Partitioning cographs into forests and independent sets

by

Anurag Sanyal

B.Tech., Indian Institute of Technology Jodhpur, 2018

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

Approval

Name:	**Anurag Sanyal**
Degree:	**Master of Science (Computing Science)**
Title:	**Partitioning cographs into forests and independent sets**
Examining Committee:	**Chair:** Joseph Peters Professor **Pavol Hell** Senior Supervisor Professor **Binay Bhattacharya** Supervisor Professor **César Hernández Cruz** External Examiner Associate Professor Department of Mathematics, Faculty of Science National Autonomous University of México
Date Defended:	**August 4, 2020**

Abstract

To determine the chromatic number of a graph, we seek to partition the vertices into minimum number of independent sets. Similarly, for arboricity, we seek to partition the vertices into minimum number of sets, each of which induces a forest. Both problems seek to partition the vertices into sets that induce a sparse subgraph, and both are NP-hard in general and can be solved in polynomial time on cographs. In this thesis, we consider a mixed problem, where a graph is partitioned into p forests and q independent sets. It is known that for each p and q, the partition problem has a finite complete set of minimal cograph obstructions. For the cases where $p = 0$ or $p = 1$, a minimal obstruction characterization of (p, q)-partitionability of cographs was previously known. However, it was also known that the number of minimal obstructions grows exponentially with p. We consider the next case of $p = 2$ and $q = 1$, and provide a complete list of minimal cograph obstructions. We also provide polynomial time certifying algorithms for the cases $p = 1$ for any q, and $p = 2$ and $q = 1$.

We also consider a vertex deletion version of the partition problem. Here, r vertices are allowed to be deleted so that the remaining graph admits a partition into p forests and q independent sets. For this problem, we provide a complete list of minimal cograph obstructions when $p = q = r = 1$, and $p = r = 1$, $q = 2$.

Keywords: Vertex arboricity; chromatic number; partition problems; cographs

Dedication

Dedicated to my parents and my little sister Preeti.

Acknowledgements

I would like to thank Dr. Pavol Hell and Dr. César Hernández Cruz for their patience, encouragement, support and guidance throughout my Masters.

I am equally thankful to my friends Archit, Dhruv, Daval, Mansi for their support and friendship.

Table of Contents

List of Figures

Chapter 1

Background and Literature

1.1 Definitions

A *graph* $G = (V, E)$ consists of a set of vertices V and a set of edges E, where $E \subseteq V \times V$. The vertex set of a graph G is denoted by $V(G)$ and its edge set by $E(G)$. If $e = uv$ is an edge, then u and v are called the endpoints of the edge e. If u and v are endpoints of an edge, they are called *adjacent* vertices and u is *neighbour* of v, and v is a *neighbour* of u. The *degree* of a vertex v is the number of vertices adjacent to v.
A *loop* is an edge whose endpoints are equal. A *simple graph* is a graph having no loops. A directed graph is a graph $D = (V, A)$ with vertex set $V(D)$ and arcs $A(D)$, where each arc is a set of ordered pairs of vertices. Following standard usage, we write vw for the ordered pair (v, w). The arc vw is directed from v to w.

Definition 1. *A* subgraph *of a graph G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. The notation $H \subseteq G$ denotes the subgraph relation between two graphs, and we say G contains H or H is a subgraph of G.*

Definition 2. *The subgraph H of a graph G is said to be* induced *if for any two vertices u and $v \in V(H)$, u and v are adjacent in H if and only if u and v are adjacent in G. For any set $S \subset V(G)$, we denote by $G[S]$ the subgraph induced by S in G.*

For a graph G and a vertex $v \in V(G)$, we denote by $G - v$ the graph obtained by removing the vertex v from G.

Definition 3. *The* complement $\overline{G}$ *of a graph G is the graph with the same vertex set $V(G)$ and an edge $e = uv \in E(\overline{G})$ if and only if $e = uv \notin E(G)$.*

Definition 4. *A* complete *graph is a graph with all possible edges, i.e., all pairs of distinct vertices are adjacent.*

A *clique*, C, in a graph $G = (V, E)$ is a subset of the vertices, $C \subseteq V(G)$, such that the subgraph induced by C in G is a complete graph. The clique number of G, denoted by $\omega(G)$, is the size of the largest complete subgraph of G.

Definition 5. *An* independent set *in G is a set of vertices such that no two of which are adjacent.*

We denote by $\alpha(G)$, the size of the largest independent set in G.

The clique cover number of G, denoted by $k(G)$, is the smallest number of complete subgraphs needed to cover the vertices of G.

Definition 6. *A bipartite graph is a graph such that the vertex set $V(G)$ can be partitioned into two independent sets X and Y. The sets X and Y are called* parts *of the bipartition.*

A complete bipartite graph $V(G) = (X, Y)$ is a bipartite graph with parts X and Y where edge $e = uv \in E(G)$ if and only if u and v lie in different parts.

A path in a graph G is a sequence of distinct vertices $P = \{v_1, v_2, ..., v_n\}$ such that $v_i v_{i+1} \in E(G)$ for all $i \in \{1, 2, ..., n-1\}$. The integer n is the *length* of the path. A walk in a graph G is defined similarly except the vertices need not be distinct.

A cycle in a graph G is a sequence of vertices $C = \{v_1, v_2, ..., v_n\}$ such that $v_1 = v_n$ and $v_i v_{i+1} \in E(G)$ for all $i \in \{1, 2, ..., n-1\}$, where n is the length of the cycle.

Definition 7. *The* girth *of a graph is defined as the length of the shortest cycle in G.*

A graph G is said to be *connected* if for any two vertices u and v in the vertex set, there exists a path P in G with u and v as the endpoints of P. A graph that is not connected is said to be *disconnected.*

Definition 8. *Colouring a graph G is an assignment of labels to the vertices of a graph so that for each edge $e = uv \in E(G)$, the vertices u and v have different labels.*

The label received by a vertex is the *colour* of that vertex. A colouring that uses at most k colours is called a k-colouring. Alternatively, one can also visualize a k-colouring as a partition of the vertex set of G into k independent sets. For any graph on n vertices, it is possible to assign a colouring with n labels where each vertex gets a different label. Thus, every graph on n vertices is n-colourable.

The *chromatic number* of G, denoted by $\chi(G)$, is the smallest number of colours k so that G has a k-colouring, or equivalently, the smallest number of independent sets needed to cover the vertices of G.

We introduce the following notations for undirected graphs:
K_n is the complete graph on n vertices.
$K_{r,s}$ denotes the complete bipartite graph where $|X| = r$ and $|Y| = s$.
P_n is a simple graph on n vertices with vertex set $V = \{0, 1, 2, .., n-1\}$ such that $v_i v_j \in E$ if and only if either $i = j + 1$ or $i = j - 1$.
C_n is a simple graph on n vertices with vertex set $V = \{0, 1, 2, .., n-1\}$ such that $v_i v_j \in E$ if and only if either $i = (j + 1)$ or $(j - 1)$ modulo n.

The intersection of a clique and an independent set in a graph G can be at most one. Hence, $\omega(G) \leq \chi(G)$ and $\alpha(G) \leq k(G)$. These equalities are dual to one another since $\omega(G) = \alpha(\overline{G})$ and $k(G) = \chi(\overline{G})$.

Definition 9. *Consider a class of graphs $\mathcal{G}$, a* minimal obstruction *to $\mathcal{G}$ is a graph H such that $H \notin \mathcal{G}$, and $H - v \in \mathcal{G}$ for every vertex $v \in V(H)$.*

Definition 10. *Consider a class of graphs $\mathcal{G}$, a set $\mathcal{F}$ of minimal obstructions to G is* complete *if every graph not in G contains an induced subgraph from $\mathcal{F}$.*

Definition 11. *Consider a class of graphs $\mathcal{F}$, a graph G is said to be $\mathcal{F}$-free, if G does not contain any member of $\mathcal{F}$ as an induced subgraph.*

1.2 Decision problems and tractability

Many problems in computer science can be posed as *decision* problems: problems that have a simple "yes" or "no" answer. This is in contrast to *optimization* problems, in which each feasible (i.e., "legal") solution has an associated value, and we wish to find a feasible solution with the best value. The optimization problems can be further divided into maximization problems, in which the objective is to find the largest "legal" value of a given parameter and minimization problems. In a minimization problem, the objective is to find the smallest legal value of a certain parameter. For example, in a problem we will call Max-Independent set, we are given an undirected graph G, and the objective is to find the largest set of independent vertices.

We can convert an optimization problem into a decision problem as follows: fix a bound on the parameter we aim to optimize and then for that fixed parameter we ask does there exists a feasible solution of size at least k (for maximization problems) or at most k (for minimization problems)? For example, a decision problem related to the Max-Independent set is the independent set problem.

Definition 12. *The independent set problem*
Input: An undirected graph $G = (V, E)$, integer k
Question: Does G contain an independent set of size at least k?

A decision problem is classified as *decidable* if there exists a terminating algorithm that correctly determines the answer for any given input. Problems that are not decidable are called *undecidable*. The halting problem is an example of an undecidable problem. The objective is to determine from the description of an arbitrary computer program and an input, whether the program will ever terminate or not.

In this class of decidable problems, the problems that can be computed using a similar amount of computational resources are grouped in a *complexity class*. The resources could be the time required to finish the computation or the space available for computation. Decidable problems are distinguished on the basis of whether they can be solved *efficiently* or not. Informally, the class of problems for which there exists an algorithm that runs in polynomial time, i.e., in time $O(n^k)$, for some fixed k, (with respect to some notion of the *size* n of the input) are said to be efficient and are grouped in a complexity class called P.

There are decidable problems that are not in P, i.e., can not be solved in time $O(n^k)$ for any k [41]. There are also decidable problems about which it is not known if they are in P or not in P. A subclass of these problems plays an important role in literature, the class of so-called NP-complete problems.

In the next section, we formally define the notions of a *problem*, of *solving* the problem, of the *size* of the input, of the class NP, and of NP-complete problems.

1.2.1 NP-completeness and the classes P and NP

Before we can formally define the complexity classes P and NP, we need a few notions from formal-language theory. Informally, we encode everything in Boolean strings, strings of 0's and 1's. For a decision problem, we construct a language of finite strings over 0's and 1's. A finite string in the language represents an instance of our decision problem. Using these constructs, we define what it means for an algorithm to solve a problem.

Formally, an *alphabet* $\sum$ is a finite set of symbols. A language L is defined as a set of strings that can be constructed from the alphabets in $\sum$. We define $\sum^*$ as the language of all finite strings that can be constructed over $\sum$. It can be observed that for any language L over $\sum$, $L \subseteq \sum^*$.

Consider $\sum = \{0, 1\}$, given an instance I of a decision problem S, we encode I into a boolean string; the length of this string is defined as the input size. The language L of the decision problem S consists of all strings which represent an encoding of an instance I of S in $\sum^*$. An algorithm A *accepts* a string $x \in \sum^*$ if A takes x as an input and outputs 1 and *rejects* if the output is 0. The set of all strings accepted by A is the language accepted by A. The algorithm decides the language L if for all strings $x \in L$, A accepts x, and for all strings $x \notin L$, A rejects x. There exists an algorithm that solves S if there exists an algorithm that decides the language L of S.

The class P consists of problems for which there exists an algorithm A that decides their language L in polynomial time. More specifically, they are problems that can be solved in time $O(n^k)$, for some constant k, where n is the size of the input. Algorithm A is called a *polynomial time algorithm*. In the language-theoretic framework we define P as follows.

Definition 13. *The complexity class P is the class of problems for which there exists a polynomial time algorithm that decides the corresponding language $L \subseteq \sum^*$.*

The class NP consists of those problems for which given a "certificate" of a solution, we can verify the certificate in polynomial time, i.e., the "yes" instances can be verified in polynomial time. Consider the independent set problem defined above. For a given instance of the independent set problem, we have an undirected graph $G = (V, E)$ and an integer k, a certificate would be a set $S \subseteq V$ of vertices such that $|S| = k$. Since checking if the set S is edgeless takes polynomial time, the independent set problem belongs to the class NP.

Definition 14. *A language L belongs to NP if there exist a polynomial time algorithm A and a constant c such that $L = \{x \in \{0,1\}^*$: there exists a certificate y with $y = O(|x|^c)$ such that $A(x, y) = 1\}$.*

We say that A *accepts* L in polynomial time. Any problem in P is also in NP, since if a problem is in P, then we can solve it in polynomial time without needing the certificate. A problem is NP-complete if it is in NP and is as "hard" as any other problem in NP. Thus, NP-complete consists of the *hardest* problems in class NP. To show that a decision problem is no harder or easier than another decision problem, we use transformation procedures called *reductions*, and they have the following characteristics:

- The transformation takes polynomial time.
- "yes"-instances are mapped to "yes"-instances and "no"-instances are mapped to "no"-instances.

We say that a language L_1 of a decision problem is polynomial time reducible to L_2, written $L_1 \leq_p L_2$ if there exists a polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in L_1$ if and only if $f(x) \in L_2$.

Definition 15. *A language $L \subseteq \{0,1\}^*$ is NP-complete if :*

- *$L \in NP$.*
- *$L' \leq_p L$ for every $L' \in NP$.*

If a language satisfies only the second property, we say that L is NP-hard. We now define the following problem.

Definition 16. *The boolean satisfiability problem*
Input: A boolean formula $F(x_1, x_2, ..., x_n)$.
Question: Does F evaluate to true?

The first problem that was proven NP-complete was the boolean satisfiability problem [11]. Furthermore, 21 basic problems known as Karp's 21 problems, were proven NP-complete [50], including the independent set problem defined above and the following problems defined below.

Definition 17. *The k-clique problem*
Input: An directed graph $G = (V, E)$ and an integer k.
Question: Does G contain a clique of size k?

The k-colouring problem we discussed in 1.1 was also proved NP-complete in [50].

A literal in a boolean formula is an occurrence of a variable or its negation. A boolean formula is in conjunctive normal form (CNF) if it is expressed as an AND of clauses, each of which is the OR of one or more literals. A boolean formula is in 3-conjunctive normal form (3-CNF) if each clause has exactly three distinct literals. A formula ϕ is satisfiable if there exists a truth assignment such that ϕ evaluates to true.

Definition 18. *The 3-CNF satisfiability problem*
Input: A formula ϕ in 3-CNF.
Output: Is ϕ satisfiable?

1.3 Restricted graph classes

As discussed above, the colouring problem is NP-complete on the general class of graphs. However, if we restrict the class of graphs, it is possible to obtain efficient algorithms for the colouring problem as well as many other similar problems. In this section, we discuss the various classes of graphs on which these kinds of problems can be efficiently solved. In section 1.6, we also a large class of important graph classes of bounded tree-width and bounded clique-width on which these problems can be solved in linear time.

In this section, we focus on other graph classes, including the class of planar graphs and the class of perfect graphs and its subclasses.

Definition 19. *A graph is* planar *if it can be embedded in the plane without edges crossing each other.*

The edges, faces and vertices of a planar graph are related by Euler's formula as follows.

Theorem 1. *(Euler's Formula [64])*
If a connected planar graph G has exactly n vertices, e edges, and f faces, then $n-e+f = 2$.

To understand the next result, we first define the subdivisions of a graph. A *subdivision* of a graph is a graph obtained from it by replacing edges with pairwise internally vertex-disjoint paths.

Theorem 2. *(Kuratowski's theorem [52])*
A graph is planar if and only if it does not contain a subdivision of K_5 or $K_{3,3}$.

There are many linear time algorithms to test planarity, e.g.,[47, 5].

Theorem 3. *([47, 5]) Planar graphs can be recognized in linear time.*

Next, we consider the class of perfect graphs.

Definition 20. *A graph G is a* perfect graph *if the chromatic number of every induced subgraph of G equals the size of the largest clique in the subgraph.*

The following result is known as the perfect graph theorem.

Theorem 4. *([55]) For an undirected graph G, the following statements are equivalent:*

1. $\omega(G[A]) = \chi(G[A])$ *(for all $A \subseteq V$).*
2. $\alpha(G[A]) = k(G[A])$ *(for all $A \subseteq V$).*
3. $\omega(G[A]).\alpha(G([A]) \geq |V(G)|$ *(for all $A \subseteq V$).*

Corollary 5. *A graph G is perfect if and only if its complement $\overline{G}$ is perfect.*

The strong perfect graph theorem [9] provides a minimal obstruction characterization of perfect graphs.

Theorem 6. *([9]) A graph G is perfect if and only if G and $\overline{G}$ contain no induced odd cycles.*

Next, we consider various subclasses of perfect graphs. The first subclass that we study is the class of chordal graphs. A *chord* in a cycle $C = \{v_1, v_2, ..., v_n\}$ is an edge that connects any two non-consecutive vertices v_i, v_j in the cycle.

Definition 21. *A graph G is a* chordal graph *if every cycle of length greater than three has a chord.*

Theorem 7. *([2] and [37]) Chordal graphs are perfect.*

Chordal graphs are characterized by the possession of a *perfect elimination ordering.*

An ordering α of G is a bijection $\alpha : V \to \{1, 2, .., n\}$. For $1 \leq i \leq n$, we define $\mathcal{L}_i$ to be the set of vertices with labels greater than $i - 1$.

$$\mathcal{L}_i := \{v_i, v_{i+1}, .., v_n\}.$$

Definition 22. *A* simplicial *vertex of a graph G is a vertex v such that the neighbours of v induce a clique in G.*

The ordering α is a *perfect elimination ordering* if, for $1 \leq i \leq n$, the vertex v_i is simplicial in the graph $G[\mathcal{L}_i]$.

Theorem 8. *([31]) A graph G is chordal if and only if G has a perfect elimination ordering.*

An algorithm called the maximum cardinality search (MCS) [61] for chordal graphs orders the vertices in reverse order beginning with an arbitrary vertex $v \in V$ for which it sets $\alpha(v) = n$. At each step, the algorithm selects an unlabelled vertex u as the next vertex such that u is adjacent to the largest number of labelled vertices, with ties broken arbitrarily. The MCS algorithm runs in $O(|V| + |E|)$ time and can be used to recognize chordal graphs as demonstrated by the next theorem.

Theorem 9. *([60]) An undirected graph $G = (V, E)$ is chordal if and only if the ordering produced by the MCS algorithm is a perfect elimination ordering.*

Testing whether an ordering of the vertices is a perfect vertex elimination scheme can be implemented to run in $O(|V| + |E|)$ time [34].

Corollary 10. *Chordal graphs can be recognized in polynomial time.*

Next, we present a subclass of perfect graphs called comparability graphs. The class of comparability graphs include all bipartite graphs.

Definition 23. *An* orientation *of an undirected graph $G = (V, E)$ is a directed graph $D = (V, A)$ such that $V(G) = V(D)$ and for each edge $uv \in E$, either $uv \in A$ or $vu \in A$.*

Definition 24. *A* transitive orientation *of a graph G is an orientation D such that $V(D) = V(G)$, and whenever $xy \in A(D)$ and $yz \in A(D)$ are arcs in D, then G must contain an edge xz which is oriented from x to z in D. A simple graph G is a* comparability graph *if it has a transitive orientation.*

Proposition 11. *([34]) Comparability graphs are perfect.*

Next, we consider the class of interval graphs.

Definition 25. *An* interval graph *is an undirected graph where the vertices represent the intervals on the real line, and two vertices form an edge if and only if their corresponding intervals intersect.*

The following theorem states the relationship between chordal graphs, comparability graphs and interval graphs.

Theorem 12. *([33]) Let G be an undirected graph. The following statements are equivalent:*

1. *G is an interval graph.*
2. *G contains no chordless four cycles, and its complement $\overline{G}$ is a comparability graph.*
3. *The maximal cliques of G can be linearly ordered such that for every vertex v of G, the maximal cliques containing v occur consecutively.*

Definition 26. *An* astroidal triple *is an independent set containing three vertices such that any two vertices are joined by a path that avoids the neighbourhood of the third vertex.*

Using the above definition, we describe the minimal obstruction characterization of interval graphs.

Theorem 13. *([53]) An undirected graph G is an interval graph if and only if the following two conditions are satisfied:*

- *G is a chordal graph.*
- *G contains no astroidal triple.*

A *clique matrix* of a graph G is a matrix M, where the rows of M represent the vertices of G, and the columns of M represent the maximal cliques of the graph such that $M(i, j) = 1$ if and only if vertex $v_i \in V(G)$ belongs to a maximal clique c_j in G. Interval graphs have consecutive 1's property in the clique matrix.

Theorem 14. *([33]) A graph is an interval graph if and only if its maximal clique can be linearly ordered such that for every vertex in the graph, the maximal cliques to which it belongs occur consecutively in the linear order.*

Using Theorem 13 and 14, a recognition algorithm for interval graphs can be developed as follows. First, we test if the given graph G is chordal, which takes $O(|V| + |E|)$ time, as discussed earlier. If G is chordal, we test for the consecutive $1's$ property of its clique matrix. Booth and Leuker [5] proved that the above step takes $O(|V| + |E|)$ time.

Theorem 15. *([5]) Interval graphs can be recognized in linear time.*

Several other linear time recognition algorithms are known for interval graphs, for example, [51] and [12].

Next, we move our discussion to the class of split graphs.

Definition 27. *A* split graph *is a graph in which the vertex set of G can be partitioned into a disjoint clique and an independent set.*

In general, the partition mentioned above is not unique for a split graph. Moreover, the clique need not be a maximum clique, and the independent set need not be a maximum independent set.

