$ which contains the point p intersects both u and v. This is due to the fact that $l(x) \in [p, q(v_i)]$ which implies that x contains the whole arc $[q(v_i), p]$ including both $r(u)$ and $r(v)$. Now assume that the arc $x \in D_i$ does not contain p and intersects u. Since $r(u) \in [q(v_i), p]$ and $l(x) \notin [q(v_i), p]$, the only way that the arc x could intersect the arc u is that x intersects the arc $[l(u), q(v_i)]$, assuming that $q(v_i) \in u$ (otherwise there is no intersection). Note that $[l(u), q(v_i)]$ is contained in the arc $[l(v), q(v_i)]$ (since we assumed $l(u)$ is not on the left side of $l(v)$), which is contained in the arc v.

The arguments above imply that the partition of $V(G) - D_i$ into V_1 and V_2 is a 2-bounding partition. □

Lemma 5.3.2. *There exists a $O(n \log n)$ time algorithm to find a 2-bounding ordering for any circular arc containment graph.*

Proof. Fix a reference point p as an arbitrary point on the circle different than the endpoints of the arcs in $V(G)$. For each vertex $v \in V(G)$, define $q(v)$ as in the proof of Lemma 5.3.1 above. Let $v_1, v_2, \cdots v_n$ be an ordering of $V(G)$ in which $q(v_i)$ is not on the right side of $q(v_{i+1})$, $1 \leq i < n$. (Such ordering can be found in $O(n \log n)$.) Using a similar argument as in the proof of Lemma 5.3.1 above, one can prove that this ordering is a 2-bounding ordering. □

Combining these two lemmas with Corollary 5.1.2 and Theorem 5.1.4, we obtain the following result:

Theorem 5.3.3. *For any matrix M, the list M-partition problem, restricted to circular arc and circular arc containment graphs, can be solved in polynomial time (of order $O(n^{8m+3})$).*

One natural direction to further extend the scope of our results would be to consider extensions of the class of circular arc and the class of circular arc containment graphs. This is partially inspired by our successful extension of the polynomial results for interval and interval containment graphs to circular arc and circular arc containment graphs (as we saw in Theorem 5.3.3 above). The reference site [97] lists four graph classes as well-known minimal extensions of circular arc graphs, namely: 1) balanced 2-interval graphs, 2) polygon-circle graphs, 3) spider graphs, 4) upper domination perfect graphs. Now we show that for each of these classes, there are NP-complete list M-partition problems. The first class (namely, balanced 2-interval graphs) is defined as the intersection graphs of two equal length intervals on a line [74, 149]. This class contains line graphs as subclass ([149]), and thus the 3-coloring problem is NP-complete when restricted to this class (see [197]). The next two classes both contain circle graphs (the intersection graphs of chords on a circle) as subclass. In Section 5.5 we introduce one NP-complete list M-partition problem restricted to circle graphs (Theorem 5.5.1). As for the class of upper domination perfect graphs, it is known (see [97]) that this class contains $\{P_5, \overline{C_6}\}$-free graphs as subclass. It is sufficient to find some NP-complete list M-partition problem for the class of complements of the graphs in this class, namely $\{\overline{P_5}, C_6\}$-free graphs which contain $\{C_3, C_6\}$-free graphs as subclass. Note that any graph with girth larger than 6 is $\{C_3, C_6\}$-free, and the 3-coloring problem is known to be NP-complete when restricted to graphs with girth larger than 6

[206]. As for circular arc containment graphs, the only minimal extension suggested by the reference site [97] is the class of comparability graphs. This class is studied in the next section, in which we show that there are NP-complete list M-partition problems restricted to comparability graphs (see Corollary 5.4.1). The conclusion is that, any natural minimal extension of circular arc and circular arc graphs, as listed in the website [97], contain at least one NP-complete list M-partition problem, while for the class of circular arc and circular arc containment graphs all list M-partition problems are polynomial.

5.4 Comparability Graphs

The class of comparability graphs contains bipartite graphs as subclass. Thus applying Theorem 4.4.1 yields the following corollary:

Corollary 5.4.1. *Suppose M is a 0-diagonal 1-free matrix. Then the list M-partition problem, restricted to comparability graphs, can be solved in polynomial time (of order $O(n^2)$) if M corresponds to a bipartite co-circular arc graph, and otherwise it is NP-complete.*

It is open to find a similar dichotomy result when M is not 1-free. One implication of the above result is that, our tools using profiles (introduced in Section 5.1) are not likely to work for comparability graphs and arbitrary matrices M (as there are NP-complete problems). However, by restricting the clique-covering number (see Section 1.2) of the input graph, we can apply our tools. More precisely we have the following result:

Lemma 5.4.2. *Given a constant $t > 0$, there exists a $O(n^2)$ time algorithm to find a t-bounding ordering for any comparability graph G with clique-covering number upper-bounded by t.*

Proof. We recall that we assume a transitive orientation of the edges of G is given as part of input (In fact such orientation can be found in linear time, see Section 3.1.3.) It is easy to see that, having a transitive orientation, a topological ordering $v_1, v_2, \cdots v_n$ of G can be found in $O(n^2)$ time (So $v_i \to v_j$ only if $i < j$.) We claim that this is a t-bounding ordering. Suppose $1 \leq i \leq n$ is given. Since the clique-covering number of G is upper-bounded by t, the set $V(G) - D_i$ can be partitioned into t cliques $Q_1, Q_2, \cdots Q_t$ (some cliques could be empty). Let $u, v \in V(G) - D_i$ be two distinct vertices belonging to the same clique Q_p. Without loss of generality, suppose $u \to v$. Assume a vertex $w \in D_i$ is adjacent to u. The fact that $v_1, v_2, \cdots v_n$ is a topological ordering of $V(G)$ and that $w \in D_i, u \notin D_i$ imply that $w \to u$. Now the fact that $u \to v$ and the transitive property of G imply that $w \to v$. This means $N(u) \cap D_i \subseteq N(v) \cap D_i$, which implies that the partition $Q_1, Q_2, \cdots Q_t$ is a t-bounding partition. □

Combining this lemma with Corollary 5.1.2 and Theorem 5.1.4, we obtain the following result:

Theorem 5.4.3. *For any matrix M, the list M-partition problem, restricted to comparability graphs with clique-covering number upper-bounded by a constant $t > 0$, can be solved in polynomial time (of order $O(n^{4mt+2})$).*

If M is a 1-diagonal matrix then a necessary condition for a graph G to have an M-partition is that its clique-covering number should not exceed m. This condition can be checked in $O(n^3)$ time for comparability graphs G [163]. If this condition does not hold then G has no M-partition, otherwise the clique-covering number of G is bounded by m, which allows us to apply Theorem 5.4.3 above. This leads to the following corollary:

Corollary 5.4.4. *Suppose M is a 1-diagonal matrix. Then the list M-partition problem, restricted to comparability graphs, can be solved in polynomial time (of order $O(n^{4m^2+2})$).*

In order to connect our results for 0-diagonal and 1-diagonal matrices, we prove the separation property (see Section 4.2):

Lemma 5.4.5. *Any 01-diagonal matrix M has the separation property for comparability graphs. Moreover, there exists an algorithm giving the separation property with running time $O(n^{30klm})$, where k and l are the sizes of the blocks A and B of M, respectively (see Figure 1.2).*

Proof. By applying Theorem 4.3.4, it is enough to find a polynomial-time algorithm ALG which takes as input an instance $I = (G, L)$ of the list M-partition problem in which G is a comparability graph and a split pair (V_Q, V_S) in I, and breaks I into sub-instances in which the sets V_Q and V_S are separated. We assume that the edges of G are transitively oriented. Let $v_1, v_2, \cdots v_{|V_Q|}$ be a topological ordering of the set V_Q. Note that $v_i \to v_j$ if $i < j$, since V_Q is a clique. For each $v \in V_S$, define p_v (q_v, respectively) as the largest (smallest, respectively) index $1 \le i \le |V_Q|$ for which $v_i \to v$ ($v \to v_i$, respectively). If no such index p_v (q_v, respectively) exists then set it to 0 ($|V_Q| + 1$, respectively). The transitive property implies that $v_i \to v$ for $1 \leq i \leq p_v$ and $v \to v_i$ for $q_v \leq i \leq |V_Q|$. This also implicitly implies that $p_v < q_v$ and v is non-adjacent to all vertices v_i with $p_v < i < q_v$. Suppose for two vertices $u, v \in V_S$, the intervals $[p_u, q_u]$ and $[p_v, q_v]$ are disjoint. Without loss of generality, assume that $q_u < p_v$. Then we have $u \to q_u$ and $q_u \to v$. Then the transitive property implies that $u \to v$ which is a contradiction, since u and v are non-adjacent (as they both belong to the stable set V_S). This means that all the intervals $[p_v, q_v]$ ($v \in V_S$) are pairwise intersecting, and thus they all share a common element which is an integer. We denote this element by s.

Now for any solution f of I, let $r_1^f, r_2^f, r_3^f, r_4^f$ be the functions such that for each color $c \in [m]$:

- $r_1^f(c)$ ($r_2^f(c)$, respectively) is the largest value $0 \leq t \leq |V_Q|$ such that for each $i \in [1, t]$ ($i \in [|V_Q| - t + 1, |V_Q|]$, respectively), we have $f(v_i) \neq c$.
- $r_3^f(c)$ ($r_4^f(c)$, respectively) is the largest value $0 \leq t \leq |V_Q|$ such that for each $i \in [s - t, s - 1]$ ($i \in [s + 1, s + t]$, respectively), we have $f(v_i) \neq c$.

Let J_f be the tuple $(f(v_s), r_1^f, r_2^f, r_3^f, r_4^f)$ and $\widetilde{J}$ be the set of all possible choices for such tuples. More precisely, to generate all tuples $(c_s, r_1, r_2, r_3, r_4)$ in $\widetilde{J}$ we consider all colors $c_s \in [m]$ and all integers $0 \leq r_1(c), r_2(c), r_3(c), r_4(c)) \leq |V_Q|$, for $c \in [m]$. Thus $\widetilde{J}$ can be generated in $O(|V_Q|^{4m})$ time. Note that $\widetilde{J}$ contains the tuples J_f for all possible solutions f of I.

Now for each tuple $J = (c_s, r_1, r_2, r_3, r_4) \in \widetilde{J}$, define I_J as a sub-instance of I which is obtained by applying the constraints imposed by the tuple J. More precisely, pre-color the vertex v_s with c_s and for each color $c \in [m]$, remove c from the list of all vertices v_j for $j \in [1, r_1(c)] \cup [|V_Q| + 1 - r_2(c), |V_Q|] \cup [s - r_3(c), s - 1] \cup [s + 1, s + r_4(c)]$. Note that for any solution f of I we have the following property: for any two (not necessarily distinct) colors c and c' with $M(c, c') = 0$ ($M(c, c') = 1$, respectively) and any vertex $v \in V_S$, if either $p_v > r_1^f(c)$ or $|V_Q| + 1 - r_2^f(c) > q_v$ ($q_v > r_4^f(c) + s + 1$ or $s - r_3^f(c) - 1 > p_v$, respectively) then $f(v) \neq c'$. Now based on this property we apply the following changes to each sub-instance I_J (where $J = (c_s, r_1, r_2, r_3, r_4) \in \widetilde{J}$): for any two (not necessarily distinct) colors c and c' with $M(c, c') = 0$ ($M(c, c') = 1$, respectively) remove c' from the list of any vertex $v \in V_S$ if either $p_v > r_1(c)$ or $|V_Q| + 1 - r_2(c) > q_v$ ($q_v > r_4(c) + s + 1$ or $s - r_3(c) - 1 > p_v$, respectively). Additionally, we update the lists based on the pre-colored vertex v_s (which makes this vertex separated from the set V_S). It is easy to see that after these changes, the sets V_Q and V_S are separated in each sub-instance I_J, and that any solution f of I is a solution of the sub-instance I_{J_f} as well. This implies that I is broken into sub-instances I_J, for $J \in \widetilde{J}$. Moreover, we can construct each sub-instance I_J in $O(n)$ time. Thus all such sub-instances can be generated in $O(n^{4m+1})$ time. □

Combining this lemma with Observation 4.2.2 and Corollaries 5.4.1 5.4.4 leads to the following dichotomy:

Theorem 5.4.6. *Suppose M is a 01-diagonal matrix in which the block A is 1-free. Then the list M-partition problem, restricted to comparability graphs, can be solved in polynomial time (of order $O(n^{150m^5})$) if A corresponds to a bipartite co-circular arc graph, and otherwise it is NP-complete.*

5.5 Other Intersection Graphs

Theorems 5.2.4 and 5.3.3 show that the technique of profiles (as formulated in Corollary 5.1.2) can successfully handle the intersection graphs of intervals on a line, arcs on a circle and chords between two parallel lines (i.e., the case of permutation graphs). During our research, we considered some other classes of intersection graphs including point-interval, trapezoid, disk, etc (see [97] for general references of these graph classes). The attempts to apply the technique of profiles to these classes did not meet with success, mostly because there is no obvious reason for the existence of an M-profile-bounding ordering. So these classes are not included in this thesis.

A more formal indication that the technique of profiles may not work is that, for some of these classes, there exist NP-complete list M-partition problems. For example, consider the class of balanced 2-interval graphs which we introduced in Section 5.3. Recall that the graphs in this class are intersection graphs (of two equal length intervals on a line), and that the 3-coloring problem is NP-complete when restricted to this class. Another example is the class of circle graphs, defined as the intersection graphs of chords in a circle (see [89] for general reference). We particularly chose this class because its structure is similar to permutation graphs (as the intersection graphs of chords between two parallel lines), and as we saw at the end of Section 5.3, the class of circle graphs was used to find some examples of NP-complete list M-partition problems for several extensions of the class of circular arc graphs including polygon-circle graphs and spider graphs. For circle graphs, we introduce one NP-complete list M-partition problem, which is actually a list H-coloring problem:

Theorem 5.5.1. *Let H be the complement of the graph shown in Figure 5.1. (So based on our definition of complement in Section 1.2, the only vertices with loops in H are t and f.) Then list H-coloring problem is NP-complete even when restricted to circle graphs.*

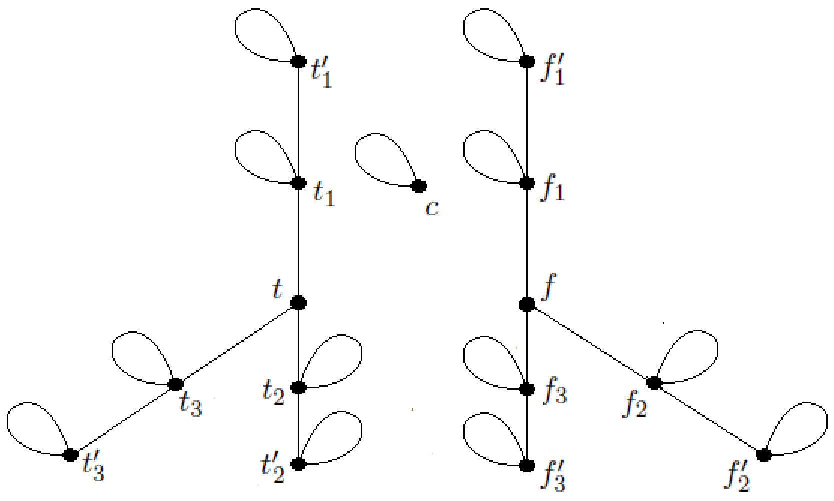

Figure 5.1: Let H be the complement of the depicted graph. Then the list H-coloring problem is NP-complete for circle graphs.

Proof. Note that, for the sake of clarity of the proof, we labeled the vertices of H using small Latin letters (with possibly subscripts and accents) rather than index numbers. Our aim is to reduce the NAE 3-SAT problem without negated variables (see Section 4.1 for its definition) to the list H-coloring problem for circle graphs. Suppose an instance of the NAE 3-SAT problem (without negated variables) with the variables $x_1, x_2, \cdots x_n$ and the clauses $Cl_1, Cl_2, \cdots Cl_r$ is given. We introduce an instance $I = (G, L)$ of the list H-coloring problem with G being a circle graph constructed as follows: move in the clockwise direction on the circle and for each clause Cl_i $(i = 1, 2, \cdots r)$, draw three successive non-intersecting chords v_1^i, v_2^i, v_3^i in the clockwise direction, each corresponding to a different literal of Cl_i. (Recall that Cl_i has exactly three literals.) Refer to Figure 5.2(a) for an example of such chords when we have three clauses. Recall that by color we mean the vertices of H which are labeled as shown in Figure 5.1. The colors f_j are called false colors and the colors t_j

are called true colors ($j = 1, 2, 3$). Assign the list $\{f_j, t_j\}$ to each chord v^i_j ($1 \le i \le r$, $1 \le j \le 3$). This means any chord corresponding to a literal x can take either a false color or a true color. In the former case we define the value of x to be FALSE and in the latter case to be TRUE.

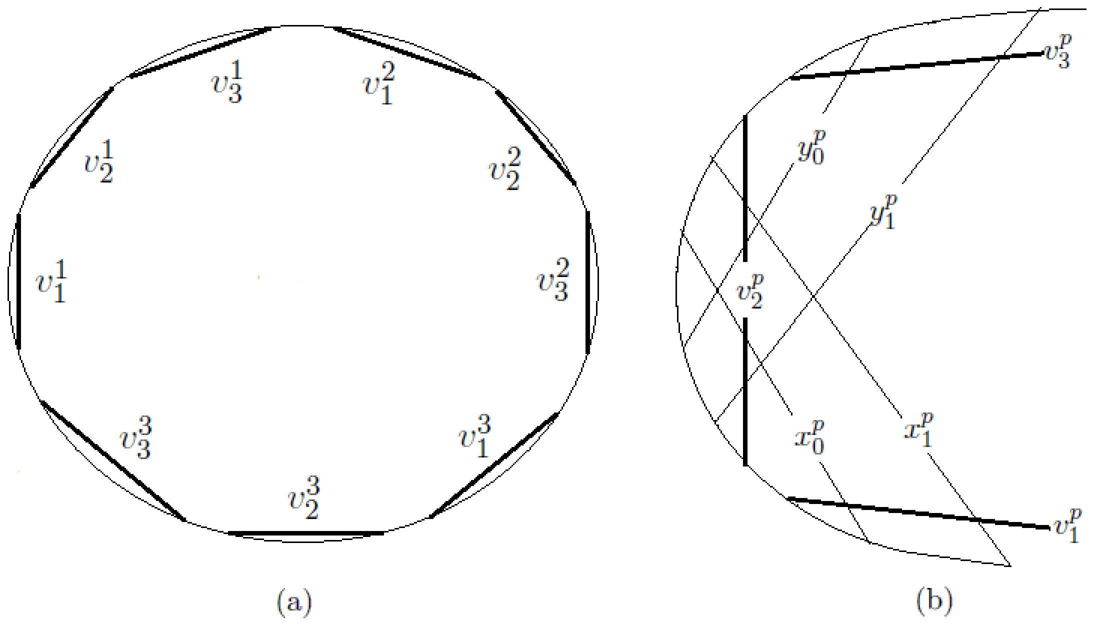

Figure 5.2: The configuration of a circle graph G based on an instance of the NAE 3-SAT problem.

One problem which arises from this configuration is that, two distinct chords u and v may correspond to the same literal, and thus we have to make sure that they both either take a false color or a true color. To handle this problem, we draw a chord connecting the midpoints of u and v, and assign the list $\{t, f\}$ to it. This guarantees that in any list H-coloring of G, any two chords corresponding to the same literal must either both have a false color or both have a true color (otherwise the chord connecting their midpoints cannot take any color from its list).

The next step is to forbid the literals in each clause from all taking false or all taking true colors at the same time. To do so, for each clause Cl_p ($1 \le p \le r$), we draw two intersecting chords x^p_0 and y^p_0 such that, among all the chords v^i_j ($1 \le i \le r$, $1 \le j \le 3$), x^p_0 intersects only the chords v^p_1 and v^p_2, and y^p_0 intersects only the chords v^p_2 and v^p_3. (see Figure 5.2(b)). We assign the list $\{f'_1, f'_2, c\}$ and $\{f'_2, f'_3, c\}$ to x^p_0 and y^p_0, respectively. It is easy to see that this configuration forbids the chords v_1^p, v_2^p and v_3^p from all having false colors at the same time in any list H-coloring of G. Similarly, we draw two intersecting chords x_1^p and y_1^p such that, among all the chords v_j^i, x_1^p intersects only the chords v_1^p and v_2^p, and y_1^p intersects only the chords v_2^p and v_3^p. We assign the list $\{t_1', t_2', c\}$ and $\{t_2', t_3', c\}$ to x_1^p and y_1^p, respectively. This configuration forbids the chords v_1^p, v_2^p and v_3^p from all having true colors at the same time in any list H-coloring of G. This way, the chords v_1^p, v_2^p and v_3^p cannot all have false colors or all have true colors at the same time in any list H-coloring of G. Thus, any list H-coloring of G corresponds to a solution of the NAE 3-SAT instance and vice versa. This defines a polynomial-time reduction from the NAE 3-SAT problem without negated variables to the list H-coloring problem restricted to circle graphs. $\square$

It would be interesting to see whether there exists a graph H without loops for which the list H-coloring problem is NP-complete when restricted to circle graphs.

Chapter 6

Line, Quasi-Line and Claw-free Graphs

In this chapter we study the (list) M-partition problem for the class of line graphs and its extensions to the class of quasi-line graphs and then to the class of claw-free graphs. In Section 3.2 we gave the definition of these classes and discussed their importance and relevance in studying the M-partition problem. Recall that these three classes are also important in the structural theory of graphs (see Section 6.1).

