

University of Warwick institutional repository: <http://go.warwick.ac.uk/wrap>

A Thesis Submitted for the Degree of PhD at the University of Warwick

<http://go.warwick.ac.uk/wrap/60496>

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it. Our policy information is available from the repository home page.

Library Declaration and Deposit Agreement

1. STUDENT DETAILS

Please complete the following:

Full name:CHRISTOPHER PURCELL.....

University ID number:1057802.....

2. THESIS DEPOSIT

2.1 I understand that under my registration at the University, I am required to deposit my thesis with the University in BOTH hard copy and in digital format. The digital version should normally be saved as a single pdf file.

2.2 The hard copy will be housed in the University Library. The digital version will be deposited in the University's Institutional Repository (WRAP). Unless otherwise indicated (see 2.3 below) this will be made openly accessible on the Internet and will be supplied to the British Library to be made available online via its Electronic Theses Online Service (EThOS) service.

[At present, theses submitted for a Master's degree by Research (MA, MSc, LLM, MS or MMedSci) are not being deposited in WRAP and not being made available via EthOS. This may change in future.]

2.3 In exceptional circumstances, the Chair of the Board of Graduate Studies may grant permission for an embargo to be placed on public access to the hard copy thesis for a limited period. It is also possible to apply separately for an embargo on the digital version. (Further information is available in the *Guide to Examinations for Higher Degrees by Research*.)

2.4 If you are depositing a thesis for a Master's degree by Research, please complete section (a) below. For all other research degrees, please complete both sections (a) and (b) below:

(a) Hard Copy

I hereby deposit a hard copy of my thesis in the University Library to be made publicly available to readers (please delete as appropriate) EITHER immediately OR after an embargo period of months/years as agreed by the Chair of the Board of Graduate Studies.

I agree that my thesis may be photocopied. YES / NO (Please delete as appropriate)

(b) Digital Copy

I hereby deposit a digital copy of my thesis to be held in WRAP and made available via EThOS.

Please choose one of the following options:

EITHER My thesis can be made publicly available online. YES / NO (Please delete as appropriate)

OR My thesis can be made publicly available only after.....[date] (Please give date)
YES / NO (Please delete as appropriate)

OR My full thesis cannot be made publicly available online but I am submitting a separately identified additional, abridged version that can be made available online.
YES / NO (Please delete as appropriate)

OR My thesis cannot be made publicly available online. YES / NO (Please delete as appropriate)

3. GRANTING OF NON-EXCLUSIVE RIGHTS

Whether I deposit my Work personally or through an assistant or other agent, I agree to the following:

Rights granted to the University of Warwick and the British Library and the user of the thesis through this agreement are non-exclusive. I retain all rights in the thesis in its present version or future versions. I agree that the institutional repository administrators and the British Library or their agents may, without changing content, digitise and migrate the thesis to any medium or format for the purpose of future preservation and accessibility.

4. DECLARATIONS

(a) I DECLARE THAT:

- I am the author and owner of the copyright in the thesis and/or I have the authority of the authors and owners of the copyright in the thesis to make this agreement. Reproduction of any part of this thesis for teaching or in academic or other forms of publication is subject to the normal limitations on the use of copyrighted materials and to the proper and full acknowledgement of its source.
- The digital version of the thesis I am supplying is the same version as the final, hard-bound copy submitted in completion of my degree, once any minor corrections have been completed.
- I have exercised reasonable care to ensure that the thesis is original, and does not to the best of my knowledge break any UK law or other Intellectual Property Right, or contain any confidential material.
- I understand that, through the medium of the Internet, files will be available to automated agents, and may be searched and copied by, for example, text mining and plagiarism detection software.

(b) IF I HAVE AGREED (in Section 2 above) TO MAKE MY THESIS PUBLICLY AVAILABLE DIGITALLY, I ALSO DECLARE THAT:

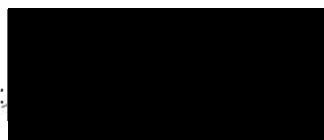
- I grant the University of Warwick and the British Library a licence to make available on the Internet the thesis in digitised format through the Institutional Repository and through the British Library via the EThOS service.
- If my thesis does include any substantial subsidiary material owned by third-party copyright holders, I have sought and obtained permission to include it in any version of my thesis available in digital format and that this permission encompasses the rights that I have granted to the University of Warwick and to the British Library.

5. LEGAL INFRINGEMENTS

I understand that neither the University of Warwick nor the British Library have any obligation to take legal action on behalf of myself, or other rights holders, in the event of infringement of intellectual property rights, breach of contract or of any other right, in the thesis.

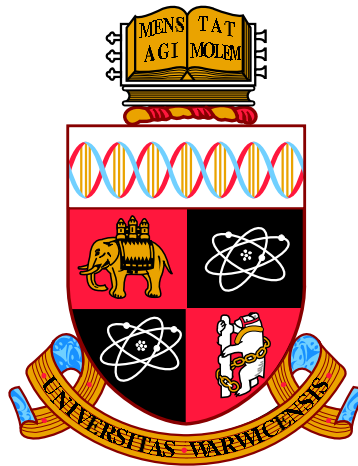
Please sign this agreement and return it to the Graduate School Office when you submit your thesis.

Student's signature:



Date:

26/2/2014



**Cliques, Colouring and Satisfiability:
from structure to algorithms**

by

Christopher Purcell

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Mathematics Institute

October 2013

THE UNIVERSITY OF
WARWICK

Contents

List of Figures	iv
Acknowledgments	v
Declarations	vii
Abstract	viii
Chapter 1 Introduction	1
1.1 Definitions and Notation	4
I Colouring and Related Problems	8
Chapter 2 Vertex 3-Colourability in Claw-free Graphs	11
2.1 Introduction	11
2.2 NP-completeness results	14
2.3 Polynomial-time results	18
2.3.1 Infinitely many forbidden induced subgraphs	18
2.3.2 Finitely many forbidden induced subgraphs	19
2.4 (claw, Φ_k, Φ_{k+1})-free graphs	28
2.5 Conclusion	30
Chapter 3 Dominating Induced Matchings in graphs without a skew star	31
3.1 Introduction	31
3.2 Preliminaries	33
3.3 Solution to the problem in the class of $S_{1,2,2}$ -free graphs	36
3.3.1 Reduction to graphs of bounded vertex degree	36
3.3.2 Reduction to graphs of bounded chordality	37

3.4	Solution to the problem in the class of $S_{1,2,3}$ -free graphs	40
3.5	Conclusion	52

II Independent Sets and Related Problems 53

Chapter 4 The Maximum Weight Independent Set Problem in Graphs

Without Large Apples		56
4.1	Introduction	56
4.2	A Sufficient Condition for Claw-Freeness of (A_k, A_{k+1}, \dots) -free Atoms	59
4.3	Polynomial Results	66
4.3.1	Graphs of Bounded Vertex Degree	67
4.3.2	Apex-Minor-Free Graphs	67
4.4	Conclusion	71

Chapter 5 Sparse Regular Induced Subgraphs in $2P_3$ -free Graphs 73

5.1	Introduction	73
5.2	Finding maximum k -regular graphs containing a P_3	74
5.3	Finding maximum P_3 -free k -regular graphs	75
5.3.1	Step 1: generation of the family S	76
5.3.2	Step 2: solving the problem	80
5.4	Conclusion	81

Chapter 6 Independent Domination in Finitely Defined Classes of Graphs 82

6.1	Introduction	82
6.2	Preliminaries	83
6.3	Generation of maximal independent sets	86
6.3.1	Independent domination in $P_2 + P_3$ -free graphs	87
6.3.2	Weighted independent domination in $(P_5, 2P_3)$ -free graphs . .	92
6.3.3	More subclasses P_5 -free graphs	96
6.4	Modular Decomposition	97
6.5	Decreasing graphs	104
6.6	Conclusion	106

III Satisfiability and Related Problems 108

Chapter 7 Boundary Properties of the Satisfiability Problem 109

7.1	Introduction	109
7.2	Hereditary, limit and boundary properties of graphs	112
7.3	A limit property of satisfiability	113
7.4	Minimality of the limit property	114
7.5	Is \mathcal{S} a unique boundary property?	116
7.6	Conclusion	117

List of Figures

2.1	The graphs $T_{i,j,k}^1$ (left) and Φ_k (right)	12
2.2	The <i>diamond</i> , <i>gem</i> and <i>crystal</i> graphs, left to right.	13
2.3	Diamond implantation	13
2.4	Triangle implantation	14
2.5	Graphs Φ'_k , Φ''_k and $\Phi_{k,p}$ (from top to bottom)	15
2.6	The graph $T_{i,j,k}^\Delta$ (left) and the graph $T_{i,j,k}^{3\Delta}$	20
3.1	Graphs H_i (left) and $S_{i,j,k}$ (right)	32
3.2	A diamond (left) and a butterfly (right).	35
3.3	Two copies of an induced $S_{1,2,2}$ with their labellings	41
4.1	Two smallest apples A_4 and A_5	57
6.1	The graphs $S_{i,j,k}$ (a) and $T_{i,j,k}$ (b)	84
6.2	The graphs <i>bull</i> , <i>banner</i> and A (listed from left to right).	101
6.3	The graphs P_3^* (left) and P_4^* (right)	102
7.1	The formula graph of $(x_1 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee x_3 \vee x_4)$	110
7.2	Graphs H_n (left) and $S_{i,j,k}$ (right)	114

Acknowledgments

This thesis is dedicated to the memory of Tom Smith, whose delight in mathematics and fierce desire to understand the universe were and always will be a constant source of inspiration. Chapter 4, being concerned with the so-called *large-apple-free* graphs, would have been of particular interest to Tom, fond as he was both of the fruit and of idiosyncratically named mathematical objects. Without his encouragement I would never have embarked upon a Ph.D., and I am eternally grateful to him.

There are of course many people without whom this thesis would not have been possible, starting with my family. My parents, David and Su Purcell, have given me selfless, unconditional love and support. My sister Helen is my oldest friend and closest confidant, and I am so proud of her achievements. I am very lucky to have their three loves. My whole family deserve thanks for their part in my life. I couldn't possibly list them all, but I'd like to mention my grandmothers June Purcell and Angeles Pearson, and my godparents Jeremy Purcell and Miriam Pearson, for all they've done for me.

I am extremely lucky to have many wonderful friends, and again I can't possibly list all those who deserve my thanks here. However, I owe a great deal to Ellie Bagnall, Luke Turner, Steve Marchant, James Kay and Puck Rombach, who have each been an integral part of my support network. Hopefully they can say the same of me, and long may it continue.

At Royal Holloway, University of London, I received a great education and I thoroughly enjoyed my time there. I also owe a debt to St Mary's Catholic Secondary School in Bishop's Stortford, where I began my mathematical journey. I would like to thank all my mathematical educators, but particularly: Stefanie Gerke for turning

me into a combinatorialist; Mrs Parry and Miss Gallard for helping me fall in love with mathematics; Mrs Hogarth, who recognised that I might have been educatable in the first place.

Not quite last, and nowhere near least (and in many ways most), I thank my supervisor Vadim V. Lozin. His teaching, advice and research expertise have been excellent and his patience has been frankly superhuman. I also thank him for the research on which we collaborated and extend these thanks to all my co-authors.

Finally, my research and education at Warwick was supported by the Centre for Discrete Mathematics and its Applications (DIMAP) and I thank all its staff and students as well as the staff of the Mathematics Institute. I would especially like to thank Yvonne Carty and Carole Fisher, whose patience I no doubt tested. As a member of DIMAP, my research is partially funded by the EPSRC Science and Innovation Award EP/D063191/1.

Declarations

All the work in this thesis is original work that has not been submitted for a degree at another university.

Chapter 2 is based on [1], which is joint work with Vadim V. Lozin.

Chapter 3 is based on [2], which is joint work with Nicholas Korpelainen and Vadim V. Lozin. A proof of the results of the third section of this chapter first appeared in [3]; here we correct an error in the original proof and make some other improvements.

Chapter 4 is based on [4], which is joint work with Vadim V. Lozin and Martin Milanič.

Chapter 5 is based on [5], which is joint work with Vadim V. Lozin and Raffaele Mosca.

Chapter 6 is based on [6], which is joint work with Vadim V. Lozin and Raffaele Mosca.

Chapter 7 is based on [7], which is joint work with Vadim V. Lozin.

Abstract

We examine the implications of various structural restrictions on the computational complexity of three central problems of theoretical computer science (colourability, independent set and satisfiability), and their relatives. All problems we study are generally NP-hard and they remain NP-hard under various restrictions. Finding the greatest possible restrictions under which a problem is computationally difficult is important for a number of reasons. Firstly, this can make it easier to establish the NP-hardness of new problems by allowing easier transformations. Secondly, this can help clarify the boundary between tractable and intractable instances of the problem.

Typically an NP-hard graph problem admits an infinite sequence of narrowing families of graphs for which the problem remains NP-hard. We obtain a number of such results; each of these implies necessary conditions for polynomial-time solvability of the respective problem in restricted graph classes. We also identify a number of classes for which these conditions are sufficient and describe explicit algorithms that solve the problem in polynomial time in those classes. For the satisfiability problem we use the language of graph theory to discover the very first boundary property, i.e. a property that separates tractable and intractable instances of the problem. Whether this property is unique remains a big open problem.

Chapter 1

Introduction

In 1996, the American Mathematical Society published a volume of research papers originally presented as part of the Second DIMACS¹ Implementation Challenge. The Challenge began in September 1992, and culminated in a three-day workshop held at the DIMACS Center at Rutgers University in October, 1993. The volume contains 28 of the papers presented at the workshop under the common title “Cliques, coloring and satisfiability” [8]. These are three central problems of combinatorial optimization and theoretical computer science that are of fundamental importance from both a practical and theoretical point of view.

The practical importance of these problems is due to the fact that they find numerous applications across various fields. For instance, graph colouring is often used in scheduling algorithms. The problem of finding maximum cliques is at the heart of any clustering algorithm and finds applications in bioinformatics [9], computer vision and pattern recognition [10], while SAT-solvers are central tools in artificial intelligence, software testing [11], etc.

Developing our understanding of these problems is also of great theoretical importance. The universe of computational problems is a deep and beautiful one. The class of NP-complete problems, to which each of the titular problems belongs, is a vital part of our understanding of that universe. This class is sometimes thought of as being “at least as difficult” as any problem in NP. Indeed, any problem in NP may be reduced to an NP-complete problem, and thus knowledge about such problems may lead to an answer to the most famous problem in theoretical computer science, i.e. whether or not P and NP coincide. Each of the three main problems considered in this thesis remains difficult under substantial restrictions on the input instance.

¹DIMACS – The Center for Discrete Mathematics and Theoretical Computer Science – is a collaboration between Rutgers University, Princeton University, and the research firms AT&T, Bell Labs, Applied Communication Sciences, and NEC.

On the other hand, for each of them there are restrictions under which the problem becomes polynomial-time solvable. The goal of this thesis is to analyse how the structure of the input instances influences the complexity of solving the problems. In other words, the goal is to identify restrictions under which the problems either remain NP-complete or can be solved in polynomial time.

For the graph theory problems we study restrictions given in the form of forbidden induced subgraphs. For SATISFIABILITY, we develop a similar approach by associating a graph with each instance of the problem.

A graph properties (or class of graphs) admits a description in terms of forbidden induced subgraphs if and only if it is closed under vertex deletion. In other words, every induced subgraph of a graph in the class is also in the class. These properties are known as *hereditary* properties. The world of hereditary properties is rich and contains many classes of theoretical or practical importance, such as chordal, bipartite, planar, perfect, interval, comparability graphs, etc. All these classes have been defined in different terms, but all of them admit a uniform description in terms of forbidden induced subgraphs.

The induced subgraph characterization is a useful tool for considering inclusion relationships between hereditary classes of graphs. It is not difficult to see that a hereditary class X contains a hereditary class Y if and only if every graph which is forbidden for X contains an induced subgraph which is forbidden for Y . Showing that, for example, a bipartite graph is a perfect graph is therefore a simple task. Such results might otherwise be very difficult to obtain, and the tools used might be clumsy and ad hoc. Indeed, in 1969, the *Journal of Combinatorial Theory* published a paper entitled “An interval graph is a comparability graph” [12]. One year later, the same journal published another paper entitled “An interval graph is not a comparability graph” [13]. Both the interval graphs and the comparability graphs form hereditary classes. Apparently, in 1969 the induced subgraph characterization was not available for at least one of them. Nowadays, it is available for both classes.

This example shows that the problem of finding the set of minimal forbidden induced subgraphs for a hereditary class X is an important task. However, this problem is generally far from being trivial. For instance, for the class of perfect graphs this problem was open for several decades [14].

The induced subgraph characterization is also important because it provides a systematic way to study various problems on hereditary classes. When we look at a particular problem from the NP-complete side, we want to identify the best possible restrictions under which the problem remains NP-complete by extending the list of forbidden induced subgraphs. In a sense, we want to identify “minimal difficult”

classes. The role of such classes is as important as the role of minimal forbidden induced subgraphs. The difficulty is that such classes may not exist and typically an NP-complete problem admits an infinite narrowing sequence $X_1 \supset X_2 \supset \dots$ of hereditary classes each of which is difficult for the problem. This information is also valuable, because it suggests that if we want to identify a polynomially solvable case for the problem, we need to exclude (forbid) a graph from each class of the sequence $X_1 \supset X_2 \supset \dots$. Moreover, if we want to develop a polynomial-time algorithm in a hereditary class Y defined by *finitely many* forbidden induced subgraphs, we need to exclude a graph from the intersection of all classes in the sequence $X_1 \supset X_2 \supset \dots$. This intersection is called a *limit class* and a minimal limit class is called a *boundary class* for the problem. The importance of this notion is due to the fact that an NP-complete problem is polynomial-time solvable in a hereditary class Y defined by finitely many forbidden induced subgraphs if and only if Y does not contain any boundary class for the problem. However, identifying boundary classes is even harder than identifying minimal forbidden induced subgraphs for a hereditary class. Therefore, in some cases we restrict ourselves to the weaker task of identifying limit classes. For a general discussion of boundary properties of graphs and a short survey of related results, see [15]

We apply this approach to several NP-complete problems closely related to cliques, colouring and satisfiability. We start by looking at two problems related to vertex colouring in Part 1 of the dissertation. In particular, we identify several limit classes and several polynomially solvable cases for vertex 3-COLOURABILITY. In Part 2, we switch to independent sets, which is the notion complement to cliques. The reason we study independent sets instead of cliques is that they provide a more convenient language to describe restrictions in terms of forbidden induced subgraphs. In this part, we deal with the MAXIMUM INDEPENDENT SET problem, some of its generalizations (MAXIMUM SPARSE REGULAR INDUCED SUBGRAPHS), and MINIMUM MAXIMAL INDEPENDENT SET (also known as INDEPENDENT DOMINATION). The latter problem restricted to the class of so-called SAT-graphs is equivalent to SATISFIABILITY. We switch to this problem in Part 3 of the dissertation, where we use the language of graph theory in order to identify the first boundary property for the SATISFIABILITY problem, and as a corollary, a boundary property for INDEPENDENT DOMINATION.

1.1 Definitions and Notation

Graph Theory

All graphs in this dissertation are finite, simple graphs. The vertex set of a graph G is denoted $V(G)$ and the edge set $E(G)$.

If there is an edge between vertices u and v , we say they are *adjacent* or *neighbours*, and *non-adjacent* or *non-neighbours* otherwise. The set of all vertices adjacent to v is called the *neighbourhood* of v , denoted by $N(v)$. The *degree* of a vertex, $d(v)$, is the size of its neighbourhood. The *antineighbourhood* of a v is the set of vertices outside v which are non-adjacent to v , and is denoted $A(v)$. For a given set of vertices U , we define $N_U(v)$ and $A_U(v)$ to be the intersection of U with the neighbourhood and antineighbourhood of v respectively. The neighbourhood of U is the union of the neighbourhoods of its members, and its antineighbourhood is the union of the antineighbourhoods of its members in $V(G) \setminus U$. If X, Y are sets of vertices in a graph G , and every member of Y is in $N(X)$, then X is said to *dominate* Y .

A path between two vertices u, v in a graph G is a set of vertices $\{u, x_1, x_2, \dots, x_{k-1}, v\}$ such that $E(G)$ includes the edges $ux_1, x_i, x_{i+1}, x_{k-1}v$. We sometimes refer to such a path as a $u - v$ *path* and we say it has *length* k . The distance from u to v , denoted $d(u, v)$, is the length of a shortest path between them and those vertices at distance k from u are sometimes called the k -*neighbourhood* of u . If u and v are adjacent, $\{u, v\}$ is a $u - v$ path and the distance between them is 1. The distance between two sets of vertices $U_1, U_2 \subseteq V(G)$ is the minimum shortest $u_1 - u_2$ path for $u_1 \in U_1, u_2 \in U_2$. If there is a path between any two vertices in G , we say that G is *connected*.

Two graphs G_1, G_2 are said to be *isomorphic* if there is a bijection ϕ between their vertex sets that preserves edges. In other words $uv \in E(G_1)$ if and only if $\phi(u)\phi(v) \in E(G_2)$.

A subgraph of G is a subset of the vertices and edges of G , and is said to be *induced* by a set of vertices U if it can be obtained from G by deleting the vertices outside U and their incident edges. We denote such a subgraph by $G[U]$. If G contains no induced subgraphs isomorphic to a graph in a set M , we say that G is M -free. The *complement* of G , denoted \bar{G} , has the same vertex set as G and with $uv \in E(\bar{G})$ if and only if $uv \notin E(G)$.

If every pair of vertices in U is adjacent, we call U a *clique* of G . If no two vertices in U are adjacent, then U is an *independent set* of G . A graph is *bipartite* if it can be partitioned into two independent sets, and *complete bipartite* if every

possible edge between those two parts is in the graph.

As usual, P_n , C_n and K_n stand, respectively, for a chordless path, a chordless cycle and a complete graph on n vertices. $K_{n,m}$ is the complete bipartite graph with parts of size n and m . A wheel W_n is obtained from a cycle C_n by adding a dominating vertex, i.e. a vertex adjacent to every vertex of the cycle. A *forest* is a graph with no cycles and a *tree* is a connected forest.

The *disjoint union* of two graphs G_1, G_2 is the graph G with $V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$, and is denoted $G_1 + G_2$. The disjoint union of a graph G and itself is denoted $2G$ and this notation is extended to the disjoint union of m copies of G , denoted mG .

For positive integers i, j, k , the graph $S_{i,j,k}$ is the tree with three leaves, at distance i, j and k from the unique vertex of degree 3. For example, $S_{1,1,1}$ is isomorphic to $K_{1,3}$. The value of i, j or k may be 0, in which case $S_{i,j,k}$ is just a path. The class of graphs whose every connected component is of the form $S_{i,j,k}$ is denoted \mathcal{S} . This class is important and appears a number of times in this thesis.

For some particular graphs we use special names:

- a *claw* is $K_{1,3}$;
- a *triangle* is a K_3 ;
- a *diamond* is the graph obtained from K_4 by removing one edge;
- a *butterfly* is two triangles sharing a vertex.

For positive integers i, j, k , the graph $S_{i,j,k}$ is the tree with three leaves, at distance i, j and k from the unique vertex of degree 3. For example, $S_{1,1,1}$ is a claw or $K_{1,3}$. The value of i, j or k may be 0, in which case $S_{i,j,k}$ is just a path. The class of graphs whose every connected component is of the form $S_{i,j,k}$ is denoted \mathcal{S} . This class is important and appears a number of times in this thesis.

A vertex v is said to *distinguish* a set of vertices U if it has at least one neighbour and one non-neighbour in U . A set of vertices M in G that is not distinguished by any vertex in $V(G) \setminus M$ is called a *module*. Clearly a single vertex is a module, as is the empty set and the set of all vertices in a graph. These are called *trivial* modules, all other modules are said to be *non-trivial*. A graph whose every module is trivial is a *prime* graph.

A graph every vertex of which has degree k is *k-regular*, the set of *regular graphs* is the union of the k -regular graphs for all values of k . A 1-regular (induced) subgraph is called an (*induced*) *matching*.

A *tree decomposition* of a graph $G = (V, E)$ is a pair (T, \mathcal{X}) consisting of a tree $T = (\mathcal{I}, F)$ and a collection of subsets of the graph's vertex set $\mathcal{X} = \{X_i : i \in \mathcal{I}, X_i \subseteq V\}$ such that

- $\bigcup_{i \in \mathcal{I}} X_i = V$,
- for every $e = \{u, v\} \in E$, there exists an $i \in \mathcal{I}$ such that $u, v \in X_i$, and
- for every $v \in V$, the subgraph of T induced by $\{X_i \in \mathcal{X} : v \in X_i\}$ is a tree.

The *width of a tree decomposition* (T, \mathcal{X}) is $w((T, \mathcal{X})) = \max\{|X_i| : i \in \mathcal{I}\} - 1$. The *tree-width* of a graph G , $tw(G)$, is the minimum width of a tree decomposition of G . Graphs of tree-width 0 are precisely the edgeless graphs, and graphs of tree-width at most 1 are forests. We refer to [16] for an excellent tutorial on tree-width.

The *clique-width* of a graph G is the minimum number of labels needed to construct G using the following four operations:

1. Creating a new vertex v with label i (denoted by $i(v)$).
2. Taking the disjoint union of two labelled graphs G and H (denoted by $G \oplus H$).
3. Joining each vertex with label i to each vertex with label j for $i \neq j$ (denoted by $\eta_{i,j}$).
4. Renaming label i to j (denoted by $\rho_{i \rightarrow j}$).

