Graph Colouring with Input Restrictions

SONG, JIAN

Graph Colouring with Input Restrictions

Jian Song

School of Engineering and Computing Sciences
Durham University

A thesis submitted for the degree of
Doctor of Philosophy

December 2012

Contents

List of Figures

List of Tables

Abstract

In this thesis, we research the computational complexity of the graph colouring problem and its variants including precolouring extension and list colouring for graph classes that can be characterised by forbidding one or more induced subgraphs. We investigate the structural properties of such graph classes and prove a number of new properties. We then consider to what extent these properties can be used for efficiently solving the three types of colouring problems on these graph classes. In some cases we obtain polynomial-time algorithms, whereas other cases turn out to be NP-complete.

We determine the computational complexity of k-Colouring, k-Precolouring Extension and List k-Colouring on P_k-free graphs. In particular, we prove that 4-Precolouring on P_8-free graphs is NP-complete, 4-Precolouring Extension on P_7-free graphs is NP-complete, and List 4-Colouring on P_6-free graphs is NP-complete. In addition, we show the existence of an integer r such that 4-Colouring is NP-complete for P_r-free graphs with girth 4. In contrast, we determine for any fixed girth $g \geq 4$ a lower bound $r(g)$ such that every $P_{r(g)}$-free graph with girth at least g is 3-colourable. We also prove that 3-List Colouring is NP-complete for complete graphs minus a matching. We present a polynomial-time algorithm for solving 4-Precolouring Extension on $(P_2 + P_3)$-free graphs, a polynomial-time algorithm for solving List 3-Colouring on $(P_2 + P_4)$-free graphs, and a polynomial-time algorithm for solving List 3-Colouring on sP_3-free graphs. We prove that List k-Colouring for $(K_{s,t}, P_r)$-free graphs is also polynomial-time solvable. We obtain several new dichotomies by combining the above results with some known results.

Declaration

No part of this thesis has previously been submitted for any degree at any institution. The results contained in this thesis are based on six research papers [18, 19, 41, 42, 43, 44], all of which have been subject to peer review. In particular, the paper [19] and the paper [41] are the journal versions of the paper [18] and the paper [42], respectively. A chapter of this thesis contains several sections, and each section is based on one result in one of these papers. At the beginning of each chapter, we mention where the result in every section of this chapter has been published. Although the results in this thesis are obtained by joint research, I have actively participated in the discussions leading to these results and my contribution to them has been significant.

Acknowledgements

Perhaps three years does not sound like a very long time, but the past three years have meant a lot to me. In October 2009 I started my PhD research at the School of Engineering and Computing Sciences of Durham University. The beginning of my research life was not very easy. Although I had lived in the UK for a few years, many things including local culture were still quite new to me. I must say that it was my supervisor, Daniël Paulusma, who helped me through this tough period by assisting me to conquer the barriers I met and by teaching me how to do research in Theoretical Computer Science. He always found time for me whenever I needed it. I cannot see any way to learn so much in the past three years without him. Thank you for all your help, Daniël.

I would also like to thank you, Hajo Broersma, for being my second supervisor before leaving Durham University, for guiding me so much in my research, and for advising me when I met difficulties. A big thanks to you, Petr Golovach, for patiently teaching me a variety of research topics and techniques, and for all the inspiring discussions and suggestions. Your help has been invaluable for my development as a researcher. I have much enjoyed the time of doing research together with all of you, Daniël, Hajo and Petr.

Matthew Johnson, thank you for agreeing to be my second supervisor after Hajo left and for treating me as a friend. David Manlove and Tom Friedetzky, thank you for agreeing to be my examiners, for your time reading my thesis, and for making the viva so interesting and pleasant. I would also like to thank you, Brendan Hodgson, for bringing me to Durham University, which has led me to meet so many nice people and make so many friends.

Speaking of my friends, it was always fun to spend time with them. I would like to thank all of you, my friends, for sharing your happiness with me. Especially, I would like to thank you, Pim van 't Hof. Thank you for always giving me help not only for my

research but also for daily life whenever I needed it. Finally, I thank my parents who have tried as much as possible for supporting me all the time. My dear mum and dad, I love you.

Dedicated to my parents.

Chapter 1

Introduction

Graph colouring involves the labeling of the vertices of some given graph by integers called colours such that no two adjacent vertices receive the same colour. The goal is to minimize the number of colours. The problem of graph colouring was first raised by Francis Guthrie while he was trying to color a map of the counties of England. Later in 1890, Heawood [50] proved that every planar map can be colored with no more than five colors. After then, a number of researchers tried to reduce the number of colours to four, until the Four Colouring Theorem was proved by Appel and Haken [1]. Graph colouring is one of the most fundamental concepts in both structural and algorithmic graph theory, which arises in a vast number of theoretical and practical applications such as classical settings like map colouring and job scheduling, as well as more recent settings like time slot allocation, aircraft scheduling, biprocessor tasks, frequency assignment and pattern matching. Many variants and generalisations of this concept have been studied over the years, and there are some very good surveys [85, 93] and books [14, 45, 60] on the subject. We survey *computational complexity* results of the graph colouring problem and its variants precolouring extension and list colouring when the input is *restricted to some special graph class*. Before presenting these results we first state the necessary terminology.

1.1 Graph Colouring Terminology

For the colouring problems that we consider self-loops, directed edges and multiple edges are irrelevant. Hence, we only consider finite undirected graphs with no multiple edges

and no self-loops, i.e., a *graph* G is an ordered pair (V, E) that consists of a finite set V of elements called *vertices* and a set E of unordered pairs uv with $u, v \in V$ called *edges*. The sets V and E are called the *vertex set* and *edge set* of G, respectively.

A *colouring* of a graph $G = (V, E)$ is a mapping from the vertex set of G to an infinite set of positive integers, i.e., $\phi : V \to \{1, 2, \ldots\}$, such that $\phi(u) \neq \phi(v)$ whenever $uv \in E$. We call $\phi(u)$ the *colour* of u. We let $\phi(U) = \{\phi(u) \mid u \in U\}$ for a subset $U \subseteq V$. A *k-colouring* of G is a colouring ϕ of G with $1 \leq \phi(u) \leq k$ for all $u \in V$. In that case we say that G is *k-colourable*. The problem Colouring is that of testing whether a given graph admits a k-colouring for some given integer k. If k is *fixed*, i.e., not part of the input, then we denote the problem as k-Colouring. The smallest integer k for which a graph G is k-colourable is called the *chromatic number* of G denoted $\chi(G)$. The Precolouring Extension problem takes as input a graph G, an integer k and a precolouring of G, i.e., a mapping $\phi_W : W \to \{1, 2, \ldots k\}$ for some subset $W \subseteq V$, and asks whether ϕ_W can be extended to a k-colouring of G. If k is fixed, we denote this problem as k-Precolouring Extension.

A *list assignment* of a graph $G = (V, E)$ is a function $\mathcal{L}$ that assigns a list $L(u)$ of so-called *admissible* colours to each $u \in V$. If $L(u) \subseteq \{1, \ldots, k\}$ for each $u \in V$, then $\mathcal{L}$ is also called a *k-list assignment*. Equivalently, $\mathcal{L}$ is a k-list assignment if $|\bigcup_{u \in V} L(u)| \leq k$. The *size* of a list assignment $\mathcal{L}$ is the maximum list size $|L(u)|$ over all vertices $u \in V$. We say that a colouring $\phi : V \to \{1, 2, \ldots\}$ *respects* $\mathcal{L}$ if $\phi(u) \in L(u)$ for all $u \in V$. The List Colouring problem is to test whether a given graph has a colouring that respects some given list assignment. For a fixed integer k, the List k-Colouring problem has as input a graph G with a k-list assignment $\mathcal{L}$ and asks whether G has a colouring that respects $\mathcal{L}$. For a fixed integer ℓ, the ℓ-List Colouring problem has as input a graph G with a list assignment $\mathcal{L}$ of size at most ℓ and asks whether G has a colouring that respects $\mathcal{L}$.

Note that k-Colouring can be viewed as a special case of k-Precolouring Extension by choosing $W = \emptyset$, and that k-Precolouring Extension can be viewed as a special case of List k-Colouring by choosing $L(u) = \{\phi_W(u)\}$ if $u \in W$ and $L(u) = \{1, \ldots, k\}$ if $u \in V \setminus W$. Finally, List k-Colouring can be readily seen as a special case of k-List Colouring. All these relationships are displayed in Figure 1.1.

In computational complexity theory, the class P contains all decision problems that are solvable in polynomial time, and the class NP contains all decision problems for which

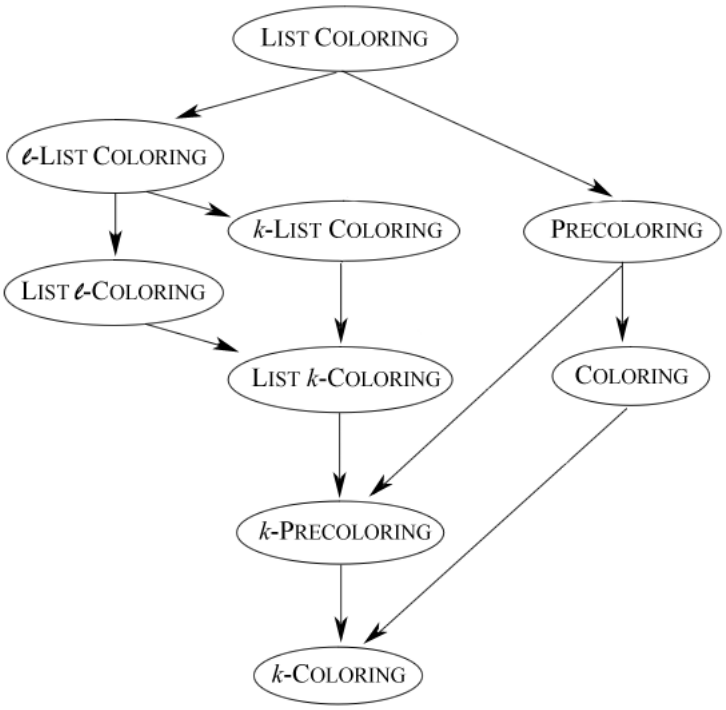

Figure 1.1: Relationships between COLOURING and its variants. An arrow from one problem to another problem indicates that the latter problem is a special case of the first one, whereas k and ℓ are any two integers for which $\ell \geq k$.

a candidate solution can be verified in polynomial time. Trivially $\mathsf{P} \subseteq \mathsf{NP}$ holds. It is widely believed that $\mathsf{P} \neq \mathsf{NP}$, but this problem is still open. A decision problem is called NP-complete if it is in NP and every other problem in NP can be transformed to this problem in polynomial time. Theorem 1.1.1 gives the computational complexity of the considered problems on general graphs.

Theorem 1.1.1. *For general graphs, k-COLOURING, k-PRECOLOURING EXTENSION, LIST k-COLOURING and k-LIST COLOURING are polynomial-time solvable if $k \leq 2$ and NP-complete if $k \geq 3$.*

Proof. It is well-known that k-COLOURING is NP-complete for all $k \geq 3$ (see [35]). Erdös, Rubin and Taylor [32] and Vizing [94] observed that 2-LIST COLOURING is polynomial-time solvable on general graphs. Then Theorem 1.1.1 follows from the relationships as displayed in Figure 1.1. □

Due to Theorem 1.1.1 one has put restrictions on the input graph. In this thesis we follow this line of research and only consider input graphs that belong to some special graph class. Before presenting the results known in the literature and our own results we first state the necessary definitions of these graph classes.

1.2 Definitions of Graph Classes and Other Terminology

In this section we give the definitions of a number of graph classes known in the literature. We first recall some basic graph notions, some of which we need in later chapters as well. We refer to the textbook of Diestel [29] for any undefined graph terminology.

The *union* of two graphs G and H is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H)$. If $V(G) \cap V(H) = \emptyset$, then we call the union of G and H the *disjoint union* of G and H denoted $G + H$. We denote the disjoint union of r copies of G by rG. The graph K_r denotes the *complete* graph on r vertices, i.e., $V(K_r) = \{u_1, \ldots, u_r\}$ and $E(K_r) = \{u_i u_j \mid 1 \leq i < j \leq r\}$. The vertex set of a complete graph is called a *clique*. The size of a largest clique in a graph G is called the *clique number* of G, denoted $\omega(G)$. For $r \geq 1$, the graph P_r denotes the *path* on r vertices, i.e., $V(P_r) = \{u_1, \ldots, u_r\}$ and $E(P_r) = \{u_i u_{i+1} \mid 1 \leq i \leq r-1\}$. For $r \geq 3$, the graph C_r denotes the *cycle* on r vertices, i.e., $V(C_r) = \{u_1, \ldots, u_r\}$ and $E(C_r) = \{u_i u_{i+1} \mid 1 \leq i \leq r-1\} \cup \{u_r u_1\}$. The *length* of a path or cycle is its number of edges. A cycle is *odd* if it has odd length. The graph rP_1 denotes the *independent set* on r vertices. The size of a largest independent set in a graph G is called the *independence number* of G, denoted $\alpha(G)$. For positive integers p and q, the *Ramsey number* $R(p, q)$ is the smallest number of vertices n such that all graphs on n vertices contain an independent set of size p or a clique of size q. Ramsey's Theorem (see e.g. [29]) states that such a number exists for all positive integers p and q.

Let $G = (V, E)$ be a graph. The *degree* $\deg_G(u)$ of a vertex u in G is the number of edges incident with it, or equivalently the size of its *neighbourhood* $N_G(u) = \{v \in V \mid uv \in E\}$. A vertex of degree 1 is also called a *pendant* vertex or a *leaf*. The minimum and maximum degree of G are denoted $\delta(G)$ and $\Delta(G)$, respectively. If all vertices are of the same degree, then G is said to be *regular*. If all vertices have degree equal to p for some integer p, then G is also called *p-regular*. A path between two vertices u and v is also called a *(u, v)-path*. The *distance* $\text{dist}_G(u, v)$ between two vertices u and v in G is the length of a shortest (u, v)-path. The maximum distance in a graph G is called the *diameter* of G denoted $\text{diam}_G = \max\{\text{dist}_G(u, v) \mid u, v \in V\}$. A graph H is

a *subgraph* of a graph G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. For a subset $S \subseteq V(G)$, we let $G[S]$ denote the *induced* subgraph of G, i.e., that has vertex set S and edge set $\{uv \in E(G) \mid u, v \in S\}$. For a graph H, we write $H \subseteq G$ and $H \subseteq_i G$ to denote that H is a subgraph or an induced subgraph of G, respectively. We say that G is *connected* if for every pair of distinct vertices u and v, there exists a path connecting u and v, whereas G is *k-connected* for some integer $k \geq 1$ if $G[V \setminus U]$ is connected for every subset $U \subseteq V$ of at most $k-1$ vertices. If a graph is not connected, then it is called *disconnected*. A maximal connected subgraph of G is called a *connected component*. The *girth* of G is the length of a shortest cycle in G; if G has no cycles, then the girth of G is equal to ∞. The *complement* of G denoted by $\overline{G}$ has vertex set V and an edge between two distinct vertices if and only if these vertices are not adjacent in G. A set $D \subseteq V(G)$ *dominates* G if each vertex of G is either in D or adjacent to some vertex in D. In that case, D is called a *dominating set* of G. If $D = \{u\}$, then we call u a *dominating vertex* of G.

For two graphs G and H, a vertex mapping $f : V(G) \to V(H)$ is called a (*graph*) *isomorphism* when $uv \in E(G)$ if and only if $f(u)f(v) \in E(H)$. In that case we say that G and H are *isomorphic*. Let G be a graph and $\{H_1, \ldots, H_p\}$ be a set of graphs. We say that G is *$(H_1, \ldots, H_p)$-free* if G has no induced subgraph isomorphic to a graph in $\{H_1, \ldots, H_p\}$; if $p = 1$, we may write H_1-free instead of (H_1)-free.

Proposition 1.2.1. *Let H' be an induced subgraph of a graph H. Then every H'-free graph is H-free.*

The following preprocessing technique is often used in graph colouring. Let $G = (V, E)$ be a graph with a k-list assignment $\mathcal{L}$ for some integer $k \geq 1$. Let $W \subseteq V$ be the set of those vertices of G whose lists have size at most 2. Remove all vertices of $V \setminus W$ with degree at most $k-1$ from G. Iterate this until we obtain a graph $G^*_{\geq k}$ such that all vertices not in W have degree at least k. Note that $G^*_{\geq k}$ may be the empty graph. We observe the following.

Proposition 1.2.2. *Let $G = (V, E)$ be a graph with a k-list assignment $\mathcal{L}$ for some integer $k \geq 1$. Let $W \subseteq V$ be the set of those vertices of G whose lists have size at most 2. Then $G^*_{\geq k}$ can be obtained in polynomial time, and $G^*_{\geq k}$ has a colouring that respects the restriction of $\mathcal{L}$ to $V(G^*_{\geq k})$ if and only if G has a colouring that respects $\mathcal{L}$. Moreover, for any set of graphs $\{H_1, \ldots, H_p\}$, $G^*_{\geq k}$ is $(H_1, \ldots, H_p)$-free if G is $(H_1, \ldots, H_p)$-free.*

Proof. Because we must search for at most $|V|$ times for a vertex of degree at most

$k-1$, the graph $G^*_{\geq k}$ can be obtained in polynomial time. Let $(H_1, \ldots, H_p)$ be a set of graphs. Because we only remove vertices in order to get $G^*_{\geq k}$, we find that $G^*_{\geq k}$ is $(H_1, \ldots, H_p)$-free if G is $(H_1, \ldots, H_p)$-free.

We now prove the remaining statement. If G has a colouring that respects $\mathcal{L}$, then $G^*_{\geq k}$ has a colouring that respects the restriction of $\mathcal{L}$ to $V(G^*_{\geq k})$, because $G^*_{\geq k}$ is a subgraph of G. To prove the reverse implication, suppose that $G^*_{\geq k}$ has a colouring that respects the restriction of $\mathcal{L}$ to $V(G^*_{\geq k})$. By considering the vertices of $V \backslash V(G^*_{\geq k})$ in the reverse order of their removal, there is always a colour from $\{1, \ldots, k\}$ available to colour them. □

We will now give the definitions of a number of graph classes that have been studied as regards the Colouring problem and the variants we consider. Most of these graph classes can be found in textbooks on special graph classes such as the textbooks of Brandstädt, Le and Spinrad [14] and Golumbic [45]. As we shall see, most (but not all) of them can be characterised by one or more forbidden induced subgraphs.

An *asteroidal triple* in a graph is a set of three mutually non-adjacent vertices such that each pair of them are joined by a path that avoids the neighbourhood of the third including the third vertex itself, and *AT-free graphs* are exactly those graphs that contain no such triple. We refer to the Ph.D. Thesis of Koehler [66] for a characterisation of AT-free graphs in terms of forbidden induced subgraphs.

An *anticycle* is the complement of a cycle. Anticycles are also called co-cycles. A graph is *odd-(anti)cycle-free* if it contains no induced odd (anti)cycle. A graph is a *tree* if it is connected and cycle-free. A graph is a *forest* if each of its connected components is a tree. Equivalently, a forest is a graph with girth $g = \infty$. A graph is a *linear forest* if each of its connected components is a path. A graph is *bipartite* if its vertex set can be partitioned into two sets A and B such that every edge has one of its end-vertices in A and the other one in B. Equivalently, a graph is bipartite if and only if it is odd-cycle-free. The complement of a bipartite graph is called a *co-bipartite* graph. Hence, a graph is a co-bipartite graph if and only if it is odd-anticycle-free. For $r \geq 1$ and $s \geq 1$, the graph $K_{r,s}$ denotes the *complete bipartite* graph with partition classes of size r and s, respectively, i.e., $V(K_{r,s}) = \{u_1, \ldots, u_r\} \cup \{v_1, \ldots, v_s\}$ and $E(K_{r,s}) = \{u_i v_j \mid 1 \leq i \leq r \text{ and } 1 \leq j \leq s\}$. For $r \geq 1$, the graph $K_{1,r}$ is also called a *star*. In particular, the graph $K_{1,3}$ is called a *claw*, and a $K_{1,3}$-free graph is called *claw-free*. Equivalently, a graph is a complete bipartite graph if and only if it is bipartite, connected and $(P_1 + P_2)$-free.

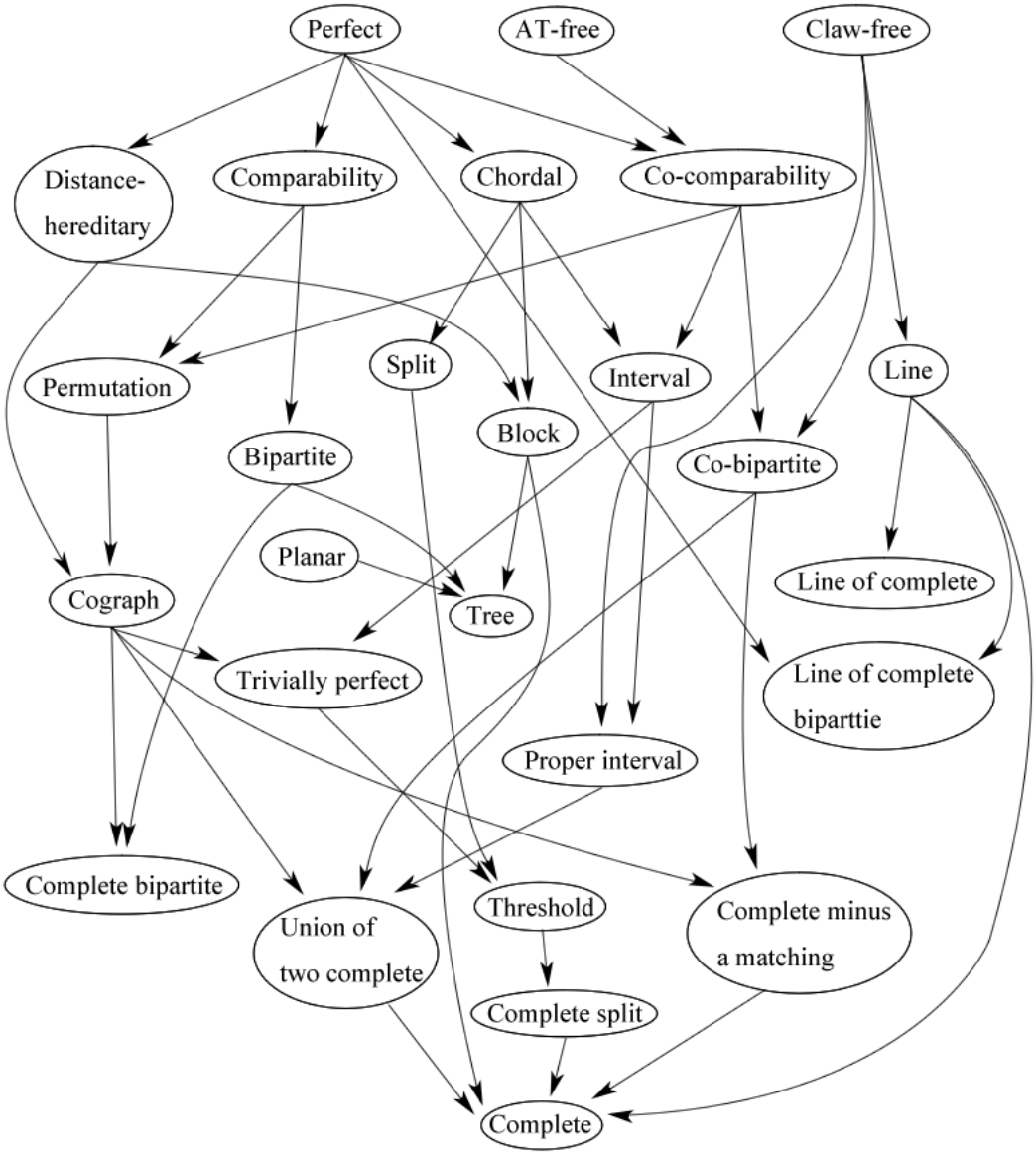

Figure 1.2: Relationships between graph classes; an arrow from one graph class to another one is to indicate that the second graph class is a proper subclass of the first one. All graph classes displayed in this figure can be found in Table 1.3 showed in Section 1.6.

A *hole* in a graph is an induced cycle of length at least 5, and an *antihole* is the complement of a hole. A graph is *odd-(anti)hole-free* if it contains no odd (anti)hole. In the literature, an induced C_4 is sometimes considered to be a hole as well, but this is

not the case under our definition (which is from [14]). A C_3-free graph is also called a *triangle-free* graph, and a P_4-free graph is also called a *cograph*. Cographs and bipartite graphs are examples of perfect graphs. A graph is *perfect* if the chromatic number of every induced subgraph equals the size of a largest clique in that subgraph. By the Strong Perfect Graph Theorem [22], a graph is perfect if and only if it is odd-hole-free and odd-antihole-free.

A *matching* in a graph is a set of pairwise non-adjacent edges, i.e., no two edges have a common end-vertex. A matching is *perfect* if every vertex of the graph is incident to exactly one edge of the matching. A *complete graph minus a matching* is obtained from a complete graph K_n after removing the edges of some (possibly empty and not necessarily perfect) matching M. Equivalently, a graph G on n vertices is a complete graph minus a matching if and only if G is $(3P_1, P_1 + P_2)$-free if and only if G has minimum degree at least $n - 2$ (4.2).

A graph G is *distance-hereditary* if $\text{dist}_{G'}(u, v) = \text{dist}_G(u, v)$ for all induced subgraphs $G' \subseteq_i G$ and all pairs of vertices $u, v \in V(G')$. Equivalently, a graph is distance-hereditary if and only if it contains no hole, no induced domino (the graph obtained from a cycle with vertices a, b, c, d, e, f by adding the edge cf), no induced gem (the graph obtained from a P_4 by adding a dominating vertex) and no induced house (the graph isomorphic to $\overline{P}_5$) [4].

A graph is a *comparability graph* if there exists an assignment of exactly one direction to each of its edges such that (a, c) is a directed edge whenever (a, b) and (b, c) are directed edges. We refer to Gallai [34] for a characterisation in terms of forbidden induced subgraphs. The complement of a comparability graph is called a *co-comparability graph*. The class of co-comparability graph is properly contained in the class of AT-free graphs (see [24]). A graph is a *permutation graph* if line segments connecting two parallel lines can be associated to its vertices in such a way that two vertices are adjacent if and only if their line segments intersect. Equivalently, a graph is a permutation graph if and only if it is a comparability and a co-comparability graph [30].

A *chord* of a cycle C is an edge between two vertices $u, v \in V(C)$ with $uv \notin E(C)$. A graph is *chordal* if every induced cycle on four or more vertices has a chord. Equivalently, a graph is chordal if and only if it contains no induced cycle on four or more vertices. Chordal graphs are also called triangulated graphs. A graph is a *block graph* if it is connected and every maximal 2-connected component is a complete graph. The *diamond*

denoted K_4^- is the graph K_4 minus an edge. Then a graph is a block graph if and only if it is chordal and has no induced diamond [4].

A graph G is a *split* graph if its vertices can be partitioned into a clique and an independent set; if every vertex in the independent set is adjacent to every vertex in the clique, then G is a *complete* split graph. Split graphs coincide with $(2P_2, C_4, C_5)$-free graphs [33]. A graph G is a *threshold graph* if there exists a real number t (the threshold) together with real weights w_v associated to each vertex $v \in V(G)$ such that for any two vertices u and v, $uv \in E(G)$ if and only if $w_u + w_v \geq t$. Equivalently, a graph is a threshold graph if and only if it is $(2P_2, C_4, P_4)$-free [21]. A clique in a graph G is *maximal* if it is not properly contained in some other clique of G. A graph is a *trivially perfect graph* if for every induced subgraph the size of a maximum independent set in that subgraph equals its number of maximal cliques. A graph is a trivially perfect graph if and only if it is (C_4, P_4)-free [98].

A graph is a (not necessarily disjoint) *union of two complete graphs* if and only if it is $(3P_1, P_4, C_4)$-free. Equivalently, a graph G is the complement of a union of two complete graphs if and only if it is $(C_3, P_4, 2P_2)$-free. This can be seen as follows. First, suppose that G is the complement of a union of two complete graphs. Then G is a disjoint union of a complete bipartite graph F and a (possibly empty) independent set. Therefore, G is $(C_3, P_4, 2P_2)$-free. Now suppose that G is $(C_3, P_4, 2P_2)$-free. Due to the $2P_2$-freeness, G contains at most one connected component F with more than one edge. Due to the (C_3, P_4)-freeness, F must be a bipartite graph with bipartition classes A and B. Since F is connected and P_4-free, every vertex in A is adjacent to every vertex in B. Hence, G is a disjoint union of a complete bipartite graph and a (possibly empty) independent set, which in turn shows that G is the complement of a union of two complete graphs.

A graph is an *interval graph* if intervals of the real line can be associated with its vertices in such a way that two vertices are adjacent if and only if their corresponding intervals overlap. Equivalently, a graph is interval if and only if it is AT-free and chordal [76]. An interval graph is *proper* if it has an interval representation in which no interval is properly contained in any other interval. Equivalently, a graph is a proper interval if and only if it is interval and claw-free [96]. Proper interval graphs are also known as linear interval graphs and as indifference graphs. Moreover, the class of proper interval graphs coincides with the class of unit interval graphs, which are interval graphs that have an interval representation in which all intervals have the same length [87].

The *line graph* of a graph G with edges $e_1, \ldots, e_p$ is the graph $\texttt{line}(G)$ with vertices $u_1, \ldots, u_p$ such that there is an edge between any two vertices u_i and u_j if and only if e_i and e_j share an end-vertex in G. Beineke [5] gave a list of nine forbidden induced subgraphs that characterize line graphs. A line graph may not be perfect. However, a line graph of a bipartite graph is well-known to be perfect; this can be seen by using the Strong Perfect Graph Theorem after observing that an odd cycle on at least five vertices in a line graph corresponds to such a cycle in the original graph and every odd antihole contains the graph C_5 as a subgraph. Line graph of complete graphs are also called triangular graphs.

We say that we *identify* two vertices u and v in a graph G if we remove both vertices and replace them by a new vertex adjacent to precisely those vertices to which u and v were adjacent. If u and v were neighbours in G, then this operation is also called an *edge contraction*. A graph G contains a graph H as a *minor* if it can be modified into H by a sequence of edge contractions, edge deletions and vertex deletions. A graph is *planar* if it can be drawn in the plane so that its edges intersect only at their end-vertices. By Kuratowski's Theorem [74], a graph is planar if and only if it contains neither K_5 nor $K_{3,3}$ as a minor.

Figure 1.2 shows the relationships between the various graph classes. In this figure, an arrow from one graph class to another one is used to indicate that the first graph class is a proper superclass of the second one. We refer to the textbook of Brandstädt, Le and Spinrad [14] where these inclusions are shown for a number of perfect graphs except for the following classes: block graphs, co-bipartite graphs, complete graphs, complete bipartite graphs, complete split graphs, complete graphs minus a matching, line graphs of complete bipartite graphs, and unions of two complete graphs. The arrows entering and leaving these graph classes in Figure 1.2 can be readily seen from the definitions of the corresponding graph classes and their characterisations by forbidden induced subgraphs. The same holds for the remaining graph classes, which are the classes of AT-free graphs, claw-free graphs, line graphs, line graphs of complete graphs, and planar graphs.

We finish this section by giving the definitions of the basic NP-complete problems that we use for our NP-hardness reductions. The problem Satisfiability is the problem of determining if the variables of a given Boolean formula in conjunctive normal form can be assigned in such a way as to make the formula evaluate to `TRUE`. Karp [63] proved that the problem is NP-complete even if there are three variables per clause. In that case

the problem is called the 3-Satisfiability problem. A variant of the 3-Satisfiability problem is called Not-All-Equal 3-Satisfiability with un-negated literals only. This NP-complete problem [88] is also known as Hypergraph 2-Colourability and is defined as follows. Given a set $X = \{x_1, x_2, \ldots, x_n\}$ of logical variables, and a set $C = \{C_1, C_2, \ldots, C_m\}$ of three-literal clauses over X in which all literals are un-negated, does there exist a truth assignment for X such that each clause contains at least one true literal and at least one false literal?

1.3 Results for H-free Graphs

Recall that cographs, triangle-free graphs and claw-free graphs are P_4-free graphs, C_3-free graphs and $K_{1,3}$-free graphs, respectively. In this section we focus on these and other graph classes characterised by forbidden induced subgraphs.

Král' et al. [69] completely determined the computational complexity of Colouring for graph classes characterised by one forbidden induced subgraph. We show similar dichotomy results for the problems Precolouring Extension, List Colouring and k-List Colouring in Chapter 3 and Chapter 4. We summarize these results in the following theorem. Note that the first part of statement (iv) is also given in Theorem 1.1.1.

Theorem 1.3.1. *Let H be a fixed graph. Then the following four statements hold.*

(i) Colouring *is polynomial-time solvable for H-free graphs if H is an induced subgraph of P_4 or of $P_1 + P_3$; otherwise it is* NP*-complete for H-free graphs [69].*

(ii) Precolouring Extension *is polynomial-time solvable for H-free graphs if H is an induced subgraph of P_4 or of $P_1 + P_3$; otherwise it is* NP*-complete for H-free graphs.*

(iii) List Colouring *is polynomial-time solvable for H-free graphs if H is an induced subgraph of P_3; otherwise it is* NP*-complete for H-free graphs.*

*(iv) For all $\ell \leq 2$, ℓ-*List Colouring *is polynomial-time solvable. For all $\ell \geq 3$, ℓ-*List Colouring *is polynomial-time solvable for H-free graphs if H is an induced subgraph of P_3; otherwise it is* NP*-complete for H-free graphs.*

Due to Theorem 1.3.1 we can from now on assume that the upper bound on the number of available colours is fixed. In the literature many papers have appeared that considered the complexity of k-Colouring, k-Precolouring Extension and List k-Colouring for H-free graphs. Below we survey these. We start with the following two theorems. The first one is due to Král' et al. [69] but has also been proven by Kamiński and Lozin [61]. The second one is due to Holyer [54], who settled the case $k = 3$, and Leven and Galil [77] who settled the case $k \geq 4$.

Theorem 1.3.2. *For all $k \geq 3$ and all $g \geq 3$, the k-Colouring problem is NP-complete for graphs of girth at least g.*

Theorem 1.3.3. *For all $k \geq 3$, the k-Colouring problem is NP-complete for line graphs of k-regular graphs.*

Theorem 1.3.2 implies that for any $k \geq 3$, k-Colouring is NP-complete for the class of H-free graphs whenever H contains a cycle. Because line graphs are claw-free, Theorem 1.3.3 implies that for all $k \geq 3$, the k-Colouring is NP-complete on H-free graphs whenever H is a forest with a vertex of degree at least 3. This means that only the case in which H is a linear forest remains. We consider this case below.

Combining the result from Balas and Yu [3] on the number of maximal independent sets in an sP_2-free graph and a result from Tsukiyama et al. [92] on the enumeration of such sets leads to the known result that List k-Colouring is polynomial-time solvable on sP_2-free graphs for any two integers k and s. We extended this result by showing that List 3-Colouring is polynomial-time solvable on sP_3-free graphs for all $s \geq 1$ (4.4). We also showed that List 3-Colouring is polynomial-time solvable on $(P_2 + P_4)$-free graphs in the same paper. In addition, we made the following proposition.

Proposition 1.3.4. *Let H be a graph. If List 3-Colouring is solvable in polynomial time for H-free graphs, then it is also solvable in polynomial time for $(H+P_1)$-free graphs.*

Proof. Let G be an $(H + P_1)$-free graph with a 3-list assignment $\mathcal{L}$. If G is H-free, we are done. Suppose that G contains an induced subgraph H' that is isomorphic to H. Because G is $(H + P_1)$-free, every vertex in $V(G)\backslash V(H')$ must be adjacent to a vertex in H'. We guess a colouring of $V(H')$ that respects the restriction of $\mathcal{L}$ to H'. Afterwards we apply Theorem 1.1.1. Since H' has a fixed size, the number of guesses is polynomially bounded. □

We proved that 4-Precolouring Extension is polynomial-time solvable on $(P_2 + P_3)$-free graphs (2.1). Hoàng et al. [53] proved that for all $k \geq 1$, List k-Colouring is polynomial-time solvable on P_5-free graphs. This was generalised by Couturier et al. [27] who showed that for all $k \geq 1$ and all $s \geq 0$, List k-Colouring is polynomial-time solvable on $(sP_1 + P_5)$-free graphs. Couterier et al. [27] also showed that their result is tight by proving that List k-Colouring is NP-complete for some integer k on H-free graphs, whenever H is a supergraph of $P_1 + P_5$ with at least five edges. In particular, they proved that List 5-Colouring is NP-complete on $(P_2 + P_4)$-free graphs.