In the first section, we offer some partial dichotomy results for the class of line graphs. Unlike the graphs based on geometric representations (which we studied in the previous chapter), we introduce some NP-complete list M-partition problems for line graphs. The well-known examples are the 3-coloring [197] and the stable cutset problems (see Section 2.4) [30]. We will introduce some additional NP-complete list M-partition problems for line graphs (Theorem 6.1.5). Thus, having dispelled the hope for a polynomial complexity for all list M-partition problems, we focus on the dichotomy problem for several special cases of the matrix M (Theorems 6.1.1 to 6.1.2).

In Section 6.2 we consider quasi-line graphs. They are even more difficult than line graphs, as we introduce a group of list M-partition problems which are NP-complete for this class but polynomial for line graphs (see Observation 6.2.3). For these graphs, we prove a partial dichotomy result for the list M-partition problem with M being restricted to 01-diagonal matrices (Theorem 6.2.4). However, for the non-list version, using the structural theorem of Chudnovsky and Seymour, we prove that the M-partition problem can be solved in polynomial time when M is a 1-diagonal 0-free matrix (Theorem 6.2.5).

The last section is dedicated to claw-free graphs, for which we offer a partial dichotomy result for 01-diagonal matrices (Theorem 6.2.4). Extending any of these dichotomies is a definite direction for future work.

In this section we use a graphic model to facilitate the representation of matrices. In this model, for each color i we draw a vertex v_i whose appearance depends on the value of $M(i,i)$. Then for each pair (i,j) of distinct colors, we draw an edge $e_{i,j}$ whose appearance depends on the value of $M(i,j)$. The details of this notation are given in Figure 6.1 along with an example. Note that some of our symbols for vertices and edges represent more than one possibility (e.g., the black vertex with a white X within). This allows one diagram to represent a group of matrices. We should mention that a similar symbolic representation of matrices, called *trigraphs*, have been used in several papers (see for example [129]), in which alternative symbols were used for vertices v_i and $e_{i,j}$. A notable feature of trigraphs is that, $e_{i,j}$ is a non-edge if $M(i,j) = 0$, an edge if $M(i,j) = *$ and a double-edge (two parallel line) if $M(i,j) = 1$. Since most of the matrices used in this section contain many * entries, using trigraphs may result in drawing so many edges which reduces the clarity. For this reason, we introduced our own version to have a more clear graphic representation.

$$\begin{bmatrix} 0 & * & 1 \\ * & 1 & 0 \\ 1 & 0 & * \end{bmatrix}$$

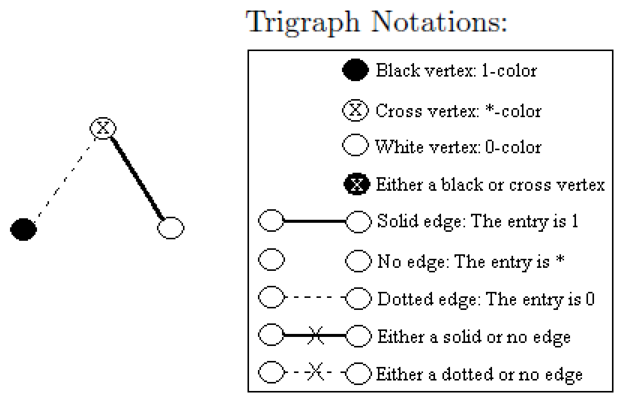
Trigraph Notations:
Black vertex: 1-color
Cross vertex: *-color
White vertex: 0-color
Either a black or cross vertex
Solid edge: The entry is 1
No edge: The entry is *
Dotted edge: The entry is 0
Either a solid or no edge
Either a dotted or no edge

Figure 6.1: The trigraph notations with an example.

6.1 Line Graphs

In this section we focus on line graphs. As far as the M-partition problem is concerned, restricting the input graph to line graphs is equivalent to partitioning the edges of general graphs. For this reason, we use the term *edge-partition of the graph G* interchangeably with the term "partition of the line graph of G" throughout this section. When talking about the complexity of an M-edge-partition problem, the notation n still stands for the number of vertices of the input graph. We say that an instance $I' = (G', L')$ of the list M-edge-partition problem is a sub-instance of another instance $I = (G, L)$ (of the list M-edge-partition problem) when G' is a subgraph (not necessarily induced) of G and $L'(e) \subseteq L(e)$ for any edge $e \in E(G')$.

A 1-free *-diagonal matrix is said to be a *non-interval pattern* if its corresponding graph is not an interval graph. The main dichotomy results proved in this section are listed below:

Theorem 6.1.1. *Suppose M is a 01-diagonal matrix. Then the list M-partition problem. restricted to line graphs, can be solved in polynomial time (of order $O(n^{8m^2})$) if M does not contain 3-coloring, and otherwise it is NP-complete.*

Theorem 6.1.2. *Suppose M is a 1-free *-diagonal matrix. Then the list M-partition problem, restricted to line graphs, can be solved in polynomial time (of order $O(n^2)$), unless M is a non-interval pattern, in which case it is NP-complete.*

Theorem 6.1.3. *Suppose M is a matrix in which the blocks B,S^* and C^* (see Figure 1.2) are 0-free, 1-free and all *, respectively. Then the list M-partition problem, restricted to line graphs, can be solved in polynomial time (of order $O(n^{8m^2})$) if M does not contain 3-coloring, stable cutset or any non-interval pattern, and otherwise it is NP-complete.*

Theorem 6.1.4. *Suppose M is a matrix in which the block B has size $\geq m - 2$ (i.e., there are at most two non 1-colors). Then the list M-partition problem, restricted to line graphs, can be solved in polynomial time (of order $O(n^{8m^2})$).*

Next we offer a set of additional NP-complete patterns for line graphs:

Theorem 6.1.5. *The list M-partition problem, restricted to line graphs, is NP-complete when M is any of the matrices shown in Figure 6.2.*

Note that some of the matrices shown in Figure 6.2 do not contain the well-known NP-complete patterns of 3-coloring, stable set or non-interval patterns. Thus they constitute new NP-complete patterns for line graphs which are not included in the aforementioned dichotomy results.

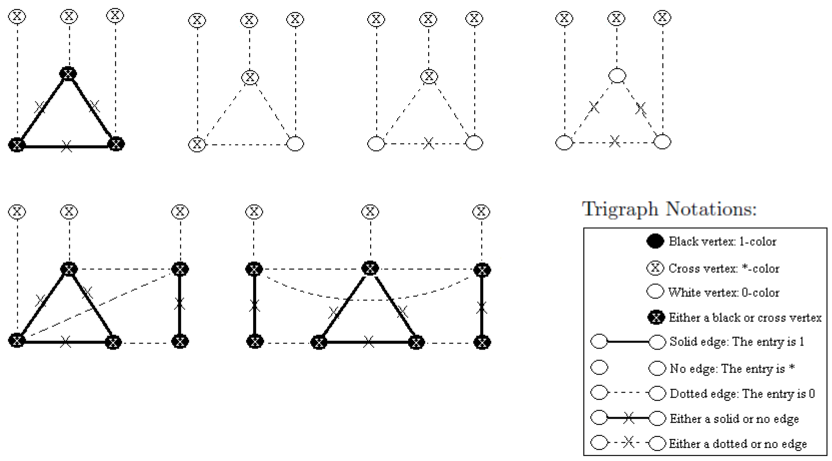
Trigraph Notations:
Black vertex: 1-color
Cross vertex: *-color
White vertex: 0-color
Either a black or cross vertex
Solid edge: The entry is 1
No edge: The entry is *
Dotted edge: The entry is 0
Either a solid or no edge
Either a dotted or no edge

Figure 6.2: Additional NP-complete patterns for line graphs.

The proofs of these results are based on some auxiliary tools which we explain shortly. The proof of each theorem will be given once the necessary concepts are explained. Suppose an arbitrary instance $I = (G, L)$ of the list M-edge-partition problem is given. Our aim is to break I into sub-instances which have simpler structure. The intuition is, if a 1-color c is used for at least four edges in a solution of I then all the edges having this color must share a common endpoint. We use this fact to construct sub-instances of I in which the 1-colors (and some other colors contributing to 1 entries in M) appear only in the list of the edges incident with a fixed vertex.

To put it formally, we define several parameters for any solution f of I: define D_f as the set of colors used by f for at most three edges (So all the colors in $[m] - D$ are used for at least four edges.) Define $g_f : D_f \to 2^{E(G)}$ as a function in which $g_f(c)$ is the set (possibly empty) of edges e colored with c, for $c \in D_f$ (So $f(e) = c$.) Obviously for any two distinct colors $c, c' \in D_f$ we have $g_f(c) \cap g_f(c') = \emptyset$. Given an arbitrary subset $D \subseteq [m]$ of colors, we say that a color $c \in D$ is a *centralized color of* D if $M(c, c') = 1$ for some color $c' \in D$ (not necessarily distinct from c). Let c be a centralized color of the set $[m] - D_f$.The fact that c is used by f for at least four edges (as $c \notin D_f$) implies that all the edges having the color c in f must share a common endpoint which we call the *center of the color c in f*. Note that any two centralized colors c, c' of $[m] - D_f$ with $M(c, c') = 1$ will have the same center in

f. Define h_f as the function from the set of centralized colors of $[m] - D_f$ to the set $V(G)$, where $h_f(c)$ is the center of the color c in f.

Let J_f be the tuple (D_f, g_f, h_f) and $\widetilde{J}$ be the set of all possible choices for such tuples. More precisely, to generate all tuples $J = (D, g, h)$ in $\widetilde{J}$ we consider:

1. All subsets $D \subseteq [m]$,
2. All functions $g : D \to 2^{E(G)}$ with $|g(c)| \leq 3$ and $g(c) \cap g(c') = \emptyset$ for any two distinct colors $c, c' \in D$,
3. All functions h mapping each centralized color of $[m] - D$ to a vertex of G in such a way that $h(c) = h(c')$ whenever $M(c, c') = 1$.

Note that $\widetilde{J}$ contains the tuples J_f for all possible solutions f of I (However, it may contain tuples not corresponding to any solution.) Nevertheless, the total number of elements in $\widetilde{J}$ is polynomially bounded. More precisely, we have $O(1)$ many choices for the subset D, $O(|E(G)|^3)$ many choices for each value $g(c)$ which implies having $O(n^{6m})$ many choices for the function g, and $O(n^m)$ many choices for the function h. Thus $\widetilde{J}$ can be generated in $O(n^{7m})$ time.

Now for each tuple $J = (D, g, h) \in \widetilde{J}$, define I_J as a sub-instance of I which is obtained by applying the constraints imposed by the tuple J. More precisely, apply the following procedure to the instance I to obtain I_J:

1. For each color $c \in D$, pre-color all the edges in $g(c)$ with c and remove c from the lists of all other edges, i.e., those not in the set $g(c)$. (This can be done in $O(|E(G)|)$ time.)
2. For any centralized color c of $[m] - D$, remove c from the lists of all edges not incident with the vertex $h(c)$. (This can be done in $O(|E(G)|)$ time.)

It is easy to see that any solution f of I is a solution of the sub-instance I_{J_f} as well. This implies that I is broken into sub-instances I_J, for $J \in \widetilde{J}$. Moreover, we can construct each sub-instance I_J in $O(n^2)$ time (see the procedure above). Thus all such sub-instances can be generated in $O(n^{7m+2})$ time. Note that some sub-instances I_J may be invalid (i.e., some of their lists could be empty). This means that they have no solution and thus we can ignore them and only focus on the valid ones.

The sub-instances I_J have some good structural properties. To describe them, let us fix a tuple $J = (D, g, h) \in \widetilde{J}$. We call the vertices $h(c)$, for $c \in D$, the *centers* of I_J. We typically denote the centers by $v_1, v_2, \cdots v_r$ (So r is a constant no greater than m.) Note that the colors in the set D only appear in the singleton lists. So we can get rid of these colors by updating the lists based on the edges having these singleton lists and then eliminating these edges. More precisely, given an arbitrary instance $I' = (G', L')$ of the list M-edge-partition problem, we define the *simplification* of I, as a sub-instance obtained from I' as follows: update the lists based on the set of the edges having singleton list in I, and then eliminate all these edges. It should be clear how a solution of I' can be converted to a solution of its simplification and vice versa. For any instance of the list M-edge-partition problem, the set of *available colors* is the set of colors which appears in at least one list. Also, by *centralized colors of an instance* (of an arbitrary list M-partition problem) we mean the centralized colors of the set of the available colors of that instance. Let $I'_J = (G, L')$ be the simplification of I_J. Clearly the set of the available colors of I'_J is a subset of $[m] - D$. Thus the function h maps each centralized color c of I'_J to some center (a vertex v_i for some $1 \leq i \leq r$) which we call the *center of the color* c. Now the structural properties of I'_J can be described as follows:

1. There is a subset $\{v_1, v_2, \cdots v_r\}$ of $V(G)$ called the *centers* (of I'_J), where $r \leq m$.
2. Any centralized color c (of I'_J) is present only in the list of the edges incident with a fixed center, known as the center of c.
3. Any two centralized colors c and c' (of I'_J) with $M(c, c') = 1$ have the same center.

The above properties are exploited extensively in this section. So let us call any instance of the list M-edge-partition problem with the above properties a *regular instance.* Our arguments can be summarized in the following lemma:

Lemma 6.1.6. *Any instance $I = (G, L)$ of the list M-edge-partition problem can be broken in polynomial time (of order $O(n^{7m+2})$) into sub-instances such that the simplification of each sub-instance is a regular instance.*

Using this lemma, our task of solving the general instances of the list M-edge-partition problem is reduced to solving regular instances. So let $I = (G, L)$ be a regular instance. The *residue instance of* I, denoted by $I_{res} = (G_{res}, L)$, is obtained from I by removing all its centers from G. We define the *residue matrix of* I, denoted by $M_{res}(I)$, as the matrix obtained from M by deleting the rows and colomns corresponding to the colors that are not the available colors of I_{res}. Note that any available color of I_{res} is a non-centralized color of I, and thus the matrix $M_{res}(I)$ is 1-free. This is an important property, since I_{res} is an instance of the list $M_{res}(I)$-edge-partition problem. Thus, to solve I_{res}, we have to deal with a 1-free principal sub-matrix of M which, as we will see later, is a major contributing factor in simplifying the original problem of solving I. Obviously any solution of I is also a solution of I_{res}, but the reverse may not always be the case. A solution of I_{res} is called *extendable to* I if it can be extended to a solution of I by defining a coloring for edges incident with the centers of I. This means solving I is equivalent to finding an extendable solution of I_{res}. So we shift our focus to studying the extendable solutions of I_{res}. In particular, we are interested in converting the problem of finding an extendable solution of I_{res} to solving an instance of the list $M_{res}(I)$-edge-partition problem, as the matrix $M_{res}(I)$ is 1-free and presumably easier to handle than M.

Let $D \subseteq [m]$ be a set of colors consisting of non-centralized colors of a regular instance I such that for every two distinct colors $c, c' \in D$ we have $M(c, c') = *$. In the instance I, the set D is called *acceptable* for a non-center vertex v if there is a list M-edge-partition f of the edges between v and the centers of I such that it does not forbid any color $c \in D$ for the vertex v (i.e., $M(c, f(e)) \neq 0$ for any edge e between v and a center). If D is not acceptable then it is called *forbidden* (for the vertex v). Note that the decision of whether D is acceptable or forbidden for a non-center vertex v depends only on solving an instance of the list M-edge-partition problem limited to the vertex v and all the centers of I (whose number is $O(1)$), and thus it can be decided in $O(1)$ time. Given a solution f_{res} of I_{res}, we say that f_{res} is *locally extendable* for a vertex $v \in V(G_{res})$ if the set of colors $f_{res}(e)$, for all edges $e \in E(G_{res})$ incident with v, is acceptable for v. Note that a necessary condition for f_{res} to be extendable is that it should be locally extendable for all non-center vertice. This condition is not necessarily sufficient. In order to make it sufficient, we impose another constraint: I is said to be *strictly regular* if (in addition to being regular) for any center of I, any two distinct edges e and e' incident with this center are separated (i.e., $M(c, c') \neq 0$ for any color $c \in L(e)$ and $c' \in L(e')$). Recall that determining whether a set is acceptable or not can be decided in $O(1)$ time. Thus we obtain the following observation:

Observation 6.1.7. *Given a strictly regular instance I, a solution f_{res} of I_{res} is extendable to I if and only if it is locally extendable for any vertex $v \in V(G_{res})$. Additionally, we can* *extend the solution f_{res} to a solution of I or decide that it is not extendable in $O(n)$ time.*

Proof. The forward direction is trivial as stated before. Now suppose f_{res} is locally extendable for any vertex $v \in V(G_{res})$. Thus for any vertex $v \in V(G_{res})$ there must be a list M-edge-partition f_v of the edges connecting v to any center of I which does not conflict with the coloring f_{res}. The coloring f_v can be found in $O(1)$ time, since the number of centers is $O(1)$. Note that we can put together the coloring functions f_v, for $v \in V(G_{res})$, without any conflict, since I is strictly regular. By doing so we will obtain a coloring for the edges incident with the centers of I which can be used to extend f_{res} to a solution of I. □

The next lemma shows that the cost of having strictly regular instances is polynomial:

Lemma 6.1.8. *Any regular instance $I = (G, L)$ of the list M-edge-partition problem can be broken in polynomial time (of order $O(n^{2m^2})$) into strictly regular sub-instances.*

Proof. Let $v_1, v_2, \cdots v_r$ be the centers of I. We define several parameters for any solution f of I: define D_f as a function in which $D_f(i)$, for $i = 1, 2, \cdots r$, is the set of colors used by f in at least one edge incident with the center v_i. Note that for any two distinct colors $c, c' \in D_f(i)$ we have $M(c, c') \neq 0$. Each 0-color in each set $D_f(i)$ must be used for exactly one edge incident with v_i. So we define the function g_f in which, for any 0-color $c \in D_f(i)$, $i = 1, 2, \cdots r$, $g_f(i, c)$ is the (only) edge incident with v_i which is colored with c.

Let J_f be the pair (D_f, g_f) and $\widetilde{J}$ be the set of all possible choices for such pairs. More precisely, to generate all the pairs $J = (D, g)$ in $\widetilde{J}$ we consider:

1. All subsets $D(i) \subseteq [m]$ in which $M(c, c') \neq 0$ for any two distinct colors $c, c' \in D(i)$, for $i = 1, 2, \cdots r$,

2. All functions g which map any pair $(i, c) \in \{1, 2, \cdots r\} \times [m]$ in which c is a 0-color in $D(i)$ to an edge e incident with the center v_i.

Note that $\widetilde{J}$ contains the pairs J_f for all possible solutions f of I (As before, it also may contain pairs not corresponding to any solution.) Nevertheless, the total number of pairs in $\widetilde{J}$ is polynomially bounded. More precisely, for $i = 1, 2, \cdots r$, we have $O(1)$ many choices for the set $D(i)$ and $O(n)$ many choices for the edge $g(i, c)$. Thus $\widetilde{J}$ can be generated in $O(n^{mr})$ time.

Now for each pair $J = (D, g) \in \widetilde{J}$, define I_J as a sub-instance of I which is obtained by applying the constraints imposed by the pair J. More precisely, apply the following procedure to the instance I to obtain I_J:

1. Remove from the list of any edge incident with the center v_i any color which is not in $D(i)$, for $i = 1, 2, \cdots r$. (This can be done in $O(|E(G)|)$ time.)

2. For any 0-color $c \in D(i)$, pre-color the edge $g(i, c)$ with c, for $i = 1, 2, \cdots r$. (This can be done in $O(1)$ time.)