An intuitive understanding of the tree-width parameter is that it measures how “tree-like” a graph is. This author knows no similar visualisation of clique-width. However, a graph class where either parameter is bounded has a simple structure in some sense. Often, restricting a problem to such graph classes yields an efficient solution.

Algorithms and Complexity

Here, we briefly outline the basic definitions and concepts necessary to understand the results of this thesis. For a full treatment of the topic see e.g. [17].

In this thesis, a *problem* is a mathematical question to be answered, along with a description of the variables involved in the question (*input*), and a description of the properties that the answer must have (*output*). Many of the problems we consider in this thesis are *decision problems*, in other words the answer is either yes or no. An *instance* of a problem is obtained by specifying values for the variables of the problem. An *algorithm* is a set of instructions for solving a problem. An

algorithm is only said to *solve* a problem Π if it can be applied to any instance I of Π and always produce a correct output or *solution*.

In general, we wish to find the “fastest” algorithm to solve a given problem. We define the *running time* of an algorithm to be the number of elementary operations the algorithm makes as a function of the size of the input, usually denoted n . We say an algorithm runs in *polynomial time* if its running time is bounded by a polynomial in n . We say that a problem is *polynomially solvable* if an algorithm exists which solves the problem in polynomial time. The class of polynomially solvable decision problems is usually denoted P . Informally, we think of problems that are in P as “easy” to solve. In other words, an algorithm that runs in polynomial time is considered “efficient”. If no such algorithm exists, we consider the problem to be “hard”.

The other main class we consider in this thesis is denoted NP and is intuitively thought of as the class of decision problems whose “yes” answers can be checked efficiently. More formally, a decision problem Π is in NP if each instance I for which the correct answer is “yes” admits a proof of this answer that can be verified in polynomial time. For example, consider the problem whose input is a graph G and whose output should be “yes” or “no” in answer to the question “does G contain an independent set of size k ”. This problem is in NP because, if the answer is “yes”, specifying the independent set would constitute such a proof.

In order to compare the complexity of solving problems more formally, we introduce the notion of reduction between problems. A *polynomial time reduction* from a problem Π_1 to a problem Π_2 is an algorithm which solves Π_1 in polynomial time, given polynomially many uses of a hypothetical algorithm that solves Π_2 called an oracle. If Π_1 can be reduced to Π_2 , then any polynomial time algorithm that solves Π_2 can be converted to a polynomial time algorithm that solves Π_1 . Informally, we think of Π_2 as being “at least as hard” as Π_1 . A problem Π is *hard* for a complexity class \mathcal{C} if every problem in \mathcal{C} may be reduced to Π in polynomial time. The class of such problems is denoted \mathcal{C} -hard. If Π is itself in \mathcal{C} , it is said to be *complete* for the class \mathcal{C} , or \mathcal{C} -complete. Many of the problems considered in this thesis are NP -complete, and these problems are thought of as the “hardest” problems in NP .

There is no known proof that NP contains any problem that is not in P . In other words it may be the case that efficiently checkable problems are also efficiently solvable problems. as we’ve mentioned, this is one of the most famous open problems in computer science, sometimes called the P vs. NP problem.

Part I

Colouring
and
Related Problems

Introduction

Many graph problems fall into the broad category of colouring problems. The main problem in this category is VERTEX COLOURING. We say that a colouring of the vertices of a graph is *proper* if no two adjacent vertices have the same colour. VERTEX COLOURING is the problem of finding a proper colouring with the minimum number of colours.

One may also consider colouring the edges of a graph in a similar manner. A total colouring is a colouring of both the vertices and edges of the graph. Edge and total colourings of a graph G are equivalent to colouring the vertices of special graphs obtained from G . The line graph $L(G)$ has a vertex for each edge of G and an edge between two vertices if and only if the corresponding edges of G share an endpoint. Clearly colouring the vertices of $L(G)$ corresponds exactly to colouring the edges of G . The total graph $T(G)$ can be defined in a similar manner, i.e. for each vertex or edge in G there is a vertex in $T(G)$, with an edge in $T(G)$ for each vertex-vertex, edge-edge and edge-vertex adjacency in G . Famously, the faces of any planar graph may be coloured with four colours so that two faces sharing an edge have different colours. This is equivalent to colouring the vertices of the dual of a plane graph G , whose vertex set is the faces of G , with an edge between two vertices if the corresponding faces share an edge. One may also consider a list colouring, that is a colouring of the vertices of a graph where each vertex is restricted to some finite list of colours.

In this thesis we use the language of vertex colouring. The problem of deciding whether there exists a proper colouring of G using at most k colours is known as the k -COLOURABILITY problem. It was one of Karp's 21 NP-complete problems given in [18], and was in fact the first to be mentioned in the introduction to that seminal paper. It remains a difficult problem under a variety of substantial restrictions, and yet in some cases efficient solutions are known, i.e. a polynomial time algorithm exists to solve the problem.

For example, 3-COLOURABILITY is known to be NP-complete even for graphs

of vertex degree at most four, but for graphs of vertex degree at most 3, a polynomial time solution exists (the complexity gap is made more precise in [19]). The problem can also be solved efficiently for locally connected graphs [20].

Given the rich possibilities for research into such restrictions, it is hardly surprising that colourability remains a central problem of graph theory and computational complexity theory. Indeed, a number of papers have recently investigated restricting the problem to various classes of graphs defined by a set of forbidden induced subgraphs, otherwise known as hereditary classes.

Maffray and Preissmann showed that, for $k \geq 4$, k -COLOURABILITY remains NP-complete for triangle-free graphs, reducing from graphs of vertex degree at most 4 to triangle-free graphs of vertex degree at most 4, by replacing certain vertices with a special gadget based on the Mycielski Graph [21].

A series of papers by Randerath gave a number of similarly framed results. In [22], Randerath identified every pair of connected graphs A, B such that every $\{A, B\}$ -free graph is 3-colourable. Randerath, Schiermeyer and Tewes gave good examples of different approaches to the problem of graph colouring in [23]. Firstly, they gave structural analyses of certain hereditary classes of graphs. Then a structural analysis of only the non-perfect K_4 -free members of the considered graph class, making use of Tucker's polynomial solution for perfect K_4 -free graphs [24]. Finally, they make use of a reformulation of 3-colourability in terms of propositional logic, and also consider precolouring a special set and extending the colouring to the rest of the graph.

In this part of the thesis, we study two problems related to vertex colouring. One of them is 3-colourability. This problem is also closely related to a number of partition problems. For example, the STABLE- Π PARTITION problem asks if a graph can be partitioned into an independent set and a graph in the class Π . If Π is the class of all bipartite graphs, STABLE- Π PARTITION coincides with 3-colourability, and in that sense generalises the problem. In Chapter 3 we consider the DOMINATING INDUCED MATCHING problem, sometimes called EFFICIENT EDGE DOMINATION in the literature. This problem can be formulated in a number of ways, for example one might ask if a graph can be partitioned into an induced matching and an independent set, and in this sense the problem is also a special case of STABLE- Π PARTITION. Alternatively, one could ask for a 3-colouring of the graph such that two colour classes induce a matching, or a sort of 2-colouring in which every neighbour of a white vertex is coloured black, and each black vertex has exactly one black neighbour.

Chapter 2

Vertex 3-Colourability in Claw-free Graphs

2.1 Introduction

In an attempt to find the greatest restrictions under which COLOURABILITY remains NP-complete, we consider two types of restrictions. We restrict ourselves to claw-free graphs, a class which has received considerable attention in the literature due to the many attractive properties of graphs in this class (see for example [25, 26, 27, 28]). We also restrict ourselves to 3-COLOURABILITY, which is minimal in the sense that k -COLOURABILITY is NP-complete for $k \geq 3$, but there exists an algorithm to solve 2-COLOURABILITY which runs in polynomial time. The restriction to claw-free graphs is also minimal in a certain sense. For any induced subgraph H of a claw, 3-COLOURABILITY can be solved in polynomial time in the class of H -free graphs. On the other hand in the class of claw-free graphs, 3-COLOURABILITY is NP-complete. This is because the 3-COLOURABILITY problem in claw-free graphs includes, as a subproblem, EDGE 3-COLOURABILITY of general graphs, i.e. the problem of determining whether the edges of a given graph can be assigned colours from set $\{0, 1, 2\}$ so that any two edges sharing a vertex receive different colours. Indeed, by associating with a graph G its line graph $L(G)$ (i.e. the graph with $V(L(G)) = E(G)$ and two vertices being adjacent in $L(G)$ if and only if the respective edges of G have a vertex in common), one can transform the question of edge 3-colourability of G into the question of vertex 3-colourability of $L(G)$. In conjunction with the NP-completeness of EDGE 3-COLOURABILITY [29], this implies the NP-completeness of (vertex) 3-COLOURABILITY of line graphs. It is known that every line graph is claw-free. Moreover, the line graphs constitute a *proper* subclass

of claw-free graphs, which can be characterized by 8 additional forbidden induced subgraphs (see e.g. [30] for the complete list of minimal non-line graphs). In this chapter we study the computational complexity of the problem in other subclasses of claw-free graphs. First, we derive a necessary condition for the polynomial-time solvability of the problem in such classes and then reveal several areas where this condition becomes sufficient. In the remainder of this section we introduce and recall some notation, and define three operations on graphs with which we obtain our results.

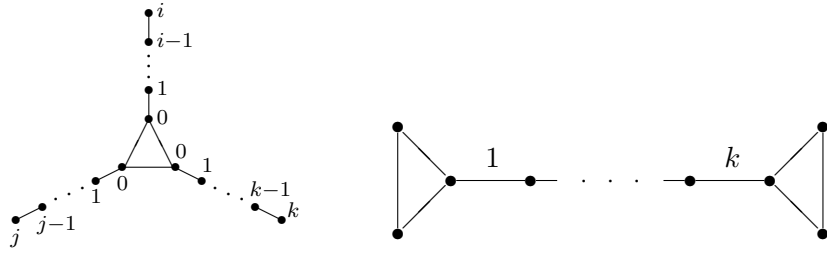


Figure 2.1: The graphs $T_{i,j,k}^1$ (left) and Φ_k (right)

By $T_{i,j,k}^1$ and Φ_k we denote the graphs represented in Figure 2.1 on the left and right, respectively. Note that each case the subscript values are non-negative, for example $T_{0,0,0}^1$ is a triangle, and Φ_0 is the so-called butterfly graph, obtained from two triangles by joining them at a single vertex.

We can assume that every vertex has degree at least 3. Indeed, if v is a vertex of G of degree less than three, then G has a 3-colouring if and only if the graph obtained from G by deleting v has one. Moreover, whenever we deal with claw-free graphs, we can restrict ourselves to graphs of vertex degree at most four. Indeed, it is not difficult to verify that every graph with five vertices contains either a triangle or its complement or a C_5 . Therefore, every graph with a vertex of degree five or more contains either a K_4 or a claw or W_5 . Since K_4 and W_5 are not 3-colourable, we conclude that every claw-free graph, which is 3-colourable, has maximum vertex degree at most 4.

Let us repeat that a *diamond* is the graph obtained from a K_4 by deleting an edge. Diamonds are of special interest in the vertex 3-COLOURABILITY problem. So we will introduce special terminology related to this graph. In a diamond, the vertices of degree 3 will be called *central* and the vertices of degree 2 *peripheral*. If a graph G contains a diamond D and if both central vertices of D have degree 3 in G (i.e. they do not have neighbours in G outside D), then we will call D a *pure*

diamond. If additionally both peripheral vertices of D have degree at most 3 in G , then D will be called a *perfect diamond*. We also define two extensions of the diamond, the *gem* and *crystal*, represented in Figure 2.2

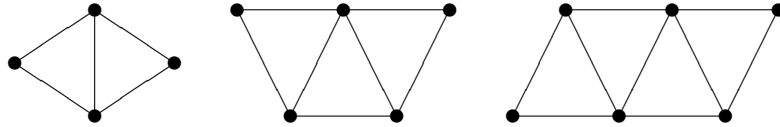


Figure 2.2: The *diamond*, *gem* and *crystal* graphs, left to right.

The importance of diamonds for the vertex 3-COLOURABILITY problem is due to the obvious fact that the peripheral vertices of any diamond have the same colour in any 3-colouring of G , if G has any. This allows us to introduce the following three operations.

The first of the operations is called *diamond implantation* and consists of splitting the neighbourhood of a vertex v into two subsets, say A and B , replacing v by two new vertices, say a and b , connecting a to each vertex in A , connecting b to each vertex in B , and creating a diamond with peripheral vertices a and b (see Figure 2.3 for an illustration). If we denote by G' the graph obtained from G by implanting a diamond, then clearly G' is 3-colourable if and only if G is. Also, if G is claw-free and both A and B are cliques, then G' is claw-free too.

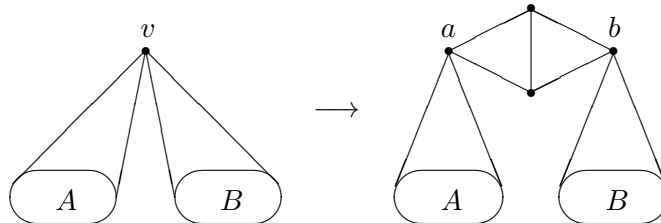


Figure 2.3: Diamond implantation

The second operation is called *pure diamond contraction* and it is opposite to diamond implantation. It consists of deleting the central vertices of a pure diamond and identifying its peripheral vertices. In other words, this is the operation that transforms the graph on the right of Figure 2.3 to the graph on the left. Again, it is obvious that if G' is the graph obtained from G by contracting a pure diamond, then G' is 3-colourable if and only if G is. Moreover, it is not difficult to see that if

G is claw-free, then G' also is claw-free.

The third operation is *perfect diamond deletion*. If a graph G contains a perfect diamond D , then deletion of the vertices of D results in a graph which is 3-colourable if and only if G is, and which is claw-free if and only if G is.

Finally, we introduce one more useful transformation, which is shown in Figure 2.4 and which is called *triangle implantation*. Clearly, if G' is the graph obtained from G by implanting a triangle into a triangle of G , then G' is 3-colourable if and only if G is.

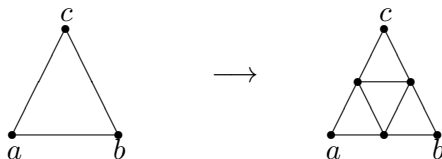


Figure 2.4: Triangle implantation

2.2 NP-completeness results

In this section we establish several results on the NP-completeness of the 3-COLOURABILITY problem in subclasses of claw-free graphs. We start by proving three lemmas. The first of them improves the following known fact: the 3-COLOURABILITY problem is NP-complete in the class of (claw,diamond)-free graphs of maximum vertex degree at most four. Now we strengthen this result in the following way. Denote

$$M_k^1 = \{\text{claw, diamond, } K_4, C_4, C_5, \dots, C_k\}.$$

Lemma 1. *For any natural $k \geq 4$, the 3-COLOURABILITY problem in the class of M_k^1 -free graphs of maximum vertex degree at most four is NP-complete.*

Proof. It is known that the EDGE 3-COLOURABILITY problem is NP-complete in the class of $(C_3, C_4, C_5, \dots, C_k)$ -free cubic graphs (i.e. graphs in which every vertex has degree 3) for any natural k [31]. Colouring edges of a graph G is equivalent to colouring vertices of $L(G)$, the line graph of G . It is known that the line graph of any graph is claw-free. Also, it is not difficult to see that if G is a C_3 -free cubic graph, then $L(G)$ is a (diamond, K_4)-free regular graph of degree four. Finally, if G is $(C_3, C_4, C_5, \dots, C_k)$ -free, then $L(G)$ is (C_4, C_5, \dots, C_k) -free. \square

To prove our next two lemmas, we introduce the following notation. We denote by Φ'_k , Φ''_k , $\Phi_{k,p}$ the graphs represented in Figure 2.5.

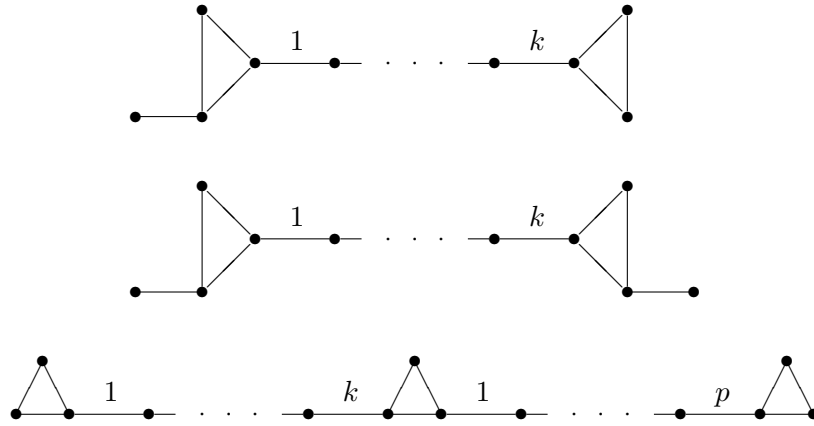


Figure 2.5: Graphs Φ'_k , Φ''_k and $\Phi_{k,p}$ (from top to bottom)

Now we denote

$$M_k^2 = \{\text{claw, gem, } K_4, C_4, C_5, \dots, C_k, \Phi'_1, \Phi'_3, \dots, \Phi'_{2k+1}, \Phi''_0, \Phi''_2, \dots, \Phi''_{2k}\}.$$

Lemma 2. *For any natural $k \geq 4$, the 3-COLOURABILITY problem is NP-complete in the class of M_k^2 -free graphs of maximum vertex degree at most four.*

Proof. We prove the lemma by a reduction from the same problem in the class of (claw, gem, K_4 , C_4 , C_5 , \dots , C_k)-free graphs of maximum degree four, where the problem is NP-complete by Lemma 1 (since Lemma 1 deals with a smaller class of graphs).

Let G be a (claw, gem, K_4 , C_4 , C_5 , \dots , C_k)-free graph of maximum degree four. It is not difficult to see that every diamond D in G is pure, i.e. the central vertices of D have no neighbours in G outside D , since otherwise a claw, gem or K_4 arises.

In polynomial time, we will transform G into an M_k^2 -free graph G^* of degree at most four, which is 3-colourable if and only if G is. Without loss of generality, we will assume that G is connected and has no vertices of degree 1 or 2. We will also assume that k is large enough, say $k = 10$. To describe the transformation, let us introduce some terminology.

A triangle in G will be called *branching* if it is contained in an induced $T_{1,1,1}^1$ (see Figure 2.1 for the graph $T_{i,j,k}^1$). Taking into account the assumption that G has no vertices of degree less than 3, it is not difficult to see that every triangle in G is

either branching or is contained in a diamond, and that it is sufficient for a triangle to be contained in an induced $T_{0,1,1}^1$ for it to be branching.

A vertex x of G will be called *splittable* if the neighbourhood of x can be partitioned into two disjoint cliques with no edges between them. It is easy to see that every vertex of a branching triangle is splittable, with one clique comprising of the other vertices of the triangle.

Now we describe the transformation. Let T be a branching triangle and v a vertex of T . We split the neighbourhood of v into two cliques A and B and apply the diamond implantation $k + 1$ times, i.e. replace v with two new vertices a and b , connect a to b by a chain of $k + 1$ diamonds, and connect a to every vertex in A and connect b to every vertex in B . We apply this operation to every vertex of every branching triangle of G and denote the resulting graph by G^* . It is not difficult to see that G can be transformed into G^* in polynomial time and that G^* is 3-colourable if and only if G is.

Obviously, G^* is (claw, gem, $K_4, C_4, C_5, \dots, C_k$)-free, since diamond implantation cannot create an induced claw, gem, or K_4 , and by applying diamond implantation to a vertex v we increase the length of any cycle C_p with $p \geq 4$ containing v .

Now let us show that G^* is $(\Phi_0'', \Phi_1'', \dots, \Phi_{2k}'')$ -free. Assume G^* contains an induced Φ_t'' for some value of t . Clearly, both triangles in Φ_t'' are branching. Therefore, both of them belong to G , since during the transformation we did not introduce any new branching triangle. Also, according to the transformation all triangles in the $2k$ -neighbourhood of any branching triangle belong to implanted diamonds. Therefore, the value of t must be strictly greater than $2k$, i.e. G^* is $(\Phi_0'', \Phi_1'', \dots, \Phi_{2k}'')$ -free.

Similarly we show that G^* is $(\Phi_1', \Phi_3', \dots, \Phi_{2k+1}')$ -free. Indeed, let G^* contain an induced Φ_t' and let T_1 be the triangle of this Φ_t' with a pendant edge and T_0 the triangle without one. Clearly T_1 is branching and therefore belongs to G . If T_0 is also branching, then the result follows as before. If T_0 belongs to a diamond, then its vertex meeting the path connecting it to T_1 must be a peripheral vertex of a diamond. On the other hand, we know that every vertex of an odd distance at most $2k + 1$ from T_1 in G^* is a central vertex of one of the implanted diamonds. Therefore, G^* cannot contain an induced Φ_t' with an odd $t \leq 2k + 1$. This completes the proof of the lemma. \square

Finally, we prove one more lemma for which we introduce one more notation:

$$M_k^3 = \{\text{claw, crystal, } K_4, C_4, C_5, \dots, C_k, \Phi_1'', \Phi_2'', \dots, \Phi_k''\} \cup \{\Phi_{i,j} : \text{even } i, j \leq k\}.$$

Lemma 3. *For any natural $k \geq 4$, the 3-COLOURABILITY problem is NP-complete for M_k^3 -free graphs of maximum vertex degree at most four.*

Proof. We reduce the problem from the class of M_k^2 -free graphs of degree at most four. Let G be a graph in this class, and let G' be obtained from G by implanting a triangle into each branching triangle. It is not difficult to see that this operation does not create any of the following forbidden induced subgraphs: $\{\text{claw}, K_4, C_4, C_5, \dots, C_k\}$. It does create a gem, but not a crystal.

In order to see that G' does not contain copies of Φ_t'' for $1 \leq t \leq k$, observe that triangle implantation breaks every branching triangle into three new branching triangles. We will call these three new branching triangles *adjacent*. Obviously, no two adjacent branching triangles belong to the same induced Φ_k'' , and the distance between any two non-adjacent branching triangles is greater than k , since it was greater than k in G . Therefore, G' is $(\Phi_1'', \Phi_2'', \dots, \Phi_k'')$ -free.

Finally, we show that G' does not contain graphs in the set $\{\Phi_{i,j} : i, j \text{ are even and } i, j \leq k\}$. Let T be a branching triangle. Every vertex of T starts a branch which we call an even branch if it contains a triangle which together with T create a Φ_p -graph with an even $p \leq k$. In G every branch of a branching triangle is even. However, by implanting a triangle we change the parity of two branches. Therefore, in G' every branching triangle has only one even branch. Since the central triangle in a $\Phi_{i,j}$ -graph is branching, we conclude that $\Phi_{i,j}$ -graphs with two even branches are forbidden in G' . \square

To further strengthen the NP-completeness results, let us denote by

\mathcal{S}_k^1 the class of M_k^1 -free graphs of maximum vertex degree at most four,

\mathcal{S}_k^2 the class of M_k^2 -free graphs of maximum vertex degree at most four,

\mathcal{S}_k^3 the class of M_k^3 -free graphs of maximum vertex degree at most four.

Also, with every graph G we associate the following three parameters: for $i \in \{1, 2, 3\}$, let $\kappa^i(G)$ be the maximum k such that $G \in \mathcal{S}_k^i$. If G belongs to no class \mathcal{S}_k^i , we define $\kappa^i(G)$ to be 0, and if G belongs to all classes \mathcal{S}_k^i , then $\kappa^i(G)$ is defined to be ∞ . Also, for a set of graphs M , we define $\kappa^i(M) = \sup\{\kappa^i(G) : G \in M\}$.

Theorem 4. *Let M be a set of graphs and X the class of M -free graphs of vertex degree at most 4. If $\kappa^i(M) < \infty$ for any $i \in \{1, 2, 3\}$, then the 3-COLOURABILITY problem is NP-complete in the class X .*

Proof. Let $i \in \{1, 2, 3\}$. To prove the theorem, we will show that if $\kappa^i(M) < \infty$ then there is a k such that $\mathcal{S}_k^i \subseteq X$. Denote $k := \kappa^i(M) + 1$ and let G belong to \mathcal{S}_k^i . Assume that G does not belong to X . Then G contains a graph $A \in M$ as an induced subgraph. From the choice of G we know that A belongs to \mathcal{S}_k^i , but then $k \leq \kappa^i(A) \leq \kappa^i(M) < k$, a contradiction. Therefore, $G \in X$ and hence, $\mathcal{S}_k^i \subseteq X$. \square

2.3 Polynomial-time results

We now proceed to polynomial-time results. Let M be a set of graphs. The results of the previous section suggest that, unless $P = NP$, the 3-COLOURABILITY problem is polynomial-time solvable in the class of claw- and M -free graphs only if

$$\kappa^1(M) \text{ is unbounded and } \kappa^2(M) \text{ is unbounded and } \kappa^3(M) \text{ is unbounded.} \quad (2.1)$$

In the present section, we identify several areas where condition 2.1 is sufficient for polynomial-time solvability of the problem. First of all, let us reveal two major ways to push the parameters $\kappa^i(M)$ to infinity. In order to unbind $\kappa^i(M)$ we need to include in M a graph from the class \mathcal{S}_k^i for each value of k . This is possible if

- either M contains infinitely many graphs, one for each value of k ,
- or M includes a graph which belongs to classes \mathcal{S}_k^i for all values of k , i.e. a graph from the intersection $\bigcap_{k \geq 4} \mathcal{S}_k^i$.