Randerath and Schiermeyer [83] showed that 3-Colouring is polynomial-time solvable on P_6-free graphs. This was generalised by Broersma et al. [16] who showed that List 3-Colouring is polynomial-time solvable for P_6-free graphs. For P_6-free graphs it is also known that 5-Precolouring Extension is NP-complete [16] and that List 4-Colouring is NP-complete (4.5). For P_7-free graphs it is known that 6-Colouring is NP-complete [16] and that 4-Precolouring Extension is NP-complete (3.3). Woeginger and Sgall [97] proved that 5-Colouring is NP-complete for P_8-free graphs and that 4-Colouring is NP-complete for P_{12}-free graphs. The latter result was improved by Le, Randerath and Schiermeyer [75], who showed that 4-Colouring is NP-complete for P_9-free graphs. We improved this further by showing that 4-Colouring is NP-complete for P_8-free graphs (2.2).

Taking into account the relationships displayed in Figure 1.1 showed in Section 1.1, we summarize all the results in Theorem 1.3.5. Note that the first part of statement (iv) is also given in Theorem 1.1.1.

Theorem 1.3.5. *Let H be a fixed graph. Then the following five statements hold:*

(i) k-Colouring is NP-complete for H-free graphs if

- *$k \geq 3$ and $H \supseteq_i C_r$ for some $r \geq 3$ [69, 61]*
- *$k \geq 3$ and $H \supseteq_i K_{1,3}$ [54, 77]*
- *$k \geq 4$ and $H \supseteq_i P_8$*
- *$k \geq 6$ and $H \supseteq_i P_7$ [16].*

(ii) k-Precolouring Extension is NP-complete for H-free graphs if

- *$k \geq 4$ and $H \supseteq_i P_7$*

- $k \geq 5$ *and* $H \supseteq_i P_6$ *[16].*

(iii) List k-Colouring *is* NP*-complete for H-free graphs if*

- $k \geq 4$ *and* $H \supseteq_i P_6$
- $k \geq 5$ *and* $H \supseteq_i P_2 + P_4$ *[27]*

(iv) List k-Colouring *is polynomially solvable for H-free graphs if* $k \leq 2$ *or*

- $k \leq 3$ *and* $H \subseteq_i sP_1 + P_2 + P_4$ *for some* $s \geq 0$
- $k \leq 3$ *and* $H \subseteq_i sP_1 + P_6$ *for some* $s \geq 0$ *[16]*
- $k \leq 3$ *and* $H \subseteq_i sP_3$ *for some* $s \geq 0$
- $k \geq 1$ *and* $H \subseteq_i sP_1 + P_5$ *for some* $s \geq 0$ *[27]*
- $k \geq 1$ *and* $H \subseteq_i sP_2$ *for some* $s \geq 0$ *[3, 92].*

(v) 4-Precolouring Extension *is polynomially solvable for H-free graphs if*

- $H \subseteq_i P_2 + P_3$.

As a consequence of Theorem 1.3.5 we obtain dichotomy results for k-Colouring, k-Precolouring Extension and List k-Colouring when H is small. This is shown in Theorem 1.3.6.

Theorem 1.3.6. *The following three statements hold:*

(i) For any graph H on at most six vertices, 3-Colouring*,* 3-Precolouring Extension*, and* List 3-Colouring *are polynomial-time solvable on H-free graphs if H is a linear forest, and* NP*-complete otherwise.*

(ii) For any graph H on at most five vertices, 4-Colouring*, and* 4-Precolouring Extension *are polynomial-time solvable on H-free graphs if H is a linear forest, and* NP*-complete otherwise.*

(iii) For any graph H on at most four vertices and any integer $k \geq 3$, k-Colouring, k-Precolouring Extension, and List k-Colouring *are polynomial-time solvable on H-free graphs whenever H is a linear forest, and* NP*-complete otherwise.*

Note that statement (ii) of Theorem 1.3.6 is not valid for List 4-Colouring due to exactly one missing case, which is the complexity of List 4-Colouring for (P_2+P_3)-free graphs.

Table 1.1 shows a summary of the existing results for P_r-free graphs. This table is obtained from Theorem 1.3.5 as well. We include this table, because it is the most updated version of similar tables from other papers [19, 53, 75, 85, 97].

	k-Colouring				k-Precolouring Extension				List k-Colouring			
r	$k=3$	$k=4$	$k=5$	$k\geq 6$	$k=3$	$k=4$	$k=5$	$k\geq 6$	$k=3$	$k=4$	$k=5$	$k\geq 6$
$r\leq 5$	P	P	P	P	P	P	P	P	P	P	P	P
$r=6$	P	?	?	?	P	?	NP-c	NP-c	P	NP-c	NP-c	NP-c
$r=7$	?	?	?	NP-c	?	NP-c	NP-c	NP-c	?	NP-c	NP-c	NP-c
$r\geq 8$	?	NP-c	NP-c	NP-c	?	NP-c	NP-c	NP-c	?	NP-c	NP-c	NP-c

Table 1.1: The complexity of k-Colouring, k-Precolouring Extension and List k-Colouring on P_r-free graphs for fixed k and r.

1.4 Results for (H_1, H_2)-free Graphs

When we forbid two induced subgraphs, only partial results are known. We survey these below and also mention a number of results for $(H_1, \ldots, H_p)$-free graphs for $p \geq 3$.

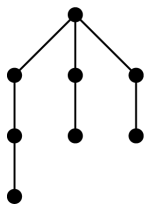

Figure 1.3: The graph $S_{2,2,3}$

For $1 \leq h \leq i \leq j$, let $S_{h,i,j}$ denote the tree with only one vertex x of degree 3, and the tree has exactly three leaves, which are of distance h, i and j to x, respectively. For example, Figure 1.3 shows the graph $S_{2,2,3}$. Schindl [89] showed the following result. Following his notation, we let $A_{h,i,j}$ denote the line graph of $S_{h,i,j}$.

Theorem 1.4.1 ([89]). *Let $\{H_1, \ldots, H_p\}$ be a finite set of graphs. The* Colouring *problem is* NP*-complete for $(H_1, \ldots, H_p)$-free graphs if the complement of each H_i has a*

connected component isomorphic neither to any graph $A_{h,i,j}$ for $1 \leq h \leq i \leq j$ nor to any path P_r for $r \geq 1$.

Theorem 1.4.1 has several other interesting consequences. For instance, Schindl [89] showed that it implies that Colouring is NP-complete for graphs with no odd hole C_r and no odd antihole $\overline{C_r}$ on at most r vertices for any fixed $r \leq 7$, whereas Colouring is well-known to be polynomial-time solvable on perfect graphs, which are precisely those graphs with no odd hole and no odd antihole [22].

By Theorem 1.3.2, Colouring is NP-complete for $(H_1, \ldots, H_p)$-free graphs if every H_i contains an induced cycle. Král' et al. [69] proved that the Colouring problem is NP-complete for $(K_4, K_4^-, K_{1,3})$-free graphs and for $(H_1, \ldots, H_p)$-free graphs if either every H_i contains an induced claw, or every H_i contains some induced subgraph that is a spanning subgraph of $2P_2$. Dabrowski et al. [28] proved that Colouring is polynomial-time solvable on $(C_3, S_{1,2,3}, S_{1,1,2} + P_2)$-free graphs. Král' et al. [69] showed that for any fixed graph H_2, the Colouring problem is polynomial-time solvable on (C_3, H_2)-free graphs if and only if it is so for (C_3^+, H_2)-free graphs. Here, C_3^+ denotes the graph obtained from C_3 after adding one pendant vertex. Maffray and Preismann [78] showed that 3-Colouring is NP-complete even on $(C_3, K_{1,5})$-free graphs. We showed that 4-Colouring is NP-complete for (C_3, P_{164})-free graphs (2.3). Randerath [82] showed that the Colouring problem is polynomial-time solvable for $(C_3, S_{1,1,2})$-free graphs and for (C_3, H^L)-free graphs, where H^L is the "letter-H" graph, i.e., the graph obtained from a claw by adding two pendant vertices to exactly one of its leaves. Randerath and Schiermeyer [84] showed that Colouring is polynomial-time solvable on (C_3, cross)-free graphs, where the *cross* is the graph obtained from the graph $K_{1,4}$ by adding one pendant vertex to one of its leaves. We showed that Colouring can be solved in polynomial time for $(C_3, 2P_3)$-free graphs [17]. In particular, we showed that every $(C_3, 2P_3)$-free graph can be coloured by at most five colours. Recently, Pyatkin [81] improved this result by showing that every $(C_3, 2P_3)$-free graph is 4-colourable. We also showed that Colouring can be solved in polynomial time for $(C_3, P_2 + P_4)$-free graphs in Section 4.3. By combining the above results from [17, 19, 78, 82, 84] with some new results, Dabrowski et al. [28] showed the following result (also see [17]).

Theorem 1.4.2. *Let H be a fixed graph on at most six vertices. Then* Colouring *is polynomial-time solvable for (C_3, H)-free graphs if H is a forest not isomorphic to $K_{1,5}$, and otherwise it is* NP*-complete.*

Golovach and Paulusma [39] made the following proposition, which is based on the fact that for all $r, t \geq 1$, (K_r, P_t)-free graphs can be coloured with at most $(t-1)^{r-2}$ colours [49].

Proposition 1.4.3. Colouring *is polynomial-time solvable on* (K_r, F)*-free graphs for some linear forest* F *if* k-Colouring *is polynomial-time solvable on* F*-free graphs for all* $k \geq 1$.

Then by combining Proposition 1.4.3 with Theorem 1.3.5 one finds that Colouring is polynomial-time solvable on $(K_r, sP_1 + P_5)$-free graphs and on (K_r, sP_2)-free graphs for all $r \geq 1$ and $s \geq 0$. Using the results above combined with some other results on graph classes of bounded clique-width [11, 12, 13], Golovach and Paulusma [39] gave the following summary for the case when exactly two induced graphs H_1 and H_2 are forbidden. Recall that C_3^+ denotes the graph with vertices a, b, c, d and edges ab, ac, ad, bc, whereas F_5 denotes the 5-vertex fan also called the gem, which we defined as the graph with vertices a, b, c, d, e and edges $ab, bc, cd, ea, eb, ec, ed$.

Theorem 1.4.4. *Let* H_1 *and* H_2 *be two fixed graphs. Then the following holds:*

(i) Colouring *is* NP*-complete for* (H_1, H_2)*-free graphs if*

1. $H_1 \supseteq_i C_r$ *for some* $r \geq 3$ *and* $H_2 \supseteq_i C_s$ *for some* $s \geq 3$
2. $H_1 \supseteq_i K_{1,3}$ *and* $H_2 \supseteq_i K_{1,3}$
3. H_1 *and* H_2 *contain a spanning subgraph of* $2P_2$ *as an induced subgraph*
4. $H_1 \supseteq_i C_3$ *and* $H_2 \supseteq_i K_{1,r}$ *for some* $r \geq 5$
5. $H_1 \supseteq_i C_3$ *and* $H_2 \supseteq_i P_{164}$
6. $H_1 \supseteq_i C_r$ *for* $r \geq 4$ *and* $H_2 \supseteq_i K_{1,3}$
7. $H_1 \supseteq_i C_r$ *for* $r \geq 5$ *and* H_2 *contains a spanning subgraph of* $2P_2$ *as an induced subgraph*
8. $H_1 \supseteq_i K_4$ *or* $H_1 \supseteq_i K_4^-$, *and* $H_2 \supseteq_i K_{1,3}$
9. $H_1 \supseteq_i C_r + P_1$ *for* $3 \leq r \leq 4$ *or* $H_1 \supseteq_i \overline{C_r}$ *for* $r \geq 6$, *and* H_2 *contains a spanning subgraph of* $2P_2$ *as an induced subgraph.*

(ii) Colouring *is polynomial-time solvable for* (H_1, H_2)*-free graphs if*

1. *H_1 or H_2 is an induced subgraph of $P_1 + P_3$ or of P_4*
2. *$H_1 \subseteq_i C_3 + P_1$ and $H_2 \subseteq_i K_{1,3}$*
3. *$H_1 \subseteq_i C_3^+$, and $H_2 \subseteq_i K_{1,r}$ for $r \leq 4$*
4. *$H_1 \subseteq_i C_3^+$ and $H_2 \neq K_{1,5}$ is a forest on at most six vertices*
5. *$H_1 \subseteq_i C_3^+$, and $H_2 \subseteq_i sP_2$ or $H_2 \subseteq_i sP_1 + P_5$ for $s \geq 0$*
6. *$H_1 = K_r$ for $r \geq 4$, and $H_2 \subseteq_i sP_2$ or $H_2 \subseteq_i sP_1 + P_5$ for $s \geq 0$*
7. *$H_1 \subseteq_i F_5$ or $H_1 \subseteq_i \overline{P_5}$, and $H_2 \subseteq_i P_1 + P_4$*
8. *$H_1 \subseteq_i F_5$ and $H_2 \subseteq_i P_5$*

The NP-completeness results from Theorem 1.4.4 carry over to the Precolouring Extension problem. Other than those results, not many additional results seem to be known for the Precolouring Extension problem on (H_1, H_2)-free graphs. Both Hujter and Tuza [57] and Jansen and Scheffler [59] showed that Precolouring Extension can be solved in polynomial time on P_4-free graphs. Hence, Precolouring Extension can be solved in polynomial-time on (H_1, H_2)-free graphs whenever $H_1 \subseteq_i P_4$ or $H_2 \subseteq_i P_4$.

Golovach and Paulusma [39] also completely classified the complexity of List Colouring and ℓ-List Colouring ($\ell \geq 3$) for (H_1, H_2)-free graphs.

Theorem 1.4.5 ([39])**.** *Let H_1 and H_2 be two fixed graphs. Then* List Colouring *is polynomial-time solvable for (H_1, H_2)-free graphs in the following cases:*

1. *$H_1 \subseteq_i P_3$ or $H_2 \subseteq_i P_3$*
2. *$H_1 \subseteq_i C_3$ and $H_2 \subseteq_i K_{1,3}$*
3. *$H_1 = K_r$ for some $r \geq 3$ and $H_2 = sP_1$ for some $s \geq 3$.*

In all other cases, even 3-List Colouring *is* NP*-complete for (H_1, H_2)-free graphs.*

Not so many results are known for k-Colouring on (H_1, H_2)-free graphs. We only mention the following. We showed that for all integers $k, r, s, t \geq 1$, the List k-Colouring problem, and thus, the k-Colouring problem is polynomial-time solvable on $(K_{s,t}, P_r)$-free graphs (4.6). By taking $s = t = 2$, this means that for all integers $k, r \geq 1$, the k-Colouring problem is polynomial-time solvable on (C_4, P_r)-free graphs. Consequently, for all integers $g \geq 5$ and $k, r \geq 1$, the k-Colouring problem is

polynomial-time solvable on P_r-free graphs of girth at least g. This result is best possible with respect to the girth due to our result of that 4-Colouring is NP-complete on (C_3, P_{164})-free graphs (2.3).

1.5 Results for Bounded Graph Parameters

In Table 1.2 we summarize known results on colouring when some graph parameter is bounded. We will apply some of these results in later chapters.

Graph parameter	Col	Precol	List Col	k-Col	k-Precol	List k-Col	k-List Col
All graphs	NP-c	NP-c	NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Treewidth $\leq p$ $(p \geq 1)$	P	P	P	P	P	P	P
Clique-width ≤ 1	P	P	P	P	P	P	P
Clique-width ≤ 2	P	P	NP-c	P	P	P	$k \geq 3$:NP-c
Clique-width $\leq p$ $(p \geq 3)$	P	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Degree ≤ 2	P	P	P	P	P	P	P
Degree ≤ 3	P	P	NP-c	P	P	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Degree ≤ 4	NP-c	NP-c	NP-c	$k = 3$:NP-c $k = 4$: ? $k \geq 5$:YES	$k = 3$:NP-c $k = 4$: ? $k \geq 5$:YES	$k = 3$:NP-c $k = 4$:NP-c $k \geq 5$:NP-c	$k = 3$:NP-c $k = 4$:NP-c $k \geq 5$:NP-c
Girth $\geq p$ $(p \geq 3)$	NP-c	NP-c	NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Clique number ≤ 1	P	P	P	P	P	P	P
Clique number $\leq p$ $(p \geq 2)$	NP-c	NP-c	NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Independence number ≤ 1	P	P	P	P	P	P	P
Independence number ≤ 2	P	P	NP-c	P	P	P	$k \geq 3$:NP-c
Independence number $\leq p$ $(p \geq 3)$	NP-c	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Diameter ≤ 1	P	P	P	P	P	P	P
Diameter ≤ 2	NP-c	NP-c	NP-c	$k = 3$: ? $k \geq 4$:NP-c	$k = 3$: ? $k \geq 4$:NP-c	$k = 3$: ? $k \geq 4$:NP-c	$k = 3$: ? $k \geq 4$:NP-c
Diameter $\leq p$ $(p \geq 3)$	NP-c	NP-c	NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Asteroidal number ≤ 1	P	P	P	P	P	P	P
Asteroidal number ≤ 2	?	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Asteroidal number $\leq p$ $(p \geq 3)$	NP-c	NP-c	NP-c	?	?	?	$k \geq 3$:NP-c

Table 1.2: The computational complexity of Colouring (Col), Precolouring Extension (Precol), List Colouring (List Col), k-Colouring (k-Col), k-Precolouring Extension (k-Precol), List k-Colouring (List k-Col), and k-List Colouring (k-List Col) for graphs with some graph parameter that is bounded by some fixed constant. This bound is an upper bound for all cases with one exception: the fixed bound on the girth is a lower bound. We write "P", "NP-c" and "?" to indicate that a certain problem is polynomial-time solvable, NP-complete or open, respectively, whereas "YES" means that every instance from the corresponding graph class is a yes-instance.

We only explain those results that cannot be deduced from results that follow from the relationships in Figure 1.1 showed in Section 1.1.

All graphs. All results on general graphs follow from Theorem 1.1.1. Because every graph has girth at least 3, this case is also covered by the row in Table 1.2 that corresponds to the girth, i.e., choose $g = 3$.

Graphs of bounded treewidth. A *tree decomposition* of G is a pair $(\mathcal{T}, \mathcal{X})$, where $\mathcal{X}$ is a collection of subsets of V, called *bags*, and $\mathcal{T}$ is a tree whose vertices are the sets of $\mathcal{X}$, such that the following three properties are satisfied.

- $\bigcup_{X \in \mathcal{X}} X = V$,
- for each edge $uv \in E$, there is a bag $X \in \mathcal{X}$ with $u, v \in X$,
- for each $x \in V$, the set of bags containing x forms a connected subtree of $\mathcal{T}$.

The *width* of a tree decomposition $(\mathcal{T}, \mathcal{X})$ is the size of a largest bag in $\mathcal{X}$ minus 1. The *treewidth* of G is the minimum width over all possible tree decompositions of G. We will frequently use the following theorem due to Jansen and Scheffler [59].

Theorem 1.5.1. *Let $\mathcal{G}$ be a graph class of treewidth at most t, then* List Colouring *can be solved in time $O(nk^{t+1})$ for n-vertex graphs of $\mathcal{G}$ that have a k-list assignment.*

Due to Theorem 1.5.1, List Colouring is polynomial-time solvable for known graph classes such as cycles, series-parallel graphs, d-outerplanar graphs (for fixed d), Halin graphs and trees (see Bodlaender [9]). To keep Figure 1.2 showed in Section 1.2 somewhat readable, we only included the class of trees in this figure.

Graphs of bounded clique-width. The graph parameter *clique-width* is defined via a graph construction process where only a limited number of vertex labels are available; vertices that share the same label at a certain point of the construction process must be treated uniformly in subsequent steps. In particular, one can use the following four operations: the creation of a new vertex with label i, the vertex-disjoint union of already constructed labeled graphs, the relabeling of all vertices of label i with label j, and the insertion of all possible edges between vertices of label i and label j. The clique-width of a graph G is the smallest number k of labels that suffices to construct G by means of these four operations. For instance, every complete graph has clique-width 1. Any graph of bounded treewidth has bounded clique-width, whereas the converse statement is not true (e.g. consider the class of cliques).

Theorem 1.5.2. *Let $\mathcal{G}$ be a graph class of bounded clique-width. The following two statements hold:*

(i) Colouring *can be solved in polynomial time on $\mathcal{G}$*

(ii) For all $k \geq 1$, List k-Colouring *can be solved in linear time on $\mathcal{G}$.*

Proof. Statement (i) is due to Kobler and Rotics [64] (also see Rao [86]). Statement (ii) follows from the fact that for any fixed integer k, the List k-Colouring problem can be expressed in so-called Monadic Second Order Logic with logical formulas that do not use edge set quantifications, and such problems are linear-time solvable on graphs of bounded clique-width, as shown by Courcelle, Makowsky and Rotics [25]. □

Statement (i) of Theorem 1.5.2 is not valid for Precolouring Extension and List Colouring. For instance, Bonomo, Durán and Marenco [10] proved that Precolouring Extension is NP-complete for distance-hereditary graphs, which have clique-width at most 3 [46]. Also, complete graphs minus a matching are readily seen to have clique-width at most 3. However, already 3-List Colouring is NP-complete for this graph class (4.2). Theorem 1.5.2 together with the above statement explains the row in Table 1.2 that corresponds to the case when the clique-width is at most p for some fixed $p \geq 3$. We now consider the two rows corresponding to clique-width at most 1 and 2. A graph has clique-width at most 1 if and only if it is a disjoint union of complete graphs. A graph has clique-width at most 2 if and only if it is a cograph [26]. As such we copied the rows for graphs of clique-width at most 1 and at most 2, respectively, from the corresponding rows of Table 1.3, which we provide in Section 1.6.

Graphs of bounded degree. Kratochvíl and Tuza [72] showed that List Colouring is polynomial-time solvable on graphs of maximum degree 2. This result is tight, as they also showed that already 3-List Colouring is NP-complete even for planar graphs of maximum degree 3. Chlebík and Chlebíková [20] showed that Precolouring Extension is polynomial-time solvable on graphs of maximum degree 3. This result is tight, because already 3-Colouring is NP-complete for graphs of maximum degree 4 (see [36]). Kochol, Lozin and Randerath [65] explored to what extent the degree-3 condition can be relaxed by determining the complexity of 3-Colouring for each graph class $\mathcal{G}(\mathcal{H})$ defined as follows. A graph class $\mathcal{G}(\mathcal{H})$ consists of all graphs of maximum degree at most 4, for which the neighbourhood of each vertex of degree 4 induces some graph isomorphic

to $\mathcal{H}$. They show that 3-Colouring is NP-complete for a class $\mathcal{G}(\mathcal{H})$ if $\mathcal{H}$ contains one of the graphs in $\{4P_1, 2P_1 + P_2, 2P_2\}$, and linear-time solvable otherwise. Note that the problem k-Precolouring Extension is trivially polynomial-time solvable on graphs with maximum degree at most p if $p \leq k - 1$. Golovach and Paulusma [39] showed that List 3-Colouring is NP-complete for graphs of maximum degree 3, in which all cycles have arbitrarily large girth and any two degree-3 vertices are of arbitrary large distance from each other. However, the complexity of 4-Colouring and 4-Precolouring Extension is not known for graphs of maximum degree 4. Due to Theorem 1.3.3, the k-Colouring problem is NP-complete on $(2k-2)$-regular line graphs for all $k \geq 3$.

Graphs of large girth. The results in Table 1.2 follow from Theorem 1.3.2. Kamiński and Lozin [61] also showed that for all fixed $g \geq 3$, the 3-Colouring problem is NP-complete for line graphs of 3-regular graphs of girth at least g.

Graphs of bounded clique number. A graph has clique number 1 if and only if it is a disjoint union of complete graphs. As such we refer to the corresponding row of Table 1.3 provided in Section 1.6 for the complexity results in this case. By choosing $g = 4$ in the statement of Theorem 1.3.2, one obtains that k-Colouring is NP-complete on triangle-free graphs for all $k \geq 3$. Consequently, k-Colouring is NP-complete on graphs of clique number at most 2 for all $k \geq 3$.

Graphs with bounded independence number. Recall that a graph has independence number α if and only if it is $(\alpha+1)P_1$-free. Hence, all results for Colouring, Precolouring Extension, List Colouring and k-List Colouring follow from Theorem 1.3.1. The List k-Colouring problem is constant-time solvable on rP_1-free graphs for all $k \geq 1$ and $r \geq 1$, because every colouring of any graph G on more than kr vertices respects no k-list assignment.

Graphs of bounded diameter. We first note that a graph has diameter 1 if and only if it is a complete graph. Hence, we can copy the corresponding row of Table 1.3 showed in Section 1.6 for this case. We claim that for all $k \geq 1$ the problems k-Colouring and k-Precolouring Extension are polynomially equivalent on all graphs of diameter at most p for any $p \geq 1$. This can be seen as follows. First, k-Colouring is a special case of k-Precolouring Extension. Second, if we are given a graph G of diameter at most p with a precolouring ϕ_W for some $W \subseteq V(G)$, then we identify any two vertices of W that are coloured alike. This operation is allowed, because it does not increase the diameter of

the graph. For the same reason we may add an edge between the remaining precoloured vertices (which all have distinct colours). Hence we have obtained an equivalent instance of k-Colouring for graphs of diameter at most p. It is not difficult to see that k-Colouring is NP-complete for graphs of diameter d for all pairs (k, d) with $k \geq 3$ and $d \geq 2$ except when $(k, d) \in \{(3, 2), (3, 3)\}$. Recently, Mertzios and Spirakis [80] solved one of the two remaining cases by showing that 3-Colouring is NP-complete even for triangle-free graphs $G = (V, E)$ of diameter 3, radius 2 and minimum degree $\delta = \theta(|V|^{\epsilon})$ for every $0 \leq \epsilon \leq 1$.

Graphs with bounded asteroidal number. Recall that an asteroidal triple in a graph is a set of three mutually non-adjacent vertices such that each two of them are joined by a path that avoids the neighbourhood of the third. An *asteroidal set* in a graph G is an independent set $S \subseteq V(G)$, such that every triple of vertices of S forms an asteroidal triple. The *asteroidal number* is the size of a largest asteroidal set in G. Complete graphs are exactly those graphs that have asteroidal number at most one, and that AT-free graphs are exactly those graphs that have asteroidal number at most two. Hence we refer to Table 1.3 provided in Section 1.6 for these two cases. Because Colouring and k-List Colouring ($k \geq 3$) are NP-complete for graphs of independence number at most 3 (see Table 1.2), we find that these problems are NP-complete for the class of graphs with asteroidal number at most 3. The complexity of the problems k-Colouring, k-Precolouring Extension and List k-Colouring on graphs with asteroidal number at most p is not known for any fixed $p \geq 3$.

1.6 Results for the Graphs in Figure 1.2

Before we discuss the results known for the graph classes in Figure 1.2 showed in Section 1.2, we first introduce the following well-known concept. An *edge colouring* of a graph $G = (V, E)$ is a mapping $\phi : E \to \{1, 2, \ldots\}$ such that $\phi(e) \neq \phi(f)$ for any distinct edges that have a common end-vertex. We call $\phi(e)$ the *colour* of e. A *k-edge colouring* of G is a colouring ϕ of G with $1 \leq \phi(e) \leq k$ for all $e \in E$. In that case we say that G is *k-edge colourable*. The *chromatic index* of a graph G is the smallest integer k for which G is k-edge colourable. Note that an edge mapping $\phi : E(G) \to \{1, \ldots, k\}$ is a k-edge colouring of G if and only if ϕ is a k-colouring of $\texttt{line}(G)$. Hence the chromatic index of G is equal to the chromatic number of $\texttt{line}(G)$. Vizing's theorem [95] tells us that

the chromatic index of any graph G, or equivalently, the chromatic number of $\texttt{line}(G)$ is either $\Delta(G)$ or $\Delta(G)+1$. An edge $e=uv$ of a graph G is a *dominating* edge if $\{u,v\}$ is a dominating set of G.

Theorem 1.6.1. *For all $k \geq 1$, the* List k-Colouring *problem is constant-time solvable on line graphs of graphs that have a dominating edge.*

Proof. Let $k \geq 1$ be an integer and G be a graph on n vertices with a dominating edge $e=uv$ and a k-list assignment $\mathcal{L}$. Because e is dominating, $\deg_G(u) \geq \frac{n}{2}$ or $\deg_G(v) \geq \frac{n}{2}$. Hence, $\Delta(G) \geq \frac{n}{2}$. By Vizing's theorem, the chromatic number of $\texttt{line}(G)$ is at least $\Delta(G) \geq \frac{n}{2}$. This means that $\texttt{line}(G)$ has no colouring that respects $\mathcal{L}$ if $k \leq \frac{n}{2}-1$. Suppose that $k \geq \frac{n}{2}$. Then $n \leq 2k$, which is a constant because k is fixed. Hence the result follows. □

Table 1.3 provides a summary of the complexity results for the problems Colouring, Precolouring Extension, List Colouring, k-Colouring, k-Precolouring Extension, List k-Colouring and k-List Colouring for the graph classes displayed in Figure 1.2 showed in Section 1.2. Below we explain the results in this table. We consider the rows of Table 1.3 one by one, and only explain those results that cannot be deduced from results for other graph classes using the relationships in Figures 1.1 and 1.2 that are showed in Section 1.1 and Section 1.2. Due to Theorem 1.1.1 we may restrict ourselves to $k \geq 3$ when considering the problems k-Colouring, k-Precolouring Extension, List k-Colouring and k-List Colouring. However, instances from some graph classes may always be yes-instances for some of these problems for certain values of k. Such cases are mentioned explicitly in Table 1.3 by denoting them with a "YES" just as we did in Table 1.2 showed in Section 1.5. Also the other abbreviations used in Table 1.3 originate from Table 1.2. For clarity we copied the first row of Table 1.2; this is the row corresponding to general graphs.

All graphs. Recall that all results on general graphs follow from Theorem 1.1.1; also see Table 1.2.

AT-free graphs. Stacho [90] proved that 3-Colouring is polynomial-time solvable on AT-free graphs. Recently, Kratsch and Müller [71] extended this result by showing that even List k-Colouring is polynomial-time solvable on these graphs for any fixed integer $k \geq 1$. All NP-completeness results follow from the corresponding results for co-comparability graphs. The complexity of Colouring for AT-free graphs is not known.

Graph Class	Col	Precol	List Col	k-Col	k-Precol	List k-Col	k-List Col
All	NP-c	NP-c	NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
AT-free	?	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Bipartite	P	NP-c	NP-c	$k \geq 2$:YES	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Block	P	P	P	P	P	P	P
Chordal	P	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Claw-free	NP-c	NP-c	NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Co-bipartite	P	P	NP-c	P	P	P	$k \geq 3$:NP-c
Co-comparability	P	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Cograph	P	P	NP-c	P	P	P	$k \geq 3$:NP-c
Comparability	P	NP-c	NP-c	P	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Complete	P	P	P	P	P	P	P
Complete bipartite	P	P	NP-c	$k \geq 2$:YES	P	P	$k \geq 3$:NP-c
Complete minus matching	P	P	NP-c	P	P	P	$k \geq 3$:NP-c
Complete split	P	P	NP-c	P	P	P	$k \geq 3$:NP-c
Distance-hereditary	P	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Interval	P	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Line	NP-c	NP-c	NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Line of complete	P	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Line of complete bipartite	P	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Perfect	P	NP-c	NP-c	P	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Permutation	P	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Planar	NP-c	NP-c	NP-c	$k = 3$:NP-c $k \geq 4$:YES	$k \geq 3$:NP-c	$k \geq 3$:NP-c	$k \geq 3$:NP-c
Proper interval	P	NP-c	NP-c	P	P	P	$k \geq 3$:NP-c
Split	P	P	NP-c	P	P	P	$k \geq 3$:NP-c
Threshold	P	P	NP-c	P	P	P	$k \geq 3$:NP-c
Tree	P	P	P	$k \geq 2$:YES	P	P	P
Trivially perfect	P	P	NP-c	P	P	P	$k \geq 3$:NP-c
Union of two complete	P	P	NP-c	P	P	P	$k \geq 3$:NP-c

Table 1.3: The computational complexity of the colouring problems for the graph classes in Figure 1.2 showed in Section 1.2.

Recall that AT-free graphs are exactly those graphs with asteroidal number at most 2. Hence, the missing entry in Table 1.3 is included in Open Problem 3.

Bipartite graphs. A graph is readily seen to be bipartite if and only if it is 2-colourable. All NP-completeness results follow from the result of Kratochvíl [70] who proved that 3-Precolouring Extension is NP-complete for planar bipartite graphs.

Block graphs. Bonomo, Durán and Marenci [10] showed that List Colouring can be solved in polynomial time for block graphs.

Chordal graphs. We show how to solve List k-Colouring in polynomial time for chordal graphs. A graph is chordal if and only if it has a tree decomposition whose set of bags

is exactly the set of its maximal cliques [37]. Moreover, such a tree decomposition can be constructed in linear time [7]. This means that we can do as follows. Let G be a chordal graph with a k-list assignment $\mathcal{L}$. We compute the size $\omega(G)$ of a largest clique in G in linear time. If $\omega(G) \geq k+1$, then G has no colouring respecting $\mathcal{L}$. If $\omega(G) \leq k$, then the treewidth of G is at most $k-1$, and we apply Theorem 1.5.1. The polynomial-time result for Colouring follows from the corresponding result for perfect graphs. All NP-completeness results follow from the corresponding results for interval graphs.

Claw-free graphs. All (NP-completeness) results follow from the corresponding results for line graphs. The 3-Colouring problem is even NP-complete for claw-free graphs of maximum degree 4 [65]. Kamiński and Lozin [62] gave a necessary condition for the polynomial-time solvability of 3-Colouring in subclasses of claw-free graphs defined by forbidding a finite number of induced subgraphs.

Co-bipartite graphs. Hujter and Tuza [56] showed that Precolouring Extension can be solved in polynomial time on co-bipartite graphs. The polynomial-time result for List k-Colouring follows from the corresponding result for co-comparability graphs. All NP-completeness results follow from the corresponding results for complete graphs minus a matching.

Co-comparability graphs. All polynomial-time results follow from the corresponding results of AT-free graphs and perfect graphs. All NP-completeness results follow from the corresponding results for permutation graphs.

Cographs. Both Hujter and Tuza [57] and Jansen and Scheffler [59] showed that Precolouring Extension can be solved in polynomial time on cographs. The polynomial-time result for List k-Colouring follows from the corresponding result for permutation graphs. All NP-completeness results follow from the corresponding results for complete bipartite graphs.

Comparability graphs. All polynomial-time results follow from the corresponding results for perfect graphs. All NP-completeness results follow from the corresponding results for bipartite graphs.

Complete graphs. All (polynomial-time) results follow from the corresponding results for block graphs.

Complete bipartite graphs. The proof of Theorem 4.5 in the paper by Jansen and Scheffler [59] is to show that List Colouring is NP-complete on P_4-free graphs but in

fact shows that 3-List Colouring is NP-complete for complete bipartite graphs. All polynomial-time results follow from the corresponding results for bipartite graphs and cographs.

Complete graphs minus a matching. Recall that we proved that 3-List Colouring is NP-complete for complete graphs minus a matching (4.2). All polynomial-time results follow from the corresponding results for co-bipartite graphs.