It is easy to see that any solution f of I is a solution of the sub-instance I_{J_f} as well. This implies that I is broken into sub-instances I_J, for $J \in \widetilde{J}$. Moreover, we can construct each sub-instance I_J in $O(n^2)$ time (see the procedure above). Thus all such sub-instances can be generated in $O(n^{m^2+2})$ time. It is easy to see that any sub-instance is strictly regular. □

This lemma further simplifies our task by reducing it to solving strictly regular instances. Now we can give the proofs for two of our main results:

Proof. (Theorem 6.1.4) By applying Lemmas 6.1.6 and 6.1.8 we may assume I is strictly regular. Note that we have at most two non 1-colors which implies we have at most two non-centralized colors of I. This means I_{res} has at most two available colors. Thus finding an extendable solution for I_{res} can be easily formulated as a 2-SAT problem by using Observation 6.1.7. It is easy to see that this 2-SAT problem can be solved in $O(n^2)$ time. □

The following straightforward observation is used in several proofs in the rest of this section:

Observation 6.1.9. *Suppose M is a 1-free matrix. Then an instance $I = (G, L)$ of the list M-edge-partition problem has a solution if and only if, for each connected component G' of G, the instance (G', L) has a solution.*

Now we give the proof of Theorem 6.1.1:

Proof. It is known that the 3-coloring problem for line graphs is NP-complete ([197]). So assume that M does not contain 3-coloring. Suppose an instance $I = (G, L)$ of the list M-edge-partition problem is given. By applying Lemmas 6.1.6 and 6.1.8 we may assume that I is strictly regular. So it is sufficient to find an extendable solution of $I_{res} = (G_{res}, L)$. Recall that I_{res} is an instance of the list $M_{res}(I)$-edge-partition problem in which the matrix $M_{res}(I)$ is 1-free. Suppose a vertex v in G_{res} has degree at least three. Let e_1, e_2 and e_3 be three distinct edges incident with v. If I_{res} has any solution f (not necessarily extendable) then colors $f(e_i)$ for $i = 1, 2, 3$ will induce a pattern identical to 3-coloring (since $M_{res}(I)$ is

1-free), which is a contradiction! So we may terminate the algorithm by giving a negative answer if such a vertex v exists. Otherwise, G_{res} will be a union of disjoint paths and cycles. Note that $M_{res}(I)$ is 1-free, and thus according to Observation 6.1.9, we can reduce the task of finding an extendable solution of I_{res} to finding an extendable solution for each path and cycle in G_{res}. In the case of cycle, by considering all the pre-colorings of an arbitrary edge of the cycle, we can break the problem to (constantly many) sub-instances for the path obtained by removing that edge. So we may assume that G_{res} is indeed a path $w_1, w_2, \cdots w_t$. Note that according to Observation 6.1.7, finding an extendable solution for I_{res} is equivalent to finding a solution of I_{res} which is locally extendable for each vertex w_i, $i = 1, 2, \cdots t$. We introduce a dynamic programming algorithm to solve this problem. For $i = 1, 2, \cdots t-1$ and color $c \in L(w_i w_{i+1})$, define $P(c, i)$ as the sub-problem of finding a solution for the path $w_1, \cdots w_{i+1}$ (i.e., a solution for the instance (G_i, L), in which G_i is the graph induced by vertices $w_1, w_2, \cdots w_{i+1}$) which is locally extendable for all vertices w_j, $j = 1, 2, \cdots i+1$, and the edge $w_i w_{i+1}$ has color c. Solving $P(c, i)$ for $i > 1$ involves deciding a color c' for edge $w_{i-1} w_i$. This color must obey rules imposed by matrix $M_{res}(I)$, list L and forbidden sets of w_i (More precisely, the set $\{c, c'\}$ should not be a forbidden set for w_i.) Since we have a bounded number of choices for c', we may come up with all possible choices for the colors c' in $O(1)$ time. Coloring the rest of the path is equivalent to solving the sub-problem $P(c', i-1)$ (since $M_{res}(I)$ is 1-free). This scheme produces a dynamic programming algorithm which can be performed in $O(t) = O(n)$ time. □

To prove the other results, we need to describe some more concepts. Let $I = (G, L)$ be a regular instance and let $D \subseteq [m]$ be a subset of colors consisting of non-centralized colors of I such that for every two distinct colors $c, c' \in D$ we have $M(c, c') = *$. The set D is called a *minimal forbidden set* for a non-center vertex v (of I) if D is *forbidden* for v while every proper subset of D is acceptable for v. Now assume I is strictly regular. Using Observation 6.1.7, we can further reduce I to I_{res} at the cost of introducing some forbidden sets for each vertex of I_{res}. To analyze the situation better, we define the concept of *minimal forbidden set* without reference to any particular instance: we say that D is a *minimal forbidden set for the matrix M* if there exists a strictly regular instance I such that D is a minimal forbidden set for a non-center vertex v of I. In this case, we say that the instance I is an instance *corresponding to the set D*.

It is important to note that the instance I can be limited only to a single non-center vertex v and at most m centers. This is because other non-center vertices will have no effect on a set being acceptable or forbidden for v. So we have a bounded number of choices for I, and thus the set of minimal forbidden sets of M can be generated in $O(1)$ time. As we will see later in this section, the structure of the set of minimal forbidden sets has a role in determining the complexity of the list M-edge-partition problem. The following lemma provides one example of such a role:

Lemma 6.1.10. *Let M be a 1*-diagonal matrix which does not contain any non-interval pattern. Suppose all minimal forbidden sets of M (if there is any) have size one. Then the list M-edge-partition problem can be solved in polynomial time (of order $O(n^{8m^2})$).*

Proof. By applying Lemmas 6.1.6 and 6.1.8 we can assume that I is strictly regular. According to Observation 6.1.7 it is sufficient to find an extendable solution of I_{res}. For any non-center vertex v of I, let D_v be the set of colors c for which the set $\{c\}$ is a minimal forbidden set for v (in I). The fact that M does not have any minimal forbidden set of size more than one implies that for any non-center vertex v of I, a set D is forbidden if and only if it contains a color $c \in D_v$. So we construct a new instance I' from I_{res} by removing any color in D_v from the list of all the edges incident with v, for any non-center vertex v. It is easy to see that any extendable solution of I_{res} is a solution of I' and vice versa. Note that I' is an instance of the list $M_{res}(I)$-edge-partition problem and $M_{res}(I)$ is a 1-free principal sub-matrix of M which does not contain any non-interval pattern. Thus by applying Theorem 6.1.2 (which will be proved later in this section) we conclude that I' can be solved in $O(n^2)$ time. Note that constructing I' (from I) can be done in $O(n^2)$ time. □

The following lemma offers a sufficient (but not necessary) condition for a matrix to have no minimal forbidden set of size more than 1:

Lemma 6.1.11. *Let M be a 1*-diagonal matrix which contains none of the patterns shown in Figure 6.3. Then any minimal forbidden set of M has size* 1.

Proof. Suppose to the contrary for some 1*-diagonal matrix M containing none of the matrices in Figure 6.3 there exists a strictly regular instance $I = (G, L)$ consisting of centers $v_1, v_2, \cdots v_r$ and vertex v for which $D = [d]$, for some $d \geq 2$, is a minimal forbidden set. This means the set $D - \{1\}$ is an acceptable set for v. This implies that there exists a solution f_0 of I for which $M(0, f_0(vv_i)) \neq 0$, $i = 1, 2, \cdots r$. Let S_0 be the set of indices $1 \leq i_0 \leq r$ for which $M(1, f_0(vv_{i_0})) = 0$. Note that S_0 must be non-empty, otherwise the coloring f_0 will have no conflict with the color 1 for v, which defies the definition of D as a forbidden set for v. Similarly, by exchanging the role of the color 1 and the color 0, define the coloring f_1 and the set S_1. Suppose for some index $i_0 \in S_0$ and $i_1 \in S_1$ we have $M(f_0(vv_{i_0}), f_1(vv_{i_1})) \neq 0$. Then it is easy to see that the colors $0, 1, c_0 = f_1(vv_i)$ and $c_1 = f_0(vv_i)$ induce in the matrix M one of the patterns represented in Figure 6.3(a), which is a contradiction! Thus for all indices $i_0 \in S_0$ and $i_1 \in S_1$ we have $M(f_0(vv_{i_0}), f_1(vv_{i_1})) = 0$. This implies that $S_0 \cap S_1 = \emptyset$, for otherwise if $i \in S_0 \cap S_1$ then for colors $c_0 = f_1(vv_i)$ and $c_1 = f_0(vv_i)$ we must have $M(c_0, c_1) = 0$ which contradicts with the fact that I is strictly regular (Note that c_0 and c_1 both belong to the list of edge vv_i.) Let f be the M-edge-partition obtained from f_0 by replacing the color of any edge vv_{i_0} with $f_1(vv_{i_0})$, for $i_0 \in S_0$. Note that for any color $c \in D$ we have $M(c, f(vv_i)) \neq 0$, $i = 1, 2, \cdots r$. The fact that D is a forbidden set for v implies that f could not be a solution for I. This means for some index $i_0 \in S_0$ and $j \notin S_0$ we must have $M(c_1', c_1'') = 0$ where $c_1' = f_1(vv_{i_0})$ and $c_1'' = f_0(vv_j)$. Now define $c_1 = f_0(vv_{i_0})$ and $c_0 = f_1(vv_i)$ for some arbitrary index $i \in S_1$. A simple argument shows that the colors $0, 1, c_0, c_1, c_1', c_1''$ are distinct and induce in the matrix M one of the patterns represented in Figure 6.3(b) or (c) (depending on whether $M(c_0, c_1'')$ is equal to 0 or not, respectively). This is again a contradiction. $\square$

Note that the matrices represented in Figure 6.3(a) and (b) each has a two-element minimal forbidden set, but this is not the case for the matrices in part (c). Thus the above Lemma is not offering a complete characterization of the patterns having only singleton minimal forbidden sets. Further extending the above lemma to a complete characterization could be an interesting future work.

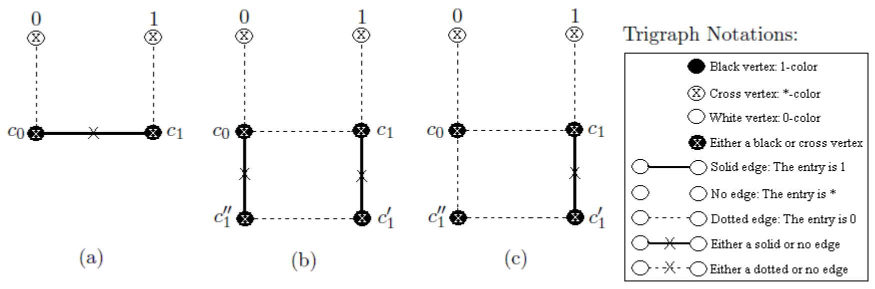

Figure 6.3: A group of patterns such that all the minimal forbidden sets of any matrix M avoiding these patterns have size 1.

By using these lemmas we give the proof of Theorem 6.1.3:

Proof. It is known that the list M-edge-partition problem is NP-complete when M is 3-coloring, stable cutset ([197, 30]) or a non-interval pattern (see Theorem 6.1.2). So assume that M is a matrix free of these patterns. Suppose first that M contains at least one 0-color c_0. Then not containing stable cutset implies that for any two distinct *-colors c_1 and c_2, we must have $M(c_1, c_2) = *$, for otherwise the fact that the block S^* is 1-free will yield $M(c_1, c_2) = 0$ which means colors c_0, c_1, c_2 will induce a stable cutset, which is a contradiction. Since block C^* is entirely $*$, we may conclude that for any *-color c_1 we have $M(c_1, c) = *$ for any color $c \in [m]$. This means in solving any instance $I = (G, L)$ of the list M-edge-partition problem, without loss of generality, we can pre-color any edge e with any *-color which is in $L(e)$ (if there is any). Thus the task of solving I is reduced (after updating the lists based on the set of pre-colored edges) to coloring the rest of the edges whose lists do not contain any *-color. This is equivalent to solving a sub-instance I' of the list M'-edge-partition problem where M' is obtained from M by removing the block S^*. So M' is a 01-diagonal matrix and Theorem 6.1.1 implies that solving I' is polynomial, since M (and thus M') does not contain 3-coloring. Now suppose M has no 0-color. Thus it is a 1*-diagonal matrix. It is easy to see that M cannot contain any of the matrices in Figure 6.3. The argument goes as follows: suppose M contains one of these patterns, then the fact that block C^* is entirely $*$ implies that c_0 and c_1 are both *-colors. Now if M contains one of the patterns shown in (a), then $M(c_0, c_1)$ can be neither * (for then colors $0, 1, c_0, c_1$ form a non-interval pattern) nor 1 (since S^* is 1-free), which is a contradiction. Now suppose M contains one of the patterns shown in (b). Then the restrictions of M implies that $M(c_1, c_1') = M(c_0, c_1'') = *$, and that colors c_1' and c_1'' are either both *-colors or 1-colors. The former case leads to forming a non-interval pattern by colors c_0, c_1, c_1', c_1'', and the latter case contradicts the fact that the block B is 0-free. Next, suppose M contains one of the patterns shown in (c). Then again the fact that block C^* is entirely * implies all colors used for the pattern to be *-colors. Also we must have $M(c_1, c_1') = *$, since S^* is 1-free. But then colors $1, c_1, c_1', c_1''$ form a non-interval pattern. The conclusion is that M does not contain any of these patterns, and thus by Lemma 6.1.11 and 6.1.10 the theorem follows. □

To prove Theorem 6.1.5 we need the following two lemmas as tools:

Lemma 6.1.12. *Let M be a matrix which has a minimal forbidden set D of size at least three consisting only of *-colors. Then the list M-edge-partition problem is NP-complete even when restricted to strictly regular instances.*

Proof. Without loss of generality, we may assume $D = [d]$ for some $3 \leq d \leq m$. Let $I_D = (G_D, L_D)$ be the instance corresponding to the minimal forbidden set D. As discussed before, we may assume that G_D consists of the centers $w_1, w_2, \cdots w_r$ and the vertex w for which D is a (minimal) forbidden set. Suppose an arbitrary graph G is given with partition $V_1 \cup V_2$ of $V(G)$ and a list $L(v) \subseteq [3]$ assigned to any of its vertices. Build a new graph G' by adding r new vertices $v_1, v_2, \cdots v_r$ to G, and for each $v \in V_1$ make a copy of instance I_D in G' with the vertex w and w_is in $V(H)$ identified with the vertex v and v_is $(i = 1, 2, \cdots r)$, respectively (i.e., add a new edge between each v_i and v with the list $L_D(ww_i)$ assigned to it). Next for each $v \in V_1$ (in G') add $d-3$ pendant edges to it (i.e., edges with v as one endpoint and a new vertex as the other endpoint) with the lists $\{c\}$ $(c = 3, 4, \cdots d-1)$ assigned to them (Each list is assigned to exactly one pendant edge.) This construction yields an instance $I = (G', L)$ of the list M-edge-partition problem. Note that I is strictly regular with centers $v_1, v_2, \cdots v_r$. The fact that the instance I_D is strictly regular implies that solving I is equivalent to list 3-coloring the edges of the graph G (with respect to the lists $L(u)$, $u \in V(G)$) with the (only) condition that for any vertex $v \in V_1$ not all the three colors 0,1 and 2 must appear among the edges incident with this vertex. Note that in this problem two adjacent edges can have the same color, since the colors 0,1 and 2 are *-colors. Let us call this problem *not all different (NAD) list 3-coloring.*

We show this problem to be NP-complete by reducing the NAE 3-SAT problem without negated variables (see Section 4.1 for its definition) to it. Suppose an instance of the NAE 3-SAT problem (without negated variables) with the variables $x_1, x_2, \cdots x_n$ and the clauses $Cl_1, Cl_2, \cdots Cl_t$ is given. We construct an instance of the NAD list 3-coloring as follows: to each literal x_i assign a set of $2t$ edges, each one called *a copy of* x_i, and all have list $\{0, 1\}$ assigned to them and share a common endpoint in V_1 (These common endpoints are distinct for different variables.) We add a pendant edge to each common endpoint with list $\{2\}$. This construction guarantees that in any NAD list 3-coloring of G, the copies of the variable x_i $(i = 1, 2, \cdots n)$ are either all colored with 0 or all colored with 1. We interpret this common color as the value of the variable x_i (0 for FALSE and 1 for TRUE). To enforce the restrictions posed by NAE 3-SAT we use the gadget shown in Figure 6.4.

This gadget has three input edges a, a' and a'' (with the list $\{0, 1\}$). A simple argument

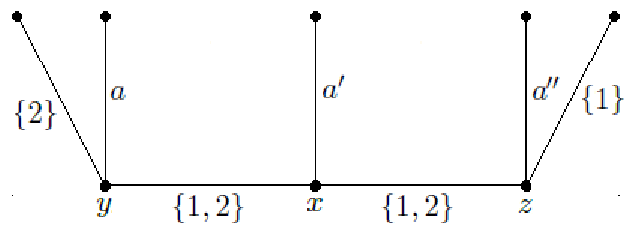

Figure 6.4: The gadget which forbids the edges a,a' and a'' (as its inputs) from all having the color 0 at the same time. Here the vertices x, y and z belong to V_1 and other vertices (excluding the free endpoints of the inputs edges) belong to V_2.

shows that there is a NAD list 3-coloring of this gadget for any pre-coloring combination of the input edges except when they are all colored with 0. Thus for each clause Cl_j we make a new copy of the gadget with the input edges a, a' and a'' identified with a copy of each of the literals in Cl_j (Each copy of any variable is used in at most one gadget.) This way, we make sure that the variables in any clause do not all assume the value FALSE at the same time. By exchanging the role of the color 0 and 1, we make another gadget which forbids only the case when all input edges are colored with 1. Again by a similar construction we can use this gadget to make sure that the variables in any clause do not all assume the value 1 at the same time. Note that we use each copy of any variable in at most one gadget, which explains why we made $2t$ copies of each variable. This construction implies that the instance of the NAE 3-SAT has a solution if and only if our graph has a NAD list 3-coloring. This completes our reduction. □

Lemma 6.1.13. *Given a matrix M, let M' be the matrix obtained from M by applying one of these operations: 1) turning some 1 entries to *, 2) turning each 1 entry to 0 or *. Then the list M-edge-partition problem restricted to regular instances can be reduced in $O(n)$ time to the list M'-edge-partition problem.*

Proof. Let $I = (G, L)$ be a regular instance of the list M-edge-partition problem with the centers $v_1, v_2, \cdots v_r$. The definition of regular instances implies that for any entry $M(c_1, c_2) = 1$, all the edges which have c_1 or c_2 in their lists share a common center v_i, and thus are pairwise adjacent. This implies that any solution for the instance (G, L) of the list M'-edge-partition problem, where M' is obtained from M by the first operation, is also a solution for instance I and vice versa. This implies an $O(1)$ time reduction in the case of the first operation. Now as for the second operation, do the following changes to I: for each center v_i and each edge $e = vv_i$ incident with v_i, replace e with a new edge vv_e (having the same list as e), in which v_e is a new vertex used only by e. After all these changes, remove all the centers v_i, $i = 1, 2, \cdots r$. Call the new instance I'. This modification can be done in $O(|E(G)|)$ time and it guarantees that in I', for any entry $M(c_1, c_2) = 1$, all the edges having the color c_1 or c_2 in their lists are pairwise non-adjacent. This means any solution of I is a solution for the instance I' of the list M'-edge-partition problem and vice versa. □

Now we give the proof of Theorem 6.1.5:

Proof. The patterns shown in Figure 6.2 are exactly the closure of the patterns shown in Figure 6.5. So, by applying Lemma 6.1.13, it is enough to prove that the patterns shown in Figure 6.5 are NP-complete patterns. To do so, for each of these patterns we display (in Figure 6.5) an instance below it which implies the set $D = \{0, 1, 2\}$ is a minimal forbidden set for that pattern. Thus, by Lemma 6.1.12 the theorem follows. □

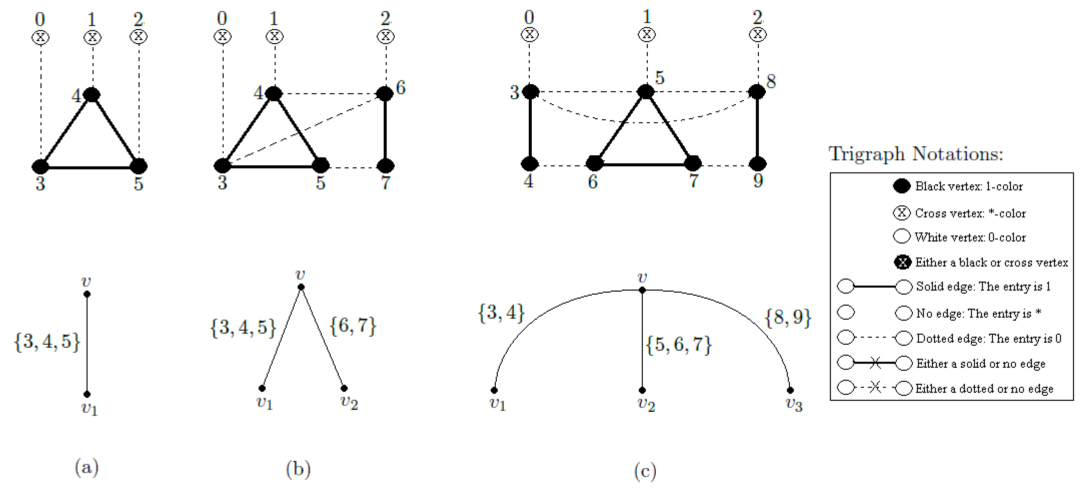

Figure 6.5: Some examples of the patterns which have $D = \{0, 1, 2\}$ as a minimal forbidden set, along with the corresponding instances shown below each pattern (where D is a minimal forbidden set for the vertex v and other vertices are the centers).

To prove Theorem 6.1.2 we need a different set of tools. Let M be a 1-free *-diagonal matrix, and let H be the corresponding graph of M'. Note that every vertex of H has a loop.

Thus, the list M-partition problem is equivalent to the reflexive list H-coloring problem (see Section 2.2). For this problem, when the input graph is restricted to line graphs, we use the term of *reflexive list H-edge-coloring.*

A *chooser (edge-chooser, respectively)* is a path P with the starting vertex (edge, respectively) a and the ending vertex (edge, respectively) b, and lists $L(p) \subseteq [m]$ assigned to each vertex (edge, respectively) p, and the following properties:

1. $L(a) = \{c_1, c_2\}$ and $L(b) = S_1 \cup S_2$, for two distinct colors c_1 and c_2 and two subsets S_1, S_2 of $[m]$,

2. In any list M-partition (M-edge-partition, respectively) f of P, if $f(a) = c_1$ then we must have $f(b) \in S_1$, and if $f(a) = c_2$ then we must have $f(b) \in S_2$,

3. The reverse property (2) above, namely: for $i = 1, 2$ and any color $c' \in S_i$, there exists a list M-partition (M-edge-partition, respectively) f of P such that $f(a) = c_i$ and $f(b) = c'$.

To describe the above properties, we say that the chooser P *takes c_1 to S_1 and c_2 to S_2.* The vertice (edges, respectively) a and b are called the input and the output vertex (edge, respectively), respectively. The idea of chooser has been used (with slight modifications) in several proofs related to the NP-completeness of some H-coloring problems (see for example [118, 190]) in which the choosers were used for vertex coloring. However, in our case we modify them for edge-coloring, which makes only a little change as the line graph of a path is again a path.

Denote by H_s, $s \geq 3$, the graph obtained from the cycle with s vertices by adding a loop at each vertex.