Since an independent set of G is a complete subgraph in the complement of G and vice versa, split graphs are closed under taking the complement.

Theorem 16. *An undirected graph G is a split graph if and only if its complement $\overline{G}$ is a split graph.*

The following is known for the class of split graphs.

Theorem 17. *([30]) Let G be an undirected graph. The following conditions are equivalent:*

1. *G is a split graph.*
2. *G and $\overline{G}$ are chordal graphs.*
3. *G contains no induced subgraph isomorphic to $2K_2$, C_4, or C_5.*

A polynomial time recognition algorithm for split graphs can be developed using the minimal obstruction characterization in Theorem 17 and can be found in [39].

Corollary 18. *Split graphs can be recognized in polynomial time.*

Authors in [45] provide a linear time algorithm for recognizing split graphs. Next, we introduce our final subclass of perfect graphs, cographs. Cographs are one of the most popular and intensively studied classes of perfect graphs. Since cographs are perfect, many intractable problems can be solved in polynomial time on the class of cographs [34]. In this thesis, our focus will be on the class of cographs and the various partition problems on cographs.

Definition 28. *A graph G is a* cograph *if it has no induced subgraph isomorphic to P_4, the path on four vertices.*

There is also an equivalent recursive definition of cographs. First, we define two operations on cographs, which are the building blocks of the recursive definition.

Definition 29. *The* disjoint union *of two graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ denoted by $G = (G_1 + G_2)$, is a graph G such that G has the vertex set $V = V_1 + V_2$ and the edge set $E = (E_1 + E_2)$.*

Definition 30. *The* join *of two graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ denoted by $G = (G_1 \oplus G_2)$, is a graph G such that G has the vertex set $V = (V_1 + V_2)$ and the edge set $E = (E_1 + E_2 + E_{12})$, where E_{12} contains all edges $e = uv$, where $u \in V_1$ and $v \in V_2$.*

Theorem 19. *([16]) A cograph is a graph that can be constructed recursively as follows:*

- *K_1 is a cograph.*
- *If G and H are cographs, then $G \oplus H$ is a cograph.*
- *If G and H are cographs, $G + H$ is a cograph.*

From the above characterization of cographs, one can see that cographs are closed under taking the complement.

Theorem 20. *An undirected graph G is a cograph if and only if its complement $\overline{G}$ is a cograph.*

A unique tree representation of cographs is used to develop polynomial time algorithms for problems such as isomorphism, clique determination and colouring. The rooted tree representing the parse structure of a cograph is called a *cotree* of the cograph. The leaves of a cotree are the vertices of the corresponding cograph, and each internal node represents either the join operation or the disjoint union operation alternatively. The cotree for a particular cograph thus constructed is unique up to a permutation of the children of the internal nodes. This cotree representation of cographs can be used to determine the chromatic number of cographs in polynomial time as follows:

- Initialize leaves of cotree with $\chi = 1$.
- Perform the addition operation on join nodes, i.e., for a join node $\chi = \sum_1^k \chi_i$ where χ_i is the chromatic number of child i, and the join node has k children.
- Perform the max operation on disjoint union nodes, i.e., for a disjoint union node $\chi = max(\chi_1, \chi_2, ..., \chi_k)$ where χ_i is the chromatic number of child i, and the disjoint union node has k children.

There also exist linear time algorithms to determine the chromatic number of cographs in linear time [10]. Since cographs are perfect, this also calculates the clique number.

The cotree characterization can also be used to develop linear time recognition algorithm for cographs. The linear time algorithm in [16] builds a cotree, starting from a single vertex and adding a new vertex at each step of the computation. When a vertex x is added, the cotree has to be updated using at most $O(deg(x))$ operations, which is far from being obvious.

Theorem 21. *([16]) Cographs can be recognized in linear time.*

Lastly, we discuss a result by Damaschke.

Definition 31. *A property* Π *of graphs is* hereditary *if any induced subgraph of a graph having property* Π *also has property* Π.

For example, the class of cographs is a hereditary class of graphs since every induced subgraph of a P_4-free graph is also P_4-free. Let $\mathcal{C}$ be a hereditary class of graphs, and $\mathcal{F}_\mathcal{G}$ represents the minimal obstructions of a hereditary class $\mathcal{G}$, i.e., $G \in \mathcal{G}$ if and only if there is no $F \in \mathcal{F}$ with $F \leq_i G$, where $\leq_i$ represents the induced subgraph relation.

Theorem 22. *([19]) For any hereditary property* Π*, the class of cographs has a finite complete set of minimal obstructions.*

1.4 Colouring and related partition problems

In this section, we study the graph colouring problem and various related problems. We also discuss how various partition problems can be seen as an extension of the colouring problem.

Graph colouring is one of the central problems in graph theory. As discussed in section 1.1, determining the chromatic number of a graph is NP-hard in general. Finding a k-colouring is polynomial for $k = 1, 2$; however, it is NP-complete for $k \geq 3$.

We have also seen earlier how the k-colouring problem can be seen as a partition problem of the vertex set such that there is a restriction on the adjacencies of the vertices in each set, i.e., there should be no edges between vertices in the same set. However, for vertices in two different sets, there are no constraints on their adjacencies. Hence colouring, in essence, is a partition problem with some constraints imposed on the parts. At this point, it is possible to see that we can derive the following abstraction for the partition problems:

- The first objective is to divide the vertex set of a graph into pre-specified sets; for example, in the case of k-colouring, the objective is to partition into k disjoint(possibly empty) sets.

- Impose constraints on the relationship of vertices within each set; for example, all the vertices in each set must be non-adjacent.

- Impose constraints on the relationship of vertices across sets; for example, in k-colouring, the constraint is that there are no constraints; we refer to this as the "don't care" constraint.

Naturally, the question arises if interesting problems could be obtained by varying each of the above constraints: for example, we could fix the size of each disjoint set or specify the intersection between each set. We could fix the adjacency constraints within each set differently, similarly, for the adjacency constraints across the sets. In the following discussion, we analyze different problems that arise from this abstraction. The above abstraction will remain a constant in all of the partition problems that we will discuss.

For a given partition problem, we are also interested in a minimal obstruction characterization whenever possible.

The first problem that we discuss is the cocolouring problem: the objective here is also to partition the vertex set into disjoint sets, but we change the adjacency constraints as follows. Each set induces either an independent set or a clique. This problem has no constraints on adjacencies across the sets.

Definition 32. Cocolouring *of a graph G is an assignment of colours to the vertices of G such that each colour class induces either an independent set or a clique in G.*

Definition 33. *The* cochromatic number *of G is the minimum number k for which such an assignment is possible.*

A (k, ℓ)-cocolouring of a graph is a cocolouring where the objective is to partition the vertex set into at most k independent sets and at most ℓ cliques. The case of $(1, 1)$-partition thus corresponds to the class of split graphs, i.e., split graphs have cochromatic number 2.

Similarly, the case of $(2, 0)$ for cocolouring corresponds to the class of bipartite graphs. The cocolouring problem was originally studied in [54], and can be seen as a natural extension of the colouring problem. Determining the cochromatic number of a graph is NP-hard in general [40].

Now, if one allows for more generalization of the adjacency constraints, we obtain the H-colouring problem. Here the objective is also partitioning the vertex set of a given graph into disjoint sets, but another graph is used to model the adjacency constraints across sets and within each set.

Let H be a fixed graph, whose vertices are referred to as colours.

Definition 34. *An H-*colouring *of a graph G is an assignment of colours to the vertices of G such that the adjacent vertices of G obtain adjacent colours.*

For a fixed graph H, the decision problem here is, given an input graph G, is it possible to H-colour G? An H-colouring of G also describes a homomorphism from $G \to H$.

Definition 35. *A* homomorphism *$f : G \to H$ is a mapping f of $V(G)$ to $V(H)$ such that $f(g)f(g') \in E(H)$ whenever $gg' \in E(G)$.*

When $H = K_n$, an H-colouring of G is just an n-colouring of G. Hence, H-colouring can also be seen as a generalization of the k-colouring problem.

Finally, we arrive at the M-partition problem, where a matrix is used to model the adjacency constraints [28]. For each fixed $\{0, 1, *\}$-matrix M, the M-partition problem on a given graph G, is the problem of finding a partition of $V(G)$ that respects the constraints specified by the matrix M.

Definition 36. *An M-partition of a graph G is a partition of $V(G)$ into parts $V_1, V_2, ..., V_m$ such that for distinct vertices $u \in V_i$, $v \in V_j$, we have $uv \in E(G)$ if $M(i, j) = 1$, and $uv \notin E(G)$ if $M(i, j) = 0$. Note that we admit $i = j$; in particular, if $M(i, i) = 0$, the set V_i is independent in G, and if $M(i, i) = 1$, it is a clique. Also note that * means no restriction.*

The M-partition problem generalizes all partition problems that we have discussed previously. It is easy to see that M-partition problems include all homomorphism problems (and hence all graph colouring problems). Indeed, if H is a graph, we let M be the adjacency matrix of H with 1's replaced by *, then an M-partition of a graph G is precisely a homomorphism of G to H.

In the list version of the M-partition problem, every vertex v has a list $L(v)$ such that v can only be assigned to a vertex in $L(v)$. List versions also exist for k-colouring, and H-colouring problems.

1.5 Partition problems and restricted classes of graphs

All the problems that we have introduced in the previous section are NP-complete on the general class of graphs. In this section and also in section 1.6, we try and restrict the classes of graphs and see whether efficient algorithms can be obtained.

1.5.1 Graph colouring

We begin our discussion with the colouring problem. Using a greedy strategy, where in some ordering of the vertices, the next vertex is coloured with the smallest colour, we obtain a colouring that uses at most $\Delta + 1$ colours, where Δ is the maximum degree of any vertex in the graph.

Proposition 23. *For any graph G, $\chi(G) \leq \Delta(G) + 1$.*

This upper bound can be easily improved by sorting the vertices by their degree and then applying the greedy colouring.

Proposition 24. *([63]) If a graph G has a degree sequence $d_1 \geq d_2 ... \geq d_n$, then $\chi(G) \leq 1 + max_i(min\{d_i, i-1\})$.*

The bound $\chi(G) \leq 1 + \Delta(G)$ can be improved further if a graph is not complete or does not contain any odd cycles. The following result is known as the Brooks' theorem.

Theorem 25. *(Brooks' theorem [7]) If G is a connected graph other than a complete graph or an odd cycle, then $\chi(G) \leq \Delta(G)$.*

It is trivial to see that any graph which contains a complete subgraph on k vertices cannot be coloured with k colours. However, the opposite is not true as there exist triangle-free graphs with an arbitrarily large chromatic number [57].

Theorem 26. *([57]) For any positive integer k, there exists a triangle-free graph with chromatic number k.*

Hence, the chromatic number of a graph can be arbitrarily larger than the size of the largest clique in the graph. In general, it is also possible to have large girth and yet still have an arbitrarily large chromatic number. We have the following result by Erdős.

Theorem 27. *([26]) For all k, l, there exists a graph G, with girth(G) $> l$ and $\chi(G) > k$.*

We begin our discussion of the colouring problem on the restricted classes of graphs that we have discussed earlier. We start with the class of planar graphs. First, we note that Theorem 26 does not hold for planar graphs as the graphs that appear in the proof of Theorem 26 are not planar. Using the Euler's formula, we know that any planar graph has at most $3n - 6$ edges, such that a graph has a vertex of degree at most five. Since any subgraph of a planar graph is also planar, an inductive proof proves that any planar graph is six-colourable. Heawood improved the bound to five colours.

Theorem 28. *(Five-colour theorem [42]) Every planar graph is five-colourable.*

We know that planar graphs can not contain K_5 as an induced subgraph. So is it possible to bound the chromatic number of a planar graph by at most four? The popular version of this question was called the four-colour conjecture. Is it possible to colour all the maps that can be drawn on a plane using at most four colours? The maps correspond to planar graphs as follows. A region in the map is a vertex, and there is an edge between two vertices if their corresponding regions share a boundary. The four-colour conjecture was proved in 1976 by Kenneth Appel and Wolfgang Haken.

The four-colour theorem states that any planar graph can be coloured using at most four colours.

Theorem 29. *(Four-Colour Theorem [1]) Every planar graph is four-colourable.*

For triangle-free planar graphs, we have the following result by Grőtzsch [36].

Theorem 30. *([36]) Every triangle-free planar graph is three-colourable.*

A three-colouring of triangle-free planar graphs can be found in linear time [24].

The k-colouring problem is polynomial on the class of perfect graphs. A perfect graph is k-colourable if and only if it does not contain K_k as a subgraph. A polynomial time algorithm for determining the chromatic number is known for perfect graphs, using the ellipsoid method for linear programming [35]. However, no purely combinatorial algorithm is known.

For chordal graphs, there exists an algorithm that correctly calculates the chromatic number and all maximal cliques of a chordal graph $G = (V, E)$ in $O(|V| + |E|)$ time using the perfect elimination ordering of graph G. Recall that a graph is chordal if and only if it has a perfect elimination ordering (Theorem 8). The following can be said about the graphs that have a perfect elimination ordering.

Theorem 31. *There is a polynomial time algorithm to solve the colouring problem on graphs with perfect elimination ordering.*

Proof. Suppose the graph G has a perfect elimination ordering as follows: $\{v_1, v_2, .., v_n\}$, and let k denote the size of the largest clique in G. We enumerate the colours as $1, 2, ..,$. We begin with the first vertex in the perfect elimination ordering and assign it colour 1. For every vertex v_i, $i > 1$, we assign the smallest colour unused by the neighbours of v_i among $v_1, v_2, .., v_{i-1}$. Each vertex v_i is guaranteed to have most k neighbours in $v_i + 1, , .., v_n$, otherwise G has a clique of size $k + 1$. Hence, every vertex can be assigned a colour between $1, 2, ..k$. Hence, at most k colours are used in the algorithm. We know that $\chi(G) \geq k$. Hence, the algorithm correctly determines the colouring of G. The colouring algorithm discussed here runs in $O(n)$ time. □

Corollary 32. *There exists a polynomial time algorithm that correctly determines the chromatic number on the class of chordal graphs.*

Interval graphs are also characterized by the existence of a perfect elimination ordering, as can be seen from Theorem 14. The ordering produced by Theorem 14 is actually a perfect elimination ordering.

Corollary 33. *There exists a polynomial time algorithm that correctly determines the chromatic number on the class of interval graphs.*

From Theorem 17, we know that split graphs are chordal graphs, the complements of which are also chordal.

Corollary 34. *There exists a polynomial time algorithm that correctly determines the chromatic number on the class of split graphs.*

1.5.2 Cocolouring

Recognizing (k, l)-partitionable graphs is NP-complete for $k \geq 3$ or $l \geq 3$. A polynomial time recognition algorithm for the cases $(2, 1)$ and $(1, 2)$ follows from [28].

A minimal obstruction to (k, l)-cocolouring is a graph G such that the graph G does not admit a (k, l)-partition, but any proper induced subgraph of G does. For the (k, ℓ)-cocolouring problem, a minimal obstruction characterization is known for cographs.

Theorem 35. *([22]) A cograph G is (k, ℓ)-partitionable if and only if it does not contain any $(\ell + 1)^* K_{k+1}$ as an induced subgraph.*

The graph $(p^* K_r)$ is the configuration formed by p copies of a clique of size r with forbidden edges between corresponding vertices in the cliques. Please see figure 1.1.

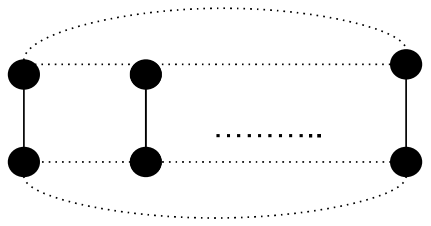


Figure 1.1: The configuration $(p^* K_2)$, the dotted edges are forbidden edges, the edges that are neither there nor forbidden are optional.

Theorem 36. *([23]) A maximum induced (k, ℓ)-cocolourable subgraph can be computed in time $O((k^3 \ell + k \ell^3) n)$ in cographs defined by their cotree.*

For chordal graphs, the minimal obstruction characterization and a polynomial time recognition algorithm for the (k, ℓ)-cocolouring problem is given in [45].

A set of subgraphs is independent if they are vertex disjoint with no edges joining any two subgraphs. The following is known for the finite minimal obstruction characterization of chordal graphs for the (k, ℓ)-partition problem.

Theorem 37. *([45]) A chordal graph is (k, ℓ)-partitionable if and only if it does not have $(\ell + 1)$ independent copies of K_{k+1}.*

We note that $(\ell + 1)$ independent copies of K_{k+1} is just the configuration $(\ell + 1)^* K_{k+1}$ with all optional edges missing. Let $f(G, r)$ denote the maximum number of independent copies of K_r in G, and by $g(G, r)$ the minimum number of cliques of G, which meet all K_r of G.

Theorem 38. *([45]) Let G be a chordal graph, and let $r \leq 1$ be an integer. Then $f(G, r) = g(G, r)$.*

Since k and ℓ are fixed, there are only polynomially many subgraphs of G with $(\ell + 1)(k + 1)$ vertices. Hence, Theorem 38 gives a polynomial time recognition algorithm for chordal (k, l)-partitionable graphs. The authors in [45] provide a $O(m(n + m))$ algorithm for recognition of chordal (k, ℓ)-partitionable graphs, where n is the number of vertices in the graph and m is the number of edges. The case of $k = 1$ and $\ell = 1$ corresponds to the class of split graphs. For this case, the authors provide a more efficient algorithm that runs in $O(m + n)$ time. For the cocolouring problem on cographs, the following is known.

Theorem 39. *([23]) A maximum induced (k, ℓ)-cocolourable subgraph can be computed in time $O((\ell^3 k + \ell k^3)n)$ in cographs defined by their cotree.*

Theorem 40. *([23]) For any cograph G given by its cotree, a minimum cocolouring is obtained by the greedy cocolouring algorithm in time $O(n^{3/2})$.*

1.5.3 H-colouring

For the H-colouring problem, the following dichotomy is known.

Theorem 41. *([46]) If H is bipartite, then the H-colouring problem is in P. If H is not bipartite, then the H-colouring problem is NP-complete.*

The following is also known for the H-colouring problem.

Theorem 42. *([43]) If H has no edges, then the H-colouring problem has a finite complete set of minimal obstructions. Otherwise, the H-colouring problem has infinitely many minimal obstructions.*

A single minimal obstruction characterization is known for the class of perfect graphs for the H-colouring problem: a perfect graph is H-colourable if and only if it does not contain a clique of size one greater than the chromatic number of H.

1.5.4 M-partition

The k-colouring problem and the (k, ℓ)-cocolouring problem can be modelled as M-partition problems. For example, the three-colouring problem and the $(2, 2)$-cocolouring problem correspond to the following matrices M_1, M_2, respectively (Please see Figure 1.2).

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

Figure 1.2: Matrices M_1 and M_2 respectively.

Theorem 43. *([29]) When asterisks are absent in M, the size of all minimal obstructions to M-partitions is bounded.*

For general matrices, the following dichotomy is known regarding the M-partition problem on small patterns.

Theorem 44. *([29]) Let the pattern M be a symmetric $\{0, 1, *\}$-matrix of size at most four, with a $\{0, 1\}$-diagonal. If M contains, as a principal submatrix, the pattern of three-colouring, or its complement, then the M-partition problem is NP-complete. Otherwise, the M-partition problem is polynomial time solvable.*

In general, it is not known for which matrices the M-partition problem is polynomial time solvable, and for which matrices the problem is NP-complete.

For the class of split graphs, the following is known.

Theorem 45. *([59]) For any matrix M, the M-partition problem has a finite complete set of minimal split obstructions.*

Thus, all M-partition problems for split graphs are polynomial time solvable.

The M-partition problem is also polynomial time solvable on the class of interval graphs.

Theorem 46. *([62]) For all patterns M, the list version of the M-partition problem is polynomial time solvable on the class of interval graphs.*

On the other hand, it is not known for which matrices, there are finite minimal interval graph obstructions and for which matrices the number of obstructions are infinite.

For the class of chordal graphs, the following is known regarding the M-partition problem.

Theorem 47. *([27]) For an $m \times m$ matrix M such that $m < 5$, the M-partition problem is known to be polynomial on the class of chordal graphs.*

Theorem 48. *([27]) For matrices M such that $m < 4$, M has a finite complete set of minimal split obstructions, except for two matrices, which have infinitely many minimal chordal obstructions.*

Theorem 49. *Any minimal M-obstruction cograph G has at most $O(8^m/\sqrt{m})$ vertices.*

Hence, the M-partition problem has a finite complete set of minimal cograph obstructions [20, 27], and thus, the M partition problem is polynomial on the class of cographs.

Theorem 50. *([27]) There exists a matrix M for which the M-partition problem has a cograph minimal obstruction with $m^2/4$ vertices.*

1.6 Clique-width and cographs

A graph can also be considered as a logical structure. Hence, many properties of graphs are definable in first-order logic and second-order logic. In this section, we study a fragment of second-order logic called Monadic Second Order (MSO) logic and study its variations MSO_1 and MSO_2. In first-order logic, formulas over graphs are built up from atomic formulas $E(x, y)$ and $x = y$ using boolean connectives such as negation $\neg$, conjunction $\wedge$, disjunction $\vee$, logical implication $\Rightarrow$, and logical bi-implication $\Leftrightarrow$. The formula $E(x, y)$ represents the adjacency structure of the given graph, $x = y$ denotes the equality relation, and the variables x, y range over the vertices of the graph. In MSO, we have additional atomic formulas $X(x)$, for set variables X and existential quantification and universal quantification is permitted over set variables. In MSO_1, quantification is permitted only over sets of vertices, whereas in MSO_2 quantification is permitted over sets of edges as well as sets of vertices. The following MSO_1 formula defines the property of three-colourabilty of a graph.