Complete split graphs. The proof of Theorem 2 in the paper by Golovach and Heggernes [38] shows that 3-List Colouring is NP-complete for complete bipartite graphs. By adding all possible edges between vertices in one bipartition class of their complete bipartite gadget graph, one can in fact prove that 3-List Colouring is NP-complete for complete split graphs. All polynomial-time results follow from the corresponding results for threshold graphs.

Distance-hereditary graphs. Recall that Bonomo, Durán and Marenco [10] proved that the Precolouring Extension problem is NP-complete for distance-hereditary graphs. The NP-completeness result for k-List Colouring follows from the corresponding result for cographs. Because distance-hereditary graphs have clique-width at most 3 [46], List k-Colouring is polynomial-time solvable for these graphs by Theorem 1.5.2. The polynomial-time result for Colouring follows from the corresponding result for perfect graphs.

Interval graphs. All polynomial-time results follow from the corresponding results for co-comparability graphs. All NP-completeness results follow from the corresponding results for proper interval graphs.

Line graphs. Recall that Holyer [54] showed that 3-Colouring is NP-complete for line graphs of regular graphs, and recall that Leven and Galil [77] extended this result by showing that k-Colouring is NP-complete on these graphs for all $k \geq 4$.

Line graphs of complete graphs. König [67] showed that Colouring is polynomial-time solvable on line graphs of complete graphs. Bonomo, Durán and Marenco [10] proved that Precolouring Extension is NP-complete for line graphs of complete graphs. Because complete graphs have a dominating edge, List k-Colouring is polynomial-time solvable on this graph class for all $k \geq 1$ by Theorem 1.6.1. Kubale [73] showed in Theorem 8 of his paper that 3-List Colouring is NP-complete on line graphs $\texttt{line}(G)$

of bipartite graphs G with $\Delta(G) = 3$. By a small modification of the proof of this result we find that 3-List Colouring is NP-complete on line graphs of complete graphs.

Line graphs of complete bipartite graphs. Colbourn [23] showed that Precolouring Extension is NP-complete for line graphs of complete bipartite graphs $K_{n,n}$. By another small modification of the proof of Theorem 8 of the aforementioned paper of Kubale [73] we find that 3-List Colouring is NP-complete on line graphs of complete bipartite graphs. Because complete bipartite graphs have a dominating edge, List k-Colouring is polynomial-time solvable on this graph class for all $k \geq 1$ by Theorem 1.6.1. The polynomial-time result for Colouring follows from the corresponding result for perfect graphs.

Perfect graphs. Grötschel [47] showed that Colouring on a perfect graph can be solved in polynomial time. All NP-completeness results follow from the corresponding results for comparability graphs.

Permutation graphs. Jansen [58] showed that Precolouring Extension is NP-complete for permutation graphs. The NP-completeness result for k-List Colouring follows from the corresponding result for cographs. All polynomial-time results follow from the corresponding results for co-comparability graphs.

Planar graphs. Garey, Johnson and Stockmeyer [36] proved that 3-Colouring is NP-complete even for planar graphs with maximum degree 4, whereas every planar graph is 4-colourable by the Four Colour Theorem [1]. We note that Dvořák, Kawarabayashi and Thomas [31] showed that 3-Colouring can be solved in linear time for triangle-free planar graphs improving a result of Kowalik [68] who gave the first $o(n^2)$-time algorithm for solving 3-Colouring on triangle-free planar graphs.

Proper interval graphs. Marx [79] showed that Precolouring Extension is NP-complete for proper interval graphs. The NP-completeness result for k-List Colouring follows from the corresponding result for unions of two complete graphs. All polynomial-time results follow from the corresponding results for interval graphs.

Split graphs. Hujter and Tuza [57] showed that Precolouring Extension can be solved in polynomial time on split graphs. The polynomial-time result for List k-Colouring follows from the corresponding result for chordal graphs. All NP-completeness results follow from the corresponding results for threshold graphs.

Threshold graphs. All polynomial-time results follow from the corresponding results for split graphs. All NP-completeness results follow from the corresponding results for complete split graphs.

Trees. All (polynomial-time) results follow from the corresponding results for block graphs and bipartite graphs.

Trivially perfect graphs. All polynomial-time results follow from the corresponding results for cographs. All NP-completeness results follow from the corresponding results for threshold graphs.

Unions of two complete graphs. The proof of Theorem 11 in the paper by Jansen [58] is to show that List Colouring is NP-complete for unions of two complete graphs, but in fact shows that 3-List Colouring is NP-complete for these graphs. All polynomial-time results follow from the corresponding results for cographs.

1.7 Generalizing Graph Colouring

For two graphs G and F, a mapping $f : V(G) \to V(F)$ is called a *homomorphism* from G to F if $f(u)f(v) \in E(F)$ whenever $uv \in E(G)$. The Homomorphism problem is that of testing whether a given graph G allows a homomorphism to some given graph F. This is a generalisation of the Colouring problem. If F is the complete graph on k vertices, then testing whether there exists a homomorphism from a graph G to F is equivalent to testing whether G is k-colourable. The most famous result in this area is the Hell-Nešetřil dichotomy [51]. According to this dichotomy, deciding whether a given graph allows a homomorphism to a fixed graph F is polynomial-time solvable if F is bipartite, and NP-complete otherwise. Since graph homomorphisms are beyond the scope of this thesis, we do not discuss graph homomorphisms any further but refer to Hell and Nešetřil [52] for a survey on them.

1.8 Open Problems

Recall that Table 1.1 provided in Section 1.3 shows the complexity of k-Colouring, k-Precolouring Extension and List k-Colouring on P_r-free graphs on P_r-free graphs for fixed k and r. This leads to the following open problem.

Open Problem 1. *Complete the classification of the complexity of* k-Colouring, k-Precolouring Extension *and* List k-Colouring *for* H*-free graphs.*

As we have seen earlier this chapter, Golovach and Paulusma [39] completely classified the complexity of List Colouring and ℓ-List Colouring ($\ell \geq 3$) for (H_1, H_2)-free graphs. But the classification of the complexity of Precolouring Extension for (H_1, H_2)-free graphs is still open.

Open Problem 2. *Complete the classification of the complexity of* Precolouring Extension *for* (H_1, H_2)*-free graphs and the complexity of* List Colouring *for* (H_1, H_2, H_3)*-free graphs.*

Finally, the entries in Table 1.2 provided in Section 1.5 marked "?" are also open.

Open Problem 3. *Determine the complexity of the entries in Table 1.2 marked "?".*

1.9 Thesis Overview

We consider the Colouring problem and its variants restricted to those graphs that can be characterised by one or more forbidden induced subgraphs. Note that one may also forbid a graph H as a (not necessarily induced) subgraph; we refer to Golovach, Paulusma and Ries [40] for more details on this. Also note that some of the polynomial-time algorithms in this thesis are far from practical: our main objective was to research the borderline of being tractable or intractable. For example, the $2^n n^{O(1)}$ time algorithm by Björklund, Husfeldt and Koivisto [6] to solve Coloring may be more practical than our polynomial-time algorithms. This thesis is based on six research papers [18, 19, 41, 42, 43, 44], the results of which we have already stated in Section 1.3 and Section 1.4. The paper [19] and the paper [41] are the journal versions of the paper [18] and the paper [42], respectively. In the coming chapters we will prove these results.

In Chapter 2, we focus on the problems Colouring and k-Colouring. We first prove that 4-Colouring on $(P_2 + P_3)$-free is NP-complete, and 4-Colouring on P_8-free graphs is NP-complete. We then show the existence of an integer r such that 4-Colouring is NP-complete for P_r-free graphs with girth 4. In contrast, we also determine for any fixed girth $g \geq 4$ a value $r(g)$ such that every $P_{r(g)}$-free graph with girth at least g is 3-colourable. Here we tried to make $r(g)$ as large as possible.

In Chapter 3, we investigate the problem Precolouring Extension and the problem k-Precolouring Extension. We first give a computational complexity classification for Precolouring Extension. We show that for a fixed graph H, Precolouring Extension is polynomial-time solvable for H-free graphs if H is an induced subgraph of P_4 or of $P_1 + P_3$; otherwise Precolouring Extension is NP-complete for H-free graphs. Next, we present a polynomial-time algorithm for solving 4-Precolouring Extension for (P_2+P_3)-free graphs. We finish this chapter by showing that 4-Precolouring Extension on P_7-free graphs is NP-complete.

In Chapter 4, we focus on the List Colouring problem and its two variants List k-Colouring and k-List Colouring. We first present a computational complexity classification for k-List Colouring on H-free graphs, which immediately yields a complexity classification for List Colouring on H-free graphs. We show that for a fixed integer ℓ and a fixed graph H, ℓ-List Colouring is polynomial-time solvable on H-free graphs if $\ell \leq 2$ or H is an induced subgraph of P_3; otherwise ℓ-List Colouring is NP-complete for H-free graphs. We then show that 3-List Colouring for $(3P_1, P_1 + P_2)$-free graphs is NP-complete. Next we settle the computational complexity of List 3-Colouring for $(P_2 + P_4)$-free graphs and for sP_3-free graphs for any fixed s by presenting a polynomial-time algorithm for each of the graph classes. We then show that List 4-Colouring on P_6-free graphs is NP-complete. Finally, we show a polynomial-time algorithm for List k-Colouring on $(K_{s,t}, P_r)$-free graphs for all integers $k, r, s, t \geq 1$.

Chapter 2

Colouring

The main ingredients of this chapter are from the following papers.

Section 2.1:

[41] P.A. Golovach, D. Paulusma and J. Song. 4-Coloring H-free graphs when H is small. *Discrete Applied Mathematics*, 161:140–150, 2013.

[42] P.A. Golovach, D. Paulusma and J. Song. 4-Coloring H-free graphs when H Is small. In: *Proceedings of the 38th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2012)*, volume 7147 of *Lecture Notes in Computer Science*, 289–300, 2012.

Section 2.2:

[19] H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song. Updating the complexity status of coloring graphs without a fixed induced linear forest. *Theoretical Computer Science*, 414:9–19, 2012.

[18] H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song. Narrowing down the gap on the complexity of coloring Pk-free graphs. In: *Proceedings of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2010)*, volume 6410 of *Lecture Notes in Computer Science*, pages 63–74, 2010.

Section 2.3, 2.4:

[44] P.A. Golovach, D. Paulusma and J. Song. Coloring graphs without short cycles and long induced paths. In: *Proceedings of the 18th International Symposium on Fundamentals of Computation Theory (FCT 2011)*, volume 6914 of *Lecture Notes in Computer Science*, pages 193–204, 2011.

In this chapter, we focus on the problems Colouring and k-Colouring. We first prove that 4-Colouring on $(P_2 + P_3)$-free is NP-complete in Section 2.1, and 4-Colouring on P_8-free graphs is NP-complete in Section 2.2. For graphs with girth 4, we show the existence of an integer r such that 4-Colouring is also NP-complete for P_r-free graphs with such girth in Section 2.3. In contrast, we also determine for any fixed girth $g \geq 4$ a value $r(g)$ such that every $P_{r(g)}$-free graph with girth at least g is 3-colourable in Section 2.4. Here we tried to make $r(g)$ as large as possible.

2.1 4-Colouring for $(P_2 + P_3)$-free Graphs

In this section, we present a polynomial-time algorithm for solving 4-Colouring on (P_2+P_3)-free graphs. This algorithm heavily relies on a number of structural properties, some of which are valid even for sP_3-free graphs for any fixed integer $s \geq 1$. In the latter case we prove them for sP_3-free graphs, because we consider sP_3-free graphs in Section 4.4 and there we need these properties as well. Before showing these properties, we first introduce some additional terminology.

Let $G = (V, E)$ be a graph. Let I be an independent set in G, and let X be a subset of $V \setminus I$. We write $I(X) := N_G(X) \cap I$ and $I(\overline{X}) := I \setminus N_G(X)$, so $I = I(X) \cup I(\overline{X})$ and $I(X) \cap I(\overline{X}) = \emptyset$. If every vertex in $N_G(I) \setminus X$ has at most one neighbour in $I(\overline{X})$ then we say that X *pseudo-dominates* I. This notion plays a crucial role in the design of our algorithm. An example of a set X that pseudo-dominates a set I is illustrated in Figure 2.1.

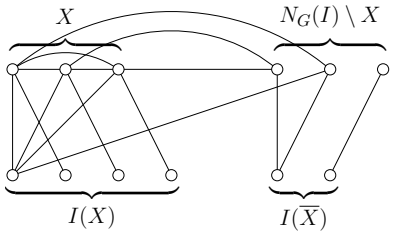

Figure 2.1: A set X that pseudo-dominates a set I.

Recall that for positive integers p and q, the Ramsey number $R(p, q)$ is the smallest number of vertices n such that all graphs on n vertices contain an independent set of size p or a clique of size q. Also recall that Ramsey's Theorem states that such a number

exists for all positive integers p and q. Using Ramsey numbers we can prove the following structural result on pseudo-dominating sets.

Lemma 2.1.1. *Let I be an independent set in a k-colourable sP_3-free graph $G = (V, E)$ for some integer $s \geq 2$. Then $G[V \setminus I]$ contains a set X of cardinality at most $R(s, k+1)$ that pseudo-dominates I.*

Proof. We may assume that $G[V \setminus I]$ contains more than $R(s, k+1)$ vertices; otherwise there is nothing to prove. Let X be a subset of $V \setminus I$ with $R(s, k+1)$ vertices that dominates the maximum number of vertices in I over all subsets of $V \setminus I$ of size $R(s, k+1)$.

Suppose that X does not pseudo-dominate I. Then there is a vertex $v \in N_G(I) \setminus X$ adjacent to at least two vertices in $I(\overline{X})$. This means that any vertex $x \in X$ has two neighbours u_x, w_x in I that are not adjacent to any vertex in $X \setminus \{x\}$; otherwise the set $X' = (X \cup \{v\}) \setminus \{x\}$ of size $R(s, k+1)$ dominates more vertices than X, contradicting the maximality of X. From the definition of Ramsey number $R(s, k+1)$, we find that X has an independent set $\{x_1, \ldots, x_s\}$ or a clique of size $k+1$. In the first case we have an induced sP_3 consisting of the s paths $u_{x_i} x_i w_{x_i}$, contradicting the sP_3-freeness of G. In the second case, when X has a clique of $k+1$ vertices, G is not k-colourable, which contradicts our assumptions as well. This completes the proof. □

We note that the upper bound on the size of X in Lemma 2.1.1 can be slightly improved for $s = 2$, as we showed in the paper [17].

Before we prove our main result of this section, we need two more lemmas.

Lemma 2.1.2. *Let G be an sP_3-free graph that contains a set X and an independent set I, such that X pseudo-dominates I. Let $k \geq 1$. If $I(\overline{X})$ contains more than $k(s-1)$ vertices with degree at least k in G, then G is not k-colourable.*

Proof. Let G, I and X be defined as in the statement of the lemma. Let $k \geq 1$. Let S be the subset of vertices in $I(\overline{X})$ that have degree at least k in G. Suppose $|S| > k(s-1)$. In order to derive a contradiction, assume G has a k-colouring ϕ.

Every vertex in G with degree at least k must have at least two neighbours with the same colour in ϕ. Thus, because every vertex in S has degree at least k, every vertex in S has at least two neighbours with the same colour. Because S contains more than $k(s-1)$ vertices, this means that there exist s vertices $u_1, \ldots, u_s$ in S and a colour j such that each u_i has (at least) two neighbours x_i and y_i with colour j. The set

$\{x_1, \ldots x_s\} \cup \{y_1, \ldots, y_s\}$ is an independent set because all its vertices have the same colour, namely colour j. Because X pseudo-dominates I, a vertex u_i is neither adjacent to x_h nor to y_h whenever $h \neq i$. This implies that the s paths $x_i u_i y_i$ form an induced sP_3 in G, which is not possible. This contradiction yields that G is not k-colourable. □

Lemma 2.1.3. *Let $G = (V, E)$ be an sP_3-free graph with a k-list assignment $\mathcal{L}$ for some integer $k \geq 1$. Let $W \subseteq V$ be the set of those vertices in G whose lists have size at most 2. If every vertex in $V \setminus W$ has degree at least k, and G has a colouring that respects $\mathcal{L}$, then G contains a set D of size at most $k \cdot R(s, k+1) + (k^2+3) \cdot (s-1)$ that dominates $V \setminus W$.*

Proof. Let $G = (V, E)$ be an sP_3-free graph with a k-list assignment $\mathcal{L}$ for some integer $k \geq 1$. Let $W \subseteq V$ be the set of those vertices in G whose lists have size at most 2. Assume that every vertex in $V(G) \setminus W$ has degree at least k, and that G has a colouring that respects $\mathcal{L}$. The second assumption implies that G is K_{k+1}-free. If $s = 1$, then every component of G is a complete graph at most k vertices. Consequently, G has no vertex of degree at least k, and the statement of the lemma holds. Assume that $s \geq 2$ and assume without loss of generality that G is not $(s-1)P_3$-free. Let S be the vertex set of an induced subgraph of G that is isomorphic to $(s-1)P_3$, hence $|S| = 3(s-1)$. If S is a dominating set of G, then the statement of the lemma holds.

Suppose S is not a dominating set of G. Let G' be the graph obtained from G after removing $S \cup W$ and all vertices in $N_G(S)$. Because G is (K_{k+1}, sP_3)-free and S induces an $(s-1)P_3$, we find that G' is (K_{k+1}, P_3)-free. Hence, every component of G' is isomorphic to a graph from $\{K_1, \ldots, K_k\}$.

We partition the vertices of G' into at most k independent sets $I_1, \ldots, I_k$ as follows. First we form I_1 by taking exactly one vertex from each component of G'. We remove I_1 from G' and repeat the above step to obtain I_2 if there were any vertices of G' left. We proceed in this way until all vertices of G' have been used. This will happen after at most k steps, because every component of G' has at most k vertices at the start of this procedure.

We apply Lemma 2.1.1 to each I_h in order to find a set X_h with $|X_h| \leq R(s, k+1)$ in G that pseudo-dominates I_h for $h = 1, \ldots, k$.

We apply Lemma 2.1.2 to G and each I_h in order to find that $|I_h(\overline{X_h})| \leq k(s-1)$ for $h = 1, \ldots, k$. Then $D = S \cup X_1 \cup \cdots \cup X_k \cup I_1(\overline{X_1}) \cup \cdots \cup I_k(\overline{X_k})$ has at most

$3(s-1) + k \cdot R(s, k+1) + k \cdot k \cdot (s-1) = k \cdot \ R(s, k+1) + (k^2+3) \cdot (s-1)$ vertices. Because D is a dominating set in G, this completes the proof of Lemma 2.1.3. □

Because any $(P_2 + P_3)$-free graph is $2P_3$-free, we can apply Lemma 2.1.3 for $W = \emptyset$, $k = 4$ and $s = 2$. After observing that the Ramsey number $R(2, 5) = 5$, this leads us to the following lemma that is crucial for our algorithm.

Corollary 2.1.4. *Let $G = (V, E)$ be a $(P_2 + P_3)$-free graph of minimum degree at least 4. If G has a 4-colouring, then G contains a dominating set D of size at most 39.*

Note that Lemma 2.1.4 involves a minimum degree condition. This is not a problem since we can apply Proposition 1.2.2 by setting $L(u) = \{1, 2, 3, 4\}$ for every $u \in V(G)$ for the list assignment $\mathcal{L}$ in the statement of this proposition.

Broersma et al. [16] showed that 3-Precolouring Extension is polynomial-time solvable for P_6-free graphs. They note that their proof of this result can be used to show the stronger statement that List 3-Colouring can be solved in polynomial time for P_6-free graphs. Because every $(P_2 + P_3)$-free graph is P_6-free, we obtain the following lemma which we need for proving the correctness of our algorithm.

Lemma 2.1.5. *The* List 3-Colouring *problem can be solved in polynomial time for the class of $(P_2 + P_3)$-free graphs.*

We also need the following lemma, which follows immediately from Lemma 2.1.5.

Lemma 2.1.6. *Let $G = (V, E)$ be a $(P_2 + P_3)$-free graph. Then a partition of V into three (possibly empty) independent sets I_1, I_2, I_3 can be found in polynomial time if it exists.*

We now give a polynomial-time algorithm for the 4-Colouring problem restricted to $(P_2 + P_3)$-free graphs. Let G be a $(P_2 + P_3)$-free graph that is an instance of 4-Colouring. By Proposition 1.2.2 we may assume that G has minimum degree at least 4. We also assume that each vertex u has been assigned an initial list $L_0(u) = \{1, 2, 3, 4\}$ of admissible colours.

Outline. Our algorithm is a branching algorithm. The main idea is to obtain in polynomial time a polynomial-bounded set $\mathscr{L}$ of list assignments for G that have the following two properties. First, G has a 4-colouring if and only if G has a colouring that respects at least one list assignment in $\mathscr{L}$. Second, for every list assignment in $\mathscr{L}$, we either have

that all its lists have size at most two or else that the union of its lists that contain at least 2 colours has size 3; in the first case we can use Theorem 1.1.1, and in the second case we can use Lemma 2.1.5 after removing all vertices with a single colour in their list from G. Because we obtain $\mathscr{L}$ in polynomial time and its size is bounded by a polynomial, this means that the total running time of our algorithm is polynomial.

Our algorithm consists of two phases. In Phase 1 we first check for a "small" dominating set D. Such a set D must exist in the case that G is 4-colourable, as we prove later. Because D has small size, the total number of 4-colourings of $G[D]$ will be "small" as well. The algorithm considers every 4-colouring of $G[D]$. Given such a 4-colouring of D, it partitions the remaining vertices of G in four different ways using Lemma 2.1.6. We use these partitions, together with further structural properties of $(P_2 + P_3)$-free graphs, for a branching procedure. At the end of Phase 1, we either have found that G has no 4-colouring or we have obtained a set $\mathscr{L}$ of list assignments, for which we will prove the desired properties specified in the outline. In Phase 2 we consider the list assignments of $\mathscr{L}$ one by one to determine whether G has a colouring respecting at least one of them.

We now describe Phases 1 and 2 in detail. Here, we use the following terminology. If we say that we "colour the vertices of a set U according to their lists", then we mean that we assign every vertex $u \in U$ a colour that is in the list of u, and moreover, such that two adjacent vertices in U do not get the same colour. Afterwards, for every $u \in U$, we may remove the colour of u from the list of every neighbour of u in $N_G(U)$. This is what we call *updating* the list assignment. Also, when colouring a vertex, say with colour i, then we set its list of admissible colours to $\{i\}$. After proving a number of lemmas, we show in Theorem 2.1.14 that our algorithm is correct and that it runs in polynomial time.

Phase 1. Determining the set $\mathscr{L}$.

Step 1. Check if G has a dominating set of size at most 39. If such a set does not exist, then return `NO`. Otherwise, let D be such a dominating set.

Step 2. Check if $G[D]$ is 4-colourable. If not, then return `NO`.

Assume that $G[D]$ is 4-colourable and set $\mathscr{L} = \emptyset$. Perform Steps 3-9 for every 4-colouring ϕ_D of $G[D]$.

Step 3. First update the list assignment. Then, for $i = 1, \ldots, 4$, let $D_i \subseteq D$ be the subset of vertices with colour i, and let $F_i = G[V \setminus (D \cup N_G(D_i)]$. Note that $V(F_h) \cap V(F_i) \neq \emptyset$

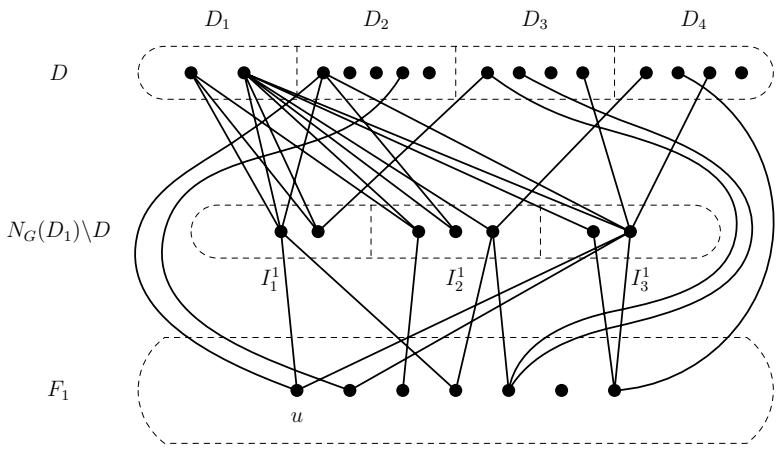

Figure 2.2: A graph G decomposed as $V(G) = D \cup (N_G(D_1) \setminus D) \cup V(F_1)$, where edges inside the different parts are not displayed; note that vertex u belongs to $F_1 \cap (N_G(D_2) \setminus D) \cap F_3 \cap F_4$ in this particular example.

is possible for $h \neq i$. For $i = 1, \ldots, 4$ check whether $N_G(D_i) \setminus D$ can be partitioned into three independent sets, where one or more of such sets are allowed to be empty; in particular, all three sets are empty if $N_G(D_i) \setminus D = \emptyset$. If such a partition does not exist for some i, then stop considering ϕ_D. Otherwise, let I_1^i, I_2^i, I_3^i be such a partition for $i = 1, \ldots, 4$. Figures 2.2 and 2.3 illustrate that

$$\begin{aligned} V(G) &= D \cup I_1^1 \cup I_2^1 \cup I_3^1 \cup I_1^2 \cup I_2^2 \cup I_3^2 \cup I_1^3 \cup I_2^3 \cup I_3^3 \cup I_1^4 \cup I_2^4 \cup I_3^4 \\ &= D \cup I_1^i \cup I_2^i \cup I_3^i \cup V(F_i) \qquad \text{for } i = 1, \ldots 4, \end{aligned}$$

where two sets I_j^i and $I_{j'}^{i'}$ may intersect but only if $i \neq i'$.

Step 4. For $i = 1, \ldots, 4$, determine the set Q_i of isolated vertices of F_i, i.e., that have no neighbours in F_i. For $i = 1, \ldots, 4$, let F_i' be the graph obtained from F_i by removing all vertices of Q_i.

If some F_i' is "small", then deal with this case in Step 5. Otherwise move on to Step 6. This case distinction is mainly made for technical reasons, i.e., it will simplify later statements.

Step 5. Check if there exists a graph F_i' that has at most two vertices. If so, then pick an arbitrary such F_i' and do as follows. Colour the vertices of Q_i with colour i. Consider every possible colouring of the vertices of F_i' according to their lists. Each time, update

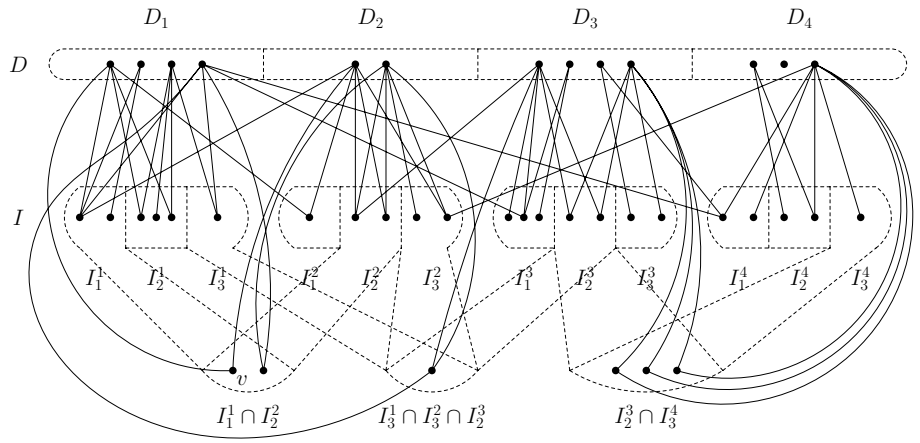

Figure 2.3: A graph G decomposed as $V(G) = D \cup \bigcup_{i,j} I_j^i$, where edges inside the different parts are not displayed; note that $I_1^1 \cap I_2^2 \neq \emptyset$, $I_3^1 \cap I_3^2 \cap I_2^3 \neq \emptyset$ and $I_2^3 \cap I_3^4 \neq \emptyset$, whereas all other sets I_j^i do not intersect in this particular example; also note for instance that $v \in I_1^1 \cap I_2^2$ belongs to $F_3 \cap F_4$ as well.

the list assignment and put the resulting list assignment in $\mathscr{L}$. Stop considering ϕ_D.

From now on assume that F_i' consists of at least three vertices for all $1 \leq i \leq 4$.

Step 6. For $i = 1, \ldots, 4$ and $j = 1, \ldots, 3$ do as follows. Check if $I_j^i \neq \emptyset$. If so, then do as follows. Find a vertex $a_j^i \in D_i$ that has the maximum number of neighbours in I_j^i over all vertices in D_i; we allow $a_j^i = a_{j'}^i$ for some $j \neq j'$. Define $\tilde{I}_j^i = I_j^i \cap N_G(a_j^i)$ if $I_j^i \neq \emptyset$, and $\tilde{I}_j^i = \emptyset$ otherwise.

For $i = 1, \ldots, 4$, let $I^i = I_1^i \cup I_2^i \cup I_3^i \setminus (\tilde{I}_1^i \cup \tilde{I}_2^i \cup \tilde{I}_3^i)$. Let $I^* = I^1 \cup I^2 \cup I^3 \cup I^4$.

Now, process the graphs F_i' further by first preforming Step 7 and then Step 8. Note that if some F_i' is non-bipartite, then F_i' is not processed at all in Step 7. Otherwise, F_i' is either connected and bipartite, or else disconnected and bipartite. In the latter case, F_i' is the disjoint union of at least two edges due to the $(P_2 + P_3)$-freeness of G and the fact that F_i' contains no isolated vertices by definition; as we shall see this property will be crucial.

Step 7. For $i = 1, \ldots, 4$ do as follows.

7a. If F_i' is connected and bipartite, then do as follows. Give all the vertices of one partition class colour i. Consider both possibilities. In both cases, colour the vertices of

Q_i with colour i, update the list assignment, put the resulting list assignment in $\mathscr{L}$ and restore all lists to the situation at the end of Step 6.

7b. If F'_i is disconnected and bipartite, then do as follows for every j with $\tilde{I}^i_j \neq \emptyset$. Consider every edge in F'_i that has no end-vertex with list $\{i\}$ already. If both end-vertices are adjacent to all but at most three vertices of $\tilde{I}^i_j$, then arbitrarily pick one of these end-vertices and colour it with i. If exactly one end-vertex is adjacent to all but at most three vertices of $\tilde{I}^i_j$, then colour that end-vertex with colour i. Afterwards, let S^i_j be the set of edges in F'_i, both end-vertices of which are not coloured i. Consider every possible colouring of the end-vertices of the edges in S^i_j according to their lists. Each time, colour the vertices of Q_i with colour i, update the list assignment, put the resulting list assignment in $\mathscr{L}$, and restore the lists to the situation at the end of Step 6.

Step 8. For $i = 1, \ldots, 4$, do as follows. If F'_i is connected or non-bipartite, then choose an edge $e^i = u^i v^i$ of F'_i. Otherwise, i.e., if F'_i is disconnected and bipartite, then choose for all $1 \leq j \leq 3$ a vertex u^i_j that is adjacent to all but at most three vertices in $\tilde{I}^i_j$. Here, it is allowed that $\{u^i, v^i\} \cap \{u^{i^*}, v^{i^*}\} \neq \emptyset$ for any two connected graphs F'_i and F'_{i^*} with indices $i^* < i$ and that $u^i_j = u^{i^*}_{j^*}$ for any two disconnected graphs F'_i and F'_{i^*} with indices $i^* \leq i$ and $1 \leq j^* \leq j \leq 3$.

After considering all $1 \leq i \leq 4$, let M be the set that consists of the following vertices: the vertices u^i, v^i for every connected F'_i and the vertices u^i_1, u^i_2, u^i_3 for every disconnected F'_i; note that $|M| \leq 12$. Check whether there exists a colouring of $G[M]$ that respects the lists of the vertices in M, and moreover, that neither colours u^i nor v^i with colour i for each connected F'_i, and that colours none of u^i_1, u^i_2, u^i_3 with colour i for each disconnected F'_i. If so, then call such a colouring *suitable* and M a *suitable branch set*, and continue as described below.

For each connected F'_i, let $\tilde{I}^i(\bar{e}^i)$ be the set of vertices in $(\tilde{I}^i_1 \cup \tilde{I}^i_2 \cup \tilde{I}^i_3) \setminus M$ that are adjacent neither to u^i nor to v^i. For each disconnected F'_i, let $\tilde{I}^i_j(\bar{u}^i_j)$ be the set of vertices in $\tilde{I}^i_j \setminus M$ that are not adjacent to u^i_j. Colour M with a suitable colouring. For $i = 1, \ldots, 4$, if F'_i is connected, then colour all vertices in $\tilde{I}^i(\bar{e}^i)$ according to their lists, and if F'_i is disconnected, then colour all vertices in every $\tilde{I}^i_j(\bar{u}^i_j)$ according to their lists. Afterwards, colour all remaining uncoloured vertices in I^* according to their lists. Only then update the resulting list assignment, put it in $\mathscr{L}$ and restore all lists to the situation at the end of Step 7.

Repeat the above procedure until all suitable branch sets, all their suitable colourings,

all colourings of the vertices in the sets $\tilde{I}^i(\bar{e}^i)$, all colourings of the vertices in the sets $\tilde{I}^i_j(\bar{u}^i_j)$ and all colourings of any remaining uncoloured vertices in I^* have been considered.

Phase 2. Determining if G has a colouring that respects a list assignment in $\mathscr{L}$.

Do as follows for every $\mathcal{L} \in \mathscr{L}$. Determine the set $U_\mathcal{L}$ of vertices of G that have a list of size 1. Colour every vertex in $U_\mathcal{L}$ with the (unique) colour from its list. If there are two adjacent vertices in $U_\mathcal{L}$ coloured alike, then stop considering $\mathcal{L}$. If such vertices do not exist, then update $\mathcal{L}$ and remove $U_\mathcal{L}$ from G. Denote the resulting graph and list assignment by G' and $\mathcal{L}'$, respectively. If all lists in $\mathcal{L}'$ have size at most 2, then apply Theorem 1.1.1. If the union of all lists in $\mathcal{L}'$ has size 3, then apply Lemma 2.1.5. If this leads to a colouring of G' respecting $\mathcal{L}'$, then return `YES`. If after considering all $\mathcal{L} \in \mathscr{L}$ no `YES`-answer has been returned, then return `NO`.

We prove the correctness of our algorithm and analyze its running time in Theorem 2.1.14. For doing this, we need the following lemmas.

Lemma 2.1.7. *For $i = 1, \ldots, 4$ and $j = 1, \ldots, 3$, the number of vertices of I^i_j that is not adjacent to a^i_j in Step 6 of Phase 1 is at most 38.*

Proof. In order to obtain a contradiction, suppose that there exists a pair of indices (i, j) such that $a_0 = a^i_j$ is not adjacent to 39 vertices $b_1, \ldots, b_{39}$ in I^i_j. Consider a vertex b_h for some $1 \leq h \leq 39$. Because b_h is in $N_G(D_i)$, it has a neighbour $a_h \in D_i$; note that $a_h \neq a_0$ by definition. Suppose that a_h is not adjacent to two vertices c and c' of I^i_j that are neighbours of a_0. Then G contains an induced $P_2 + P_3$, where $P_2 = a_h b_h$ and $P_3 = c a_0 c'$. This is not possible. By our choice of a_0, we find that a_h cannot be adjacent to all neighbours of a_0 in I^i_j; otherwise a_h has more neighbours in I^i_j than a_0. We conclude that a_h is adjacent to all but one neighbour of a_0 in I^i_j. By our choice of a_0, this implies that a_h cannot be adjacent to a vertex $b_{h'}$ with $h' \neq h$. Hence, we found that D_i contains vertices $a_1, \ldots, a_{39}$ (where each a_i is adjacent to b_i and to all but one neighbour of a_0 in I^i_j). However, then $|D| \geq |D_i| \geq 40$, which is not possible as D has size at most 39 according to Step 1 of Phase 1. This completes the proof of Lemma 2.1.7. □

Lemma 2.1.8. *For each edge uv in each F'_i, there exists at most one vertex in each $\tilde{I}^i_j$ that is adjacent neither to u nor to v.*

Proof. Suppose that there exists a pair of indices (i, j) such that $\tilde{I}^i_j$ contains two vertices b and b' that are both adjacent neither to u nor to v. Then G contains an induced $P_2 + P_3$, where $P_2 = uv$ and $P_3 = ba^i_j b'$. This is not possible. □

Lemma 2.1.9. *For all $1 \leq i \leq 4$ and all $1 \leq j \leq 3$, if F'_i is a disjoint union of at least two edges, then all but at most one edge of F'_i have at least one end-vertex that is adjacent to all but at most three vertices of $\tilde{I}^i_j$.*

Proof. Suppose that F'_i is a disjoint union of at least two edges. In order to obtain a contradiction, let st and uv be two edges in F'_i, such that each vertex of $\{s, t, u, v\}$ is not adjacent to at least four vertices of $\tilde{I}^i_j$.