Lemma 6.1.14. *The reflexive list H_s-edge-coloring problem is NP-complete for $s \in \{4, 5\}$.*

Proof. Assume the vertices of H_s are named $0, 1, \cdots s-1$ in the circular order. We reduce the NAE 3-SAT problem without negated variables to the reflexive list H_s-edge-coloring problem. Suppose an instance of the NAE 3-SAT problem (without negated variables) with the variables $x_1, x_2, \cdots x_n$ and the clauses $Cl_1, Cl_2, \cdots Cl_t$ is given. We construct an instance I of the reflexive list H_s-edge-coloring problem as follows: to each literal x_i we assign a set of $2t$ edges, each one called *a copy of x_i*, and all have the list $\{0, 1\}$ assigned to them and share a common endpoint (These common endpoints are distinct for different variables.)

We add a pendant edge with the list $\{2, s-1\}$ to each common endpoint. This construction guarantees that in any solution of I, the copies of x_i ($i = 1, 2, \cdots n$) are either all colored with 0 or all colored with 1. We interpret this common color as the value of the variable x_i in the NAE 3-SAT problem (0 for FALSE and 1 for TRUE).

Let CH_1, CH_2 and CH_3 be the edge-choosers such that CH_1 takes 0 to $\{1\}$ and 1 to $\{1, 2\}$, CH_2 takes 0 to $\{2\}$ and 1 to $\{1, 2\}$ and CH_3 takes 0 to $\{0, 3\}$ and 1 to $[s-1] - \{1\}$. Figure 6.6(a) and (b) shows their detailed structures (The reader can easily check that they are edge-choosers.)

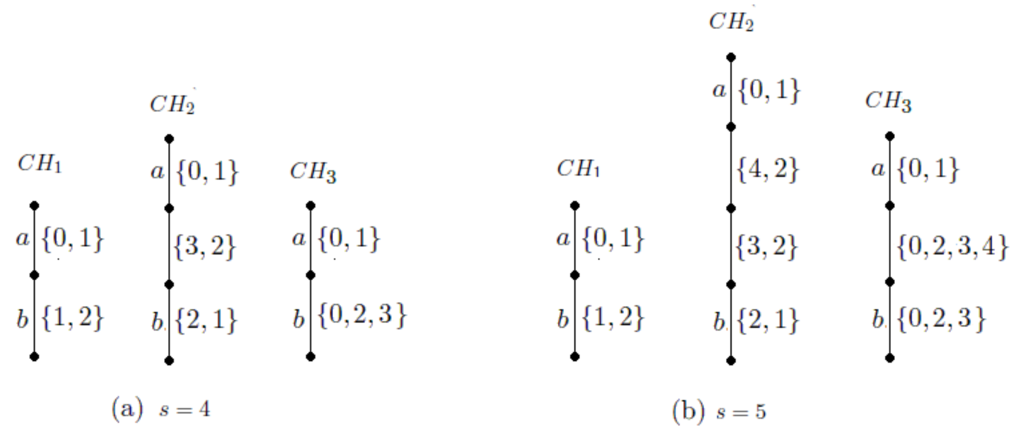

Figure 6.6: The gadgets CH_1, CH_2 and CH_3 for (a) $s = 4$, and (b) $s = 5$.

We construct a gadget called *0-clause* by connecting the output edges of the edge-chooser CH_1, CH_2 and CH_3 to a common endpoint. We denote these output edges of CH_1, CH_2 and CH_3 by b, b' and b'' and the input edges by a, a' and a'', respectively. We claim that this gadget satisfies the following property: for all pre-coloring combinations of the input edges, except when all have the color 0, the gadget has a reflexive list H_s-edge-coloring, otherwise (when all have the color 0) it does not. To prove this claim, note that if a and a' are both pre-colored with 0 then b and b' must have the color 1 and 2, respectively. Thus the only permitted colors for b'' are 1 and 2. This implies that a'' assumes the color 1 (otherwise there is no such coloring). Now if at least one of the edges a and a' takes the color 1, then we can have b and b' having the same color. If this common color is 1 then we can color b'' with 0 (regardless of the color of a''), and if it is 2 then we can color b'' with either 3 or 2 (depending on the color of a'').

Now for each clause we make a new 0-gadget with its input edges identified with copies of the variables in that clause (using each copy of every variable in at most one gadget). This construction guarantees that the literals of Cl_j will not all assume the value FALSE at the same time. We can define the *1-gadget* similar to 0-gadget by changing the role of the color 0 and $s-1$ with the color 1 and 2, respectively. Using a similar construction (using the 1-gadget instead of the 0-gadget) we can make sure that the variables of every clause will not all assume the value TRUE at the same time. This suggests that the NAE 3-SAT instance is satisfiable if and only if I has a solution. It is easy to see that this reduction is polynomial. $\square$

An *asteroidal triple* of H is a stable set consisting of three vertices in $V(H)$ such that for any two of them there exists a path between them which contains no vertex adjacent to the third vertex.

Lemma 6.1.15. *The reflexive list H-edge-coloring problem is NP-complete when H contains an asteroidal triple.*

Proof. Suppose the asteroidal triple in H consists of the vertices $0, 1, 2$. We reduce the NAE 3-SAT problem without negated variables to the reflexive list H-edge-coloring problem. Suppose an instance of the NAE 3-SAT problem (without negated variables) with the variables $x_1, x_2, \cdots x_n$ and the clauses $Cl_1, Cl_2, \cdots Cl_r$ is given. We construct an instance I of the reflexive list H-edge-coloring problem as follows: to each literal x_i we assign a set of $2t$ edges, each one called *a copy of* x_i, and all have the list $\{0, 1\}$ assigned to them and share a common endpoint (These common endpoints are distinct for different variables). This construction and the fact that the colors 0 and 1 are non-adjacent in H guarantees that in any solution of I, the copies of x_i ($i = 1, 2, \cdots n$) are either all colored with 0 or all colored with 1. We interpret this common color as the value of the variable x_i (0 for FALSE and 1 for TRUE).

Feder et al. in [118] (see Theorem 2.3) construct the following choosers, assuming that the vertices $0, 1, 2$ form an asteroidal triple in H: P which takes 0 to $\{0, 1\}$ and 1 to $\{1, 2\}$, P' which takes 0 to $\{1, 2\}$ and 1 to $\{2, 0\}$ and P'' which takes 0 to $\{2, 0\}$ and 1 to $\{0, 1\}$. By turning vertices to edges in these choosers we can easily turn them to edge-choosers (as any chooser is a path). Let T be a gadget obtained from the edge-choosers P, P' and P'' by identifying their last vertices (i.e., the degree 1 endpoints of the output edges). In T, the input edges of P, P' and P'' are called a, a' and a'', respectively. It is easy to see that T has a reflexive list H-edge-coloring for all pre-coloring combinations of edges a, a', a'' except when they all have either color 0 or color 1. Thus by making a copy of T for each clause and identifying its input edges with the copies of variables of that clause (using each copy of every variable in at most one gadget), we have a polynomial-time reduction of the NAE 3-SAT problem without negated variables to the reflexive list H-edge-coloring problem. □

Now we are in a position to give the proof for the final theorem:

Proof. (Theorem 6.1.2) Feder et al. in [118] proved that the reflexive list H-coloring problem is polynomial when H is an interval graph. The exact running time can be inferred from their proof to be $O(n^2)$. Note that this result includes the reflexive list H-edge-coloring problem as a special case and thus it clearly proves the forward direction of the theorem. Now to prove the backward direction, a result in [164] states that if H is not an interval graph then it must contain either a chordless cycle with 4 or 5 vertices, or an asteroidal triple as induced subgraph. In the former case Lemma 6.1.14 and in the latter case Lemma 6.1.15 imply that the reflexive list H-edge-coloring problem is NP-complete. □

6.2 Quasi-Line Graphs

We consider the case of 0-diagonal matrices first. A result proved in the next section (Theorem 6.3.1) offers a complete dichotomy for the list M-partition problem, for any 0-diagonal matrix M, when the input graphs are restricted to any subclass of claw-free graphs for which the 3-coloring problem is NP-complete (see Observation 6.3.2). This condition holds for quasi-line graphs, since they contain line graphs for which the 3-coloring problem is NP-complete. Thus as a special case of Theorem 6.3.1 we have the following corollary:

Corollary 6.2.1. *Suppose M is a 0-diagonal $k \times k$ matrix. Then the list M-partition problem, restricted to quasi-line graphs, can be solved in polynomial time (of order $O(n^{2k+3})$) if M does not contain 3-coloring, and otherwise it is NP-complete.*

The case of 1-diagonal matrices for quasi-line graphs is more challenging. Let M be a 1-diagonal matrix. Then solving the list M-partition problem for quasi-line graphs is equivalent to solving the list $\overline{B}$-partition problem for co-quasi-line graphs (see Observation 2.7.1). Note that bipartite graphs are co-quasi-line. Thus, if $\overline{M}$ is 1-free (equivalently when M is 0-free) then we can apply Theorem 4.4.1 to obtain the following corollary:

Corollary 6.2.2. *Suppose M is a 1-diagonal 0-free matrix. Then the list M-partition problem, restricted to quasi-line graphs, can be solved in polynomial time (of order $O(n^2)$) if $\overline{B}$ corresponds to a bipartite co-circular arc graph, and otherwise it is NP-complete.*

It is still open to find a similar dichotomy result for the case when M is not necessarily 0-free. Recall that according to Theorem 6.1.1, any 1-diagonal matrix is a polynomial pattern for line graphs. Thus the above corollary introduces patterns which are NP-complete for quasi-line graphs while polynomial for line graphs:

Observation 6.2.3. *For any 1-diagonal 0-free matrix M, for which $\overline{B}$ does not correspond to a bipartite co-circular arc graph, the list M-partition problem is NP-complete for quasi-line graphs, but polynomial for line graphs.*

As for the separation property, Theorem 6.3.5 in the next section states that any matrix M, in which the block C is 0-free, has the separation property for claw-free graphs. This conclusion applies to quasi-line graphs as well, since the separation property is hereditary (see Observation 4.2.1). Also recall that Theorem 4.3.3 offers the separation property when the block C is *-free (for all graphs). Thus combining these facts with Observation 4.2.2 and the last two corollaries leads to the following dichotomy result:

Theorem 6.2.4. *Suppose M is a 01-diagonal matrix in which the block B is 0-free and the block C is 0-free or *-free. Then the list M-partition problem, restricted to quasi-line graphs, can be solved in polynomial time (of order $O(n^{72m^4})$) if M does not contain 3-coloring and the complement of a matrix corresponding to a graph which is not a bipartite co-circular arc graph, and otherwise it is NP-complete.*

As we will see in the next section, the same dichotomy also holds for claw-free graphs. But an extra advantage of quasi-line graphs is that, the structural theorem of Chudnovsky and Seymour for claw-free graphs ([54]) becomes simple and tractable when limited to quasi-line graphs. We use this structural theorem to derive a result analogous to Corollary 6.2.2 for the non-list version:

Theorem 6.2.5. *Suppose M is a 1-diagonal 0-free $l \times l$ matrix. Then the M-partition problem (the non-list version), restricted to quasi-line graphs, can be solved in polynomial time (of order $O(n^{13l+1})$).*

Due to the complexity of the structural theorem in general (i.e., to describe the structure of claw-free graphs), we were not able to follow the same approach for the class of claw-free graphs. The rest of this section is dedicated to proving the above theorem. We explain the structural theorem first.

A *strip* (S, X, Y) is a claw-free graph S with two cliques X and Y such that for any $u \in X$ ($v \in Y$, respectively), the set $N(u) - X$ ($N(v) - Y$, respectively) is a clique in S. Suppose S is a unit interval graph, i.e., an interval graph in which the length of each interval is one unit. Let $v_1, v_2, \cdots v_n$ be the vertices of S in the increasing order based on the mid-point of the corresponding intervals. Suppose the sets $X = \{v_1, \cdots v_p\}$ and $Y = \{v_{n-q+1}, \cdots v_n\}$ (for some integers $0 \leq p, q \leq n$) both induce cliques in S. Then it is easy to see that (S, X, Y) is a strip. This special type of strip is called *unit interval strip*. The concept of *multi-digraph* generalizes the concept of digraph by allowing more than one arc to be drawn between any pair of vertices. Suppose a multi-digraph H with strips (S_e, X_e, Y_e) associated to each arc $e \in E(H)$ are given. For each $v \in V(H)$, define C_v as the union of the sets X_e, for all arcs e coming out of v, and the sets $Y_{e'}$, for all arcs e' coming into v. Construct a graph G from the disjoint union of the graphs S_e, $e \in E(H)$, and making each set C_v, for $v \in V(H)$, a clique (by drawing an edge between any two distinct vertices in C_v). In this case, G is called a *composition of strips*, and H is called its *underlying multi-digraph*. The structural theorem of Chudnovsky and Seymour for quasi-line graphs is as follows:

Theorem 6.2.6. *([54]) Any quasi-line graph containing no homogeneous pair of cliques (defined in Section 2.4) is either a proper circular arc graph or a composition of unit interval strips.*

We use the above theorem to prove Theorem 6.2.5. Before doing so, we need to introduce some tools and lemmas. The general approach should be clear: these tools help to handle the case of G having a homogeneous pair of cliques and the case when G is a composition of unit interval strips. Note that the case of G being a proper circular arc graph is already handled by Theorem 5.3.3 in Chapter 5.

Lemma 6.2.7. *Suppose M is a 1-diagonal 0-free $l \times l$ matrix. Given a quasi-line graph G with a homogeneous pair of cliques (X, Y), let G' be the (quasi-line) graph obtained from G by contracting each set X and Y into a single vertex x and y, respectively, and joining x and y if and only if $X \cup Y$ is a clique (in G). Then G has a M-partition if and only if G' has* *a M-partition. Furthermore, we can obtain a M-partition for G from a given M-partition for G' in $O(n)$ time.*

Proof. If x and y are not adjacent in G', then there must be at least one vertex $v_x \in X$ which is not adjacent to some vertex $v_y \in Y$. For the other case in which x and y are adjacent in G', simply let v_x and v_y be arbitrary vertices in X and Y, respectively. Suppose a M-partition f of G is given. Define a coloring f' of G' as follows: $f'(u) = f(u)$ for $u \in V(G') - \{x, y\}$, $f'(x) = f(v_x)$ and $f'(y) = f(v_y)$. It is easy to check that f' is a M-partition for G'. Now suppose a M-partition f' for G' is given. Define the coloring f for G as follows: $f(u) = f'(u)$ for $u \in V(G) - X \cup Y$, $f(u) = f'(x)$ for $u \in X$ and $f(u) = f'(y)$ for $u \in Y$. Again it is easy to check that f is a M-partition for G (Note that $M(f'(x), f'(y)) = *$ if x and y are non-adjacent in G'.) □

The above lemma allows us to remove the homogeneous pairs of cliques by contracting each clique into a vertex. A result of Everett et al. [115] provides an $O(n^5)$ time algorithm to find a homogeneous pair of cliques in any graph or decide that none exists. Using this algorithm and applying the above lemma repeatedly to the homogeneous pair of cliques in each step, we will be able to make the graph free from such pairs. Thus we have the following tool:

Lemma 6.2.8. *Suppose M is a 1-diagonal 0-free $l \times l$ matrix. Given a graph G, we can obtain an induced subgraph G' of G in $O(n^6)$ time with no homogeneous pair of cliques such that G has a M-partition if and only if G' has a M-partition. Furthermore, we can obtain a M-partition for G from a given M-partition for G' in $O(n)$ time.*

This way we can get rid of homogeneous pairs of cliques in a quasi-line graph, and this paves the way for applying Theorem 6.2.6. So now we discuss how to handle the case in which G is a composition of unit interval strips.

A unit interval strip (S, X, Y) is called a *bi-interval* strip if $V(S) = X \cup Y$. Suppose a multi-digraph H is composed of two vertices u and v and $r \geq 0$ arcs e_i from u to v $(1 \leq i \leq r)$. To each arc e_i we assign a bi-interval strip (S_i, X_i, Y_i). Then the strip (S, X, Y), where S is the composition of bi-interval strips (S_i, X_i, Y_i) $(i = 1, 2, \cdots r)$ with the underlying multi-digraph H and $X = \cup_{i=1}^{r} X_i$ and $Y = \cup_{i=1}^{r} Y_i$, is called a *multi-interval strip*.

Lemma 6.2.9. *Suppose for some fixed integer $l > 0$, a graph G does not contain any stable set of size $l+1$. If G is a composition of unit interval strips with the underlying multi-digraph H then it is also a composition of unit interval and multi-interval strips with an underlying multi-digraph H^* which has at most $(l+2)^2 + l$ arcs. Furthermore, both H^* and its corresponding strips can be found in $O(n^2)$ time, assuming that the digraph H and the unit interval strips corresponding to G are given as part of the input.*

Proof. Without loss of generality, assume that $V(S_e) \neq \emptyset$ for any $e \in E(H)$, for otherwise we can remove e from H. Let E_1 be the set of arcs $e \in E(H)$ with (S_e, X_e, Y_e) not being a bi-interval strip. Note that $|E_1| \leq l$, for otherwise by picking an arbitrary vertex from each set $V(S_e) - V(X_e) \cup V(Y_e)$ $(e \in E_1)$, we will produce a stable set larger than l in G, which contradicts our assumptions.

Let H_2 be the multi-digraph obtained from H by removing the arcs in E_1. Add two new vertices u_X and v_Y to H_2 and replace any arc $e = (u, v) \in E(H_2)$ in which $X_e = \emptyset$ ($Y_e = \emptyset$, respectively) with the arc $e' = (u_X, v)$ ($e' = (u, v_Y)$, respectively), and define $S_{e'} = S_e$. Call the new multi-arc digraph H_3. Clearly H_3 has the same number of arcs and the same strip composition as H_2, and it has the additional property that all arcs $e \in E(H_2)$ with $X_e = \emptyset$ ($Y_e = \emptyset$, respectively) will begin with (end in, respectively) the vertex u_X (v_Y, respectively). Note that any isolated vertex in H_3 will have no effect in its strip composition, and thus we may remove all such vertices. Now each vertex $v \in V(H_3) - \{v_X, u_Y\}$ will be the end point of some arc e_v, and thus, in the strip composition, we will have X_{e_v} or Y_{e_v} as a subset of C_v. This implies that $C_v \neq \emptyset$. Now picking an arbitrary vertex from each set C_v produces a stable set in G of size $|V(H_3)| - 2$. The fact that G has no stable set of size $l+1$ implies that $|V(H_3)| \leq l+2$.

For any two vertices $u, v \in V(H_3)$ define $A(u, v)$ as the set of of arcs from u to v. Replace all arcs (if any) in $A(u, v)$ with a new arc e^* (from u to v) with the corresponding multi-interval strip $(S_{e^*}, X_{e^*}, Y_{e^*})$ where S_{e^*} is the induced graph of $\cup_{e \in A(u,v)} S_e$ in G, $X_{e^*} = \cup_{e \in A(u,v)} X_e$ and $Y_{e^*} = \cup_{e \in A(u,v)} Y_e$. Let H_4 be the multi-digraph obtained by these replacements. Again it is easy to check that the final strip composition of H_4 is the same as H_3, however H_4 has no multi-arcs, i.e., two distinct arcs with the same end-points and orientation. This implies that $|E(H_4)| \leq |V(H_4)|^2 \leq (l+2)^2$. Add E_1 to H_4 along with their original corresponding unit interval strips. Call the new multi-digraph H^*. Note that H^* has the same strip composition as H (namely the graph G), and also $|E(H^*)| \leq |E(H_4)| + |E_1| \leq (l+2)^2 + l$. All these steps take $O(n^2)$. □

Lemma 6.2.10. *Suppose M is a 1-diagonal 0-free $l \times l$ matrix. Let (S, X, Y) be an arbitrary multi-interval strip whose unit interval strips are given, and let L be a list function for S defined as follows: $L(u) = L_1$ for any vertex $u \in X$ and $L(u) = L_2$ for $u \in Y$, where $L_1, L_2 \subseteq [l]$ are two arbitrary sets of colors. Then the instance $I = (S, L)$ of the list M-partition problem can be solved in $O(n^2)$ time.*

Proof. Suppose there are non-adjacent vertices $x_0 \in X$ and $y_0 \in Y$. Then there must be colors $c_1 \in L_1$ and $c_2 \in L_2$ such that $M(c_1, c_2) = *$, otherwise we cannot color x_0 and y_0 in any list M-partition. In this case, the coloring f defined as $f(x) = c_1$ for all $x \in X$ and $f(y) = c_2$ for all $y \in Y$, is an obvious solution of I and we are done. So suppose there are no non-adjacent vertices $x_0 \in X$ and $y_0 \in Y$. (This can be checked in $O(n^2)$ time.) This means S is a clique, and thus a cograph, and all list M-partition problems can be solved in linear time for cographs [123]. □

Now we show how to solve the M-partition problem for compositions of unit interval strips:

Lemma 6.2.11. *Suppose M is a 1-diagonal 0-free $l \times l$ matrix. Then the M-partition problem for the graphs which are the compositions of unit interval strips can be solved in polynomial time (of order $O(n^{2l+2})$).*

Proof. Let G be a composition of unit interval strips. The underlying multi-digraph of G and the unit interval strips corresponding to each arc can be found in $O(n^5)$ time according to [210]. We can check in $O(n^{l+3})$ time whether G contains a stable set of size $l + 1$. If so then G has no M-partition and we can terminate the algorithm right away. Otherwise according to Lemma 6.2.9, G will be a composition of unit interval and multi-interval strips with an underlying multi-digraph H^* such that $|E(H^*)| \leq (l + 2)^2 + l = O(1)$, and both H^* and strips S_e (for $e \in E(H^*)$) can be found in $O(n^2)$ time. Suppose a M-partition f of G is given. For each strip S_e ($e \in E(H^*)$) define L_e^1, L_e^2 and L_e^3 as the set of colors used by f for at least one vertex in the set $X_e, S_e - X_e \cup Y_e$ and Y_e, respectively. These sets have the following properties:

Condition 1: For any $e \in E(H^*)$, the instance $I_e = (S_e, L_e)$ of the list M-partition problem has a solution, where $L_e(u)$ is either L_e^1, L_e^2 or L_e^3 depending on whether u is in $X_e, S_e - X_e \cup Y_e$ or Y_e, respectively.