2.3.1 Infinitely many forbidden induced subgraphs

The *chordality* of a graph is the length of the largest chordless cycle. Therefore, (C_p, C_{p+1}, \dots) -free graphs have chordality at most $p - 1$. It is not difficult to see that for each value of p the set $\{C_p, C_{p+1}, \dots\}$ satisfies the necessary condition 2.1 for polynomial-time solvability of the 3-COLOURABILITY problem in subclasses of claw-free graphs. Now let us show that boundedness of the chordality is also a sufficient condition.

Theorem 5. *For each value of p , the 3-COLOURABILITY problem is polynomial-time solvable in the class of claw-free graphs of chordality at most p .*

Proof. Let G be a claw-free graph of chordality at most p . If the maximum vertex degree of G is more than 4, then G is not 3-colourable. Therefore, we assume that the vertex degree is bounded by 4 in G . It is known [32] that for each d and p there is a number $f(d, p)$ such that the tree-width of graphs of degree at most d and chordality

at most p is at most $f(d, p)$. Therefore, G is of bounded tree-width and hence is of bounded clique-width. According to [33] for each fixed k , the k -COLOURABILITY problem is polynomial-time solvable for graphs of bounded clique-width. \square

One more way to satisfy condition 2.1 by infinitely many subgraphs is to include in the set M of forbidden graphs large graphs of the form Φ_k , i.e. $\Phi_k, \Phi_{k+1}, \dots$ for some value of k . Let us show that this way also leads to a polynomially-solvable case.

Theorem 6. *For each value of k , the 3-COLOURABILITY problem is polynomial-time solvable in the class of (claw, $\Phi_k, \Phi_{k+1}, \dots$)-free graphs.*

Proof. Again we assume that a (claw, $\Phi_k, \Phi_{k+1}, \dots$)-free graph G is of degree at most 4, since otherwise it is not 3-colourable. We also assume without loss of generality that G is connected. If G contains no triangle then the maximum vertex degree is at most 2 since G is claw-free. In this case the problem is trivial (and G is of bounded tree- and clique-width). Now assume G contains a triangle. Then the set of vertices of G of distance at least $k + 2$ from the triangle must be K_3 -free, since otherwise a Φ_t with $t \geq k$ arises (due to connectedness of G). Therefore, the set of vertices of G of distance at least $k + 2$ from the triangle induces a graph of bounded tree- and clique-width. As a result, we conclude that G is of bounded tree- and clique-width, because there are only finitely many vertices of G of distance at most $k + 1$ from the triangle (as the degree of G is bounded). Thus, the problem is polynomial-time solvable for G . \square

2.3.2 Finitely many forbidden induced subgraphs

In the rest of this chapter we prove several polynomial-time results for subclasses of claw-free graphs with special emphasis on subclasses defined by additionally forbidding a single induced subgraph H , in which case H must belong to the intersection of all three classes $\mathcal{S}^1, \mathcal{S}^2$ and \mathcal{S}^3 , since otherwise the problem is NP-complete. The structure of graphs in this intersection is characterized in the following theorem, where by $T_{i,j,k}^\Delta$ we denote the graph represented in Figure 2.6 (left).

Theorem 7. *A graph G is in the intersection $\mathcal{S}^1 \cap \mathcal{S}^2 \cap \mathcal{S}^3$ if and only if each of its connected components is either a Φ_i with an odd i or a $T_{i,j,k}^\Delta$ with an even i or an induced subgraph of one of these two graphs.*

Proof. In order to prove this theorem let us recall the forbidden induced subgraphs of $\mathcal{S}^1, \mathcal{S}^2$ and \mathcal{S}^3 .

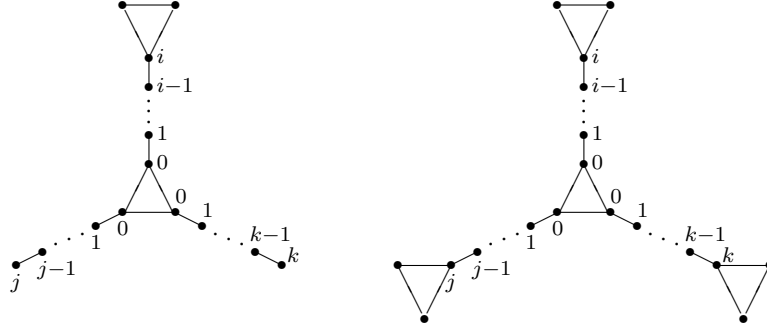


Figure 2.6: The graph $T_{i,j,k}^{\Delta}$ (left) and the graph $T_{i,j,k}^{3\Delta}$

$$\mathcal{S}^1 \subset \text{Free}(\{\text{claw, diamond, } K_4\} \cup \{C_k : k \geq 4\})$$

$$\mathcal{S}^2 \subset \text{Free}(\{\text{claw, gem, } K_4\} \cup \{C_k : k \geq 4\} \cup \{\Phi'_i : \text{odd } i\} \cup \{\Phi''_i : \text{even } i\})$$

$$\mathcal{S}^3 \subset \text{Free}(\{\text{claw, crystal, } K_4\} \cup \{C_k : k \geq 4\} \cup \{\Phi''_i : i \geq 0\} \cup \{\Phi_{i,j} : \text{even } i, j\})$$

It is easy to check that the graphs Φ_i for odd i and $T_{i,j,k}^{\Delta}$ for even i do not contain any of the graphs listed above as induced subgraphs. Therefore if every connected component of a graph G is a Φ_i with an odd i or a $T_{i,j,k}^{\Delta}$ with an even i or an induced subgraph of one of these two graphs, then G is in the intersection. Now let us show the reverse inclusion.

Clearly, every graph in the intersection $\mathcal{S}^1 \cap \mathcal{S}^2 \cap \mathcal{S}^3$ is diamond-free, since every graph in \mathcal{S}^1 is diamond-free. It is not difficult to see that every connected diamond-free graph in \mathcal{S}^2 is of the form $T_{i,j,k}^{3\Delta}$ represented in Figure 2.6 with all three indices i, j, k being even or an induced subgraph of a graph of this form. Similarly, every connected diamond-free graph in \mathcal{S}^3 is of the form $T_{i,j,k}^{3\Delta}$ with at most one index being even.

Let G be a connected diamond-free graph belonging both to \mathcal{S}^2 and to \mathcal{S}^3 . If G contains no induced Φ_i with an odd i , then clearly G is of the form $T_{i,j,k}^{\Delta}$ or an induced subgraph of a graph of this form.

Now assume G contains an induced subgraph H isomorphic to Φ_i with an odd i . Since G belongs to \mathcal{S}^2 , it is of the form $T_{i,j,k}^{3\Delta}$ with i, j, k even or an induced

subgraph of a graph of this form. Therefore, neither of the triangles of H is the central triangle of $T_{i,j,k}^{3\Delta}$. If G contains at least one more triangle (except those contained in H), then it must contain an induced $\Phi_{k,p}$ with even k and p (see Figure 2.5). But then G does not belong to \mathcal{S}^3 . This contradiction shows that G coincides with $H = \Phi_i$. \square

According to this theorem and the results of the preceding section, the 3-COLOURABILITY problem in a class of (claw, H) -free graphs is solvable in polynomial time only if H contains at most two triangles in each of its connected components. The case of one triangle in each component of H is solved in polynomial time by combining the following two facts:

- if H is a graph every connected component of which is of the form $T_{i,j,k}^1$, then the clique-width of (claw, H) -free graphs of bounded vertex degree is bounded by a constant [34],
- the k -colourability problem on graphs of bounded clique-width is solvable in polynomial time [33].

In the case of two triangles in the same connected component of H there are only two results available in the literature. One of them deals with the class of (claw, H) -free graphs, where $H = \Phi_0 = T_{0,0,0}^\Delta$. A polynomial-time solution for the problem in this class was presented in [23]. Unfortunately, the authors of [23] did not claim any time complexity for their solution. A linear-time algorithm for this class was also presented in [35]. Moreover, in [35] the polynomial-time solvability of the problem was extended from $(\text{claw}, T_{0,0,0}^\Delta)$ -free graphs to $(\text{claw}, T_{0,0,k}^\Delta)$ -free graph for an arbitrary value of k . Below we extend this short list of polynomially solvable cases by two new classes of (claw, H) -free graphs with H containing two triangles in the same connected component.

2.3.2.1 (claw, Φ_1) -free graphs

We start with the following two technical lemmas.

Lemma 8. *If G is a (claw, Φ_1) -free graph containing a pure diamond, then the graph obtained from G by contracting the diamond is also (claw, Φ_1) -free. Moreover, the new graph is 3-colourable if and only if G is.*

Proof. Let G contain a pure diamond D induced by vertices a, b, c, d with b and c being the central vertices of D . The operation of contraction of D consists in

deletion of vertices b and c and identification of vertices a and d . We denote the vertex obtained by identifying a and d by ad and the resulting graph by G' .

If G' contains an induced copy of one of the forbidden graphs claw or Φ_1 , then vertex ad must belong to this copy. Moreover, it is not difficult to see that ad must be a vertex of degree 3 in the copy. But then either a or d is a vertex of degree 3 in an induced copy of a claw or Φ_1 in the graph G . This contradiction shows that G' is (claw, Φ_1)-free. Since the peripheral vertices in any induced diamond must have the same colour in any 3-colouring of the graph, we conclude that G' is 3-colourable if and only if G is. \square

Lemma 9. *Let G be a connected (claw, Φ_1)-free graph such that*

- (1) G contains no K_4 ,
- (2) G contains no pure diamond,
- (3) G contains a chordless cycle C_k with $k > 10$,
- (4) $3 \leq \deg(v) \leq 4$ for every vertex $v \in V(G)$,

then every vertex of G not in C which has a neighbour on C is adjacent to exactly two consecutive vertices of the cycle.

Proof. Denote the vertices of C by u_0, u_1, \dots, u_{k-1} . Let v be a vertex outside C with a neighbour on C , say v is adjacent to u_1 . To avoid the claw induced by u_0, u_1, u_2, v , we conclude that v has a neighbour in $\{u_0, u_2\}$, say v is adjacent to u_2 . If v has no other neighbours on C , we are done. So, assume v has more neighbours on C . We split the analysis into two cases according to the number of neighbours of v on C .

Case 1: v has three neighbours on C . Denote the third neighbour by u_i . If neither $i = 0$ nor $i = 3$, then u_i is the centre of the claw induced by u_{i-1}, u_i, u_{i+1}, v , a contradiction. Therefore, assume without loss of generality that $i = 3$. To avoid a pure diamond induced by v, u_1, u_2, u_3 we conclude without loss of generality that u_2 has a neighbour w , and to avoid the claw induced by w, u_1, u_2, u_3 we conclude without loss of generality that w is adjacent to u_3 . Since G is K_4 -free, vertex w cannot be adjacent to v . Therefore, w is adjacent to u_4 , since otherwise the vertices u_3, u_4, v, w induce a claw in G , and hence w is not adjacent to u_0 , since otherwise the vertices w, u_0, u_2, u_4 induce a claw in G .

Since $\deg(u_0) > 2$, we may consider a neighbour x of u_0 . Then x is different from v , since v has only three neighbours on C by our assumption, and x is different from w , since otherwise w, u_0, u_2, u_4 induce a claw. We also know that x is not adjacent to u_2 or u_3 , since otherwise u_2 or u_3 have degree more than 4.

Assume first that x is not adjacent to u_1 . Then x must be adjacent to u_{k-1} to avoid the claw induced by u_0, u_{k-1}, u_1, x . But then G contains Φ_1 induced either by $x, u_{k-1}, u_0, u_1, u_2, v$ (if x is not adjacent to v) or by $u_{k-1}, u_0, x, v, u_2, u_3$ (if x is adjacent to v), which is a contradiction.

Assume now that x is adjacent to u_1 . If x is not adjacent to w , then vertices x, u_0, u_1, u_2, u_3, w induce a Φ_1 , a contradiction. If x is adjacent to w , then it must also be adjacent to u_4 (to avoid the claw induced by w, x, u_2, u_4), and consequently, x must be adjacent to u_5 (to avoid the claw induced by u_4, u_3, u_5, x), but then x has degree more than 4. This final contradiction completes the proof of Case 1.

Case 2: v has four neighbours on C . It is not difficult to see that v must be adjacent to two pairs of consecutive vertices of C , since otherwise a claw arises.

Case 2.1: Assume first that v is adjacent to two sets of two consecutive vertices, separated by a single vertex, say u_1, u_2, u_4, u_5 . Since $\deg(u_6) > 2$, we know that u_6 must have a third neighbour, say w .

Case 2.1.1: Assume w is not adjacent to u_5 . Then it must be adjacent to u_7 to avoid a claw centred at u_6 . Also, w must be adjacent to u_4 (to avoid the Φ_1 induced by v, u_4, u_5, u_6, u_7, w) and hence to u_3 (to avoid a claw). Now we consider the vertex u_8 which also must have a third neighbour, say x . This neighbour must be different from w , since w already has four neighbours.

Case 2.1.1.1: Assume x is not adjacent to u_7 . Then it must be adjacent to u_9 (to avoid a claw), which implies it must be adjacent to u_6 (to avoid the Φ_1 induced by w, u_6, u_7, u_8, u_9, x), and to u_5 (again to avoid a claw). But now v, u_4, u_5, x, u_8, u_9 induce a Φ_1 , a contradiction.

Case 2.1.1.2: Suppose x is adjacent to u_7 . Since u_4 is of degree 4, we conclude that x is adjacent to u_3 (to avoid the Φ_1 induced by u_3, u_4, w, u_7, u_8, x) and hence to u_2 (to avoid a claw). But now v, u_1, u_2, x, u_7, u_8 induce a Φ_1 . This contradiction completes the proof of Case 2.1.1

Case 2.1.2: Assume w is adjacent to u_5 . To avoid the Φ_1 induced by u_1, u_2, v, u_5, u_6, w , we conclude that w has a neighbour in $\{u_1, u_2\}$. Since w must have four neighbours on C which are two pairs of consecutive vertices of C , we know that w has no neighbours outside the cycle and that w is not adjacent to u_7 . Therefore, any third neighbour x of u_7 is different from w . Then x is adjacent to u_6 , since otherwise x must be adjacent to u_8 (to avoid a claw) in which case w, u_5, u_6, u_7, u_8, x induce a Φ_1 . Now we conclude that x is adjacent to u_4 (to avoid a Φ_1 induced by v, u_4, u_5, u_6, u_7, x) and hence to u_3 (to avoid a claw).

We know that w must have two consecutive neighbours on C one of which is from $\{u_1, u_2\}$. If the second of these two neighbours is different from u_3 , then these neighbours together with w, u_6, u_7, x induce a Φ_1 . Therefore, we must conclude that w is adjacent to u_2, u_3 .

To complete the proof of this case, we consider a third neighbour z of u_0 , which is clearly different from v, w, x since all of them have degree 4. By analogy with Case 2.1.1 (i.e. by symmetry with vertex w) we conclude that z is adjacent to u_1 . Also, since u_4 and u_5 have degree 4 already, we know that z is not adjacent to u_4 and u_5 . But then z, u_0, u_1, v, u_4, u_5 induced a Φ_1 . This contradiction completes the proof of Case 2.1.2, and hence of Case 2.1.

Case 2.2: Assume now that v is adjacent to two sets of two consecutive vertices, say u_1, u_2, u_j, u_{j+1} , such that C contains at least two vertices between u_2 and u_j and at least two vertices between u_{j+1} and u_1 . Without loss of generality we will assume that the path connecting u_2 to u_j is not shorter than the path connecting u_{j+1} to u_1 on the cycle. This implies in particular that $j \geq 7$. Now consider the vertex u_3 , which must have some neighbour not on the cycle, say w .

Case 2.2.1: Assume w is not adjacent to u_2 . Then it must be adjacent to u_4 to avoid a claw. Therefore, it is adjacent to u_1 (to avoid a Φ_1 induced by v, u_1, u_2, u_3, u_4, w) and hence to u_0 (again to avoid a claw). But now the neighbourhood of w satisfies conditions of Case 2.1, which is impossible.

Case 2.2.2: Suppose w is adjacent to u_2 . In order to avoid the Φ_1 induced by $w, u_3, u_2, v, u_j, u_{j+1}$, we conclude that vertex w must have two more consecutive neighbours on C at least one of which must belong to $\{u_j, u_{j+1}\}$. This implies in particular that any third neighbour x of u_0 must be different from w . Also by symmetry with w we conclude that x is adjacent to u_1 and has two more consecutive neighbours on C at least one of which must belong to $\{u_j, u_{j+1}\}$. But then x, u_0, u_1, u_2, u_3, w induce a Φ_1 in G . This contradiction completes the proof of Case 2.2.2, and hence of Case 2.2.

Case 2.3: Assume that all four neighbours of v are consecutive on the cycle, say v is adjacent to u_1, u_2, u_3, u_4 . Since $\deg(u_5) > 2$, vertex u_5 must have some other neighbour outside the cycle, say w .

Case 2.3.1: Assume w is not adjacent to u_4 . Then it must be adjacent to u_6 to avoid a claw. This implies that w is adjacent to u_3 , since otherwise the vertices v, u_3, u_4, u_5, u_6, w induce a Φ_1 , and therefore w is adjacent to u_2 , since otherwise u_3, u_2, u_4, w induce a claw. But now the vertices v, u_1, u_2, w, u_5, u_6 induce a Φ_1 in G , a contradiction.

Case 2.3.2: Suppose now that w is adjacent to u_4 . If w is adjacent to u_1 , then it must also be adjacent to u_2 or u_3 to avoid a claw, in which case the neighbourhood of vertex w satisfies conditions either of Case 2.1 or of Case 2.2, which is impossible. Therefore w is not adjacent to u_1 . Then it must be adjacent to u_2 (to avoid the Φ_1 induced by u_1, u_2, v, u_4, u_5, w) and hence it must also be adjacent to u_3 (to avoid the claw induced by u_1, u_2, u_3, w). Since $\deg(u_0) > 2$, we may consider a neighbour x of u_0 . This neighbour must be different from w and not adjacent to w , since w already has four neighbours, and similarly it is not adjacent to u_2, u_3 . By analogy with Case 2.3.1, x must be adjacent to u_1 . But then x, u_0, u_1, u_2, u_3, w induce a Φ_1 . \square

Theorem 10. *The 3-COLOURABILITY problem can be solved in the class of (claw, Φ_1)-free graphs in polynomial time.*

Proof. If a (claw, Φ_1)-free graph contains a K_4 or a vertex of degree more than 4, it is not 3-colourable. Therefore, we assume that the maximum vertex degree of the input graph is 4 and it is K_4 -free (which can obviously be verified in polynomial time). We also contract any pure diamond if the graph has any. Finally, we delete vertices of low degree (1 or 2). This leaves us with a (possibly disconnected) graph satisfying conditions (1), (2) and (4) of Lemma 9 and we deal with each connected component G separately.

First we verify if G contains a chordless cycle of length at least 10. This can be done in polynomial time as follows. Determine if G contains an induced path P of length 9. If not, then G has no chordless cycle of length at least 10. Otherwise, delete all internal vertices of P and all their neighbours (which are not in P). Then check if the end vertices of P belong to the same connected component of the resulting graph. Checking this for each copy of an induced P_9 gives an answer to the question.

If G has no chordless cycle of length at least 10, then we apply a solution of Theorem 5. Otherwise, let $C = (u_0, u_1, \dots, u_k)$ ($k \geq 10$) be such a cycle. From Lemma 9 every vertex of G which has a neighbour on C is adjacent to exactly two consecutive vertices of C . Since G has no vertices of degree less than three, each vertex of C has a neighbour outside C .

Let T be a triangle formed by a vertex v outside C and two consecutive vertices u_i, u_{i+1} of C . We will call v the top vertex of T and u_i, u_{i+1} its base. Clearly no two triangles can have the same base, since otherwise either a claw or a K_4 arises. Two triangles will be called adjacent if they share of vertex of C , neighbouring if they are separated by exactly one edge of the cycle, and distant if

they are separated by more than one edge of the cycle.

First we observe that there is no edge connecting the top vertices of two distant triangles, since otherwise an induced Φ_1 arises. Then we notice that no triangle T can have two neighbouring triangles, since otherwise the top vertex of T must be adjacent to the top vertices of both neighbouring triangles (to avoid an induced Φ_1), in which case an induced claw arises. Therefore, the set of triangles consists of pairs of adjacent triangles, i.e. for every triangle there is a unique adjacent triangle and a unique neighbouring triangle. The top vertices of neighbouring triangles are necessarily adjacent (to avoid an induced Φ_1) and hence the top vertices of adjacent triangles are necessarily non-adjacent (since otherwise a claw arises). Thus the degree of each top vertex is at least 3. To conclude the proof, let us show that the degree of each top vertex is exactly 3. Let us denote by T_i the triangle with base u_i, u_{i+1} and by v_i its top vertex. Suppose T_{i-1} is an adjacent triangle and T_{i+2} is a neighbouring triangle for T_i . Assume v_i has a fourth neighbour, say w . Then, to avoid the claw induced by v_i, v_{i+2}, u_i, w , vertex w must be adjacent to v_{i+2} . Now, to avoid Φ_1 induced by $v_{i-1}, u_{i-1}, u_i, v_i, v_{i+2}, w$, vertex w must be adjacent to v_{i-1} . This, in turn, implies that w is adjacent to v_{i-3} and hence to v_{i-3} , and so on. As a result, we conclude that w must be adjacent to all top vertices, which is impossible, since the degree of w is at most 4. Therefore, we conclude that G consists of the cycle C and $|V(C)|/3$ top vertices. It is not difficult to see that such a graph is 3-colourable. \square

2.3.2.2 (claw, Φ_3)-free graphs

In the case of (claw, Φ_3)-free graphs a polynomial-time solution is based on the following technical lemma.

Lemma 11. *Let G be a connected (claw, Φ_3)-free graph such that*

- (1) G contains no K_4 ,
- (2) G contains no perfect diamond,
- (3) G contains Φ_0 ,
- (4) $3 \leq \deg(v) \leq 4$ for every vertex $v \in V(G)$,
- (5) every diamond sequence consists of at most four diamonds,

then G has at most $13 \cdot 2^{13}$ vertices.

Proof. Let G contain an induced Φ_0 with two triangles $v_0v_1u_1$ and $v_0x_1y_1$. Assume G contains a vertex v_k of distance $k \geq 14$ from v_0 and let $P = (v_k, v_{k-1}, \dots, v_2, v_1, v_0)$ be a shortest path connecting v_k to v_0 . We split the proof into two basic cases, each of which leads to a contradiction.

Case 1: v_2 is not adjacent to u_1 . Consider the vertex v_4 . Since $\deg(v_4) \geq 3$, this vertex must have at least one more neighbour, say z . Then z must be adjacent to v_3 , since otherwise z is adjacent to v_5 (to avoid a claw), in which case $v_0, u_1, v_1, v_2, v_3, v_4, v_5, z$ induce a Φ_3 . Consequently, z must be adjacent to v_2 , since otherwise $x_1, y_1, v_0, v_1, v_2, v_3, v_4, z$ induce a Φ_3 . This implies, in particular, that $\deg(v_4) = 3$, since any other neighbour w of v_4 must also be adjacent to v_3 and v_2 (for similar reasons), in which case either v_2, v_3, z, w induce a K_4 (if z is adjacent to w) or v_1, v_2, z, w induce a claw (if z is not adjacent to w). As a result, $\deg(v_3) = \deg(z) = 3$. Indeed, if v_3 has a fourth neighbour, say w , then w is not adjacent to v_4 (since $\deg(v_4) = 3$) and therefore w is adjacent to v_2 (since otherwise $G[v_3, v_4, v_2, w]$ is a claw), but then either v_2, v_3, z, w induce a K_4 (if z is adjacent to w) or v_1, v_2, z, w induce a claw (if z is not adjacent to w).

If the degree of v_3 is also 3, then the diamond induced by v_2, v_3, v_4, z is perfect, which contradicts our assumption. Therefore, we may consider a vertex w adjacent to v_2 . We know that w is not adjacent to v_3 (since $\deg(v_3) = 3$) and therefore w is adjacent to v_1 (since otherwise $G[v_1, v_2, v_3, w]$ is a claw). Finally, since $\deg(v_5) \geq 3$ we may consider a vertex s (different from v_4 and v_6) which is adjacent to v_5 . This vertex is not adjacent to v_4 (since $\deg(v_4) = 3$) and therefore it is adjacent to v_6 (since otherwise $G[v_4, v_5, v_6, s]$ is a claw). But then $w, v_1, v_2, v_3, v_4, v_5, v_6, s$ induce a Φ_3 . This completes the proof that this case leads to a contradiction.