We claim that s is adjacent to all but at most one neighbour of u in $\tilde{I}^i_j$, or else that u is adjacent to all but at most one neighbour of s in $\tilde{I}^i_j$. In order to obtain a contradiction, suppose that s is not adjacent to two vertices in $\tilde{I}^i_j \cap N_G(u)$, one of which we call b, and that u is not adjacent to two vertices c, c' in $\tilde{I}^i_j \cap N_G(s)$. Recall that $\tilde{I}^i_j$ is an independent set. Then G contains an induced $P_2 + P_3$, e.g., $P_2 = bu$ and $P_3 = csc'$. This is not possible. Hence, we may assume without loss of generality that s is adjacent to all but at most one neighbour of u in $\tilde{I}^i_j$.

By the same argument as above, we find that s is adjacent to all but at most one neighbour of v in $\tilde{I}^i_j$, or else that v is adjacent to all but at most one neighbour of s in $\tilde{I}^i_j$. Lemma 2.1.8 tells us that $\{u, v\}$ dominates all but at most one vertex of $\tilde{I}^i_j$. Consequently, in the first case, s is adjacent to all but at most three vertices of $\tilde{I}^i_j$, and in the second case v is adjacent to all but at most three vertices of $\tilde{I}^i_j$. Hence, in both cases we find a vertex of $\{s, t, u, v\}$ that is adjacent to all but at most three vertices of $\tilde{I}^i_j$. This is in contradiction with our assumption on $\{s, t, u, v\}$. Hence, we have proven Lemma 2.1.9. □

Lemma 2.1.10. *In Step 7b of Phase 1, each S^i_j contains at most one edge.*

Proof. In Step 7b of Phase 1, a graph F'_i is disconnected and bipartite and has at least three vertices. Then, because G is $(P_2 + P_3)$-free, F'_i is a disjoint union of at least two edges. Then Lemma 2.1.9 tells us that all but at most one edge of F'_i have at least one end-vertex adjacent to all but at most three vertices of $\tilde{I}^i_j$. Hence, a set S^i_j contains at most one edge. □

Lemma 2.1.11. *In Step 8 of Phase 1, each $\tilde{I}^i(\bar{e}^i)$ contains at most one vertex, and each $\tilde{I}^i_j(\bar{u}^i_j)$ contains at most three vertices.*

Proof. Consider a set $\tilde{I}^i(\bar{e}^i)$ for some $e^i = u^i v^i$ in F'_i in Step 8 of Phase 1. By definition, $\tilde{I}^i(\bar{e}^i)$ consists of vertices that are adjacent neither to u^i nor to v^i. We apply Lemma 2.1.8 and find that $\tilde{I}^i(\bar{e}^i)$ contains at most one vertex. We note that each set $\tilde{I}^i_j(\bar{u}^i_j)$ in Step 8 of Phase 1 contains at most three vertices by definition. □

Lemma 2.1.12. *Let $\mathcal{L}$ be a list assignment in the set $\mathscr{L}$ in Phase 2. Then either all lists in $\mathcal{L}$ have size at most 2, or the union of the lists in $\mathcal{L}$ that contain at least two colours has size 3.*

Proof. Let $\mathcal{L} \in \mathscr{L}$. The algorithm has added $\mathcal{L}$ to $\mathscr{L}$ in Step 5, 7a, 7b, or 8, when considering some 4-colouring ϕ_D of D. Note that for all $w \in D$, $L(w) = \{\phi_D(w)\}$, and consequently, $|L(w)| = 1$. The sizes of the lists of the vertices in $V(G) \setminus D$ depend on which step $\mathcal{L}$ was added to $\mathscr{L}$. Hence, we distinguish the following four cases.

Case 1. $\mathcal{L}$ was added to $\mathscr{L}$ in Step 5.
Then there exists a graph F'_i that has at most two vertices. Note that

$$V(G) = D \cup (N_G(D_i) \setminus D) \cup Q_i \cup V(F'_i).$$

Let $w \in V(G) \setminus D$. If $w \in N_G(D_i) \setminus D$, then $i \notin L(w)$, because w is adjacent to a vertex in D_i, which has colour i. If $w \in Q_i$, then $L(w) = \{i\}$ due to Step 5, hence $|L(w)| = 1$. If $w \in V(F'_i)$, then $|L(w)| = 1$ due to Step 5. Hence, the union of the lists in $\mathcal{L}$ that contain at least 2 colours does not contain colour i, and consequently, has size at most 3. This means that either all lists in $\mathcal{L}$ have size at most 2, or the union of the lists in $\mathcal{L}$ that contain at least 2 colours has size 3.

Case 2. $\mathcal{L}$ was added to $\mathscr{L}$ in Step 7a.
Suppose that this happened when considering $1 \leq i \leq 4$. Then F'_i is connected and bipartite. Let B^1_i and B^2_i be the two partition classes of F'_i. We may assume without loss of generality that $\mathcal{L}$ is obtained after the algorithm assigned colour i to every vertex of B^1_i. Note that

$$V(G) = D \cup (N_G(D_i) \setminus D) \cup Q_i \cup B^1_i \cup B^2_i.$$

Let $w \in V(G) \setminus D$. If $w \in N_G(D_i) \setminus D$, then $i \notin L(w)$. If $w \in Q_i$, then $L(w) = \{i\}$ due to Step 7a, hence $|L(w)| = 1$. If $w \in B^1_i$, then $L(w) = \{i\}$ due to Step 7a, hence $|L(w)| = 1$.

If $w \in B_i^2$, then $i \notin L(w)$, because the algorithm updates the list assignment in Step 7a after colouring each vertex of B_i^1 with colour i, and each vertex of B_i^2 is adjacent to at least one vertex of B_i^1 as F_i' is connected. Hence, the union of the lists that contain at least 2 colours does not contain colour i, and consequently, has size at most 3.

Case 3. $\mathcal{L}$ was added to $\mathscr{L}$ in Step 7b.
Suppose that this happened when considering $1 \leq i \leq 4$. Then F_i' is disconnected and bipartite. Because F_i' has at least three vertices and G is $(P_2 + P_3)$-free, this means that F_i' is a disjoint union of at least two edges. Let T_i be the set of vertices of F_i' that have list $\{i\}$ in $\mathcal{L}$. Let U_i denote the union of all vertices in the edges of $S_1^i \cup S_2^i \cup S_3^i$; here we let $S_j^i = \emptyset$ if $\tilde{I}_j^i = \emptyset$. Let $W_i = V(F_i') \setminus (T_i \cup U_i)$. Note that

$$V(G) = D \cup (N_G(D_i) \setminus D) \cup Q_i \cup T_i \cup U_i \cup W_i.$$

Let $w \in V(G) \setminus D$. If $w \in N_G(D_i) \setminus D$, then $i \notin L(w)$. If $w \in Q_i$, then $L(w) = \{i\}$ due to Step 7b, hence $|L(w)| = 1$. If $w \in T_i$, then $L(w) = \{i\}$ by definition, hence $|L(w)| = 1$. If $w \in U_i$, then $|L(w)| = 1$ due to Step 7b. If $w \in W_i$, then $i \notin L(w)$, because i is an end-vertex of an edge, the other end-vertex of which has colour i according to Step 7b. Hence, the union of the lists that contain at least 2 colours does not contain colour i, and consequently, has size at most 3.

Case 4. $\mathcal{L}$ was added to $\mathscr{L}$ in Step 8.
Then the algorithm has obtained $\mathcal{L}$ starting from some suitable branch set M and some suitable colouring of $G[M]$. Note that

$$\begin{aligned} V(G) &= D \cup I^* \cup \tilde{I}_1^1 \cup \tilde{I}_2^1 \cup \tilde{I}_3^1 \cup \tilde{I}_1^2 \cup \tilde{I}_2^2 \cup \tilde{I}_3^2 \cup \tilde{I}_1^3 \cup \tilde{I}_2^3 \cup \tilde{I}_3^3 \cup \tilde{I}_1^4 \cup \tilde{I}_2^4 \cup \tilde{I}_3^4 \\ &= D \cup (I_1^i \setminus \tilde{I}_1^i) \cup (I_2^i \setminus \tilde{I}_2^i) \cup (I_3^i \setminus \tilde{I}_3^i) \cup \tilde{I}_1^i \cup \tilde{I}_2^i \cup \tilde{I}_3^i \cup V(F_i) \quad \text{for } i = 1, \ldots 4. \end{aligned}$$

Let $w \in V(G) \setminus D$. If $w \in I^*$, then $|L(w)| = 1$ due to Step 8. If $w \in \tilde{I}^i(\bar{e}^i) \cup \{u^i, v^i\}$ for some edge $e^i = u^i v^i$ with $u^i, v^i \in M$, then $|L(w)| = 1$ due to Step 8. If $w \in \tilde{I}_j^i(\bar{u}_j^i) \cup \{u_j^i\}$ for some vertex $u_j^i \in M$, then $|L(w)| = 1$ due to Step 8 as well. In all other cases, w belongs to a set $\tilde{I}_j^i$ for at least one $1 \leq i \leq 4$ and some $1 \leq j \leq 3$. If F_i' is connected or non-bipartite, then w is adjacent to a vertex in M, which is an end-vertex of some chosen edge $e^i = u^i v^i$. If F_i' is disconnected and bipartite, then w is adjacent to a vertex in M, which is some chosen vertex u_j^i. In both cases, this neighbour of w is coloured with a colour not equal to i, because the colouring of M is assumed to be suitable. Because

$i \notin L(w)$ by definition, this means that $|L(w)| \leq 2$. Hence, all lists of $\mathcal{L}$ have size at most 2. This completes the proof of Lemma 2.1.12. □

Lemma 2.1.13. *If $\mathscr{L}$ contains a list assignment that is respected by some colouring of G, then the algorithm returns* `YES`.

Proof. Let $\mathcal{L} \in \mathscr{L}$ be a list assignment that is respected by some colouring ϕ of G. Because $\mathcal{L}$ is respected by ϕ, colouring every vertex in $U_{\mathcal{L}}$ with the (unique) colour from its list does not result in two adjacent vertices with the same colour; each $u \in U_{\mathcal{L}}$ receives colour $\phi(u)$.

We remove all vertices in $U_{\mathcal{L}}$ from G and denote the resulting graph and list assignment by G' and $\mathcal{L}'$, respectively. Let ϕ' be the restriction of ϕ to $V(G')$. Then $\mathcal{L}'$ is respected by ϕ'.

Lemma 2.1.12 tells us that either all lists in $\mathcal{L}$ have size at most 2, or the union of the lists in $\mathcal{L}$ that contain at least two colours has size 3. Consequently, either all lists in $\mathcal{L}'$ have size at most 2, or the union of the lists in $\mathcal{L}'$ has size 3. In the first case the algorithm applies Theorem 1.1.1. In the second case the algorithm applies Lemma 2.1.5. In both cases the algorithm will conclude that G' has a colouring that respects $\mathcal{L}'$ (because ϕ' is such a colouring). Hence, it will return `YES`. This completes the proof of Lemma 2.1.13. □

Theorem 2.1.14. *The* 4-Colouring *problem can be solved in polynomial time for $(P_2 + P_3)$-free graphs.*

Proof. Let $G = (V, E)$ be a $(P_2 + P_3)$-free graph with n vertices. Recall that we may assume that G has minimum degree at least 4 due to Proposition 1.2.2.

Correctness. We start with proving that our algorithm is correct, i.e., that its output is `YES` if and only if G has a 4-colouring.

First suppose that the output of our algorithm is `YES`. Note that such an output only occurs in Phase 2. Hence, a graph G' has a colouring respecting a list assignment $\mathcal{L}'$, where G' and $\mathcal{L}'$ are obtained by removing all vertices from G that have a list of size 1, i.e., belong to a set $U_{\mathcal{L}}$ for some $\mathcal{L} \in \mathscr{L}$. Colouring the vertices in $U_{\mathcal{L}}$ with the (unique) colour from their list does not yield two adjacent vertices coloured alike, as otherwise the algorithm would have stopped considering $\mathcal{L}$ and thus would not have modified $\mathcal{L}$ into $\mathcal{L}'$. Because of this and because the algorithm updates $\mathcal{L}$ before removing $U_{\mathcal{L}}$, we can

extend the colouring of G' to a colouring of G by assigning every vertex that is not in G', i.e., that belongs to $U_{\mathcal{L}}$, the unique colour in its list. Because every list in every list assignment of $\mathscr{L}$ is a subset of $\{1, 2, 3, 4\}$, the resulting colouring is a 4-colouring of G.

Now suppose that G has a 4-colouring ϕ. Lemma 2.1.4 tells us that G has a dominating set of size at most 39. Consequently, our algorithm will find such a dominating set in Step 1. Because G is 4-colourable, $G[D]$ is 4-colourable. Hence, the algorithm does not return `NO` in Step 2. Instead it considers each 4-colouring of $G[D]$ including the 4-colouring ϕ_D of $G[D]$ with $\phi_D(a) = \phi(a)$ for all $a \in D$.

In Step 3, the algorithm checks if a partition into three (possibly empty) independent sets I_1^i, I_2^i, I_3^i of $N_G(D_i) \setminus D$ exists for $i = 1, \ldots, 4$. Because all vertices in each $N_G(D_i) \setminus D$ are adjacent to a vertex in D_i, i.e., to a vertex a with colour $\phi(a) = i$ and because ϕ is a 4-colouring, we find that the restriction of ϕ to the vertices of $N_G(D_i) \setminus D$ is a 3-colouring of $G[N_G(D_i) \setminus D]$. This means that $N_G(D_i) \setminus D$ can be partitioned into three (possibly empty) independent sets corresponding to the three colour classes of this 3-colouring. Hence, the algorithm will find independent sets I_1^i, I_2^i, I_3^i that form a partition of $N_G(D_i) \setminus D$ for $i = 1, \ldots, 4$. Note that these four partitions into three independent sets may be different than the ones induced by ϕ. This does not matter; for our correctness proof we only need the algorithm to find *some* partition I_1^i, I_2^i, I_3^i of $N_G(D_i) \setminus D$ for $i = 1, \ldots, 4$, and the fact that the restriction of ϕ to $N_G(D_i) \setminus D$ is a 3-colouring ensures that this is going to happen.

In Step 4, the algorithm determines for $i = 1, \ldots, 4$, the set Q_i that consists of all isolated vertices in F_i and constructs the graph F_i' obtained from F_i by removing the vertices of Q_i.

In Step 5, the algorithm checks whether there exists a graph F_i' that has at most two vertices for some $1 \leq i \leq 4$. If so, then the algorithm considers each possible colouring of these vertices, so including the colouring of F_i' that corresponds to ϕ. In addition, it colours each vertex of Q_i with colour i. We may assume without loss of generality that $\phi(u) = i$ for all $u \in Q_i$. If $\phi(u) \neq i$ for some $u \in Q_i$, then we may redefine ϕ by setting $\phi(u) := i$ for the following reason. All neighbours of u in G belong to $N_G(D_i) \setminus D$, i.e., are adjacent to a vertex in D_i, which ϕ has assigned colour i, and as such, no neighbour of u is assigned colour i by ϕ. Hence, the algorithm goes to Phase 2 with a set $\mathscr{L}$ of list assignments that include a list assignment $\mathcal{L}$ that is respected by ϕ. Then it will return `YES` due to Lemma 2.1.13.

From now on, suppose that every F'_i has at least three vertices. We observe that F'_i has at least two edges, because F'_i contains no isolated vertices.

In Step 6, the algorithm determines the set $\tilde{I}^i_j$ for $i = 1, \ldots, 4$ and $j = 1, \ldots, 3$. It also determines the set I^* that consists of all vertices of $N_G(D) \setminus D$ that are not in some set $\tilde{I}^i_j$.

In Step 7, the algorithm checks for $i = 1, \ldots, 4$ whether F'_i is connected and bipartite, or whether F'_i is disconnected and bipartite. Here, we observe that a graph F'_i may be non-bipartite, and in that case the algorithm does not process F'_i in Step 7. Otherwise, after processing F'_i, the algorithm will place one or more new list assignments in $\mathscr{L}$. Then $\mathscr{L}$ may contain a list assignment that is respected by ϕ in the following two cases.

The first case is in Step 7a, when F'_i is connected and bipartite for some $1 \leq i \leq 4$, such that $\phi(u) = i$ for every vertex u in one partition class of F'_i. Because the algorithm considers both partition classes of F'_i, one of the two created list assignments that are to be placed in $\mathscr{L}$ is respected by ϕ. Consequently, the algorithm will return `YES` due to Lemma 2.1.13.

The second case *may* be in Step 7b. We first note that in this step the algorithm colours a vertex of all but at most one edge with colour i due to Lemma 2.1.10. Hence, if F'_i is a disconnected and bipartite graph, in which all but at most one edge contain a vertex coloured i by ϕ, and moreover, such that the algorithm picks exactly those vertices u with $\phi(u) = i$ to get colour i for some $1 \leq j \leq 3$ with $\tilde{I}^i_j \neq \emptyset$, then the resulting list assignment is respected by ϕ. In that case, the algorithm will return `YES` due to Lemma 2.1.13. We emphasize that in this step the algorithm considers at most three possible assignments of colour i to vertices in F'_i, namely one assignment for each nonempty $\tilde{I}^i_j$. If in each case the algorithm assigns colour i to one or more different vertices than the ones that are coloured i by ϕ, then the resulting list assignment that is placed in $\mathscr{L}$ will not be respected by ϕ. We take this into account when analyzing Step 8.

Assume that the algorithm has not yet placed a list assignment in $\mathscr{L}$ that is respected by ϕ.

In Step 8, the algorithm creates list assignments by processing the graphs F'_i for $i = 1, \ldots, 4$ in sequential order. Let $1 \leq i \leq 4$ and consider a graph F'_i. In line with Step 8, we distinguish between the following two cases.

Case 1. Suppose that F'_i is connected or non-bipartite.

If ϕ colours an end-vertex of every edge in F_i' with colour i, then F_i' must be a bipartite graph; the set of vertices coloured i and the set of vertices not coloured i form the two partition classes. In that case, the algorithm has already placed, namely in Step 7a or 7b, a list assignment in $\mathscr{L}$ that is respected by ϕ. This is in contradiction with our assumption that the algorithm has not yet done this. Hence, F_i' contains at least one edge $e = uv$ with $\phi(u) \neq i$ and $\phi(v) \neq i$.

Case 2. Suppose that F_i' is disconnected and bipartite.
Then, because G is $(P_2 + P_3)$-free and F_i' contains no isolated vertices, F_i' is a disjoint union of edges. Because F_i' has at least three vertices, this means that F_i' has at least two edges. The algorithm considers the sets $\tilde{I}_j^i$ for $j = 1, \ldots, 3$ in sequential order. Let $1 \leq j \leq 3$ and consider a set $\tilde{I}_j^i$. Let Z_i be the set of vertices in F_i' that are adjacent to all but at most three vertices of $\tilde{I}_j^i$. By Lemma 2.1.9 we find that $Z_i \neq \emptyset$, because all but at most one edge in F_i' contains a vertex adjacent to all but at most three vertices of $\tilde{I}_j^i$, and F_i' has at least two edges. Suppose that every vertex in Z_i is coloured i by ϕ. Then every one of those edges that contains a vertex adjacent to all but at most three vertices of $\tilde{I}_j^i$ contains exactly one vertex of Z_i, because two vertices that are both coloured with colour i cannot be adjacent. However, in that case, the algorithm would already have placed a list assignment in $\mathscr{L}$ that is respected by ϕ, namely in Step 7b. Hence, Z_i contains a vertex u with $\phi(u) \neq i$.

By our case analysis we find that there exists a suitable branch set M, such that the restriction of ϕ to M is a suitable colouring of $G[M]$. Our algorithm will detect this in one of the branches in Step 8. At some point it will also colour the vertices in each $\tilde{I}^i(\bar{e}^i)$, the vertices in each $\tilde{I}_j^i(\bar{u}_j^i)$ and all remaining uncoloured vertices in I^* according to ϕ, because it considers all possibilities exhaustively. We conclude that after Step 8 has finished, the algorithm has put a list assignment $\mathcal{L}$ in $\mathscr{L}$ that is respected by ϕ. Then, by Lemma 2.1.13, it will return `YES`. This completes our correctness proof.

Running time analysis. We prove that Phase 1 can be performed in polynomial time and leads to a set $\mathscr{L}$ of polynomial size by showing that each step of Phase 1 performs in polynomial time and each time the algorithm places a polynomial number of list assignments in $\mathscr{L}$.

The algorithm performs Step 1 in $O(n^{39})$ time by brute force. In Step 2 we find at most $4^{|D|} \leq 4^{39}$ different 4-colourings of $G[D]$. The algorithm performs Step 3 in polynomial time by applying Lemma 2.1.6 at most four times. It performs Step 4 in

linear time, because it only has to detect the isolated vertices in each F_i. Afterwards, it has immediately obtained the graphs F'_i.

The algorithm performs Step 5 in linear time; in addition to considering at most one colouring of some set Q_i of isolated vertices, it needs to consider at most 3^2 colourings of a set of at most two vertices that can be detected in linear time; note that each vertex of such a set has indeed a list of size at most 3, because it is adjacent to an already coloured vertex in D and the algorithm updated the list assignment in Step 3. If the algorithm starts Phase 2 directly after Step 5, then we have a set $\mathscr{L}$ of size at most $4^{39} \cdot 3^2$, which is a constant. Otherwise, we must continue our running time analysis with Step 6.

For $i = 1, \ldots, 4$ and $j = 1, \ldots, 3$, the algorithm determines a required vertex $a^i_j \in D_i$ in Step 6 in polynomial time. By Lemma 2.1.7, each a^i_j is adjacent to all but at most 38 vertices of I^i_j, hence the set I^* has at most $4 \cdot 3 \cdot 38 = 456$ vertices.

The algorithm performs Step 7a in polynomial time, because it only has to check whether the graphs F'_i are connected and bipartite, and if this is the case, then it only has to construct two list assignments, each of which corresponds to the partition class of F'_i whose vertices are coloured with colour i; note that the vertices in Q_i are coloured in only one way. Hence, it places at most $4 \cdot 2 = 8$ different list assignments in $\mathscr{L}$.

The algorithm performs Step 7b in polynomial time. This can be seen as follows. The algorithm checks in polynomial time whether a graph F'_i is disconnected and bipartite, i.e., whether F'_i is a disjoint union of at least two edges. If so, then it places at most $4 \cdot 3 \cdot 3^2 = 108$ different list assignments in $\mathscr{L}$, because each set S^i_j contains at most one edge according to Lemma 2.1.10, and the vertices in Q_i are coloured in only one way for each $1 \leq j \leq 3$ with nonempty $\tilde{I}^i_j$.

In Step 8 the algorithm considers in worst case all edges e^i in every F'_i that is connected or non-bipartite, all colourings of their end-vertices u^i and v^i, all colourings of the vertices in $\tilde{I}^i(\bar{e}^i)$, all possible triples of vertices u^i_1, u^i_2, u^i_3 in every F'_i that is a disjoint union of at least two edges, all the colourings of these triples, all colourings of the vertices in $\tilde{I}^i_1(\bar{u}^i_1) \cup \tilde{I}^i_2(\bar{u}^i_2) \cup \tilde{I}^i_3(\bar{u}^i_3)$ and all colourings of the remaining uncoloured vertices in I^*. The number of edges e^i is at most n^2. The number of colourings to be considered for the two end-vertices of an edge e^i is at most 3^2. The number of colourings to be considered for the vertices in a set $\tilde{I}^i(\bar{e}^i)$ is at most 3, because $\tilde{I}^i(\bar{e}^i)$ has size at most 1 due to Lemma 2.1.11. The number of triples of vertices u^i_1, u^i_2, u^i_3 is at most n^3. The number of colourings to be considered for each such triple is at most 3^3. The number of colourings

to be considered for the vertices in $\tilde{I}_1^i(\bar{u}_1^i) \cup \tilde{I}_2^i(\bar{u}_2^i) \cup \tilde{I}_3^i(\bar{u}_3^i)$ is at most 9^3, because each $\tilde{I}_j^i(\bar{u}_j^i)$ has size at most 3 due to Lemma 2.1.11. Recall that I^* has at most 152 vertices. Hence, the set of remaining uncoloured vertices of I^* in some branch has at most 3^{152} colourings. This means that the total number of list assignments obtained in Step 9 is at most $p(n) = n^2 \cdot 3^2 \cdot 3 \cdot n^3 \cdot 3^3 \cdot 9^3 \cdot 3^{152}$. Hence, if we start Phase 2 after Step 9, then we have a set $\mathscr{L}$ of size at most $4^{39}(8 + 108 + p(n))$, which is polynomial. We also conclude that Phase 1 can be performed in polynomial time.

In Phase 2, the algorithm preprocesses each $\mathcal{L} \in \mathscr{L}$ in quadratic time; first it colours every vertex in $U_{\mathcal{L}}$ with the unique colour in its list, then it checks whether there exist two vertices in $U_{\mathcal{L}}$ that are coloured alike, and if not, it updates $\mathcal{L}$ and then removes all vertices in $U_{\mathcal{L}}$ from G. Afterwards, Lemma 2.1.12 implies that the algorithm can apply (in polynomial time) Theorem 1.1.1 or else Lemma 2.1.5 for every resulting graph G' and list assignment $\mathcal{L}'$. Because $\mathscr{L}$ has polynomial size as we deduced above, this means that our algorithm can perform Phase 2 in polynomial time. This completes the proof of Theorem 2.1.14. □

2.2 4-Colouring for P_8-free Graphs

In this section we prove that 4-Colouring is NP-complete for P_8-free graphs. We use a reduction from the 3-Satisfiability problem. Recall that this problem is NP-complete [63]. We consider an arbitrary instance I of 3-Satisfiability that has variables $\{x_1, x_2, \ldots, x_n\}$ and clauses $\{C_1, C_2, \ldots, C_m\}$ and define a graph G_I. Next we show that G_I is P_8-free and that G_I is 4-colourable if and only if I has a satisfying truth assignment.

Here is the construction that defines G_I.

- For each clause C_j we introduce a 7-vertex cycle with vertex set

$$\{b_{j,1}, b_{j,2}, c_{j,1}, c_{j,2}, c_{j,3}, d_{j,1}, d_{j,2}\}$$

and edge set

$$\{b_{j,1}c_{j,1}, c_{j,1}d_{j,1}, d_{j,1}c_{j,2}, c_{j,2}d_{j,2}, d_{j,2}c_{j,3}, c_{j,3}b_{j,2}, b_{j,2}b_{j,1}\}.$$

We say that these vertices are of b-type, c-type and d-type, respectively. They induce disjoint 7-cycles (i.e., cycles on seven vertices) in G_I which we call *clause-*

components.

- For each variable x_i we introduce a copy of a K_2, i.e., two vertices joined by an edge $x_i\overline{x}_i$. We say that both x_i and $\overline{x}_i$ are of x-type, and we call the corresponding disjoint K_2s in G_I *variable-components*.

- For every clause C_j we fix an arbitrary order of its variables $x_{i_1}, x_{i_2}, x_{i_3}$. For $h = 1, 2, 3$ we either add the edge $c_{j,h}x_{i_h}$ or the edge $c_{j,h}\overline{x}_{i_h}$ depending on whether x_{i_h} or $\overline{x}_{i_h}$ is a literal in C_j, respectively.

- We add an edge between any x-type vertex and any b-type vertex. We also add an edge between any x-type vertex and any d-type vertex.

- We introduce one additional new vertex a which we make adjacent to all b-type, c-type and d-type vertices.

See Figure 2.4 for an example of a graph G_I. In this example C_1 is a clause with ordered literals $x_{i_1}, \overline{x}_{i_2}, x_{i_3}$ and C_m is a clause with ordered literals $\overline{x}_1, x_{i_3}, x_n$. The thick edges indicate the connections between the literal vertices and the c-type vertices of the clause gadgets. We omitted the indices from the labels of the clause gadget vertices to increase the visibility.

We now prove two lemmas. Lemma 2.2.1 shows that the graph G_I is P_8-free (in fact it shows a slightly stronger statement as this will be of use for us in Section 3.3). In Lemma 2.2.2 we prove that G_I admits a 4-colouring if and only if I has a satisfying truth assignment.

Lemma 2.2.1. *The graph G_I is P_8-free. Moreover, every induced path in G_I on seven vertices contains a.*

Proof. Let P be an induced path in G_I. We show that G_I is P_8-free by proving that P has at most seven vertices. We also show that P contains a in the case that P has exactly seven vertices. We distinguish a number of cases and subcases.

Case 1. $a \notin V(P)$.

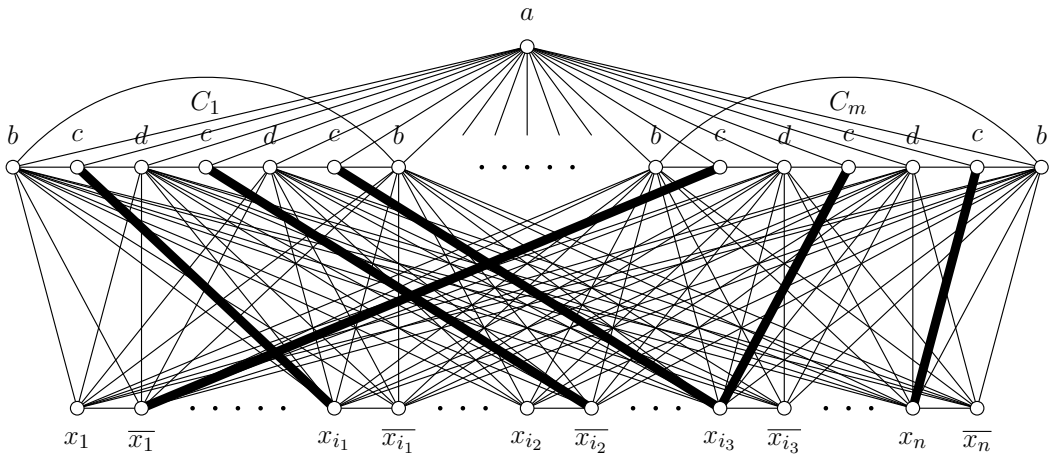

Figure 2.4: The graph G_I in which clauses $C_1 = \{x_{i_1}, \overline{x}_{i_2}, x_{i_3}\}$ and $C_m = \{\overline{x}_1, x_{i_3}, x_n\}$ are illustrated.

Case 1a. P contains no x-type vertex.
This means that P is contained in one clause-component, which is isomorphic to an induced 7-cycle. Consequently, P has at most six vertices.

Case 1b. P contains exactly one x-type vertex.
Let x_i be this vertex. Then P contains vertices of at most two clause-components. Since x_i is adjacent to all b-type and d-type vertices, we then find that P contains at most two vertices of each of the clause-components. Hence P has at most five vertices.

Case 1c. P contains exactly two x-type vertices.
First suppose that these vertices are adjacent, say P contains x_i and $\overline{x}_i$. By the same reasoning as above we find that P has at most four vertices.

Now suppose the two x-type vertices of P are not adjacent. By symmetry, we may assume that P contains x_h and x_i. If P contains no b-type vertex and no d-type vertex, then there is no subpath in P from x_h to x_i, a contradiction. If P contains two or more vertices of b-type and d-type, then P contains a cycle, another contradiction. Hence P contains exactly one vertex z that is of b-type or d-type. Then $x_h z x_i$ is a subpath in P. If both x_h and x_i have a neighbour in $V(P)\backslash\{z\}$, then this neighbour must be of c-type, and consequently an end-vertex of P (because a c-type vertex is adjacent to only one x-type vertex). Hence P contains at most five vertices.

Case 1d. P contains at least three x-type vertices.
Then P contains no b-type vertex and no d-type vertex, because such vertices would have degree 3 in P. However, on the other hand the three x-type vertices come from at least two different variable-components. Since any c-type vertex is adjacent to exactly one x-type vertex, P must contain a b-type or d-type vertex to connect the x-type vertices of P to one another. We conclude that this subcase is not possible.

Case 2. $a \in V(P)$.
First suppose a is an end-vertex of P. If $|V(P)| \geq 2$ then P contains exactly one vertex that is of b-type, c-type or d-type. Since every x-type vertex is adjacent to only one other x-type vertex, this means that P can have at most four vertices.

Now suppose a is not an end-vertex of P. Then P contains exactly two vertices that are of b-type, c-type or d-type. By the same arguments as above, we then find that P has at most seven vertices. This completes the proof of Lemma 2.2.1. □

Lemma 2.2.2. *The graph G_I is 4-colourable if and only if I has a satisfying truth assignment.*

Proof. Suppose we have a 4-colouring of G_I with colours $\{1, 2, 3, 4\}$. We may assume without loss of generality that a has colour 1, that $b_{1,1}$ has colour 3 and that $b_{1,2}$ has colour 4. This implies that all x-type vertices have a colour from $\{1, 2\}$. Furthermore, for $i = 1, \ldots, n$, if x_i has colour 1 then $\overline{x}_i$ has colour 2, and vice versa. Hence we find that all b-type and d-type vertices have a colour from $\{3, 4\}$. Then by symmetry we may assume that every $b_{j,1}$ has colour 3 and every $b_{j,2}$ has colour 4. This means that every $c_{j,1}$ has a colour from $\{2, 4\}$, every $c_{j,2}$ has a colour from $\{2, 3, 4\}$ and every $c_{j,3}$ has a colour from $\{2, 3\}$. Now suppose there is a clause C_j with each of its three literals coloured by colour 2. Then $c_{j,1}$ must have colour 4 and $c_{j,3}$ must have colour 3. Consequently, $d_{j,1}$ has colour 3 and $d_{j,2}$ has colour 4. Then $c_{j,2}$ cannot have a colour in a proper 4-colouring of G_I. Hence this is not possible and we find that at least one literal in every clause is coloured by colour 1. This means we can define a truth assignment that sets a literal to `FALSE` if the corresponding x-type vertex has colour 2, and to `TRUE` otherwise. So a 4-colouring of G_I implies a satisfying truth assignment for I.

For the converse, suppose I has a satisfying truth assignment. We use colour 1 to colour the x-type vertices representing the true literals and colour 2 for the false literals. Since each clause contains at least one true literal, we note that we can colour $c_{j,1}$, $c_{j,2}$

and $c_{j,3}$ and also all other remaining vertices in a straightforward way. This implies a 4-colouring for G_I and completes the proof of Lemma 2.2.2. □

By Lemmas 2.2.1 and 2.2.2 we obtain the main result of this section.

Theorem 2.2.3. *The* 4-Colouring *problem is* NP*-complete for* P_8*-free graphs.*

2.3 4-Colouring for (C_3, P_{164})-free Graphs

In Section 2.4, we determine for any fixed girth $g \geq 4$ a value $r(g)$ such that every $P_{r(g)}$-free graph with girth at least g is 3-colourable. We tried to make $r(g)$ as large as possible. In contrast, in this section we show that 4-Colouring is NP-complete for (C_3, P_{164})-free graphs. As the problem is clearly in NP, we are left to prove NP-hardness. The NP-hardness reductions for k-Colouring for P_r-free graphs involve the presence of triangles in the gadgets. For example, for the gadget used in the proof for the NP-complete result of 4-Colouring on P_8-free graphs in Section 2.2, every variable-component $x_i\overline{x}_i$ together with every b-type vertex or with every d-type vertex form a triangle. In that case, all the b-type and d-type vertices will be given a colour different from x_i and $\overline{x}_i$ for all i. Hence, the main task is to design a triangle-free gadget. Then we replace a number of edges of a gadget that is used in the NP-hardness reductions by this triangle-free gadget. We first present this gadget and its properties. We then show how to incorporate it in our final gadget that proves our NP-hardness reduction, which is from the problem Not-All-Equal 3-Satisfiability with un-negated literals only. Recall that this problem is NP-complete [88].