Condition 2: for any two distinct arcs $e_1, e_2 \in E(H^*)$, any two indices $i, j \in \{1, 2, 3\}$ and any two colors $c_1 \in L^i_{e_1}$ and $c_2 \in L^j_{e_2}$, we must have $M(c_1, c_2) = *$, except the following cases for which there is no restriction on $M(c_1, c_2)$:

1. $i = j = 1$, and e_1 and e_2 have a common starting vertex.
2. $i = j = 3$, and e_1 and e_2 have a common end vertex.
3. $i = 1, j = 3$, and the start vertex of e_1 is the same as the end vertex of e_2.
4. $i = 3, j = 1$, and the end vertex of e_1 is the same as the start vertex of e_2.

It is easy to check that the reverse is also true. In other words, given the sets $L^1_e, L^2_e, L^1_e \in [l]$ assigned to each arc $e \in E(H^*)$, if the above two conditions hold then we can construct a M-partition for G by solving the instances $I_e = (S_e, L_e)$ independently, for $e \in E(H^*)$. Thus the task of finding a M-partition for G is reduced to finding the sets L^1_e, L^2_e, L^1_e ($e \in E(H^*)$) with the above conditions. Since $|E(H^*)| = O(1)$, the total number of choices for all sets L^i_e, $i = 1, 2, 3$, $e \in E(H^*)$, is $O(1)$. We branch over all possible choices. For each choice, we have to make sure that the above two conditions hold: for the first condition, note that the strip (S_e, X_e, Y_e) ($e \in E(H^*)$) is either a unit interval or a multi-interval strip. In the former case apply Theorem 5.2.4 and in the latter case apply Lemma 6.2.10 to conclude that that the first condition can be checked in $O(n^{2l+2})$ time. The second condition can be checked easily in $O(1)$ time. The solutions that we found by solving the instances I_e (when checking the first condition) will constitute the final solution. □

Now we can give the proof of Theorem 6.2.5:

Proof. By Lemma 6.2.8, we can obtain a quasi-line graph G' in $O(n^6)$ time which has no homogeneous pair of cliques. Thus based on Theorem 6.2.6, G' is either a proper circular arc graph or a composition of unit interval strips. We can check in $O(n)$ time whether G is a proper circular arc graph ([103]). If so then by applying Theorem 5.3.3 we can find a M-partition for G (or decide its non-existence) in $O(n^{13l})$ time, otherwise, we can apply Lemma 6.2.11 and the theorem follows. □

6.3 Claw-free Graphs

For the 0-diagonal matrices we offer the following complexity result:

Theorem 6.3.1. *Suppose M is a 0-diagonal $k \times k$ matrix which does not contain 3-coloring. Then the list M-partition problem, restricted to claw-free graphs, can be solved in polynomial time (of order $O(n^{2k+3})$).*

Proof. Let $I = (G, L)$ be an instance of the list M-partition problem, in which G is a claw-free. Here is the intuitive framework of this proof: observe that if $M(c, c') = 1$, for two distinct colors c and c', then in any solution of I, at least one of these colors should be used for at most two vertices (otherwise G will contain a claw which is contrary to our assumption). Using this observation, we break I into sub-instances in order to get rid of all the colors which are used at most twice. This way, in each sub-instance we will also get rid of 1 entries in M which simplifies the problem to a great extent. To put it formally, we define several parameters for any solution f of I: define D_f as the set of colors used by f for at most two vertices. Define $g_f : D_f \to 2^{V(G)}$ as a function in which $g_f(c)$ is the set of vertices v (possibly empty) colored with c (so, $f(v) = c$), for $c \in D_f$. Obviously $g_f(c) \cap g_f(c') = \emptyset$ for any two distinct colors $c, c' \in D_f$. Note that for any two distinct colors $c, c' \notin D_f$ we must have $M(c, c') \neq 1$, otherwise the vertices colored by c and c' will contain an induced subgraph isomorphic to a claw as each color c and c' is used for at least three vertices. But this is contrary to our assumption that G is claw-free.

Let J_f be the pair (D_f, g_f) and $\widetilde{J}$ be the set of all possible choices for such pairs. More precisely, to generate all the pairs $J = (D, g)$ in $\widetilde{J}$ we consider:

1. All subsets $D \subseteq [m]$ such that $M(c, c') \neq 1$ for any two distinct colors $c, c' \notin D$,
2. All functions $g : D \to 2^{V(G)}$ with $|g(c)| \leq 2$ and $g(c) \cap g(c') = \emptyset$ for any two distinct colors $c, c' \in D$.

Note that the set $\widetilde{J}$ contains the pairs J_f for all possible solutions f of I, and it can be generated in $O(n^{2k})$ time (since we have $O(1)$ many choices for the subset D and $O(n^2)$ many choices for each value $g(c)$).

Now for each pair $J = (D, g) \in \widetilde{J}$, define I_J as a sub-instance of I which is obtained by applying the constraints imposed by the pair J. More precisely, for each color $c \in D$, pre-color all the vertices in $g(c)$ with c (in the instance I) and remove c from the lists of all other vertices (i.e., those not in the set $g(c)$). It is easy to see that any solution f of I is a solution of the sub-instance I_{J_f} as well. This implies that I is broken into sub-instances I_J, for $J \in \widetilde{J}$. Moreover, we can construct each sub-instance I_J in $O(n)$ time. Thus all such sub-instances can be generated in $O(n^{2k+1})$ time.

Now we show how to solve a sub-instance I_J. We first attempt to get rid of the pre-colored vertices (i.e., the vertices in $\cup_{c \in D} g(c)$). We do so by updating the lists based on the set of the pre-colored vertices and then eliminate these vertices from I_J. Call the new instance $I' = (G', L')$. Note that I' is an instance of the list M'-partition problem, where M' is the principal sub-matrix of M obtained by considering only the colors in $[m] - D$. Note that M' is 1-free due to the condition we imposed on the set D. We can check in $O(n^3)$ time whether G' contains a triangle with vertices v_1, v_2, v_3. If it does then I' has no solution, since any solution f' requires that the colors $f'(v_i)$ $(i = 1, 2, 3)$ induce a sub-matrix identical to 3-coloring in M' which contradicts our assumption. Thus we may assume that G' is triangle-free. Recall that G' is also claw-free. This implies that G' cannot contain any vertex with degree more than two, and thus it is a union of paths and cycles. Since M' is 1-free, by applying Observation 6.1.9, we can further reduce solving I' to solving the instances (G'', L') of the list M'-partition problem for all connected components G'' of G' which is either a path or cycle. It is easy to see that the clique-width of any path or cycle is ≤ 3, and the k-expression constructing it can be found in linear time. Thus by applying Corollary 1.3.3, each instance (G'', L') can be solved in linear time and the theorem follows. □

Observation 6.3.2. *Theorem 6.3.1 yields a dichotomy result for the list M-partition problem when M is a 0-diagonal $k \times k$ matrix and the input graph is restricted to claw-free graphs or any subclass of claw-free graphs for which the 3-coloring problem is NP-complete. The dichotomy is as follows: the problem can be solved in polynomial time (of order $O(n^{2k+3})$) if M does not contain 3-coloring, and otherwise it is NP-complete.*

Considering the special case of the irreflexive list H-coloring problem (of the list M-partition problem) leads to the following dichotomy result for H-coloring of claw-free graphs:

Corollary 6.3.3. *Suppose H is a graph without loops on k vertices. If H contains no triangle then the list H-coloring problem for claw-free graphs can be solved in polynomial time (of order $O(n^{2k+3})$).*

The list M-partition problem when M is a 1-diagonal matrix is equivalent to the list $\overline{M}$-partition problem for co-claw-free graphs (see Observation 2.7.1). Note that all bipartite graphs are co-claw-free. Thus if $\overline{M}$ is 1-free (equivalently when M is 0-free) we can apply Theorem 4.4.1 to obtain the following result:

Corollary 6.3.4. *Suppose M is a 1-diagonal 0-free matrix. Then the list M-partition problem for claw-free graphs can be solved in polynomial time (of order $O(n^2)$) if $\overline{B}$ corresponds to a bipartite co-circular arc graph, and otherwise it is NP-complete.*

It is still open to find a similar dichotomy result for the case when M is not necessarily 0-free.

Now we focus on the separation property for claw-free graphs. We conjecture that any matrix not containing 3-coloring has the separation property for claw-free graphs. We prove this conjecture with the additional assumption that the block C (see Figure 1.2) is 0-free:

Theorem 6.3.5. *Let M be a 01-diagonal matrix which does not contain 3-coloring and its block C is 0-free. Then M has the separation property for the class of claw-free graphs. Moreover, there exists an algorithm giving the separation property with running time $O(n^{18klm})$, where k and l are the sizes of the blocks A and B, respectively.*

Proof. By applying Theorem 4.3.4, it is enough to find a polynomial-time algorithm ALG which, given any instance $I = (G, L)$ of the list M-partition problem with G being a claw-free and a split pair (V_Q, V_S) in I, breaks I into sub-instances in which the sets V_Q and V_S are separated. The intuition of this proof is very similar to that of Theorem 6.3.1: observe that if $M(c, c') = 1$ for a 0-color c and a 1-color c' then in any solution of I either the color c is used at most twice or the color c' is not used at all (otherwise G will contain a claw which is contrary to our assumption). Using this observation, we break I into sub-instance in order to get rid of all the colors which are used at most twice. This way, in each sub-instance we will also get rid of 1 entries in the block C which makes it all $*$, and clearly implies that the sets V_Q and V_S are separated. To put it formally, we define several parameters for any solution f of I: define S_f as the set of 0-colors used by f for at most two vertices in V_S. Define Q_f as the set of 1-colors used by f for at least one vertex in V_Q. Define $g_f : S_f \to 2^{V_S}$ as a function in which $g_f(c)$ is the set of vertices $v \in V_S$ (possibly empty) colored with c (so, $f(v) = c$), for $c \in S_f$. Obviously $g_f(c) \cap g_f(c') = \emptyset$ for any two distinct colors $c, c' \in S_f$. Suppose for a color $c \notin S_f$ and $c' \in Q_f$ we have $M(c, c') = 1$. Then any three distinct vertices colored with c and any vertex colored with c' (in f) will induce a claw in G (note that V_S is a stable set), which is a contradiction. Keeping in mind that the block C is 0-free, we may conclude the important property that for any color $c \notin S_f$ and $c' \in Q_f$ we have $M(c, c') = *$.

Let J_f be the tuple (S_f, Q_f, g_f) and $\widetilde{J}$ be the set of all possible choices for such tuples. More precisely, to generate all tuples $J = (S, Q, g)$ in $\widetilde{J}$ we consider:

1. All subsets $S, Q \subseteq [m]$ such that $M(c, c') = *$ for all colors $c \notin S$ and $c' \in Q$,

2. All functions $g : S \to 2^{V_S}$ with $|g(c)| \leq 2$ and $g(c) \cap g(c') = \emptyset$ for any two distinct colors $c, c' \in S$.

Note that the set $\widetilde{J}$ contains the tuples J_f for all possible solutions f of I, and it can be generated in $O(n^{2k})$ time (since we have $O(1)$ many choices for each subset S, Q, and we have $O(n^2)$ many choices for each value $g_f(c)$, $c \in S$).

Now for each tuple $J = (S, Q, g) \in \widetilde{J}$, define I_J as a sub-instance of I which is obtained by applying the constraints imposed by the tuple J. More precisely, apply the following procedure to the instance I to obtain I_J:

1. For each color $c \in S$, pre-color all the vertices in $g(c)$ with c and remove c from the lists of all other vertices, i.e., those not in the set $g(c)$. (This can be done in $O(n)$ time.)

2. For any vertex $v \in V_Q$, remove from the list $L(v)$ any color which is not in Q. (This can be done in $O(n)$ time.)

In order to get rid of the pre-colored vertices (i.e., vertices in $\cup_{c \in S} g(c)$) we update the lists based on these vertices. This way, any pre-colored vertex becomes separated from all other vertices. As for other vertices, note that the property that $M(c, c') = *$ for any color $c \notin S$ and $c' \in Q$ implies that any vertex in V_S which is not pre-colored is separated from any vertex in V_Q. Thus the sets V_Q and V_S are separated in the sub-instance I_J. Also note that any solution f of I is a solution of the sub-instance I_{J_f}, which implies that I is broken into sub-instances I_J, for $J \in \widetilde{J}$. Moreover, we can construct each sub-instance I_J in $O(n^2)$ time. Thus all such sub-instances can be generated in $O(n^{2k+2})$ time. $\square$

Recall that Theorem 4.3.3 offers the separation property when the block C is *-free (for all graphs). Combining this fact, Theorem 6.3.5 above, Observation 4.2.2, Theorem 6.3.1 and Corollary 6.3.4, we obtain the following dichotomy:

Theorem 6.3.6. *Suppose M is a 01-diagonal matrix in which the block B is 0-free and the block C is 0-free or *-free. Then the list M-partition problem, restricted to claw-free graphs,* *can be solved in polynomial time (of order $O(n^{72m^4})$) if M does not contain 3-coloring or the complement of a matrix corresponding to a graph which is not a bipartite co-circular arc graph, and otherwise it is NP-complete.*

Observe that the above dichotomy is the same as the one given in Theorem 6.2.4 for quasi-line graphs. This suggests that for the family of matrices M studied in these two theorems, the extension from quasi-line graphs to claw-free graphs does not change the dichotomy of list M-partition problems. It would be interesting to further investigate if this is the case for other matrices as well. In other words, if there is any matrix M for which the list M-partition problem is polynomial for quasi-line graphs, but NP-complete for claw-free graphs.

Chapter 7

Graph Classes with Forbidden Subgraphs

In this chapter we study the M-partition problem for the graph classes which can be characterized by a set of forbidden subgraphs. Following the literature, we assume that all graphs in this chapter are without loops. (Refer to Section 3.3 for more detail on the literature of these graphs.) Here we should mention that the class of bipartite graphs is one example of a class defined by a set of infinitely many forbidden subgraphs, namely odd cycles. We already gave a partial dichotomy result for this class in Chapter 4, which covers 0-diagonal 1-free matrices (see Theorem 4.4.1). For the same group of matrices, the same dichotomy holds for general graphs according to [124]. This observation makes us suspect that analyzing the list M-partition problem for general bipartite graphs could be as difficult as solving this problem for general graphs. Thus we mainly focus on the case when the set of forbidden subgraphs is finite, particularly when it has a single member H (namely, the class of H-free graphs).

In the first section we consider the class of H-free graphs for arbitrary graph H. First, we prove some partial dichotomy results which indicate that many M-partition problems for H-free graphs have the same complexity as when limited to bipartite graphs (Theorems 7.1.1 and 7.1.1). These results will be used in the subsequent sections. Also recall that all list M-partition problems are polynomial for the class of P_4-free graphs (= cographs), according to [180, 123]. We prove that for all graphs H different from P_4 and its induced subgraphs, there are some NP-complete M-partition problems for the class of H-free graphs (Corollary

7.1.4). Recall that identifying graph classes for which all list M-partition problems are polynomial is an existing research direction (see Section 1.1).

In Section 7.2 and 7.3 we focus on two special cases of H-free graphs, namely P_5-free and bull-free graphs. Recall that both the P_5 and the bull are one-vertex extensions of P_4, and the importance of these graph classes in studying the vertex coloring and the k-coloring problems has been discussed in Section 3.3. We offer some partial dichotomy results for these classes (Theorems 7.2.4 and 7.2.5 for P_5-free graphs, and Theorem 7.3.2 for bull-free graphs). We recall that, in Section 6.3, we have already studied another example of H-free graphs, namely claw-free graphs. (Refer to that section for more details.)

7.1 H-free Graphs

In this section we study the complexity of the list M-partition problem for the class of H-free graphs, mainly focusing on identifying those classes for which all M-partition problems are polynomial.

Note that if H is non-bipartite then H-free graphs will contain bipartite graphs as subclass. So applying Theorem 4.4.1 yields the following result:

Theorem 7.1.1. *Let H be a non-bipartite graph. Suppose M is a 0-diagonal 1-free matrix. Then the list M-partition problem, restricted to H-free graphs, can be solved in polynomial time (of order $O(n^2)$) if M corresponds to a bipartite co-circular arc graph, and otherwise it is NP-complete.*

Similarly, when H is non-co-bipartite, by applying Theorem 4.4.1 to the complement we obtain the following dichotomy:

Theorem 7.1.2. *Let H be a non-co-bipartite graph. Suppose M is a 1-diagonal 0-free matrix. Then the list M-partition problem, restricted to H-free graphs, can be solved in polynomial time (of order $O(n^2)$) if $\overline{B}$ corresponds to a bipartite co-circular arc graph, and otherwise it is NP-complete.*

Theorems 7.1.1 and 7.1.1 are useful tools for analyzing the complexity of the list M-partition problem for H-free graphs. They will be used in the next sections for studying P_5-free and bull-free graphs. Another implication of the above theorems is that there are NP-complete patterns for H-free graphs when H is either non-bipartite or non-co-bipartite.

Therefore, we now focus on those graphs H not included in these theorems, namely when both H and its complement are bipartite. A simple application of Ramsey theorem (see [25]) along with some elementary arguments imply that such graphs H are limited to a few specific graphs, namely C_4, P_4, P_3, P_2 and the complement of these graphs. The cases of P_2, P_3, P_4 and their complements all lead to subclasses of cographs, as cographs are P_4-free. Thus the list M-partition problem is polynomial for these classes according to [123]. As for C_4, note that its complement is an induced subgraph of P_5. Using a result proved in the next section (Lemma 7.2.1), we obtain the following corollary:

Corollary 7.1.3. *Suppose M is a 1-diagonal $l \times l$ matrix. Then the list M-partition problem, restricted to C_4-free graphs, can be solved in polynomial time (of order $O(n^{l^6})$).*

The case of the list M-partition problem for C_4-free graphs when M is a 0-diagonal matrix, is left as an open problem. Nevertheless recall that the 3-coloring problem is NP-complete for this class (according to [206]). So as a corollary of this analysis we may conclude that:

Corollary 7.1.4. *Let H be an arbitrary graph. If H is not P_4 or any of its induced subgraphs then there exist NP-complete list M-partition problems, even when restricted to H-free graphs, otherwise all list M-partition problems, restricted to H-free graphs, are polynomial.*

7.2 P_5-free Graphs

Since the graph P_5 is non-co-bipartite, Theorem 7.1.2 offers a complete dichotomy for the list M-partition problem for P_5-free graphs when M is a 1-diagonal 0-free matrix. Now we focus on the case of 0-diagonal matrices. Hoàng et al. [196] proved that the k-coloring problem can be solved in polynomial time for P_5-free graphs. Their approach can be easily extended to any 0-diagonal matrix:

Lemma 7.2.1. *Suppose M is a 0-diagonal $k \times k$ matrix. Then the list M-partition problem, restricted to P_5-free graphs, can be solved in polynomial time (of order $O(n^{k^6})$).*

Before proving this result, we need the following two lemmas as tools:

Lemma 7.2.2. *Let $\widetilde{G}$ be a hereditary graph class. Suppose we are given an algorithm to solve any list M-partition problem, restricted to the connected graphs in $\widetilde{G}$, in $f(n)$ time. Then the list M-partition problem, restricted to $\widetilde{G}$, can be solved in $O(n \cdot f(n) + n^2)$ time.*

Proof. Let $I = (G, L)$ be an arbitrary instance of the list M-partition problem with $G \in \widetilde{G}$. Let $C_1, C_2, \cdots C_r$ be the connected components of G. We can find these connected components in $O(|E(G)|)$ time (see [25]). For $i = 1, 2, \cdots r$ and a non-empty subset $S \subseteq [m]$, define $I_{(i,S)}$ as the instance $(G[C_i], L_S)$ of the list M-partition problem where $L_S(u) = L(u) \cap S$ $(u \in V(C_i))$. Let f be a solution of I, and S_i be the set of colors used by f for at least one vertex in the component C_i. It is easy to see that these sets satisfy the following two conditions:

1. For any two indices $1 \leq i \neq j \leq r$, there are no colors $c \in S_i$ and $c' \in S_j$ with $M(c, c') = 1$,

2. The instance $I_{(i,S_i)}$ has a solution for $i = 1, 2, \cdots r$ (which is the solution f).

Note that the reverse is also true, i.e., given subsets $S_i \subseteq [m]$ satisfying the above conditions, we can find a solution for I by putting the solutions of the instances $I_{(i,S_i)}$ together. This can be performed in $O(n \cdot f(n))$ time. This reduces our task to finding such subsets S_i.

To do so, let M' be a $(2^m - 1) \times (2^m - 1)$ symmetric matrix whose rows and columns correspond to non-empty subsets of $[m]$. For any two non-empty subsets $S, T \subseteq [m]$, the entry $M(S, T)$ is defined to be * if and only if for any two colors $c \in S$ and $c' \in T$ we have $M(c, c') \neq 1$, otherwise it is defined to be 0. Denote by K_r the complete graph on r vertices $w_1, w_2, \cdots w_r$. Define the list L' for K_r as follows: $L'(w_i)$ is the set of all subsets $S \subseteq [m]$ for which the instance $I_{(i,S)}$ has a solution. An easy argument shows that the sets S_i have the aforementioned two conditions if and only if they induce a solution for the instance $I' = (K_r, L')$ of the list M'-partition problem (by defining S_i to be the color of the vertex w_i). Thus it is sufficient to solve the instance I'.