Case 2: v_2 is adjacent to u_1 . Assume v_1 has a fourth neighbour, say z . Then z is not adjacent to v_0 (since $\deg(v_0)$ already has four neighbours) and therefore z is adjacent to v_2 (since otherwise $G[v_0, v_1, v_2, z]$ is a claw). We know that z is not adjacent to u_1 (since G is K_4 -free), which implies that z is adjacent to v_3 (since otherwise $G[u_1, v_2, v_3, z]$ is a claw). But then we can obtain from the path P a new path satisfying the condition of Case 1 by replacing the vertex v_2 by z . This argument allows us to assume that $\deg(v_1) = \deg(u_1) = 3$, i.e. the diamond induced by v_0, v_1, u_1, v_2 is pure.

Now we consider vertex v_3 and a neighbour u_3 of v_3 different from v_2 and v_4 (since $\deg(v_3) \geq 3$). We conclude that u_3 is adjacent to v_2 , since otherwise u_3 is adjacent to v_4 (to avoid the claw $G[v_2, v_3, v_4, u_3]$) in which case $x_1, y_1, v_0, v_1, v_2, v_3, v_4, u_3$ induce a Φ_3 . This implies that u_3 is adjacent to v_4 , since otherwise we fall in conditions of Case 1 with respect to the graph Φ_0 induced by vertices v_1, u_1, v_2, v_3, u_3 .

If v_3 would have one more neighbour, say w , then similarly w is adjacent to v_4 , in which case either v_3, u_3, v_4, w induce a K_4 (if u_3 is adjacent to w) or v_5, v_4, u_3, w induce a claw (if u_3 is not adjacent to w). This argument shows that the diamond induced by v_2, v_3, u_4, v_4 is pure.

Now we repeat the arguments of the last paragraph with respect to the vertex v_5 and conclude that G contains a pure diamond induced by vertices v_4, v_5, u_5, v_6 . Repeating the arguments once more we conclude that G contains a pure diamond induced by vertices v_6, v_7, u_7, v_8 . One more repetition leads to one more pure diamond and therefore to a contradiction, since G cannot contain a sequence of five pure diamonds according to our assumption. We conclude that this case also leads to a contradiction.

The above analysis shows that every vertex of G is of distance at most 13 from v_0 . Now it is not difficult to show that G has at most $13 \cdot 2^{13}$ vertices. \square

Theorem 12. *The 3-COLOURABILITY problem can be solved in the class of (claw, Φ_3)-free graphs in polynomial time.*

Proof. If a (claw, Φ_3)-free graph contains a K_4 or a vertex of degree more than 4, it is not 3-colourable. Therefore, we assume that the maximum vertex degree of an input graph is 4 and it is K_4 -free (which can obviously be verified in polynomial time). We delete vertices of low degree (1 or 2) and delete any perfect diamond (if there is any). This leaves us with a (possible disconnected) graph satisfying conditions (1), (2) and (4) of Lemma 11 and we deal with each connected component G separately.

If G is Φ_0 -free, then we apply a known algorithm [23, 35] to solve the problem for G in polynomial time. Now assume G contains a Φ_0 . As long as G has a sequence of five pure diamonds, contract the middle diamond. It is not difficult to see that this operation does not create a claw or a Φ_3 in G . This preprocessing reduces G to a graph satisfying all conditions of Lemma 11, in which case the problem becomes trivial. \square

2.4 (claw, Φ_k, Φ_{k+1})-free graphs

In this section, we show that the 3-COLOURABILITY problem is polynomial-time solvable in the class of (claw, Φ_k, Φ_{k+1})-free graphs for any fixed value of k . On the one hand, this result improves Theorem 6. On the other hand, these two results can be viewed as incomparable, since Theorem 6 proves more than just polynomial-time solvability of the problem in the class of (claw, $\Phi_k, \Phi_{k+1}, \dots$)-free graphs. It proves that graphs of bounded degree in this class have bounded tree- and clique-width. For

($\text{claw}, \Phi_k, \Phi_{k+1}$)-free graphs, we prove polynomial-time solvability only. The proof follows the same strategy as in the case of (claw, Φ_3)-free graphs. We start with a technical lemma characterizing those ($\text{claw}, \Phi_k, \Phi_{k+1}$)-free graphs that contain a Φ_0 .

Lemma 13. *Let G be a connected ($\text{claw}, \Phi_k, \Phi_{k+1}$)-free graph for some fixed value of k , such that*

- (1) G contains Φ_0 ,
- (2) $3 \leq \deg(v) \leq 4$ for every vertex $v \in V(G)$,
- (3) G contains no K_4 ,

then G has at most $2^{k+4} + 1$ vertices.

Proof. Let G contain an induced Φ_0 with two triangles $v_0v_1u_1$ and $v_0x_1y_1$. Assume G contains a vertex v_{k+4} of distance $k+4$ from v_0 and let $P = (v_{k+4}, v_{k+3}, \dots, v_2, v_1, v_0)$ be a shortest path connecting v_{k+4} to v_0 . We observe that v_2 is adjacent neither to x_1 nor to y_1 , since otherwise v_3, v_2, v_1 together with x_1 or y_1 induce a claw. We again split the proof by contradiction into two basic cases.

Case 1: v_2 is not adjacent to u_1 . Consider the vertex v_{k+2} . Since every vertex has degree at least 3, v_{k+2} must have a neighbour z . Vertex z cannot be adjacent to any vertex v_i with $i \leq k-1$, since otherwise P is not a shortest path. On the other hand, to avoid a claw, z must be adjacent to v_{k+1} , since otherwise z is adjacent to v_{k+3} , in which case $z, v_{k+3}, v_{k+2}, \dots, v_1, v_0, u_1$ induce a Φ_{k+1} . But now either the vertices $z, v_{k+2}, v_{k+1}, \dots, v_1, v_0, u_1$ induce a Φ_k (if z is not adjacent to v_k) or the vertices $z, v_{k+1}, v_k, \dots, v_1, v_0, x_1, y_1$ induce a Φ_k (if z is adjacent to v_k). Therefore, Case 1 is impossible.

Case 2: v_2 is adjacent to u_1 . Consider the vertex v_{k+3} . As before, v_{k+3} must have a neighbour z , and this neighbour cannot be adjacent to any vertex v_i with $i \leq k$. To avoid a claw, z must be adjacent to v_{k+2} , since otherwise z is adjacent to v_{k+4} , in which case $z, v_{k+4}, v_{k+3}, \dots, v_1, u_1$ induce a Φ_{k+1} . But now either the vertices $z, v_{k+3}, v_{k+2}, \dots, v_1, u_1$ induce a Φ_k (if z is not adjacent to v_{k+1}) or the vertices $z, v_{k+2}, v_{k+1}, \dots, v_1, v_0, x_1, y_1$ induce a Φ_{k+1} (if z is adjacent to v_{k+1}). Therefore, Case 2 is also impossible.

Therefore, there can be no vertex of distance more than $k+3$ from v_0 . It is not difficult to see that every vertex of distance $i > 0$ from v_0 has at most 2 neighbours of distance $i+1$, since otherwise a claw or a K_4 arises. Since v_0 has

4 neighbours, simple induction shows that the total number of vertices is at most $2^{k+4} + 1$ vertices. \square

Theorem 14. *For any fixed value of k , the 3-COLOURABILITY problem can be solved in the class of $(\text{claw}, \Phi_k, \Phi_{k+1})$ -free graphs in polynomial time.*

Proof. If a claw-free graph contains either a K_4 or a vertex of degree more than 4, then it is not 3-colourable. Therefore, we assume that the input graph is K_4 -free and has no vertices of degree more than 4. We remove vertices of degree 1 or 2, because they can easily be given colours different to their neighbours. If G is Φ_0 -free, then we apply a known algorithm to solve the problem for G in polynomial time. Otherwise, we find a Φ_0 in G . This preprocessing reduces G to a graph satisfying all conditions of Lemma 13, in which case the problem becomes trivial. \square

2.5 Conclusion

In this chapter we have identified three infinitely large sequences of graph classes, for which 3-COLOURABILITY is NP-complete, the intersection of which defines a so-called limit class. Each of these sequences may be defined by forbidding induced subgraphs from the instances of the problem, which implies a necessary condition for a polynomial-time solution to 3-COLOURABILITY in a subclass of claw-free graphs, namely to forbid a graph from each class in the sequence, or indeed to forbid a graph from the limit class itself.

We have found a number of cases where this condition is also sufficient. In particular, we have shown that the 3-COLOURABILITY problem has a polynomial-time solution in the class of (claw, Φ_k) -free graphs for the cases $k = 1, 3$.

A number of open problems are immediately suggested by these results. The complexity of 3-colourability in (claw, Φ_k) -free graphs for arbitrary values of k , and in particular for $k = 2$, remains unknown. Whether the limit class is minimal, and therefore also a boundary class, is also an interesting open problem.

Chapter 3

Dominating Induced Matchings in graphs without a skew star

3.1 Introduction

We now turn our attention to the problem of determining whether the vertices of a graph can be partitioned into two subsets B and W so that B induces a graph of vertex degree 1 (also known as an induced matching) and W induces a graph of vertex degree 0 (i.e. an independent set). Throughout the chapter we call the vertices of B black and the vertices of W white and say that a graph partitionable into an induced matching and an independent set admits a *black-white partition*. This problem appears in the literature under various names, such as EFFICIENT EDGE DOMINATION [36, 37, 38, 39, 40] or DOMINATING INDUCED MATCHING [41, 42, 43, 44], and finds applications in various fields, such as parallel resource allocation of parallel processing systems [45] and encoding theory and network routing [38]. This problem also has relations to some other algorithmic graph problems, such as 3-COLOURABILITY and MAXIMUM INDUCED MATCHING. In particular, it is not difficult to see that every graph admitting a black-white partition is 3-colourable. Also, in [37] it was shown that if a graph admits a black-white partition, then the black vertices form an induced matching of maximum size.

From an algorithmic point of view, the DOMINATING INDUCED MATCHING problem is difficult, i.e. it is NP-complete [38]. Moreover, it remains difficult under substantial restrictions. For instance, in [46] it was shown that the problem is NP-complete for cubic graphs, and in [37] this result was extended to d -regular graphs for an arbitrary $d \geq 3$. The problem was also shown to be NP-complete for bipartite graphs [40] and planar bipartite graphs [39]. The NP-completeness results

for bounded degree graphs and for bipartite graphs have been strengthened in [42] as follows.

Denote by \mathcal{S}_k the class of $(C_3, \dots, C_k, H_1, \dots, H_k)$ -free bipartite graphs of vertex degree at most 3, where C_k is a chordless cycle on k vertices and H_k is the graph represented in Figure 3.1. Associate with every graph G a parameter $\kappa(G)$, which is the maximum k such that $G \in \mathcal{S}_k$. If G belongs to no class \mathcal{S}_k , then $\kappa(G)$ is defined to be 0, and if G belongs to all classes \mathcal{S}_k , then $\kappa(G)$ is defined to be ∞ . Finally, for a set of graphs M , define $\kappa(M) = \sup\{\kappa(G) : G \in M\}$.

Theorem 15. [42] *Let M be a set of graphs and X the class of M -free bipartite graphs of vertex degree at most 3. If $\kappa(M) < \infty$, then the DOMINATING INDUCED MATCHING problem is NP-complete in the class X .*

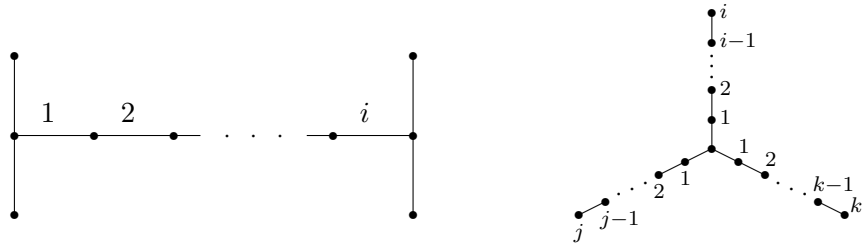


Figure 3.1: Graphs H_i (left) and $S_{i,j,k}$ (right)

This theorem is reminiscent of the hardness results of the previous chapter, and it motivates the work of this chapter in the following way. Unless $P = NP$, Theorem 15 provides a necessary condition for polynomial-time solvability of the problem in classes of graphs defined by forbidden induced subgraphs. In particular, given a set M of forbidden graphs, the problem is polynomial-time solvable in the class of M -free graphs only if $\kappa(M) = \infty$. Three basic ways to unbind the parameter κ is to include in the set M of forbidden graphs

- (1) arbitrarily large cycles,
- (2) arbitrarily large graphs of the form H_k ,
- (3) a graph G with $\kappa(G) = \infty$.

Nearly all polynomial-time results available in the literature deal with graph classes of the first type. This includes bipartite permutation graphs [40], convex graphs [44], chordal graphs [39] and hole-free graphs [36]. Nothing is known about the complexity of the problem in classes of the second type, and only two results are available for classes of the third type. By definition, $\kappa(G) = \infty$ if and only if G

belongs to all classes \mathcal{S}_k , i.e. G belongs to the intersection $\bigcap \mathcal{S}_k$ taken over all possible values of k . Let us denote this intersection by \mathcal{S} . Thus, according to Theorem 30, if M is a finite set, then the problem is polynomial-time solvable in the class of M -free graphs only if M contains a graph from \mathcal{S} . We believe that the converse is also true and formally state this as a conjecture.

Conjecture 16. *Let M be a finite set of graphs. Unless $P = NP$, the DOMINATING INDUCED MATCHING problem is polynomial-time solvable in the class of M -free graphs if and only if M contains a graph from \mathcal{S} .*

Clearly, to prove the conjecture it is sufficient to consider finite sets M consisting of a single graph G that belongs to \mathcal{S} . It is not difficult to see that $G \in \mathcal{S}$ if and only if every connected component of G is of the form $S_{i,j,k}$ (see Figure 3.1). The smallest non-trivial graph of this form is a claw.

We have alluded to the attention that claw-free graphs have recently received in the literature [25, 26, 27, 28]. In particular, in [28] Minty develops a polynomial-time algorithm for the MAXIMUM INDEPENDENT SET problem in claw-free graphs, which extends the celebrated solution for the MAXIMUM MATCHING problem due to Edmonds [47]. Recently, this solution was further extended to the class of $S_{1,1,2}$ -free graphs [48]. We will return to this problem in the next chapter.

The class of claw-free graphs is also easy for the DOMINATING INDUCED MATCHING problem [43], which is one of the two polynomially solvable cases of type 3 for the problem. The second solvable case deals with P_7 -free graphs, as was recently proved in [41] (note that $P_7 = S_{0,3,3}$). In the present chapter, we extend the solution for claw-free graphs first to $S_{1,2,2}$ -free graphs and then to $S_{1,2,3}$ -free graphs. Throughout the chapter we call $S_{1,2,3}$ a skew star.

The organization of the chapter is as follows. In the next section we introduce basic terminology, and begin to outline our solution strategy by describing a number of useful reductions that help solve the problem. In Section 3.3 we solve the problem for $S_{1,2,2}$ -free graphs. A solution in this class first appeared in [3], and here we correct a mistake in the original proof and make further improvements to the proof. In Section 3.4 this solution is extended to skew star-free graphs.

3.2 Preliminaries

We view the DOMINATING INDUCED MATCHING problem as the problem of colouring the vertices of a graph with two colours, black and white, so that no white vertex has a white neighbour and every black vertex has exactly one black neighbour. Assigning

one of the two possible colours to the vertices of G will be called *colouring* of G . A colouring is *partial* if only part of the vertices of G are assigned colours, otherwise it is *total*. In a partial colouring, a black vertex that has a black neighbour is called *matched*. A partial colouring is *valid* if

- no two white vertices are adjacent,
- no black vertex has more than one black neighbour,
- every unmatched black vertex has at least one uncoloured neighbour.

A total colouring is *valid* if no two white vertices are adjacent and every black vertex has exactly one black neighbour.

Our strategy in solving the problem is to incrementally extend a partial valid colouring according to certain rules. This strategy suggests a more general framework for the problem, in which the graph is given together with a partial valid colouring. The question is to determine if the partial colouring can be extended to a total valid colouring. We will refer to this more general version of the problem as EXTENSION TO DOMINATING INDUCED MATCHING (EDIM for short).

Among the various rules used in our algorithm, the following three are obvious:

R1: each neighbour of a white vertex must be coloured black;

R2: each neighbour of a matched black vertex must be coloured white;

R3: each vertex that has two black neighbours must be coloured white.

Two other rules that will be used in the chapter are not so obvious but also are simple:

R4: if a vertex v belongs to a triangle T and has a neighbour w outside T , then v and w must be coloured differently;

R5: in any induced C_4 , any two adjacent vertices must be coloured differently.

Given a graph G and a partial colouring of its vertices, we can obviously ignore those coloured vertices that have no neighbours among uncoloured ones. We shall call such vertices *irrelevant*. Removing irrelevant vertices from the graph can reduce the problem to a more specific instance. In particular, we recall that the diamond and butterfly graphs are those depicted in Figure 3.2, and give the following reduction which is valid for arbitrary graphs.

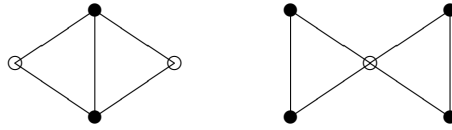


Figure 3.2: A diamond (left) and a butterfly (right).

Lemma 17. *The EDIM problem can be reduced in polynomial time from an arbitrary graph G to an induced subgraph G' of G such that G' is (diamond, butterfly, K_4)-free and every vertex of G' has at most one neighbour of degree 1.*

Proof. Since K_4 is not 3-colourable, no graph G containing a K_4 has a black-white partition. This immediately reduces the problem from general graphs to K_4 -free graphs. Also, by direct inspection, the reader can easily check that the diamond and butterfly have unique valid colouring represented in Figure 3.2. Therefore, if a graph G contains a copy of an induced diamond or butterfly, the vertices of this copy can be coloured and removed from the graph, since they become irrelevant after colouring all their neighbours.

Finally, assume that G contains a vertex that has more than one neighbour of degree 1. If G admits a black-white partition, then all these neighbours, except possibly one, are white. Moreover, if one of these neighbours must be black, then *anyone* of them can be assigned this colour. Therefore, all but one neighbour of degree 1 can be coloured white and removed from the graph. \square

A useful characterisation of (diamond, butterfly, K_4)-free graphs is provided by the following lemma.

Lemma 18. *Let G be a (diamond, butterfly, K_4)-free graph and v a vertex of G . Then the neighbourhood of v contains at most one edge.*

Proof. Assume $N(v)$ contains two edges e_1 and e_2 . If these edges share a vertex, then their endpoints together with v induce either a *diamond* or a K_4 . If neither e_1, e_2 nor any other two edges in the neighbourhood of v share a vertex, then the endpoints of e_1 and e_2 together with v induce a butterfly. \square

3.3 Solution to the problem in the class of $S_{1,2,2}$ -free graphs

The solution for $S_{1,2,2}$ -free graphs is based on a reduction of the problem to graphs of bounded clique-width. The reduction consists of two steps. In the first step, we reduce the problem from the entire class of $S_{1,2,2}$ -free graphs to graphs of bounded vertex degree in this class. In the second step, we further reduce the problem to graphs of bounded chordality, i.e. graphs without long induced cycles. Together, bounded vertex degree and bounded chordality imply bounded clique-width. The polynomial-time solvability of the problem on graphs of bounded clique-width is due to the following lemma.

Lemma 19. *The EDIM problem can be solved in polynomial time in any class of graphs where clique-width is bounded by a constant.*

Proof. In [33], it was shown that any decision problem expressible in $\text{MSOL}(\tau_1, p)$ can be solved in linear time in any class of graphs of bounded clique-width. $\text{MSOL}(\tau_1)$ is a Monadic Second-Order Logic with quantification over subsets of vertices, but not of edges. $\text{MSOL}(\tau_1, p)$ is the extension of $\text{MSOL}(\tau_1)$ by unary predicates representing labels attached to vertices. It is known that DOMINATING INDUCED MATCHING is expressible in $\text{MSOL}(\tau_1)$ [43], and therefore, EXTENSION TO DOMINATING INDUCED MATCHING is expressible in $\text{MSOL}(\tau_1, p)$. \square

3.3.1 Reduction to graphs of bounded vertex degree

In the first step of our solution, we reduce the problem from the entire class of $S_{1,2,2}$ -free graphs to graphs of bounded vertex degree in this class. This step is valid even for the larger class of $S_{2,2,2}$ -free graphs.

Lemma 20. *The EDIM problem in the class of $S_{2,2,2}$ -free graphs can be reduced in polynomial time to graphs of vertex degree of at most 6 in this class.*

Proof. Let G be an $S_{2,2,2}$ -free graph. According to Lemma 17, we may assume without loss of generality that G is (diamond, butterfly, K_4)-free and every vertex of G has at most one neighbour of degree 1. Suppose G has a vertex v of degree at least 7.

Assume G admits a black-white partition in which v is coloured white. Then every neighbour of v is coloured black. By Lemma 18, the neighbourhood of v contains at most one edge. Therefore, v has at least 3 neighbours which are isolated in the subgraph of G induced by $N(v)$. Moreover, each of these three vertices

must have its own black neighbour. But then G contains an induced $S_{2,2,2}$. This contradiction shows that in any black-white partition of G (if there is any) all vertices of degree at least 7 must be coloured black.

From now on we assume that v is coloured black. If two non-adjacent neighbours of v , say x and y , have another common neighbour, say z , then v, x, y , and z form an induced C_4 , in which case both x and y must be coloured white (Rule *R5*) and can be removed from G . This reduces the problem to the case where no two non-adjacent neighbours of v have another common neighbour.

If the neighbourhood of v contains an edge, say a, b , then every vertex of $N(v) \setminus \{a, b\}$ must be coloured white (Rule *R4*), which reduces the problem to the case where the degree of v is 2.

Assume now that $N(v)$ is an independent set, and since v has at most one neighbour of degree 1, v has at least six neighbours each of which has a private neighbour different from v . Let Q be a set of six private neighbours of six vertices in $N(v)$. By Ramsey's Theorem, any set of six vertices contains either a triangle or its complement. If Q contains the complement of a triangle, then clearly G contains an induced $S_{2,2,2}$, which is impossible. Therefore, Q must contain a triangle T . Let T' be the set of three vertices of $N(v)$ adjacent (pairwise privately) to the vertices of T . If the graph admits a black-white partition, then exactly one vertex of T must be coloured white, and therefore, exactly one vertex of T' must be coloured black. Therefore, v cannot have black neighbours outside of T' and hence all vertices outside of T' must be coloured white. This reduces the problem to the case where the degree of v is at most 4.

The above discussion provides a reduction from general $S_{2,2,2}$ -free graphs to graphs of maximum vertex degree at most 6 in this class. That this reduction can be implemented in polynomial time is obvious. \square

3.3.2 Reduction to graphs of bounded chordality

The next lemma implements the second step in our solution.

Lemma 21. *The EDIM problem in the class of $S_{1,2,2}$ -free graphs of vertex degree at most 6 can be reduced in polynomial time to $(C_9, C_{10}, C_{11}, \dots)$ -free graphs in this class.*

Proof. Let G be a connected $S_{1,2,2}$ -free graph of vertex degree at most 6. By Lemma 17, we also assume that G is (diamond, butterfly, K_4)-free. Suppose G contains a chordless cycle $C = (1, 2, 3, \dots, k-1, k)$ of length $k \geq 9$. If G coincides with C , then the problem is trivial. Otherwise, G contains a vertex v outside

C which has at least one neighbour on C . Keeping in mind that the graph is (diamond,butterfly)-free, we conclude that v has at most 3 neighbours on the cycle, since otherwise v is the centre of an induced $S_{1,2,2}$. Also if v has exactly one neighbour on C , or two neighbours of minimum distance at least 3 along the cycle, or three neighbours, then there is an induced $S_{1,2,2}$ centred at a neighbour of v on C . From the above discussion it follows that v has exactly two neighbours on C , either $i, i + 2$ or $i, i + 1$. Let us show that

R6: if v is adjacent to i and $i + 2$, then v and $i + 1$ must be coloured white.

Indeed, since $v, i, i + 1, i + 2$ create a C_4 , vertices v and $i + 1$ must have the same colour (Rule *R5*). Assume v and $i + 1$ are coloured black, which implies $i, i + 2$ are white and therefore $i - 1, i + 3$ are black. If the graph admits a black-white partition, v has a black neighbour, say w . If w is adjacent neither to i nor to $i + 2$ then $G[i - 1, i, i + 2, i + 3, v, w] = S_{1,2,2}$, and if w is adjacent both to i and to $i + 2$ then $G[i, i + 2, v, w] = \text{diamond}$. Therefore, w has exactly one neighbour in $\{i, i + 2\}$, say i . Observe that replacing $i + 1$ by v creates another cycle C' of length k , and from the above discussion we know that w cannot have more than two neighbours on C' . Therefore, w is not adjacent to $i - 2$. But then $G[i - 2, i - 1, i, i + 1, i + 2, w] = S_{1,2,2}$. This contradiction proves validity of Rule *R6*.

Applying Rule *R6* as long as possible and removing irrelevant vertices from the graph leaves us with the case when every vertex outside C that has a neighbour on C is adjacent to exactly two consecutive vertices of C . Also, since the graph is (K_4 , diamond, butterfly)-free, we conclude that every vertex of C that has a neighbour outside C is adjacent to exactly one vertex outside C . Moreover, the problem can be further reduced to the case when *every* vertex of C has a neighbour outside C . This can be done according to the following rules. Assume $i, i + 1, \dots, i + p, i + p + 1$ is a list of consecutive vertices on C such that i and $i + p + 1$ have neighbours outside C , while $i + 1, \dots, i + p$ have no neighbours outside C .