The edge-replacing gadget

We define four independent sets A, B, C and D with $|A| = |B| = 10$ and $|C| = |D| = 5$. We add an edge between every vertex in A and every vertex in B. We also add an edge between every vertex in C and every vertex in D. This leads to two vertex-disjoint complete bipartite graphs with partition classes A, B and C, D, respectively.

For every subset $A_i \subseteq A$ of five vertices, we create two cycles Q_i and T_i, each on 5 new vertices. We say that Q_i is a Q-cycle and that T_i is a T-cycle. We add 5 edges between the vertices of Q_i and A_i in such a way that these edges form a matching. We do the same for T_i and A_i. We also add 5 matching edges between the vertices of Q_i

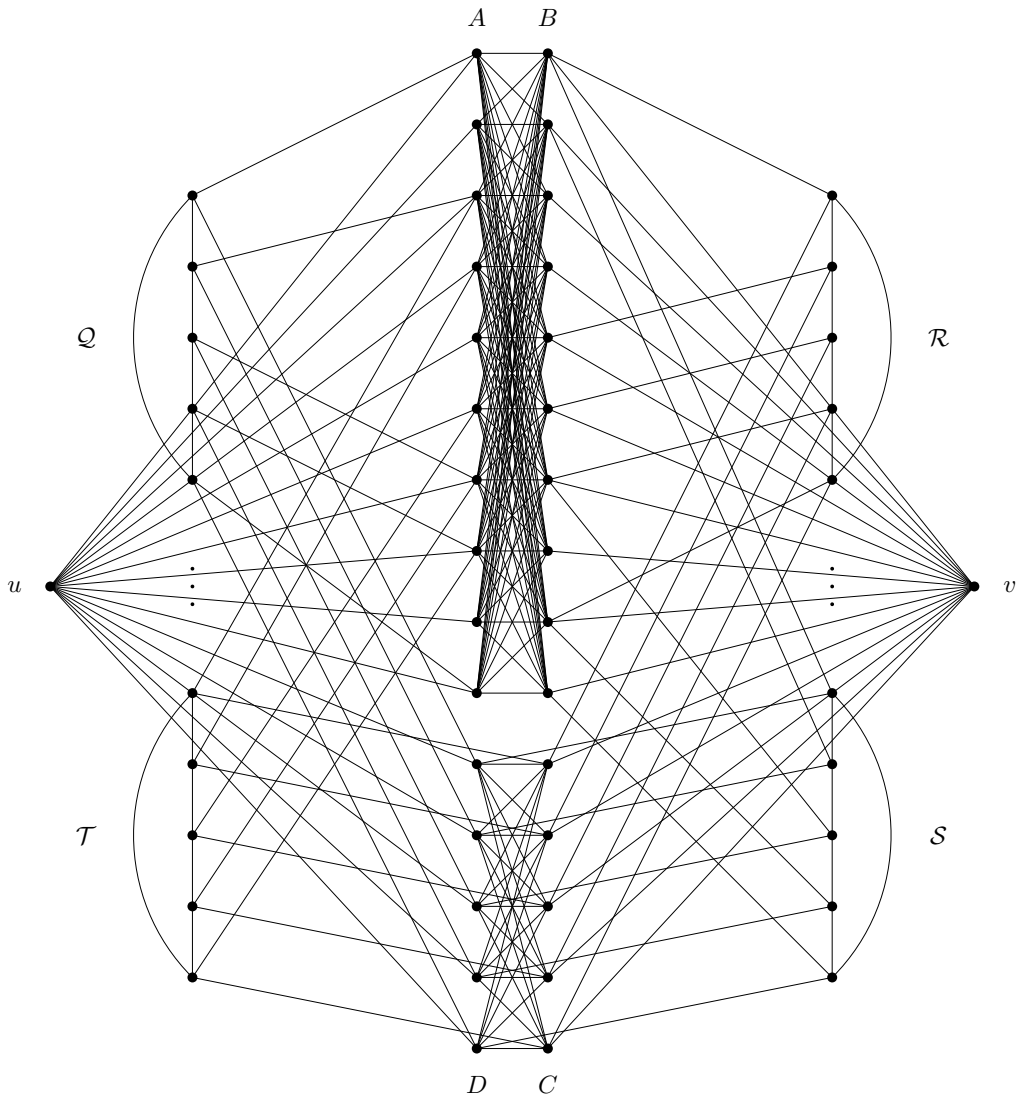

Figure 2.5: The graph F; only one Q-cycle, R-cycle, S-cycle and one T-cycle are displayed.

and D, and do the same for T_i and C. We let $\mathcal{Q}$ and $\mathcal{T}$ denote the set of all $\binom{10}{5}(= 252)$ Q-cycles and $\binom{10}{5}(= 252)$ T-cycles, respectively. Similarly, we define two sets $\mathcal{R}$ and $\mathcal{S}$ of all $\binom{10}{5}(= 252)$ R-cycles and $\binom{10}{5}(= 252)$ S-cycles, respectively. Here, each R-cycle and each S-cycle correspond to exactly one subset $B_i \subseteq B$ of five vertices. For each such B_i there are matchings between its vertices and the vertices in its R-cycle and S-cycle, respectively. There is also a matching between the vertices of each R-cycle and C, and between the vertices of each S-cycle and D. Finally, we add a new vertex u adjacent to every vertex of $A \cup D$, and a new vertex v adjacent to every vertex of $B \cup C$. Since any two neighbours of every vertex are not adjacent in the resulting graph called F, we obtain that F is C_3-free; see Figure 2.5.

Lemma 2.3.1 states some useful properties of F that we will use later on.

Lemma 2.3.1. *The graph F is 4-colourable. Moreover, u and v are given different colours in every 4-colouring of F.*

Proof. We first show that F is 4-colourable. We choose a vertex $c \in C$ and a vertex $d \in D$, which we give colour 1 and 2, respectively. We give each vertex of $(A \cup D) \setminus \{d\}$ colour 3 and each vertex of $(B \cup C) \setminus \{c\}$ colour 4. We give u colour 1 and v colour 3. Consider a Q-cycle. We give its vertex adjacent to d colour 1 and colour its other four vertices by colours 2 and 4. Consider a T-cycle. We give its vertex adjacent to c colour 4 and colour its other four vertices by colours 1 and 2. By symmetry, we can also give the vertices of every R-cycle and S-cycle an appropriate colour such that in the end we have obtained a 4-colouring of F.

We now prove that u and v are not coloured alike in every 4-colouring of F. Let ϕ be a 4-colouring of F. First suppose that $|\phi(C)| \geq 2$ and $|\phi(D)| \geq 2$. Because C and D are partition classes of a complete bipartite graph, we then may without loss of generality assume that $\phi(C) = \{1, 4\}$ and $\phi(D) = \{2, 3\}$. This means that u can only get a colour from $\{1, 4\}$ and v can only get a colour from $\{2, 3\}$. Hence, u and v are not coloured alike.

In the remaining case, we assume without loss of generality that $|\phi(D)| = 1$. If $|\phi(A)| = 4$, then we cannot colour a vertex in B. Hence, $|\phi(A)| \leq 3$. If $|\phi(A)| = 3$, then every vertex of $B \cup \{u\}$ receives the same colour. Because v is adjacent to the vertices of B, this means that v must receive a different colour.

Suppose that $|\phi(A)| \leq 2$. Then A contains a subset A_i of five vertices that are coloured alike, say with colour 3. We observe that u does not get colour 3. Consider the

Q-cycle corresponding to A_i. Its five vertices can neither be coloured with colour 3 nor with the colour in $\phi(D)$. Because this cycle needs at least three colours, this means that $\phi(D) = \phi(A_i) = \{3\}$. Because every vertex of A is adjacent to every vertex of B, colour 3 is not used on B, so $|\phi(B)| \leq 3$.

First suppose that $|\phi(B)| \leq 2$. Then B contains a subset B_j of five vertices that are coloured alike, say with colour 4. We consider the S-cycle corresponding to B_j. Because every vertex of this cycle is not only adjacent to a vertex of B_j with colour 4 but also adjacent to a vertex of D with colour 3, we find that only colours 1 and 2 are available to colour its five vertices. This is not possible. Hence, $|\phi(B)| = 3$, so $\phi(B) = \{1, 2, 4\}$. This means that v must receive colour 3, whereas we already deduced that u does not get colour 3. This completes the proof of Lemma 2.3.1. □

Using the edge-replacing gadget

We now present our reduction for showing that 4-Colouring is NP-complete for the class of (C_3, P_{164})-free graphs. This reduction is from the Not-All-Equal 3-Satisfiability problem with un-negated literals only. Recall that this problem is NP-complete [88]. We consider an arbitrary instance I of Not-All-Equal 3-Satisfiability with un-negated literals only that has variables $\{x_1, x_2, \ldots, x_n\}$ and clauses $\{C_1, C_2, \ldots, C_m\}$. From I we first construct a graph G_I^*. We then explain how to incorporate our edge-replacing gadget F. This will yield a graph G_I'. In Lemma 2.3.2 and Lemma 2.3.4 we will show that G_I' is (C_3, P_{164})-free. In Lemma 2.3.6 we will show that G_I' is 4-colourable if and only if I has a satisfying truth assignment in which each clause contains at least one true literal and at least one false literal.

Here is the construction that defines the graph G_I^*.

- For each clause C_j we introduce a gadget with vertex set

$$\{a_{j,1}, a_{j,2}, a_{j,3}, b_{j,1}, b_{j,2}, c_{j,1}, c_{j,2}, c_{j,3}, d_{j,1}, d_{j,2}\}$$

  and edge set

$$\{a_{j,1}c_{j,1}, a_{j,2}c_{j,2}, a_{j,3}c_{j,3}, b_{j,1}c_{j,1}, c_{j,1}d_{j,1}, d_{j,1}c_{j,2}, c_{j,2}d_{j,2}, d_{j,2}c_{j,3}, c_{j,3}b_{j,2}, b_{j,2}b_{j,1}\},$$

and a disjoint gadget called the *copy* that has vertex set

$$\{a'_{j,1}, a'_{j,2}, a'_{j,3}, b'_{j,1}, b'_{j,2}, c'_{j,1}, c'_{j,2}, c'_{j,3}, d'_{j,1}, d'_{j,2}\}$$

and edge set

$$\{a'_{j,1}c'_{j,1}, a'_{j,2}c'_{j,2}, a'_{j,3}c'_{j,3}, b'_{j,1}c'_{j,1}, c'_{j,1}d'_{j,1}, d'_{j,1}c'_{j,2}, c'_{j,2}d'_{j,2}, d'_{j,2}c'_{j,3}, c'_{j,3}b'_{j,2}, b'_{j,2}b'_{j,1}\}.$$

We say that all these vertices (so, including the vertices in the copy) are of a-type, b-type, c-type and d-type, respectively. We call the gadget and its copy *clause gadgets*.

- Every variable x_i is represented by a vertex in G_I^*, and we say that these vertices are of x-type.

- For every clause C_j we fix an arbitrary order of its variables $x_{i_1}, x_{i_2}, x_{i_3}$ and add edges $c_{j,h}x_{i_h}$ and $c'_{j,h}x_{i_h}$ for $h = 1, 2, 3$.

- We add an edge between every x-type vertex and every b-type vertex. We also add an edge between every x-type vertex and every d-type vertex.

- We add an edge between every a-type vertex and every b-type vertex. We also add an edge between every a-type vertex and every d-type vertex.

In Figure 2.6 we illustrate an example in which C_j is a clause with ordered variables $x_{i_1}, x_{i_2}, x_{i_3}$. The thick edges indicate the connection between the variables vertices and the c-type vertices of the two copies of the clause gadget. The dashed thick edges indicate the connections between the a-type and c-type vertices of the two copies of the clause gadget. We omitted the indices from the labels of the clause gadget vertices to increase the visibility.

Before we show how to obtain the graph G'_I, we introduce the following terminology. Let H be some graph. An *F-identification* of an edge $st \in E(H)$ is the following operation. We remove the edge st from H but keep the vertices s and t. We take a copy of

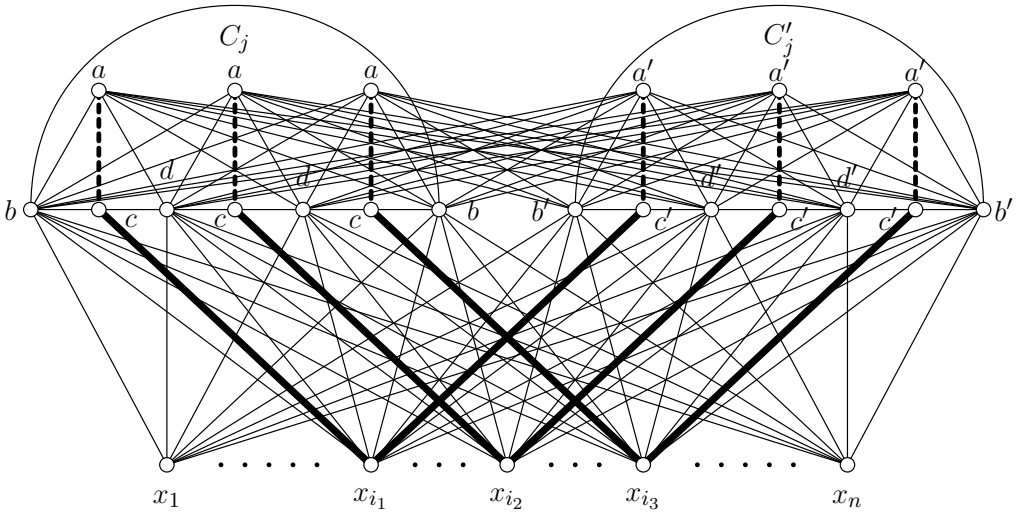

Figure 2.6: The graph G_I^* for the clause $C_j = \{x_{i_1}, x_{i_2}, x_{i_3}\}$.

F and remove u and v from it. We then add an edge between s and $N_F(u)$ and an edge between t and $N_F(v)$. Note that by symmetry we could reverse the role of u and v in this operation. Lemma 2.3.1 ensures that s and t have different colours in any 4-colouring of the new graph G'_I.

In order to obtain G'_I from G_I^* we first apply consecutive F-identifications on all edges between a-type and c-type vertices, on all edges between c-type and x-type vertices and on all edges between two b-type vertices. We take a complete graph on four new vertices $r_1, \ldots r_4$ called r-type vertices, and apply consecutive F-identifications on each edge between them. This leads to a graph K. We connect K to the modified graph G_I^* by adding an edge between every $a_{i,j}$ and every vertex in $\{r_2, r_3, r_4\}$ and an edge between every $a'_{i,j}$ and every vertex in $\{r_1, r_3, r_4\}$. This completes the construction of G'_I.

We need the following lemma.

Lemma 2.3.2. *The graph G'_I is C_3-free.*

Proof. In order to prove the lemma we must check if G'_I has an edge the end vertices of which share a common neighbour. Recall that F is C_3-free and that u and v do not form an edge. Hence, we only have to consider edges in G'_I that are also in G_I^*. Such edges connect the following vertices: a-type with b-type, a-type with d-type, b-type with

c-type, c-type with d-type, b-type with x-type, and d-type with x-type. By construction, the end vertices of all these edges have no common neighbour. We conclude that G'_I is C_3-free. □

In order to prove that G'_I is P_{164}-free, we first need the following lemma.

Lemma 2.3.3. *Let P be an induced path in F. Then the following statements hold:*

(i) If P starts in u and ends in v, then $|V(P)| \leq 8$.

(ii) If P starts in u and contains v as an internal vertex, then $|V(P)| \leq 9$.

(iii) If P starts in u and does not contain v, then $|V(P)| \leq 45$.

(iv) If P does not contain u or v as end-vertices, then $|V(P)| \leq 90$.

Proof. Let P be an induced path in F. We assume that P has maximal length under each of the four statements (i)-(iv) we must show.

We first prove (i). Suppose that P starts in u and ends in v. Let w and x denote the neighbours of u and v on P, respectively, Then $w \in A \cup D$ and $x \in B \cup C$. If $w \in A$ and $x \in B$ then $|V(P)| = 4$. If $w \in D$ and $x \in C$ then $|V(P)| = 4$ as well. By symmetry, we are left to consider the case in which $w \in A$ and $x \in C$. Then the neighbour of w must be on a Q-cycle or T-cycle. Because this cycle has five vertices, P can contain at most four vertices of it. When P leaves the cycle, its next vertex must be on C; this vertex is adjacent to v, so it must be x. Hence, we find that $|V(P)| \leq 8$, as desired.

We now prove (ii). Suppose that P starts in u and contains v as an internal vertex. Let w be the neighbour of u on P, and let x, x' be the neighbours of v on P where x denotes the neighbour of v that is closest to w on P. Then $w \in A \cup D$ and $\{x, x'\} \subset B \cup C$. Suppose that w is neither adjacent to x nor to x'. If $w \in A$, then $x, x' \in C$. If $w \in D$, then $x, x' \in B$. In both cases, the subpath of P that goes from w to x cannot contain a vertex from $B \cup C$, because such a vertex is adjacent to v. This subpath cannot contain a vertex from $A \cup D$ either, because then u has more than one neighbour of P. Hence, these two cases are not possible. We find that w must be adjacent to x. If x' has another neighbour on P, then this neighbour must be on an R-cycle or S-cycle, and P must end on this cycle. Because P can contain at most four vertices from an R-cycle or S-cycle, we find that $|V(P)| \leq 9$, as desired.

We now prove (iii). Suppose that P starts in u and does not contain v. Let w be the neighbour of u on P. Then $w \in A \cup D$. We suppose that $w \in D$. The case in

which $w \in A$ uses the same arguments but leads to a shorter path since if $w \in A$ then P alternates between vertices from the set C with $|C| < |B|$ instead of B and path segments of R-cycles (as we will see soon). We find that P has maximum length if the neighbour of w on P belongs to C and if after passing through this neighbour P alternates as much as possible between vertices from B and path segments of R-cycles. Because every R-cycle has five vertices, P can use at most 4 of them. If P uses four vertices of such a cycle, at least two vertices of B are excluded, i.e., they cannot be used due to the matching edges between the cycles and B. We choose those cycles that exclude the same two vertices of B. In this way we can let the number of vertices from B that are on P be 8 and the number of cycles that P passes through be 9, where P uses three vertices from the first and last cycle, and four vertices from the other 8 cycles. As P also contains u, w and a vertex from C, we find that $|V(P)| \leq 45$, as desired.

Finally, we prove (iv) by applying the same arguments as (iii), i.e., we can let P alternate as much as possible between vertices from B and path segments of R-cycles and as much as possible between vertices from D and path segments of Q-cycles. In this way we find that $|V(P)| \leq 45 \cdot 2 = 90$; note that this bound is not tight, but this will be irrelevant for our purposes. □

We are now ready to prove that G'_I is P_{164}-free.

Lemma 2.3.4. *The graph G'_I is P_{164}-free (but not P_{163}-free).*

Proof. We call a path Q in G'_I an *F-path* if the following two conditions are both met. Firstly, there exists an edge e in G^*_I on which we applied an F-identification such that Q starts in one end-vertex of e and ends in the other end-vertex of e. Secondly, none of the internal vertices of Q belongs to G^*_I, i.e., they are all contained in the corresponding copy of F used in the F-identification. We say that Q is of type ac, bb, or cx if the end-vertices of e are of type a, c, or b, b, or c, x, respectively.

Let P be an induced path in G'_I. First suppose that P contains no vertex of K. We need four claims; the first three claims are required to prove the last claim.

Claim 1. P contains at most two different F-paths of ac-type.

We prove Claim 1 as follows. Suppose that P contains at least three different F-paths of ac-type. Then P contains no vertices of b-type or d-type, because such vertices would have degree 3 in P. Hence the three F-paths in P must be made connected by vertices of

x-type. Because every c-type vertex is adjacent to exactly one x-type vertex and the set of x-type vertices is independent, this is not possible. Hence, we have proven Claim 1.

Claim 2. P contains at most two different F-paths of cx-type.

We prove Claim 2 as follows. Suppose that P contains at least three different F-paths of cx-type. Because every x-type vertex is adjacent to exactly two c-type vertices, this means that P contains at least two x-type vertices. Consequently, P has at least one vertex that is of b-type or d-type in order to ensure its connectivity. Because a vertex of type b or d is adjacent to all x-type vertices, we then find that P contains exactly two x-type vertices. One of these two x-type vertices must be adjacent to two c-type vertices. This is not possible, as this x-type vertex is also adjacent to the b-type or d-type vertex of P. Hence, we have proven Claim 2.

Claim 3. If P contains an F-path of bb-type, then P does not contain any other F-path.

We prove Claim 3 as follows. Let P contain an F-path of bb-type; note that P contains at least two b-type vertices. In order to obtain a contradiction, suppose that P contains another F-path. First suppose that this second F-path is of bb-type. Then P contains at least four b-type vertices. Then, in order to be connected, P must contain at least one a-type or x-type vertex. Such a vertex would be adjacent to all b-type vertices, and consequently have degree at least four in P. This is not possible as P is an induced path. Hence, P contains only one F-path of bb-type. Now suppose that the second F-path is of ac-type or of cx-type. In both cases P contains a vertex, namely of a-type or x-type, that is adjacent to two b-type vertices and to one other vertex, namely its neighbour on the second F-path. Then P is not an induced path. Hence, this is not possible, and we have proven Claim 3.

Claim 4. The following statements hold:

(i) If P starts in an a-type vertex and ends in an a-type vertex, then $|V(P)| \leq 29$.

(ii) If P starts in an a-type vertex and does not end in an a-type vertex, then $|V(P)| \leq 73$.

(iii) If P starts in an a-type vertex and has no other a-type vertex, then $|V(P)| \leq 66$.

(iv) If P neither starts nor ends in an a-type vertex, then $|V(P)| \leq 117$.

We prove Claim 4 as follows. By Claims 1-3, we find that P contains at most four different F-paths. This leads to five different cases.

Case 1. P contains exactly four different F-paths.
By Claims 1–3, two of these F-paths are of type ac and two are of type cx. Then P contains neither b-type nor d-type vertices. In particular, moving along P and denoting the types of the vertices of P that belong to G_I^* leads to the following sequences of types: a, c, x, c, a. We say that the *ordered type* of P is $T(P) = acxca$. Also, in the remaining cases we make use of this notation.

We use Lemma 2.3.3 to deduce the following. If P starts in an a-type vertex and ends in an a-type vertex then $|V(P)| \leq 5 + 4 \cdot 6 = 29$. If P starts in an a-type vertex and does not end in an a-type vertex then $|V(P)| \leq 5 + 4 \cdot 6 + 44 = 73$. If P neither starts nor ends in an a-type vertex then $|V(P)| \leq 44 + 5 + 4 \cdot 6 + 44 = 117$. We note that statement (iii) of Claim 4 is satisfied in a trivial way, because P contains two a-type vertices.

Case 2. P contains exactly three different F-paths.
By Claims 1–3, either two of these F-paths are of type ac and one is of type cx, or two are of type cx and one is of type ac. In the first case, P contains at least two a-type vertices. Then P cannot contain a b-type or d-type vertex, because such a vertex would be adjacent to the two a-type vertices and the x-type vertex. This means that an x-type vertex must connect the two F-paths of type ac. This is not possible, because in that case P would contain four different F-paths. Hence, P has two F-paths of type cx and one of type ac. Then the two F-paths of type cx either share an x-type vertex, or P contains at least two x-type vertices. Then P cannot contain a b-type or d-type vertex; in the first case the x-type vertex will have three neighbours on P and in the second case the b-type or d-type vertex will have three neighbours on P. Hence, we find that $T(P) = acxc$ or $T(P) = cxca$. We observe that statement (i) of Claim 4 is not applicable in this case; P only contains one a-type vertex. We find that $|V(P)| \leq 4 + 3 \cdot 6 + 44 = 66$ if P starts in an a-type vertex. This shows statements (ii) and (iii). If P neither starts nor ends in an a-type vertex, then $|V(P)| \leq 44 + 4 + 3 \cdot 6 + 44 = 110 \leq 117$. Hence, statement (iv) in Claim 4 is valid as well.

Case 3. P contains exactly two different F-paths.

By Claim 3, these two F-paths are not of type bb. We find that

$$T(P) \in \{acx, cabac, cabxc, cadac, cadxc, cxbac, cxbxc, cxc, cxdac, cxdxc, xca\}.$$

We use the arguments of Cases 1 and 2 to find that the bounds in Claim 4 are valid.

Case 4. P contains exactly one F-path.
We find that $T(P)$ is contained in one of

$$\begin{aligned} & \{abac, abxc, adac, adxc, caba, cabc, cabx, cada, cadc, cadx\} \\ \cup\ & \{cbac, cbxc, cdac, cdxc, cxba, cxbc, cxbx, cxda, cxdc, cxdx\} \\ \cup\ & \{dcbbcd, xbac, xbxc, xdac, xdxc\}. \end{aligned}$$

We use the arguments of Cases 1 and 2 to find that the bounds in Claim 4 are valid.

Case 5. P contains no F-paths.
If P contains no vertices of G_I^*, then $|V(P)| \leq 90$ by Lemma 2.3.3. Suppose P contains at least one vertex from G_I^*. We find that $T(P)$ is contained in one of

$$\begin{aligned} & \{aba, abc, abx, ada, adc, adx\} \\ \cup\ & \{babc, badc, bcdcdc, bxbc, bxdc\} \\ \cup\ & \{cbabc, cbadc, cbxbc, cbxdc, cdabc, cdadc, cdcb, cdcdcb, cdxbc, cdxdc\} \\ \cup\ & \{dabc, dadc, dcdcb, dxbc, dxdc\} \\ \cup\ & \{xba, xbc, xbx, xda, xdc, xdx\}. \end{aligned}$$

Note that some of the strings in the above set are substrings of others; we added these for ease of verification. We use the arguments of Cases 1 and 2 to find that the bounds in Claim 4 are also valid in this case. This completes the proof of Claim 4.

By Claim 4, we conclude that $|V(P)| \leq 117$ as $29 \leq 66 \leq 73 \leq 117$, if P contains no vertex of K. By using Claim 4, we now prove the desired upper bound on $|V(P)|$ in case P does contain one or more vertices from K. We distinguish a number of cases depending on the number of r-type vertices. Throughout the case analysis we assume that P is an induced path in G_I' of maximal length.

Case 1. P contains no r-type vertices.
Then either P contains only vertices of $V(G_I')\backslash V(K)$ or only vertices of K. We already

showed that $|V(P)| \leq 164$ in the first case. By Lemma 2.3.3, we find that $|V(P)| \leq 90 \leq 164$ in the second case.

Case 2. P contains exactly one r-type vertex.
Because P is maximal, the r-type vertex of P has two neighbours on P. If both these neighbours are a-type vertices then there are no other vertices of K on P, and we find that $|V(P)| \leq 73 + 1 + 73 = 147$ due to Claim 4. If both these neighbours are from K, then we find that $|V(P)| \leq 44 + 1 + 44 = 99$, due to Lemma 2.3.3. In the remaining case, one neighbour is of a-type and one neighbour is from K, and we find that $|V(P)| \leq 73 + 1 + 44 = 118$.

Case 3. P contains exactly two r-type vertices.
Because P is maximal, each r-type vertex has two neighbours. By construction, r-type vertices are not adjacent to each other. Hence, we obtain three subpaths after removing the two r-vertices, say $P = P_1 r P_2 r P_3$, where r denotes an r-type vertex. Because a-type vertices are adjacent to three r-type vertices, we find that each a-type vertex is adjacent to at least one r-type vertex on P. This means that P_1 and P_3 have at most one a-type vertex, and this vertex must be one of their end-vertices if it exists. It also means that P_2 has at most two a-type vertices, which can only be end-vertices of P_2. If P_1 belongs to K, then $|V(P_1)| \leq 44$ by Lemma 2.3.3; otherwise $|V(P_1)| \leq 73$ by Claim 4. The same goes for P_3. If P_2 belongs to K then $|V(P_2)| \leq 6$ by Lemma 2.3.3; otherwise $|V(P_2)| \leq 29$ by Claim 4. We conclude that $|V(P)| \leq 66 + 1 + 29 + 1 + 66 = 163$ by Claim 4. By following the proof of Lemma 2.3.3 and by letting $T(P_1) = cxca$, $T(P_2) = acxca$, and $T(P_3) = acxc$, we find an induced path on 163 vertices in G'_I. Hence, G'_I is not P_{163}-free.

Case 4. P contains exactly three r-type vertices.
Because each a-type vertex is adjacent to three r-type vertices, we find that each a-type vertex is adjacent to at least two r-type vertices. Hence, P can contain at most two a-type vertices. If P contains no a-type vertex, then $|V(P)| \leq 3 + 2 \cdot 6 + 2 \cdot 44 = 103$. If P contains exactly one a-type vertex, then this a-type vertex has two neighbours in P and these neighbours are of r-type. Then $|V(P)| \leq 44 + 4 + 6 + 44 = 98$. If P contains exactly two a-type vertices, then both these vertices have two neighbours on P and these two neighbours are of r-type. Then $|V(P)| \leq 44 + 5 + 44 = 93$.

Case 5. P contains all four r-type vertices.
Then P cannot contain an a-type vertices. Hence, P is a path in K. As such, $|V(P)| \leq$

$4 + 3 \cdot 6 + 2 \cdot 44 = 110$.

We note that in all five cases, $|V(P)| \leq 163$ holds. Hence, G'_I is P_{164}-free, as desired. Moreover, in Case 3, we identified an induced path in G'_I on 163 vertices. Consequently, G'_I is not P_{163}-free. This completes the proof of Lemma 2.3.4. □

We also need the following lemma.

Lemma 2.3.5. *The graph G^*_I has a 4-colouring in which every $a_{j,h}$ has colour 1 and every $a'_{j,h}$ has colour 2 if and only if I has a truth assignment in which each clause contains at least one true and at least one false literal.*

Proof. Suppose G^*_I has a 4-colouring in which every $a_{j,h}$ has colour 1 and every $a'_{j,h}$ has colour 2. Since $a_{1,1}$ with colour 1 and $a'_{1,1}$ with colour 2 are adjacent to every b-type vertex, we may assume by symmetry that every $b_{j,1}$ and every $b'_{j,1}$ has colour 3, whereas every $b_{j,2}$ and every $b'_{j,2}$ has colour 4. Since every a-type vertex is also adjacent to every d-vertex, every d-type vertex must have a colour from $\{3, 4\}$. This implies the following. Firstly, it implies that all x-type vertices have a colour from $\{1, 2\}$. Secondly, it implies that every $c_{j,1}$ has a colour from $\{2, 4\}$, every $c_{j,2}$ has a colour from $\{2, 3, 4\}$ and every $c_{j,3}$ has a colour from $\{2, 3\}$. Thirdly, it implies that every $c'_{j,1}$ has a colour from $\{1, 4\}$, every $c_{j,2}$ has a colour from $\{1, 3, 4\}$ and every $c_{j,3}$ has a colour from $\{1, 3\}$.

Now suppose there is a clause C_j with each of its three literals coloured by colour 2. Then $c_{j,1}$ must have colour 4 and $c_{j,3}$ must have colour 3. Consequently, $d_{j,1}$ has colour 3 and $d_{j,2}$ has colour 4. Then $c_{j,2}$ cannot have a colour in a proper 4-colouring. Hence this is not possible and we find that at least one literal in every clause is coloured by colour 1. By considering the copies, in a similar way we find that at least one literal in every clause is coloured by colour 2. Hence, we can define a truth assignment that sets a literal to `FALSE` if the corresponding x-type vertex has colour 2, and to `TRUE` otherwise. So the graph G^*_I has a 4-colouring in which every $a_{j,h}$ has colour 1 and every $a'_{j,h}$ has colour 2 implies a truth assignment for I in which each clause contains at least one true and at least one false literal.

For the converse, suppose I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal. We use colour 1 to colour the x-type vertices representing the true literals and colour 2 for the false literals. Since each clause contains at least one true literal, we can colour $c_{j,1}$, $c_{j,2}$ and $c_{j,3}$, respecting the precolouring. Similarly, since each clause contains at least one false literal, we can colour

$c'_{j,1}$, $c'_{j,2}$ and $c'_{j,3}$, respecting the precolouring. We colour all other remaining uncoloured vertices in a straightforward way. This completes the proof of Lemma 2.3.5. □

The following lemma is the last lemma we need in order to state our main result.

Lemma 2.3.6. *The graph G'_I is 4-colourable if and only if I has a satisfying truth assignment in which each clause contains at least one true literal and at least one false literal.*

Proof. Suppose that G'_I is 4-colourable. By Lemma 2.3.1, the vertices of the edges on which we applied F-identifications do not have the same colour. This means that G^*_I is 4-colourable. It also means that the vertices $r_1, \ldots, r_4$ are not coloured alike. We may assume without loss of generality that r_i gets colour i for $i = 1, \ldots, 4$. Then, by construction, every $a_{j,h}$ has colour 1 and every $a'_{j,h}$ has colour 2 in G'_I, and consequently in G^*_I. By Lemma 2.3.5, I has a truth assignment in which each clause contains at least one true and at least one false literal.

Suppose that I has a truth assignment in which each clause contains at least one true and at least one false literal. By Lemma 2.3.5, G^*_I has a 4-colouring in which every $a_{j,h}$ has colour 1 and every $a'_{j,h}$ has colour 2. By Lemma 2.3.1 we can extend the 4-colouring of G^*_I to a 4-colouring of G'_I. □

The main result of this section follows directly from Lemmas 2.3.2, 2.3.4 and 2.3.6 after recalling that 4-Colouring is in NP and that Not-All-Equal 3-Satisfiability with un-negated literals only is NP-complete and observing that the construction of G'_I can be carried out in polynomial time.

Theorem 2.3.7. *The 4-Colouring problem is NP-complete even for (C_3, P_{164})-free graphs.*

2.4 Colouring Graphs without Short Cycles and Long Induced Paths

We consider the relation between the girth of a graph and the length of a forbidden induced path for the k-Colouring problem. As a start, recall that graphs with girth $g = \infty$ are forests, and consequently, these graphs are 2-colourable. What if g is finite? In this section we determine, for any fixed girth $g \geq 4$, a value $r(g)$ such that every

$P_{r(g)}$-free graph with girth at least g is 3-colourable. Here we tried to make $r(g)$ as large as possible. However, in general it is not trivial to construct, for given k and g, a k-chromatic P_r-free graph of girth g for r as small as possible, or, for given k and r, a k-chromatic P_r-free graph of girth g for g as large as possible. For example, the Grötzsch graph [48] is 4-chromatic, P_6-free and of girth 4. Hence, the bound of Sumner [91] is tight. Brinkmann and Meringer [15] constructed a 4-chromatic P_{10}-free graph with girth 5. Hence, the bound in Table 2.1 for P_{10}-free graphs is tight with respect to the girth. We are not aware of examples of 4-chromatic graphs of girth at least 6 without long induced paths and expect that some of our bounds in Table 2.1 can be improved. Sumner [91] showed that every P_5-free graph of girth at least 4 is 3-colourable. Randerath and Schiermeyer [85] extended this result by showing that for all $r \geq 4$, every P_r-free graph of girth at least 4 is $(r-2)$-colourable. We extend the result of Sumner [91] in another direction. Our results lead to Table 2.1. Note that for the cases $g \in \{4, 5, 7\}$ the values are slightly worse than the value for the general case $g \geq 8$; the difference between them is 1. The proofs of the results in Table 2.1 are constructive, i.e., yield polynomial-time 3-colouring algorithms.

girth	forbidden induced path
$g = 4$	P_5-free [91]
$g = 5$	P_7-free
$g = 6$	P_{10}-free
$g = 7$	P_{12}-free
$g \geq 8$	P_r-free for $r = 2g + \lceil \frac{g-2}{4} \rceil - 3$

Table 2.1: 3-colourable P_r-free graphs of given girth.