Note that the matrix M' can be constructed in $O(1)$ time. Also each list $L'(w_i)$ can be constructed by solving the instances $I_{(i,S)}$ for all non-empty subsets $S \subseteq [m]$, which can be performed in $O(f(n))$ time. Thus the instance I' can be constructed in $O(n \cdot f(n))$ time. Note that K_r is a complete graph which is a special type of cographs. Recall that any list M'-partition problem can be solved in linear time for the class of cographs ([180, 123]). Thus I' (once constructed) can be solved in $O(n)$ time and the lemma follows. □

The concept of *chain graph* was first defined in [281]. We say that a pair (X, Y) of two disjoint non-empty subsets of $V(G)$ *induces a chain graph* in G if there exists an ordering

$y_1, y_2, \cdots y_{|Y|}$ of Y such that the set of neighborhood of y_i in the set X contains the neighborhood of y_{i+1} in the set X (so, $N(y_{i+1}) \cap X \subseteq N(y_i) \cap X$), $i = 1, 2, \cdots |Y| - 1$. This ordering of Y is called the *chain ordering* of the set Y with respect to the set X.

Lemma 7.2.3. *There exists a polynomial-time algorithm (of order $O(n^{2m+1})$) which, given an instance $I = (G, L)$ of the list M-partition problem and a pair (X, Y) inducing a chain graph in G as input, breaks I into sub-instances in which the sets X and Y are separated.*

Proof. Let $y_1, y_2, \cdots y_{|Y|}$ be a chain ordering of Y with respect to X. The definition of chain graph implies that the neighbourhood of any vertex $x \in X$ in Y consists of vertices $y_1, y_2, \cdots y_{d_x}$ for some integer $0 \leq d_x \leq |Y|$. The case of $d_x = 0$ is interpreted as x being adjacent to no vertex in Y. We define a set of ranges for each color in any solution f of I as follows: let p_f, q_f be the functions such that for each color $c \in [m]$, $p_f(c)$ ($q_f(c)$, respectively) is the largest index $r \geq 0$ for which the color c does not occur in the first (the last, respectively) r vertices of Y (So $f(y_i) \neq c$ for $i \in [1, p_f(c)]$ and $i \in [|Y|+1-q_f(c), |Y|]$.) Let J_f be the pair (p_f, q_f) and $\widetilde{J}$ be the set of all possible choices for such pairs. More precisely, to generate all the pairs (p, q) in $\widetilde{J}$, we consider all integers $0 \leq p(c), q(c) \leq |Y|$, for $c \in [m]$. Note that $\widetilde{J}$ contains the pairs J_f for all possible solutions f of I (However, it may contain pairs not corresponding to any solution.) Nevertheless, we can generate all pairs of functions in $\widetilde{J}$ in $O(|Y|^{2m})$ time. Now for each pair $J = (p, q) \in \widetilde{J}$, define I_J as a sub-instance of I which is obtained by applying the constraints imposed by the pair J. More precisely, for each color $c \in [m]$, remove c from the list of the first $p(c)$ vertices and the last $q(c)$ vertices of the set Y, which can be done in $O(|Y|)$ time. Note that for any solution f of I, we have the following property: for any two (not necessarily distinct) colors c and c' with $M(c, c') = 0$ ($M(c, c') = 1$, respectively) and any vertex $x \in X$ with $c' \in L(x)$, if $d_x > p_f(c)$ ($|Y| - d_x > q_f(c)$, respectively) then $f(x) \neq c'$. Now based on this property we apply the following changes to any sub-instance I_J, $J = (p, q) \in \widetilde{J}$: for any two (not necessarily distinct) colors c and c' with $M(c, c') = 0$ ($M(c, c') = 1$, respectively) and any vertex $x \in X$ with $c' \in L(x)$, if $d_x > p(c)$ ($|Y| - d_x > q(c)$, respectively) then remove c' from the list $L(x)$. All these changes can be performed in $O(|X|)$ time. It is easy to see that after these changes, the sets X and Y are separated in the sub-instance I_J, and that any solution f of I is a solution of the sub-instance I_{J_f} as well. This implies that I is broken into sub-instances I_J, for $J \in \widetilde{J}$. Moreover, we can construct each sub-instance I_J in $O(n)$ time. Thus all such sub-instances can be generated in $O(n^{2m+1})$ time. □

This lemma is similar to Lemma 2 in [196], in which the same conclusion is made for the case of the k-coloring problem (a special case of the M-partition problem). So the above lemma is in fact a generalization of this result, however, we employed a different approach to prove it.

Now we are ready to present the proof of Lemma 7.2.1 which is based on the same approach and methods used in [196]:

Proof. Let $I = (G, L)$ be an arbitrary instance of the list M-partition problem, in which G is P_5-free. By applying Lemma 7.2.2, we may assume that G is connected. A set $D \subseteq V(G)$ is called a dominating set if any vertex $v \in V(G) - D$ is adjacent to at least one vertex in D. It is known that every connected P_5-free graph has either a clique or an an induced P_3 subgraph which is a dominating set (Theorem 8 in [8]). Let k be the size of the matrix M. Note that if G contains a clique of size $k + 1$ then clearly I has no solution. This can be checked in $O(n^{k+1})$ time. Otherwise we may deduce that G has a dominating set D with size $\leq k$, which can be found in $O(n^{k+1})$ time.

Let $d_1, d_2, \cdots d_{|D|}$ be an arbitrary ordering of set D. Define F_1 as the set of the neighborhood of the vertex d_1 in the set $V(G) - D$. Then define F_2 as the neighborhood of the vertex d_2 in the set $V(G) - D - F_1$, and continue this process where in step $1 < i \leq |D|$, define F_i as the set of the neighborhood of the vertex d_i in the set $V(G) - D - \cup_{j=1}^{i-1} F_j$. Note that the sets F_i form a partition of the set $V(G) - D$, and can be found in $O(n)$ time.

The instance I can be broken into all possible pre-colorings of the set D. Note that since $|D| \leq k$, all these pre-colorings can be generated in $O(1)$ time. For this reason, without the loss of generality, we may assume that in the instance I, the set D is already pre-colored (So the lists of all the vertices in D are singleton.) We also update the lists based on D (i.e., for any vertex $v_0 \in D$ and $v \in V(G)$, any color in $L(v)$ which conflicts with the only color in $L(v_0)$ will be removed, see Section 4.1). This ensures that the set D and $V(G) - D$ are separated in the instance I. This updating can be done in $O(n)$ time.

Note that by limiting any solution f of I to the set F_i, we get an M-partition for the graph $G[F_i]$. On the other hand, since the single color in the list $L(d_i)$ is not used in this M-partition, we may conclude that I has a solution only if each graph $G[F_i]$, $i = 1, 2, \cdots |D|$, has a list M_i-partition, where M_i is obtained from M by removing the color in the list $L(d_i)$ from it. Since M_i has a smaller size than M, we can handle the M_i-partition problem using recursion: define $f(n, k)$ $(n, k \geq 1)$ as the running time needed to solve any instance of the list M-partition problem for any 0-diagonal matrix M of size k and any P_5-free graph with n vertices. This implies that for each graph $G[F_i]$, we find an M_i-partition or decide that none exists in $O(f(n, k-1))$ time. If some graph $G[F_i]$ has no M_i-partition then I also will have no solution and we can terminate the algorithm right away. Otherwise, each set F_i is partitioned into classes $X_1^i, X_2^i, \cdots X_{k-1}^i$ such that each class is a (possibly empty) stable set.

An important property of these classes is that any two non-empty classes X_p^i and X_q^j $(1 \leq i < j \leq |D|, 1 \leq p, q < k)$ belonging to two distinct sets F_i and F_j induce a chain graph in G. To prove this, suppose (X_p^i, X_q^j) does not induce a chain graph in G. This implies that there exist distinct vertices $u, v \in X_p^i$ and distinct vertices $u', v' \in X_q^j$ such that u is adjacent to u', but not to v', and v is adjacent to v' but not to u'. Recall that the definition of the sets F_i implies that the vertex d_i is adjacent to u and v but not to u' and v'. This means the vertices d_i, u, v, u', v' induce a P_5 in G which is contrary to the assumption that G is P_5-free.

Let $P_1, P_2, \cdots P_r$ be an arbitrary ordering of all the pairs (X_p^i, X_q^j), for $1 \leq i < j \leq |D|, 1 \leq p, q \leq k-1$, for which both classes X_p^i and X_q^j are non-empty. Note that each pair P_i induces a chain graph in G. So by repeatedly applying Lemma 7.2.3 to each pair P_i, $i = 1, 2, \cdots r$, we can break I into sub-instances in which the sets in each pair are separated. More precisely, we perform the following procedure to break I:

1. Initially set $\widetilde{I} = \{I\}$,

2. For $i = 1, 2, \cdots r$ do as follows: for each instance $I' \in \widetilde{I}$, apply Lemma 7.2.3 to break I' into sub-instances in which the sets in the pair P_i are separated, and replace I' with these sub-instances (in the set $\widetilde{I}$).

It is easy to see that after this procedure, for any sub-instance in $\widetilde{I}$, the sets in each pair P_i $(1 \leq i \leq r)$ are separated. This implies that any two distinct sets F_i and F_j are separated. Also note that I is broken into the sub-instance in $\widetilde{I}$ (as a result of repeatedly breaking each member in the above procedure). To analyze the running time of breaking I, let a_i be the number of sub-instances in $\widetilde{I}$ in the i-th iteration (i.e., when we are considering the pair P_i). Recall that the running time of the algorithm given in Lemma 7.2.3 is bounded by $O(n^{2k+1})$. Thus we have $a_{i+1} = O(n^{2k+1} \cdot a_i)$, for $i = 1, 2, \cdots r-1$, which implies that $a_i = O(n^{(i-1)(2k+1)})$. Note that $r \leq \binom{k}{2} \cdot (k-1)^2$. Thus the running time of the above procedure is $O(n^{k^5})$.

Let $I' = (G, L')$ be a sub-instance in $\widetilde{I}$. Since the sets F_i are pairwise separated in I', solving I' consists of solving the list M-partition problem for each sub-instance $(G[F_i], L')$ independently. Now as the single color in the list $L(d_i)$ should not be used for any vertex in F_i, each sub-instance $(G[F_i], L')$ is actually an instance of the list M_i-partition problem (where M_i is obtained from M by removing the row and the column corresponding to the color in the singleton list $L(d_i)$). Thus we can solve all these sub-instances in $O(f(n, k-1))$ time. In conclusion we obtain the following recursion for the running time $f(n, k)$ (keep in mind that we applied Lemma 7.2.2 in the beginning of this proof):

$$f(n, k) = O(n^{k+2} + n(n^{k^5} + 1) \cdot f(n, k-1) + n^2)$$

This implies that $f(n, k) = O(n^{k^6})$ □

Having studied the cases of 0-diagonal and 1-diagonal matrices for P_5-free graphs, we now study the separation property for these graphs. Recall that when the block C is either *-free or all * then the separation property holds for general graphs (Theorem 4.3.3). So applying Observation 4.2.2 along with the results of this section leads to the following dichotomy results:

Theorem 7.2.4. *Suppose M is a 01-diagonal matrix in which the block B is 0-free and the block C is either *-free or all *. Then the list M-partition problem, restricted to P_5-free graphs, can be solved in polynomial time (of order $O(n^{12m^8})$) if $\overline{B}$ corresponds to a bipartite co-circular arc graph, and otherwise it is NP-complete.*

Theorem 7.2.5. *Suppose M is a 01-diagonal matrix in which the block C is either *-free or all *. Then the list M-partition problem, restricted to $\{P_5, \overline{P_5}\}$-free graphs, can be solved in polynomial time (of order $O(n^{6m^8})$).*

Recall that the class of $\{P_5, \overline{P_5}\}$-free graphs, mentioned in Theorem 7.2.5, is an important subclass of P_5-free graphs, partially since it is a self-complementary class (see Section 3.3 for more details).

Note that in both theorems above, some restrictions are imposed on the block C. To justify these restrictions, we introduce many 01-diagonal matrices M with blocks A and B for which the list A-partition and the list B-partition problems are both polynomial for P_5-free graphs, while the list M-partition problem itself is NP-complete for P_5-free graphs. This serves as a strong evidence that without any restriction on the block C, the separation property does not always hold for P_5-free graphs, unless $P = NP$. To analyze the situation, we define the **-clique number* of any 0-diagonal matrix M as the maximum cardinality of a subset $S \subseteq [m]$ of colors such that the principal submatrix of M whose rows and columns are corresponding to the colors in S is 1-free. We also define the *-clique number of any 1-diagonal matrix M as the *-clique number of its complement. The following theorem offers many matrices M for which the separation property even for $\{P_5, \overline{P_5}\}$-free graphs does not hold, unless $P = NP$.

Theorem 7.2.6. *Let A' and B' be a 0-diagonal $k \times k$ and a 1-diagonal $l \times l$ matrices, respectively. Suppose both A' and B' have *-clique number ≥ 3. Then there exists a 01-diagonal matrix whose blocks A and B are identical to A' and B', respectively, and the list M-partition problem is NP-complete even when restricted to $\{P_5, \overline{P_5}\}$-free graphs.*

Proof. The proof is similar to Theorem 3.2 in [131]. The definition of *-clique number implies that there must be three colors c_0, c_1, c_2 in A' such that $A'(c_i, c_j) \neq 1$, $0 \leq i \neq j \leq 2$. Similarly, there must be three colors c'_0, c'_1, c'_2 in B' such that $B'(c'_i, c'_j) \neq 0$, $0 \leq i \neq j \leq 2$. Let H be a graph with six vertices $c_0, c_1, c_2, c'_0, c'_1, c'_2$. To facilitate the exposition, call the vertices in the first part white vertices and the vertices in the second part black vertices. The adjacencies of H will be defined later. Let $I = (G, L)$ be an instance of the list H-coloring problem with this condition: G is a bipartite graph and the lists of all the vertices in the first part of G consist only of the white vertices (of H), and the lists of all the vertices in the second part of G consists only of the black vertices (of H). Feder el al. [124] showed that solving the instances of the list H-coloring problem with the aforementioned condition is NP-complete if H is not a co-circular arc graph. So let us define the adjacency of H in such a way that it becomes a non co-circular arc graph. For example, we could take H to have the edges $c_i c'_{i-1}$ and $c_i c'_{i+1}$ for $i = 0, 1, 2$ (index calculations are all modulo 3). To prove that the graph H is not a co-circular arc graph, we need a few definitions. Suppose a graph F with clique-covering number at most two is given. The auxiliary graph F' is constructed by assigning a vertex to each edge of F such that two distinct edges of F are adjacent in F' if and only if their endpoints form a chordless cycle of order 4 in F. A result of P. Hell et al. [181] states that F is a circular arc graph if and only if F' is bipartite. To use this result, let F be the complement of H. Note that $V(F)$ can be partitioned into two cliques (namely the sets $\{c_0, c_1, c_2\}$ and $\{c'_0, c'_1, c'_2\}$), and the three edges $c_i c'_i$, for $i = 0, 1, 2$, induce a triangle in F'. Thus F' cannot be bipartite, and it implies that H is not a co-circular arc graph.

Let M be a matrix whose blocks A and B are indentical to A' and B', respectively, and whose block C is defined as follows: for $i, j = 0, 1, 2$, let $C(c_i, c'_j) = *$ if the vertices c_i and c'_j are adjacent in H, define all other entries of C to be 0. Now we show how to reduce the list H-coloring problem for the instances $I = (G, L)$ with the aforementioned condition to the list M-partition problem for $\{P_5, \overline{P_5}\}$-free graphs. Obtain the graph G' from G by adding all edges between the pairs of vertices in the second part of G. Now it is easy to see that I has a solution if and only if the instance (G', L) of the list M-partition problem has a solution. This completes our reduction. Note that graph G' is a split graph, and thus is $\{P_5, \overline{P_5}\}$-free. □

In the above theorem, note that the list A-partition and the list B-partition problems for $\{P_5, \overline{P_5}\}$-free graphs both can be solved in polynomial time (see Theorem 7.2.5). Thus for the matrices M introduced in the above theorem, the separation property is not likely to hold for P_5-free graphs (unless $P = NP$). But on the other hand, by bringing the *-chromatic number down to 2, we can prove the separation property:

Theorem 7.2.7. *Suppose M is a 01-diagonal matrix in which the blocks A and B both have *-chromatic number ≤ 2. Then the list M-partition problem, restricted to $\{P_5, \overline{P_5}\}$-free graphs, can be solved in polynomial time (of order $O(n^{2m^6})$).*

Proof. Let $I = (G, L)$ be an instance of the list M-partition problem, in which G is $\{P_5, \overline{P_5}\}$-free. Using Lemmas 4.3.2 and Lemma 7.2.1, we can break I into list-homogeneous sub-instances in $O(n^{m^6+2kl+3})$ time. Let $I' = (G, L')$ be one of these sub-instances. Let (VA, VB) be the homogeneous partition of I'. A necessary condition for I' to have a solution is that the subgraph $G[VA]$ and $G[VB]$ should be A-partitionable and B-partitionable, respectively. Given that G is $\{P_5, \overline{P_5}\}$-free, according to Lemma 7.2.1, we can find such partitions or declare their non-existance in $O(n^{m^6})$ time. Suppose these partitions exist (otherwise I' has no solution). Let $VA_1, VA_2, \cdots VA_k$ and $VB_1, VB_2, \cdots VB_l$ be the A-partition and the B-partition of $G[VA]$ and $G[VB]$, respectively. The fact that A has *-chromatic number ≤ 2 implies that in any solution of I, the vertices in each part VA_i, $i = 1, 2, \cdots k$, cannot take more than two colors. The same conclusion also holds for each part VB_j, $j = 1, 2, \cdots l$. Based on this observation, define $\widetilde{D}$ as the set of all functions $D : [m] \to 2^{[m]}$ with $|D(i)| \leq 2$, for $i \in [m]$. For each function $D \in \widetilde{J}$, define the sub-instance I'_D by applying the following changes to I':

1. For $i \in [k]$, remove from the list of any vertex in VA_{i+1} any color which is not in the set $D(i)$. (This can be done in $O(n)$ time.)

2. For $i \in [m] - [k]$, remove from the list of any vertex in VB_{i-k+1} any color which is not in the set $D(i)$ (can be done in $O(n)$ time.)

It is easy to see that I' is broken into sub-instances I'_D, for $D \in \widetilde{D}$. Note that we can construct each sub-instance I'_D in $O(n)$ time (see the procedure above). Thus all such sub-instances can be generated in $O(n)$ time (since the total number of functions in $\widetilde{D}$ is $O(1)$). Note that for any sub-instance I'_D, each list will have at most two members, and thus we can reduce the problem of solving I'_D to an instance of the 2-SAT problem which can be solved in $O(n^2)$ time. Thus I' can be solved in $O(n^2)$ time. $\square$

Finding similar complexity results for the case in which one of the blocks (A or B) has *-chromatic number ≤ 2 and the other ≥ 3 is an open problem.

7.3 Bull-free Graphs

Recall that the bull is a graph consisting of a triangle with two pendant edges (See Figure 3.1, and refer to Section 3.3 for the historic background.) The bull is a non-bipartite graph and its compliment is isomorphic to the bull again (i.e., it is self-complementary). So by using Theorems 7.1.1 and 7.1.2, we can deduce a complete dichotomy for the list M-partition problem restricted to bull-free graphs when M is 0-diagonal 1-free or 1-diagonal 0-free. The additional advantage of the class of bull-free graphs is that, it has the separation property:

Lemma 7.3.1. *Any 01-diagonal matrix M has the separation property for bull-free graphs. Moreover, there exists an algorithm giving the separation property with running time $O(n^{30klm})$, where k and l are the sizes of the blocks A and B of M, respectively.*

Proof. By applying Theorem 4.3.4, it is enough to find a polynomial-time algorithm ALG which, given any instance $I = (G, L)$ of the list M-partition problem with G being a bull-free and a split pair (V_Q, V_S) in I, breaks I into sub-instances in which the sets V_Q and V_S are separated. We claim that the set V_Q can be partitioned into two sets Q_0 and Q_1 such that the pair (V_S, Q_i), for $i = 0, 1$, induces a chain graph in G. To prove this, suppose the pair (V_Q, V_S) does not induce a chain graph, otherwise the claim is proved by defining $Q_0 = V_Q$ and $Q_1 = \emptyset$. This implies that there exist two distinct vertices $u_0, u_1 \in V_S$ and two distinct vertices $u'_0, u'_1 \in V_Q$ such that u_i is adjacent to u'_i but not to u'_{i+1}, for $i = 0, 1$ (index calculations are modulo 2). This means any vertex $u \in V_Q - \{u'_0, u'_1\}$ together with vertices u_0, u_1, u'_0, u'_1 induce a bull in G, unless u is adjacent to u_0 or u_1 (or both). This implies that set V_Q can be partitioned into sets Q_0 and Q_1 such that any vertex in Q_i is adjacent to u_i, for $i = 0, 1$. Now suppose for some $0 \leq i \leq 1$, the pair (V_S, Q_i) does not induce a chain graph in G. This implies that there exist distinct vertices $v_0, v_1 \in V_S$ and distinct vertices $v'_0, v'_1 \in Q_i$ such that v_j is adjacent to v'_j but not to v'_{j+1}, for $j = 0, 1$. Now vertices v_0, v_1, v'_0, v'_1 and u_i together induce a bull in G, which is a contradiction. Thus the partition $V_Q = Q_0 \cup Q_1$ proves the claim.