R7: If $p = 1$, then $i + 1$ must be coloured white.

Indeed, if $i + 1$ is black, then, by Rule *R4*, i and $i + 2$ are white. But then black vertex $i + 1$ has no black neighbours in G . Therefore, $i + 1$ must be coloured white.

R8: If $p = 2$, then $i, i + 3$ must be coloured white and $i + 1, i + 2$ must be coloured black.

Indeed, if $i + 1$ is white, then $i + 2$ is black (Rule $R1$) and therefore $i + 3$ is white (Rule $R4$). But then black vertex $i + 2$ has no black neighbours in G . This contradiction shows that $i + 1$ must be coloured black. Symmetrically, $i + 2$ must be coloured black. This implies that $i, i + 3$ must be coloured white.

$R9$: If $p \geq 3$, then replacing the path $i, i + 1, i + 2, i + 3, i + 4$ by an edge $(i, i + 4)$ transforms G into an $S_{1,2,2}$ -free graph G' which has a black-white partition if and only if G has.

To see this, assume first that G has a black-white partition. We know that i is adjacent to a vertex outside C , while $i + 1$ is not, i.e. there is a triangle containing i but not $i + 1$. Therefore, by Rule $R4$, i and $i + 1$ must be coloured differently. Suppose i is black, then $i + 1$ is white, implying that $i + 2$ and $i + 3$ are black and $i + 4$ is white. Therefore, by deleting from G the vertices $i + 1, i + 2, i + 3$ and connecting i to $i + 4$ we obtain a graph G' which also has a black-white partition. If i is white, then $i + 1$ and $i + 2$ are black, $i + 3$ is white and $i + 4$ is black, and again G' has a black-white partition. The converse statement (that a black-white partition of G' implies a black-white partition of G) can be shown by analogy.

We can safely apply this transformation if none of the vertices $i + 1, i + 2, i + 3$ is pre-coloured black or if $i + 2$ is the only vertex among these three which is pre-coloured black. Let us show that we can always assume this. Indeed, if $i + 1$ has been coloured black in a previous stage of the algorithm, then by Rule $R4$ vertex i must be coloured white and hence can be removed from the graph. Also, if $i + 3$ is pre-coloured black, then i must also be coloured black (else $i + 1$ is black by Rule $R4$ and $i + 2$ is white by Rule $R3$, in which case $i + 1$ cannot be matched) and hence $i + 1$ is white (Rule $R4$) and $i + 2$ is black (Rule $R1$), in which case vertices $i + 1, i + 2, i + 3$ can be removed from the graph.

Finally, let us show that G' is $S_{1,2,2}$ -free. Indeed, G can be obtained from G' by subdividing an edge. Therefore, if G contains an $S_{1,2,2}$ then so does G' , which is impossible.

Applying rules $R7, R8, R9$ as long as possible and removing irrelevant vertices from the graph reduces the problem to the case when every vertex outside C with a neighbour on C is adjacent to exactly two consecutive vertices of C , and every vertex of C has exactly one neighbour outside C , i.e. C is of even length. Moreover, without loss of generality, every even edge belongs to a triangle and every odd edge

does not belong to any triangle. By Rule *R4*, the endpoints of odd edges must be coloured differently, which in turn implies that the endpoints of even edges must be coloured differently. In other words, the colours of the vertices alternate along the cycle, while all its neighbours outside the cycle are black. This means that we can choose arbitrarily one of the two possible ways to colour the vertices of the cycle. By colouring, for instance, the odd vertices of C white and removing them from the graph, and repeating this procedure for each cycle of length at least 9, we reduce the problem to graphs without long induced cycles.

Finding an induced cycle of length at least 9 can be done in $O(n^9)$ time. All other operations of the reduction can also be implemented in polynomial time. \square

We now summarize the above discussion in the following conclusion.

Theorem 22. *The (EXTENSION TO) DOMINATING INDUCED MATCHING PROBLEM can be solved in the class of $S_{1,2,2}$ -free graphs in polynomial time.*

Proof. By Lemmas 20 and 21, the EDIM problem can be reduced from $S_{1,2,2}$ -free graphs to graphs of degree at most 6 and of chordality (the length of a longest induced cycle) at most 8. It has been shown in [32] that if a graph has chordality at most c and maximum degree at most k , then its tree-width is at most $k(k-1)^{c-3}$. Also, in [49] it was shown that for any graph G , the clique-width of G does not exceed $3 \cdot 2^{\text{tw}(G)-1}$, where $\text{tw}(G)$ denotes the tree-width of G . Therefore, Lemmas 20 and 21 reduce the problem from $S_{1,2,2}$ -free graphs to graphs of bounded clique-width. Together with Theorem 19 this implies a polynomial-time solution to the problem in the class of $S_{1,2,2}$ -free graphs. \square

3.4 Solution to the problem in the class of $S_{1,2,3}$ -free graphs

We solve the problem for $S_{1,2,3}$ -free graphs by reducing it to $S_{1,2,2}$ -free graphs. Let G be an $S_{1,2,3}$ -free graph. We may also assume that G is (diamond, butterfly, K_4)-free by Lemma 17.

If G has no induced copy of $S_{1,2,2}$, then the problem can be solved for G in polynomial time by Theorem 22. If G contains an induced copy of $S_{1,2,2}$, we want to destroy it by colouring at least one of its vertices white (or two adjacent vertices black, in which case a white vertex necessarily appear according to Rule *R2*).

Destroying a single copy of an $S_{1,2,2}$ is a simple task, since there are only finitely many ways to colour the vertices of this copy. The difficulty is that G may

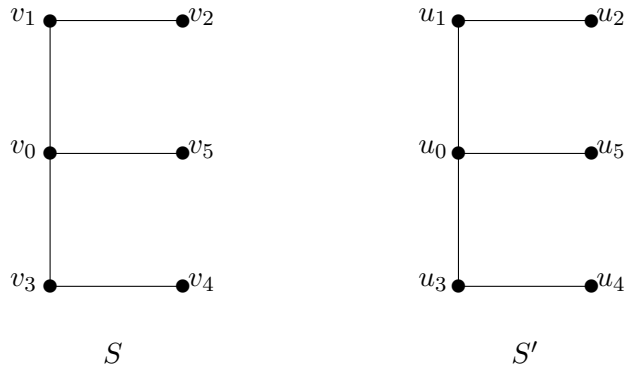


Figure 3.3: Two copies of an induced $S_{1,2,2}$ with their labellings

have many copies of an $S_{1,2,2}$. We will show that by colouring the vertices of one copy of $S_{1,2,2}$ and then by propagating the colours according to the rules of Section 6.2, we either destroy all other copies of $S_{1,2,2}$ or leave only one way to colour the vertices of those copies that have survived. To this end, we fix an induced copy of $S_{1,2,2}$ and denote it by S . Assume that there is one more induced copy of $S_{1,2,2}$. We denote the second copy by S' and we denote the vertices of S and S' as shown in Figure 3.3.

First of all, let us show that S and S' cannot be far from each other, since otherwise an induced skew star arises.

Lemma 23. *The distance between S and S' is at most 2.*

Proof. Assume to the contrary that the distance between S and S' is at least 3. Then the shortest path connecting S to S' contains at least two vertices. We denote by x the vertex of the path that has a neighbour in S' , by y the neighbour of x on the path and by z the neighbour of y non-adjacent to x . Note that z may or may not belong to S . But in either case, z must have a neighbour t non-adjacent both to y and x , since otherwise a diamond arises. We split the proof into cases depending on the number of neighbours of x in S' and show that in all possible cases the graph contains a skew star. Note that x can have at most four neighbours in S' , since otherwise a *diamond* or a *butterfly* appears.

Case 1: x has one neighbour in S' . Up to symmetry, the following subcases are possible.

- 1.1: If x is adjacent to u_0 , then $u_0, u_1, u_3, u_4, x, y, z$ induce a skew star.
- 1.2: If x is adjacent to u_1 , then $u_0, u_5, u_3, u_4, u_1, x, y$ induce a skew star.
- 1.3: If x is adjacent to u_2 , then $u_0, u_5, u_3, u_4, u_1, u_2, x$ induce a skew star.

Case 2: x has two neighbours in S' . Up to symmetry, the following subcases are possible.

- 2.1: If x is adjacent to u_0, u_1 or to u_0, u_2 , then $u_0, u_5, u_3, u_4, x, y, s$ induce a skew star.
- 2.2: If x is adjacent to u_0, u_5 , then $u_0, u_1, u_3, u_4, x, y, s$ induce a skew star.
- 2.3: If x is adjacent to u_1, u_2 , then $u_0, u_5, u_3, u_4, u_1, x, y$ induce a skew star.
- 2.4: If x is adjacent to u_1, u_3 or to u_1, u_4 , then $u_1, u_2, u_0, u_5, x, y, s$ induce a skew star.
- 2.5: If x is adjacent to u_1, u_5 , then $u_1, u_2, u_0, u_3, x, y, s$ induce a skew star.
- 2.6: If x is adjacent to u_2, u_4 , then $x, u_2, y, s, u_4, u_3, u_0$ induce a skew star.
- 2.7: If x is adjacent to u_2, u_5 , then $x, u_2, y, s, u_5, u_0, u_3$ induce a skew star.

Case 3: x has three neighbours in S' . Note that x cannot have three consecutive neighbours in S' , since otherwise an induced diamond appears. Up to symmetry, the following subcases are possible.

- 3.1: If x is adjacent to u_0, u_1, u_4 or to u_0, u_2, u_4 , then $x, u_4, u_0, u_5, y, s, t$ induce a skew star.
- 3.2: If x is adjacent to u_0, u_2, u_5 , then $u_0, u_1, u_3, u_4, x, y, s$ induce a skew star.
- 3.3: If x is adjacent to u_1, u_2, u_3 , then $x, u_2, y, s, u_3, u_0, u_5$ induce a skew star.
- 3.4: If x is adjacent to u_1, u_2, u_4 , then $x, u_2, y, s, u_4, u_3, u_0$ induce a skew star.
- 3.5: If x is adjacent to u_1, u_2, u_5 , then $x, u_2, y, s, u_5, u_0, u_3$ induce a skew star.
- 3.6: If x is adjacent to u_1, u_3, u_5 or to u_1, u_4, u_5 , then $x, u_5, u_1, u_2, y, s, t$ induce a skew star.
- 3.7: If x is adjacent to u_2, u_4, u_5 , then $x, u_5, u_2, u_1, y, s, t$ induce a skew star.

Case 4: x has four neighbours in S' . Note that x can have neither three consecutive neighbours in S' (else an induced diamond appears) nor two pairs of adjacent neighbours (else an induced butterfly appears). Up to symmetry, the following subcases are possible.

4.1: If x is adjacent to u_0, u_2, u_4, u_5 , then $x, u_2, u_4, u_3, y, s, t$ induce a skew star.

4.2: If x is adjacent to u_1, u_2, u_3, u_5 or to u_1, u_2, u_4, u_5 , then $x, u_2, u_5, u_0, y, s, t$ induce a skew star.

□

Now we fix a colouring of S and propagate it as long as possible. We want to show that this operation either destroys S' (i.e. leads to a white vertex or two adjacent black vertices in S') or forces the central vertex of S' to be black.

Lemma 24. *If the shortest distance between S and S' is 2, then S' is destroyed.*

Proof. Let x be a vertex that has a neighbour v in S and a neighbour u in S' . It is not difficult to see that x must have at least two neighbours in each copy of $S_{1,2,2}$, since otherwise an induced $S_{1,2,3}$ can be easily found similarly to Case 1 of Lemma 23. This tells us that we always know the colour of x .

If x has both a black neighbour and a white neighbour in S , then x must be coloured black and its neighbour in S' white, in which case we are done. If x is adjacent to two vertices in S of the same colour, it must take the opposite colour.

Since the graph is diamond-free, there must exist three vertices a, b, c in S such that x, a, b, c induce either a C_4 or a P_4 and three vertices a', b', c' in S' such that x, a', b', c' induce either a C_4 or a P_4 . If x, a', b', c' induce a C_4 , then at least one of a', b', c' must be white and we are done. So, assume x, a', b', c' induce a P_4 .

Case 1: Suppose x, a, b, c induce a C_4 . Without loss of generality we may assume that c has another neighbour, say d , in S . If x is not adjacent to d , then x, a, c, d, a', b', c' induce a skew star, and if x is adjacent to d , then x, d, a, b, a', b', c' induce a skew star, a contradiction in both cases.

Case 2: Suppose x, a, b, c induce a P_4 , and let u be the second neighbour of x in S . Then u is not adjacent to b , since otherwise x, a, b, u induce a C_4 (Case 1). If, in addition, u is not adjacent to a , then x, u, a, b, a', b', c' induce a skew star. So, we assume that u is adjacent to a . Then either a or b has degree 3 in S . If a has degree 3 with v being its third neighbour in S , then x is not adjacent to v (else x, a, u, v induce a diamond) and hence a, v, b, c, a', b', c' induce a skew star. Let b have degree 3 in S , i.e. $b = v_0$ and without loss of generality $a = v_1$. Then x is not adjacent to v_5 , else x, v_1, v_0, v_5 induce a C_4 . Therefore, if x is not adjacent to v_4 , then $v_0, v_5, v_3, v_4, v_1, x, a'$ induce a skew star, and if x is adjacent to v_4 , then $x, v_2, v_4, v_3, v_0, a', b'$ induce a skew star. A contradiction in both cases completes the proof. □

The above lemma reduces the analysis to the case when the distance between S and S' is at most one. To analyse this case, we distinguish between two types of colourings of S which we call alternating and non-alternating.

Definition 25. *A colouring of S is alternating if the even-indexed vertices are black and the odd-indexed vertices are white or the even-indexed vertices are white and the odd-indexed vertices are black.*

Lemma 26. *If the shortest distance between S and S' is 1 and the colouring of S is not alternating, then S' is destroyed.*

Proof. It is not difficult to see that if the colouring of S is not alternating, then it must have two adjacent black vertices. We denote these vertices by x and y . We may also assume that y has a white neighbour $z \in S$ (non-adjacent to x).

If x or y has a neighbour in S' , then this neighbour must be white and we are done. Therefore, in what follows we assume that neither x nor y has a neighbour in S' .

Suppose z is adjacent to $u_1 \in S'$. Then z must have at least one neighbour in $\{u_0, u_2, u_3\}$, since otherwise $u_1, u_2, u_0, u_3, z, y, x$ induce a skew star. If z is adjacent to u_0 or u_2 , then either u_0, u_1 or u_1, u_2 are two adjacent black vertices in S' . If z is adjacent neither to u_0 nor to u_2 , but is adjacent to u_3 , then u_0 must be white (since z, u_1, u_0, u_3 induce a C_4). This proves the lemma in the case when z is adjacent to u_1 . If z is adjacent to u_0 or u_3 , the proof is similar.

Assume now that z has no neighbours in $\{u_1, u_0, u_3\}$ and suppose z is adjacent to u_2 . Then either the vertices $u_0, u_5, u_3, u_4, u_1, u_2, z$ (if z has no neighbours in $\{u_4, u_5\}$) or the vertices $z, u_2, y, x, a, u_3, u_0$ (if z has a neighbour $a \in \{u_4, u_5\}$) induce a skew star. This contradiction shows that z cannot be adjacent to u_2 . By symmetry it cannot be adjacent to u_4 . This also implies that z is not adjacent to u_5 , since otherwise an induced $S_{1,2,3}$ can be easily found.

The above analysis shows that if S has two adjacent black vertices, then we may assume that neither these vertices nor any of its neighbours in S have neighbours in S' . To complete the analysis, we distinguish between the following two cases.

1. Assume first that v_0 is black. Then, to avoid an alternating colouring of S , we conclude that (up to symmetry) either v_1 is black or v_5 is black.
 - If v_1 is black, then according to the above assumption no vertex of S except for v_4 has a neighbour in S' . If v_4 has a neighbour in S' , then this neighbour together with the vertices of S induce a skew star, and if v_4

has no neighbour in S' , then the distance between S and S' more than 1, contradicting the assumption.

- If v_5 is black, then vertices v_0, v_1, v_3, v_5 have no neighbours in S' , and vertices v_2 and v_4 are black. Without loss of generality suppose that v_2 has a neighbour a in S' . Then a is also adjacent to v_4 , since otherwise $v_0, v_5, v_3, v_4, v_1, v_2, a$ induce a skew star. But then a is white, as being adjacent to two black vertices v_2 and v_4 .

2. Suppose now that v_0 is white. Then v_1, v_3, v_5 are black and, without loss of generality, v_2 is black (to avoid an alternating colouring of S). Therefore, we assume that v_0, v_1, v_2 have no neighbours in S' .

- If v_3 and v_5 have no neighbours in S' , then v_4 must have a neighbour in S' , in which case this neighbour together with the vertices of S induce a skew star.
- Suppose now that v_3 has a neighbour a in S' . We may also assume that a has a neighbour $b \in S'$ non-adjacent to v_3 . If v_4 is black, then a is white and we are done. If v_4 is white, then it must have a neighbour in $\{a, b\}$, since otherwise $v_3, v_4, a, b, v_0, v_1, v_2$ induce a skew star. If v_4 is adjacent to a , then a is black and hence b is white, and if v_4 is adjacent to b , then v_4, v_3, a, b induce a C_4 and hence a is white.
- If v_3 has no neighbours in S' , while v_5 has, we may assume, as before, that v_5 is adjacent to a vertex $a \in S'$ and non-adjacent to a neighbour $b \in S'$ of a . Then $v_0, v_3, v_1, v_2, v_5, a, b$ induce a skew star.

□

Lemma 27. *If the shortest distance between S and S' is 1 and the colouring of S is alternating, then either S' is destroyed or u_0 is coloured black.*

Proof. Suppose, for a contradiction, that for an alternating colourings of S , neither u_0 is black nor S' is destroyed. Under this assumption we can further assume that a vertex v of S'

- (1) has neighbours of at most one colour, since otherwise v is necessarily black, in which case every neighbour of v in S' is white and hence S' is destroyed,
- (2) has at most one black neighbour in S , since otherwise v is white and hence S' is destroyed.

The following observation also will be helpful in the proof.

- (*) If a vertex $v \in S$ has a neighbour in S' , then there are vertices $a, b, c \in S'$ inducing a path P_3 such that v is adjacent to a and non-adjacent to b, c . Indeed, if v has a neighbour in S' , then, assuming that the input graph is diamond-free, we can always find a neighbour $a \in S'$ of v which can be extended to an induced $P_3 = (a, b, c)$ such that v is not adjacent to b . Then v is also not adjacent to c , since otherwise vertices v, a, b, c induce a C_4 , in which case the colour of v completely defines the colours of a, b, c and at least one of these three vertices is white (and hence S' is destroyed).

Consider first the case that the even vertices of S are coloured black. Then v_2 cannot have a neighbour a in S' , since otherwise, according to (1) and (2), vertices $v_0, v_5, v_3, v_4, v_1, v_2, a$ induce a skew star. Symmetrically, v_4 can have no neighbours in S' .

Suppose that v_1 has a neighbour a in S' . Since v_1 is white, we conclude that a is black and hence $a \neq u_0$, or else we are done. Since a is not the central vertex of S' , there must exist vertices b, c, d such that a, b, c, d induce a P_4 . Moreover, we can assume that

- (3) b has no neighbours in S . Indeed, if b has a black neighbour x in S , then b must be coloured white, as a vertex with two black neighbours x and a , in which case S' is destroyed, and if b has a white neighbour in S , then b must be coloured black, in which case S is destroyed as well, as having two matched black vertices a and b .
- (4) c has no white neighbour in S , since otherwise c is black and hence b is white (as a vertex with two black neighbours a and c), in which case S' is destroyed.

This implies that

- (5) a is adjacent to v_3 , since otherwise vertices $v_1, v_2, a, b, v_0, v_3, v_4$ induce a skew star (according to (1)),
- (6) d has a neighbour in $\{v_1, v_3\}$, since otherwise both $a, v_1, v_3, v_4, b, c, d$ and $a, v_3, v_1, v_2, b, c, d$ induce skew stars, and hence d is black,
- (7) c has no neighbours in S by symmetry with b ,
- (8) a is adjacent to v_5 , since otherwise both $v_1, v_2, v_0, v_5, a, b, c$ and $v_3, v_4, v_0, v_5, a, b, c$ induce skew stars.
- (9) d is adjacent to all three vertices v_1, v_3, v_5 by symmetry with a .

Clearly, either b or c is the central vertex of S' , and hence one of them, say c , has one more neighbour in S' , say e . Vertex e cannot have a white neighbour in S , since otherwise e is black and hence c is white, which destroys S' . But then vertices $a, v_5, v_1, v_2, b, c, e$ induce a skew star. This contradiction shows that v_1 has no neighbour in S' . By symmetry, v_3 has no neighbour in S' .

Suppose now that v_5 has a neighbour a in S' . According to the previous discussion, v_5 is the only neighbour of a in S . Consider a vertex $b \in S'$ adjacent to a . By analogy with (3), vertex b has no neighbour in S . But now a skew star can be easily found.

Finally for this colouring, suppose that v_0 has a neighbour S' . Then, according to (*), there are vertices $a, b, c \in S'$ inducing a path P_3 such that v_0 is adjacent to a and non-adjacent to b, c . But then vertices $v_0, v_1, v_3, v_4, a, b, c$ induce a skew star. This contradiction shows that v_0 has no neighbours in S' and completes the proof of the lemma for the alternating colouring of S when the even vertices of S are coloured black.

We now consider the case that the even vertices of S are coloured white. Suppose first that v_1 has a neighbour S' . Then, according to (*), there are vertices $a, b, c \in S'$ inducing a path P_3 such that v_1 is adjacent to a and non-adjacent to b, c . By (1) and (2), v_1 is the only neighbour of a in S . Also, b has no white neighbours in S , since otherwise b is black and hence a is white (as a vertex with two black neighbours v_1 and b), in which case S' is destroyed. Therefore, to avoid a skew star induced by vertices $v_1, v_2, a, b, v_0, v_3, v_4$, we conclude that b is adjacent to v_3 . Then c has no white neighbours in S , since otherwise c is black and hence b is white. But then a skew star arises induced either by $v_1, v_2, v_0, v_5, a, b, c$ (if c is not adjacent to v_5) or by $v_3, v_4, b, c, v_0, v_1, v_2$ (if c is adjacent to v_5 and hence non-adjacent to v_3 by (2)). This contradiction shows that v_1 , and symmetrically v_3 , have no neighbours in S' .

Assume v_5 has a neighbour a in S' . Then by (*) we can assume that there is a vertex $b \in S'$ adjacent to a and non-adjacent to v_5 . Similarly to the previous case, v_5 is the only neighbour of a in S , and b has no white neighbours in S . But now vertices $v_0, v_1, v_3, v_4, v_5, a, b$ induce a skew star. This contradiction shows that v_5 has no neighbours in S' .

Suppose that v_2 has a neighbour a in S' . Since a is black, it is not the central vertex of S' and hence a can be extended to an induced $P_4 = (a, b, c, d)$. We may assume that b, c have no neighbours in S , because if b has a (white) neighbour, then b is black and hence c is white (since a and b form a pair of black matched vertices), and if c has a (white) neighbour, then c is black and hence b is white (as a vertex

with two black neighbours a and c). Now we see that a must be adjacent to v_4 , since otherwise a skew star arises induced either by $v_0, v_5, v_3, v_4, v_1, v_2, a$ (if a is not adjacent to v_0) or by $v_0, v_5, v_3, v_4, a, b, c$ (if a is adjacent to v_0). Then d must have a neighbour in $\{v_2, v_4\}$ (since otherwise vertices $a, v_4, v_2, v_1, b, c, d$ induce a skew star) and hence d is black. Clearly, either b or c is the central vertex of S' , and hence one of them, say c , has one more neighbour in S' , say e . Vertex e cannot have a white neighbour in S , since otherwise e is black and hence c is white (as a vertex with two black neighbours e and d), which destroys S' . But then vertices $a, v_2, v_4, v_3, b, c, e$ induced a skew star. This contradiction shows that v_2 , and symmetrically v_4 , have no neighbours in S' .

Finally, suppose that v_0 has a neighbour S' . Then, according to (*), there are vertices $a, b, c \in S'$ inducing a path P_3 such that v_0 is adjacent to a and non-adjacent to b, c . But then vertices $v_0, v_1, v_3, v_4, a, b, c$ induce a skew star. This contradiction shows that v_0 has no neighbours in S' and completes the proof of the lemma. \square

Lemma 28. *If S and S' share a vertex, then either S' is destroyed or u_0 is coloured black, in which case the colouring of S is alternating.*

Proof. Assume for a contradiction that S and S' share at least one vertex, but neither S' is destroyed nor u_0 is coloured black. Under this assumption, we conclude that

- (1) every vertex common for S and S' is black.
- (2) S and S' do not share two adjacent vertices, since regardless of their colours these two vertices necessarily destroy S' .
- (3) S and S' do not share two vertices that are of distance 2 in S' . Indeed, if vertices x, y, z create a P_3 in S' and x, z belong to S , then x and z are black by (1) and hence y is white, which destroys S' .