We start with some additional terminology. For a vertex v and subset $U \subseteq V$ we define $\text{dist}(v, U) = \min\{\text{dist}(v, u) \mid u \in U\}$; note that $\text{dist}(v, U) = 0$ if and only if $v \in U$. For a subset $U \subseteq V$ and integer s, we define $N^{=s}(U) = \{v \in V \mid \text{dist}(v, U) = s\}$ and $N^{\leq s}(U) = \{v \in V \mid \text{dist}(v, U) \leq s\}$. We make two assumptions that are valid throughout this section. First, we may assume that the graphs we consider are connected. Second, we may assume that they have minimum degree at least 3; this follows from Proposition 1.2.2 by setting $L(u) = \{1, 2, 3\}$ for every $u \in V(G)$ for the list assignment $\mathcal{L}$ in the statement of this proposition.

Proposition. *Let G be a graph and u be a vertex of degree at most 2. Then G is*

3-*colourable if and only if* $G - u$ *is* 3-*colourable.*

We show the four new bounds in Table 2.1 in Theorems 2.4.1–2.4.4, respectively.

Theorem 2.4.1. *Every* P_7*-free graph of girth* 5 *is* 3*-colourable, and such a* 3*-colouring can be found in* $O(n^2)$ *time.*

Proof. Let $G = (V, E)$ be a connected P_7-free graph with minimum degree at least 3 such that $g(G) = 5$. Consider $uv \in E$ and let $U = \{u, v\}$. See Figure 2.7 for an example. We first prove four useful properties of the sets $N^{=s}(U)$:

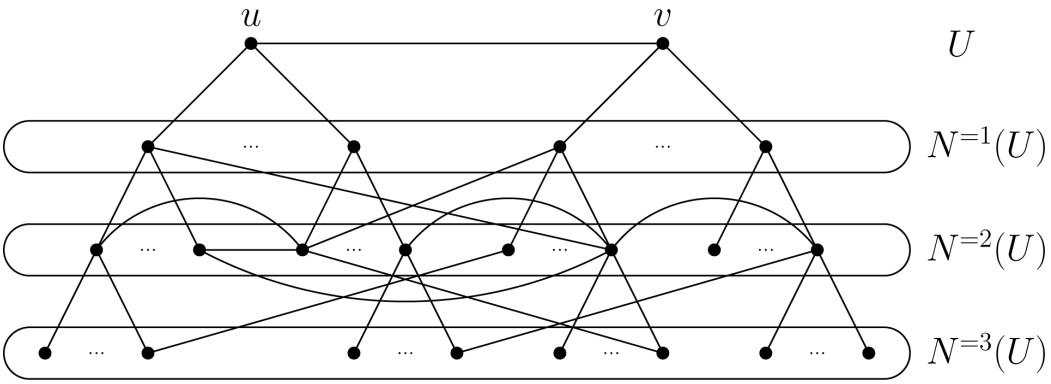

Figure 2.7: A P_7-free graph of girth 5.

1. $N^{=1}(U)$ is an independent set;
2. $N^{=2}(U)$ induces a bipartite graph;
3. $N^{=3}(U)$ is an independent set;
4. $N^{=s}(U) = \emptyset$ for $s \geq 4$.

We first prove property 1. Suppose that w_1w_2 is an edge in $N^{=1}(U)$. Then none of w_1 and w_2 is adjacent to both of u and v, because of the C_3-freeness of G. Hence we may assume without loss of generality that w_1 is adjacent to u and w_2 is adjacent to v. Then the cycle uvw_2w_1u is a C_4, which is not possible. We prove property 2 as follows. In order to obtain a contradiction, suppose that $G[N^{=2}(U)]$ contains an odd cycle $C_l = x_1x_2 \cdots x_lx_1$. Because $g(G) = 5$ and G is P_7-free, $l = 5$ or $l = 7$. Let w be a vertex in $N^{=1}(U)$ adjacent to x_1 and assume without loss of generality that w is adjacent

to u. Then $wv \notin E$. If $l = 5$, then w is not adjacent to $x_2, \ldots, x_5$, because $g(G) = 5$. By the same reason, $x_1x_2x_3x_4$ is an induced path in G. Then $vuwx_1 \ldots x_4$ is an induced P_7. Hence $l = 7$. Because $g(G) = 5$, we find that w is adjacent to none of the vertices x_2, x_3, x_6, x_7. By the same reason, $x_1x_2x_3x_4$ and $x_1x_7x_6x_5$ are induced paths in G. If w is not adjacent to x_4, then $vuwx_1 \ldots x_4$ is an induced P_7. Hence, w must be adjacent to x_4. If w is not adjacent to x_5, then $vuwx_1x_7x_6x_5$ is an induced P_7. Hence, w must be adjacent to x_5. However, now we have a triangle wx_4x_5w, which is not possible because $g(G) = 5$. We conclude that property 2 must hold.

We prove property 4 before property 3. Suppose that property 4 is not true. First suppose that $N^{=5}(U) \neq \emptyset$. Then, for a vertex $x \in N^{=5}(U)$, there is a path $w_1 \ldots w_4x$ where $w_i \in N^{=i}(U)$. We assume without loss of generality that w_1 is adjacent to u. However, then $uvw_1 \ldots w_4x$ or $vuw_1 \ldots w_4x$ is an induced P_7. Hence $N^{=5}(U) = \emptyset$, and consequently, $N^{=s}(U) = \emptyset$ for $s \geq 5$. This means that for property 4 to be non-valid, we must have $N^{=4}(U) \neq \emptyset$.

First suppose that $G[N^{=4}(U)]$ contains an edge xy. There is a path $w_1w_2w_3$ such that $w_1 \in N^{=1}(U)$, $w_2 \in N^{=2}(U)$, $w_3 \in N^{=3}(U)$ and $w_3x \in E$. Assuming that $uw_1 \in E$, we get the induced path $vuw_1w_2w_3xy$ isomorphic to P_7. Hence, such an edge xy does not exist implying that $N^{=4}(U)$ is an independent set. Suppose that a vertex $x \in N^{=4}(U)$ is adjacent to at least two vertices z_1 and z_2 in $N^{=3}(U)$ and let $uw_1w_2z_1x$ be a (u, x)-path in G. Because $g(G) = 5$, we find that $z_1z_2 \notin E$ and $w_2z_2 \notin E$. Then $vuw_1w_2z_1xz_2$ is an induced P_7. Hence, x is adjacent to exactly one vertex in $N^{=3}(U)$. Recall that $N^{=5}(U) = \emptyset$. Then we find that $d(x) = 1$. However, G has no such vertices, because G has minimum degree at least 3. We conclude that property 4 must hold.

For the proof of property 3, we first suppose that $x_1x_2x_3$ is an induced path in $G[N^{=3}(U)]$. Then there are adjacent vertices $w_1 \in N^{=1}(U)$ and $w_2 \in N^{=2}(U)$ such that $w_2x_1 \in E$. Because w_1 is adjacent to a vertex in U, we assume without loss of generality that w_1 is adjacent to u and get the path $vuw_1w_2x_1x_2x_3$. Because $g(G) \geq 5$, this path is an induced P_7. Hence $G[N^{=3}(U)]$ is P_3-free.

We now suppose that xy is an edge in $G[N^{=3}(U)]$. Because G has minimum degree at least 3 and $G[N^{=3}(U)]$ is P_3-free, y is adjacent to at least two vertices z_1 and z_2 in $N^{=2}(U)$. By definition, there are adjacent vertices $w_1 \in N^{=1}(U)$ and $w_2 \in N^{=2}(U)$ such that $w_2x \in E$, and we can assume that $uw_1 \in E$. Because $g(G) = 5$, we find that z_1, z_2, w_2 are three different vertices that are pairwise non-adjacent, and that z_1w_1 and

z_2w_1 cannot be both edges in G. We assume without loss of generality that $z_1w_1 \notin E$. Then $vuw_1w_2xyz_1$ is an induced P_7, which is not possible. We conclude that property 3 must hold, and we have proven all four properties.

Using these four properties we construct a 3-colouring of G as follows. We colour the vertices u and v by the colours 1 and 2 respectively, and all the vertices of the independent set $N^{=1}(U)$ by 3. The vertices of the bipartite graph $G[N^{=2}(U)]$ are coloured by 1 and 2. Finally, the vertices of the independent set $N^{=3}(U)$ are coloured by 3. The computational complexity for colouring u and v is $O(1)$. The computational complexity for the rest of the algorithm is $O(n^2)$. Hence, the algorithm performs in time $O(n^2)$. This completes the proof of Theorem 2.4.1. □

Theorem 2.4.2. *Every P_{10}-free graph of girth 6 is 3-colourable, and such a 3-colouring can be found in $O(n^2)$ time.*

Proof. Let $G = (V, E)$ be a connected P_{10}-free graph with minimum degree at least 3 and girth 6. Let $U = \{x_1, \ldots, x_6\}$ be the vertex set of a C_6 in G (vertices are enumerated in the cycle order). We observe that this 6-vertex cycle is induced, because $g(G) = 6$. Denote by X_i the set of vertices of $N^{=1}(U)$ adjacent to x_i for $i = 1, \ldots, 6$. Using the (C_3, C_4, C_5)-freeness of G, we observe the following:

1. X_i is independent for $1 \leq i \leq 6$;
2. $X_i \cap X_j = \emptyset$ for $1 \leq i < j \leq 6$;
3. if $y_iy_j \in E$ for $y_i \in X_i$, $y_j \in X_j$ and $1 \leq i < j \leq 6$, then $j - i = 3$.

Let $H_1, \ldots, H_m$ be the connected components of $G[V \setminus N^{\leq 1}(U)]$. We need the following claim.

Claim 1. Each graph H_j is either an isolated vertex or a star $K_{1,t}$ for some $t \geq 1$.

We prove Claim 1 as follows. Consider a graph H_j for some $1 \leq j \leq m$. First we show that H_j is P_4-free. In order to obtain a contradiction, suppose that H_j contains an induced path $v_1v_2v_3v_4$. Let $z_1 \ldots z_s$ be a shortest path such that $z_1 \in U$ and $z_s \in \{v_1, \ldots, v_4\}$. Without loss of generality we assume that $z_1 = x_1$, $z_2 \in X_1$ and either $z_s = v_1$ or $z_s = v_2$.

Because $g(G) = 6$, we find that z_{s-1} is adjacent to exactly one vertex of the path $v_1v_2v_3v_4$. If $z_s = v_1$ then $x_5x_4 \ldots x_1z_2 \ldots z_{s-1}v_2 \ldots v_4$ is an induced path with at least

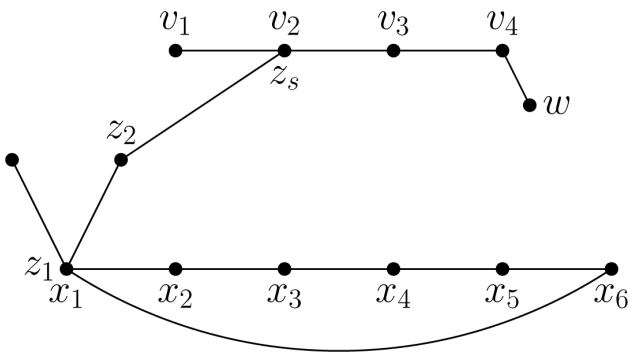

Figure 2.8: A subgraph of a P_{10}-free graph of girth 6.

10 vertices. Hence $z_s = v_2$, and moreover, any shortest path between U and $\{v_1, \ldots, v_4\}$ neither contains v_1 nor v_4. If $s \geq 4$ then $x_5x_4 \ldots x_1z_2 \ldots z_{s-1}v_2v_3v_4$ is and induced path with at least 10 vertices. Hence $s = 3$, i.e., v_2 is adjacent to a vertex of X_1. If v_4 is adjacent to a vertex y in some set X_j then x_jyv_4 is another shortest path between U and $\{v_1, \ldots, v_4\}$. However, we already deduced that such paths do not contain v_4. Hence, $v_4 \notin N^{\leq 2}(U)$. The graph G has no vertices of degree one, and therefore v_4 is adjacent to a vertex $w \in V(H_j)$. Because $g(G) = 6$, we find that w is not adjacent to v_1, v_2, v_3 and z_2. This means that $x_5x_4 \ldots x_1z_2v_2v_3v_4w$ is an induced P_{10}. This is not possible. See Figure 2.8 for an example. Hence, H_j is P_4-free. Observe that every connected P_4-free graph without induced C_3 and C_4 is either an isolated vertex or a star. This completes the proof of Claim 1.

We are now ready to construct a 3-colouring of G. Using properties 1–3, we colour vertices x_1, x_3, x_5 and all vertices of X_2, X_4, X_6 with colour 1, and x_2, x_4, x_6 and all vertices of X_1, X_3, X_5 with colour 2. Now we colour each H_j. If H_j consists of an isolated vertex, then we colour this vertex with colour 3. Suppose that H_j is a star $K_{1,t}$. Let w be its central vertex and $z_1, \ldots, z_t$ be its leaves.

If $w \notin N^{\leq 2}(U)$, then we colour $z_1, \ldots, z_t$ with colour 3 and w with colour 1. Now let w be adjacent to a vertex of X_i for some $1 \leq i \leq 6$. In this case we colour w with colour 3. It remains to prove that each leaf z_s can be coloured by 1 or 2. Suppose that it is not so for some z_s. Then z_s is adjacent to two vertices in the sets $X_1, \ldots, X_6$ that have colour 1 and 2, respectively. By symmetry, we assume that z_s is adjacent to $y_1 \in X_1$. Because $g(G) = 6$, we find that z_s is not adjacent to vertices X_2 and X_6, and therefore, z_s must be adjacent to some vertex $y_4 \in X_4$ in order to have a neighbour with

colour 1. Because $g(G) = 6$, we find that w is not adjacent to any vertices of $X_1 \cup X_4$. By symmetry, we can assume that $i = 2$, i.e., that w is adjacent to a vertex $y_2 \in X_2$. Because G has minimum degree at least 3 and $x_1x_2 \cdots x_6x_1$ is an induced cycle in G, we find that $X_3 \neq \emptyset$. Let $y_3 \in X_3$. However, then $y_2wz_sy_1x_1x_6 \dots x_3y_3$ is an induced P_{10} due to $g(G) = 6$. This means that each z_s is adjacent either only to vertices coloured by 1 or only to vertices coloured by 2 in the sets $X_1, \dots, X_6$. In the first case we can colour z_s with colour 2, and in the second case we can colour z_s with colour 1. It takes $O(n)$ time for colouring vertices in $N^{\leq 1}(U)$. It takes $O(n^2)$ time for colouring all H_j graphs. Hence, the algorithm performs in $O(n^2)$ time overall. This completes the proof of Theorem 2.4.2. □

Theorem 2.4.3. *Every P_{12}-free graph of girth 7 is 3-colourable, and such a 3-colouring can be found in linear time.*

Proof. Let $G = (V, E)$ be a connected graph of girth 7. Let $U = \{x_1, \dots, x_7\}$ be the vertex set of a C_7 in G (vertices are enumerated in the cycle order). We observe that this 7-vertex cycle is induced, because $g(G) = 7$. Denote by X_i the set of vertices of $N^{=1}(U)$ adjacent to x_i for $i = 1, \dots, 7$. Using the (C_3, C_4, C_5, C_6)-freeness of G, we observe the following:

1. $X_i \cap X_j = \emptyset$ for $1 \leq i < j \leq 7$;
2. $X_1 \cup \dots \cup X_7$ is independent.

Let H be the subgraph of G induced by the set $V \setminus (\{x_1, \dots, x_7\} \cup X_1 \cup \dots \cup X_6)$; note that $X_7 \subseteq V(H)$. We claim that H is bipartite. In order to obtain a contradiction, suppose that H contains an odd cycle $C_l = v_1v_2 \cdots v_lv_1$. Because $g(G) = 7$ and G is P_{12}-free, we deduce that $l \in \{7, 9, 11\}$. Let $y_1 \dots y_s$ be a shortest path, such that $y_1 \in U$ and $y_s \in \{v_1, \dots, v_l\}$. We may assume without loss of generality that $y_s = v_1$. By definition, $s \geq 2$.

Suppose that $l < 11$. Then y_{s-1} is not adjacent to any vertex of $\{v_2, \dots, v_l\}$. If $y_1 = x_7$, then G has an induced path $x_2 \dots x_6y_1 \dots y_sv_2 \dots v_{l-1}$ with at least 12 vertices, which is not possible. See Figure 2.9 for an example when $l = 9$. If $y_1 = x_1$, then we find that G has an induced path $x_6x_5 \dots x_2y_1 \dots y_sv_2 \dots v_{l-1}$ with at least 13 vertices, which is not possible either. If $y_1 \in \{x_2, \dots, x_6\}$, then we apply the same arguments as for the case $y_1 = x_1$.

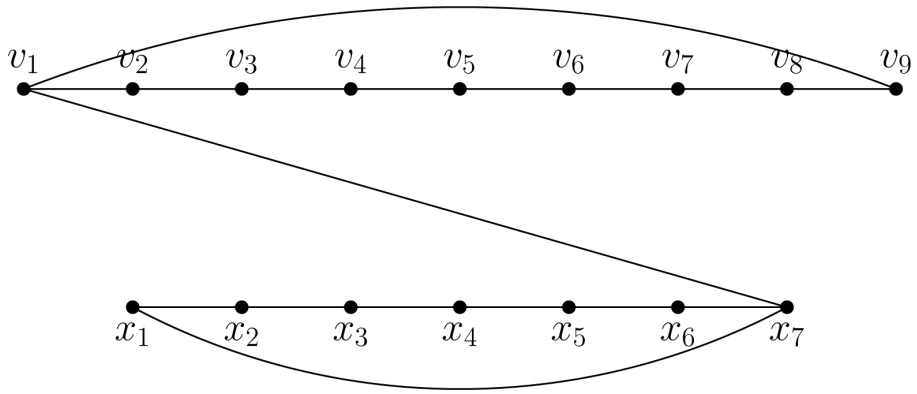

Figure 2.9: A subgraph of a P_{12}-free graph of girth 7.

Suppose that $l = 11$. If y_{s-1} is adjacent to exactly one vertex of $\{v_1, \ldots, v_l\}$, then we use the same arguments as for the case $l < 11$. Let y_{s-1} be adjacent to at least two vertices of $\{v_1, \ldots, v_l\}$. Because $g(G) = 7$, we deduce that y_{s-1} is adjacent to exactly two vertices, namely v_1, v_6 or v_1, v_7. By symmetry, we may assume that y_{s-1} is adjacent to v_1, v_7. If $y_1 = x_7$, then G has an induced path $x_2 \ldots x_6 y_1 \ldots y_s v_2 \ldots v_6$ on at least 12 vertices, which is not possible. If $y_1 = x_1$, then G has an induced path $x_6 x_5 \ldots x_2 y_1 \ldots y_s v_2 \ldots v_6$ on at least 13 vertices, which is not possible either. If $y_1 \in \{x_2, \ldots, x_6\}$, then we apply the same arguments as for the case $y_1 = x_1$. We conclude that H is bipartite.

We now colour G as follows. We colour x_1, x_3, x_5 with colour 1, x_2, x_4, x_6 with colour 2, the vertices of $X_1 \cup \ldots \cup X_6$ and x_7 with colour 3, and finally all the vertices of the (bipartite) graph H with colours 1 and 2. It is not difficult to see that the algorithm performs in linear time. This completes the proof of Theorem 2.4.3. □

Theorem 2.4.4. *Every P_r-free graph with $r = 2g + \lceil \frac{g-2}{4} \rceil - 3$ and $g \geq 8$ is 3-colourable, and such a 3-colouring can be found in linear time.*

Proof. Let $G = (V, E)$ be a connected graph of girth $g \geq 8$. Let $U = \{x_1, \ldots, x_g\}$ be the vertex set of a C_g in G (vertices are enumerated in the cycle order). We observe that this g-vertex cycle is induced. Let $s = \lceil \frac{g-2}{4} \rceil - 1$. We will prove the following two properties of the sets $N^{=t}(U)$:

1. $N^{=t}(U)$ is independent for $1 \leq t \leq s$;
2. each $x \in N^{=t}(U)$ is adjacent to exactly one vertex in $N^{=t-1}(U)$ for $1 \leq t \leq s$.

We first prove property 1. In order to obtain a contradiction, suppose that $N^{=t}(U)$ contains two adjacent vertices y and z for some $1 \leq t \leq s$. By the definition of $N^{=t}(U)$, we find that U contains two vertices x_i, x_j with $\text{dist}(y, x_i) = t$ and $\text{dist}(z, x_j) = t$. Because the distance between x_i and x_j in the cycle $G[U]$ is at most $\lfloor \frac{g}{2} \rfloor$, we find that $G[N^{\leq t}(U)]$ contains a cycle of length at most $2t + 1 + \lfloor \frac{g}{2} \rfloor \leq 2\lceil \frac{g-2}{4} \rceil - 1 + \lfloor \frac{g}{2} \rfloor < g$. This is not possible. Hence, property 1 is valid.

The proof of property 2 is similar. Suppose that $x \in N^{=t}(U)$ is adjacent to at least two different vertices in $N^{=t-1}(U)$. This implies that there are two paths between x and U of length at most t. Therefore G has a cycle of length at most $g - 1$, which is not possible. Hence, property 2 is valid as well.

We now distinguish two cases; first we consider the case $g = 9$ and then the case $g \neq 9$.

Case 1. $g = 9$.
Then $r = 17$, so G is P_{17}-free, and $s = 1$. Let $X \subseteq N^{=1}(U)$ be the set of vertices adjacent to x_9 and let $Y \subseteq N^{=2}(U)$ be the set of vertices adjacent to the vertices of X. Because G has no cycles of length less than 9, we can deduce two extra properties in addition to properties 1 and 2:

3. Y is independent;

4. vertices of Y are not adjacent to the vertices of $N^{=1}(U) \setminus X$.

Let H be the subgraph of G induced by the set $V \setminus (N^{\leq 1}(U) \cup Y)$. We claim that H is bipartite. In order to obtain a contradiction, suppose that H contains an odd cycle $C_l = v_1 v_2 \cdots v_l v_1$. Because $g = 9$ and G is P_{17}-free, we find that $9 \leq l \leq 17$. Let $y_1 \ldots y_t$ be a shortest path such that $y_1 \in U$ and $y_t \in \{v_1, \ldots, v_l\}$. Suppose that $y_t = v_1$. We observe that $t \geq 3$. Because $g = 9$ and $l \leq 17$, we find that y_{t-1} is adjacent to at most one vertex of $\{v_2, \ldots, v_l\}$.

Suppose that y_{t-1} is adjacent to zero vertices of $\{v_2, \ldots, v_l\}$. If $y_1 = x_1$, then G has an induced path $x_8 x_7 \ldots x_2 y_1 \ldots y_t v_2 \ldots v_{l-1}$ on at least 17 vertices. This is not possible, because G is P_{17}-free. See Figure 2.10 for an example. If $y_1 \in \{x_2, \ldots, x_9\}$, then we use the same arguments.

Suppose that y_{t-1} is adjacent to exactly one vertex of $\{v_2, \ldots, v_l\}$. Because $g = 9$, we find that $l = 15$ or $l = 17$. This implies that the 7-vertex sets $\{v_2, \ldots, v_8\}$ and $\{v_{l-6}, v_{l-5}, \ldots, v_l\}$ are disjoint. Because $g = 9$, we find that y_{t-1} is not adjacent to any

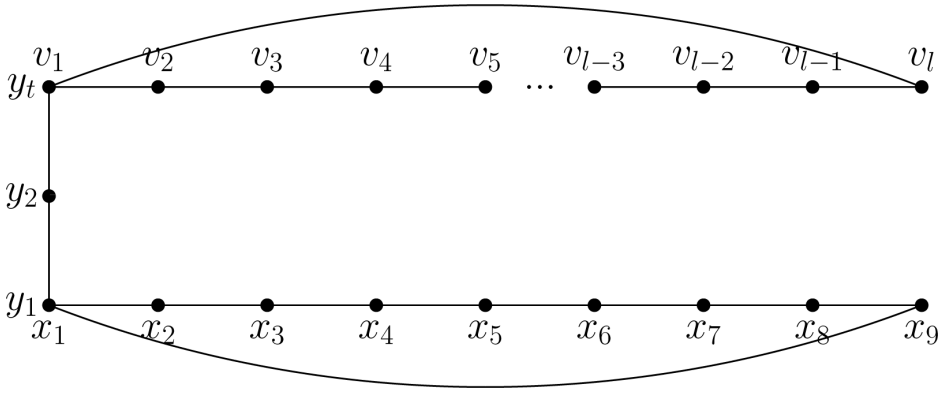

Figure 2.10: A subgraph of a P_{17}-free graph of girth 9.

vertex of $\{v_2, \ldots, v_7, v_{l-5}, v_{l-4}, \ldots, v_l\}$. Because $g = 9$ and $l \leq 17$, we find in addition that y_{t-1} cannot be adjacent to both v_8 and v_{l-6}. By symmetry, we may assume that y_{t-1} is not adjacent to v_8. If $y_1 = x_1$, then $x_8x_7 \ldots x_2y_1 \ldots y_tv_2 \ldots v_8$ is an induced path in G with at least 17 vertices. This is not possible, because G is P_{17}-free. If $y_1 \in \{x_2, \ldots, x_9\}$, then we use the same arguments. We conclude that H is bipartite.

Using properties 1-4 and the fact that H is bipartite, we can colour G as follows. We colour vertices x_1, x_3, x_5, x_7 with colour 1, vertices x_2, x_4, x_6, x_8 with colour 2, vertex x_9 and the vertices of the (independent) set $N^{=1}(U) \setminus X$ with colour 3, all vertices in X and Y with colour 1 and 3, respectively, and finally, all vertices of the (bipartite) graph H with colours 1 and 2.

Case 2. $g \neq 9$.

Let H be the subgraph of G induced by the set $V \setminus N^{\leq s}(U)$. We claim that H is bipartite. In order to obtain a contradiction, suppose that H contains an odd cycle $C_l = v_1v_2 \cdots v_lv_1$. Because G is P_r-free for $r = 2g + \lceil \frac{g-2}{4} \rceil - 3$, we find that $g \leq l \leq 2g + \lceil \frac{g-2}{4} \rceil - 2$.

Let $y_1 \ldots y_t$ be a shortest path such that $y_1 \in U$ and $y_t \in \{v_1, \ldots, v_l\}$. We may assume without loss of generality that $y_1 = x_1$ and $y_t = v_1$. Recall that $s = \lceil \frac{g-2}{4} \rceil - 1$ and observe that $t \geq s + 2$. If y_{s-1} is adjacent to at least two vertices of $\{v_2, \ldots, v_l\}$, then $l \geq 3g - 6$. However, we also have $3g - 6 > 2g + \lceil \frac{g-2}{4} \rceil - 2 \geq l$ as $g \geq 8$. Hence, this is not possible, and consequently, y_{t-1} is adjacent to at most two vertices of $\{v_1, \ldots, v_l\}$.

First suppose that y_{s-1} is adjacent to zero vertices of $\{v_2, \ldots, v_l\}$. Then G has an induced path $x_{g-1}x_{g-2} \ldots x_2y_1 \ldots y_tv_2 \ldots v_{l-1}$ on at least $g + l + t - 4$ vertices. However,

$g+l+t-4 \geq 2g+s-2 = 2g+\lceil \frac{g-2}{4} \rceil - 3 = r$, which is not possible as G is P_r-free.

Now suppose that y_{s-1} is adjacent to exactly one vertex of $\{v_2, \ldots, v_l\}$. By definition, y_{t-1} is not adjacent to any vertex of $\{v_2, \ldots, v_{g-2}\} \cup \{v_l, v_{l-1}, \ldots, v_{l-g+4}\}$. Moreover, y_{t-1} cannot be adjacent to both v_{g-1} and v_{l-g+3}. The reason is that in that case G has a cycle of length $l-g+3-(g-2)+1 = l-2g+6 \leq 2g+\lceil \frac{g-2}{4} \rceil -2-2g+6 < 3g-6-2g+6 = g$. We assume without loss of generality that y_{t-1} is not adjacent to v_{g-1}. Then we find that G has an induced path $x_{g-1}x_{g-2}\ldots x_2 y_1 \ldots y_t v_2 \ldots v_{g-1}$ on $2g+t-4$ vertices, which is not possible as $2g+t-4 \geq 2g+s-2 = r$ and G is P_r-free. We conclude that H is bipartite.

Using properties 1 and 2 and the fact that H is bipartite, we colour G as follows. First suppose that g is even. For $1 \leq i \leq g/2$, we colour x_{2i-1} with colour 1 and x_{2i} with colour 2. Then we colour the vertices of the (independent) sets $N^{=1}(U), \ldots, N^{=s}(U)$ with colours 1 and 3, where we alternate the colours starting with colour 3. Finally we colour the vertices of the (bipartite) graph H with colours 1 and 2 if $N^{=s}(U)$ was coloured by 3, and with colours 2 and 3 otherwise.

Now suppose that g is odd. Then $g \geq 11$. Let $X \subseteq N^{=1}(U)$ be the set of vertices adjacent to x_g. By property 2, the vertices of X are not adjacent to any vertex of $\{x_1, \ldots, x_{g-1}\}$. Because $g \geq 11$, we have $s \geq 2$. For $1 \leq i \leq \lfloor g/2 \rfloor$, we colour x_{2i-1} with colour 1 and x_{2i} with colour 2. Then we colour x_g and the vertices of the (independent) set $N^{=1}(U) \backslash X$ with colour 3, and the vertices of X with colour 1. Then we colour the vertices of the (independent) sets $N^{=2}(U), \ldots, N^{=s}(U)$ with colour 2 and 3, where we alternate the colours starting with colour 2. Finally, we colour the vertices of the (bipartite) graph H with colours 1 and 3 if $N^{=s}(U)$ was coloured by 2, and with colours 1 and 2 otherwise. It takes linear time to colour the vertices in $\{x_1, ..., x_g\} \cup N^{=1}(U) \cup, ..., \cup N^{=s}(U)$. It also takes linear time to colour the bipartite graph H. Hence, the algorithm performs in linear time overall. This completes the proof of Theorem 2.4.4. □

Chapter 3

Precolouring Extension

The main ingredients of this chapter are from the following papers.

Section 3.1:

[43] P.A. Golovach, D. Paulusma and J. Song. Closing complexity gaps for coloring problems on H-free graphs. In: *Proceedings of the 23nd International Symposium on Algorithms and Computation (ISAAC 2012)*, *Lecture Notes in Computer Science*, to appear.

Section 3.2:

[41] P.A. Golovach, D. Paulusma and J. Song. 4-Coloring H-free graphs when H is small. *Discrete Applied Mathematics*, 161:140–150, 2013.

[42] P.A. Golovach, D. Paulusma and J. Song. 4-Coloring H-free graphs when H Is small. In: *Proceedings of the 38th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2012)*, volume 7147 of *Lecture Notes in Computer Science*, 289–300, 2012.

Section 3.3:

[19] H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song. Updating the complexity status of coloring graphs without a fixed induced linear forest. *Theoretical Computer Science*, 414:9–19, 2012.

[18] H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song. Narrowing down the gap on the complexity of coloring Pk-free graphs. In: *Proceedings of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2010)*, volume 6410 of *Lecture Notes in Computer Science*, pages 63–74, 2010.

In this chapter, we investigate the problem Precolouring Extension and the problem k-Precolouring Extension. We first give a computational complexity classification for Precolouring Extension in Section 3.1. In particular, we show that for a fixed graph H, Precolouring Extension is polynomial-time solvable for H-free graphs if H is an induced subgraph of P_4 or of $P_1 + P_3$; otherwise Precolouring Extension is NP-complete for H-free graphs. Next, based on the algorithm for solving 4-Colouring for $(P_2 + P_3)$-free graphs that is introduced in Section 2.1, we present a polynomial-time algorithm for solving 4-Precolouring Extension for $(P_2 + P_3)$-free graphs in Section 3.2. In contrast, we show that 4-Precolouring Extension on P_7-free graphs is NP-complete in Section 3.3 by using a similar approach for proving Theorem 2.2.3 which states that 4-Colouring on P_8-free graphs is NP-complete.

3.1 Precolouring Extension for H-free Graphs

We use statement (i) of Theorem 1.3.1 and a number of other results from the literature to obtain the following dichotomy, which complements statement (i) of Theorem 1.3.1.

Theorem 3.1.1. *Let H be a fixed graph. If H is an induced subgraph of P_4 or of P_1+P_3, then* Precolouring Extension *can be solved in polynomial time for H-free graphs; otherwise it is* NP*-complete for H-free graphs.*

Theorem 3.1.1 shows that Precolouring Extension is polynomial-time solvable on $(P_1 + P_3)$-free graphs, which contain the class of $3P_1$-free graphs, i.e., complements of triangle-free graphs. As such, Theorem 3.1.1 also generalizes a result of Hujter and Tuza [56] who showed that Precolouring Extension is polynomial-time solvable on complements of bipartite graphs. Below we prove Theorem 3.1.1.

Proof. Let H be a fixed graph. If H is not an induced subgraph of P_4 or of $P_1 + P_3$, then the statement (i) of Theorem 1.3.1 tells us that Colouring, and consequently, Precolouring Extension is NP-complete for H-free graphs. Jansen and Scheffler [59] showed that Precolouring Extension is polynomial-time solvable for P_4-free graphs. Hence, we are left with the case $H = P_1 + P_3$.

Let (G, k, ϕ_W) be an instance of Precolouring Extension, where G is a (P_1+P_3)-free graph, k is an integer and $\phi_W : W \to \{1, \ldots, k\}$ is a precolouring defined on some subset $W \subseteq V(G)$. We first prove how to transform (G, k, ϕ_W) in polynomial time into a new instance $(G', k', \phi_{W'})$ with the following properties:

(i) G' is a $3P_1$-free subgraph of G, $k' \leq k$ and $\phi_{W'} : W' \to \{1, \ldots, k\}$ is the restriction of ϕ_W to some $W' \subseteq W$;

(ii) $(G', k', \phi_{W'})$ is a yes-instance if and only if (G, k, ϕ_W) is a yes-instance.

Suppose that G is not $3P_1$-free already. Then G contains at least one triple T of three independent vertices. Let $u \in T$. Here we make the following choice if possible: if there exists a triple of three independent vertices that intersects with W, then we choose T to be such a triple and pick $u \in T \cap W$.

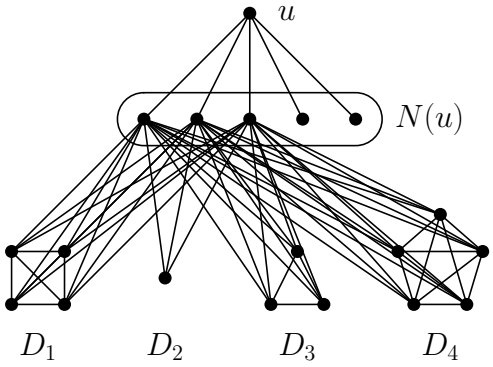

Figure 3.1: An example of a graph G that shows a vertex u that belongs to a set of (at least) three independent vertices and the corresponding complete graphs $D_1, \ldots, D_p$. In this example, $p = 4$ and edges between vertices in $N(u)$ have not been displayed. Also note that the vertices in $V(D_1) \cup \cdots \cup V(D_4)$ have the same neighbours in $N(u)$, as stated in Claim 1.

Let $S = V(G) \setminus (\{u\} \cup N(u))$. Because G is $(P_1 + P_3)$-free, $G[S]$ is the disjoint union of a set of complete graphs $D_1, \ldots, D_p$ for some $p \geq 2$; note that $p \geq 2$ holds, because the other two vertices of T must be in different graphs D_i and D_j. We refer to Figure 3.1 for an example. We will use the following claim.

Claim 1. Every vertex in $V(D_1) \cup \cdots \cup V(D_p)$ is adjacent to exactly the same vertices in $N(u)$.