Let P_0 and P_1 be the pair (Q_0, V_S) and (Q_1, V_S), respectively. Recall that each pair P_i induces a chain graph in G. In the first step, we apply Lemma 7.2.3 for the instance I and the pair P_0 to break I into sub-instances in which the sets Q_0 and V_S are separated. In the second step, for each sub-instance I' of these sub-instances, we apply Lemma 7.2.3 for I' and the pair P_1 to break I' into sub-instances in which the sets Q_1 and V_S are separated. Let $\widetilde{I}$ be the set of all sub-instances produced in the second step. Note that in any of these sub-instances the sets V_Q and V_S are separated. Also note that I is broken into the sub-instance in $\widetilde{I}$. To analyze the running time of breaking I, note that in the first step we produce $O(n^{2m+1})$ many sub-instance, and in the second step we break each sub-instance into $O(n^{2m+1})$ many sub-instances. Thus $|\widetilde{I}| = O(n^{4m+2})$. Note that the partition $V_Q = Q_0 \cup Q_1$ can be found in $O(n^4)$ time. Thus the total running time of breaking I is $O(n^{4m+2})$. According to Theorem 4.3.4, there exists an algorithm giving the separation property with running time $O(n^{30klm})$. □

Using these results, along with Observation 4.2.2, we can deduce the following dichotomy:

Theorem 7.3.2. *Suppose M is a 01-diagonal matrix in which the block A and B are 1-free $k \times k$ and 0-free $l \times l$ matrices, respectively (see Figure 1.2). Then the list M-partition problem, restricted to bull-free graphs, can be solved in polynomial time (of order $O(n^{60klm})$) if both A and $\overline{B}$ correspond to bipartite co-circular arc graphs, and otherwise it is NP-complete.*

Bibliography

[1] American institute of mathematics. The perfect graph conjecture workshop, october 30-november 3, 2002. Available online at http://www.aimath.org/WWN/perfectgraph. Accessed: 16 June 2013.

[2] Albertson, M. O., Catlin, P. A., and Gibbons, L. Homomorphisms of 3-chromatic graphs, ii. *Congressus Numerantium 47* (1985), 19–28.

[3] Alon, N. Restricted colorings of graphs. *Surveys in combinatorics 187* (1993), 1–33.

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

[5] Andrews, P. B. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*, 2nd ed. Kluwer Academic Publishers, Berlin, 2002.

[6] Appel, K., and Haken, W. Every planar map is four colourable. *Illinois Journal of Mathematics 21*, 3 (1977), 429–567.

[7] Arbib, C., and Mosca, R. On (P_5, diamond)-free graphs. *Discrete Mathematics 250* (2002), 1–22.

[8] Bacso, G., and Tuza, Z. Dominating cliques in P_5-free graphs. *Periodica Mathematica Hungarica 21*, 4 (1990), 303–308.

[9] Bang-Jensen, J., and Hell, P. The effect of two cycles on the complexity of colouring by directed graphs. *Discrete Applied Mathematics 26* (1990), 1–23.

[10] Bang-Jensen, J., Hell, P., and MacGillivray, G. The complexity of colourings by semi-complete digraphs. *SIAM Journal on Discrete Mathematics 1* (1988), 281–298.

[11] Bang-Jensen, J., Hell, P., and MacGillivray, G. On the complexity of colouring by superdigraphs of bipartite graphs. *Discrete Mathematics 109* (1992), 27–44.

[12] Bang-Jensen, J., Hell, P., and MacGillivray, G. Hereditarily hard h-colouring problems. *Discrete Mathematics 138* (1995), 75–92.

[13] Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., and Schieber, B. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM 48*, 5 (2001), 1069–1090.

[14] Barto, L. The dichotomy for conservative constraint satisfaction problems revisited. In *Logic in Computer Science (LICS), 2011 26th Annual IEEE Symposium on* (2011), pp. 301–310.

[15] Barto, L., Kozik, M., and Niven, T. The csp dichotomy holds for digraphs with no sources and sinks (a positive answer to a conjecture of bang-jensen and hell). *SIAM Journal on Computing 38* (2008), 1782–1802.

[16] Beineke, L. W. Characterizations of derived graphs. *Journal of Combinatorial Theory 9* (1970), 129–135.

[17] Bender, E. A., Richmond, L. B., and Wormald, N. C. Almost all chordal graphs split. *J. Austral. Math. Soc. 38*, 2 (1985), 214–221.

[18] Benzaken, C., and Hammer, P. L. Linear separation of dominating sets in graphs. *Annals of Discrete Mathematics 3* (1978), 1–10.

[19] Berge, C. Les problemes de coloration en theorie des graphes. *Publ. Inst. Statist. Univ. Paris 9* (1960), 123–160.

[20] Berge, C. *Graphs and Hypergraphs.* Amsterdam: North-Holland, 1973.

[21] Bixby, R. A composition for perfect graphs. *North-Holland mathematics studies 88* (1984), 221–224.

[22] Bloom, G., and Burr, S. On unavoidable digraphs in orientations of graphs. *Journal of Graph Theory 11* (1987), 453–462.

[23] Bodirsky, M., Kára, J., and Martin, B. The complexity of surjective homomorphism problems – a survey. *CoRR abs/1104.5257* (2011).

[24] Boliac, R., and Lozin, V. V. An augmenting graph approach to the stable set problem in P_5-free graphs. *Discrete Applied Mathematics 131* (2003), 567–575.

[25] Bondy, J. A., and Murty, U. S. R. *Graph theory with applications*, vol. 290. Macmillan London, 1976.

[26] Booth, K. S., and Lueker, G. S. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of computer and system sciences 13*, 3 (1976), 335–379.

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

[28] Brandstädt, A. Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics 152*, 1 (1996), 47–54.

[29] Brandstädt, A., Dragan, F. F., Le, H. O., and Mosca, R. New graph classes of bounded clique-width. *Theory of Computing Systems 38*, 5 (2005), 623–645.

[30] Brandstädt, A., Dragan, F. F., Le, V. B., and Szymczak, T. On stable cutsets in graphs. *Discrete Applied Mathematics 105*, 1 (2000), 39–50.

[31] Brandstädt, A., and Hammer, P. L. On the stability number of claw-free P_5-free and more general graphs. *Discrete Applied Mathematics 95* (1999), 163–167.

[32] Brandstädt, A., Hammer, P. L., Le, V. B., and Lozin, V. V. Bisplit graphs. *Discrete Mathematics 299* (2005), 11–32.

[33] Brandstädt, A., and Kratsch, D. On the restriction of some np-complete graph problems to permutation graphs. In *Fundamentals of Computation Theory* (1985), pp. 53–62.

[34] Brandstädt, A., Le, V. B., and Spinrad, J. *Graph Classes: A Survey*. SIAM Monographs on Discrete Mathematics and Applications, 1999.

[35] Brandstädt, A., and Lozin, V. V. On the linear structure and clique-width of bipartite permutation graphs. *Ars Combinatoria 67*, 1 (2003), 273–281.

[36] Brandstädt, A., and Mosca, R. On the structure and stability number of p_5- and co-chair-free graphs. *Discrete Applied Mathematics 132* (2004), 47–65.

[37] Brandstädt, A., and V. Le, T. S. The complexity of some problems related to graph 3-colorability. *Discrete Applied Mathematics 89* (1998), 59–73.

[38] Broersma, H., Golovach, P. A., Paulusma, D., and Song, J. Narrowing down the gap on the complexity of coloring P_k-free graphs. In *Graph Theoretic Concepts in Computer Science* (2010), pp. 63–74.

[39] Broersma, H., Golovach, P. A., Paulusma, D., and Song, J. On coloring graphs without induced forests. In *Algorithms and Computation*. Springer, 2010, pp. 156–167.

[40] Brown, D. E. Several characterizations for unit interval bigraphs. *manuscript* (2005).

[41] Brown, D. E., Lundgren, J. R., and Flink, S. C. Characterizations of interval bigraphs and unit interval bigraphs. *Congressus Numerantium 157* (2002), 79–93.

[42] Brown, D. E., Lundgren, J. R., and Miller, C. Variations on interval graphs. *Congressus Numerantium 149* (2001), 77.

[43] Bulatov, A. A dichotomy theorem for constraints on a three-element set. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on* (2002), pp. 649–658.

[44] Bulatov, A. Tractable conservative constraint satisfaction problems. In *Logic in Computer Science, 2003. Proceedings. 18th Annual IEEE Symposium on* (2003), IEEE, pp. 321–330.

[45] Bulatov, A., and Jeavons, P. Algebraic structures in combinatorial problems. Technical Report MATH-AL-4-2001.

[46] Bulatov, A., Krokhin, A., and Jeavons, P. Constraint satisfaction problems and finite algebras. In *Automata, languages and programming*. 2000, pp. 272–282.

[47] Burlet, M., and Fonlupt, J. Polynomial algorithm to recognize a meyniel graph. *North-Holland mathematics studies 88* (1984), 225–252.

[48] Cameron, K., Eschen, E. M., Hoàng, C. T., and Sritharan, R. The complexity of the list partition problem for graphs. *SIAM Journal on Discrete Mathematics 21*, 4 (2007), 900–929.

[49] Chernyak, Z. A., and Chernyak, A. A. About recognizing (α, β) classes of polar graphs. *Discrete Mathematics 62* (1986), 133–138.

[50] Chudnovsky, M. Berge trigraphs. *Journal of Graph Theory 53* (2006), 1–55.

[51] Chudnovsky, M. The structure of bull-free graphs ithree-edge-paths with centers and anticenters. *Journal of Combinatorial Theory, Series B 102* (2012), 233–251.

[52] Chudnovsky, M., Cornuéjols, G., Liu, X., Seymour, P., and Vuškovic, K. Recognizing berge graphs. *Combinatorica 25*, 2 (2005), 143–186.

[53] Chudnovsky, M., Robertson, N., Seymour, P., and Thomas, R. The strong perfect graph theorem. *Annals of Mathematics 164* (2006), 51–229.

[54] Chudnovsky, M., and Seymour, P. The structure of claw-free graphs. *Surveys in Combinatorics* (2005), 153–171.

[55] Chvátal, V. Website on perfect graph problems.

[56] Chvátal, V. Star-cutsets and perfect graphs. *Journal of Combinatorial Theory, Series B 39*, 3 (1985), 189–199.

[57] Chvátal, V., and Hammer, P. L. Aggregation of inequalities in integer programming. *Annals of Discrete Mathematics 1* (1977), 145–162.

[58] Chvátal, V., Hoáng, C. T., Mahadev, N. V. R., and de Werra, D. Four classes of perfectly orderable graphs. *Journal of Graph Theory 11* (1987), 481–495.

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

[60] Cohen, J. E. *Food webs and niche space. Monographs in Population Biology.* Princeton University Press, 1978.

[61] Cook, K., Dantas, S., Eschen, E. M., Faria, L., De-Figueiredo, C. M. H., and Klein, S. $2K_2$ partitions into non-empty parts. *Discrete Mathematics 310* (2010), 1259–1264.

[62] Corneil, D. G., Lerchs, H., and Burlingham, L. S. Complement reducible graphs. *Discrete Applied Mathematics 3* (1981), 163–174.

[63] Corneil, D. G., Perl, Y., and Stewart, L. K. A linear recognition algorithm for cographs. *SIAM Journal on Computing 14* (1985), 926–934.

[64] Cornuéjols, G., and Cunningham, W. H. Compositions for perfect graphs. *Discrete Mathematics 55* (1985), 245–254.

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

[66] Courcelle, B. The expression of graph properties and graph transformations in monadic second-order logic. In *Handbook of Graph Grammars and Computing by Graph Transformations*, G. Rozenberg, Ed., vol. 1. 1997, pp. 313–400.

[67] Courcelle, B., Engelfriet, J., and Rozenberg, G. Handle-rewriting hypergraph grammars. *Journal of computer and system sciences 46* (1993), 218–270.

[68] Courcelle, B., Makowsky, J. A., and Rotics, U. Linear time solvable optimization problems on graphs of bounded clique width. *Theory of Computing Systems 33* (2000), 125–150.

[69] Courcelle, B., and Olariu, S. Upper bounds to the clique-width of graphs. *Discrete Applied Mathematics 101* (2000), 77–114.

[70] Cournier, A., and Habib, M. A new linear algorithm for modular decomposition. In *Trees in Algebra and Programming-CAAP'94*. 1994, pp. 68–84.

[71] Couturier, J. F., Golovach, P. A., Kratsch, D., and Paulusma, D. List coloring in the absence of a linear forest. In *Graph-Theoretic Concepts in Computer Science* (2011), pp. 119–130.

[72] Couturier, J. F., Golovach, P. A., Kratsch, D., and Paulusma, D. On the parameterized complexity of coloring graphs in the absence of a linear forest. *Journal of Discrete Algorithms* (2012), 56–62.

[73] Creignou, N., Khanna, S., and Sudan, M. *Complexity Classifications of Boolean Constraint Satisfaction Problems.* SIAM Monographs on Discrete Mathematics and Applications, 2001.

[74] Crochemore, M., Hermelin, D., Landau, G. M., and Vialette, S. Approximating the 2-interval pattern problem. In *Algorithms–ESA 2005.* 2005, pp. 426–437.

[75] Cunningham, W. H. *A Combinatorial Decomposition Theory.* Ph.D. thesis, University of Waterloo, 1973.

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

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

[78] Cygan, M., Pilipczuk, M., Pilipczuk, M., and Wojtaszczyk, J. The stubborn problem is stubborn no more: a polynomial algorithm for 3-compatible colouring and the stubborn list partition problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms* (2011), pp. 1666–1674.

[79] Dabrowski, K., Lozin, V. V., Raman, R., and Ries, B. Colouring vertices of triangle-free graphs. In *Graph Theoretic Concepts in Computer Science* (2010), pp. 184–195.

[80] Dalmau, V. Constraint satisfaction problems in non-deterministic logarithmic space. In *Automata, Languages and Programming.* 2002, pp. 414–425.

[81] Damaschke, P. Induced subgraphs and well-quasi-ordering. *Journal of Graph Theory 14*, 4 (1990), 427–435.

[82] Dantas, S., De-Figueiredo, C. M. H., Gravier, S., and Klein, S. Finding *h*-partitions efficiently. *Informatique théorique et applications 39*, 1 (2005), 133–144.

[83] Das, S., Sen, M., Roy, A. B., and West, D. B. Interval digraphs: An analogue of interval graphs. *Journal of Graph Theory 13*, 2 (1989), 189–202.

[84] David, S. J. The np-completeness column: an ongoing guide. *Journal of Algorithms 6* (1985), 434–451.

[85] De-Figueiredo, C. M. H., Klein, S., Kohayakawa, Y., and Reed, B. A. Finding skew partitions efficiently. *Journal of Algorithms 37*, 2 (2000), 505–521.

[86] De-Figueiredo, C. M. H., and Maffray, F. Optimizing bull-free perfect graphs. *SIAM Journal on Discrete Mathematics 18* (2004), 226–240.

[87] De-Figueiredo, C. M. H., Maffray, F., and Porto, O. On the structure of bull-free perfect graphs. *Graphs and Combinatorics 13* (1997), 31–55.

[88] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (chordal graphs). Available online at http://www.graphclasses.org/classes/gc_32.html. Accessed: 28 October 2012.

[89] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (circle graphs). Available online at http://www.graphclasses.org/classes/gc_132.html. Accessed: 28 October 2012.

[90] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (circular arc graphs). Available online at http://www.graphclasses.org/classes/gc_133.html. Accessed: 28 October 2012.

[91] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (cographs). Available online at http://www.graphclasses.org/classes/gc_151.html. Accessed: 28 October 2012.

[92] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (comparability graphs). Available online at http://www.graphclasses.org/classes/gc_72.html. Accessed: 28 October 2012.

[93] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (containment graphs of circular arcs). Available online at http://www.graphclasses.org/classes/gc_136.html. Accessed: 28 October 2012.

[94] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (interval containment bigraphs). Available online at http://www.graphclasses.org/classes/gc_891.html. Accessed: 28 October 2012.

[95] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (interval graphs). Available online at http://www.graphclasses.org/classes/gc_234.html. Accessed: 28 October 2012.

[96] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (line graphs). Available online at http://www.graphclasses.org/classes/gc_249.html. Accessed: 28 October 2012.

[97] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (main page). Available online at http://www.graphclasses.org/classes. Accessed: 28 October 2012.

[98] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (perfectly orderable graphs). Available online at http://www.graphclasses.org/classes/gc_4.html. Accessed: 28 October 2012.

[99] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (permutation graphs). Available online at http://www.graphclasses.org/classes/gc_288.html. Accessed: 28 October 2012.

[100] de Ridder, H. N., e. a. Information system on graph classes and their inclusions (proper interval bigraphs). Available online at http://www.graphclasses.org/classes/gc_882.html. Accessed: 28 October 2012.

[101] de Souza-Francisco, R., Klein, S., and Nogueira, L. T. Characterizing (k,l)-partitionable cographs. *Electronic Notes in Discrete Mathematics 22* (2005), 277–280.

[102] Demange, M., Ekim, T., and de Werra, D. Partitioning cographs into cliques and stable sets. *Discrete Optimization 2* (2005), 145–153.

[103] Deng, X., Hell, P., and Huang, J. Linear time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM Journal on Computing 25* (1996), 390–403.

[104] Dirac, G. A. On rigid circuit graphs. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* (1961), vol. 25, pp. 71–76.

[105] Duchet, P. Classical perfect graphs. *Annals of Discrete Mathematics 21* (1984), 67–96.

[106] Dushnik, B., and Miller, E. W. Partially ordered sets. *American Journal of Mathematics 63*, 3 (1941), 600–610.

[107] Ebbinghaus, H., and Flum, J. *Finite Model Theory.* Perspectives in Mathematical Logic. Springer- Verlag, New York, 1995.

[108] Edmonds, J. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B 69* (1965), 125–130.

[109] Edmonds, J. Paths, trees and flowers. *Canad. J. Math 17* (1965), 449–467.

[110] Ekim, T., Hell, P., Stacho, J., and de Werra, D. Polarity of chordal graphs. *Discrete Applied Mathematics 156* (2008), 2469–2479.

[111] Ekim, T., Mahadev, N. V. R., and de=Werra, D. Polar cographs. *Discrete Applied Mathematics 156*, 10 (2008), 1652–1660.

[112] Enright, J., Stewart, L., and Tardos, G. On list colouring and list homomorphism of permutation and interval graphs. *CoRR abs/1206.5106* (2012).

[113] Erdös, P., Rubin, A. L., and Taylor, H. Choosability in graphs. *Congressus Numerantium 26* (1979), 125–157.

[114] Even, S., Pnueli, A., and Lempel, A. Permutation graphs and transitive graphs. *Journal of the ACM 19*, 3 (1972), 400–410.

[115] Everett, H., Klein, S., and Reed, B. An algorithm for finding homogeneous pairs. *Discrete Applied Mathematics 72* (1997), 209–218.

[116] Everett, H., Klein, S., and Reed, B. An optimal algorithm for finding clique-cross partitions. In *Proceedings of the Twenty-ninth Southeastern International Conference on Combinatorics, Graph Theory and Computing* (1998), vol. 135, pp. 171–177.

[117] Faudree, R., Flandrin, E., and Ryjáček, Z. Claw-flee graphs - a survey. *Discrete Mathematics 164* (1997), 87–147.

[118] Feder, T., and Hell, P. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory, Series B 72*, 2 (1998), 236–250.

[119] Feder, T., and Hell, P. Matrix partitions of perfect graphs. *Claude Berge Memorial Volume* (2004).

[120] Feder, T., and Hell, P. Full constraint satisfaction problems. *SIAM Journal on Computing 36*, 1 (2006), 230–246.

[121] Feder, T., and Hell, P. Matrix partitions of perfect graphs. *Discrete Mathematics 306*, 19 (2006), 2450–2460.

[122] Feder, T., and Hell, P. On realizations of point determining graphs and obstructions to full homomorphisms. *Discrete Mathematics 308* (2008), 1639–1652.

[123] Feder, T., Hell, P., and Hochstättler, W. Generalized colourings (matrix partitions) of cographs. In *Graph Theory in Paris*. 2007, pp. 149–167.

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

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

[126] Feder, T., Hell, P., and Huang, J. List homomorphisms of graphs with bounded degrees. *Discrete Mathematics 307* (2007), 386–392.

[127] Feder, T., Hell, P., and Huang, J. Extension problems with degree bounds. *Discrete Applied Mathematics 157* (2009), 1592–1599.

[128] Feder, T., Hell, P., Jonsson, P., Krokhin, A., and Nordh, G. Retractions to pseudo-forests. *SIAM Journal on Discrete Mathematics 24* (2010), 101–112.

[129] Feder, T., Hell, P., Klein, S., and Motwani, R. List partitions. *SIAM Journal on Discrete Mathematics 16*, 3 (2003), 449–478.

[130] Feder, T., Hell, P., Klein, S., Nogueira, L. T., and Protti, F. List matrix partitions of chordal graphs. *Lecture Notes in Computer Science 2976* (2004), 100–108.

[131] Feder, T., Hell, P., Klein, S., Nogueira, L. T., and Protti, F. List matrix partitions of chordal graphs. *Theoretical Computer Science 349*, 1 (2005), 52–66.

[132] Feder, T., Hell, P., Král, D., and Sgall, J. Two algorithms for general list matrix partitions. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms* (2005), Society for Industrial and Applied Mathematics, pp. 870–876.

[133] Feder, T., Hell, P., and Nekooei-Rizi, S. Partitioning chordal graphs. *Electronic Notes in Discrete Mathematics 38* (2011), 325–330.

[134] Feder, T., Hell, P., and Shklarsky, O. Partitions of generalized split graphs. *manuscript* (2012).