Let v be a vertex common for S and S' . Observe that v has a neighbour in S and a neighbour in S' , but none of these neighbours is shared both by S and by S' (according to (2)). We also claim that

- (4) every neighbour z of v in S must be white, because if z is black, then every neighbour of v in S' is white (as a vertex adjacent to a matched black vertex), in which case S' is destroyed.
- (5) every neighbour of v in S' has no other neighbours in S . Indeed, if a neighbour $x \in S'$ of v is adjacent to a white vertex of S , then x is black and the pair v, x

of two black matched vertices destroys S' , and if x is adjacent to one more black vertex of S , then x must be coloured white, which destroys S' as well.

Assume first that $v = v_2$ and x is a neighbour of v_2 in S' . Then, taking into account (5), vertices $v_0, v_5, v_3, v_4, v_1, v_2, x$ induce a skew star. This contradiction shows that S and S' cannot share v_2 . By symmetry, they cannot share v_4 .

Assume next that $v = v_5$. Then v_5 is black, v_0 is white (see (1) and (4)) and hence v_1, v_3 are black. Clearly we can find two vertices x and y in S' such that x is a neighbour of v_5 and y is a neighbour of x . By (2) and (3), neither x nor y belong to S . Also, x has no other neighbours in S (by (5)), and y cannot be adjacent to a white vertex of S , since otherwise y is black and hence x is white (as a vertex with two black neighbours v and y), in which case S' is destroyed. Therefore, we may assume that y is adjacent neither to v_2 nor to v_4 , since otherwise y must be white (as a vertex adjacent to a black matched vertex), in which case S' is destroyed. Now

- if y is adjacent neither to v_1 nor to v_3 , then vertices $v_0, v_3, v_1, v_2, v_5, x, y$ induce a skew star,
- if y is adjacent to both v_1 and v_3 , then y must white (as a vertex with two black neighbours), in which case S' is destroyed,
- if y is adjacent to exactly one vertex in $\{v_1, v_3\}$, say y is adjacent to v_1 but not to v_3 , then vertices $v_1, v_2, y, x, v_0, v_3, v_4$ induce a skew star.

Therefore, S and S' cannot share v_5 .

Suppose now that $v = v_1$. Then v_1 is black, v_0, v_2 are white (see (1) and (4)) and hence v_3, v_5 are black. Let x be a neighbour of v_1 in S' , and $y \in S'$ a neighbour of x . By (2) and (3), neither x nor y belong to S , and by (5), x has no other neighbours in S . As before, y cannot be adjacent to a white vertex of S , since otherwise y is black, x is white and S is destroyed. Therefore, y cannot be adjacent to v_4 , because in this case v_4 is a black matched vertex and hence y is white, which destroys S' . As a result, y is adjacent to v_3 , since otherwise vertices $v_1, v_2, x, y, v_0, v_3, v_4$ induce a skew star, and non-adjacent to v_5 , since otherwise y is white (as a vertex with two black neighbours v_3 and v_5) and hence S' is destroyed. This also implies that v_4 is coloured white, since otherwise y must be coloured white (as a vertex adjacent to a black matched vertex) and hence S' is destroyed. Thus, the colouring of S is alternating.

By the initial assumption, u_0 is not black, and hence $v_1 \neq u_0$. Therefore, either $x = u_0$ or $y = u_0$. Suppose $x = u_0$ and let z be the third neighbour of x in

S' . Similarly to y , vertex z does not belong to S and is adjacent to v_3 . But now vertices v_3, y, x, z induce a C_4 , and hence x must be black and y, z must be white, in which case S' is destroyed.

Assume now that $y = u_0$. We may consider a neighbour z of y in S' which does not belong to S , because y has exactly one neighbour in S and three neighbours in S' . We conclude that z cannot be adjacent to a white vertex of S , since otherwise z is black, y is white and S is destroyed. Therefore, z cannot be adjacent to v_4 , because in this case v_4 is a black matched vertex and hence z is white, which destroys S' . Also, z cannot be adjacent to v_1 , since otherwise vertices v_1, x, y, z form an induced C_4 , in which case x and z are white and S' is destroyed. This implies that z is adjacent to v_3 , since otherwise vertices $v_3, v_4, y, z, v_0, v_1, v_2$ induce a skew star. Therefore, vertex v_3 does not belong to S' (else S' contains a triangle) and we may consider one more neighbour of y in S' , say e . Similarly to z vertex e must be adjacent to v_3 . But now vertices v_3, y, z, e induce a diamond and hence z, e are white and S' is destroyed. This contradiction shows that S and S' cannot share v_1 . By symmetry, they cannot share v_3 .

Finally, we assume that $v = v_0$. Then v_0 is black, v_1, v_3, v_5 are white (see (1) and (4)) and hence v_2, v_4 are black, i.e. the colouring of S is alternating. Since v_0 is coloured black, we cannot have $v_0 = u_0$ (by the initial assumption). By (5), x has no other neighbours in S , and by (2) and (3), x and y do not belong to S . Let us show that

- y has no neighbours in S . Indeed, y cannot be adjacent to a white vertex of S , since otherwise y is black and hence x is white, which destroys S' . Also, y is not adjacent to v_0 , or else v_0, x, y is a triangle in S' . Assume y is adjacent to v_4 . Then y is not adjacent to v_2 , since otherwise y is white (as a neighbour of two black vertices v_2 and v_4), in which case S' is destroyed. But now vertices $v_0, v_5, v_1, v_2, x, y, v_4$ induce a skew star. This contradiction shows that y is not adjacent to v_4 , and by symmetry, y is not adjacent to v_2 .

Since y has no neighbours in S , vertex z cannot belong to S . Clearly, z is not adjacent to v_0 , because v_0, x, y, z induce a P_4 in S' . Let us show that

- z is adjacent neither to v_2 nor to v_4 . For a contradiction, let z be adjacent to v_2 . Then we may assume that z is not adjacent to v_4 , since otherwise it must be coloured white, which destroys S' . Also, we may assume that z is not adjacent to a white vertex of S , since otherwise z is a black vertex matched with v_2 implying that y is white and hence S' is destroyed. But now vertices

$v_0, v_1, v_3, v_4, x, y, z$ induce a skew star. This contradiction shows that z is not adjacent to v_2 , and by symmetry, z is not adjacent to v_4 .

Since z is adjacent neither to v_2 nor to v_4 , it must have at least two neighbours in $\{v_1, v_3, v_5\}$. Without loss of generality, let us assume that z is adjacent to v_1 and v_3 (the other case is similar). Clearly, either x or y is the centre of S' .

Assume first that x is at the centre of S' , and let e be the third neighbour of x in S' . Similarly to y , vertex e does not belong to S and has no neighbours in S . But then vertices $z, v_1, v_3, v_4, y, x, e$ induced a skew star.

Now suppose y is the centre of S' , and again let denote the third neighbour of y in S' by e . Similarly to z , vertex e does not belong to S and has no black neighbours in S . Also, z cannot have a neighbour in $\{v_1, v_3\}$, since otherwise this neighbour together with e, y, z create an induced C_4 , in which case vertex y must be coloured y destroying S' . But now vertices $v_0, v_1, v_3, v_4, x, y, e$ induce a skew star. This contradiction completes the proof of the lemma. \square

Theorem 29. *The DOMINATING INDUCED MATCHING problem can be solved for $S_{1,2,3}$ -free graphs in polynomial time.*

Proof. Let G be an $S_{1,2,3}$ -free graph with n vertices. In $O(n^6)$ time we check if G contains an induced copy of $S_{1,2,2}$. If not, the problem can be solved for G in polynomial time. If G contains an induced copy of $S_{1,2,2}$, we denote it by S and consider all possible colourings of its vertices. There are finitely many such colourings. With each of them we associate a subproblem. If the colouring of S is not alternating, then in the respective subproblem all copies of $S_{1,2,2}$ are destroyed by Lemmas 26 and 28 and hence these subproblems can be solved in polynomial time.

Now let us analyse subproblems corresponding to two alternating colourings of S . We claim that each of them can also be solved in polynomial time. Consider any of these two subproblems and denote it by Π and its input by H . If H has an induced copy of $S_{1,2,2}$, we know by Lemmas 27 and 28 that the central vertex of this copy is coloured black. Therefore, this copy admits *only one* of the two alternating colourings. Therefore, every subproblem of Π , except possibly one, can be solved in polynomial time. Therefore, in polynomial time we reduce the problem from H to a graph with strictly fewer vertices. Therefore, Π and hence the original problem can be solved in polynomial time. \square

3.5 Conclusion

At the beginning of this chapter we conjectured that, unless $P=NP$, the hereditary classes for which DIM is solvable in polynomial time are exactly the classes forbidding a graph in \mathcal{S} . To prove this conjecture, we need only concern ourselves with classes defined by a single forbidden induced subgraph in \mathcal{S} , the smallest non-trivial case being the claw-free graphs, which has been shown to yield a polynomial solution. This solution has been extended to the so-called fork-free graphs, and in an attempt to support the conjecture further, we extend this result first to $S_{1,2,2}$ -free graphs and then to graphs without a skew star.

The conjecture remains open, but future work might well include examining various classes defined by forbidding an $S_{i,j,k}$. In particular, since a solution exists for P_7 -free graphs and skew star-free graphs, the obvious next step is to examine $S_{1,3,3}$ -free graphs. Of course, the ultimate aim of this work is to obtain a general result for all i, j, k .

Part II

Independent Sets and Related Problems

Introduction

There are many algorithmic graph problems associated with the notion of an independent set, the central one being the MAXIMUM INDEPENDENT SET problem (MIS for short), which is the problem of finding in a graph an independent set of maximum cardinality. This problem is computationally difficult, i.e. it is NP-hard, and it is closely related to several other problems of combinatorial optimization, such as MAXIMUM CLIQUE or MINIMUM VERTEX COVER. Indeed, the independent sets of a graph G are the cliques of the complement of G , and the vertex covers of G are exactly the complements of its independent sets. At first glance, the close relationship between the three problems suggests that it is only a matter of taste which problem to study, because a solution to one of them immediately implies a solution to the others. This is true, but only up to a certain point. For instance, the MINIMUM VERTEX COVER problem admits fixed-parameter tractable algorithms, while MAXIMUM CLIQUE and MAXIMUM INDEPENDENT SET do not. Also, when we study the problems restricted to particular classes of graphs, the language of independent sets seems to be more convenient than the language of cliques. For instance, it is known that the MAXIMUM INDEPENDENT SET problem can be solved in polynomial time in the class of line graphs, which is equivalent to the MAXIMUM MATCHING problem in general graphs. Therefore, the MAXIMUM CLIQUE problem can be solved in polynomial time for the complements of line graphs. However, describing this solution in the terminology of cliques may be hard going. Since these results are equivalent, it may just be a matter of taste in this case. However, many similar results are known for classes defined by forbidding small connected graphs or their disjoint unions (see e.g. [48], [50],[51],[52]). These graphs are easier to name (e.g. claw, fork, C_k ...) and this may explain our preference for independent sets over cliques.

A zetetic reader may ask what makes the MAXIMUM INDEPENDENT SET problem simple in the class of line graphs. The answer to this question is “the absence of a claw” and this answer comes from the result of Alekseev, who showed in [53] that the problem is solvable in polynomial time in a class of graphs defined by finitely

many forbidden induced subgraphs only if the set of forbidden subgraphs contains a graph from the class \mathcal{S} , i.e. a graph every connected component of which is of the form $S_{i,j,k}$. Again, obtaining and describing this result in the terminology of cliques would be much harder and less natural. If we apply this result to the class of line graphs (which is described by 9 minimal forbidden induced subgraphs), we observe that the only forbidden graph which belongs to \mathcal{S} is a claw (i.e. $S_{1,1,1}$). The fundamental importance of the claw for the MAXIMUM INDEPENDENT SET problem was confirmed by extending the solution from line graphs to claw-free graphs.

Recently, the result for claw-free graphs was generalized to graphs without apples. In this dissertation, we go further and obtain in chapter 4 some polynomial-time results for graphs without *large apples*.

The MAXIMUM INDEPENDENT SET problem has several important generalizations. One of them deals with finding in a graph maximum k -regular induced subgraphs. For $k = 0$, this problem coincides with MIS, and for $k = 1$, it is known as MAXIMUM INDUCED MATCHING. The latter problem is even harder than MIS, because it is NP-hard already for claw-free graphs. In chapter 5, generalizing some of the previously known results, we show that finding a maximum k -regular induced subgraph in a $2P_3$ -free graph can be solved in polynomial time for *any* value of k .

One more important problem concerning independent sets is INDEPENDENT DOMINATION, also known as MINIMUM MAXIMAL INDEPENDENT SET. It is also more difficult than MAXIMUM INDEPENDENT SET, in the sense that it is NP-hard for claw-free graphs. We study this problem in chapter 6, where we obtain polynomial-time algorithms for graphs in some particular classes.

Chapter 4

The Maximum Weight Independent Set Problem in Graphs Without Large Apples

4.1 Introduction

Given a graph G and an assignment of real-valued *weights* to its vertices, the MAXIMUM WEIGHT INDEPENDENT SET (WIS) problem is that of finding in G an independent set I of maximum total weight. This problem generalises the MAXIMUM INDEPENDENT SET (MIS) problem in the sense that WIS coincides with MIS if each vertex has weight 1.

An *apple* A_k is the graph obtained from a chordless cycle C_k of length $k \geq 4$ by adding a vertex that has exactly one neighbour on the cycle (see Figure 4.1). A graph G is *apple-free* if it contains no A_k , for $k \geq 4$, as an induced subgraph. The class of apple-free graphs is a common generalization of claw-free graphs and chordal graphs. Both these classes have been extensively studied in the literature and both enjoy many attractive properties (see, e.g. [54, 55, 56, 57, 25, 58, 59]). In particular, WIS, while NP-hard in general, admits polynomial-time solutions when restricted to chordal graphs or graphs with no induced claw.

As we mentioned in the last chapter, the solution to the problem in the class of claw-free graphs was proposed by Minty in 1980 [28]. The importance of this solution is due to the fact that it generalizes the celebrated matching algorithm by Edmonds [60], which is “one of the most involved of combinatorial algorithms” [61], according to Lovász and Plummer.

For nearly three decades, the solution for claw-free graphs remained unim-

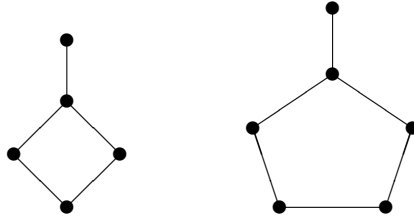


Figure 4.1: Two smallest apples A_4 and A_5

proved. Recently, it was generalized in two different ways: first, Lozin and Milanič [48] showed how to solve the WIS problem in the class of fork-free graphs (see also [62] for a different solution to the unweighted version of the problem) and then Brandstädt et al. proved that the problem is polynomially solvable for apple-free graphs [50].

In this chapter, we study a further generalization of the class of apple-free graphs called *graphs without large apples*: these are (A_k, A_{k+1}, \dots) -free graphs for values of k strictly greater than 4.

Echoing the results of Chapter 2, and the motivation for the results of Chapter 3, we recall the definition of the parameter $\kappa(M)$, where M is a set of graphs, in order to state a hardness result proved in [63].

For $k \geq 3$, let \mathcal{S}_k be the class of $(C_3, \dots, C_k, H_1, \dots, H_k)$ -free planar graphs of vertex degree at most 3, where C_i denotes the cycle on i vertices and H_i is the graph obtained from a path with i edges by attaching two new vertices of degree one to each of the two endpoints of the path (see Figure 3.1). To every graph G we associate the parameter $\kappa(G)$, which is the maximum k such that $G \in \mathcal{S}_k$. If G belongs to no class \mathcal{S}_k , then $\kappa(G)$ is defined to be 0, and if G belongs to all classes \mathcal{S}_k , then $\kappa(G)$ is defined to be ∞ . Finally, for a set of graphs M , $\kappa(M)$ is defined as $\kappa(M) = \sup\{\kappa(G) : G \in M\}$.

Theorem 30. *Let M be a set of graphs and X the class of M -free planar graphs of maximum degree at most 3. If $\kappa(M) < \infty$, then the maximum independent set problem is NP-hard in the class X .*

Theorem 30 suggests that, unless $P=NP$, the maximum independent set problem is solvable in polynomial time in the class of M -free graphs *only if* the parameter κ is unbounded in the set M . There are three basic ways to unbind this parameter in M :

- (1) include in M a graph G with $\kappa(G) = \infty$;

- (2) include in M infinitely many cycles;
- (3) include in M infinitely many graphs of the form H_i .

The literature contains only a few results dealing with classes of the third type (see, e.g. [64]); most polynomial-time results refer to classes of the first two types. Examples of the first type include claw-free graphs [65, 28, 66, 67], fork-free graphs [48] and mK_2 -free graphs [68, 69, 70], and examples of the second type include chordal graphs [71], bipartite graphs and other subclasses of perfect graphs [72].

Observe that each of the mentioned results belongs to precisely one of the three categories specified above. The importance of apple-free graphs is due to the fact that it is the first class generalizing simultaneously graphs of two types: type 1 (claw-free graphs) and type 2 (chordal graphs). Graphs without large apples, i.e. (A_k, A_{k+1}, \dots) -free graphs with $k \geq 5$, additionally include fork-free graphs and $2K_2$ -free graphs, which is not the case for $k = 4$. The complexity of the MAXIMUM INDEPENDENT SET problem for graphs without large apples is unknown even for $k = 5$.

The key result of the present chapter is a sufficient condition for claw-freeness of (A_k, A_{k+1}, \dots) -free graphs for any value of k . This result is based on a deep combinatorial analysis of the structure of graphs without large apples. We show that the condition is satisfied by graphs of bounded vertex degree of sufficiently large tree-width. Together with polynomial-time solvability of the problem in graphs of bounded tree-width, this leads to the conclusion that for any fixed k and Δ , the WIS problem is solvable in polynomial time for (A_k, A_{k+1}, \dots) -free graphs of vertex degree at most Δ . We also show that the condition is satisfied by apex-minor-free graphs of sufficiently large tree-width, a class that will be defined later. This implies an efficient solution to the problem in (A_k, A_{k+1}, \dots) -free graphs excluding a fixed apex graph as a minor.

Graphs of tree-width at most k , also known as *partial k -trees*, generalize trees and are very important from an algorithmic viewpoint, since many graph problems that are NP-hard for general graphs are solvable in linear time when restricted to graphs of tree-width at most k [73]. In particular, the WIS problem is solvable in linear time (in the number of vertices) for graphs that belong to any class of bounded tree-width.

Modules, clique separators, and a useful decomposition

Recall that a *module* M of a graph G is not distinguished by any vertex in $G \setminus M$. That is to say, no vertex outside M has both a neighbour and a non-neighbour in

M . A module is *trivial* if it consists of the whole graph, a single vertex, or the empty set and a graph is *prime* if all its modules are trivial. It is well-known that the WIS problem can be reduced in polynomial time from any hereditary class X to prime graphs in X (see e.g. [48]).

A *clique separator* in a connected graph G is a subset K of vertices of G , which induces a complete graph, such that the graph $G - K$ is disconnected. Whitesides [74] and Tarjan [75] showed that WIS can be reduced in polynomial time to graphs without clique separators. Moreover, Brandstädt and Hoàng recently combined decomposition by clique separators with modular decomposition to obtain a more general decomposition scheme:

Theorem 31 (Brandstädt and Hoàng [76]). *Let X be a class of graphs. If the WIS problem can be solved in polynomial time for those induced subgraphs of graphs in X which are prime and have no clique separators, then WIS is solvable in polynomial time for graphs in X .*

Therefore, when developing a polynomial-time solution to the WIS problem in a certain hereditary class, we can safely focus on graphs which are prime and have no clique separators. Throughout the section we call such graphs *atoms*.

4.2 A Sufficient Condition for Claw-Freeness of (A_k, A_{k+1}, \dots) -free Atoms

In this section, we state and prove a sufficient condition for claw-freeness of (A_k, A_{k+1}, \dots) -free atoms, which generalizes a result from [77], and leads to the solutions to the WIS problem in certain subclasses of graphs without large apples, developed in Section 4.3. The result relies on the following notion.

Definition 32. *Let C be a chordless cycle in a graph G . We say that C is r -invisible if every vertex outside C has at least r consecutive non-neighbours in C .*

For example, every chordless cycle in a graph is 0-invisible, every 3-cycle in a K_4 -free graph is 1-invisible, and the only connected graphs containing an r -invisible chordless cycle C such that $|V(C)| = r$ are cycles themselves. Clearly, if a cycle is r -invisible, then it is also r' -invisible, for every $r' \leq r$.

Why do we consider r -invisible cycles? If a graph contains a large induced apple, then it necessarily contains an induced claw, as well as a long chordless cycle. Of course, the converse of this statement does not hold; a graph containing an induced claw and a long chordless cycle could contain no apples at all. So the

following question arises: Can we define a property of cycles such that every graph containing an induced claw and a chordless cycle satisfying the property contains a large apple? For atoms, we show that this is indeed the case, and the right property is that of high invisibility. As we show in the next lemma, the presence of an induced claw and of a chordless cycle of sufficiently high invisibility in an atom forces the presence of a large induced apple.

Lemma 33. *Let $k \geq 4$, and let G be an (A_k, A_{k+1}, \dots) -free atom that contains a $128k$ -invisible chordless cycle. Then G is claw-free.*

Proof. Let G be an (A_k, A_{k+1}, \dots) -free atom that contains a $128k$ -invisible chordless cycle C ; that is, every vertex in $V(G) \setminus C$ has at least $128k$ consecutive non-neighbours on C .

Since G is prime, it is connected. Furthermore, we may assume that G is not a cycle (or we are done); in particular, since C is $128k$ -invisible and there exists a vertex outside C , this implies that $|V(C)| \geq 128k$.

Let us now prove a few helpful statements valid for any $(k - 1)$ -invisible sufficiently long cycle in G , including the cycle C .

Claim 34. *Let C' be an $(l - 1)$ -invisible chordless cycle with $|V(C')| \geq 4l$, and v a vertex outside the cycle. If v is adjacent to at least one vertex of C' , then*

- v has at least two and at most four neighbours on C' .

Moreover,

- if v has exactly two neighbours on C' , then they are consecutive vertices of C' ,
- if v has exactly three neighbours on C' , then they are consecutive vertices of C' ,
- if v has four neighbours on C' , then they create two pairs of consecutive vertices of C' ,
- if v is adjacent to a vertex that has no neighbours on C' , then v has two neighbours on C' .

In particular,

- v cannot have an isolated neighbour on C' , i.e. if v has a neighbour i on C' , then v must also be adjacent to $i - 1$ or $i + 1$.

In addition, if u is another vertex outside C' with at least one neighbour on C' , then

- if $N(v) \cap C'$ properly contains $N(u) \cap C'$, then u is adjacent to v , unless $N(v) \cap C' = \{i, i+1, i+2, i+3\}$ and $N(u) \cap C' = \{i+1, i+2\}$, in which case u and v are non-adjacent,
- if one of u and v has exactly three neighbours on C' and $N(v) \cap N(u) \cap C' = \emptyset$, then u and v are non-adjacent,
- if one of u and v has four neighbours on C' and $|N(v) \cap N(u) \cap C'| \leq 1$, then u and v are non-adjacent.

All of the above statements can be checked by direct inspection, except for the case when v is adjacent to vertices $i-1$ and $i+1$ on C' (that is, to exactly two vertices at distance 2 along the cycle). In order to prove that this is not possible, let us consider the set A of all vertices of G adjacent to exactly two vertices of $C' - \{i\}$, namely to $i-1$ and to $i+1$. In particular, $i, v \in A$. Let B be the connected component of the complement of $G[A]$ containing i and v , and let z be a vertex of G distinguishing B . Without loss of generality, we may assume that z is adjacent to v and non-adjacent to i (obviously, B contains two non-adjacent vertices distinguished by z). To avoid a large induced apple, we must conclude that z is adjacent to $i-1$ and $i+1$ and to no other vertex of C' . But then z belongs to A and consequently to B , which contradicts the choice of z .

According to Claim 34, all vertices outside C can be partitioned into four types according to the number of neighbours on the cycle (type j standing for the vertices with exactly j neighbours on C). If the neighbours of a vertex v of type 4 are not consecutive on C , we will say that v is of type 4 with *opposite* neighbours.

To prove the lemma, we suppose for contradiction that G contains an induced claw $K = (a; b, c, d)$ with a being the centre. By Claim 34, K cannot have more than two vertices on C . The rest of the proof is partitioned into several cases according to the size of the intersection of K and C .

Case 1: $K \cap C = \{a, d\}$. Let y be the other neighbour of a on C . To avoid a claw with three vertices on C , we conclude that y is adjacent both to b and to c . By Claim 34, $N(b) \cap C$ cannot properly contain $N(c) \cap C$, and vice versa. To avoid a large induced apple, we conclude that $N(b) \cap C = N(c) \cap C$.

Denote by A the set $\{v \in V : N(v) \cap C = N(b) \cap C\}$ and by U the vertex set of the connected component of the complement of $G[A]$ which contains b and c . Since G is prime, there must exist a vertex z that distinguishes U . Obviously, $z \notin A$, i.e. $N(z) \cap C \neq N(b) \cap C$. Also, it is not difficult to see that z distinguishes two vertices of U that are not adjacent in G . Without loss of generality we may

assume that z is adjacent to b , but not to c . Since no vertex of C distinguishes A , we know that z does not belong to C . Also,

- z has no neighbours on C , since otherwise a large induced apple arises. To see this, assume z has at least one neighbour on C and let $I = \{c_i, c_{i+1}, \dots, c_{j-1}, c_j\}$ be a largest set of consecutive vertices of C in which c_i and c_j have neighbours in $\{b, c, z\}$ while the other vertices have no neighbours in $\{b, c, z\}$. We observe that the set $C - I$ contains at least one vertex, since otherwise $N(z) \cap C = N(b) \cap C = N(c) \cap C = \{c_i, c_j\}$ contradicting the choice of z .