We prove Claim 1 as follows. First suppose that w and w' are two vertices in two different graphs D_i and D_j, such that w is adjacent to some vertex $v \in N(u)$. Then w' is adjacent to v, as otherwise w' and u, v, w form an induced $P_1 + P_3$ in G, which is not possible. Now suppose that w and w' are two vertices in the same graph D_i, say D_1, such that w is adjacent to some vertex $v \in N(u)$. Because $p \geq 2$, the graph D_2 is nonempty. Let w^*

be in D_2. As we just showed, the fact that w is adjacent to v implies that w^* is adjacent to v as well. By repeating this argument with respect to w^* and w', we then find that w' is adjacent to v. Hence, we have proven Claim 1.

We now proceed as follows. First suppose that $u \in W$. Let $\phi_W(u) = x$. We assign colour x to an arbitrary vertex of every D_i that does not contain a vertex coloured with x already and that contains at least one vertex outside W. Now suppose that $u \notin W$. Then by our choice of u no vertex from $V(D_1) \cup \cdots \cup V(D_p)$ belongs to W. In that case, u must be adjacent to every vertex of W. We let x be a new colour not used by ϕ_W, and we assign x to u and also to an arbitrary vertex of every D_i. Afterward, in both cases, we remove all vertices coloured x from G. We let G' denote the resulting graph, and we let $W' \subseteq W$ denote the resulting set of precoloured vertices. We observe the following. If $u \in W$, then by symmetry we may assume that $\phi_W(u) = x = k$. If $u \notin W$, then we already deduced that u is adjacent to every vertex of W. If every colour of $\{1, \ldots, k\}$ is used on W by ϕ_W, then (G, k, ϕ_W) is a no-instance because there is no colour available for u. As we can detect this situation in polynomial time, we may assume without loss of generality that this is not the case. Then, by symmetry, we may assume that $\phi_W(W) \subseteq \{1, \ldots, k-1\}$, and consequently, we can take $x = k$ in this case as well. We now prove that $(G', k-1, \phi_{W'})$ is a yes-instance if and only if (G, k, ϕ_W) is a yes-instance.

First suppose that $(G', k-1, \phi_{W'})$ is a yes-instance. Then G' allows a $(k-1)$-colouring ϕ' that extends $\phi_{W'}$. We assign colour k to every vertex that we removed from G. Because those vertices form an independent set of G, this results in a k-colouring ϕ of G that extends $\phi_{W'}$. Because every vertex removed from W had colour k and because $W' \subseteq W$, we find that ϕ is a colouring of G that extends ϕ_W.

Now suppose that (G, k, ϕ_W) is a yes-instance. Then G allows a k-colouring ϕ that extends ϕ_W. Let $V^* \subseteq V(D_1) \cup \cdots \cup V(D_p)$ be the set of vertices that we removed from G besides vertex u, so $V(G') = V(G) \setminus (\{u\} \cup V^*)$. We note that our algorithm would assign colour k to every vertex of $\{u\} \cup V^*$, whereas ϕ may colour a vertex from V^* with a colour different from k. However, whenever $v \in \{u\} \cup V^*$ belongs to W, we do have $\phi(v) = k$. The reason is that both ϕ and the colouring prescribed by our algorithm give such a vertex v the same colour, as they both extend ϕ_W, and the algorithm would assign colour k to v. We first show how to modify ϕ such that it assigns colour k to all the other vertices of $\{u\} \cup V^*$ as well.

Let v be a vertex in $(\{u\} \cup V^*) \setminus W$ with $\phi(v) \neq k$. First suppose that $v = u$. As $u \notin W$, our choice of u implies that $W \subseteq N(u)$, and consequently, $\phi_W(W) \subseteq \{1, \ldots, k-1\}$ as we already argued. Because $W \subseteq N(u)$, we find that ϕ_W does not use colour $\phi(u)$. Because $\phi_W(W) \subseteq \{1, \ldots, k-1\}$, we also find that ϕ_W does not use colour k. Then, by symmetry, we may modify ϕ by assigning colour k to every vertex of G that had colour $\phi(u)$ and vice versa. Hence, we may assume without loss of generality that $\phi(u) = k$.

Now suppose that $v \in V^*$. Then $v \in V(D_i)$ for some $1 \leq i \leq p$. If D_i does not contain a vertex w with colour $\phi(w) = k$, then we change the colour of v into k. We may do so for the following two reasons. First, all neighbours of v outside D_i are adjacent to u with $\phi(u) = k$, and as such these neighbours of v did not receive colour k from ϕ. Hence, ϕ does not assign colour k to any neighbour of v in G. Second, $v \notin W$ by assumption, which means that we still extend ϕ_W when we change $\phi(v)$. If D_i does contain a vertex w with colour $\phi(w) = k$, then we swap the colours of v and w. We may do so for the following three reasons. First, w is the only neighbour of v with colour k, and v is the only neighbour of w with colour $\phi(v)$, because D_i is a complete graph, and because v and w have exactly the same set of neighbours outside D_i due to Claim 1. Second, $v \notin W$ by assumption. Third, $w \notin W$, as otherwise $\phi_W(w) = k$ because ϕ extends ϕ_W, and in that case we would have put w in V^* instead of v.

Due to the above, we may assume without loss of generality that every vertex $v \in \{u\} \cup V^*$ has colour k. Recall that our algorithm puts a vertex from every D_i in V^* unless $V(D_i) \subseteq W$ and D_i contains no vertex precoloured with colour k by ϕ_W. We then find that every vertex from $V(G) \setminus (\{u\} \cup V^*)$ that is not precoloured by ϕ_W is adjacent to a vertex $\{u\} \cup V^*$, i.e., to a vertex that received colour k from ϕ. Because the vertices of $V(G) \setminus (\{u\} \cup V^*)$ that are precoloured by ϕ_W have a colour not equal to k, this means that the set of vertices in G that are given colour k by ϕ is $\{u\} \cup V^*$. Consequently, the restriction ϕ' of ϕ to $V(G') = V(G) \setminus (\{u\} \cup V^*)$ is a $(k-1)$-colouring of G'. Because ϕ extends ϕ_W and $W' = W \setminus (\{u\} \cup V^*)$, we find that ϕ' is a colouring of G' that extends $\phi_{W'}$.

We observe that $(G', k-1, \phi_{W'})$ satisfies condition (i) except that G' may not be $3P_1$-free. Therefore we repeat the step described above until the resulting graph is $3P_1$-free, and consequently both conditions (i) and (ii) are satisfied. This takes polynomial time in total, because every step takes polynomial time and in every step the number of vertices of the graph reduces by at least 1. Hence, we may assume without loss of

generality that in our initial instance (G, k, ϕ_W), the graph G is $3P_1$-free.

We now apply the algorithm of Hujter and Tuza [56] for solving Precolouring Extension on complements of bipartite graphs. Because G is $3P_1$-free, G has no three mutually nonadjacent vertices. Suppose that u and v are two nonadjacent vertices in W. Then every vertex of $V(G) \setminus \{u, v\}$ is adjacent to at least one of $\{u, v\}$. This means that we can remove u, v if they are both coloured alike by ϕ_W in order to obtain a new instance $(G - \{u, v\}, k - 1, \phi_{W \setminus \{u,v\}})$ that is a yes-instance of Precolouring Extension if and only if (G, k, ϕ_W) is a yes-instance. If u and v are coloured differently by ϕ_W, then we add an edge between them. We perform this step for any pair of non-adjacent vertices in W. Afterward, we have found in polynomial time a new instance (G^*, k^*, ϕ_{W^*}) with the following properties. First, $|V(G^*)| \leq |V(G)|$, $k^* \leq k$ and $\phi_{W^*} : W^* \to \{1, \ldots, k\}$ is a precolouring defined on some clique W^* of G^*. Second, (G^*, k^*, ϕ_{W^*}) is a yes-instance if and only if (G, k, ϕ_W) is a yes-instance. Hence, we may consider (G^*, k^*, ϕ_{W^*}) instead. Because W^* is a clique, we find that (G^*, k^*, ϕ_{W^*}) is a yes-instance if and only if G^* is k^*-colourable. Because G is $3P_1$-free and G^* is obtained by only removing vertices from G, we find that G^* is $3P_1$-free as well. This means that we can solve the Colouring problem with input (G^*, k^*) by using the statement (i) of Theorem 1.3.1 (which in this particular case comes down to computing the size of a maximum matching in the complement of G^*). The running time of our transformation is $O(n^2)$. The running time of the algorithm by Hujter and Tuza [56] is $O(n^3)$. Therefore the overall running time of our algorithm is $O(n^3)$. This completes the proof for the case $H = P_1 + P_3$, and we have proven Theorem 3.1.1. □

3.2 4-Precolouring Extension for $(P_2 + P_3)$-free Graphs

In this section, we present a polynomial-time algorithm for solving 4-Precolouring Extension on $(P_2 + P_3)$-free graphs by generalizing Theorem 2.1.14 in the following way.

Theorem 3.2.1. *The* 4-Precolouring Extension *problem can be solved in polynomial time for* $(P_2 + P_3)$*-free graphs.*

Proof. Let $G = (V, E)$ be a $(P_2 + P_3)$-free graph with a set $W \subseteq V$ such that each vertex in W is precoloured with a colour from $\{1, 2, 3, 4\}$. We may assume without loss of generality that every vertex in $V \setminus W$ has degree at least 4. This can be seen

as follows. We consecutively remove vertices of $V \setminus W$ with degree at most 3 from G until this is no longer possible. We denote the remaining graph, which we obtain in polynomial time, by G^*. Because we only removed vertices, G^* is $(P_2 + P_3)$-free. Then due to Proposition 1.2.2, G has a 4-colouring extending the precolouring of W if and only if G^* has a 4-colouring extending the precolouring of W. Because G^* is also $2P_3$-free, we may apply Lemma 2.1.3 for $k = 4$ and $s = 2$ to find a set D of at most 39 vertices that dominates $V \setminus W$ in the case that G^* has a 4-colouring extending the precolouring of W. We put all vertices of W in D and run the remainder of the algorithm of Theorem 2.1.14 for graph $G[V(G^*) \cup W]$ under the additional condition that we let the algorithm only consider 4-colourings of D that do not change the colours of the vertices of W as prescribed by the given precolouring of W. As such, the number of different 4-colourings of D considered by the algorithm is still at most 4^{39}. Hence, the correctness proof and running time analysis are exactly the same as in the proof of Theorem 2.1.14. □

3.3 4-Precolouring Extension for P_7-free Graphs

In this section we show that 4-Precolouring Extension is NP-complete for the class of P_7-free graphs. We use a reduction from the Not-All-Equal 3-Satisfiability problem with un-negated literals only. Recall that this problem is NP-complete [88]. We consider an arbitrary instance I of Not-All-Equal 3-Satisfiability that has variables $\{x_1, x_2, \ldots, x_n\}$ and clauses $\{C_1, C_2, \ldots, C_m\}$, and we define the graph G_I^* from Section 2.3 with a precolouring on some vertices of G_I^*. Let us first repeat the construction of G_I^*.

- For each clause C_j we introduce a gadget with vertex set

$$\{a_{j,1}, a_{j,2}, a_{j,3}, b_{j,1}, b_{j,2}, c_{j,1}, c_{j,2}, c_{j,3}, d_{j,1}, d_{j,2}\}$$

and edge set

$$\{a_{j,1}c_{j,1}, a_{j,2}c_{j,2}, a_{j,3}c_{j,3}, b_{j,1}c_{j,1}, c_{j,1}d_{j,1}, d_{j,1}c_{j,2}, c_{j,2}d_{j,2}, d_{j,2}c_{j,3}, c_{j,3}b_{j,2}, b_{j,2}b_{j,1}\},$$

and a disjoint gadget called the *copy* with vertex set

$$\{a'_{j,1}, a'_{j,2}, a'_{j,3}, b'_{j,1}, b'_{j,2}, c'_{j,1}, c'_{j,2}, c'_{j,3}, d'_{j,1}, d'_{j,2}\}$$

and edge set

$$\{a'_{j,1}c'_{j,1}, a'_{j,2}c'_{j,2}, a'_{j,3}c'_{j,3}, b'_{j,1}c'_{j,1}, c'_{j,1}d'_{j,1}, d'_{j,1}c'_{j,2}, c'_{j,2}d'_{j,2}, d'_{j,2}c'_{j,3}, c'_{j,3}b'_{j,2}, b'_{j,2}b'_{j,1}\}.$$

We say that all these vertices (so including the vertices in the copy) are of a-type, b-type, c-type and d-type, respectively. They induce $2m$ disjoint 10-vertex components in G_I^* which we will call *clause-components*. We precolour every $a_{j,h}$ by 1 and every $a'_{j,h}$ by 2.

- Every variable x_i will be represented by a vertex in G_I^*, and we say that these vertices are of x-type.

- For every clause C_j we fix an arbitrary order of its variables $x_{i_1}, x_{i_2}, x_{i_3}$ and add edges $c_{j,h}x_{i_h}$ and $c'_{j,h}x_{i_h}$ for $h = 1, 2, 3$.

- We add an edge between every x-type vertex and every b-type vertex. We also add an edge between every x-type vertex and every d-type vertex.

- We add an edge between every a-type vertex and every b-type vertex. We also add an edge between every a-type vertex and every d-type vertex.

In Figure 3.2 we illustrate an example in which C_j is a clause with ordered variables $x_{i_1}, x_{i_2}, x_{i_3}$. The thick edges indicate the connection between the variables vertices and the c-type vertices of the two copies of the clause gadget. The dashed thick edges indicate the connections between the (precoloured) a-type and c-type vertices of the two copies of the clause gadget. We omitted the indices from the labels of the clause gadget vertices to increase the visibility.

The following lemma states that the graph G_I^* is P_7-free. This has been proven implicitly in Lemma 2.3.4 but for clarity reasons we provide an explicit proof here.

Lemma 3.3.1. *The graph G_I^* is P_7-free.*

Proof. Let P be an induced path in G_I^*. We show that P has at most six vertices. We distinguish the following cases.

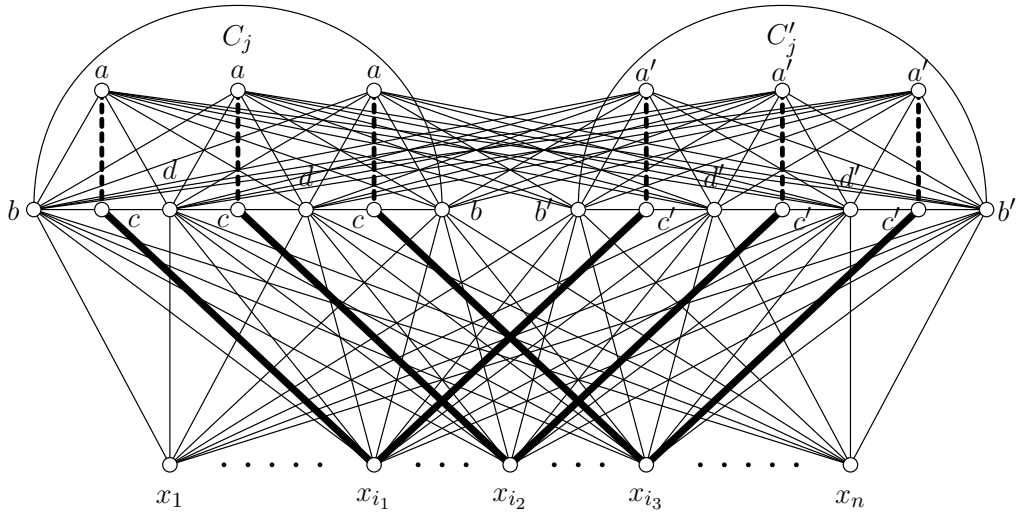

Figure 3.2: The graph G_I^* for the clause $C_j = \{x_{i_1}, x_{i_2}, x_{i_3}\}$.

Case 1. P contains no a-type vertex.

Let I' be the instance obtained from I after adding a copy of every clause. Then P is contained in a graph isomorphic to the graph $G_{I'}$ as defined in Section 2.2 after removing the negative variables. Hence, by Lemma 2.2.1, P contains at most six vertices.

Case 2. P contains exactly one a-type vertex.

We may assume without loss of generality that $a_{1,1}$ is this vertex.

Suppose $a_{1,1}$ is an end-vertex of P. If the (only) neighbour of $a_{1,1}$ is a b-type or d-type vertex, then P contains no other b-type or d-type vertex. This implies that P can contain at most two other vertices, namely one x-type vertex and one c-type vertex. Suppose the neighbour of $a_{1,1}$ on P is a c-type vertex (so this neighbour must be $c_{1,1}$). Then P contains no b-type vertex and no d-type vertex. Hence P contains at most two other vertices, namely an x-type vertex and another c-type vertex. Hence P has at most four vertices.

Suppose $a_{1,1}$ is not an end-vertex of P. Suppose $c_{1,1}$ is a neighbour of $a_{1,1}$ on P. Let z be the other neighbour of $a_{1,1}$ on P. Then z must be a b-type or d-type vertex. This means that P contains no other b-type or d-type vertex, and if P contains an x-type vertex this vertex must be adjacent to z. Then $c_{1,1}$ is an end-vertex of P, and P contains

at most two other vertices, namely an x-type vertex and another c-type vertex. Hence P has at most five vertices.

Suppose $c_{1,1}$ is not a neighbour of $a_{1,1}$ on P. Then both neighbours of $a_{1,1}$ are b-type or d-type vertices. This means that P contains no x-type vertex. Consequently, P can contain at most two other (c-type) vertices. Hence P has at most five vertices.

Case 3. P contains exactly two a-type vertices.
Suppose P contains no b-type vertex and no d-type vertex. Then the a-type vertices must be the end-vertices of P that must be joined in P by two c-type vertices and one x-type vertex. Hence P has five vertices.

Suppose P contains a vertex z that is a b-type or d-type vertex. Because such a vertex is adjacent to both a-type vertices, P contains no other b-type or d-type vertex and P does not contain an x-type vertex. Then P might only contain at most two other vertices (that are of c-type). This means that P has at most five vertices.

Case 4. P contains at least three a-type vertices.
Then P contains no b-type vertex and no d-type vertex. Hence, every a-type vertex of P must be an end-vertex of P. This is not possible and completes the proof of Lemma 3.3.1. □

By Lemmas 2.3.5 and 3.3.1 we obtain the main result of this section.

Theorem 3.3.2. *The* 4-Precolouring Extension *problem is* NP*-complete for P_7-free graphs.*

Chapter 4

List Colouring

The main ingredients of this chapter are from the following papers.

Section 4.1, 4.2, 4.5:

[43] P.A. Golovach, D. Paulusma and J. Song. Closing complexity gaps for coloring problems on H-free graphs. In: *Proceedings of the 23nd International Symposium on Algorithms and Computation (ISAAC 2012)*, *Lecture Notes in Computer Science*, to appear.

Section 4.3, 4.4:

[19] H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song. Updating the complexity status of coloring graphs without a fixed induced linear forest. *Theoretical Computer Science*, 414:9–19, 2012.

[18] H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song. Narrowing down the gap on the complexity of coloring Pk-free graphs. In: *Proceedings of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2010)*, volume 6410 of *Lecture Notes in Computer Science*, pages 63–74, 2010.

Section 4.6:

[44] P.A. Golovach, D. Paulusma and J. Song. Coloring graphs without short cycles and long induced paths. In: *Proceedings of the 18th International Symposium on Fundamentals of Computation Theory (FCT 2011)*, volume 6914 of *Lecture Notes in Computer Science*, pages 193–204, 2011.

In this chapter, we focus on the List Colouring problem and its two variants List k-Colouring and k-List Colouring. We first present a computational complexity classification for k-List Colouring on H-free graphs in Section 4.1, which immediately yields a complexity classification for List Colouring on H-free graphs. In particular, we show that for a fixed integer ℓ and a fixed graph H, ℓ-List Colouring is polynomial-time solvable on H-free graphs if $\ell \leq 2$ or H is an induced subgraph of P_3; otherwise ℓ-List Colouring is NP-complete for H-free graphs. We then show that 3-List Colouring for $(3P_1, P_1+P_2)$-free graphs is NP-complete in Section 4.2. Next we settle the computational complexity of List 3-Colouring for (P_2+P_4)-free graphs and for sP_3-free graphs for any fixed s by presenting a polynomial-time algorithm for each of the graph classes in Section 4.3 and Section 4.4, respectively. In particular, the algorithm for List 3-Colouring on sP_3-free graphs relies on structural properties of 3-colourable sP_3-free graphs as we showed in Section 2.1. We then show that List 4-Colouring on P_6-free graphs is NP-complete in Section 4.5 by modifying the construction of the gadget used for proving Theorem 3.3.2 which states that 4-Precolouring Extension is NP-complete for P_7-free graphs. Finally, we show a polynomial-time algorithm for List k-Colouring on $(K_{s,t}, P_r)$-free graphs for all integers $k, r, s, t \geq 1$ in Section 4.6.

4.1 List Colouring and ℓ-List Colouring for H-free Graphs

In this section we classify List Colouring and ℓ-List Colouring on H-free graphs. In order to show these results, we need the following lemma, which is well-known (cf. [10]). We give its proof in order to explain the bound on the running time stated in this lemma.

Lemma 4.1.1. List Colouring *can be solved in* $O((n+k)^{\frac{5}{2}})$ *time on* n*-vertex complete graphs with a* k*-list assignment.*

Proof. Let $G = (V, E)$ be a complete graph on n vertices $u_1, \ldots, u_n$ with some k-list assignment $\mathcal{L}$. Let $V(G) = \{u_1, \ldots, u_n\}$. Then we construct a bipartite graph B as follows. One partition class of B consists of n vertices $u_1, \ldots, u_n$, whereas the other partition class consists of vertices $1, \ldots, k$. We add an edge between two vertices u_i and j if and only if $j \in L(u_i)$. Now G has a colouring that respects $\mathcal{L}$ if and only if B has a matching that contains an edge with u_i as one of its end-vertices for $i = 1, \ldots, n$. We can solve the latter problem in $O((n+k)^{\frac{5}{2}})$ time by using the Hopcroft-Karp algorithm [55]. □

Note that Lemma 4.1.1 implies that List Colouring is polynomial-time solvable on H-free graphs, whenever H is an induced subgraph of P_3. For all other graphs H, List Colouring is NP-complete for H-free graphs. This follows from our next result.

Theorem 4.1.2. *Let ℓ be a fixed integer, and let H be a fixed graph. If $\ell \leq 2$ or H is an induced subgraph of P_3, then ℓ-List Colouring is polynomial-time solvable on H-free graphs; otherwise ℓ-List Colouring is NP-complete for H-free graphs.*

Proof. Early papers by Erdös, Rubin and Taylor [32] and Vizing [94] already observed that 2-List Colouring is polynomial-time solvable on general graphs. Hence, we can focus on the case $\ell \geq 3$. Because the ℓ-Colouring problem is a special case of the ℓ-List Colouring problem, the following results are useful. Kamiński and Lozin [61] showed that for any $k \geq 3$, the k-Colouring problem is NP-complete for the class of graphs of girth (recall that the girth is the length of a shortest induced cycle) at least p for any fixed $p \geq 3$. Their result implies that for any $\ell \geq 3$, the ℓ-Colouring problem, and consequently, the ℓ-List Colouring problem is NP-complete for the class of H-free graphs whenever H contains a cycle.

The proof of Theorem 4.5 in the paper by Jansen and Scheffler [59] is to show that 3-List Colouring is NP-complete on P_4-free graphs but as a matter of fact shows that 3-List Colouring is NP-complete on complete bipartite graphs, which are (P_1+P_2)-free. The proof of Theorem 11 in the paper by Jansen [58] is to show that List Colouring is NP-complete for (not necessarily vertex-disjoint) unions of two complete graphs but in fact shows that 3-List Colouring is NP-complete for these graphs. As the union of two complete graphs is $3P_1$-free, this means that 3-List Colouring is NP-complete for $3P_1$-free graphs.

The above results leave us with the case when H is an induced subgraph of P_3. By Lemma 4.1.1 we can solve List Colouring in polynomial time on complete graphs. This means that we can solve ℓ-List Colouring in polynomial time on connected P_3-free graphs for any $\ell \geq 1$. Hence we have proven Theorem 4.1.2. □

4.2 3-List Colouring for Complete Graphs Minus a Matching

We prove that 3-List Colouring is NP-complete for complete graphs minus a matching. In order to prove this we use a reduction from a variant of Not-All-Equal

3-Satisfiability with un-negated literals only, which we denote as Not-All-Equal $(\leq 3, 2/3)$-Satisfiabili-ty with un-negated literals. Recall that the problem Not-All-Equal 3-Satisfiability with un-negated literals only is NP-complete [88]. The problem Not-All-Equal $(\leq 3, 2/3)$-Satisfiability with un-negated literals takes as input an instance I that has a set of variables $X = \{x_1, \ldots, x_n\}$ and a set of literal clauses $C = \{C_1, \ldots, C_m\}$ over X with the following properties. Each C_i contains either 2 or 3 literals, and these literals are all un-negated. Moreover, each literal occurs in at most three different clauses. The questions is, does there exist a truth assignment for X such that each clause contains at least one true literal and at least one false literal? One can prove that Not-All-Equal $(\leq 3, 2/3)$-Satisfiability is NP-complete by a reduction from Not-All-Equal-3-Satisfiability via a well-known folklore trick.[1]

Let I be an arbitrary instance of Not-All-Equal $(\leq 3, 2/3)$-Satisfiability with un-negated literals. We let $x_1, x_2, \ldots, x_n$ be the variables of I, and we let $C_1, C_2, \ldots, C_m$ be the clauses of I. We first define a graph G_I with a list assignment $\mathcal{L}$ of size three. We then show that G_I is a complete graph minus a matching, and that G_I has a colouring that respects $\mathcal{L}$ if and only if I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal.

We construct G_I and $\mathcal{L}$ as follows.

- We let $a_i \neq b_j$ if $a \neq b$ or $i \neq j$. We represent every variable x_i by a vertex with $L(x_i) = \{1_i, 2_i\}$ in G_I. We say that these vertices are of x-type and these colours are of 1-type and 2-type, respectively.

- For every clause C_p with two variables we fix an arbitrary order of its variables x_h, x_i and we introduce a set of vertices $C_p, a_{p,h}, a_{p,i}, b_{p,h}, b_{p,i}$ that have lists of admissible colours $\{3_p, 4_p\}$, $\{1_h, 3_p\}$, $\{1_i, 4_p\}$, $\{2_h, 4_p\}$, $\{2_i, 3_p\}$, respectively, and we add edges $C_p a_{p,h}$, $C_p b_{p,h}$, $C_p a_{p,i}$, $C_p b_{p,i}$, $a_{p,h} x_h$, $b_{p,h} x_h$, $a_{p,i} x_i$, $b_{p,i} x_i$.

  For every clause C_p with three variables we fix an arbitrary order of its variables x_h, x_i, x_j and we introduce a set of vertices $C_p, a_{p,h}, a_{p,i}, a_{p,j}, b_{p,h}, b_{p,i}, b_{p,j}$ that have lists of admissible colours $\{3_p, 4_p, 5_p\}$, $\{1_h, 3_p\}$, $\{1_i, 4_p\}$, $\{1_j, 5_p\}$, $\{2_h, 5_p\}$, $\{2_i, 3_p\}$, $\{2_j, 4_p\}$, respectively, and we add edges $C_p a_{p,h}$, $C_p b_{p,h}$, $C_p a_{p,i}$, $C_p b_{p,i}$, $C_p a_{p,j}$, $C_p b_{p,j}$,

[1] If a literal x appears in $t \geq 4$ clauses $C_1, \ldots, C_t$, then we replace x by $2t$ new literals $x_1, \ldots, x_{2t}$ and add $2t$ new clauses $(x_1, \overline{x_2}), (x_2, \overline{x_3}), \ldots, (x_{2t}, \overline{x_1})$, which guarantee that $x_1, x_3, \ldots, x_{2t-1}$ have the same values in any satisfying truth assignment. Hence, we may replace x by x_{2i-1} in C_i for $i = 1, \ldots, t$.

$a_{p,h}x_h$, $b_{p,h}x_h$, $a_{p,i}x_i$, $b_{p,i}x_i$, $a_{p,j}x_j$, $b_{p,j}x_j$.

We say that the new vertices are of C-type, a-type and b-type, respectively. We say that the new colours are of 3-type, 4-type and 5-type, respectively.

- For each variable x_j that occurs in three clauses we fix an arbitrary order of the clauses C_p, C_q, C_r, in which it occurs. Then we do as follows. First, we modify the lists of $a_{p,j}$, $a_{q,j}$, $b_{p,j}$ and $b_{q,j}$. In $L(a_{p,j})$ we replace colour 1_j with a new colour $1'_j$. In $L(a_{q,j})$ we replace colour 1_j with a new colour $1''_j$. In $L(b_{p,j})$ we replace colour 2_j with a new colour $2'_j$. In $L(b_{q,j})$ we replace colour 2_j with a new colour $2''_j$. Next we introduce four vertices $a'_{p,j}, a'_{q,j}, b'_{p,j}, b'_{q,j}$ with lists of admissible colours $\{1_j, 1'_j\}, \{1'_j, 1''_j\}, \{2_j, 2'_j\}, \{2'_j, 2''_j\}$, respectively. We say that these vertices are of a'-type or b'-type, respectively. We say that the new colours are also of 1-type or 2-type, respectively. We add edges $a_{p,j}a'_{p,j}$, $a'_{p,j}a'_{q,j}$, $a'_{p,j}x_j$, $a_{q,j}a'_{q,j}$, $b_{p,j}b'_{p,j}$, $b'_{p,j}b'_{q,j}$, $b'_{p,j}x_j$, $b_{q,j}b'_{q,j}$. For each variable x_j that occurs in at most two clauses, we do not do anything additional.

- We add an edge between any two not yet adjacent vertices of G_I whenever they have no common colour in their lists.

In Figure 4.1 we give an example, where in order to increase the visibility we display the complement graph $\overline{G_I}$ of G_I instead of G_I itself.

As can be seen from Figure 4.1, the graph $\overline{G_I}$ is isomorphic to the disjoint union of a number of P_1s and P_2s. This means that G_I is a complete graph minus a matching. We formally prove this statement in Lemma 4.2.1, whereas the hardness reduction is stated in Lemma 4.2.2.

Lemma 4.2.1. *The graph G_I is a complete graph minus a matching.*

Proof. Let $z \in V(G)$. Then we obtain the following due to the construction of G_I. If z is of C-type or x-type, then z is adjacent to all other vertices in G_I. If z is of a-type or b-type, then z is adjacent to all vertices of G_I except perhaps one vertex, which is of a'-type or b'-type, respectively. Finally, if z is of a'-type or b'-type, then z is adjacent to all vertices of G_I except one vertex, which is of a-type or b-type, respectively. We conclude that every vertex in G_I has degree at least $|V(G_I)| - 2$. Recall that complete graphs

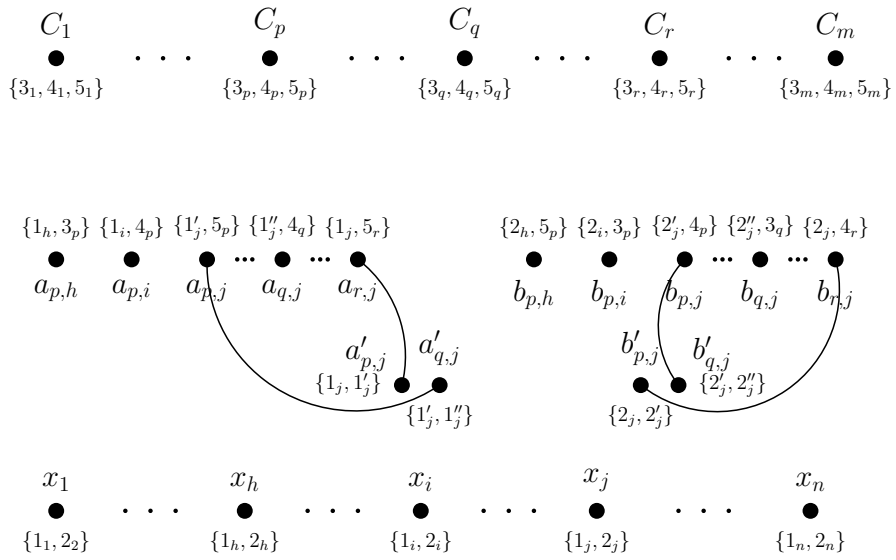

Figure 4.1: An example of a graph $\overline{G_I}$ in which a clause C_p and a variable x_j are highlighted. Note that in this example C_p is a clause with ordered variables x_h, x_i, x_j, and that x_j is a variable contained in ordered clauses C_p, C_q and C_r.

minus a matching are exactly those graphs that are $(3P_1, P_1 + P_2)$-free, or equivalently, graphs of minimum degree at least $n - 2$, where n is the number of vertices. Hence G_I is a complete graph minus a matching. □

Lemma 4.2.2. *The graph G_I has a colouring that respects $\mathcal{L}$ if and only if I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal.*

Proof. First suppose that G_I has a colouring that respects $\mathcal{L}$. Consider a variable x_j contained in ordered clauses C_p, C_q, and C_r. If the colour of x_j is 1_j then the colour of $a'_{p,j}$ is $1'_j$. Consequently, the colour of $a'_{q,j}$ is $1''_j$. We conclude that none of the colours of $a_{p,j}$, $a_{q,j}$, $a_{r,j}$ is of 1-type. Similarly, if the colour of x_j is 2_j, then none of the colours of $b_{p,j}$, $b_{q,j}$, $b_{r,j}$ is of 2-type. We use this observation as follows. Consider a clause C_p with three literals ordered as x_h, x_i, x_j. If x_h, x_i and x_j all have a 1-type colour, then $a_{p,h}$, $a_{p,i}$ and $a_{p,j}$ have colours 3_p, 4_p and 5_p, respectively. Then there is no colour available for C_p. A similar argument can be made for clauses that contain only two literals. Hence, we find that at least one literal in every clause is coloured with a 1-type colour. Analogously, we find that at least one literal in every clause is coloured with a 2-type colour. This

means that the truth assignment that sets a literal to TRUE if the corresponding x-type vertex has a 1-type colour, and to FALSE otherwise, is a satisfying truth assignment in which each clause contains at least one true and one false literal.

Now suppose that I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal. We give each x-type vertex that represents a true literal its 1-type colour, whereas we colour all other x-type vertices with their 2-type colour. Consider a variable x_j contained in ordered clauses C_p, C_q, and C_r. If the colour of x_j is 1_j then we colour $b'_{p,j}$ by 2_j and $b'_{q,j}$ by $2'_j$. Consequently, we can colour $b_{p,j}$, $b_{q,j}$, $b_{r,j}$ with their 2-type colour. If the colour of x_j is 2_j then we colour $a'_{p,j}$ by 1_j and $a'_{q,j}$ by $1'_j$. Consequently, we can colour $a_{p,j}$, $a_{q,j}$, $a_{r,j}$ with their 1-type colour. We use this observation as follows. Consider a clause C_p with literals ordered as x_h, x_i, x_j. Due to our observation and the definition of $\mathcal{L}$, we may assume without loss of generality that x_h, x_i are true and x_j is false. Then using our observation we can colour $a_{p,h}$, $a_{p,i}$, $a_{p,j}$, $b_{p,h}$, $b_{p,i}$, $b_{p,j}$ with colours 3_p, 4_p, of 1-type, of 2-type, of 2-type, 4_p, respectively. This means that we can colour C_p by 5_p. A similar argument can be made if C_p consists of two literals only. Hence, we find that G_I has a colouring that respects $\mathcal{L}$. This completes the proof of Lemma 4.2.2. □

By observing that 3-List Colouring belongs to NP and using Lemmas 4.2.1 and 4.2.2, we have proven Theorem 4.2.3.

Theorem 4.2.3. *The* 3-List Colouring *problem is* NP*-complete for complete graphs minus a matching.*

4.3 List 3-Colouring for $(P_2 + P_4)$-free Graphs

In this section we show how to test in polynomial time whether a given $(P_2 + P_4)$-free graph G has a colouring ϕ that respects some given 3-list assignment $\mathcal{L}$ of G.