$\exists X_1, X_2, X_3\ (Part(X_1, X_2, X_3)) \wedge x, y(E(x, y) \wedge \neg(x = y) \Rightarrow \neg(x \in X_1 \wedge y \in X_1) \wedge \neg(x \in X_2 \wedge y \in X_2) \wedge \neg(x \in X_3 \wedge y \in X_3))$.

The formula $Part(X_1, X_2, X_3)$ expresses that X_1, X_2, X_3 partition the vertex set, and is written as follows:
$\forall x \in V(G)((x \in X_1 \vee X_2 \vee X_3) \wedge (\neg(x \in X_1 \wedge x \in X_2)) \wedge (\neg(x \in X_2 \wedge x \in X_3)) \wedge (\neg(x \in X_1) \wedge (x \in X_3)))$.

Other examples of MSO_1 graph properties include k-colourability, (k, l)-partitionability as well as the (p, q)-partitionability that we will study in the next chapter. Next, we define the terms tree-width of a graph and clique-width of a graph. The tree-width of a graph denotes the size of the largest vertex set in a tree-decomposition of a graph.

Definition 37. *The* tree-decomposition *of a graph $G = (V, E)$ is a pair X, T, where $X = \{X_1, X_2, ..., X_n\}$ is a family of subsets of $V(G)$, and T is a tree such that:*

- $X_1 \cup X_2 \cup ... \cup X_n = V(G)$.
- *For every edge $v, w \in E(G)$ of a graph, there exists a subset X_1 that contains both v and w.*
- *If X_i, X_j both contain a vertex v, then every node X_k of tree T in the path from X_i to X_j in the tree T contains the vertex v. That is, nodes containing v induce a connected graph in T.*

Definition 38. *The* width *of a tree decomposition is $max_{i \in \{1,2,..n\}} |X_i| - 1$, and the* tree-width *of a graph G is the minimum width of a tree decomposition of G.*

When the width is fixed, the tree compositions can be computed in linear time [3].

Definition 39. Clique-width *of a graph G is defined as the minimum number of labels required to construct G using following the four operations:*

- *Creation of a new vertex with label i, (denoted $i(v)$).*
- *The disjoint union of two graphs G and H, i.e., $G + H$.*
- *Adding an edge between each vertex with label i and label j, (denoted η_{ij}).*
- *Renaming label i to j, (denoted $p_{i \to j}$).*

Any graph can be obtained using the above four operations. For example, the following expression defines a four cycle on vertices $\{a, b, d, c\}$, such that: $\eta_{12}(p_{4\to 2}(p_{3\to 1}(1(a)+2(b)+3(c)+4(d))))$. An expression using the above four operations that use at most k labels is called a k-expression defining a graph. Thus, the clique-width of a graph G is the minimum k for which a k-expression exists that defines G. It is known that graphs with tree-width at most k have clique-width at most 3.2^{k-1} [15]. Hence, any graph that has bounded tree-width also has bounded clique-width.

A large class of optimization problems can be solved efficiently using dynamic programming on graphs with bounded tree-width [4, 6]. In fact, for many problems, linear time algorithms can be obtained on graphs with bounded tree-width. Specifically, Courcelle's Theorem [17] states that problems of bounded tree-width that can be defined in MSO_2 can be solved in linear time. Moreover, the theorem also states that problems of bounded clique-width that can be defined in MSO_1 can be solved in linear time.

In this thesis, our focus is on cographs and partition problems on cographs. It is known that cographs are exactly the graphs with clique-width at most two [18]. Hence, cographs are graphs with bounded clique-width.

The various partition problems we discussed earlier, for example, the k-colourability problem, (k, l)-partitionability, and the (p, q)-partitionabilty problems that we will study in the next chapter can be defined in MSO_1. Since cographs have bounded clique-width, these problems can be solved in linear time on cographs. Hence, the partition problems that we discuss in this thesis can be solved in linear time on cographs.

1.7 Summary of our results

In this thesis, we study a partition problem where a cograph is to be partitioned into p forests and q independent sets, called the (p, q)-partition problem.

In Chapter 2, after summarizing known previous results, we present new results for $p = 2$ and $q = 1$, describing all the minimal obstructions to $(2, 1)$-partition, see Theorem 53, cf. also [44]. There are 9 minimal obstructions in total. We address the open question of whether a uniform description of minimal obstructions is possible for $(2, q)$-partitions for any $q \geq 0$. We then transform the proof of completeness of various cases with the following running times: $O(m+qn)$ for $(1, q)$-partitions $q \geq 0$, or $O(qn)$ if the cotree is given, $O(m+n)$ or $O(n)$ if the cotree is given for $(2, 0)$-partition and $(2, 1)$-partition problems.

The (p, q)-partition problem can be defined using MSO_1, and from the above discussion, we know that there exists a linear time algorithm for the problem of (p, q)-partition on cographs using Courcelle's Theorem. However, a certifying algorithm provides a certificate for both a positive outcome - a (p, q)-partition and a negative outcome - a minimal obstruction. This is useful for example, it provides the ability to test a particular implementation of the partitioning algorithm by either finding such a partition or providing a certificate(a minimal obstruction) in case no such partition exists. (Similar certifying algorithms are also described for earlier known results.)

In Chapter 3, we expand the problem to a relaxed problem in which we seek a partition into p forests and q independent sets, after at most r vertices have been deleted. We solve the problem for $p = 1$, $q = 1$, $r = 1$, and $p = 1$, $q = 2$, $r = 1$, describing all the minimal cograph obstructions to these partition problems, see Theorem 63 and Theorem 69.

Chapter 2

(p,q)-partition

2.1 Background and previous results

The *vertex-arboricity* of a graph G is the minimum integer p such that the vertices of G can be partitioned into p parts, each of which induces a *forest*. It is, in general, NP-complete to decide if a graph G has arboricity less than or equal to a fixed p, $p \geq 2$ [38]. This is a situation analogous to the chromatic number of G, which is the minimum integer q such that the graph G is q-colourable for $q \geq 3$ [32]. In the previous chapter, we discussed a polynomial time algorithm to determine the chromatic number of cographs. This can be done in a very similar fashion for vertex-arboricity. The authors in [21] have studied, for cographs, a blended problem, whereby a graph is partitioned into p parts inducing forests and q parts that are independent sets (and more general partitions). Each of these problems can be efficiently solved in the class of cographs, and in fact, characterized by a finite number of minimal cograph obstructions. This parallels the situation for a similar blended problem studied earlier, where a cograph G is to be partitioned into k independent sets and ℓ cliques [23, 22].

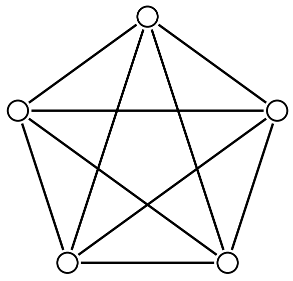
It follows from Theorem 22 that each of these problems has a characterization by a finite set of minimal cograph obstructions. Here a *minimal cograph obstruction* is a cograph G that does not admit a required partition, but each proper induced subgraph of G does admit such a partition. Thus a cograph admits a required partition if and only if it does not contain an induced subgraph isomorphic to a minimal cograph obstruction.

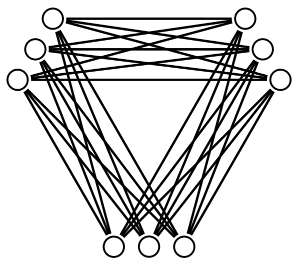
Minimal cograph obstructions for partition into k independent sets and ℓ cliques were described in [23, 25, 27, 22]; as discussed in the last chapter, the obstructions have $(k+1)(\ell+1)$ vertices and admit a partition into $k+1$ independent sets of size $\ell+1$ as well as a partition into $\ell+1$ cliques of size $k+1$.

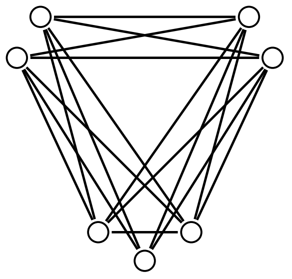
Minimal cograph obstructions for partition into p forests and q independent sets were investigated in [21]. Consider first the special case of $q = 0$, that is partitions into forests (arboricity). Since cographs are perfect, there are two minimal cograph obstructions for being a forest, i.e., admitting a partition with $p = 1$: these are the cycles C_3 and C_4.

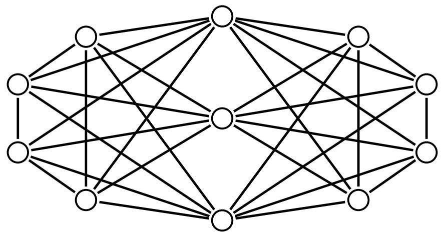
For partitions into $p = 2$ forests, there turn out to be exactly seven minimal cograph obstructions, forming the family $\mathcal{A}_2$ depicted in Figure 1.

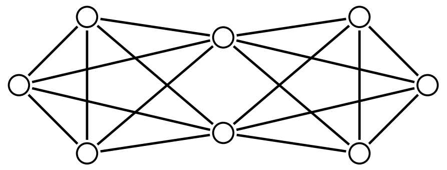
Theorem 51. *([21]) A cograph has $(2, 0)$-partition if and only if it is $\mathcal{A}_2$-free.*

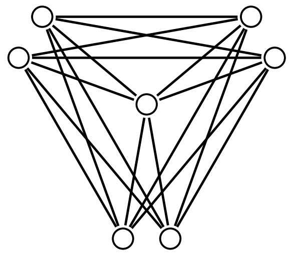

(a) K_5


(b) $\overline{3K_3}$


(c) $\overline{2K_2} \oplus (K_1 + K_2)$


(d) $2\left(\overline{2K_2}\right) \oplus \overline{K_3}$


(e) $2K_3 \oplus \overline{K_2}$


(f) $\overline{3K_2 + K_1}$

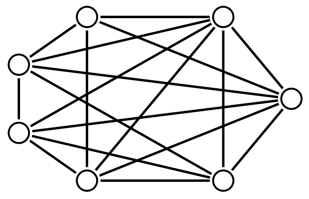

(g) $\left(\overline{2K_2} + K_3\right) \oplus \overline{K_2}$

Figure 2.1: The family $\mathcal{A}_2$.

Each of these obstructions has a natural generalization to minimal cograph obstruction for partition into p forests. For example, K_5 generalizes to K_{2p+1}, $\overline{3K_3}$ generalizes to $\overline{(p+1)K_{p+1}}$, and so on. These seven generalizations form a family $\mathcal{A}_p$, given by an explicit uniform description in [21]. They are all minimal cograph obstructions to partition into p forests. Nevertheless, it turns out that there are in general, many additional minimal cograph obstructions, and in fact, the number of minimal cograph obstructions for partition to p forests grows exponentially with p [21].

There is, however, a class of partition problems in which minimal cograph obstructions can be uniformly described. This is the class of problems generalizing the problem of independent vertex feedback set [21]. A *q-colourable vertex feedback set* of a graph G is a set

V of vertices such that $G \setminus V$ admits a q-colouring. Thus a graph admits a q-colourable vertex-feedback set if and only if it has a partition into $p = 1$ forest and q independent sets. It is shown in [21] that there are precisely two minimal cograph obstructions for such a partition, namely K_{q+3} and $\overline{(q+2)K_2}$. (Note that for $q = 0$ we again obtain C_3 and C_4 as the minimal cograph obstructions to being a forest.) This family describes all minimal cograph obstructions for partitions into $p = 1$ forest and q of independent sets, uniformly for all values of q.

Theorem 52. *([21]) Let q be a non-negative integer. A cograph G has a $(1, q)$-partition if and only if it is $\{K_{q+3}, \overline{(q+2)K_2}\}$-free.*

Proof. Since the cographs K_{q+3} and $\overline{(q+2)K_2}\}$ do not admit a $(1, q)$-partition, any graph G that has a $(1, q)$-partition does not contain K_{q+3} and $\overline{(q+2)K_2}$ as an induced subgraph.

Hence, we now prove that if G is $\{K_{q+3}, \overline{(q+2)K_2}\}$-free, then G admits a $(1, q)$-partition. We proceed by induction on q. For the base case of $q = 0$, observe that if G does not contain K_3 and $\overline{(q+2)K_2}$ as an induced subgraph, then G is a forest. Therefore, G admits a $(1, 0)$-partition. For the induction step, we assume that the result holds for all $q \leq k$, and we will prove for $q = k + 1$. Assume that G is $\{K_{k+4}, \overline{(k+3)K_2}\}$-free. Without loss of generality, we assume that G is connected. As cographs are perfect, and G is K_{k+4}-free, we find a $(k+3)$-colouring of G. If one of the colour classes contains at most one vertex, say vertex v, then we obtain a $(1, q)$-partition as follows. We take a star at v along with one of the colour classes; the remaining $(q + 1)$ colour classes form the independent sets of the desire $(1, q)$-partition. Hence, we may assume that each colour consists of at least two vertices. Since G is connected, using the cotree structure of cograph G, we obtain cographs $G_1, G_2, ..., G_l$ so that $G = \oplus_{i=1}^{l} G_i$. Hence, $\sum_{i=1}^{l} \chi(G_i) = k + 3$. There exists a $j \in \{1, 2, ..., l\}$ such that G_j is $\overline{\chi(G_j)K_2}$-free. Otherwise, G contains $\overline{(k+3)K_2}$ as an induced subgraph. Let $k' = \chi(G_j)$, observe that $k' \geq 2$ since each colour class contains at least two vertices, and G_j is $\overline{\chi(G_j)K_2}$-free. Since G_j is $\{K_{k'+1}, \overline{(k')K_2}\}$-free, G_j admits a $(1, k' - 2)$-partition. Also, note that the cograph $G \setminus V(G_j)$ can be coloured using at most $(k+3-k')$ colours. A $(k+3-k')$ colouring of $G \setminus V(G_j)$ together with the $(1, k'-2)$-partition of G_j yields a $(1, k+1)$-partition of G. □

As mentioned above, there is only one minimal cograph obstruction for partitions into ($p = 0$ forests and) an arbitrary number q of independent sets, namely K_{q+1}, which is again a family uniformly described for all values of q. This motivates the natural question of whether there are other values of p for which such uniformity is possible.

2.2 The $(2, 1)$-partition problem

In this chapter, we investigate the first open case of $p = 2$. In order to address the question of possible uniform description, we explicitly describe all minimal cograph obstructions for partition into $p = 2$ forests and $q = 1$ independent set. Each member of the family $\mathcal{A}_2$

again has a natural generalization as an obstruction for such a partition. For example, K_5 generalizes to K_6, because an independent set will take only one vertex and the remaining K_5 cannot be partitioned into two forests. Similarly, $\overline{3K_3} = K_{3,3,3}$ generalizes to $K_{3,3,3,3}$, $\overline{3K_2 + K_1} = K_{2,2,2} \oplus K_1$ generalizes to $K_{2,2,2,2} \oplus K_1$, $\overline{2K_2} \oplus (K_1 + K_2) = K_{2,2} \oplus \overline{K_{1,2}}$ generalizes to $K_{2,2,2} \oplus \overline{K_{1,2}}$, and so on. Below, we present a complete set $\mathcal{F}_1$ of minimal cograph obstructions for partition into $p = 2$ forests and $q = 1$ independent set.

Theorem 53. *A cograph has a* $(2, 1)$*-partition if and only if it does not contain an induced subgraph from* $\mathcal{F}_1$*.*

The family $\mathcal{F}_1$ contains nine cographs, and while most of them can be interpreted as generalizations of members of $\mathcal{A}_2$, some appear to be definitely new. In particular, the last member, 9, of the family $\mathcal{F}_1$, does not seem to arise from any member of the family $\mathcal{A}_2$ in any obvious fashion. Thus the evidence suggests that a uniform description of minimal cograph obstructions for all $(2, q)$-partitions seems unlikely.

The reference [21] presents a linear time dynamic programming algorithm to decide whether an input cograph G admits a $(2, 1)$-partition (or any other (p, q)-partition). As an application of our result, we will present, in Algorithm 3, page 36, a certifying algorithm for the negative outcome (i.e., a non-partitionable cograph G) by finding an actual forbidden induced subgraph.

2.2.1 The list of minimal obstructions

For brevity, we call a partition of a graph G into p forests and q independent sets, a (p, q)-*partition* of G. Thus, in the remainder of this chapter, we describe all minimal cograph obstructions for $(2, 1)$-partition.

We introduce the family of cographs $\mathcal{F}_1$. The members of the family are:

1. K_6
2. $K_{3,3,3,3}$
3. $K_{2,2,2} \oplus \overline{K_{1,2}}$
4. $K_{2,2,2,2} \oplus K_1$
5. $2K_3 \oplus K_{2,2}$
6. $(K_{2,2} + K_3) \oplus K_{2,2}$
7. $2K_{2,2} \oplus K_{3,3}$
8. $2K_{2,2} \oplus 2K_{2,2}$
9. $(K_4 + K_{3,3,3}) \oplus \overline{K_2}$

Lemma 54. *Each graph in $\mathcal{F}_1$ is a minimal cograph obstruction to $(2,1)$-partition.*

Proof. It is clear from their descriptions that each graph in the $\mathcal{F}_1$ family is a cograph. We claim that each of these graphs is a minimal obstruction for $(2,1)$-partition.

Consider first K_6: it does not have a $(2,1)$-partition, because any forest in K_6 can have at most two vertices, and hence two forests can have at most four vertices. This leaves at least two vertices, but no two vertices in K_6 form an independent set. Moreover, when a vertex is removed we have K_5, which has an obvious $(2,1)$-partition where each forest is one edge and the independent set is a single vertex. Therefore K_6 is a minimal obstruction.

For $G = K_{3,3,3,3}$, we observe that any induced forest in G has at most four vertices, and this happens only when the forest is a tree. Thus two forests can cover at most eight vertices, and since $K_{3,3,3,3}$ has no independent set of size four, it does not have a $(2,1)$-partition. When a vertex is removed, we obtain $K_{2,3,3,3}$, where we can take one independent set consisting of a part with three vertices, and cover the vertices of the remaining two parts of size three by stars centered at the remaining two two vertices. Hence $K_{3,3,3,3}$ is also a minimal obstruction.

The proof for most of the remaining obstructions follows a similar approach, and we skip the details (which are included in the last section). We do include the proof for the last two obstructions on our list, which are more interesting.

Consider the graph $2K_{2,2} \oplus 2K_{2,2}$ from 8. Any independent set must be on one side of the join, and include at most four vertices. The remaining vertices contain an induced $2\left(\overline{2K_2}\right) \oplus \overline{K_3}$, which is one of the obstructions for $(2,0)$-partition from Figure 2.1. When a vertex is deleted, we obtain the graph $((K_{1,2} + K_{2,2}) \oplus 2K_{2,2})$, which has the following $(2,1)$-partition: one independent set of four vertices on the bigger side of the join, one forest consisting of $2K_{1,2}$ on the smaller side of the join, and one forest which is a star on five vertices. Thus $2K_{2,2} \oplus K_{2,2}$ is indeed a minimal cograph obstruction for $(2,1)$-partition.

For $G = K_{2,2,2} \oplus \overline{K_{1,2}}$, we note that any subgraph of G on at least four vertices contains an induced cycle. Hence, one forest in the partition can cover at most three vertices and two forest can cover at most six vertices, and since G has no independent set of size three, $K_{2,2,2} \oplus \overline{K_{1,2}}$ is an obstruction. For $H = K_{1,2,2} \oplus \overline{K_{1,2}}$, one of the two forests will be $\overline{K_{1,2}}$, and removing another forest on three vertices, the remainder is an independent set on two vertices, yielding a required $(2,1)$-partition. For $H = K_{2,2,2} \oplus K_2$, take one of the vertices of K_2 with one of the parts in $K_{2,2,2}$ to obtain one forest. We obtain the other forest in similar way and the remainder is just an independent set of size two. Hence, $K_{2,2,2} \oplus \overline{K_{1,2}}$ is a minimal obstruction.

To prove that $G = K_{2,2,2,2} \oplus K_1$, is an obstruction, note that any forest must be a tree and hence can have at most three vertices. Two forests can cover at most six vertices, and the remaining three vertices will contain an edge, and hence not be independent. To prove that G is indeed minimal, note that both $H_1 = K_{2,2,2,2}$ and $H_2 = K_{1,2,2,2} \oplus K_1$ have a

$(2, 1)$-partition in which each forest is a tree on three vertices and the independent set has two vertices.

For $G = 2K_3 \oplus K_{2,2}$, one of the two forests can cover at most four vertices and the other forest can cover at most three vertices. Hence, two forests can cover at most seven vertices and there is no independent set of size at least three in G. Hence, G does not have a $(2, 1)$-partition. Now we will show that $H_1 = (K_2 + K_3) + \oplus K_{2,2}$ and $2K_3 \oplus K_{1,2}$ have a $(2, 1)$-partition. For H_1, the partition consists of one forest that is $2K_2$ and has four vertices. The other forest is a tree on three vertices and the remainder is just an independent set on two vertices. In H_2, the first forest consists of the middle vertex in $K_{1,2}$ along with one vertex in the each of the K_3. The other forest has four vertices consisting of $2K_2$ each. The remainder is an independent set on two vertices. Thus $G = 2K_3 \oplus K_{2,2}$ is indeed minimal.