If one of c_i and c_j has a non-neighbour in $\{b, c, z\}$, then a large apple induced by vertices from the set $I \cup \{b, c, z\}$ can be easily found.

Assume now that c_i and c_j are adjacent to each vertex of $\{b, c, z\}$. Then c_{i-1} is adjacent to each vertex of $\{b, c, z\}$, since otherwise c_j is an isolated neighbour of a vertex in $\{b, c, z\}$ (see Claim 34). Similarly, c_{j+1} is adjacent to each vertex of $\{b, c, z\}$. But then $N(z) \cap C = N(b) \cap C = N(c) \cap C = \{c_{i-1}, c_i, c_j, c_{j+1}\}$ (regardless of whether $c_{i-1} = c_{j+1}$ or not), which contradicts the choice of z .

Therefore, z is of type 0 and hence b and c are of type 2, i.e. $N(b) \cap C = N(c) \cap C = \{a, y\}$. Let K_0 denote a maximal clique containing $\{a, y\}$ in the subgraph of G induced by $\{a, y\} \cup A$. Since G has no clique separators, there must exist a (shortest) path $P = (v_0, \dots, v_r)$ connecting $z = v_0$ to C and avoiding K_0 . Without loss of generality we will assume that K and z are chosen so that the length of P is as small as possible. In particular, no vertex of P , except z , distinguishes b and c .

Suppose first that P also avoids A . Then vertex v_r is of type 2 and the remaining vertices of P are of type 0. If b and c have no neighbours on P except z , then an induced apple of order at least $|V(C)|/2$ can easily be found. Otherwise, let v_i be the common neighbour of b and c with maximum index. If $i > 1$, then vertex z together with the cycle formed by vertices b, v_i, \dots, v_r and a large portion of C create an induced apple of order at least $|V(C)|/2$. If $i = 1$, then vertex z together with the cycle formed by vertices c, v_i, \dots, v_r and a large portion of C create an induced apple of order at least $|V(C)|/2$.

Assume now that P intersects A and let i be the maximum index such that $v_i \in A$. Since K_0 is a maximal clique and P avoids K_0 , there must exist a vertex $u \in K_0 \cap A$ non-adjacent to v_i . Also, it is clear that $r > i$. If any vertex v_j of P with $j > i$ distinguishes u and v_i , then we could replace z by v_j and K by $G[a, d, u, v_i]$. Therefore, v_{i+1} is adjacent both to v_i and u and the vertices of P following v_{i+1} are not adjacent to v_i and u . Now the vertices v_i, v_{i+1}, \dots, v_r and a portion of C of length at least $|V(C)|/2 \geq 64k$ create a cycle C' which is $32k$ -invisible, and such

that vertex u has two non-consecutive neighbours on C' . This is a contradiction to Claim 34, which completes the proof of Case 1.

Ruling out Case 1 allows us to assume in what follows that

- (A) if u and v are two non-adjacent vertices outside the cycle C each of which has a neighbour on C , then $N(u) \cap C \neq N(v) \cap C$,

since otherwise a claw as in Case 1 arises.

Case 2: $K \cap C = \{b, c\}$. Then a is of type 3 or 4, and hence d has a neighbour on C . If a is of type 3, then $N(a) \cap C$ and $N(d) \cap C$ must be disjoint, which is impossible by Claim 34. Therefore, a is of type 4 and by the same claim we conclude that $N(a) \cap C$ and $N(d) \cap C$ must intersect in at least two vertices. In fact, since $|N(a) \cap C| = 4$ and $b, c \in (N(a) \cap C) \setminus (N(d) \cap C)$, we conclude that $N(a) \cap C$ and $N(d) \cap C$ intersect in exactly two vertices. If d is of type 2, then $N(a) \cap C = \{i, i+1, i+2, i+3\}$ and $N(d) \cap C = \{i+1, i+2\}$ (for some $i \in V(C)$), which is not possible again by Claim 34. If d is of type 3, then $K \cup (C \setminus N(d))$ induces a long apple. Thus, d is of type 4. Suppose that a has opposite neighbours, say $i, i+1, j, j+1$. If $\{b, c\} = \{i, j\}$, then $N(d) \cap C = \{i+1, i+2, j+1, j+2\}$, and a long portion of C together with K form a large induced apple. If $\{b, c\} = \{i, j+1\}$, then $N(d) \cap C = \{i+1, i+2, j-1, j\}$, and a long portion of C together with a and d form a large induced apple. Therefore, a has consecutive neighbours, say $i, i+1, i+2, i+3$. But then $\{b, c\} = \{i, i+3\}$ and $N(d) \cap C = \{i+1, i+2, j, j+1\}$ for some $j \notin \{i-1, \dots, i+3\}$, and a long portion of C together with K form a large induced apple.

Case 3: $K \cap C = \{a\}$. Assume one of the vertices in $\{b, c, d\}$ is of type 3, say $N(b) \cap C = \{i-1, i, i+1\}$, and let C' be the cycle obtained from C by replacing i with b . If $N(c) \cap C = \{i-2, i-1, i\}$ and $N(d) \cap C = \{i, i+1, i+2\}$ (or vice versa), then a large apple arises. Otherwise, the neighbourhoods of c and d either do not satisfy Claim 34 with respect to C or C' , or they do not satisfy assumption (A) with respect to C or C' . By the same arguments we conclude that if none of b, c, d is of type 3, then none of them is of type 2 or of type 4 with opposite neighbours.

Finally, let all three vertices b, c, d be of type 4 with consecutive neighbours on C . Then the union of their neighbours forms an interval $i < \dots < a < \dots < j$ on C . Taking into account assumption (A), we may assume without loss of generality that $i \in N(b)$, $j \in N(d)$, $i, j \notin N(c)$. Then vertex c together with the cycle obtained

from C by replacing the vertices between i and a by b and by replacing the vertices between a and j by d form a large apple.

Case 4: $K \cap C = \{d\}$. If a is of type 3 or 4, then either a new claw satisfying one of the previously studied cases arises, or the neighbourhoods of b, c on C violate Claim 34 or assumption (A). Therefore, we conclude that a is of type 2. If b or c also have neighbours on C , then either the neighbourhoods of b, c on C violate assumption (A) a large apple arises. Therefore, b and c are both of type 0.

Let y be the other neighbour of a on C . Since G has no clique separators, there must exist a (shortest) path P connecting $\{b, c\}$ to C and avoiding $\{a, d, y\}$, and since b and c are both of type 0, the vertex of P which has a neighbour on C is of type 2 and the remaining vertices of P are of type 0. But then G has a cycle C' containing vertex a and a portion of C of length at least $|V(C)|/2 \geq 64k$. It is not difficult to see that C' is $32k$ -invisible. Without loss of generality, we may assume that $b \in V(C')$ and $c \notin V(C')$, but then a is an isolated neighbour of c on C' . This is a contradiction to Claim 34.

Note that in Cases 1-4 a contradiction would also be reached under a weaker assumption that C is $8k$ -invisible.

Case 5: $K \cap C = \emptyset$ and a vertex of K has a neighbour on C . If K contains a vertex of type 3 or 4, then this vertex together with a large portion of C create a chordless cycle of length at least $64k$ intersecting K , which reduces the problem to one of the previously studied cases. Therefore, each vertex of K is of type 2 or 0.

Assume a has neighbours on C , say i and $i+1$. Then, to avoid an induced claw intersecting C , we conclude that both i and $i+1$ have exactly two neighbours among b, c, d . Without loss of generality we let i be adjacent to b and c . Then, according to assumption (A), we may assume without loss of generality that $i+1$ is adjacent to d and c . Since b and d are of type 2, we conclude, again by assumption (A), that b is adjacent to $i-1$, while d is adjacent to $i+2$. But now $(C - \{i, i+1\}) \cup K$ is a large induced apple, a contradiction.

Suppose now that a is of type 0. If at least two vertices in $\{b, c, d\}$ have neighbours on C , then, taking into account assumption (A), it is easy to find a large chordless cycle intersecting K . If exactly one vertex of K has neighbours on C , say b is adjacent to x and y , we proceed similarly as in Case 4: since G has no clique separators, there must exist a path connecting $\{a, c, d\}$ to C and avoiding $\{b, x, y\}$, but then G has a $32k$ -invisible chordless cycle (of length at least $64k$) and intersecting K .

Note that in Case 5 a contradiction would also be reached under a weaker

assumption that C is $32k$ -invisible.

Case 6: $K \cap C = \emptyset$ and no vertex of K has a neighbour on C . Let us call a *quasi-chord* for C any chordless path $P = (p_1, \dots, p_t)$ such that p_1 and p_t are of type 2 with $N(p_1) \cap C \neq N(p_t) \cap C$ and all the other vertices of the path are of type 0. The neighbours of p_1 and p_t split C into two segments. The largest of these segments together with P create a large chordless cycle. To prove the lemma, we will show that there is a quasi-chord which intersects K or has a neighbour in K .

First, let us show that there is at least one quasi-chord for C which is strictly closer to the claw than the cycle C itself.

Since G is connected, there must be a path connecting the claw to the cycle. Let $P' = (x_1, \dots, x_p)$ be a shortest path between K and C with $x_1 \in K$ and with x_p having a neighbour on C . By the choice of P' , no vertex of this path has a neighbour on C except x_p . Also, to avoid a claw intersecting C , we conclude that the neighbourhood of x_p on C consists of two consecutive vertices of the cycle, say $i, i + 1$. If the clique $\{x_p, i, i + 1\}$ can be extended to a larger clique by means of vertices of type 2, we extend it as much as possible and denote the resulting clique by Q .

Since G has no separating clique, there must exist a path connecting K to C and avoiding Q . Let $P'' = (y_1, \dots, y_t)$ be a chordless path of this type with $y_1 \in K$ and with y_t having a neighbour on C different from $i, i + 1$. We may assume that no vertex of P'' except y_t has a neighbour on C . Indeed, if a vertex y_j with $j < t$ has a neighbour on C different from $i, i + 1$, we can replace y_t by y_j , making P'' shorter. If y_j has a neighbour in $\{i, i + 1\}$, it must be adjacent to both i and $i + 1$ (since otherwise y_j is of type 1 with respect to C). Since Q is not extendable to a larger clique by means of vertices of type 2, y_j must have a non-neighbour $z \in Q$ of type 2. But then $i - 1, i, z, y_j$ induce a claw intersecting C . This proves that no vertex of P'' except y_t has a neighbour on C . To avoid a claw intersecting C , we conclude that y_t is of type 2.

Now any chordless path connecting x_p to y_t and consisting of vertices of P' and P'' (and possibly of K) forms a quasi-chord for C . Moreover, this path contains vertex x_p which is strictly closer to the claw than C .

Now we show that there is a quasi-chord for C which intersects K or has a neighbour in K . To this end, let us denote by $P = (p_1, \dots, p_t)$ a quasi-chord for C which is as close to K as possible. Assume P neither intersects K nor has a neighbour in K . Then we consider a shortest path $P' = (x_1, \dots, x_p)$ connecting K to P . By the choice of P and by the above discussion, P' must be closer to K than

C , which means no vertex of P' belongs to C or has a neighbour on C . Also, by the choice of P' we know that no vertex of P' has a neighbour on P except x_p , and to avoid a claw intersecting a large cycle we conclude that the neighbourhood of x_p on P consists of two consecutive vertices of the path, say p_i, p_{i+1} . Similarly as before, using the “no separating clique” argument, we find one more chordless path $P'' = (y_1, \dots, y_t)$, but this time P'' connects K to $C \cup P$ and avoids p_i, p_{i+1} . Finally, as before, we assume that no vertex of P'' except y_t has a neighbour in $C \cup P$.

Let P^* be a chordless path connecting x_p to y_t and consisting of vertices of P' and P'' (and possibly of K). If y_t has no neighbours on C (i.e. the neighbours of both x_p and y_t belong to P), then part of P can be replaced by P^* creating another quasi-chord for C , which is strictly closer to K than P , since it contains vertex $x_p \in P^*$. This contradicts the choice of P .

Assume y_t has neighbours on C . Then y_t must be of type 2 with respect to C (to avoid a claw intersecting C). Since $N(p_1) \cap C \neq N(y_t) \cap C$, we may assume without loss of generality that $N(y_t) \cap C \neq N(p_1) \cap C$. Then $\{p_1, \dots, p_i\} \cup P^*$ is a quasi-chord for C , which is strictly closer to K than P , since it contains vertex $x_p \in P^*$. This final contradiction shows that P intersects K or has a neighbour in K . Therefore, G contains a $32k$ -invisible cycle that intersects K or has a neighbour in K . \square

4.3 Polynomial Results

The results of the previous section suggest the following sufficient condition for polynomial-time solvability of the WIS problem for graphs without large apples.

Lemma 35. *Let X be a hereditary class of graphs and let $k \geq 4$ be an integer. Suppose that the WIS problem can be solved in polynomial time for those (A_k, A_{k+1}, \dots) -free atoms in X that contain no $128k$ -invisible chordless cycles. Then, the WIS problem can be solved in polynomial time for (A_k, A_{k+1}, \dots) -free graphs in X .*

Proof. Let G be an (A_k, A_{k+1}, \dots) -free graph from X . By Theorem 31, we may assume that G is an atom. If G is claw-free, then an independent set of maximum weight can be found in polynomial time [65, 28, 66, 67]. Assume now that G contains an induced claw. Then, by Lemma 33, G does not contain any $128k$ -invisible chordless cycles. In this case, we can solve the WIS problem in G in polynomial time using an algorithm given by the hypothesis. \square

Below we identify two large families of graph classes that satisfy the condition of Lemma 35.

4.3.1 Graphs of Bounded Vertex Degree

The first family includes graph classes of bounded vertex degree. To verify the condition of the lemma, we will show that every graph of bounded degree that contains no chordless cycles of high invisibility is of bounded tree-width. We first recall a theorem by Bodlaender and Thilikos stating that bounded degree in conjunction with bounded chordality imply bounded tree-width. More formally:

Lemma 36 ([32]). *For any two integers $k \geq 3$ and $\Delta \geq 1$ there is a number $f(k, \Delta)$ such that every (C_k, C_{k+1}, \dots) -free graph of maximum degree at most Δ has tree-width at most $f(k, \Delta)$.*

Corollary 37. *For every two positive integers $\Delta, r \geq 1$ there is a number N such that every graph of maximum degree at most Δ and with no r -invisible chordless cycles has tree-width at most N .*

Proof. Let G be a graph of maximum degree at most Δ and with no r -invisible chordless cycles. We will show that the chordality of G is less than $r(\Delta + 1)$, and the statement will follow from Lemma 36 by taking $N = f(r(\Delta + 1), \Delta)$. Suppose that G contains a chordless cycle C of length at least $r(\Delta + 1)$. Since C is not r -invisible, there exists a vertex v outside C that has at least one neighbour among every r consecutive vertices of C . Hence, $|N(v) \cap C| \geq \lfloor |V(C)|/r \rfloor \geq \Delta + 1$, which implies that the degree of v exceeds Δ , a contradiction. \square

Since the WIS problem is polynomial-time solvable for graphs of bounded tree-width (see, e.g. [73]), we conclude from Lemma 35 and Corollary 37 that

Theorem 38. *For any two integers $k \geq 4$ and $\Delta \geq 1$, the WIS problem is solvable in polynomial time in the class of (A_k, A_{k+1}, \dots) -free graphs of maximum degree at most Δ .*

4.3.2 Apex-Minor-Free Graphs

In what follows, we derive a similar conclusion for graphs that exclude a fixed apex graph as a minor. A graph H is said to be a *minor* of a graph G if H can be obtained from G by means of vertex deletions, edge deletions and edge contractions. *Contracting an edge uv* in a graph G means replacing the edge uv together with its two endpoints with a new vertex adjacent precisely to all neighbours of either u or v in G . If H is not a minor of G , we say that G is *H -minor-free* and call H a *forbidden minor* for G .

A class of graphs is called *minor closed* if with every graph G it contains all minors of G . One of the most famous examples of this type is the class of planar graphs. It is well known that any minor-closed graph class can be described by a unique *finite* set of minimal forbidden minors [78]. For instance, the class of planar graphs is exactly the class of $(K_5, K_{3,3})$ -minor-free graphs.

An *apex graph* is a graph that contains a vertex the deletion of which leaves a planar graph. For instance, both K_5 and $K_{3,3}$ are apex graphs. We call a class of graphs *apex-minor-free* if it is defined by a single forbidden minor H , which is a (non-planar) apex graph. Therefore, the planar graphs are contained in every apex-minor-free class. Our goal is to show that any apex-minor-free class of graphs satisfies the condition of Lemma 35. We will need the following result from [79], analogous to Lemma 36.

Lemma 39 ([79]). *For any integer $k \geq 3$ and any apex graph H , there is a number $f_H(k)$ such that every H -minor-free (C_k, C_{k+1}, \dots) -free graph has tree-width at most $f_H(k)$.*

We will also need a technical preparatory statement. An $n \times n$ grid G_n is the graph with the vertex set $\{1, \dots, n\} \times \{1, \dots, n\}$ such that (i, j) and (k, l) are adjacent if and only if $|i - k| + |j - l| = 1$. By a result of Robertson and Seymour [80], graphs of large enough tree-width must contain a grid of a prescribed fixed size as a minor. For apex-minor-free graph classes, even more is true. In the following lemma, an *augmented grid* is a grid G_n augmented with additional edges (and no additional vertices). Vertices (i, j) with $\{i, j\} \cap \{1, n\} \neq \emptyset$ are *boundary vertices* of the grid; the other ones are *non-boundary*. We say that a graph G can be contracted into a graph R if there exists a collection of disjoint non-empty subsets of $V(G)$ indexed by vertices of R , say $\{V_r : r \in V(R)\}$, such that each V_r induces a connected subgraph of G , the union of all V_r is equal to $V(G)$, and there exists an edge in G connecting a vertex in V_r with a vertex in $V_{r'}$ if and only if r and r' are adjacent in R . If this is the case, we also say that the set V_r gets contracted to vertex r .

Lemma 40 ([81]). *Let H be an apex graph. Let $r = 14|V(H)| - 22$. For every integer k there is an integer $g_H(k)$ such that every connected H -minor-free graph of tree-width at least $g_H(k)$ can be contracted into an $k' \times k'$ augmented grid R such that $k' \geq k$, and each vertex $v \in V(R)$ is adjacent to less than $(r+1)^6$ non-boundary vertices of the grid.*

With the help of Lemma 40 we now derive the following result.

Lemma 41. *For every apex graph H and every two integers k and l there is an integer $f(H, k, l)$ such that every connected H -minor-free graph G of tree-width at least $f(H, k, l)$ contains a k -invisible chordless cycle C of length at least l .*

Proof. Let $r = 14|V(H)| - 22$, let f_H be the function given by Lemma 39, and let g_H be the function given by Lemma 40. Furthermore, let

$$f(H, k, l) = g_H \left(f_H \left(\max\{l, (k+1)(r+1)^6\} \right) + 2 \right) .$$

We will show that the function $f(H, k, l)$ satisfies the claimed property.

Let G be a connected H -minor-free graph of tree-width at least $f(H, k, l)$. By Lemma 40, G can be contracted into an $k' \times k'$ augmented grid R where $k' \geq f_H(\max\{l, (k+1)(r+1)^6\}) + 2$ and such that each vertex $v \in V(R)$ is adjacent to less than $(r+1)^6$ non-boundary vertices of the grid. For $i, j \in \{1, \dots, k'\}$, let $V(i, j)$ denote the subset of $V(G)$ that gets contracted to the vertex (i, j) of the grid. Furthermore, let R_0 denote the $(k'-2) \times (k'-2)$ augmented sub-grid, induced by the non-boundary vertices of R . Since the tree-width of an $n \times n$ grid is n [82], and the tree-width cannot decrease by adding edges, we conclude that the tree-width of R_0 is at least $k' - 2 \geq f_H(\max\{l, (k+1)(r+1)^6\})$. Moreover, as R_0 is H -minor-free, Lemma 39 implies that R_0 contains a chordless cycle C_0 of length at least $\max\{l, (k+1)(r+1)^6\} \geq l$. By the above, every vertex $v \in V(R)$ is adjacent to less than $(r+1)^6$ vertices of R_0 . Therefore, the neighbours of v on C_0 (if any) divide the cycle into less than $(r+1)^6$ disjoint paths, whose total length is at least $|V(C_0)| - (r+1)^6$. In particular, this implies that every vertex of $V(R)$ is non-adjacent to at least $\frac{|V(C_0)| - (r+1)^6}{(r+1)^6} \geq k$ consecutive vertices of C_0 .

Let the cyclic order of vertices of R_0 on C_0 be given by $((i_1, j_1), (i_2, j_2), \dots, (i_s, j_s))$. To complete the proof, we have to lift the cycle C_0 to a chordless cycle C in G . Informally, we will replace each pair of incident edges $(i_{p-1}, j_{p-1})(i_p, j_p)$ and $(i_p, j_p)(i_{p+1}, j_{p+1})$ in C_0 with a shortest path in G connecting vertex (i_{p-1}, j_{p-1}) to vertex (i_{p+1}, j_{p+1}) , such that the internal vertices of the path all belong to V_{i_p, j_p} .

Let us now describe the procedure formally. For each $p = 0, 1, \dots, s$, let G_p denote the graph obtained from G by contracting each of the sets $V(i_{p+1}, j_{p+1}), V(i_{p+2}, j_{p+2}), \dots, V(i_s, j_s)$ into a single vertex. (In particular, $G_s = G$.) Starting with C_0 , we shall define a sequence of $s+1$ cycles C_0, C_1, \dots, C_s such that:

1. For each $p = 0, 1, \dots, s$, the cycle C_p is a chordless cycle in G_p . (In particular, C_s is a chordless cycle of G .)

2. For each $p = 0, 1, \dots, s$, the cycle C_p is of the form

$$C_p = (v_1, \dots, v_{t_p}, (i_{p+1}, j_{p+1}), \dots, (i_s, j_s))$$

for some $v_1, \dots, v_{t_p} \in V(G_p) \cap V(G)$, with $t_0 = 0$ and $t_p > t_{p-1}$ for $p = 1, \dots, s$.

3. For each $p = 0, 1, \dots, s$, every vertex $v \in V(G_p)$ is non-adjacent to at least k consecutive vertices of C_p .

A cycle C with the desired properties will then be given by the last cycle C_s of the above sequence.

First, let us verify that we can start the sequence with C_0 . Properties 1 and 2 are easily seen to be true. For property 3, we need to show that every vertex $v \in V(G_0)$ is non-adjacent to at least consecutive k vertices of C_0 . Let v be a vertex of G_0 . We have

$$V(G_0) = V(C_0) \cup \bigcup_{(i,j) \in V(R) \setminus V(C_0)} V(i, j).$$

If $v \in V(C_0)$, there is nothing to show, as in this case $v \in V(R)$. (We know that every vertex of $V(R)$ is non-adjacent to at least k consecutive vertices of C_0 .) If $v \in V(G_0) \setminus V(C_0)$, then $v \in V(i, j)$ for some $(i, j) \in V(R) \setminus V(C)$. Note that if in the grid R , the vertex (i, j) is non-adjacent to some $(i_p, j_p) \in V(C_0)$, then in G_0 , no vertex of $V(i, j)$ is adjacent to (i_p, j_p) . In particular, since (i, j) is non-adjacent in R to at least k consecutive vertices of C_0 , it follows that v is non-adjacent to at least k consecutive vertices of C_0 in this case too.

Suppose that the cycles C_0, \dots, C_p satisfying the above properties have already been constructed. We now show how to construct the cycle C_{p+1} from $C_p = (v_1, \dots, v_{t_p}, (i_{p+1}, j_{p+1}), (i_{p+2}, j_{p+2}), \dots, (i_s, j_s))$. Let us write

$$x = \begin{cases} v_{t_p}, & \text{if } p > 0; \\ (i_s, j_s), & \text{otherwise} \end{cases}$$

and

$$y = \begin{cases} (i_{p+2}, j_{p+2}), & \text{if } p < s - 1; \\ v_1, & \text{otherwise.} \end{cases}$$

Let $P = (x, v_{t_{p+1}}, \dots, v_{t_{p+1}}, y)$ be a shortest x - y path in G_{p+1} all the internal vertices of which belong to $V(i_{p+1}, j_{p+1})$. The cycle C_{p+1} is then obtained from C_p by deleting the vertex (i_{p+1}, j_{p+1}) (together with the two edges incident to it) and

joining the endpoints x and y of the so obtained path by P :

$$C_{p+1} = (v_1, \dots, v_{t_p}, v_{t_p+1}, \dots, v_{t_{p+1}}, (i_{p+2}, j_{p+2}), \dots, (i_s, j_s)).$$

We have to verify that the cycle C_{p+1} satisfies the properties 1-3. Property 1 follows directly from the construction of C_{p+1} (using the fact that C_p is a chordless cycle in G_p). Property 2 is straightforward. Finally, property 3 for C_{p+1} can be easily derived from property 3 for C_p : every vertex of G_{p+1} is either a vertex of G_p , or it belongs to $V(i_{p+1}, j_{p+1})$. In either case, the fact that every vertex of G_p is non-adjacent to at least k consecutive vertices of C_p implies that every vertex of G_{p+1} is non-adjacent to at least k consecutive vertices of C_{p+1} .