Theorem 4.3.1. *The* List 3-Colouring *problem can be solved in polynomial time for* $(P_2 + P_4)$*-free graphs.*

Proof. Let G be a (P_2+P_4)-free graph with a list assignment $\mathcal{L}$. We start by making two assumptions. Firstly, we assume that G is connected as otherwise we apply our algorithm on each component of G. Secondly, we assume that G contains an induced subgraph H

isomorphic to P_6. If not, then G would be P_6-free and we could use the polynomial-time algorithm for solving List 3-Colouring for P_6-free graphs [16].

We first guess a colouring of H respecting the restriction of $\mathcal{L}$ to $V(H)$. We then start our algorithm, which we run at most 3^6 times as this is an upper bound on the number of possible 3-colourings of H. From the description of the algorithm it will be immediately clear that its running time is polynomial in $|V(G)|$.

Our algorithm first applies the following subroutine. Let $U \subseteq V(G)$ contain all vertices that have a list consisting of exactly one colour. For every vertex $u \in U$ we remove this single colour $\phi(u)$ from the lists of its neighbours. If this results in an empty list at some vertex, then we output `NO`. We remove u from G and repeat this process in the remaining graph as long as there exists a vertex with a list of size 1. This process is called *updating* the graph. Note that during this procedure we also removed all vertices of H. We restore the vertices of H back into G. We may assume that G is still connected; otherwise, due to the $(P_2 + P_4)$-freeness of G, every component not containing H is a single vertex and can be coloured trivially. Let S be the set of vertices that still have a list of admissible colours of size 3. If $S = \emptyset$, then we can apply Theorem 1.1.1.

Suppose $S \neq \emptyset$. Let T be the set of vertices of $V(G) \setminus V(H)$ that have at least one neighbour in H. Because we coloured every vertex in H and updated G, every vertex of T has a list of exactly two admissible colours, and consequently, $S \cap (V(H) \cup T) = \emptyset$. Since G contains no induced $P_2 + P_4$, we find that $V(G) \setminus (V(H) \cup T)$, and consequently S, is an independent set in G. Since we assume that G is still connected, each vertex in S has at least one neighbour in T (so $T \neq \emptyset$).

For convenience we order the vertices of H along the P_6 as $p_1, p_2, \ldots, p_6$, starting with vertex p_1 with degree 1 in H. Let $T^* \subseteq T$ consist of all vertices in T that have a neighbour in S. Let T_1 denote the subset of vertices of T^* adjacent to p_1, p_3, p_5 and not to p_2, p_4, p_6; let T_2 denote the subset of vertices of T^* adjacent to p_2, p_4, p_6 and not to p_1, p_3, p_5; let T_3 denote the subset of vertices of T^* adjacent to p_2, p_5 and not to p_1, p_3, p_4, p_6.

Claim 1. $T^* = T_1 \cup T_2 \cup T_3$.

We prove Claim 1 as follows. Because $T_1 \cup T_2 \cup T_3 \subseteq T^*$ by definition, we only have to prove that $T^* \subseteq T_1 \cup T_2 \cup T_3$. Let $u \in T^*$. Because u has a list of two admissible colours, u is not adjacent to two adjacent vertices of H (as these vertices have different colours). By definition, u has a neighbour $v \in S$.

Case 1. u is adjacent to p_1.

Then u is not adjacent to p_2. Hence, p_4p_5 and vup_1p_2 form an induced $P_2 + P_4$ in G unless u is adjacent to a vertex of $\{p_4, p_5\}$.

Suppose u is adjacent to p_4. Then u is neither adjacent to p_3 nor to p_5. Recall that u is not adjacent to p_2. Then u must be adjacent to p_6, as otherwise vup_1p_2 and p_5p_6 form an induced $P_2 + P_4$ in G. However, now we find that p_2p_3 and vup_6p_5 form an induced $P_2 + P_4$ in G. We conclude that u cannot be adjacent to p_4. Because u is not adjacent to p_4 and u must be adjacent to a vertex of $\{p_4, p_5\}$, we find that u is adjacent to p_5. Then u is not adjacent to p_6. Recall that u is not adjacent to p_2. This means that u must be adjacent to p_3, as otherwise p_2p_3 and vup_5p_6 form an induced $P_2 + P_4$ in G. Hence, we obtain $u \in T_1$.

Case 2. u is not adjacent to p_1 but u is adjacent to p_2.

Then p_4p_5 and vup_2p_1 form an induced $P_2 + P_4$ in G unless u is adjacent to a vertex from $\{p_4, p_5\}$. Suppose u is adjacent to p_4. Then u is not adjacent to p_5. This means that u is adjacent to p_6, as otherwise p_5p_6 and vup_2p_1 form an induced $P_2 + P_4$ in G. Hence, we obtain $u \in T_2$. Suppose u is not adjacent to p_4. Then u must be adjacent to p_5. This means that u is not adjacent to p_6. Because u is adjacent to p_2, we find that u is not adjacent to p_3. Recall that u is not adjacent to p_1. Hence, we obtain $u \in T_3$.

Case 3. u is neither adjacent to p_1 nor to p_2 but u is adjacent to p_3.

Then p_5p_6 and vup_3p_2 form an induced $P_2 + P_4$ in G unless u is adjacent to a vertex from $\{p_5, p_6\}$. Suppose u is adjacent to p_5. Then u is not adjacent to p_6, and we find that p_1p_2 and vup_5p_6 form an induced $P_2 + P_4$ in G. Suppose u is adjacent to p_6. Then u is not adjacent to p_5, and we find that p_1p_2 and vup_6p_5 form an induced $P_2 + P_4$ in G. Both cases are not possible.

Because the remaining cases follow from symmetry, we conclude that $u \in T_1 \cup T_2 \cup T_3$. Hence $T^* \subseteq T_1 \cup T_2 \cup T_3$, and we have proven Claim 1.

Claim 2. *Either $T_1 \cup T_2$ or T_3 is empty.*

We prove Claim 2 as follows. Assume $T_1 \cup T_2 \neq \emptyset$ and $T_3 \neq \emptyset$. Without loss of generality, assume there is a vertex $u \in T_1$ and a vertex $v \in T_3$. By definition, u is adjacent to p_1, p_3 and p_5. Since u has a list of 2 admissible colours, p_1, p_3 and p_5 are coloured by the same colour, say colour 1. Because p_2 is adjacent to p_1, vertices p_1 and p_2 have different

colours. Thus the colours of p_2 and p_5 are different. Then v has only one admissible colour in its list. This contradiction proves Claim 2.

Using Claim 2 we distinguish two cases.

Case 1. $T_1 \cup T_2$ is empty and T_3 is not empty.

Since every vertex in T_3 has a list of 2 admissible colours, p_2 and p_5 are coloured the same. Recall that S is an independent set. Hence we can safely colour all the vertices in S by the same colour as p_2 and p_5. We are left to apply Theorem 1.1.1.

Case 2. T_3 is empty and $T_1 \cup T_2$ is not empty.

If one of T_1 and T_2 is empty, say $T_2 = \emptyset$, we proceed as in Case 1. We now assume that none of T_1 and T_2 is empty. As before, this means that p_1, p_3, p_5 must have the same colour, say colour 1, whereas p_2, p_4, p_6 also have the same colour, say colour 2. Recall that S is an independent set. Hence, we can safely colour all vertices of S that only have neighbours in T_1 by colour 1, and all vertices of S that only have neighbours in T_2 by colour 2. Afterwards we remove them from G. If no vertices of S remain we apply Theorem 1.1.1. Suppose S did not become empty. Then each (remaining) vertex of S has a neighbour in T_1 and T_2. We first try the case that all vertices of T_1 receive colour 2. For this colouring of T_1, all vertices in S get reduced lists of size at most 2, so we can again apply Theorem 1.1.1.

We are left to consider the possibility that colour 3 is used on at least one vertex of T_1. We try all possible $|T_1| = O(|V(G)|)$ choices in which we give one fixed vertex $x \in T_1$ colour 3. Below we describe what we do for each such choice.

We first update G. If G then only contains vertices that have a list of admissible colours of size 2, we apply Theorem 1.1.1. Otherwise, we restore x and all vertices of H back into G and redefine sets T_1, T_2 and S accordingly. We find that no vertex in T_2 is adjacent to x, because such vertex would have received colour 1 and would have been removed when we were updating G. Furthermore, by definition of S, no vertex in S is adjacent to x, and we may again assume that each vertex in S is adjacent to a vertex in T_1 and to a vertex in T_2.

Let y be an arbitrary vertex of T_2. Suppose there exists an edge ab such that $a \in T_2$, $b \in S$ and y is not adjacent to a, b. Then G contains an induced $P_2 + P_4$ formed by xp_1 and bap_6y. This is not possible. Hence, the vertex y is adjacent to at least one of the vertices of every edge ab with $a \in T_2$ and $b \in S$. We consider both the case in which

y gets colour 1 and in which y gets colour 3. Then, in each case, we reduce the list of admissible colours of each vertex $s \in S$ by at least one, which can be seen as follows. Suppose y gets colour $i \in \{1,3\}$. If s is adjacent to y, then s cannot receive colour i. Otherwise, as we just proved, s is adjacent to a neighbour of y in T_2. Because this neighbour is in T_2, it is adjacent to p_6 as well, and consequently it gets colour $j = 1$ if $i = 3$ and colour $j = 3$ if $i = 1$. This means that s cannot receive colour j. Hence, after updating the graph we may apply Theorem 1.1.1. This finishes Case 2, and thus the description of our algorithm is completed. □

Corollary 4.3.2. *The* Colouring *problem can be solved in polynomial time for the class of triangle-free $(P_2 + P_4)$-free graphs.*

Proof. We first note that a graph can be coloured with at most one colour if and only if it consists of isolated vertices only. Secondly, a graph can be coloured with at most two colours if and only if it is bipartite. Both cases can clearly be checked in polynomial time. By Theorem 4.3.1, we can also check in polynomial time whether a triangle-free (P_2+P_4)-free graph is 3-colourable. We complete the proof by showing that every triangle-free $(P_2 + P_4)$-free graph can always be coloured with at most 4 colours. This means that such a graph has chromatic number 4 if it is not 3-colourable.

Let G be a triangle-free $(P_2 + P_4)$-free graph. We may assume that G is connected and that $|V(G)| \geq 5$. Consider an induced P_2 of G, say with vertices v_1 and v_2. Let G' be the graph of G obtained from G after removing v_1, v_2 and all their neighbours. Then G' contains no triangle and no induced P_4, because G is triangle-free and $(P_2 + P_4)$-free. This implies that G' is bipartite. Because G is triangle-free, the set of neighbours of v_1 is an independent set and the set of neighbours of v_2 is an independent set.

Due to the above we can define the following 4-colouring of G. Assign colour 1 to all neighbours of v_1 (including v_2) and colour 2 to all neighbours of v_2 (including v_1). Assign colours 3 and 4 to all the vertices of G' corresponding to the bipartition of G'. This completes the proof of Corollary 4.3.2. □

4.4 List 3-Colouring for sP_3-free Graphs

In this section we show how to test in polynomial time whether a given sP_3-free graph admits a colouring that respects some given 3-list assignment. Our polynomial-time

algorithm heavily relies on a number of structural properties of 3-colourable sP_3-free graphs that we have shown in Section 2.1.

Theorem 4.4.1. *The* List 3-Colouring *problem can be solved in polynomial time for the class of sP_3-free graphs for any fixed integer $s \geq 1$.*

Proof. Let $G = (V, E)$ be a graph with a k-list assignment $\mathcal{L}$ for some integer $k \geq 1$. Let $W \subseteq V$ be the set of those vertices of G whose lists have size at most 2. By Proposition 1.2.2 we may assume that every vertex in $V \setminus W$ has degree at least 3. Recall that $R(s, 4)$ is the smallest number of vertices n such that all graphs on n vertices contain an independent set of size s or a clique of size 4. By using brute force, we search for a set D of size at most $3R(s, 4)+12(s-1)$ that dominates $V \setminus W$. This takes $O(|V|^{3R(s,4)+12(s-1)})$ time, which is polynomial because s is fixed. If such a set does not exist, then G has no colouring respecting $\mathcal{L}$ due to Lemma 2.1.3. If we find a dominating set D of size at most $3R(s, 4) + 12(s - 1)$, then we guess a possible 3-colouring of D that respects the restriction of $\mathcal{L}$ to D, and we apply Theorem 1.1.1. Because $|D| \leq 3R(s, 4) + 12(s - 1)$, the total number of these guesses is $3^{|D|} \leq 3^{3R(s,4)+12(s-1)}$. This number is constant because s is fixed. Hence, our algorithm runs in polynomial time. □

4.5 List 4-Colouring for P_6-free Graphs

We prove that List 4-Colouring is NP-complete for P_6-free graphs. We use a reduction from the Not-All-Equal 3-Satisfiability problem with un-negated literals; recall that this is an NP-complete problem [88]. We consider an arbitrary instance I of Not-All-Equal 3-Satisfiability with variables $x_1, x_2, \ldots, x_n$ and 3-literal clauses $C_1, C_2, \ldots, C_m$ that all contain un-negated literals only. From I we construct a graph G_I with a 4-list assignment $\mathcal{L}$. Next we show that G_I is P_6-free and that G_I has a colouring that respects $\mathcal{L}$ if and only if I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal.

To obtain the graph G_I with its 4-list assignment $\mathcal{L}$ we modify the construction of the (P_7-free but not P_6-free) graph used to prove Theorem 3.3.2 which states that 4-Precolouring Extension is NP-complete for P_7-free graphs. We do this as follows.

- For each clause C_j, we introduce five vertices $a_{j,1}, b_{j,1}, a_{j,2}, b_{j,2}, a_{j,3}$ that have lists of admissible colours $\{2, 4\}, \{3, 4\}, \{2, 3, 4\}, \{3, 4\}, \{2, 3\}$, respectively, and we add the edges $a_{j,1}b_{j,1}, b_{j,1}a_{j,2}, a_{j,2}b_{j,2}, b_{j,2}a_{j,3}$.

We take a copy C'_h for each clause C_h.

For each copy C'_h, we introduce five vertices $a'_{j,1}, b'_{j,1}, a'_{j,2}, b'_{j,2}, a'_{j,3}$ that have lists of admissible colours $\{1,4\}, \{3,4\}, \{1,3,4\}, \{3,4\}, \{1,3\}$, respectively, and we add the edges $a'_{j,1}b'_{j,1}, b'_{j,1}a'_{j,2}, a'_{j,2}b'_{j,2}, b'_{j,2}a'_{j,3}$.

We say that all these vertices (so including the vertices in the copy) are of a-type and b-type, respectively. They induce a disjoint union of $2m$ P_5s in G_I, which we call *clause-components*.

- We represent every variable x_i by a vertex, which we also denote by x_i and which we give a list of admissible colours $L(x_i) = \{1,2\}$ in G_I. We say that these vertices are of x-type.

- For every clause C_j, we fix an arbitrary order of its variables $x_{i_1}, x_{i_2}, x_{i_3}$ and add edges $a_{j,h}x_{i_h}$ and $a'_{j,h}x_{i_h}$ for $h = 1,2,3$.

- We add an edge between every x-type vertex and every b-type vertex.

In Figure 4.2 we illustrate an example in which C_j is a clause with ordered variables $x_{i_1}, x_{i_2}, x_{i_3}$. The thick edges indicate the connection between these vertices and the a-type vertices of the two copies of the clause gadget. We omitted the indices from the labels of the clause gadget vertices to increase the visibility.

We now prove two lemmas. Lemma 4.5.1 shows that the graph G_I is P_6-free. In Lemma 4.5.2 we prove that G_I has a colouring that respects $\mathcal{L}$ if and only if I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal.

Lemma 4.5.1. *The graph G_I is P_6-free.*

Proof. Let P be an induced path in G_I. We show that P has at most five vertices. We distinguish the following cases.

Case 1. P contains no x-type vertex.
This means that P is contained in one clause-component, which is isomorphic to an induced P_5. Consequently, P has at most five vertices.

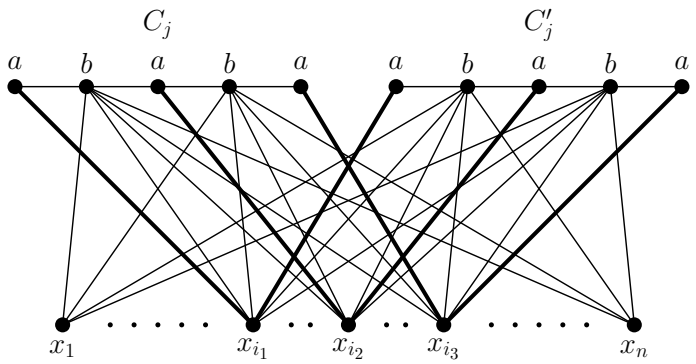

Figure 4.2: The graph G_I for the clause $C_j = \{x_{i_1}, x_{i_2}, x_{i_3}\}$.

Case 2. P contains exactly one x-type vertex.
Let x_i be this vertex. Then P contains vertices of at most two clause-components. Since x_i is adjacent to all b-type vertices, we then find that P contains at most two vertices of each of the clause-components. Hence P has at most five vertices.

Case 3. P contains exactly two x-type vertices.
Let x_h and x_i be these two vertices. If P contains no b-type vertex, then there is no subpath in P from x_h to x_i, a contradiction. If P contains two or more b-type vertices, then P contains a cycle, another contradiction. Hence P contains exactly one b-type vertex z. Then $x_h z x_i$ is a subpath in P. If x_h has a neighbour in $V(P)\backslash\{z\}$, then this neighbour must be of a-type, and consequently, an end-vertex of P (because an a-type vertex is adjacent to only one x-type vertex). The same holds for x_i. Hence P contains at most five vertices.

Case 4. P contains at least three x-type vertices.
Then P contains no b-type vertex, because such vertices would have degree 3 in P. However, then there is no subpath between any two x-type vertices in P. We conclude that this subcase is not possible. This completes the proof of Lemma 4.5.1. □

Lemma 4.5.2. *The graph G_I has a colouring that respects $\mathcal{L}$ if and only if I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal.*

Proof. First suppose that G_I has a colouring that respects $\mathcal{L}$. Consider a clause C_j with literals ordered as x_{i_1}, x_{i_2} and x_{i_3}. Suppose that x_{i_1}, x_{i_2} and x_{i_3} all have colour 2.

Since the list of $a_{j,1}$ is $\{2,4\}$, we find that $a_{j,1}$ must receive colour 4. Consequently, its neighbour $b_{j,1}$ must have colour 3. Similarly, $a_{j,3}$ must have colour 3 and $b_{j,2}$ must have colour 4. This means that $a_{j,2}$ has neighbours, namely $x_{i_2}, b_{j,1}, b_{j,2}$, with colours $2,3,4$, respectively. However, $L(a_{j,2}) = \{2,3,4\}$. Hence, this is not possible. We conclude that at least one literal in every clause is coloured with colour 1. By considering the copy gadgets, we find in a similar way that at least one literal in every clause is coloured with colour 2. This means that the truth assignment that sets a literal to `FALSE` if the corresponding x-type vertex has colour 2, and to `TRUE` otherwise, is a satisfying truth assignment of I in which each clause contains at least one true and at least one false literal.

Now suppose that I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal. We use colour 1 to colour the x-type vertices representing the true literals and colour 2 to colour the x-type vertices representing the false literals. Since each clause contains at least one true literal, we can colour $a_{j,1}$, $a_{j,2}$ and $a_{j,3}$ respecting their lists. Similarly, since each clause contains at least one false literal, we can colour $a'_{j,1}$, $a'_{j,2}$ and $a'_{j,3}$ respecting their lists. We colour all other remaining uncoloured vertices in a straightforward way. This completes the proof of Lemma 4.5.2. □

The observation that List 4-Colouring belongs to NP, together with Lemmas 4.5.1 and 4.5.2, immediately gives us the main result of this section.

Theorem 4.5.3. *The* List 4-Colouring *problem is* NP*-complete for* P_6*-free graphs.*

4.6 List k-Colouring for $(K_{s,t}, P_r)$-free Graphs

We prove the following theorem.

Theorem 4.6.1. *For all integers* $k, r, s, t \geq 1$*, the* List k-Colouring *problem can be solved in polynomial time on* $(K_{s,t}, P_r)$*-free graphs.*

Proof. Atminas, Lozin and Razgon [2] showed that for any two integers r and w, there exists an integer $b(r,w)$ such that any graph of treewidth at least $b(r,w)$ contains the path P_r as an induced subgraph or the complete bipartite graph $K_{w,w}$ as a (not necessarily induced) subgraph. We will combine this result and Ramsey's Theorem in the following way. Let $k, r, s, t \geq 1$, and let G be a $(K_{s,t}, P_r)$-free graph with

some k-list assignment $\mathcal{L}$. We use a as a short-hand notation for the Ramsey number $R(\max\{k+1,s,t\},\max\{k+1,s,t\})$. Using Bodlaender's algorithm [8] we can test in linear time whether the treewidth of G is at most $b(r,a)-1$.

First suppose that the treewidth of G is at most $b(r,a)-1$. Jansen and Scheffler [59] showed that List k-Colouring can be solved in time $O(nk^{t+1})$ on an n-vertex graph with treewidth at most t that has a k-list assignment. Hence, because $b(r,a)-1$ is a constant, we can uses this result to test in polynomial time whether G has a colouring repecting $\mathcal{L}$.

Now suppose that the treewidth of G is at least $b(r,a)$. We claim that in this case G is not k-colourable. This can be seen as follows. Due to the aforementioned result of Atminas, Lozin and Razgon [2] and our assumption that G is P_r-free, we find that G contains the complete bipartite graph $K_{a,a}$ as a subgraph. Let S_1 and S_2 be the partition classes of this complete bipartite graph. By Ramsey's Theorem, we find that both S_1 and S_2 either contain an independent set of size $\max\{k+1,s,t\}$ or a clique of size $\max\{k+1,s,t\}$. If both of them contain an independent set of size $\max\{k+1,s,t\}$, then $G[S_1 \cup S_2]$, and consequently, G contains an induced $K_{s,t}$, which is not possible. Hence, at least one of them, say S_1, contains a clique of size $\max\{k+1,s,t\}$. Then $G[S_1]$ is not k-colourable. Consequently, G is not k-colourable. Since Bodlaender's algorithm performs in linear time, List k-Colouring can be solved in time $O(nk^{t+1})$ on an n-vertex graph with treewidth at most t that has a k-list assignment. □

Bibliography

[1] K. Appel and W. Haken. Every planar map is four colorable. Contemporary mathematics 98, *Amererican Mathematical Society*, 1989.

[2] A. Atminas, V.V. Lozin. I. Razgon. Linear time algorithm for computing a small biclique in graphs without long induced path. In: *Proceedings of 13th Scandinavian Symposium and Workshops (SWAT 2012)*, volume 7357 of *Lecture Notes in Computer Science*, pages 142–152, 2012.

[3] E. Balas and C.S. Yu. On graphs with polynomially solvable maximum-weight clique problem. Networks, 19:247–253, 1989.

[4] H.J. Bandelt, H.M. Mulder. Distance–hereditary graphs. *Journal of Combinatorial Theory, Series B*, 41:182–208, 1986.

[5] L.W. Beineke. Characterization of derived graphs. *Journal of Combinatorial Theory, Series B*, 9:129–135, 1970.

[6] A Björklund, T Husfeldt and M Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39:546–563, 2009.

[7] J. R. S. Blair and B. W. Peyton. An introduction to chordal graphs and clique trees. *Graph Theory and Sparse Matrix Computation* (Ed. A. George, J.R. Gilbert, and J.W.H. Liu), 56:1–29, 1993.

[8] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[9] H.L. Bodlaender. Classes of graphs with bounded treewidth. Technical Report RUU-CS-86-22, Department of Computer Science, Utrecht University, 1986.

[10] F. Bonomo, G. Durán and J. Marenco. Exploring the complexity boundary between coloring and list-coloring. *Annals of Operations Research*, 169:3–16, 2009.

[11] A. Brandstädt, J. Engelfriet, H.-O. Le, and V.V. Lozin. Clique-Width for 4-Vertex Forbidden Subgraphs. *Theory of Computing Systems*, 39:561–590, 2006.

[12] A. Brandstädt and D. Kratsch. On the structure of (P_5, gem)-free graphs. *Discrete Applied Mathematics*, 145:155–166, 2005.

[13] A. Brandstädt, H.-O. Le, R. Mosca. Gem- and co-gem-free graphs have bounded clique-width. *International Journal of Foundations of Computer Science*, 15:163–185, 2004.

[14] A. Brandstädt, V.B. Le and J. Spinrad. Graph Classes: A Survey. *SIAM Monographs on Discrete Mathematics and Applications*, volume 3, 1999.

[15] G. Brinkmann and M. Meringer. The Smallest 4-Regular 4-Chromatic Graphs with Girth 5. Graph Theory Notes of New York 32, 40–41, 1997.

[16] H.J. Broersma, F.V. Fomin, P.A. Golovach and D. Paulusma. Three complexity results on coloring P_k-free graphs. *European Journal of Combinatorics*, to appear.

[17] H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song. Determining the chromatic number of triangle-free $2P_3$-free graphs in polynomial time. *Theoretical Computer Science*, 423:1–10, 2012.

[18] H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song. Narrowing down the gap on the complexity of coloring Pk-free graphs. In: *Proceedings of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2010)*, volume 6410 of *Lecture Notes in Computer Science*, pages 63–74, 2010.

[19] H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song. Updating the complexity status of coloring graphs without a fixed induced linear forest. *Theoretical Computer Science*, 414:9–19, 2012.

[20] M. Chlebík and J. Chlebíková. Hard coloring problems in low degree planar bipartite graphs. *Discrete Applied Mathematics*, 154:1960–1965, 2006.

[21] V. Chvátal and P.L. Hammer. Set–packing and threshold graphs. Research Report, Computer Science Department, University of Waterloo, Ontario CORR 73-21, 1973.

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

[23] C.J. Colbourn. The complexity of completing partial Latin squares. *Annals of Discrete Mathematics*, 8:25–30, 1984.

[24] D.G. Corneil, S. Olariu and L. Stewart. Asteroidal Triple-Free Graphs. *SIAM Journal on Discrete Math*, 10:399–430, 1997.

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

[26] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101:77–144, 2000.

[27] J.F. Couturier, P.A. Golovach, D. Kratsch and D. Paulusma. List coloring in the absence of a linear forest. In: *Proceedings of the 37th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2011)*, volume 6986 of *Lecture Notes in Computer Science*, pages 119–130, 2011.

[28] K. Dabrowski, V. Lozin, R. Raman and B. Ries. Colouring vertices of triangle-free graphs without forests. *Discrete Mathematics*, 312:1372–1385, 2012.

[29] R. Diestel. Graph Theory. *Springer-Verlag*, Electronic Edition, 2005.

[30] B. Dushnik and E.W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63:600–610, 1941.

[31] Z. Dvořák, K. Kawarabayashi and R. Thomas. Three-coloring triangle-free planar graphs in linear time. In: *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009)*, pages 1176–1182, Society for Industrial and Applied Mathematics, 2009.

[32] P. Erdős, A. L. Rubin, and H. Taylor. Choosability in graphs. In: *Proceedings of the West Coast Conference on Combinatorics, Graph Theory and Computing*, Congressus Numerantium XXVI, pages 125–157, 1980.

[33] S. Földes, P.L. Hammer. Split graphs. In: *Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory and Computing*, Congressus Numerantium XIX, 311–315, 1977.

[34] T. Gallai. Transitiv orientierbare Graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, 18:25–66, 1967.

[35] M.R. Garey, D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, 1979.

[36] M.R. Garey, D.S. Johnson, and L.J. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237-267, 1976.

[37] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16:47–56, 1974.

[38] P.A. Golovach and P. Heggernes. Choosability of P_5-free graphs. In: *Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science (MFCS 2009)*, volume 5734 of *Lecture Notes in Computer Science*, pages 382–391, 2009.

[39] P.A. Golovach and D. Paulusma. List Coloring in the Absence of Two Subgraphs. In: *Proceedings of the 8th International Conference on Algorithms and Complexity (CIAC 2013)*, to appear.

[40] P.A. Golovach, D. Paulusma and B. Ries. Coloring Graphs Characterized by a Forbidden Subgraph. In: *Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science (MFCS 2012)*, volume 7464 of *Lecture Notes in Computer Science*, 443–454, 2012.

[41] P.A. Golovach, D. Paulusma and J. Song. 4-Coloring H-free graphs when H is small. *Discrete Applied Mathematics*, 161:140–150, 2013.

[42] P.A. Golovach, D. Paulusma and J. Song. 4-Coloring H-free graphs when H Is small. In: *Proceedings of the 38th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2012)*, volume 7147 of *Lecture Notes in Computer Science*, 289–300, 2012.

[43] P.A. Golovach, D. Paulusma and J. Song. Closing complexity gaps for coloring problems on H-free graphs. In: *Proceedings of the 23nd International Symposium on Algorithms and Computation (ISAAC 2012)*, *Lecture Notes in Computer Science*, to appear.

[44] P.A. Golovach, D. Paulusma and J. Song. Coloring graphs without short cycles and long induced paths. In: *Proceedings of the 18th International Symposium on Fundamentals of Computation Theory (FCT 2011)*, volume 6914 of *Lecture Notes in Computer Science*, pages 193–204, 2011.

[45] M.C. Golumbic. Algorithmic Graph Theory and Perfect Graphs. Second edition, *Annals of Discrete Mathematics* 57, North Holland, 2004.

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

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

[48] H. Grötzsch. Zur Theorie der diskreten Gebilde. VII. Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel, Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg. Math.-Nat. Reihe 8, 109–120, 1958.

[49] A. Gyárfás. Problems from the world surrounding perfect graphs. *Zastosowania Matematyki Applicationes Mathematicae* XIX, 19:413–441, 1987.

[50] P.J. Heawood. Map-Colour Theorem, *Quarterly Journal of Mathematics*, Oxford University Press, 24:332–338, 1890.

[51] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.

[52] P. Hell and J. Nešetřil. Graphs and Homomorphisms. *Oxford Lecture Series in Mathematics and Its Applications* 28, Oxford University Press, 2004.

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

[54] I. Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10:718–720, 1981.

[55] J.E. Hopcroft and R.M. Karp. An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2:225–231,1973.

[56] M. Hujter and Zs. Tuza. Precoloring extension. II. Graphs classes related to bipartite graphs. *Acta Mathematica Universitatis Comenianae*, 6:1-11, 1993.

[57] M. Hujter and Z. Tuza. Precoloring extension. III. Classes of perfect graphs. *Combinatorics, Probability and Computing*, 5:35–56, 1996.

[58] K. Jansen. Complexity Results for the Optimum Cost Chromatic Partition Problem. Universität Trier, Mathematik/Informatik, Forschungsbericht 96–41, 1996.

[59] K Jansen and P. Scheffler. Generalized coloring for tree-like graphs. *Discrete Applied Mathematics*, 75:135–155, 1997.

[60] T. R. Jensen and B. Toft. Graph Coloring Problems. Wiley Press, 1995.

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

[62] M. Kamiński and V.V. Lozin. Vertex 3-colorability of Claw-free Graphs. *Algorithmic Operations Research*, 2:15–21, 2007.

[63] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, Plenum Press, pages 85–103, 1972.

[64] D. Kobler and U. Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Applied Mathematics* 126:197–221, 2003.

[65] M. Kochol, V.V. Lozin, Bert Randerath. The 3-colorability problem on graphs with maximum degree four. *SIAM Journal on Computing*, 32:1128–1139, 2003.

[66] E.G. Köhler. Graphs without asteroidal triples. Ph.D. Thesis, Technische Universit ät Berlin, Cuvillier Verlag Göttingen, 1999.

[67] D. König. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen*, 77:453–465, 1916.

[68] L. Kowalik. Fast 3-coloring Triangle-Free Planar Graphs. *Algorithmica*, 58:770–789, 2010.

[69] D. Král', J. Kratochvíl, Zs. Tuza, and G.J. Woeginger. Complexity of coloring graphs without forbidden induced subgraphs. In: *Proceedings of the 27th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2001)*, volume 2204 of *Lecture Notes in Computer Science*, pages 254-262, 2001.

[70] J. Kratochvíl. Precoloring extension with fixed color bound. *Acta Mathematica Universitatis Comenianae*, 62:139–153, 1993.

[71] D. Kratsch and H. Müller. Colouring AT-free graphs. In: *Proceedings of the 20th Annual European Symposium on Algorithms (ESA 2012)*, volume 7501 of L*ecture Notes in Computer Science*, pages 707–718, 2012.

[72] J. Kratochvíl and Z. Tuza. Algorithmic complexity of list colorings. *Discrete Applied Mathematics*, 50:297–302, 1994.

[73] M. Kubale. Some results concerning the complexity of restricted colorings of graphs. *Discrete Applied Mathematics* 36:35–46, 1992.

[74] C. Kuratowski. Sur le probleme des corbes gauches en topologie. *Fundamenta Mathematicae*, 15:271–283, 1930.

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

[76] C. Lekkerkerker and D. Boland. Representation of finite graphs by a set of intervals on the real line. *Fundamenta Mathematicae*, 51:45–64, 1962.

[77] D. Leven and Z. Galil. NP completeness of finding the chromatic index of regular graphs. *Journal of Algorithms*, 4:35–44, 1983.

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

[79] D. Marx. Precoloring extension on unit interval graphs. *Discrete Applied Mathematics*, 154:995–1002, 2006.

[80] G.B. Mertzios and P.G. Spirakis. Algorithms and almost tight results for 3-colorability of small diameter graphs. In: *Proceedings of the 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2013)*, pages 332-343, 2013.

[81] A.V. Pyatkin. Triangle-free $2P_3$-free graphs are 4-colorable. *Discrete Mathematics*, to appear.

[82] B. Randerath. 3-colorability and forbidden subgraphs. I., Characterizing pairs, *Discrete Mathematics*, 276:313–325, 2004.

[83] B. Randerath and I. Schiermeyer. 3-Colorability $\in$ P for P_6-free graphs. *Discrete Applied Mathematics*, 136:299–313, 2004.

[84] B. Randerath and I. Schiermeyer. A note on Brooks theorem for triangle-free graphs. *Australasian Journal of Combinatorics* 26:3–9, 2002.

[85] B. Randerath and I. Schiermeyer. Vertex colouring and forbidden subgraphs - a survey. *Graphs and Combinatorics* 20:1–40, 2004.

[86] M. Rao. MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theoretical Computer Science*, 377:260–267, 2007.

[87] F. S. Roberts. Indifference Graphs. In: *Proof Techniques in Graph Theory*, Academic Press, pages 139–146, 1969.

[88] T. J. Schaefer. The complexity of satisfiability problems. In: *Proceedings of the tenth annual ACM symposium on Theory of Computing*, pages 216–226, 1978.

[89] D. Schindl. Some new hereditary classes where graph coloring remains NP-hard. *Discrete Mathematics*, 295:197–202, 2005.

[90] J. Stacho. 3-Colouring AT-free graphs in polynomial time. *Algorithmica*, 64:384–399, 2012.

[91] D.P. Sumner. Subtrees of a graph and the chromatic number. In: *The Theory and Applications of Graphs (Ed. G. Chartrand), Proceedings of 4th International Graph Theory Conference*, pages 557–576, Wiley Press, 1980.

[92] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6:505–517, 1977.

[93] Zs. Tuza. Graph colorings with local restrictions - a survey. *Discussiones Mathematicae*, Graph Theory, 17:161–228, 1997.

[94] V.G. Vizing. Coloring the vertices of a graph in prescribed colors. in Diskret. Analiz., no. 29, Metody Diskret. Anal. v. Teorii Kodov i Shem 101:3–10, 1976.

[95] V.G. Vizing. On an estimate of the chromatic class of a p-graph. Diskret. Analiz, 3:25–30, 1964.

[96] G. Wegner. Eigenschaften der Nerven homologisch–einfacher Familien im R^n. Dissertation Thesis, Universität Göttingen, 1967.

[97] G.J. Woeginger and J. Sgall. The complexity of coloring graphs without long induced paths. *Acta Cybernetica*, 15:107–117, 2001.

[98] E.S. Wolk. The comparability graph of a tree. *American Mathematical Society*, 13:789–795, 1962.

Index