[135] Feder, T., Hell, P., Stacho, J., and Schell, G. Dichotomy for tree-structured trigraph list homomorphism problems. *Discrete Applied Mathematics 159* (2011), 1217–1224.

[136] Feder, T., Hell, P., and Tucker-Nally, K. Digraph matrix partitions and trigraph homomorphisms. *Discrete Appl. Mathematics 154* (2006), 2458–2469.

[137] Feder, T., Hell, P., and Xie, W. Matrix partitions with finitely many obstructions. *Electronic Notes in Discrete Mathematics 28* (2007).

[138] Feder, T., and Vardi, M. Y. The computational structure of monotone monadic snp and constraint satisfaction. *SIAM Journal on Computing 28* (1999), 57–104.

[139] Fellner, W. D. On minimal graphs. *Theoretical Computer Science 17* (1982), 103–110.

[140] Fishburn, P. C. *Interval orders and interval graphs: A study of partially ordered sets.* Wiley-Interscience Series in Discrete Mathematics, New York, 1985.

[141] Fleischner, H., Mujuni, E., Paulusma, D., and Szeider, S. Covering graphs with few complete bipartite subgraphs. *Theoretical Computer Science 410* (2009), 2045–2053.

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

[143] Foldes, S., and Hammer, P. Split graphs. *Congressus Numerantium 19* (1977), 311–315.

[144] Fouquet, J. L., and Giakoumakis, V. On semi-P_4-sparse graphs. *Discrete Mathematics 165* (1997), 277–300.

[145] Fouquet, J. L., Giakoumakis, V., Maire, F., and Thuillier, H. On graphs without P_5 and $\overline{P_5}$. *Discrete Mathematics 146* (1995), 33–44.

[146] Fulkerson, D. R., and Gross, O. A. Incidence matrices and interval graphs. *Pacific Journal of Mathematics 15* (1965), 835–855.

[147] Gallai, T. Maximum-minimum sätze über graphen. *Acta Mathematica Hungarica 9*, 3 (1958), 395–434.

[148] Galluccio, A., Hell, P., and Nešetřil, J. The complexity of h-colouring of bounded degree graphs. *Discrete Mathematics 222* (2000), 101–109.

[149] Gambette, P., and Vialette, S. On restrictions of balanced 2-interval graphs. In *Graph-Theoretic Concepts in Computer Science* (2007), pp. 55–65.

[150] Garey, M. R., and Johnson, D. S. *Computers and Intractability.* Freeman, San Francisco, 1979.

[151] Garey, M. R., Johnson, D. S., Miller, G. L., and Papadimitriou, C. H. The complexity of coloring circular arc and chords. *SIAM Journal on Algebraic Discrete Methods 1*, 2 (1980).

[152] Gavril, F. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing 1*, 2 (1972).

[153] Gavril, F. Algorithms on circular-arc graphs. *Networks 4*, 4 (1974), 357–369.

[154] Gerber, M. U., Hertz, A., and Schindl, D. P_5-free augmenting graphs and the maximum stable set problem. *Discrete Applied Mathematics 132* (2004), 109–119.

[155] Gerber, M. U., and Kobler, D. Algorithms for vertex-partitioning problems on graphs with fixed clique-width. *Theoretical Computer Science 299* (2003), 719–734.

[156] Gerber, M. U., and Lozin, V. V. On the stable set problem in special P_5-free graphs. *Discrete Applied Mathematics 125* (2003), 215–224.

[157] Giakoumakis, V., and Rusu, I. Weighted parameters in $(P_5, \overline{P_5})$-free graphs. *Discrete Applied Mathematics 80*, 2-3 (1997), 255–261.

[158] Gioan, E., and Paul, C. Split decomposition and graph-labelled trees: characterizations and fully-dynamic algorithms for totally decomposable graphs. *coRR abs/0810.1823* (2008).

[159] Golovach, P. A., and Heggernes, P. Choosability of P_5-free graphs. In *Mathematical Foundations of Computer Science 2009.* 2009, pp. 382–391.

[160] Golovach, P. A., Paulusma, D., and Song, J. Coloring graphs without short cycles and long induced paths. In *Fundamentals of Computation Theory* (2011), pp. 193–204.

[161] Golovach, P. A., Paulusma, D., and Song, J. Computing vertex-surjective homomorphisms to partially reflexive trees. In *Computer Science–Theory and Applications*. 2011, pp. 261–274.

[162] Golovach, P. A., Paulusma, D., and Song, J. 4-coloring H-free graphs when H is small. *Discrete Applied Mathematics* (2012).

[163] Golumbic, M. C. The complexity of comparability graph recognition and coloring. *Computing 18* (1977), 199–208.

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

[165] Golumbic, M. C., and Rotics, U. On the cliquewidth of perfect graph classes. In *Graph-Theoretic Concepts in Computer Science* (1999), pp. 135–147.

[166] Golumbic, M. C., and Rotics, U. On the clique-width of some perfect graph classes. *International Journal of Foundations of Computer Science 11*, 3 (2000), 423–443.

[167] Golumbic, M. C., and Shamir, R. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the ACM 40*, 5 (1993), 1108–1133.

[168] Grötschel, M., Lovász, L., and Schrijver, A. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica 1* (1981), 169–197.

[169] Gupta, U. I., Lee, D. T., and Leung, J. Y. T. Efficient algorithms for interval graphs and circular-arc graphs. *Networks 12*, 4 (2006), 459–467.

[170] Habib, M., McConnell, R., Paul, C., and Viennot, L. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition, and consecutive ones testing. *Theoretical Computer Science 234* (2000), 59–84.

[171] Habib, M., and Paul, C. A simple linear time algorithm for cograph recognition. *Discrete Applied Mathematics 145* (2005), 183–197.

[172] Hadwiger, H., Debrunner, H., and Klee, V. *Combinatorial Geometry in the Plane*. Holt Rinehardt and Winston, New York, 1964.

[173] Häggkvist, R., Hell, P., Miller, D. J., and Lara, V. N. On multiplicative graphs and the product conjecture. *Combinatorica 8*, 1 (1988), 63–74.

[174] Harary, F., Kabell, J. A., and McMorris, F. R. Bipartite intersection graphs, comment. *Math Universitatis Carolinae 23* (1982), 739–745.

[175] Harary, F., and Norman, R. Z. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo 9*, 2 (1960), 161–169.

[176] Hayward, R., and Reed, B. A. Forbidding holes and antiholes. *Perfect Graphs. John Wiley & Sons* (2001), 113–137.

[177] Hedrlín, Z., and Pultr, A. Symmetric relations (undirected graphs) with given semigroups. *Monatshefte für Mathematik 69*, 4 (1965), 318–322.

[178] Hell, P. Absolute planar retracts and the four color conjecture. *Journal of Combinatorial Theory, Series B 17*, 1 (1974), 5–10.

[179] Hell, P. Algorithmic aspects of graph homomorphisms. In *Surveys in Combinatorics* (2003), pp. 239–276.

[180] Hell, P. Graph partitions with prescribed patterns. *manuscript* (2012).

[181] Hell, P., and Huang, J. Two remarks on circular arc graphs. *Graphs and Combinatorics 13* (1997), 65–72.

[182] Hell, P., and Huang, J. Certifying lexbfs recognition algorithms for proper interval graphs and proper interval bigraphs. *SIAM Journal on Discrete Mathematics 18*, 3 (2004), 554–570.

[183] Hell, P., and Huang, J. Interval bigraphs and circular arc graphs. *Journal of Graph Theory 46* (2004), 313–327.

[184] Hell, P., Klein, S., Nogueira, L. T., and Protti, F. Partitioning chordal graphs into independent sets and cliques. *Discrete Applied Mathematics 141*, 1 (2004), 185–194.

[185] Hell, P., Klein, S., Protti, F., and Tito, L. On generalized split graphs. *Electronic Notes in Discrete Mathematics 7* (2001), 98–101.

[186] Hell, P., and Neš̃etr̃il, J. Cohomomorphisms of graphs and hypergraphs. *Mathematische Nachrichten 87*, 1 (1979), 53–61.

[187] Hell, P., and Neš̃etr̃il, J. On the complexity of h-coloring. *Journal of Combinatorial Theory, Series B 48*, 1 (1990), 92–110.

[188] Hell, P., and Neš̃etr̃il, J. Counting list homomorphisms for graphs with bounded. In *Graphs, Morphisms, and Statistical Physics: Dimacs Workshop Graphs, Morphisms and Statistical Physics, March 19-21, 2001, Dimacs Center* (2004), vol. 63, pp. 105–112.

[189] Hell, P., and Neš̃etr̃il, J. *Graphs and Homomorphisms.* Oxford University Press, 2004.

[190] Hell, P., Neš̃etr̃il, J., and Zhu, X. Duality and polynomial testing of tree homomorphisms. *Transactions of the American Mathematical Society 348*, 4 (1996), 1281–1297.

[191] Hell, P., and Rafiey, A. The dichotomy of list homomorphisms for digraphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms* (2011), pp. 1703–1713.

[192] Hell, P., and Rival, I. Absolute retracts and varieties of reflexive graphs. *Canad. J. Math. 39* (1987), 544–567.

[193] Hemminger, R. L., and Beineke, L. W. Line graphs and line digraphs. *Selected Topics in Graph Theory, Academic Press, New York* (1978), 271–305.

[194] Hoáng, C. T. *Ph.D. thesis.* Montreal, 1985.

[195] Hoàng, C. T. Efficient algorithms for minimum weighted colouring of some classes of perfect graphs. *Discrete Applied Mathematics 55* (1994), 133–143.

[196] Hoàng, C. T., Kaminski, M., Lozin, V. V., Sawada, J., and Shu, X. Deciding k-colorability of P_5-free graphs in polynomial time. *Algorithmica 57*, 1 (2010), 74–81.

[197] Holyer, I. The np-completeness of edge-coloring. *SIAM Journal on Computing 10*, 4 (1981), 718–720.

[198] Hsu, W. L. The coloring and maximum independent set problems on planar perfect graphs. *Journal of the ACM 35* (1988), 535–563.

[199] Huang, S. Improved complexity results on k-coloring P_t-free graphs. *CoRR abs/1304.5808* (2013).

[200] Irving, R. W. Np-completeness of a family of graph-colouring problems. *Discrete Applied Mathematics 5* (1983), 111–117.

[201] Ito, T., Kaminski, M., Paulusma, D., and Thilikos, D. M. On disconnected cuts and separators. *Discrete Applied Mathematics 159* (2011), 1345–1351.

[202] Jeavons, P. On the algebraic structure of combinatorial problems. *Theoretical Computer Science 200* (1998), 185–204.

[203] Jeavons, P., Cohen, D., and Gyssens, M. Closure properties of constraints. *Journal of the ACM 44* (1997), 527–548.

[204] Jensen, T. R., and Toft, B. *Graph Coloring Problems.* Wiley Interscience, Hoboken, 1995.

[205] Jung, H. A. On a class of posets and the corresponding comparability graphs. *Journal of Combinatorial Theory, Series B 24*, 2 (1978), 125–133.

[206] Kaminski, M., and Lozin, V. V. Coloring edges and vertices of graphs without short or long cycles. *Contributions to Discrete Mathematics 2* (2007), 61–66.

[207] Kaminski, M., and Lozin, V. V. Vertex 3-colorability of claw-free graphs. *Algorithmic Operations Research 21* (2007).

[208] Karp, R. M. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. New York: Plenum, 1972, pp. 85–103.

[209] Kennedy, W. S., and Reed, B. Fast skew partition recognition. In *Computational Geometry and Graph Theory*. 2008, pp. 101–107.

[210] King, A. *Claw-free graphs and two conjectures on omega, Delta, and chi.* Ph.D. thesis, McGill University, 2009.

[211] Klee, V. What are the intersection graphs of arcs in a circle? *American Mathematical Monthly 76* (1976), 810–813.

[212] Klein, S., and De-Figueiredo, C. M. H. The np-completeness of multi-partite cutset testing. *Congressus Numerantium 119* (1996), 217–222.

[213] Kolaitis, P., and Vardi, M. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences 61*, 2 (2000), 302–332.

[214] Komárek, P. Some new good characterizations for directed graphs. *Časopis pro pěstování matematiky 109*, 4 (1984), 348–354.

[215] Král, D., Kratochvíl, J., Tuza, Z., and Woeginger, G. J. Complexity of coloring graphs without forbidden induced subgraphs. *Lecture Notes in Computer Science 2204* (2001), 2–54.

[216] Kratochvíl, J. Precoloring extension with fixed color bound. *Acta Math. Univ. Comen 62* (1993), 139–153.

[217] Krausz, J. Dmonstration nouvelle d'un thorme de whitney sur les rseaux. *Mat. Fiz. Lapok 50* (1943), 75–85.

[218] Kun, G., and Ne̋setr̃il, J. Np for combinatorialists. *Electronic Notes in Discrete Mathematics 29* (2007), 373–381.

[219] Ladner, R. E. On the structure of polynomial time reducibility. *Journal of the ACM 22* (1975), 155–171.

[220] Le, V. B., Randerath, B., and Schiermeyer, I. On the complexity of 4-coloring graphs without long induced paths. *Theoretical Computer Science 389* (2007), 330–335.

[221] Lerchs, H. On cliques and kernels. Tech. Report, Dept. of Comp. Sci., Univ. of Toronto, 1971.

[222] Lévêque, B., and Maffray, F. Coloring bull-free perfectly contractile graphs. *SIAM Journal on Discrete Mathematics 21* (2007), 999–1018.

[223] Levin, L. A. Universal sequential search problems. *Problemy Peredachi Informatsii 9*, 3 (1973), 115–116.

[224] Lin, I. J., Sen, M. K., and West, D. B. Classes of interval digraphs and 01-matrices. *Congressus Numerantium 125* (1997), 201–209.

[225] Lin, M. C., and Szwarcfiter, J. L. Characterizations and recognition of circular-arc graphs and subclasses: A survey. *Discrete Mathematics 309* (2009), 5618–5635.

[226] Lovász, L. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics 2*, 3 (1972), 253–267.

[227] Lovász, L. Communication complexity: a survey. *Paths, flows, and VLSI-layout 9* (1990), 235–265.

[228] Lozin, V. V. Stability in P_5- and banner-free graphs. *European Journal of Operational Research 125* (2000), 292–297.

[229] MacGillivray, G., and Yu, M. L. Generalized partitions of graphs. *Discrete Applied Mathematics 91* (1999), 143–153.

[230] Maffray, F. On the coloration of perfect graphs. In *Recent Advances in Algorithms and Combinatorics*. 2003, pp. 65–84.

[231] Maffray, F., and Preissmann, M. On the np-completeness of the k-colorability problem for triangle-free graphs. *Discrete Mathematics 162* (1996), 313–317.

[232] Martin, B., and Paulusma, D. The computational complexity of disconnected cut and $2K_2$-partition. In *Principles and Practice of Constraint Programming–CP 2011*. 2011, pp. 561–575.

[233] Maurer, H. A., Salomaa, A., and Wood, D. Colorings and interpretations: A connection between graphs and grammar forms. *Discrete Applied Mathematics 3* (1981), 119–135.

[234] Maurer, H. A., Sudborough, J. H., and Welzl, E. On the complexity of the general coloring problem. *Information and control 51*, 2 (1981), 128–145.

[235] McConnell, R. M. Linear-time recognition of circular-arc graphs. *Algorithmica 37*, 2 (2003), 93–147.

[236] McConnell, R. M., and Spinrad, J. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms* (1994), Society for Industrial and Applied Mathematics, pp. 536–545.

[237] McConnell, R. M., and Spinrad, J. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms* (1994), pp. 536–545.

[238] McConnell, R. M., and Spinrad, J. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms* (1994), pp. 536–545.

[239] McConnell, R. M., and Spinrad, J. Linear-time transitive orientation. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms* (1997), pp. 19–25.

[240] Montanari, U. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences 7* (1974), 95–132.

[241] Müller, H. Recognizing interval digraphs and interval bigraphs in polynomial time. *Discrete Applied Mathematics 78* (1997), 189–205.

[242] Nekooei-Rizi, S. *Matrix partitions of chordal graphs.* M.Sc. thesis, Simon Fraser University, 2010.

[243] Nešetřil, J. Representations of graphs by means of products and their complexity. In *Mathematical Foundations of Computer Science 1981.* 1981, pp. 94–102.

[244] Nes̃etr̃il, J., and Pultr, A. On classes of relations and graphs determined by subobjects and factorobjects. *Discrete Mathematics 22* (1978), 287–300.

[245] Nirkhe, M. V. *Efficient algorithms for circular-arc containment graphs.* M.Sc. thesis, University of Maryland College Park, 1987.

[246] Papadimitriou, C. *Computational Complexity.* Addison-Wesley, 1994.

[247] Pnueli, A., Lempel, A., and Even, S. Transitive orientation of graphs and identification of permutation graphs. *Canad. J. Math 23* (1971), 160–175.

[248] Puech, J. Irredundance perfect and P_6-free graphs. *Journal of Graph Theory 29* (1998), 239–255.

[249] Randerath, B. 3-colourability and forbidden subgraphs. *Electronic Notes in Discrete Mathematics 5* (2000), 270–273.

[250] Randerath, B. 3-colorability and forbidden subgraphs. i: Characterizing pairs. *Discrete Mathematics 276* (2004), 313–325.

[251] Randerath, B., and Schiermeyer, I. 3-colorability $\in P$ for P_5-free graphs. *Discrete Applied Mathematics 136* (2004), 299–313.

[252] Randerath, B., and Schiermeyer, I. Vertex coloring and forbidden subgraphs - a survey. *Graphs and Combinatorics 20*, 1 (2004), 1–40.

[253] Rautenberg, W. *A Concise Introduction to Mathematical Logic*, 3rd ed. Springer, New York, 2010.

[254] Reed, B., and Sbihi, N. Recognizing bull-free perfect graphs. *Graphs and Combinatorics 11* (1995), 171–178.

[255] Robertson, N., and Seymour, P. Graph minors v, excluding a planar graph. *Journal of Combinatorial Theory, Series B 41* (1986), 92–114.

[256] Rose, D., Lueker, G., and Tarjan, R. E. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing 5*, 2 (1976), 266–283.

[257] Rotem, D., and Urrutia, J. Circular permutation graphs. *Networks 12* (1982), 429–437.

[258] Roussopoulos, N. D. A max m,n algorithm for determining the graph *h* from its line graph *g*. *Information Processing Letters 2*, 4 (1973), 108–112.

[259] Sanyal, B. K., and Sen, M. K. New characterizations of digraphs represented by intervals. *Journal of Graph Theory 22* (1996), 297–303.

[260] Schaefer, T. J. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing* (1978), vol. 14, pp. 216–226.

[261] Seinsche, S. On a property of the class of n-colourable graphs. *Journal of Combinatorial Theory, Series B 16* (1974), 191–193.

[262] Shannon, C. The zero error capacity of a noisy channel. *Information Theory, IRE Transactions on 2*, 3 (1956), 8–19.

[263] Sritharan, R. A linear time algorithm to recognize circular permutation graphs. *Networks 27* (1996), 171–174.

[264] Stacho, J. *Complexity of Generalized Colorings of Chordal Graphs.* Ph.D. thesis, Simon Fraser University, 2008.

[265] Sumner, D. P. Dacey graphs. *J. Austral. Math. Soc. 18*, 4 (1974), 492–502.

[266] Tarjan, R. E. Coloring graphs with stable cutsets. *Journal of Combinatorial Theory, Series B 34*, 3 (1983), 258–267.

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

[268] Tucker, A. An efficient test for circular-arc graphs. *SIAM Journal on Computing 9*, 1 (1980), 1–24.

[269] Tucker-Nally, K. *List M-partitions of digraphs.* M.Sc. thesis, Simon Fraser University, 2003.

[270] Tuza, Z. Graph colorings with local constraints - a survey. *Discuss. Math. Graph Theory 17* (1997), 161–228.

[271] Urrutia, J. Partial orders and euclidean geometry. In *Algorithms and Order.* 1988, pp. 387–434.

[272] van Rooij, A. M., and Wilf, H. S. The interchange graph of a finite graph. *Acta Mathematica Hungarica 16*, 3 (1965), 263–269.

[273] Vikas, N. Computational complexity of compaction to reflexive cycles. *SIAM Journal on Computing 32* (2003), 253–280.

[274] Vikas, N. Algorithms for partition of some class of graphs under compaction. In *Computing and Combinatorics.* 2011, pp. 319–330.

[275] Vizing, V. G. Coloring the vertices of a graph in prescribed colors. *Diskret. Anal 29* (1976), 3–10.

[276] West, D. B. Short proofs for interval digraphs. *Discrete Mathematics 178* (1998), 287–292.

[277] Whitesides, S. An algorithm for finding clique cut-sets. *INFO. PROC. LETT. 12*, 1 (1981), 31–32.

[278] Whitesides, S. A method for solving certain graph recognition and optimization problems, with applications to perfect graphs. *North-Holland mathematics studies 88* (1984), 281–297.

[279] Whitney, H. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics 54* (1932), 150–168.

[280] Woeginger, G. J., Sgall, J., et al. The complexity of coloring graphs without long induced paths. *Acta Cybernetica 15*, 1 (2001), 107–117.

[281] Yannakakis, M. The complexity of the partial order dimension problem. *SIAM Journal on Algebraic and Discrete Methods 3* (1982), 351–358.

[282] Zhang, P., Schon, E. A., Fischer, S. G., Cayanis, E., Weiss, J., Kistler, S., and Bourne, P. E. An algorithm based on graph theory for the assembly of contigs in physical mapping of dna. *Computer applications in the biosciences: CABIOS 10*, 3 (1994), 309–317.