Similarly, for $G = (K_{2,2} + K_3) \oplus K_{2,2}$, one of the two forests in G can have at most five vertices, and then the other forest can have at most three vertices. The remainder will have at least 3 vertices. Since G does not have an independent set of size three, G does not have a $(2, 1)$-partition. We will prove that all the graphs obtained from deleting one vertex from G have a $(2, 1)$-partition. That is, $H_1 = (K_{1,2} + K_3) \oplus K_{2,2}$, $H_2 = (K_{2,2} + K_2) \oplus K_{2,2}$ and $H_3 = (K_{2,2} + K_3) \oplus K_{1,2}$, each have a $(2, 1)$-partition. For the graph H_1, the partition has one forest on five vertices consisting of $K_{1,2}$ and $K_{2,}$, and the other forest is just a $K_{1,2}$, leaving an independent set on two vertices.

For $G = 2K_{2,2} \oplus K_{3,3}$, two forests can cover at most ten vertices . Either there is only one forest on six vertices and the other forest then can have at most four vertices, or one can obtain two forest on five vertices each. There is no independent set on four vertices., so G does not admit a $(2, 1)$-partition. To see that G is indeed minimal, note that both $H_1 = (K_{2,2} + K_{1,2}) \oplus K_{3,3}$ and $H_2 = 2K_{2,2} \oplus K_{2,3}$ have a $(2, 1)$-partition. For H_1 one such partition has one forest consisting of two copies of $K_{1,2}$ and other forest is a $K_{1,3}$, leaving an independent set of three vertices. For H_2, a partition can be obtained with two forests which are stars on five vertices each, and the remainder is just an independent set on three vertices.

Finally, we prove that the graph $(K_4 + K_{3,3,3}) \oplus \overline{K_2}$ is a minimal cograph obstruction for $(2, 1)$-partition. We consider what an independent set S must contain in order for none of the minimal cograph obstructions for $(2, 0)$-partition (from Figure 1) to remain after S is removed. Note that our graph contains $K_{2,3,3,3}$, while in Figure 1 there is both a $K_{3,3,3}$ and a $K_{1,2,2,2}$. Moreover, when S is removed there must not remain a copy of K_5. To satisfy just these restrictions, S must contain one vertex of the K_4, and three vertices of one entire part of the $K_{3,3,3}$. Since this is a maximal independent set, S must be this set; but then its removal results in a graph containing an induced $\left(\overline{2K_2} + K_3\right) \oplus \overline{K_2}$ (the last graph in Figure 1). It remains to partition the graphs resulting from deleting a vertex from $(K_4 + K_{3,3,3}) \oplus \overline{K_2}$. If a vertex in the K_4 is deleted, then we obtain a $(2, 1)$-partition by taking the independent set S as above, and two stars centered at the two vertices of the

$\overline{K_2}$, each involving one 3-vertex part of the $K_{3,3,3}$ and one vertex of the K_4. If a vertex of the $\overline{K_2}$ is deleted, we can take again the independent set S, one forest consisting of an edge from the K_4 and one part of the $K_{3,3,3}$, and one star centered at the other vertex of the $\overline{K_2}$. If a vertex v in the $K_{3,3,3}$ is deleted, we can take for the independent set the vertices in the $\overline{K_2}$, and partition the remaining vertices into two forests each consisting of one edge of the K_4 and one star on four vertices. □

2.2.2 The completeness of the list

We now provide a proof for the Theorem 53, i.e., the list of minimal cograph obstructions for $(2, 1)$-partition given in Lemma 67 is complete.

Proof. Let G be a cograph. It is easy to see that a disconnected cograph G admits a $(2, 1)$-partition if and only if each connected component of G admits a $(2, 1)$-partition. Thus we may assume G is a connected cograph which does not contain an induced subgraph from $\mathcal{F}_1$, and proceed to prove it has a $(2, 1)$-partition.

For brevity, we shall say that a graph is F-free if it does not contain F as an induced subgraph, and $\mathcal{F}_1$-free, if it doesn't contain any member of the family $\mathcal{F}_1$ as an induced subgraph.

Since G is connected, there exist cographs G_1 and G_2 such that $G = G_1 \oplus G_2$. If G_1 and G_2 are forests, then G trivially has a $(2, 1)$-partition. So, at least one of G_1, G_2 must contain an induced cycle. Without loss of generality, assume that at least G_1 has an induced cycle; since G_1 is a cograph, the only cycles possible are C_3 or C_4.

1. **Assume G_1 is C_3-free.** In this case G_1 has an induced C_4; moreover, G_1 is a bipartite graph. We will take a concrete bipartition and refer to (X, Y) as the parts. If G_2 is a forest, then we have a trivial $(2, 1)$-partition with two independent sets and a forest. Thus we may assume that G_2 also has a cycle. We have the following two subcases.

(a) **Both G_1 and G_2 are C_3-free**. This implies that both cographs G_1 and G_2 are bipartite, and each has an induced C_4. Both G_1 and G_2 cannot have more than one connected component with C_4 because G is $2K_{2,2} \oplus 2K_{2,2}$-free. Hence without loss of generality we may assume that G_2 has exactly one component, say A, with a C_4, and the other components are trees. Note that A must be a complete bipartite graph since G_2 has no induced P_4. The graph G_1 must also contain at least one connected component, say B, which is a complete bipartite graph. If G_1 has other components with an induced C_4, then one of the parts of A in G_2 has exactly two vertices, because G is $2K_{2,2} \oplus K_{3,3}$-free. If the other connected components of G_1 are trees, then one of the subgraphs A or B has a bipartition with one of the parts having exactly two vertices, since G is $K_{3,3,3,3}$-free. In either case, we can obtain a $(2, 1)$-partition of G as follows. Suppose the connected component A of the graph G_2 has a bipartition (X, Y), where X has exactly two vertices. The first forest is obtained by taking one vertex from X, the entire other part Y, and the remaining tree components of G_2. Since graph G_1 is also bipartite, another forest can be obtained by taking one of the parts of G_1 and the remaining vertex in X. The remaining vertices form an independent set in G_1.

(b) **G_1 is C_3-free but G_2 contains a C_3.** Since G_1 contains a $K_{2,2}$ and since G is $2K_3 \oplus K_{2,2}$-free and $(K_{2,2} + K_3) \oplus K_{2,2}$-free, there is exactly one component of G_2 with a C_3, and the other components of G_2 are forests. Let the set (v_1, v_2, v_3) induce a C_3 in G_2, and let B be the component of G_2 containing it. Since B is a connected cograph, we have $B = B_1 \oplus B_2$ for cographs B_1, B_2. The component B cannot contain an induced K_4 and hence none of the graphs B_1 and B_2 have a C_3. So, we assume without loss of generality that $v_1, v_2 \in V(B_1)$, and $v_3 \in V(B_2)$; moreover, B_2 must be an independent set since G is K_6-free. If B_2 has at least two elements, then B_1 must be a K_2, since G is $K_{2,2,2} \oplus \overline{K_{1,2}}$-free and $K_{2,2,2,2} \oplus K_1$-free. Hence either B_1 is a K_2 or B_2 is a K_1. We construct a $(2, 1)$-partition in both the cases.

When $B_1 = K_2$, then taking one of the parts of the bipartite graph G_1 along with one vertex in B_1 we obtain one first forest in our partition. To construct the second forest we include B_2 along with the remaining vertex in B_1 and the remaining tree components of G_2. The remainder in G_1 is the independent set in the partition.

When B_2 consists of a single vertex, then taking this vertex with one of the parts of the bipartite graph G_1 yields the first forest in the partition. The remaining parts of G_2 form a forest which becomes the second forest in the partition. The remaining part of G_1 is our independent set in the $(2, 1)$-partition.

This concludes the first case.

2. **Assume that G_1 contains C_3.** Without loss of generality we can assume that G_2 is a forest as otherwise we have a K_6, or a situation symmetric to the case 1(b). We consider several possible cases, noting that in all the cases where G_2 has at least one edge, G_1 does not contain K_4, since G is K_6-free.

(a) **Suppose first that G_2 has at least three vertices and at least one edge.** Consider a copy of C_3 on v_1, v_2, v_3 in G_1, and the component B of G_1 containing it. Since B is a connected cograph, we have $B = B_1 \oplus B_2$ for cographs B_1 , B_2. Since B does not contain a K_4, neither B_1 nor B_2 can contain a C_3. So, we assume without loss of generality that $v_1, v_2 \in V(B_1)$, and $v_3 \in V(B_2)$; moreover, B_2 is an independent set. If B_1 has an induced C_4, then B_2 will be just a single vertex because G is $K_{2,2,2} \oplus \overline{K_{1,2}}$-free and $K_{2,2,2,2} \oplus K_1$-free. (Note that G_2 contains a copy of $\overline{K_{1,2}}$ or $K_{1,2}$.) In conclusion, each component $B = B_1 \oplus B_2$ of G_1 which contains a C_3 either has a single vertex in B_2 and a bipartite B_1, or an independent set B_2 and a forest B_1. Each component of G_1 without a C_3 is bipartite.

We find a $(2, 1)$-partition of the graph G as follows. One forest will be formed by the vertices in G_2. We partition G_1 into a forest and an independent set; it suffices to partition each component B of G_1 separately. A component $B = B_1 \oplus B_2$ with C_3 which has a single vertex v in B_2 yields a star centered at v and using one part of the bipartition of B_1, with the other part of the bipartition yielding an independent set. In a component $B = B_1 \oplus B_2$ with C_3 where B_2 is an independent set and B_1 is a forest, we trivially have a desired partition. Finally, each remaining component B is bipartite and one part can be taken as a forest and the other part as an independent set.

(b) **Assume G_2 has exactly two vertices, which are adjacent.** Since G_1 does not contain K_4, it is is three colourable. One of the colour classes along with one vertex of G_2 forms one forest of the partition. Another colour class with the other vertex of G_2 yields another forest. The remainder is a single colour class which forms the independent set of the partition.

(c) **Assume G_2 has exactly two vertices, which are not adjacent.** If G_1 does not contain an induced K_4, we obtain a partition of G as in case 2(b); so we assume that G_1 has a K_4. Note that we may take G_2 for the independent set of a $(2, 1)$-partition, and it remains to find a partition of G_1 into two forests (a $(2, 0)$-partition). Clearly, it suffices to find such a partition for each component B of G_1 separately.

Note that while at least one component of G_1 has a K_4, there could be other components B of G_1 without a K_4. Such components B must have a $(2, 0)$-partition because otherwise G_1 contains a minimal cograph obstruction for $(2, 0)$-partition from the family $\mathcal{A}_2$, and adding the independent set G_2 would yield a member of $\mathcal{F}_1$. (This can be easily seen by comparing the two families.)

Now we consider components $B = B_1 \oplus B_2$ of G_1 which do contain a K_4.

Suppose first that both B_1, B_2 are bipartite. Note that both B_1 and B_2 cannot contain an induced a C_4 since G is $K_{2,2,2,2} \oplus K_1$-free. If both B_1 and B_2 are forests, then we have a trivial partition of B into two forests. Hence, say B_1, has a C_4 and B_2 is a forest. In fact, B_2 must be just an edge, say uv, because G is $K_{2,2,2,2} \oplus K_1$-free and $K_{2,2,2} \oplus \overline{K_{1,2}}$-free. In this case a $(2, 0)$-partition of B is formed by taking one star centered at u with one part of the bipartition of B_1, and one star centered at v with the other part of B_1.

Thus we may assume that one of B_1, B_2, say B_1, contains a C_3. Since G is K_6-free, B is K_5-free, and so B_2 must an independent set. Now we further consider each component $D = D_1 \oplus D_2$ of B_1. At least one such component D' must contain a C_3, but there could also be bipartite components D; all must be K_4-free.

If B_2 has at least two vertices, then exactly one component, namely D', of B_1 has a cycle (specifically a C_3). Bipartite components D cannot have a cycle (i.e., a C_4), because G is $(K_3 + K_{2,2}) \oplus K_{2,2}$-free. Moreover no other component $D \neq D'$ can have a C_3, because G is $2K_3 \oplus K_{2,2}$-free. Hence if B_2 has at least two vertices then all the components D of B_1, other than D', are forests.

Suppose that v_1, v_2, v_3 form a C_3 in D. Since $D = D_1 \oplus D_2$ is K_4-free, neither of the graphs D_1, D_2 has a C_3. So we may assume $v_1, v_2 \in V(D_1)$ and $v_3 \in V(D_2)$; moreover we may assume D_1 is a bipartite graph and D_2 is an independent set.

If the bipartite graph D_1 contains a C_4, then both D_2 and B_2 must consist of a single vertex because G is $K_{2,2,2,2} \oplus K_1$-free.

If D_1 is a forest with more than the two vertices v_1, v_2, then it contains an induced $K_{1,2}$ or $\overline{K_{1,2}}$. Therefore, at least one of D_2, B_2 must be a single vertex, since G is $K_{2,2,2,2} \oplus K_1$-free and $K_{2,2,2} \oplus \overline{K_{1,2}}$-free.

Otherwise D_1 is just the edge v_1v_2.

Finally, if there is no C_3 in D, i.e., D is bipartite, then D_1 is an independent set.

We now describe a $(2,0)$-partition of $B = B_1 \cup B_2$. Recall that B_2 is an independent set, and B_1 consists of components $D = D_1 \oplus D_2$ where each D_2 is an independent set and each D_1 is bipartite, with the following four possibilities: (i) D_1 contains a C_4, in which case D_2, as well as B_2, has a single vertex; (ii) D_1 is a forest of more than two vertices, in which case D_2 or B_2 has a single vertex; (iii) D_1 is an edge v_1v_2; or (iv) D_1 is an independent set. Moreover, in cases (ii - iv), if B_2 has more than one vertex, then all but one component D of B_1 are forests.

We first describe a $(2,0)$-partition of $B = B_1 \cup B_2$ when B_2 has at least two vertices. In this case, there is one component $D' = D'_1 \oplus D'_2$ of B_1 with D'_1 a forest with one or more vertices (cases (ii, iii)), and all other components D of B are forests themselves. We obtain a $(2,0)$-partition of G_1 as follows. If D'_1 is just an edge, say xy, the first forest consists of a star centred at the vertex x covering the independent set D'_2, along with the rest of the forest components of B'_1. The second forest is a star centred at the remaining vertex v covering the independent set B_2. If D'_1 has at least two vertices, then D'_2 is a single vertex u, and we can take D'_1 together with all other components D as one forest; the other forest will be the star centered at u and covering B_2.

Now consider a component $B = B_1 \cup B_2$ of G_1 when B_2 has a single vertex, say v. We put together one forest for a $(2,0)$-partition of B from the following forests in the various components $D = D_1 \oplus D_2$ of B. From components D of type (i) we take the star centered at the single vertex of D_2 and covering one part of the bipartition of D_1; from components D of type (ii-iv) we take the forests D_1. The other forest for a $(2,0)$-partition of B will be formed by a star centered at v and covering all the remaining vertices. (These are the other parts of all D_1 for components of type (i), as well as all D_2 for components of type (ii-iv); note that this is an independent set of vertices.)

(d) **Finally, we assume that G_2 is just a single vertex, say v.** The proof here is similar to the case 2(c), except that in the case (i), when D_1 contains an induced C_4, we can only claim that B_2 or D_2 is a single vertex, and in the case (ii), when D_1 is a forest with more than two vertices, we cannot claim anything about the size of B_2 or D_2.

Nevertheless, there is a $(2,1)$-partition of the entire G. (Since G_2 is a single vertex v, we may use v to form a star for the forests of the partition, and we no longer use G_2 as the independent set.) Before describing the partition, recall that G consists of a vertex v adjacent to all other vertices, and $G \setminus v$ has components $B = B_1 \oplus B_2$ of two kinds, either

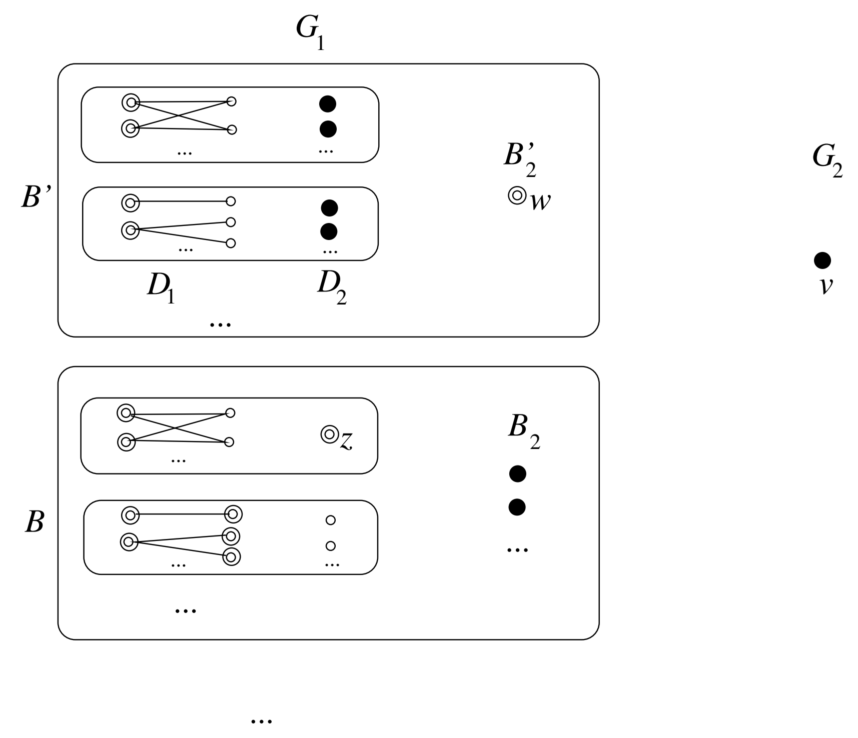


Figure 2.2: An illustration of the case 2(d): one forest is indicated by large filled circles, the other forest by double circles, the remainder is independent

B_2 is a single vertex, or B_2 is an independent set with at least two vertices. For components $B' = B'_1 \oplus B'_2$ of the first kind (where B'_2 is a single vertex w), we only note that B'_1 consists of bipartite components D. For the components B of the second kind (where B_2 is a larger independent set), we distinguish components $D' = D'_1 \oplus D'_2$ in which D'_2 consists of a single vertex z, and other components $D = D_1 \oplus D_2$ where D_2 is a larger independent set and D_1 is a forest. We now describe the first forest of a $(2,1)$-partition of G. It is a star centered at v and covering the sets D_2 of all components D of B'_1 for the components B' of the first kind (where B'_2 is a single vertex), as well as the sets B_2 of all components B of the second kind. The second forest of the partition contains, for each component B' of the first kind (where B'_2 is a single vertex w), a star centered at w and covering all first parts of the bipartitions of all D_1 of the components D of B'_1. It also contains, for each component B of the second kind, and each component D' in which D'_2 consists of a single vertex z, a star centered at z and covering the first part of the bipartition of D'_1, and containing D_1 for each component D in which D_1 is a forest. The remaining vertices are easily seen to form an independent set which we take for the desired $(2,1)$-partition of G. This verifies the completeness of the lists and hence proves Theorem 53. □

2.2.3 Algorithms for (p, q)-partition in cographs

Before we discuss the certifying algorithms for the (p, q)-partition problems that we have discussed so far, we briefly define the (p, q, r)-partition problem. A detailed discussion of the (p, q, r)-partition problem is done in the next chapter. A (p, q, r)-partition is a generalization of the (p, q)-partition where r vertices are allowed to be deleted such that the resulting graph admits a (p, q)-partition. In [21], a polynomial time algorithm was given, which for a given graph G and an integer w, generates all the triples (p, q, r) such that G admits a (p, q, r)-partition, and $p + q + r \leq w$. For a triple $T = (p, q, r)$, we define the weight of a triple $w = (p + q + r)$.

Theorem 55. *([21]) Given a cograph $G = (V, E)$ and $|V| \leq n$, the list of all the triples T with weight$(T) \leq w$ such that G admits a T-partition can be obtained in time $O(nw^7)$.*

However, Theorem 55 does not identify a minimal obstruction if G does not admit a required partition. Of course, if one knows all the minimal obstructions and there are only finitely many of them, then they can be found in polynomial time. For example, consider the case of $(1, q)$-partition, the minimal obstructions are K_{q+3} and $\overline{(q + 2)K_2}$, and by examining all n^{2q} subsets, we can determine if either is present. However, we present a more efficient algorithm based on the cotree structure of the cographs.

We convert the proof of Theorem 52 into a certifying algorithm as follows. For a $(1, q)$-partition, the proof starts with the assumption that graph G is $\{K_{q+3}, \overline{(q + 2)K_2}\}$-free. Hence, in the certifying algorithm, we first test if G contains an induced K_{q+3}. If yes, we have identified the obstruction, and no such partition is possible. Otherwise, G is $(q + 2)$-colourable. In the proof, we consider next if there exists a colour class with at most one vertex in a $(q + 2)$-colouring of G. Hence, in the next step of the algorithm we call a routine that returns with a $(q + 2)$-colouring of G and we check the number of vertices in each colour class. If we find a colour class with a single vertex, we return with a $(1, q)$-partition as dictated in the proof. Otherwise, the next step of the proof involves identifying a cograph G_j in the family of cographs $\{G_i\}_{i=1}^{l}$, where $G = \oplus_{i=1}^{l} G_i$ so that G_j is $\overline{(\chi(G_j))K_2}$-free. Correspondingly, in the certifying algorithm, we iterate on the children of the root node, determining the chromatic number of each child node and checking the conditions stated in the proof. If such a node exists, we find a minimum colouring of that node; we also recursively construct a partition of the remaining graph to construct the required $(1, q)$-partition. If no such node exists, we identify the minimum obstruction $\overline{(q + 2)K_2}$.

Algorithm 1 A certifying algorithm for $(1, q)$-partition of cograph H

```
procedure PARTITION$(1, q; H)$
    $forest_1 = \{\}$, $ind\_set_1 = \{\}$,..., $ind\_set_q = \{\}$
    For each connected component $G$ of $H$ proceed as follows:
    if $q == 0$ then
        if $|V(G)| == 1$ then
            Return the following partition of $G$:
            $forest_1 = forest_1 + V(G)$
        Test if $G$ contains $K_3$ or $K_{2,2}$
        if $G$ contains $K_3$ or $K_{2,2}$ then
            No $(1, 0)$-partition exists
        else
            Return the following partition:
            $forest_1 = forest_1 + V(G)$
    Test if $G$ contains $K_{q+3}$
    if $K_{q+3}$ found then
        No $(1, q)$-partition exists
    else
        $(X_1, X_2, ...X'_q)$ = colour $G$ with $(q + 2)$ colours
        if $|X_1| \leq 1$ or $|X_2| \leq 1$... or $|X'_q| \leq 1$ then
            $X_k$ = set containing at most one vertex
            Return the following partition of $G$:
            $forest_1 = (X_1 \oplus X_k)$, $ind\_set_1 = X_2$, $ind\_set_2 = X_3$, ....
        $|X_1| \geq 2$, $|X_2| \geq 2$, ... $|X_q| \geq 2$
        From cotree of $G$ identify cographs $G_1, G_2, .., G_\ell$
        such that $G = \oplus_{i=1}^{\ell} G_i$
        Find minimum colouring of each of $G_1, G_2, ..., G_\ell$
        $\chi(G_i)$ = chromatic number of $G_i$
        for $i = 1$ to $\ell$ do
            if $G_i$ does not contain $\overline{\chi(G_i)K_2}$ then
                $k = \chi(G_i)$
                $(forest_1, ind\_set'_1, ind\_set'_2, ...)$ = PARTITION$(1, k - 2; G_i)$
                $(X_1, X_2, ..., )$= colour graph $G \setminus V(G_i)$ using $(q + 2 - k)$ colours
                Return the following partition of $G$:
                $forest_1 = forest_1$, $ind\_set_1 = ind\_set'_1$, $ind\_set_2 = ind\_set'_2$, ..., $ind\_set_{k-1} = X_1$, ...
        $G_i$ contains $\overline{\chi(G_i)K_2}$ for all $i = 1, 2, .., l$
        Found $\overline{(q + 2)K_2}$, so conclude the algorithm, no $(1, q)$-partition exists
```

Before the certifying algorithm, we implement a pre-processing procedure that has the following two steps.

1. Compute the cotree of cograph H.
2. For each node in the cotree of H, compute the clique number of the graph induced at that node, and the largest s such that the graph induced at that node in the cotree contains $\overline{sK_2}$.

The cotree of a cograph can be constructed in time $O(m+n)$ [14]. The second step of the pre-processing can be implemented to run in time $O(n)$ as follows.

- Initialize leaves of cotree with $s = 0$ and $k = 1$.
- Perform the addition operation on join nodes, i.e., for a join node $s = \sum_1^l s_i$, where s_i is the largest $\overline{s_iK_2}$ at child i, and the join node has l children. Similarly, $k = \sum_1^l k_i$.
- Perform the max operation on disjoint union nodes, i.e., for a disjoint union node $s = max(s_1, s_2, ..., s_l)$, where s_i is the largest $\overline{s_iK_2}$ at child i, and the disjoint union node has l children. Similarly, $k = max(k_1, k_2, ..., k_l)$.

Theorem 56. *The pre-processing procedure runs in time $O(m+n)$.*

We now analyze the running time of the recursive Algorithm 1. Let $T(n,q)$ denote the maximum time for PARTITION$(1,q;H)$ over graphs H with n vertices. To implement line 3, we only have to check if the cotree node for G has maximum k greater than or equal to $q+3$, and hence, this can be done in constant time after the pre-processing step. For line 7, we compute a $(q+2)$-colouring, which must exist if there is no K_{q+3}. Using the algorithm of [10], such a colouring can be computed in time $O(n)$ since the cotree is given. Once a colouring with the minimum number of colours is obtained, we can check if each colour class has at least two vertices in time $O(n)$. Lines 9-11 can be executed in constant time. The subgraphs $G_1, ..., G_l$ in lines 13-14 can be obtained from the cotree in time $O(n)$. Let n_i denote the number of vertices in subgraph G_i. Lines 15 take $O(n)$ time since a minimum colouring of G_i can be found in $O(n_i)$ time, and $O(n_1) + O(n_2) + ... + O(n_l) = O(n)$. Line 16 also takes time $O(n)$ as the maximum k for the node G_i can be obtained from the pre-processing. Now, for the loop from lines 17-23, observe that although the number of iterations of the loop can be n, at most one recursive call is made in the entire execution of the loop. This is because a cograph that is $\{K_{k-1}, \overline{kK_2}\}$-free is guaranteed to have a $(1, k-2)$-partition by Theorem 52. The if condition in line 18 can be tested in constant time after the pre-processing. The colouring required in line 21 has already been computed in line 15. Line 24 of the algorithm takes constant time after the pre-processing. Hence, $T(n,q)$ satisfies the following equality.

$$T(n, q) = T(n_i, k-2) + O(n).$$

Here, n_i denotes the number of vertices in subgraph G_i, k is the chromatic number of graph G_i. For a connected cograph G on n vertices where $n \geq 2$, we know that $G = \oplus_{i=1}^{l} G_i$, where $l \geq 2$. Hence, $\chi(G) = \sum_{i=1}^{l} \chi(G_i)$ and therefore $\chi(G_i) < \chi(G)$. That is, $k < (q+2)$ and hence, $k - 2 < q - 1$. We now claim the following inequality.

$$T(n, q) \leq T(n, q-1) + O(n).$$

For the base case of $q = 0$, we need to check if the graph contains a K_3 or $K_{2,2}$ which can be checked in constant time from the s and k values computed in pre-processing. Otherwise, the graph is a forest and the vertex set of the graph is a $(1, 0)$-partition itself. Hence, the $(1, 0)$-partition can be computed in constant time. Hence, we conclude that $T(n) = O(qn)$, and the entire procedure takes $O(m + qn)$ time.

Theorem 57. *Algorithm 1 can be implemented in time $O(m + qn)$, or time $O(qn)$ if the cotree of H is given.*

For Algorithm 2, we perform the same pre-processing we did above. The following operations are the building blocks of Algorithm 2.

1. Testing if a cograph G has a cycle.
2. Finding the number of components of G that have a three cycle.
3. Finding the number of components of G that have a four cycle.
4. For a cograph G, identify subgraphs G_1, G_2 such that $G = G_1 \oplus G_2$.
5. Finding a bipartition (X, Y) for a bipartite graph G.
6. Testing if G contains a K_4, K_5 or K_6.
7. Testing if G contains a $K_{2,2,2,2}$ or a $K_{3,3,3,3}$.
8. Finding a three-colouring of a cograph G.
9. Finding number of vertices and edges in a component H.

All of the above operations can be done in time $O(n)$ once the cotree is known. A cograph G contains a three-cycle if and only if the largest clique size which we already computed in pre-processing is greater than two. Similarly, G contains a four-cycle if and only if the largest s computed in the pre-processing is greater than one. Finding the subgraphs G_1 and G_2 is trivial by reading the children of the node corresponding to G in the cotree. A bipartition of G is a two-colouring of G which can be found in $O(n)$ time. Similarly, the three-colouring of a cograph G in 8 can be found in $O(n)$ time. Similarly, steps 5, 6 and 7 can be computed in either in $O(n)$ time once the cotree is known, or have already been computed in the pre-processing stage. Hence, the cotree with the largest s and clique size for each node computed in the pre-processing settles most of the steps required. The algorithm applies these elementary operations in the if-else blocks. Hence, we conclude that the certifying algorithm takes $O(m + n)$ time.

Theorem 58. *Algorithm 2 can be implemented in time $O(m+n)$, or time $O(n)$ if the cotree of H is given.*

Algorithm 2 A certifying algorithm for $(2, 1)$-partition of cograph H

```
1: procedure Partition$(2, 1; H)$
2:     For each connected component $G$ of $H$ proceed as follows
3:     From cotree of $G$, identify cographs $G_1, G_2$ such that $G = G_1 \oplus G_2$.
4:     if both $G_1$ and $G_2$ are acyclic then
5:         Return the partition: $forest_1 = G_1$, $forest_2 = G_2$
6:     else if one of $G_1, G_2$, say $G_1$, contains a cycle then
7:         if both $G_1$ and $G_2$ are $C_3$-free then
8:             $G_1$ is a bipartite graph $(X, Y)$ that contains a $K_{2,2}$
9:             if $G_2$ is acyclic then
10:                Return the following partition:
11:                $forest_1 = G_1$, $forest_2 = G_2$
12:            else
13:                Ensure $G_2$ has a connected component that contains a $C_4$
14:                Find number of components in $G_1$ and $G_2$ that are not acyclic
15:                if both $G_1$ and $G_2$ contain at least two components that contain $C_4$ then
16:                    Found $2K_{2,2} \oplus 2K_{2,2}$, no $(2, 1)$-partition exists.
17:                else
18:                    $G_2$ contains exactly one component $A = (X', Y')$ that has a four cycle
19:                    if $|X'| = 2$ or $|Y'| = 2$ then
20:                        $X'$ = part with exactly two vertices
21:                        $\{u, v\} = V(X')$
22:                        Return the following $(2, 1)$partition:
23:                        $forest_1 = X' \oplus u$, $forest_2 = Y' \oplus v$, $ind\_set = Y$
24:                    else if $|X| = 2$ or $|Y| = 2$ then
25:                        $X$ = part with exactly two vertices
26:                        $\{u, v\} = V(X)$
27:                        Return the following $(2, 1)$partition:
28:                        $forest_1 = X' \oplus u, forest_2 = Y' \oplus v, ind\_set = Y$
29:                    else
30:                        Found $K_{3,3,3,3}$, no $(2, 1)$-partition exists.
31:        else if $G_1$ is $C_3$-free but $G_2$ contains a $C_3$ then
32:            $G_1$ is a bipartite graph that contains a $K_{2,2}$
33:            $G_1$ has bipartition $(X, Y)$
34:            $k_1$ = number of components of $G_1$ that have $C_4$
35:            $k_2$ = number of components of $G_2$ that have $C_4$
```

```
        if $k_1 \geq 1$ then
            Found $(K_{2,2} + K_3) \oplus K_{2,2}$, no $(2,1)$-partition exists
        else if $k_2 \geq 2$ then
            Found $2K_3 \oplus K_{2,2}$, no $(2,1)$-partition exists
        else
            $G_2$ has a component $B$ containing $C_3$, other components of $G_2$ are forests
            $B = B_1 \oplus B_2$
            if $num\_edges(B_1) \geq 1$ and $num\_edges(B_2) \geq 1$ then
                Found $K_6$, no $(2,1)$-partition exists
            else
                $B_1$ = component of $B$ with edges
                if $|B_2| \geq 2$ and $|B_1| \geq 3$ then
                    if $e_1 \geq 3$ then
                        Found $K_{2,2,2,2} \oplus K_1$, no $(2,1)$-partition exists
                    else
                        Found $K_{2,2,2} \oplus \overline{K_{1,2}}$, no $(2,1)$-partition exists.
                else if $|B_2| \geq 2$ and $|B_1| = 2$ then
                    $B_1 = K_2, V(B_1) = (u, v)$
                    Return the following $(2,1)$-partition:
                    $forest_1 = X \oplus u$
                    $forest_2 = (B_2 \oplus v)$ + acyclic components of $G_2$
                    $ind\_set = Y$
                else
                    $|B_2| = 1, V(B_2) = w$
                    Return the following $(2,1)$-partition:
                    $forest_1 = X \oplus w, forest_2$ = acyclic components of $G_2$
                    $ind\_set = Y$
    else if $G_1$ contains a $C_3$ then
        if $G_2$ contains a $C_3$ then
            Found $K_6$, no $(2,1)$-partition exists
        $e$ = no of edges in $G_2$
        if $|G_2| \geq 3$ and $e \geq 1$ then
            if $G_1$ has a $K_4$ then
                Found $K_6$, no $(2,1)$-partition exists
```

```
70:         for all components $B$ of $G_1$ containing a $C_3$ do
71:             if $num\_edges(B_1) \geq 1$ and $num\_edges(B_2) \geq 1$ then
72:                 Found $K_6$, no $(2,1)$-partition exists
73:             $B_2 =$ the component of $B$ which is edgeless
74:             if $B_1$ contains a $C_4$ then
75:                 if $|B_2| \geq 2$ then
76:                     Either $K_{2,2,2,2} \oplus K_1$ or $K_{2,2,2,2} \oplus \overline{K_{1,2}}$ found, no
77:                     $(2,1)$-partition exists
78:                 else
79:                     $|B_2| = 1$, $\{u\} = V(B_2)$
80:                     $B_1$ is complete-bipartite with bipartition $(X, Y)$
81:                     Partition $B$ as follows: $forest_1 = forest_1 + X \oplus u$
82:                     $ind\_set = ind\_set + Y$
83:             else
84:                 Partition $B$ as follows:
85:                 $forest_1 = forest_1 + B_1$
86:                 $ind\_set = ind\_set + B_2$
87:             Return the following $(2,1)$-partition:
88:             $forest_1$, $forest_2$ =acyclic components of $G_1, ind\_set$
89:
90:     else if $|G_2| = 2$ and $edges(G_2) = 1$ then
91:         if $G_1$ has a $K_4$ then
92:             Found $K_6$, no $(2,1)$-partition exists
93:         $(A, B, C) =$ Find_three_colouring$(G_1)$
94:         $\{u, v\} = V(G_2)$
95:         Return the following $(2,1)$-partition:
96:         $forest_1 = A \oplus u$, $forest_2 = B \oplus v$, $ind\_set = C$
97:     else if $|G_2| = 2$ and $num\_edges(G_2) = 0$ then
98:         if $G_1$ does not contain $K_4$ then
99:             go to 70
100:        else
101:            for all components $B$ of $G_1$ containing a $K_4$ do
102:                $B = B_1 \oplus B_2$
103:                if both $B_1$ and $B_2$ are bipartite then
104:                    if both $B_1$ and $B_2$ contain $C_4$ then
105:                        Found $K_{2,2,2,2} \oplus K_1$, no $(2,1)$-partition exists
```

```
            else if both $B_1$ and $B_2$ are acyclic then
                Partition $B$ as follows:
                $forest_1 = forest_1 + B_1$, $forest_2 = forest_2 + B_2$
            else
                $B_1$ = component of $B$ that has $C_4$
                $B_2$ = component of $B$ that is a forest
                if $edges(B_2) \geq 2$ then
                    Either $K_{2,2,2,2}$ $oplus K_1$ or $K_{2,2,2} \oplus \overline{K_{1,2}}$ found, no
                    $(2,1)$-partition exists
                $\{u, v\} = V(B_2)$
                $B_1$ has bipartition $(X, Y)$
                Partition $B$ as follows:
                $forest_1 = forest_1 + X \oplus u$
                $forest_2 = forest_2 + Y \oplus v$
        else
            One of $B_1, B_2$ contains a $C_3$
            $B_1$ = the component containing $C_3$
            if $edges(B_2) \geq 1$ then
                Found $K_6$, no $(2,1)$-partition occurs
            $B_2$ is edgeless
            if $|B_2| \geq 2$  then
                if no of components containing a cycle $\geq 2$ then
                    Either $(K_{2,2} + K_3) \oplus K_{2,2}$ or $2K_3 \oplus K_{2,2}$ occurs,
                    No $(2,1)$-partition
                Exactly one component of $B_1$ contains a cycle
                $D$ = component of $B_1$ containing $C_3$
                Other components,$D'$ of $B_1$ are forests
                $D = D_1 \oplus D_2$
                if $edges(D_1) \geq 0)$ and $edges(D_2) \geq 0$ then
                    Found $K_6$, no $(2,1)$-partition
                $D_2$ = the edgeless component of $D$
                if $D_1$ contains a cycle then
                    Either $K_{2,2,2,2} \oplus K_1$ or $K_6$, no $(2,1)$-partition
```

```
139:                     $D_1$ is a forest
140:                     if $|D_1| \geq 3$ then
141:                         if $|D_2| \geq 2$ then
142:                             Either $K_{2,2,2,2} \oplus K_1$ or $K_{2,2,2} \oplus \overline{K_{1,2}}$ occurs,
143:                             no $(2,1)$-partition
144:                         $|D_2| = 1$, $\{v\} = D_2$
145:                         Partition $B$ as follows:
146:                         $forest_1 = forest_1 + D_1 +$ acyclic components of $B_1$
147:                         $forest_2 = forest_2 + (B_2 \oplus v)$
148:                     else
149:                         $\{x, y\} = V(D_1)$
150:                         Partition $B$ as follows:
151:                         $forest_1 = forest_1 + (D_2 \oplus x) +$ acyclic components of $B_1$
152:                         $forest_2 = forest_2 + (B_2 \oplus y)$
153:                 else
154:                     $|B_2| = 1$, $\{v\} = V(B_2)$
155:                     if $D_1$ is not bipartite then
156:                         Found $K_6$, no $(2,1)$-partition
157:                     if $D_1$ is a forest then
158:                         go to 139
159:                     $D_1$ has a four cycle
160:                     $D_1$ has bipartition $(X, Y)$
161:                     if $|D_2| \geq 2$ then
162:                         Found $K_{2,2,2,2} \oplus K_1$, no $(2,1)$-partition
163:                     $\{w\} = V(D_2)$
164:                     Partition $B$ as follows:
165:                     $forest_1 = forest_1 + (X \oplus v)$
166:                     $forest_2 = fores_2 + (Y \oplus w)$
167:     else
168:         $G_1$ contains $C_3$ and $|G_2| = 1$
169:         $\{v\} = V(G_2)$
170:         if $G_1$ does not contain $K_4$ then
171:             go to 70
```

```
172:         else
173:             for all components $B$ of $G_1$ containing $K_4$ do
174:                 $B = B_1 \oplus B_2$
175:                 if both $B_1$ and $B_2$ are bipartite then
176:                     go to 103
177:                 else
178:                     One of $B_1, B_2$ contains a cycle
179:                     $B_1 =$ the component containing $C_3$
180:                     if $edges(B_2 \geq 1)$ then
181:                         Found $K_6$, no $(2,1)$-partition
182:                     $B_2$ is edgeless
183:                     for all components $D$ of $B_1$ containing $C_3$ do
184:                         $D =$ components of $B_1$ containing a three-cycle
185:                         $D = D_1 \oplus D_2$
186:                         if $edges(D_1) \geq 0)$ and $edges(D_2) \geq 0$ then
187:                             Found $K_6$, no $(2,1)$-partition
188:                         $D_2 =$ the edgeless component of $D$
189:                         if $D_1$ is not bipartite then
190:                             $K_6$ found, no $(2,1)$-partition found.
191:                         $D_1$ is bipartite with bipartition $(X, Y)$
192:                     if $|B_2| \geq 2$ then
193:                         if $D_1$ contains four cycle then
194:                             if $|D_2| \geq 2$ then
195:                                 Found $K_{2,2,2,2} \oplus K_1$, no $(2,1)$-partition
196:                             $\{z\} = V(D_2)$
197:                             Partition $B$ as follows:
198:                             $forest_1 = forest_1 + (B_2 \oplus v)$
199:                             $forest_2 = forest_2 + (X \oplus z)$
200:                             $ind\_set = ind\_set + Y$
201:                         else
202:                             $D_1$ is a forest
203:                             Partition $B$ as follows:
204:                             $forest_1 = forest_1 + (B_2 \oplus v)$
205:                             $forest_2 = forest_2 + (D_1)$
206:                             $ind\_set = ind\_set + D_2$
```

```
207:        **else**
208:            $|B_2| = 1$, $\{w\} = V(B_2)$
209:            Partition $B$ as follows:
210:            $forest_1 = forest_1 + (D_2 \oplus w)$
211:            $forest_2 = forest_2 + (X \oplus v)$
212:            $ind\_set = ind\_set + Y$
```

For another illustration, we give a certifying algorithm for the $(2,0)$-partition, which can be obtained similarly by converting the proof of Theorem 51 from [21]. For the $(2,0)$-partition algorithm, a similar analysis holds except that the algorithm is entirely composed of if-else constructs and there are no for loops as above. The algorithm also needs to test for induced P_3 for a cograph G. This can be done in constant time after the pre-processing procedure. Any graph with s at least two contains a P_3, and also a connected cograph where the largest $s = 1$ also contains a P_3. Also in line 66, the algorithms ask to compute a $(1,1)$-partition. Finding a $(1,1)$-partition takes time $O(n)$ since we have already constructed the cotree. Hence, Algorithm 3 runs in $O(m + n)$ time.

Theorem 59. *Algorithm 3 can be implemented in time $O(m + n)$, or time $O(n)$ once the cotree is known.*

Algorithm 3 A certifying algorithm for $(2,0)$-partition of cograph H

```
1: procedure PARTITION$(2,0;H)$
2:     For each connected component $G$ of $H$ proceed as follows:
3:     Construct cotree of $G$, find $G_1, G_2$ such that $G = G_1 \oplus G_2$.
4:     if both $G_1$ and $G_2$ are acyclic then
5:         Return the following partition:
6:         $forest_1 = G_1$, $forest_2 = G_2$
7:     else if  $G_1$ contains a cycle, switch $G_1, G_2$ if necessary then
8:         if both $G_1$ and $G_2$ are $C_3$-free then
9:             if $G_2$ contains a $P_3$ then
10:                Found $K_{2,2,2} \oplus K_1$, no $(2,0)$-partition exists
11:            if $|V(G_2)| \geq 3$ then
12:                if $G_2$ has an edge then
13:                    Found $K_{2,2} \oplus (K_1 + K_2)$, no $(2,0)$-partition exists
14:                else
15:                    $G_2$ is an independent set
16:                    if no of components of $G_1$ that contains a $C_4 \geq 2$  then
17:                        Found $2K_{2,2} \oplus \overline{K_3}$
18:                    Construct bipartition $(X,Y)$ for component of $G_1$ that contains a $K_{2,2}$
19:                    if $|X| \geq 3$ and $|Y| \geq 3$ then
20:                        Found $K_{3,3,3}$, no $(2,0)$-partition exists
21:                    $X =$ has at most two vertices, switch $X, Y$ if necessary
22:                    $\{u,v\} = X$
23:                    Return the following partition:
24:                    $forest_1 = G_2 \oplus u$, $forest_2 = (Y \oplus v)+$ acyclic components of $G_1$
25:            else
26:                $\{u,v\} = V(G_2)$
27:                Construct bipartition $(X,Y)$ for component of $G_1$
28:                Return the following partition:
29:                $forest_1 = X \oplus u$, $forest_2 = Y \oplus v$
30:        else if $G_1$ is $C_3$-free, $G_2$ contains a $C_3$ then
31:            Found $K_5$, no $(2,0)$-partition exists
```

```
        else if $G_1$ contains a $C_3$ then
            if $G_2$ has an edge then
                Found $K_5$, no $(2,1)$-partition occurs
            $G_2$ is an independent set
            if $|V(G_2)| \geq 2$ then
                if no of components of $G_1$ that have a cycle then
                    Found $2K_3 \oplus \overline{K_2}$ or $(K_3 + K_{2,2}) \oplus \overline{K_2}$
                    No $(2,0)$-partition exists
                $B$ = component of $G_1$ contains $K_3$
                Construct cotree for $B$, since $B$ is connected this cotree
                identifies $B_1$ and $B_2$, $B = B_1 \oplus B_2$
                if both $B_1$ and $B_2$ have edges then
                    Found $K_5$, no $(2,1)$-partition occurs
                $B_2$ = cograph which is an independent set
                Switch $B_1, B_2$ if necessary
                if $|B_2| \geq 2$ then
                    if $B_1$ has a $P_3$ then
                        Found $K_{2,2,2} \oplus K_1$, no $(2,0)$-partition exists
                    else if $B_1$ contains $(K_1 + K_2)$ then
                        Found $K_{2,2} \oplus (K_1 + K_2)$, no $(2,0)$-partition exists
                    else
                        $B_1$ has a single edge, $\{u,v\} = V(B_1)$
                        Return the following partition:
                        $forest_1 = G_2 \oplus u$, $forest_2 = B_2 \oplus v$ + acyclic components of $G_1$
                else
                    $B_2$ has a single vertex, $\{v\} = V(B_2)$
                    if $B_1$ contains a cycle then
                        Found $K_5$ or $K_{2,2,2} \oplus K_1$, no $(2,0)$-partition occurs
                    Return the following partition:
                    $forest_1 = G_2 \oplus v$, $forest_2 = B_1$ + acyclic components of $G_1$
```

```
62:     else
63:         $G_2$ has a single vertex, $\{v\} = V(G_2)$
64:         if $G_1$ contains a $K_4$ or $K_{2,2,2}$ then
65:             Found $K_5$ or $K_{2,2,2} \oplus K_1$, no $(2, 0)$-partition occurs
66:         $(forest_1, IS_1) =$ Find$(1, 1)$-partition of $G_1$
67:         Return the following partition:
68:         $forest_1 = forest_1,\ forest_2 = (IS_1 \oplus v)$
```

Chapter 3

(p,q,r)-partition

3.1 Introduction

In chapter 1, we observed that many "hard" problems like graph-colouring, vertex cover, independent set etc. can be solved in polynomial time for several restricted graph classes. The techniques and results used for a given graph class $\mathcal{C}$ can sometimes be applied to related graph classes if they differ only to a small degree. In this chapter, we extend our characterization results of (p, q)-partitionable cographs to the class of graphs "closely" related to (p, q)-partitionable cographs.

A *property* of a graph is a *set* of graphs. A graph G is said to be Π-graph if $G \in \Pi$. A *hereditary* property Π is a property such that if G is a Π-graph, then every induced subgraph of G is also a Π-graph. For a hereditary property, the "closeness" is defined in terms of the modification needed for a given graph G such that the modified graph is a Π-graph. The modifications allowed could be deletion or addition of vertices, deletion and addition of edges. This is called as the Π-graph modification problem.

Definition 40. *[8]* $\Pi(i, j, k)$*-graph modification problem for graph* $G = (V, E)$*:*
Find sets $V' \subseteq V$*,* $E \subseteq E$*,* $E^* \subseteq E^c$ *such that the graph* $G - V' - E' + E^*$ *is a* Π*-graph where* $E^c = E(\overline{G})$*,* $|V'| \leq i$*,* $|E'| \leq j$ *and* $|E^*| \leq k$*.*

The graph modification problem can also be seen as a generalization of the graph recognition problem. An example of a modification problem is the vertex deletion problem. For a given graph G, how many vertices should be removed from G such that the resulting graph is a Π-graph? This is called the *vertex deletion* problem or the *maximum induced subgraph* problem. The objective here is to minimize the number of vertices that should be removed. For any non-trivial hereditary graph property, the vertex deletion problem was shown to be NP-complete [65]. The edge deletion problem is defined analogously. The edge deletion problem is sometimes also called the maximum subgraph problem.

In this chapter, we study a vertex deletion version of the (p, q)-partition problem we discussed earlier, called the (p, q, r)-partition problem. For a given graph G, the minimum number r such that G admits a (p,q,r)-partition, is the smallest number of vertices whose deletion results in a graph that has a (p,q)-partition. The (p,q,r)-partition problem was introduced in [21].

Definition 41. *A (p,q,r)-partition of G is a partition (P,Q,R) of its vertex set such that the subgraph induced on P has vertex-arboricity p, the subgraph induced on Q is q-colourable, and $|R| \leq r$.*

We provide a minimal obstruction characterization of the (p,q,r)-partition problem for the $(1,1,1)$-case and the $(1,2,1)$-case.

Several vertex deletion problems can also be posed as (p,q,r)-partition problems. For example, consider the following vertex deletion problems.

Definition 42. *The* feedback vertex set *of a graph G is a set of vertices that meets all the cycles in the graph. If S is a feedback vertex set, then the graph $G - S$ is acyclic.*

The feedback vertex set problem is another example of the vertex deletion problem that we discussed above. The optimization problem here is to find such a set of minimum cardinality. In the weighted version, each vertex has a weight, and the objective is to minimize the weight of the sum of the vertices in the feedback vertex set.

Proposition 60. *The size of minimum feedback vertex set in a graph G is equal to the minimum r such that the graph G admits a $(1,0,r)$-partition.*

The minimum vertex feedback set is a classical combinatorial optimization problem, it was among Karp's original 21 NP-complete problems [50], and it has attracted much attention [49, 13, 66, 56].

Definition 43. *The maximum q-colourable subgraph for a given graph G is the largest induced subgraph of G that is q-colourable.*

The problem here is determining the size of the maximum q-colourable subgraph for a given graph G.

Proposition 61. *For a given graph $G = (V,E)$, let S be a maximum q-colourable subgraph in G, for a fixed q. Let r be the smallest integer such that the graph G admits a $(0,q,r)$-partition, then $|S| = |V| - r$.*

The maximum q-colourable subgraph problem is NP-complete on general graphs, for $q \geq 3$ [65]. The problem is also NP-complete on split graphs [66] but can be solved in polynomial time on interval graphs [58].

Definition 44. *The* vertex cover *of a graph G is a set of vertices X such that $G \setminus X$ is an independent set.*

Proposition 62. *The size of a minimum vertex cover in a graph is equal to the minimum r such that the graph G admits a $(0, 1, r)$-partition.*

Finding the minimum vertex cover is NP-complete for general graphs [50]. It is also NP-hard on planar graphs where the maximum degree is restricted to four and for cubic graphs [40].

3.2 The $(1, 1, 1)$-partition problem

For the $(1, 1, 1)$-partition problem, the objective is to obtain a (P, Q, R)-partition of the vertex set $V(G)$ such that P is a forest, Q is an independent set and $|R| \leq 1$. In other words, we are allowed to delete at most one vertex v such that $G - v$ admits a $(1, 1)$-partition. Below, we present complete sets $\mathcal{F}_2$ and $\mathcal{F}_3$ of minimal cograph obstruction for the $(1, 1, 1)$-partition.

Theorem 63. *A cograph admits a $(1, 1, 1)$-partition if and only if it is $\mathcal{F}_2$-free and $\mathcal{F}_3$-free.*

3.2.1 The list of minimal obstructions

We introduce the family $\mathcal{F}_2$ of cographs. The members of the family are:

1. K_5
2. $K_{3,3,3}$
3. $K_{2,2} \oplus \overline{K_{1,2}}$
4. $K_{2,2,2} \oplus K_1$
5. $2K_3 \oplus \overline{K_2}$
6. $(K_{2,2} + K_3) \oplus \overline{K_2}$
7. $2K_{2,2} \oplus \overline{K_3}$
8. $2K_2 \oplus 2K_2$

The family $\mathcal{F}_3$ consists of the following disconnected cographs.

1. $K_4 + K_4$
2. $K_4 + K_{2,2,2}$
3. $K_{2,2,2} + K_{2,2,2}$

Lemma 64. *Each graph in $\mathcal{F}_2$ is a minimal obstruction to $(1, 1, 1)$-partition.*

Proof. From their descriptions, one can see that graphs in $\mathcal{F}_2$ are obtained by repeated applications of the disjoint union or the join operation on complete graphs or complete bipartite graphs. Hence, each graph in $\mathcal{F}_2$ is a cograph.

For $G = K_5$, observe that any forest in K_5 can have at most two vertices. Any independent set can cover at most one vertex. This leaves two vertices in the remainder, where only one vertex can be assigned to the set R. Hence, K_5 does not admit a $(1,1,1)$-partition. To prove that K_5 is a minimal obstruction, it is sufficient to note that the graph K_4 admits a $(1,1,1)$-partition. Assign one vertex of K_4 to the set R; this leaves a K_3 that can be covered by a forest containing a single edge and an independent set that is just a single vertex.

For $G = K_{3,3,3}$, any forest can cover at most four vertices. With the independent set in any partition covering at most three vertices, the remainder would have at least two vertices. Since only one vertex can be assigned to the set R, $K_{3,3,3}$ is an obstruction. When a vertex is deleted, we obtain the graph $K_{2,3,3}$, which has the following $(1,1,1)$-partition. Remove one vertex of the part with two vertices, the forest is a star at the other vertex and covers four vertices. The remaining vertices form an independent set of size three.

For $G = K_{2,2} \oplus \overline{K_{1,2}}$, an independent set in G can have at most two vertices, and a forest can have at most three vertices since any four vertices in G have a cycle. The remainder contains at least two vertices, where the set R can have at most one vertex. Hence, G does not admit a $(1,1,1)$-partition. To prove that G is a minimal obstruction, we need to prove that the graphs $H_1 = K_{2,2,2}$, $H_2 = K_{1,2} \oplus \overline{K_{1,2}}$ and $H_3 = K_{2,2} \oplus K_2$ admit a $(1,1,1)$-partition. For $H_1 = K_{2,2,2}$, assign one vertex of a part to the set R. A star on three vertices covers the other vertex. An independent set of size two covers the remaining vertices. For $H_2 = K_{1,2} \oplus \overline{K_{1,2}}$, $K_{1,2}$ can be considered as the forest in a $(1,1,1)$-partition. Removing one vertex of the edge in $\overline{K_{1,2}}$ yields an independent set of size two. To obtain a partition in $H_3 = K_{2,2} \oplus K_2$, assign one vertex of K_2 to the set R. The forest is a star on three vertices, and the remainder is just an independent set of size two.

Assume that $G = K_{2,2,2} \oplus K_1$. Observe that any four vertices in G induce a cycle. Hence, a forest in G contains at most three vertices. Similarly, observe that any independent set has at most two vertices since any three vertices in G induce an edge. The remainder contains at least two vertices, where the set R can have only one. Hence, G is an obstruction to $(1,1,1)$-partition. We have already established that $K_{2,2,2}$ admits a $(1,1,1)$-partition. Hence, to establish that G is a minimal obstruction, it is sufficient to observe that the graph $K_{1,2,2} \oplus K_1$ admits the following $(1,1,1)$-partition. We take $K_{1,2}$ as the forest in the partition, the independent set has size two, and the set R contains the only vertex on one side of the join.

For $G = 2K_3 \oplus \overline{K_2}$, any forest can have at most four vertices. An independent set has size at most two in G, leaving at least two vertices in the remainder. Hence, G does not admit a $(1, 1, 1)$-partition. We will prove that all the graphs obtained from deleting one vertex from G have a $(1, 1, 1)$-partition. That is $H_1 = (K_2 + K_3) \oplus \overline{K_2}$ and $H_2 = 2K_3 \oplus K_1$, each have a $(1, 1, 1)$-partition. For H_1, a partition consists of a forest that is $2K_2$ and has four vertices. The independent set consists of two vertices on one side of the join, and the remaining vertex in H_1 can be removed. For H_2, the forest consists of the graph $2K_2$ and has four vertices. The independent set contains two vertices, and the remainder is just a vertex in the set R. Thus, $G = 2K_3 \oplus \overline{K_2}$ is a minimal obstruction.

For $G = 2K_{2,2} \oplus \overline{K_3}$, observe that any independent set must lie entirely on one side of the join. One side of the join contains two disjoint four cycles, which can not be partitioned into a forest and a vertex. Hence, an independent set must lie on this side of the join; such a set has at most four vertices. However, the remaining vertices contain an induced $K_{4,3}$, which can not be partitioned into a forest and a vertex. Therefore, G does not admit a $(1, 1, 1)$-partition. To see that G is a minimal obstruction, note that both $H_1 = (K_{2,2} + K_{1,2}) \oplus \overline{K_3}$ and $H_2 = 2K_{2,2} \oplus \overline{K_2}$ admit a $(1, 1, 1)$-partition. For H_1, one such partition has a forest consisting of two copies of $K_{1,2}$, an independent set covering three vertices and the remaining vertex in H_1 is assigned to the set R. For $H_2 = 2K_{2,2} \oplus \overline{K_2}$, a $(1, 1, 1)$-partition can be obtained with a forest that is a star on five vertices, the independent set has size four, and one vertex in the set R.

Assume that $G = (K_{2,2} + K_3) \oplus \overline{K_2}$. Any independent set must lie entirely on one side of the join. Observe that one side of the join contains a disjoint four cycle and a three cycle, which can not be partitioned into a forest and vertex. Hence, an independent set must lie on this side of the join; such a set has at most three vertices. However, the remaining vertices contain an induced $(K_2 + \overline{K_2}) \oplus \overline{K_2}$, which can not be partitioned into a forest and a vertex. Therefore, G does not admit a $(1, 1, 1)$-partition. To prove that G is a minimal obstruction, it is sufficient to provide a $(1, 1, 1)$-partition for the following graphs: $H_1 = (K_{1,2} + K_3) \oplus \overline{K_2}$, $H_2 = (K_{2,2} + K_2) \oplus \overline{K_2}$ and $H_3 = (K_{2,2} + K_3) \oplus K_1$. For $H_1 = (K_{1,2} + K_3) \oplus \overline{K_2}$, we construct a $(1, 1, 1)$-partition where we take $(K_{1,2} + K_2)$ as the forest, an independent set that contains two vertices, and one vertex in the set R. A similar partition can be constructed for H_2. For H_3, we remove the only vertex on one side of the join, we take $(K_{1,2} + K_2)$ as the forest in the partition, and the remaining two vertices form the independent set in the desired $(1, 1, 1)$-partition.

Finally, assume that $G = 2K_2 \oplus 2K_2$. Any independent set in G covers at most two vertices. A forest in G can have at most four vertices since any five vertices in G induce a cycle. The remainder contains at least two vertices, where the set R can have only one. Hence, G is an obstruction to a $(1, 1, 1)$-partition. The graph G is also a minimal obstruction since $2K_2 \oplus \overline{K_{1,2}}$ admits the following partition. We remove a vertex from $\overline{K_{1,2}}$ to obtain the independent set of the partition. The forest consists of two edges on one side of the join.

□

Lemma 65. *Let G be a disconnected cograph. If G has at least two connected components that do not admit a $(1, 1, 0)$-partition, then G does not admit a $(1, 1, 1)$-partition.* □

Lemma 66. *Let H be a disconnected cograph. If H is a minimal obstruction to $(1, 1, 1)$-partition, then H consists of exactly two connected components such that each component is a minimal obstruction to $(1, 1, 0)$-partition.*

Proof. Since H is a disconnected cograph, H consists of two or more connected components. If none of the components are an obstruction to $(1, 1, 0)$-partition, H admits a $(1, 1, 0)$-partition and hence admits a $(1, 1, 1)$-partition. Hence, at least one of the connected components does not admit a $(1, 1, 0)$-partition. If exactly one connected component does not admit a $(1, 1, 0)$-partition, then H is not minimal since a vertex v can be removed in H such that $H - v$ does not admit a $(1, 1, 1)$-partition.

Hence, H has at least two such components. Using Lemma 65, and the assumption that H is minimal, H is a graph with exactly two connected components, both of which do not admit a $(1, 1, 0)$-partition. If one of the connected components, say H_1, is not a minimal obstruction to $(1, 1, 0)$-partition, there exists an induced subgraph of H_1, which is a minimal obstruction to $(1, 1, 0)$-partition. This contradicts the assumption that H is minimal.

□

Lemma 67. *Each graph in $\mathcal{F}_3$ is a minimal cograph obstruction to $(1, 1, 1)$-partition.*

Proof. The $(1, 1, 0)$-partition has two minimal obstructions: K_4 and $K_{2,2,2}$. Lemma 65 establishes that the members of family $\mathcal{F}_3$ are indeed obstructions to $(1, 1, 1)$-partition. Lemma 66 guarantees the minimality of the obstructions. □

3.2.2 The completeness of the lists

First, we prove that the list of minimal cograph obstructions for $(1, 1, 1)$-partition given in Lemma 64 is complete for connected cographs.

Theorem 68. *A connected cograph admits a $(1, 1, 1)$-partition if and only if it is $\mathcal{F}_2$-free.*

Proof. Let G be a connected cograph; we assume that G is $\mathcal{F}_2$-free. We proceed to provide a $(1,1,1)$-partition for G. As G is a connected cograph, there exist cographs G_1 and G_2 such that $G = G_1 \oplus G_2$.

1. **Assume that both G_1 and G_2 are forests.** A forest in a cograph is a collection of stars. Hence, for cographs G_1 and G_2, each connected component is a star. Both G_1 and G_2 can not contain two stars as connected components since G does not contain $2K_2 \oplus 2K_2$ as an induced subgraph. Hence, without loss of generality, it can be assumed that one of the forests is a tree. Assume that G_2 is a tree, we obtain a $(1,1,1)$-partition as follows. Consider G_1 as the forest in the partition, removing the centre vertex of the star G_2, we obtain an independent set, as required.

2. **Suppose that exactly one of them, say G_2, is a forest.** Hence, G_1 contains an induced cycle. Since G_1 is a cograph, the only cycles possible are C_3 or C_4.
(2.1) **Assume that G_1 is C_3-free.** The graph G_1 must contain a C_4. For the bipartite graph G_1, we fix a bipartition and refer to (X, Y) as the parts. We will prove that the graph G_2, in this case, is either a single edge or an independent set. Since G is $(K_{2,2} \oplus \overline{K_{1,2}})$-free, G_2 must be either a connected cograph or an independent set. Furthermore, as G is $(K_{2,2,2} \oplus K_1)$-free, any tree G_2 can have at most two vertices, i.e., G_2 is an edge. If $|V(G_2)| \leq 2$, a $(1,1,1)$-partition can be obtained as follows. The forest is a star at a vertex in G_2, any vertex that remains in G_2 can be removed. The remainder forms an independent set in the $(1,1,1)$-partition.

If $|V(G_2)| \geq 3$, then G_2 must be an independent set, as we discussed above. Since G is $(2K_{2,2} \oplus \overline{K_3})$-free, exactly one connected component of G_1 contains an induced cycle. Let A be the connected component of G_1 containing C_4. Since A is a connected bipartite cograph, A is a complete bipartite graph. We fix a bipartition (A_1, A_2). The graph G is $K_{3,3,3}$-free; hence, one of the parts of A, say A_2, has at most two vertices. A $(1,1,1)$-partition can now be obtained by removing one vertex of A_2. The forest consists of a star at the remaining vertex of A_2 and the part A_1, and the other tree components of G_1, we take G_2 as the independent set of the partition.

(2.2) **Assume that G_1 contains a C_3.** Since G_1 contains a C_3 and G is K_5-free, G_2 must be an independent set. Consider a copy of C_3 on $\{v_1, v_2, v_3\}$ in G_1, and the component B of G_1 containing it. Since B is a connected cograph, we have $B = B_1 \oplus B_2$, for cographs B_1, B_2. Since B does not contain a K_4, neither B_1 nor B_2 can contain a C_3. So, we assume without loss of generality that $v_1, v_2 \in V(B_1)$, and $v_3 \in V(B_2)$; moreover, B_2 is an independent set. We consider the following sub-cases here.

(2.2.1) **Suppose that G_2 has at least two vertices**. The graph G_2 is an independent set, as we discussed above. Since G is $\{2K_3 \oplus \overline{K_2}, (K_{2,2} + K_3) \oplus \overline{K_2}\}$-free and G_2 contains $\overline{K_2}$, exactly one connected component of G_1, say B, contains an induced cycle.

If B_2 has at least two vertices, from the assumption that G is $(K_{2,2} \oplus \overline{K_{1,2}})$-free, we obtain that B_1 is connected. Furthermore, as G is assumed to be $(K_{2,2,2} \oplus K_1)$-free, B_1 must be a single edge, say uv. To obtain a $(1,1,1)$-partition, we remove one vertex in B_1, say u. The forest consists of a star at v and covering B_2 and the remaining tree components of G_1. We take G_2 as the independent set in the desired $(1,1,1)$-partition. If B_2 consists of a single vertex, say z, then B_1 must be a forest since G is $K_{2,2,2} \oplus K_1$-free. We remove the vertex z, B_1 along with the tree components of G_1 forms the forest in the partition. We take G_2 as the independent set in the desired $(1,1,1)$-partition.

(2.2.2) **Suppose that G_2 consists of a single vertex w.** From the assumption that G is $\{K_5, K_{2,2,2} \oplus K_1\}$-free, we infer that G_1 must be $\{K_4, K_{2,2,2}\}$-free, and admits a $(1,1)$-partition. We remove the vertex w to obtain a $(1,1,1)$-partition.

3. **Assume that both G_1 and G_2 contain an induced cycle.** We assert that this case is not possible. Since $(K_3 \oplus K_3)$ contains K_5 as an induced subgraph, which is forbidden. Similarly, $(K_{2,2} \oplus K_{2,2})$ contains $(K_{2,2,2} \oplus K_1)$, and $(K_{2,2} \oplus K_3)$, which also contains a K_5. □

Finally, we prove the Theorem 63 we presented earlier.

Proof. Assume first that G is a disconnected cograph. Since G is $\mathcal{F}_2$-free by assumption, each connected component of G admits a $(1,1,1)$-partition. As G is also $\mathcal{F}_3$-free, all the components of G except for one must admit a $(1,1,0)$-partition. Hence, the graph G admits a $(1,1,1)$-partition. □

3.3 The $(1,2,1)$-partition problem

The objective is to obtain a (P,Q,R)-partition of the vertex set $V(G)$ such that P induces a forest, Q can be partitioned into two independent sets, and $|R| \leq 1$. In other words, we are allowed to delete at most one vertex v such that $G - v$ admits a $(1,2)$-partition. Below, we present families $\mathcal{F}_4$ and $\mathcal{F}_5$ of complete minimal cograph obstructions to $(1,2,1)$-partition.

Theorem 69. *A cograph admits a $(1,2,1)$-partition if and only if it is $\mathcal{F}_4$-free and $\mathcal{F}_5$-free.*

3.3.1 The list of minimal obstructions

We introduce two families of cographs $\mathcal{F}_4$ and $\mathcal{F}_5$. The members of the family $\mathcal{F}_4$ are:

1. K_6
2. $K_{3,3,3,3}$
3. $K_{2,2,2} \oplus \overline{K_{1,2}}$
4. $K_{2,2,2,2} \oplus K_1$

5. $2K_3 \oplus K_{2,2}$

6. $(K_{2,2} + K_3) \oplus K_{2,2}$

7. $2K_{2,2} \oplus K_{3,3}$

8. $2K_{2,2} \oplus 2K_{2,2}$

9. $(K_4 + K_{2,2,2}) \oplus \overline{K_2}$

10. $2K_3 \oplus 2K_2$

11. $2K_4 \oplus \overline{K_2}$

12. $(2K_2 \oplus 2K_2) \oplus \overline{K_2}$

The family $\mathcal{F}_5$ consists of the following disconnected cographs.

1. $K_5 + K_5$

2. $K_5 + K_{2,2,2,2}$

3. $K_{2,2,2,2} + K_{2,2,2,2}$

Lemma 70. *Each graph in $\mathcal{F}_4$ is a minimal cograph obstruction to $(1, 2, 1)$-partition.*

Proof. It is clear from their descriptions that each graph in the family $\mathcal{F}_4$ is a cograph. We claim that each of these graphs is a minimal obstruction to $(1, 2, 1)$-partition.

Consider first $G = K_6$: it does not have a $(1, 2, 1)$-partition because any forest in K_6 can have at most two vertices. Observe that no two vertices in K_6 form an independent set. Hence, two independent sets can cover at most two vertices. The remainder has at least two vertices, only one of which can be assigned to the set R. Hence, G is an obstruction to $(1, 2, 1)$-partition. When a vertex is removed, we have K_5, a $(1, 2, 1)$-partition can be constructed as follows. The forest is one edge, and each independent set is a single vertex. The remainder is a vertex which is covered by the set R. Therefore, K_6 is a minimal obstruction.

For $G = K_{3,3,3,3}$, we observe that any forest can cover at most four vertices (any five vertices in G induce a cycle). An independent set in G has size at most three. Hence, two independent sets can cover at most six vertices. The remainder has at least two vertices, and hence G does not admit a $(1, 2, 1)$-partition. When a vertex is removed, we obtain $K_{2,3,3,3}$, where two independent sets have size three each. The forest is a star on four vertices, leaving one vertex in the set R. Therefore, $K_{3,3,3,3}$ is a minimal obstruction.

For $G = K_{2,2,2} \oplus \overline{K_{1,2}}$, we note that any induced subgraph on four or more vertices contains an induced cycle. Hence, a forest in any partition can cover at most three vertices. An independent set can have at most two vertices, and hence, two independent sets can cover at most four vertices. The remainder contains at least two vertices. Therefore, $K_{2,2,2} \oplus \overline{K_{1,2}}$ is an obstruction. The cograph G is also a minimal obstruction as the cographs $H_1 = K_{1,2,2} \oplus \overline{K_{1,2}}$ and $H_2 = K_{2,2,2} \oplus K_2$ admit a $(1, 2, 1)$-partition where the forest is a tree on three vertices, each independent set has size two, and the set R contains one vertex.

For $G = K_{2,2,2,2} \oplus K_1$, observe that any induced subgraph on four or more vertices contains an induced cycle. Hence, any maximal forest in G has size three. Any three vertices in G induce an edge, and hence an independent set can have at most two vertices. Two independent sets have at most four vertices, and hence, the remainder contains at least two vertices. To prove that G is minimal, note that both $H_1 = K_{2,2,2,2}$ and $H_2 = K_{1,2,2,2} \oplus K_1$ have a $(1, 2, 1)$-partition in which the forest is a tree on three vertices, each independent set has size two, and the set R has one vertex.

For $G = 2K_3 \oplus K_{2,2}$, any forest has at most four vertices. Any independent set has size at most two in G. Hence, two independent sets contain at most four vertices, leaving at least two vertices in the remainder. Hence, G does not admit a $(1, 2, 1)$-partition. We will prove that all the graphs obtained from deleting one vertex from G have a $(1, 2, 1)$-partition. That is, $H_1 = (K_2 + K_3) \oplus K_{2,2}$ and $H_2 = 2K_3 \oplus K_{1,2}$ have a $(1, 2, 1)$-partition. Both the cographs have a $(1, 2, 1)$-partition where the forest consists of two edges and contains four vertices. Each independent set has size two, and one vertex in the set R.

Suppose that $G = (K_{2,2} + K_3) \oplus K_{2,2}$. Any independent set must lie entirely on one side of the join. The bigger side of the join contains two disjoint cycles which can not be partitioned into a forest and a vertex. Hence, at least one independent set must lie on this side of the join, such a set can have at most three vertices. However, the remainder contains an induced $K_{2,2} \oplus \overline{K_{1,2}}$, which is an obstruction to $(1, 1, 1)$-partition. Therefore, G is an obstruction to $(1, 2, 1)$-partition. To prove that G is a minimal obstruction, we provide a $(1, 2, 1)$-partition for the following cographs: $H_1 = (K_{1,2}+K_3)\oplus K_{2,2}$, $H_2 = (K_{2,2}+K_2)\oplus K_{2,2}$ and $H_3 = (K_{2,2} + K_3) \oplus K_{1,2}$. The cograph H_1 admits a partition where the forest covers five vertices and consists of a star on three vertices and an edge. Each independent set has size two, and one vertex in the set R. The cograph H_3 admits a similar $(1, 2, 1)$-partition while the cograph H_2 has a partition where the forest has five vertices. One independent set has size two, while the other independent set has a single vertex. The set R contains one vertex.

For $G = 2K_{2,2} \oplus K_{3,3}$, any independent set in G must lie entirely on one side of the join. One side of the join contains two disjoint four cycles that cannot be partitioned into a forest and a vertex. Hence, at least one independent set must lie on this side of the join; such a set has at most four vertices. However, the remainder contains an induced $K_{3,3,3}$, which is an obstruction to $(1,1,1)$-partition. Hence, G is an obstruction to $(1,2,1)$-partition. To see that G is a minimal obstruction, note that both $H_1 = (K_{2,2}+K_{1,2}) \oplus K_{3,3}$ and $H_2 = 2K_{2,2} \oplus K_{2,3}$ have a $(1,2,1)$-partition. For $H_1 = (K_{2,2} + K_{1,2}) \oplus K_{3,3}$, one such partition has a forest consisting of two copies of $K_{1,2}$, the two independent sets have three vertices each, and the set R has one vertex. For $H_2 = 2K_{2,2} \oplus K_{2,3}$, a partition can be obtained where the forest is a star on five vertices, one independent set contains four vertices and the other independent set contains three vertices, and one vertex is assigned to the set R.

Consider $G = 2K_{2,2} \oplus 2K_{2,2}$. Any independent set must lie entirely on one side of the join. One side of the join contains two disjoint four cycles, which cannot be partitioned into a forest and a vertex. Hence, at least one independent set must lie on this side of the join; such a set has at most four vertices. However, the remaining vertices contain an induced $(\overline{K_3}) \oplus 2K_{2,2}$, which is an obstruction to $(1,1,1)$-partition. Therefore, G is an obstruction to $(1,2,1)$-partition. Removing a vertex, we obtain the graph $((K_{1,2}+K_{2,2}) \oplus 2K_{2,2})$, which has the following $(1,2,1)$-partition. Two independent sets containing four vertices each on the bigger side of the join, forest consisting of $2K_{1,2}$ on the smaller side of the join, leaving one vertex in the set R. Thus $2K_{2,2} \oplus K_{2,2}$ is a minimal cograph obstruction to $(1,2,1)$-partition.

To prove that $G = (K_4 + K_{2,2,2}) \oplus \overline{K_2}$ is a minimal obstruction; we consider which vertex v can be removed such that $G - v$ admits a $(1,2,0)$-partition. Note that G contains two copies of K_5 and one copy of $K_{2,2,2,2}$, both of which are an obstruction to $(1,2,0)$-partition. Hence, there is no vertex in G such that its removal yields a $\{K_5,\ K_{2,2,2,2}\}$-free graph. Therefore, G is an obstruction to (1,2,1)-partition. The cograph $(K_3 + K_{2,2,2}) \oplus \overline{K_2}$ admits a $(1,2,1)$-partition, where one independent set has size three, and the other independent set covers two vertices. The forest has five vertices and consists of an edge and a star on three vertices. This leaves a single vertex in the set R. The graph $H_2 = (K_4 + K_{1,2,2}) \oplus \overline{K_2}$ has a similar $(1,2,1)$-partition. Finally, for cograph $H_3 = (K_4 + K_{2,2,2}) \oplus K_1$, we remove the single vertex on one side of the join. The two independent sets have three vertices each, and the forest has four vertices. Hence, G is a minimal obstruction.

For $G = 2K_3 \oplus 2K_2$, any forest can cover at most four vertices. An independent set has size at most two in G. Hence, two independent sets cover at most four vertices. The remainder contains at least two vertices. Therefore, G does not admit a $(1,2,1)$-partition. Now we will show that the cographs $H_1 = (K_2 + K_3) \oplus 2K_2$, and $H_2 = 2K_3 \oplus \overline{K_{1,2}}$ admit a $(1,2,1)$-partition. For $H_1 = (K_2 + K_3) \oplus 2K_2$, the partition consists of a forest that is $2K_2$ and has four vertices. Each independent set has size two and $|R| = 1$. For $H_2 = 2K_3 \oplus \overline{K_{1,2}}$, the forest consists of the $2K_2$ and has four vertices. Each independent set has two vertices, and the set R contains one vertex. Therefore, $G = 2K_3 \oplus 2K_2$ is a minimal obstruction.

To prove that $G = 2K_4 \oplus \overline{K_2}$ is an obstruction, we consider which vertex $v \in V(G)$ can be removed such that $G-v$ admits a $(1,2,0)$-partition. Hence, for some vertex v, $G-v$ must be $\{K_5\ ,\ K_{2,2,2,2}\}$-free. However, G contains four copies of K_5. Hence, there is no vertex in G such that its removal yields a K_5-free graph; we conclude that G is an obstruction to $(1,2,1)$-partition. For $H_1 = (K_4 + K_3) \oplus \overline{K_2}$, we obtain a $(1,2,1)$-partition where each independent set has size two. The forest has two edges and contains four vertices leaving exactly one vertex in the set R. For the cograph $H_2 = 2K_4 \oplus K_1$, remove the single vertex on one side of the join. The forests has two edges and four vertices. Each independent set has size two.

Suppose that $G = (2K_2 \oplus 2K_2) \oplus \overline{K_2}$. Since we know that $H = 2K_4 \oplus \overline{K_2}$ is an obstruction to $(1,2,1)$-partition, G is also an obstruction since G contains H as a subgraph. The minimality of G follows from the fact that both $H_1 = (2K_2 \oplus \overline{K_{1,2}}) \oplus \overline{K_2}$ and $H_2 = (2K_2 \oplus 2K_2) \oplus K_1$ admit a $(1,2,1)$-partition. For $H_2 = (2K_2 \oplus 2K_2) \oplus K_1$, we remove the only vertex on one side of the join. Consider $2K_2$ as a forest of the partition. The remainder can be partitioned into two independent sets. For $H_1 = (2K_2 \oplus \overline{K_{1,2}}) \oplus \overline{K_2}$, remove one vertex from the K_2 of the $\overline{K_{1,2}}$. We take $2K_2$ as a forest in the partition, and each independent set has two vertices.

□

3.3.2 The completeness of the lists

Let G be a connected cograph and assume that G does not admit a $(1,2,0)$-partition. If the cograph G has a $(1,2,1)$-partition, then it is easy to see that there exists a vertex v such that $G-v$ has a $(1,2,0)$-partition.

From the discussion above, we can conclude that any disconnected cograph with two or more components that do not admit a $(1,2,0)$-partition is an obstruction to $(1,2,1)$-partition. More precisely, we have the following lemma.

Lemma 71. *Let G be a disconnected cograph, if G has two or more connected components which do not admit a $(1,2,0)$-partition, then G does not admit a $(1,2,1)$-partition.* □

Lemma 72. *Let H be a disconnected cograph. If H is a minimal obstruction to $(1,2,1)$-partition, then H contains exactly two connected components such that each component is a minimal obstruction to $(1,2,0)$-partition.*

Proof. Since H is a disconnected cograph, H consists of two or more connected components. If none of the components is an obstruction to $(1,2,0)$-partition, the cograph H admits a $(1,2,0)$-partition and hence admits a $(1,2,1)$-partition. Therefore, H contains at least one connected component that does not admit a $(1,2,0)$-partition. If exactly one connected component does not admit a $(1,2,0)$-partition, then H is not minimal since a vertex v can be removed such that $H-v$ also does not admit a $(1,2,1)$-partition.

Hence, H has at least two such components. Using Lemma 71, and the assumption that H is minimal, H consists of exactly two connected components such that each component does not admit a $(1,2,0)$-partition. If one of the connected components, say H_1, is not a minimal obstruction to $(1,2,0)$-partition, then there exists an induced subgraph of H_1, which is a minimal obstruction to $(1,2,0)$-partition. This contradicts the assumption that H is minimal. □

Lemma 73. *Each graph in $\mathcal{F}_5$ is a minimal cograph obstruction to $(1,2,1)$-partition.*

Proof. Since K_5 and $K_{2,2,2,2}$ are both minimal obstructions to $(1,2,0)$-partition, it follows from Lemma 71 that the members of family $\mathcal{F}_3$ are an obstruction to (1,2,1)-partition. Lemma 72 guarantees the minimality. □

We now prove that the list of minimal cograph obstructions for $(1,2,1)$-partition given in Lemma 70 is complete for connected cographs.

Theorem 74. *A connected cograph admits a $(1,2,1)$-partition if and only if it is $\mathcal{F}_4$-free.*

Proof. Let G be a connected cograph; we assume that G is $\mathcal{F}_4$-free. We proceed to provide a $(1,2,1)$-partition for G. Since G is a connected cograph, there exist cographs G_1 and G_2 such that $G = G_1 \oplus G_2$.

1. **Assume that both G_1, G_2 are forests.** If G_1 and G_2 are forests, we obtain a $(1,2,1)$-partition as follows. The graph G_1 can be considered as the forest in the partition. The graph G_2 is a forest and can be partitioned into two independent sets.

2. **Assume that exactly one of them, say G_2, is a forest.** Since the graph G_1 contains an induced cycle, and the only cycles possible in a cograph are C_3 and C_4, we consider the following subcases.

(2.1) **Assume that G_1 is C_3-free.** In this case, G_1 is a bipartite graph that contains a C_4. As G_2 is a forest, we have a trivial $(1,2,1)$-partition with two independent sets and a forest.

(2.2) **Assume that G_1 contains a C_3.** We consider several possible cases, noting that in all the cases, where G_2 has at least one edge, G_1 does not contain K_4, since G is K_6-free.

(2.2.1) **Suppose that G_2 has at least three vertices and at least one edge.** Consider a copy of C_3 on $\{v_1, v_2, v_3\}$ in G_1, and the component B of G_1 containing it. Since B is a connected cograph, there exist cographs B_1, B_2 such that $B = B_1 \oplus B_2$. Since B does not contain a K_4, neither B_1 nor B_2 contains a K_3. So, we assume without loss of generality that $v_1, v_2 \in V(B_1)$, and $v_3 \in V(B_2)$; moreover, B_2 is an independent set. If B_1 has an induced C_4, then B_2 must be a single vertex because G is $K_{2,2,2} \oplus \overline{K_{1,2}}$-free and $K_{2,2,2,2} \oplus K_1$-free. (Note that G_2 contains either a copy of $\overline{K_{1,2}}$ or $K_{1,2}$) In conclusion, each component $B = B_1 \oplus B_2$ of G_1, that contains a C_3 either has a single vertex in B_2 and a bipartite B_1, or an independent set B_2 and a forest B_1. Each component of G_1 without a C_3 is bipartite.

Now, if G_2 has exactly one connected component, i.e., G_2 is a star, then let u be the centre vertex of the star G_2. We remove the vertex u so that we obtain $G_2 - u$ as one independent set of the desired $(1,2,1)$-partition. We now partition G_1 into a forest and an independent set; it suffices to partition each component B of G_1 separately. For the components $B = B_1 \oplus B_2$ of G_1 that contain C_3 and a single vertex $v^{'}$ in B_2, we construct a $(1,1)$-partition as follows. Here, we obtain a star at $v^{'}$ using one part of the bipartition of B_1. The other part of the bipartition yields an independent set. For the components, $B = B_1 \oplus B_2$ with C_3, where B_2 is an independent set, and B_1 is a forest; we already have a desired $(1,1)$-partition. Finally, each remaining component B that does not contain a C_3 is bipartite, and we take one part as the forest and the other part as an independent set for the desired $(1,1)$-partition.

Hence, we assume that G_2 has at least two connected components such that each component has at least one edge. In that case, the graph G_1 has exactly one connected component B, which has C_3. Otherwise, G contains $2K_3 \oplus 2K_2$ as an induced subgraph. If B_2 has exactly one vertex, say $u^{'}$, we remove $u^{'}$ from the graph and obtain a $(1,2,0)$-partition as follows. From the bipartite graph $G_1 - u'$, we obtain the two independent sets in the partition, and we take G_2 as the forest in the partition. If B_2 has at least two vertices, B_1 is a forest. Moreover, B_1 is a tree(star) because G is $(2K_2 \oplus \overline{K_2} \oplus 2K_2)$-free. Let B_1 be a star at v'. We remove v', and from $G_1 - v^{'}$, we obtain the two independent sets in the partition, and we take G_2 as the forest in the desired $(1,2,1)$-partition.

(2.2.2) **Assume that G_2 has exactly two vertices which are adjacent.** Since G_2 has an edge, G_1 does not contain an induced K_4 as G is K_6-free. Therefore, G_1 is three-colourable. We take one of the colour classes along with one vertex of G_2 as the forest of our $(1,2,1)$-partition. The remaining two colour classes form the two independent sets of the partition. The remainder is just a vertex in G_2.

(2.2.3) **Assume G_2 has exactly two vertices which are not adjacent.** If G_1 does not contain an induced K_4, we obtain a partition of G as follows. Since G_1 is K_4-free, it is three-colourable. We take one of the colour classes along with one vertex of G_2 as the forest of our $(1,2,1)$-partition. The remaining two colour classes form the two independent sets of the partition. The remainder is just a vertex in G_2.

Now we assume that G_1 has a K_4. Since G is $(2K_4 \oplus \overline{K_2})$-free, exactly one connected component of G contains K_4.

Note that while at most one component of G_1 has a K_4, there could be other components B of G_1 without a K_4. Either B has $(1,1,0)$-partition or B contains a minimal cograph obstruction to $(1,1,0)$-partition, i.e., B contains either K_4 or $K_{2,2,2}$ as an induced subgraph. Since B does not contain K_4, B must contain $K_{2,2,2}$ as an induced subgraph. But then we end up with $(K_4 + K_{2,2,2}) \oplus \overline{K_2}$ from the family $\mathcal{F}_4$. Hence, the components of G_1 that do not contain K_4 have a $(1, 1, 0)$-partition.

Now we consider component B of G that contains K_4 as an induced subgraph. Since B is a connected cograph, there exist cographs B_1, B_2 such that $B = B_1 \oplus B_2$.

Suppose first that both B_1, B_2 are bipartite. Note that both B_1 and B_2 cannot contain an induced C_4 since G is $K_{2,2,2,2} \oplus K_1$-free. If both B_1 and B_2 are forests, at least one of the forests must be connected, i.e., a star since G is $(2K_2 \oplus 2K_2) \oplus \overline{K_2}$-free. Without loss of generality, we may assume that B_2 is a star. The $(1, 2, 1)$-partition, in this case, is constructed as follows. We remove a vertex in B_2 to obtain one independent set of the $(1, 2, 1)$-partition. We take G_2 as the other independent set, and B_1 as the forest in the partition.

Hence, we assume that B_1 has a C_4, and B_2 is a forest. In fact, the graph B_2 is just an edge, say uv, since G is $K_{2,2,2,2} \oplus K_1$-free and $K_{2,2,2} \oplus \overline{K_{1,2}}$-free. In this case, a $(1, 2, 1)$-partition is obtained by removing the vertex u, we take one part of the bipartite graph B_1 as one independent set of the partition, G_2 as the other independent set, and one star formed at v with the other part of B_1 as the forest in our $(1, 2, 1)$-partition.

Thus, we may assume that one of B_1, B_2, say B_1, contains a C_3. Since G is K_6-free, B should be K_5-free. Hence, B_2 must be an independent set. We further consider each connected component $D = D_1 \oplus D_2$ of B_1. At least one such component D' must contain a C_3, but there could also be bipartite components D, as well as other components that have a C_3; all must be K_4-free.

If B_2 has at least two vertices, then exactly one component, namely D', of B_1 has a cycle (specifically a C_3). Bipartite components D cannot have a four cycle because G is $((K_3 + K_{2,2}) \oplus K_{2,2})$-free. Moreover, no other component $D \neq D'$ can have a C_3, because G is $(2K_3 \oplus K_{2,2})$-free. Hence, if B_2 has at least two vertices, all the components D of B_1, other than D', are trees.

Suppose that $\{v_1, v_2, v_3\}$ form a C_3 in D'. Since $D' = D'_1 \oplus D'_2$ is K_4-free, neither of the graphs D'_1, D'_2 have a C_3. So we may assume $v_1, v_2 \in V(D'_1)$ and $v_3 \in V(D'_2)$; moreover, we may assume D'_1 is a bipartite graph, and D'_2 is an independent set.

If the bipartite graph D'_1 contains a C_4, then both D'_2 and B_2 must consist of a single vertex because G is $K_{2,2,2,2} \oplus K_1$-free.

If D'_1 is a forest with more than two vertices, then it contains an induced $K_{1,2}$ or $\overline{K_{1,2}}$. Therefore, at least one of D'_2, B_2 must be a single vertex, since G is $K_{2,2,2,2} \oplus K_1$-free and $K_{2,2,2} \oplus \overline{K_{1,2}}$-free.

Otherwise, D'_1 is just the edge v_1v_2.

Finally, for the components, $D = D_1 \oplus D_2$ that do not contain a C_3, D_1 is just an independent set.

To summarize the discussion, B_2 is an independent set, and B_1 consists of components $D = D_1 \oplus D_2$, where each D_2 is an independent set and each D_1 is bipartite, with the following four possibilities: (i) D_1 contains a C_4, in which case D_2, as well as B_2, have a single vertex; (ii) D_1 is a forest with more than two vertices, in which case either D_2 or B_2 has a single vertex; (iii) D_1 is an edge v_1v_2; or (iv) D_1 is an independent set. Moreover, in cases (ii - iv), if B_2 has more than one vertex, all but one component D of B_1 are trees.

We first describe a $(1,1,1)$-partition of $B = B_1 \oplus B_2$ when B_2 has at least two vertices. In this case, there is one component $D' = D'_1 \oplus D'_2$ of B_1 such that D' has a C_3. Here, D'_1 is a forest containing one or more vertices (cases (ii, iii)), and all other components D of B are trees. We obtain a $(1,1,1)$-partition of G_1 as follows. If D'_1 is just an edge, say xy, we remove the vertex y, and the forest consists of a star at x covering the independent set D'_2, along with the rest of the tree components of B_1. The independent set of the partition consists of B_2. If D'_1 has more than two vertices, then D'_2 is a single vertex u, we remove u and take D'_1 together with all other components D as the forest; we take B_2 as the independent set in the partition.

Now consider a component $B = B_1 \oplus B_2$ of G_1, when B_2 has a single vertex, say v. To obtain a $(1,1,1)$-partition, we remove the vertex v, we put together one forest for a $(1,1,1)$-partition of B from the following forests in the various components $D = D_1 \oplus D_2$ of B. From components D of type (i) we take the star centred at the single vertex of D_2 and covering one part of the bipartition of D_1; from components D of type (ii-iv) we take the forest D_1.

The remaining vertices in G_1 form an independent set in the $(1,2,1)$-partition. (These will be the remaining part of the bipartition of D_1 in the first case and the independent set D_2 in the second case.)

(2.2.4) **Finally, we assume that G_2 is just a single vertex, say v.** The proof here is similar to the case (2.2.3), except that in the case (i) when D_1 contains an induced C_4, we can only claim that either B_2 or D_2 is a single vertex, and in the case (ii) when D_1 is a forest with more than two vertices, we cannot claim anything about the size of B_2 or D_2. Also, we can no longer claim that at most one component of G_1 has K_4, there could be multiple components of G_1 that have K_4 as an induced subgraph. Clearly, it suffices to find a partition for each component B of G_1 separately. Before describing the partition, recall that G consists of a vertex v adjacent to all other vertices, and $G - v$ has components $B = B_1 \oplus B_2$ of two kinds, either B_2 is a single vertex, or B_2 is an independent set with at least two vertices. For components $B' = B'_1 \oplus B'_2$ of the first kind (where B'_2 is a single vertex w), we only note that B'_1 consists of bipartite components D. For the components B of the second kind (where B_2 is a larger independent set), we distinguish components $D' = D'_1 \oplus D'_2$ in which D'_2 consists of a single vertex z, and other components $D = D_1 \oplus D_2$ where D_2 is a larger independent set and D_1 is a forest. To obtain the required (1,2,1)-partition of G, we remove the single vertex of G_2, and we construct the forest of the partition as follows. For the components B' of the first kind (where B'_2 is a single vertex w), it is a star centred at w and covering the sets D_2 of all components D of B'_1. For the components B_2 of the second kind where D_2' consists of a single vertex z, the forest consists of a star centred at z covering one part of the bipartition of D_1'. Finally, for the components B_2 where D_2' is a larger independent set, we take D_1' as the forest of the partition. The remaining vertices are easily seen to form two independent sets for the desired $(1, 2, 1)$-partition. (These will be the two parts of the bipartite graph D_1', for the components of the first kind. For the components of the second kind where D_2' is a single vertex, the two independent sets will be the remaining part of the bipartite graph D_1' and B_2. Finally, for the remaining case, the two independent sets consist of B_2 and D_2'.)

3. **Assume that none of the cographs G_1, G_2 are forests:**

(3.1) **Both G_1 and G_2 are C_3-free**. This implies that both cographs G_1 and G_2 are bipartite, and both have an induced C_4. Both G_1 and G_2 cannot have more than one connected component with C_4 since G is $(2K_{2,2} \oplus 2K_{2,2})$-free. Hence, without loss of generality, we may assume that G_2 has exactly one component, say A, with a C_4, and the other components are trees. Note that A must be a complete bipartite graph since G_2 has no induced P_4. The graph G_1 also contains connected component B such that B is a complete bipartite graph. If G_1 has other components with an induced C_4, then one of the parts of A in G_2 has exactly two vertices, because G is $2K_{2,2} \oplus K_{3,3}$-free. If the other connected components of G_1 are trees, then one of the subgraphs A or B has a bipartition with one of the parts having exactly two vertices, since G is $K_{3,3,3,3}$-free. In either case, we can obtain a $(1, 2, 1)$-partition of G as follows. Suppose the connected component A of the graph G_2 has a bipartition (X, Y), where X has exactly two vertices. We remove one vertex in X. To construct the forest, we include Y along with the remaining vertex in X and the remaining tree components of G_2. From the bipartite graph G_1, we obtain the two independent sets in the desired $(1, 2, 1)$-partition.

(3.2) **Let G_1 is C_3-free, but G_2 contains a C_3.** Since G_1 contains a $K_{2,2}$, and since G is $(2K_3 \oplus K_{2,2})$-free and $((K_{2,2} + K_3) \oplus K_{2,2})$-free, there is exactly one component of G_2 with a C_3, and other components of G_2 are trees. Let the set $\{v_1, v_2, v_3\}$ induce a C_3 in G_2, and let B be the component of G_2 containing it. Since B is a connected, there exist cographs B_1 and B_2 such that $B = B_1 \oplus B_2$. The component B cannot contain an induced K_4, and hence none of the graphs B_1, B_2 have a C_3. So, we assume without loss of generality that $v_1, v_2 \in V(B_1)$, and $v_3 \in V(B_2)$; moreover, B_2 must be an independent set since G is K_6-free. If B_2 has at least two elements, then B_1 must be a K_2, since G is $(K_{2,2,2} \oplus \overline{K_{1,2}})$-free and $(K_{2,2,2,2} \oplus K_1)$-free. Hence, either B_1 is a K_2 or B_2 is a K_1. We construct a $(1, 2, 1)$-partition in both the cases as follows.

When $B_1 = K_2$, we remove one vertex of the K_2. We obtain the forest for the partition by taking a star formed by the remaining vertex in B_1 and the independent set B_2 along with the remaining tree components of G_2. The bipartite graph G_1 yields the two independent sets in the $(1, 2, 1)$-partition.

When B_2 consists of a single vertex, say v, we remove this vertex. We take $G_2 - v$ as the forest, and the two parts of the bipartite graph G_1 yield the independent sets of the $(1, 2, 1)$-partition. This concludes the proof. □

We conclude with the proof of Theorem 69 we presented in section 3.3

Proof. Assume that G is a disconnected cograph. Since G is $\mathcal{F}_2$-free by assumption, every connected component of G admits a $(1, 2, 1)$-partition. Since G is also $\mathcal{F}_3$-free, all components of G except for one admit a $(1, 2, 0)$-partition. Hence, G admits a $(1, 2, 1)$-partition. □

3.4 Concluding remarks

In this thesis, we provide a complete list of minimal obstructions for the (p, q)-partition for the case $p = 2$ and $q = 1$ on cographs. For the (p, q, r)-partition problem on cographs, we were able to find a complete list of minimal obstructions for the $(1, 1, 1)$ case, and the $(1, 2, 1)$ case. We also provided certifying algorithms for the partition problems that run in $O(m+qn)$ time for the $(1, q)$-partition problem, and $O(m+n)$ time for the $(2, 0)$ and $(2, 1)$-partition problems. We were also able to conclude that a uniform description of minimal obstructions does not exist for the case of $p = 2$, in contrast to the case of $p = 1$ for $(1, q)$-partitions, where we have only two obstructions for any $q \geq 0$. It would be interesting to see whether similar completeness results and certifying algorithms can be obtained for the class of P_4-sparse graphs. A graph is P_4 sparse if and only if any set of five vertices induce at most one P_4. The class of P_4 sparse graphs contains all cographs and a similar unique tree representation of P_4 sparse graphs can be obtained [48]. Another line of investigation would be to examine the (p, q)-partition problem and the (p, q, r)-partition problem on classes of higher clique-width. As we previously discussed, cographs are exactly the graphs with clique-width two, and hence, it would be interesting to see if similar minimal obstruction characterization results can be obtained for classes of higher clique-width.

Bibliography

[1] Kenneth I Appel and Wolfgang Haken. *Every planar map is four colorable*, volume 98. American Mathematical Soc., 1989.

[2] Claude Berge. The coloring problems in graph theory. 9:123–160, 1960.

[3] Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on computing*, 25(6):1305–1317, 1996.

[4] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1 – 45, 1998.

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

[6] Francesco Brioschi. On non-serial dynamic programming. *J. Comb. Theory, Ser. A*, 14(2):137–148, 1973.

[7] Rowland Leonard Brooks. On colouring the nodes of a network. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 37, pages 194–197. Cambridge University Press, 1941.

[8] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171 – 176, 1996.

[9] Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of mathematics, ISSN 0003-486X, Vol. 164, Nº 1, 2006, pags. 51-229*, 164, 01 2003.

[10] V. Chvátal, C. T. Hoàng, N. V. R. Mahadev, and D. De Werra. Four classes of perfectly orderable graphs. *Journal of Graph Theory*, 11(4):481–495, 1987.

[11] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.

[12] Derek Corneil, Stephan Olariu, and Lorna Stewart. The lbfs structure and recognition of interval graphs. *SIAM J. Discrete Math.*, 23:1905–1953, 01 2009.

[13] Derek G Corneil and Jean Fonlupt. The complexity of generalized clique covering. *Discrete Applied Mathematics*, 22(2):109–118, 1988.

[14] Derek G. Corneil, Yehoshua Perl, and Lorna K Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[15] Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 34(4):825–847, 2005.

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

[17] Bruno Courcelle, Johann A Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[18] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77 – 114, 2000.

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

[20] Dantas, Simone, de Figueiredo, Celina M.H., Gravier, Sylvain, and Klein, Sulamita. Finding h-partitions efficiently. *RAIRO-Theor. Inf. Appl.*, 39(1):133–144, 2005.

[21] Sebastián González Hermosillo de la Maza, Pavol Hell, César Hernández Cruz, Seyyed Aliasghar Hosseini, and Payam Valadkhan. Vertex arboricity of cographs. 2019.

[22] Raquel de Souza Francisco, Sulamita Klein, and Loana Tito Nogueira. Characterizing (k,l)–partitionable cographs. *Electronic Notes in Discrete Mathematics*, 22:277 – 280, 2005. 7th International Colloquium on Graph Theory.

[23] Marc Demange, Tınaz Ekim, and Dominique de Werra. Partitioning cographs into cliques and stable sets. *Discrete Optimization*, 2(2):145 – 153, 2005.

[24] Zdenek Dvorak, Ken-ichi Kawarabayashi, and Robin Thomas. Three-coloring triangle-free planar graphs in linear time. pages 1176–1182, 01 2009.

[25] Dennis D. A. Epple and Jing Huang. A note on the bichromatic numbers of graphs. *Journal of Graph Theory*, 65(4):263–269, 2010.

[26] P Erdős and A Rényi. On random graphs I. *Publ. math. debrecen*, 6(290-297):18, 1959.

[27] Tomás Feder, Pavol Hell, and Winfried Hochstättler. *Generalized Colourings (Matrix Partitions) of Cographs*, pages 149–167. Birkhäuser Basel, Basel, 2007.

[28] Tomas Feder, Pavol Hell, Sulamita Klein, and Rajeev Motwani. Complexity of graph partition problems. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, page 464–472, New York, NY, USA, 1999. Association for Computing Machinery.

[29] Tomás Feder and Pavol Hell. On realizations of point determining graphs, and obstructions to full homomorphisms. *Discrete Mathematics*, 308(9):1639 – 1652, 2008.

[30] Stéphane Foldes and Peter L Hammer. *Split graphs.* Universität Bonn. Institut für Ökonometrie und Operations Research, 1976.

[31] Delbert Fulkerson and Oliver Gross. Incidence matrices and interval graphs. *Pacific journal of mathematics*, 15(3):835–855, 1965.

[32] Micheal R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman and Company, New York, 1979.

[33] P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16:539–548, 1964.

[34] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics.* Elsevier Science, 2 edition, 2004.

[35] Martin Grötschel, László Lovász, and Alexander Schrijver. *Stable Sets in Graphs*, pages 272–303. Springer Berlin Heidelberg, Berlin, Heidelberg, 1988.

[36] H. GROTZSCH. Ein dreifarbensatz fur dreikreisfreie netze auf der kugel. *Wiss. Z. Martin Luther Univ. Halle-Wittenberg, Math. Nat. Reihe*, 8:109–120, 1959.

[37] András Hajnal and János Surányi. Über die auflösung von graphen in vollständige teilgraphen. *Ann. Univ. Sci. Budapest, Eötvös Sect. Math*, 1:113–121, 1958.

[38] S. L. Hakimi and E. F. Schmeichel. A note on the vertex arboricity of a graph. *SIAM Journal on Discrete Mathematics*, 2(1):64–67, 1989.

[39] PL Hammer and B Simeone. The splittance of a graph, univ. of waterloo, dept. of combinatorics and optimization, res. *Report CORR*, pages 77–39, 1977.

[40] Juris Hartmanis. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review*, 24(1):90, 1982.

[41] Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[42] PJ Heawood. Map colour theorem', quart. d. *J. Math*, 1890.

[43] Pavol Hell. Graph partitions with prescribed patterns. *European Journal of Combinatorics*, 35:335 – 353, 2014. Selected Papers of EuroComb'11.

[44] Pavol Hell, César Hernández-Cruz, and Anurag Sanyal. Partitioning cographs into two forests and one independent set. In Manoj Changat and Sandip Das, editors, *Algorithms and Discrete Applied Mathematics - 6th International Conference, CALDAM 2020, Hyderabad, India, February 13-15, 2020, Proceedings*, volume 12016 of *Lecture Notes in Computer Science*, pages 15–27. Springer, 2020.

[45] Pavol Hell, Sulamita Klein, Loana Tito Nogueira, and Fábio Protti. Partitioning chordal graphs into independent sets and cliques. *Discrete Applied Mathematics*, 141(1):185 – 194, 2004. Brazilian Symposium on Graphs, Algorithms and Combinatorics.

[46] Pavol Hell and Jaroslav Nešetřil. On the complexity of h-coloring. *Journal of Combinatorial Theory, Series B*, 48(1):92 – 110, 1990.

[47] J Hopcroft and R Tarjan. Efficient planarity testing journal of the association for computing machinery. 1974.

[48] Beverly Jamison and Stephan Olariu. Recognizing $p_4 $-sparse graphs in linear time. *SIAM Journal on Computing*, 21(2):381–406, 1992.

[49] Donald B Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.

[50] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[51] Norbert Korte and Rolf Möhring. A simple linear -time algorithm to recognize interval graphs. pages 1–16, 06 1986.

[52] Casimir Kuratowski. Sur le probleme des courbes gauches en topologie. *Fundamenta mathematicae*, 15(1):271–283, 1930.

[53] C Lekkeikerker and J Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962.

[54] Linda Lesniak and H Joseph Straight. The cochromatic number of a graph. *Ars Combinatoria*, 3:39–46, 1977.

[55] László Lovász. A characterization of perfect graphs. *Journal of Combinatorial Theory, Series B*, 13(2):95–98, 1972.

[56] Madhav V Marathe, R Ravi, and C Pandu Rangan. Generalized vertex covering in interval graphs. *Discrete Applied Mathematics*, 39(1):87–93, 1992.

[57] Jan Mycielski. Sur le coloriage des graphs. *Colloquium Mathematicae*, 3(2):161–162, 1955.

[58] Giri Narasimhan and Rachel Manber. *The Maximumk-Colorable Subgraph Problem.* PhD thesis, 1989. AAI8923342.

[59] Joseph G. Peters and Arthur L. Liestman. Partitions of generalized split graphs. 2012.

[60] ROSE DJ; TARJAN RE. Algorithmic aspects of vertex elimination on directed graphs. *S.I.A.M. J. APPL. MATH.; U.S.A.; DA. 1978; VOL. 34; NO 1; PP. 176-197; BIBL. 1 P. 1/2*, 1978.

[61] Robert E. Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984.

[62] Payam Valadkhan. *List matrix partitions of special graphs.* PhD thesis, Applied Sciences: School of Computing Science, 2013.

[63] Dominic JA Welsh and Martin B Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967.

[64] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, NJ, 1996.

[65] Mihalis Yannakakis. Node-and edge-deletion np-complete problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, page 253–264, New York, NY, USA, 1978. Association for Computing Machinery.

[66] Mihalis Yannakakis and Fanica Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24(2):133–137, 1987.