This completes the proof of the lemma. \square

Corollary 42. *For every apex graph H and every integer $k \geq 4$ there is a number N such that every connected H -minor-free (A_k, A_{k+1}, \dots) -free graph with no $128k$ -invisible chordless cycles has tree-width less than N .*

Proof. Let $N = f(H, k - 3, 512k + 1)$, where $f(\cdot)$ is the function given by Lemma 41. Suppose for contradiction that there exists an H -minor-free (A_k, A_{k+1}, \dots) -free graph G with no $128k$ -invisible chordless cycles, whose tree-width is at least $N = f(H, k - 3, 512k + 1)$. By Lemma 41, G contains a $(k - 3)$ -invisible chordless cycle C of length at least $512k + 1$. To avoid induced apples of order at least k , we conclude that every vertex in $V - C$ has at most four neighbours on C . Since C is not $128k$ -invisible, there exists a vertex v outside C that has at least one neighbour among every $128k$ consecutive vertices of C . Hence, $|V(C)| \leq 4 \times 128k = 512k$, a contradiction. \square

In conjunction with Lemma 35 and the fact the WIS problem is solvable in polynomial time for graphs of tree-width at most N [73], this implies our final result.

Theorem 43. *For every apex graph H and every integer k , there is a polynomial time algorithm for the WIS problem in the class of H -minor-free (A_k, A_{k+1}, \dots) -free graphs.*

4.4 Conclusion

The polynomial-time solvability of the MAXIMUM WEIGHT INDEPENDENT SET problem in the class of graphs without large apples generalises several results such as algorithms for claw-free and chordal graphs. It would be interesting to find out if similar generalisations can be made for related problems. Unfortunately this is not

the case for INDEPENDENT DOMINATION and MAXIMUM INDUCED MATCHING since both problems are already NP-complete for claw-free graphs.

Such a generalisation might yet be possible for the DOMINATING INDUCED MATCHING problem, since it has a polynomial solution on both chordal and claw-free graphs. Finding an answer to this question is an interesting open problem.

Chapter 5

Sparse Regular Induced Subgraphs in $2P_3$ -free Graphs

5.1 Introduction

Finding maximum regular induced subgraphs is a family of algorithmic graph problems containing several important representatives such as MAXIMUM INDEPENDENT SET, MAXIMUM CLIQUE, MAXIMUM INDUCED MATCHING. We call graphs of a fixed degree k *sparse regular graphs* and their complements *dense regular graphs*. For instance, the MAXIMUM INDEPENDENT SET and MAXIMUM INDUCED MATCHING problems deal with finding sparse regular graphs ($k = 0$ and $k = 1$, respectively), while the MAXIMUM CLIQUE problem deals with finding dense regular graphs ($k = 0$ in the complement to the graph).

Finding maximum sparse and maximum dense regular graphs is generally NP-hard for all non-negative integer values of k [83]. Moreover, this problem remains NP-hard even for many restricted graph families. On the other hand, for some specific values of k , in some particular graph classes the problem can be solved in polynomial time.

We have noted that the MAXIMUM INDEPENDENT SET problem ($k = 0$) has a polynomial-time solution in the class of claw-free graphs [84, 66, 85]. We observe that the MAXIMUM INDUCED MATCHING problem ($k = 1$) is NP-complete in claw-free graphs [86], while both problems ($k = 0$ and $k = 1$) are solvable in polynomial time for P_4 -free graphs. This is one of the very few classes defined by a *single* forbidden induced subgraph (we call such classes *monogenic*) where the problems are known to be polynomial-time solvable. Moreover, until recently, the class of P_4 -free graphs was the *only* known monogenic class with polynomial-time solvable

MAXIMUM INDUCED MATCHING problem. Only three such cases were known for the MAXIMUM INDEPENDENT SET problem¹: fork-free graphs [48], $claw + K_2$ -free graphs [87] and mK_2 -free graphs for any fixed value of m [69]. Recently, this restricted list of monogenic classes where the two problems admit polynomial-time solutions was extended by a new example, the class of $2P_3$ -free graphs [88].

The class of $2P_3$ -free graphs and some of its subclasses, such as $2K_2$ -free graphs, has received considerable attention both in the algorithmic community [89, 87, 88, 90, 91] and in the combinatorial community [92, 93]. In particular, finding maximum regular induced subgraphs in $2P_3$ -free graphs was studied in [88]. This paper observed a fundamental difference between finding sparse and dense graphs in this class and conjectured that finding a maximum regular induced subgraph H in a $2P_3$ -free graph G is polynomial if and only if H is sparse. The paper also verified both parts of the conjecture (sparse and dense) for small values of k . In particular, it proved that the MAXIMUM INDEPENDENT SET ($k = 0$) and MAXIMUM INDUCED MATCHING ($k = 1$) problems can be solved in the class of $2P_3$ -free graphs in polynomial time. Extending these results, in the present chapter we prove that finding a maximum k -regular induced subgraph is polynomially solvable in the class of $2P_3$ -free graphs for each fixed value of k . In other words, we prove the “sparse” part of the above conjecture. Moreover, our algorithm applies to *totally weighted* graphs, i.e. graphs in which both the vertices and the edges are assigned weights, and finds a solution of maximum total weight, if it exists. We emphasize that for $k > 0$, a k -regular induced subgraph does not necessarily exist.

We separate the work into two sections. We present a solution, separately, for maximum k -regular induced subgraphs containing a P_3 (Section 5.2) and maximum k -regular induced subgraphs which are P_3 -free (Section 5.3). Throughout the chapter we assume that k is a fixed constant.

5.2 Finding maximum k -regular graphs containing a P_3

We start by characterizing the structure of k -regular $2P_3$ -free graphs.

Lemma 44. *In a k -regular $2P_3$ -free graph H , at most one connected component is not a clique with $k + 1$ vertices. Moreover, if H does contain a component which is not a clique, then this component has at most $1 + \sum_{i=0}^4 (k - 1)^i k$ vertices.*

¹We exclude from our consideration trivial extensions of solvable cases obtained by adding isolated vertices (if $k = 0$) or isolated edges (if $k = 1$) to the only forbidden induced subgraph.

Proof. Obviously, every connected graph which is not a clique must contain an induced P_3 . Therefore, if H is $2P_3$ -free, then at most one of its connected components is not a clique. Moreover, if H is k -regular, then every component of H which is a clique must contain $k + 1$ vertices.

The second part of the lemma follows readily from the fact that a $2P_3$ -free graph is clearly P_7 -free and a connected P_s -free graph with maximum degree k has at most $1 + \sum_{i=0}^{s-3} (k-1)^i k$ vertices. \square

This lemma suggests an easy way to find in a $2P_3$ -free graph a maximum k -regular induced subgraph containing a P_3 .

Theorem 45. *Let G be a totally weighted $2P_3$ -free graph. For each fixed k , the problem of finding in G a k -regular induced subgraph containing a P_3 of maximum total weight can be solved in polynomial time.*

Proof. Denote $p := 1 + \sum_{i=0}^4 (k-1)^i k$ and $n := |V(G)|$. In the first step, by inspecting each subset of size at most p in G we determine if G has a *connected* k -regular induced subgraph containing a P_3 and find all subgraphs of this form. The number of such subgraphs and the time bound on this step of the algorithm is $O(n^p)$. If a *connected* k -regular induced subgraph containing a P_3 is not found in G , then G has no k -regular induced subgraph containing a P_3 .

Now for each *connected* k -regular induced subgraph H of G containing a P_3 found in Step 1, we do the following. Delete from G the vertices of $V(H)$ and of $N_G(V(H))$ and denote the resulting graph by G' . Since H contains a P_3 and G is $2P_3$ -free, the graph G' is P_3 -free, i.e. every connected component of G' is a clique. Now by inspecting all subsets of size $k + 1$ in each connected component of G' we find in each of them a maximum k -regular induced subgraph (if there is any). This can be done in $O(n^{k+1})$ time. The union of such graphs together with H give a maximum k -regular induced subgraph containing H . By comparing these graphs for all graphs H found in Step 1, we find a maximum k -regular induced subgraph of G containing a P_3 . Thus, the total complexity time of the algorithm is $O(n^{p+k+1})$, which is a polynomial for a fixed k . \square

5.3 Finding maximum P_3 -free k -regular graphs

In this section, we show that for each fixed value of k the problem of finding a maximum P_3 -free k -regular induced subgraph in a $2P_3$ -free graph G can be solved

in polynomial time. According to Lemma 44, in this case a solution (if one exists) is a collection of disjoint cliques, each of size $k + 1$.

We solve the problem in two major steps. In the first step, we generate a family S of vertex subsets of the input graph G . The generated family satisfies the following properties:

- each inclusionwise maximal set of vertices inducing a k -regular P_3 -free subgraph in G is a subset of one of the members of the family S ,
- the number of generated subsets is bounded by a polynomial in $|V(G)|$ and all of them can be found in polynomial time,
- the structure of each subset in S is simple, which allows to solve the problem in each subset in polynomial time.

In the second step, an optimal solution in G is found by solving the problem in each of the subsets of the generated family.

5.3.1 Step 1: generation of the family S

The first step is implemented in Algorithm FG (Family Generation) below. In the description of the algorithm we refer to four specific types of graphs. For a natural number k , we denote by

- Δ_i^1 ($i = 1, \dots, k$) the graph obtained from a clique C of size $k + 1$ by adding a vertex with exactly i neighbours in the clique,
- Δ_i^2 ($i = 1, \dots, k + 1$) the graph obtained from two disjoint cliques C and C' of size $k + 1$ by adding a vertex which dominates one clique and has exactly i neighbours in the other,
- Δ^3 the graph obtained from a clique C of size $k + 1$ by adding two adjacent vertices, one of which dominates the clique and one of which has no neighbours in the clique,
- Δ^4 the graph obtained from a clique C of size $k + 1$ by adding two non-adjacent vertices, both of which dominate the clique.

Also, by G_i we denote the graph induced by the vertices v_1, v_2, \dots, v_i .

Algorithm FG**Input:** a $2P_3$ -free graph G with vertex set $V(G) = \{v_1, v_2, \dots, v_n\}$.**Output:** a family S of subsets of $V(G)$. $S := \{\emptyset\}$ For $i = 1, \dots, n$, do

begin

1. [Extension of some members of S]For each $H \in S$,If there is a clique $C \subseteq V(G_i)$ of size $k + 1$ containing v_i such that $H \cup C$ induces a P_3 -free graph,then extend C to *any maximal* clique $C' \subseteq V(G_i)$ such that $H \cup C'$ induces a P_3 -free graph and add C' to H .2. [Addition of new members to S]2.1. For $j = 1, \dots, k$,For each induced Δ_j^1 in G_i containing vertex v_i in its clique C , $H := C \cup A_{G_i}(V(\Delta_j^1))$, $S := S \cup \{H\}$.2.2. For $j = 1, \dots, k + 1$,For each induced Δ_j^2 in G_i containing vertex v_i in one of its cliques C, C' , $H := C \cup C' \cup A_{G_i}(V(\Delta_j^2))$, $S := S \cup \{H\}$.2.3. For each induced Δ^3 in G_i containing vertex v_i in its clique C , $H := C \cup A_{G_i}(V(\Delta^3))$, $S := S \cup \{H\}$.2.4. For each induced Δ^4 in G_i containing vertex v_i in its clique C , $H := C \cup A_{G_i}(V(\Delta^4))$, $S := S \cup \{H\}$.

end

Lemma 46. *Let G be a $2P_3$ -free graph and S the family of subsets of $V(G)$ produced by Algorithm FG. Then:**(i) each member of S induces a P_3 -free subgraph of G ,**(ii) each inclusionwise maximal P_3 -free k -regular induced subgraph of G is contained in some member of S .**Proof.* A member of S is created either in the initialization step as the empty set or in Step 2 of some loop. Each member of S created in Step 2 is the union of a clique

and the antineighbourhood set of an induced subgraph containing a P_3 . Since G is $2P_3$ -free, this antineighbourhood set must be P_3 -free. Therefore, each member of S created in Step 2 induces a P_3 -free graph. In Step 1, a member of S is extended only if the extension preserves its P_3 -freeness. This proves the first part of the lemma.

To prove the second part, let us denote by S_i the content of the family S after i loops of the algorithm. We will show that any maximal P_3 -free k -regular induced subgraph of G_i is contained in a member H of S_i . The proof continues by induction on $i = 1, \dots, n$. For $i = 1$, the statement is trivial. Now we assume the statement holds for $i - 1$ and prove that it holds for i .

Let I be a maximal P_3 -free k -regular induced subgraph of G_i . If $v_i \notin I$, then by the induction assumption, I is contained in some member of S_{i-1} , and thus of S_i , since each member of S_{i-1} is a (not necessarily proper) subset of a member of S_i .

From now on, we assume $v_i \in I$. Let C be the connected component of I containing v_i . By the induction assumption, $I \setminus C$ is contained in some member of S_{i-1} . The extension of that member produced in Step 1 of the algorithm will be denoted by H (possibly H coincides with that member). By the first part of the lemma, we know that H induces a P_3 -free graph in G . If all vertices of C belong to H , then I is contained in H and we are done.

Therefore, we assume that at least one vertex of C does not belong to H . This means that $H \cup C$ contains an induced P_3 (since otherwise C must be added to H according to Step 1). We split the rest of the proof into cases depending on whether or not H and C have a common vertex.

Case 1: $H \cap C \neq \emptyset$.

Let K be the connected component of H containing $H \cap C$. Since H induces a P_3 -free graph, K is a clique.

1.1: $C \cup K$ is not a clique. Then let x be a vertex in $K \setminus C$ which has a non-neighbour in C . Denote by j the number of neighbours of x in X . Then $j \geq 1$ (since $K \cap C$ is not empty) and $j < k + 1$ (since x has a non-neighbour in C). Since $C \cup \{x\}$ induces a Δ_j^1 , we know that $C \cup A_{G_i}(C \cup \{x\})$ is a member of S_i produced in Step 2.1, and this subset of $V(G)$ clearly contains I .

1.2: $C \cup K$ is a clique. Since $H \cup C$ is not P_3 -free, a vertex of C must have a neighbour y in another clique (component) K' of H . Let j be the number of neighbours of y in C . Note that y cannot dominate C , because $K \cap C$ is not empty and y cannot be adjacent to a vertex in K . Thus, $1 \leq j \leq k$.

- 1.2.1: If $|K' \cap A_{G_i}(C)| \geq k + 1$, then there is at least one clique of size $k + 1$ contained in $K' \cap A_{G_i}(C)$. For each clique C' of this type, the set $C \cup C' \cup \{y\}$ induces a Δ_j^2 . Therefore, for each clique C' , the set $C \cup C' \cup A_{G_i}(C \cup C' \cup \{y\})$ is a member of S_i produced in Step 2.2. Clearly, one of these members contains I .
- 1.2.2: If $|K' \cap A_{G_i}(C)| \leq k$, then I is contained in $C \cup A_{G_i}(C \cup \{y\})$, which is a member of S_i , because $C \cup \{y\}$ induces a Δ_j^1 .

Case 2: $H \cap C = \emptyset$.

Since $H \cup C$ induces a graph containing a P_3 , we know that a vertex of C has a neighbour $x \in H$. Let K be the component (clique) of $G[H]$ containing x , and let $j \geq 1$ be the number of neighbours of x in C .

- 2.1: If $|K \cap A_{G_i}(C)| \geq k + 1$, then there is at least one clique of size $k + 1$ contained in $K \cap A_{G_i}(C)$. For each clique C' of this type, the set $C \cup C' \cup \{x\}$ induces a Δ_j^2 . Therefore, for each clique C' , the set $C \cup C' \cup A_{G_i}(C \cup C' \cup \{x\})$ is a member of S_i produced in Step 2.2. Clearly, one of these members contains I .

- 2.2: If $|K \cap A_{G_i}(C)| \leq k$, we further split the analysis into two subcases as follows.

2.2.1: $K \cup C$ is not a clique.

- 2.2.1.1: If all vertices of C have the same neighbourhood in K , then x dominates C . On the other hand, since $K \cup C$ is not a clique, there must exist a vertex $y \in K$ with no neighbours in C . Then $C \cup \{x, y\}$ induces a Δ^3 , and hence the set $C \cup A_{G_i}(C \cup \{x, y\})$ is a member of S_i generated in Step 2.3 of the algorithm, and this set clearly contains I .

- 2.2.1.2: If not all vertices of C have the same neighbourhood in K , then there must exist a vertex $y \in K$ which has both a neighbour and a non-neighbour in C . Then $C \cup \{y\}$ induces a Δ_j^1 , and hence the set $C \cup A_{G_i}(C \cup \{y\})$ is a member of S_i generated in Step 2.1 of the algorithm, and this set clearly contains I .

- 2.2.2: $K \cup C$ is a clique. Since $H \cup C$ induces a graph containing a P_3 , we know that a vertex of C has a neighbour $y \in H$ in a component (clique) K' of $G[H]$ different from K .

- 2.2.2.1: $|K' \cap A_{G_i}(C)| \geq k + 1$. This case can be settled similarly to Case 2.1.

- 2.2.2.2: $|K' \cap A_{G_i}(C)| \leq k$. If y dominates C , then $C \cup \{x, y\}$ induces a Δ^4 . In this case, the set $C \cup A_{G_i}(C \cup \{x, y\})$ is a member of S_i generated

in Step 2.4 of the algorithm, and this set contains I . If y does not dominate C , then $C \cup \{y\}$ induces a Δ_j^1 (where j is the number of neighbours of y in C), in which case the set $C \cup A_{G_i}(C \cup \{y\})$ is a member of S_i generated in Step 2.1 and containing I .

The lemma is proved. □

Lemma 47. *Let G be a $2P_3$ -free graph with n vertices and m edges and let S be the family of subsets of $V(G)$ produced by Algorithm FG. Then S contains $O(n^{2k+1})$ subsets and this family can be computed in time $\max\{O(n^{2k+2}), O(mn^{2k+1})\}$, which is also the time bound of Algorithm FG.*

Proof. New members of the family S are created in Step 2 of the algorithm. This step inspects subsets of vertices of size at most $2k + 1$, and each of these subsets can induce finitely many graphs analysed in Step 2. Therefore, S contains $O(n^{2k+1})$ members.

Each new member of S can be computed in Step 2 in $O(n)$ time. Therefore, the total complexity of Step 2 (i.e. complexity computed over all iterations of the algorithm) is $O(n^{2k+2})$. Steps 1, collectively, can be executed in $O(m|S|)$ time, i.e. in $O(mn^{2k+1})$ time by the above. Therefore, S can be computed in time $\max\{O(n^{2k+2}), O(mn^{2k+1})\}$, which is also the time bound of Algorithm GAMMA. □

5.3.2 Step 2: solving the problem

In step 2, we find a solution to the problem by solving it in each subset of the family S found in the first step.

Algorithm ABC

Input: a totally weighted $2P_3$ -free graph G .

Output: a k -regular P_3 -free induced subgraph of maximum total weight in G (if G has any).

- (A) Apply Algorithm FG to G to obtain a family S of subsets of $V(G)$.
- (B) For each $H \in S$, find a k -regular P_3 -free induced subgraph of maximum total weight in $G[H]$ by finding in each connected component of H a subset of $k + 1$ vertices inducing a subgraph of maximum total weight.
- (C) Among subgraphs found in Step (B), choose a subgraph of maximum total weight.

Theorem 48. *Algorithm ABC finds in a totally weighted $2P_3$ -free graph G with n vertices and m edges a k -regular P_3 -free induced subgraph of maximum total weight in time $\max\{O(n^{3k+3}), O(mn^{3k+2})\}$.*

Proof. The correctness of Algorithm ABC follows from Lemma 46. Now let us analyse its time complexity. From Lemma 47 we know that Step (A) can be executed in time $\max\{O(n^{2k+2}), O(mn^{2k+1})\}$, which is also a bound on the number of subsets in the family S produced in this step. By Lemma 46, each subset $H \in S$ induces a P_3 -free graph, i.e. H is a collection of disjoint cliques. Therefore, a k -regular P_3 -free induced subgraph of maximum total weight in $G[H]$ can be found in $O(n^{k+1})$ time by inspecting all subsets of size $k+1$ in each connected component of $G[H]$. As a result, the total time complexity of the algorithm is $\max\{O(n^{3k+3}), O(mn^{3k+2})\}$. \square

5.4 Conclusion

In this chapter, we proved that the problem of finding maximum *sparse* k -regular induced subgraphs can be solved in the class of $2P_3$ -free graphs in polynomial time for each fixed value of k . The complexity of finding maximum *dense* regular induced subgraphs (i.e. subgraphs whose complement is k -regular for a fixed k) in $2P_3$ -free graphs remains an open question, apart from three cases $k = 0, 1, 2$. For these three cases, the latter problem has been shown to be NP-complete in [88], where it was also conjectured that this problem is NP-complete for all values of k . Proving or disproving this conjecture is an interesting open problem.

It would also be interesting to find out whether the results of the present chapter can be extended to a larger *monogenic* class. In other words, it would be interesting to determine if there is an extension F of $2P_3$ such that the problem of finding maximum *sparse* k -regular induced subgraphs in the class of F -free graphs can be solved in polynomial time for each fixed value of k . However, no such extension is known even for $k = 0$, i.e. for the MAXIMUM INDEPENDENT SET problem. Currently, there are four minimal monogenic classes for which the complexity of this problem is unknown: P_5 -free graphs, $P_4 + P_2$ -free graphs, $P_3 + 2P_2$ -free graphs and *claw* + P_3 -free graphs (we do not mention classes defined by a forbidden induced subgraph F containing an isolated vertex v , because in this case the MAXIMUM INDEPENDENT SET problem can be easily reduced from F -free graphs to $F - v$ -free graphs). Determining the complexity of the problem in these classes of graphs is a challenging research problem

Chapter 6

Independent Domination in Finitely Defined Classes of Graphs

6.1 Introduction

The INDEPENDENT DOMINATING SET problem (or simply INDEPENDENT DOMINATION) is the problem of finding in a graph an independent dominating set of minimum cardinality. Clearly, an independent set is dominating if and only if it is *maximal*, i.e. not contained in any larger independent set. That is why INDEPENDENT DOMINATION is also known as MINIMUM MAXIMAL INDEPENDENT SET.

Computationally, this is a difficult problem, i.e. it is NP-hard. Moreover, the problem remains NP-hard under substantial restrictions, for instance, for graphs of bounded vertex degree, line graphs [94], bipartite graphs [95], etc. Of particular interest in this list of classes where the problem is NP-hard is the class of so called SAT-graphs. When restricted to the class of SAT-graphs, the problem becomes equivalent to SATISFIABILITY, the central problem of theoretical computer science. In other words, INDEPENDENT DOMINATION in the class of SAT-graphs can be viewed as SATISFIABILITY described in graph-theoretic terms. We discuss the relationship between the two problems in more detail in Section 6.2.

An important property of the class of SAT-graphs is that it can be characterized by *finitely many* forbidden induced subgraphs. We call classes of graphs possessing this property *finitely defined*. The family of finitely defined classes contains many classes of theoretical or practical importance, such as graphs of degree at most k (for a fixed k), line graphs, triangle-free graphs, cographs, split graphs,

etc. In some classes of this family the INDEPENDENT DOMINATING SET problem is NP-hard (which is the case, for instance, for graphs of degree at most $k \geq 3$, line graphs, triangle-free graphs), while in some others (such as cographs or split graphs) the problem can be solved in polynomial time. Various conditions under which the problem remains NP-hard in a finitely defined class have been revealed in [51]. We survey these conditions in Section 6.2.

In the present chapter, we look at the problem from the polynomial side, i.e. we study finitely defined classes for which the NP-hardness conditions revealed in [51] fail and derive for some of these classes polynomial-time algorithms. Some of our results deal with a more general version of the problem in which each vertex is assigned a positive integer, the *weight* of the vertex, and the problem consists in finding an independent dominating set of minimum total weight. We call it WEIGHTED INDEPENDENT DOMINATING SET problem or simply WEIGHTED INDEPENDENT DOMINATION. Let us observe that the complexity of the two versions of the problem may differ for graphs in the same class. For instance, INDEPENDENT DOMINATION for chordal graphs can be solved in polynomial time [96], while WEIGHTED INDEPENDENT DOMINATION in this class is NP-hard [52].

6.2 Preliminaries

It is known that in the class of bipartite graphs INDEPENDENT DOMINATION is NP-hard, while for split graphs the problem can be solved in polynomial time. Let us observe that split graphs can also be defined as graphs whose vertices can be partitioned into a clique and a graph of degree 0. Now let us consider an extension of this class to the class \mathcal{P} of graphs whose vertices can be partitioned into a clique and a graph of degree at most one. The class \mathcal{P} is of particular interest for the INDEPENDENT DOMINATING SET problem because it contains so-called SAT-graphs.

A SAT-graph is a graph G representing an instance of the SATISFIABILITY problem [91], whose vertices can be partitioned into a clique and a matching. The vertices in the clique part of G represent clauses and the vertices in the other part represent literals (i.e. variables and their negations). Each literal vertex x is connected to its negation \bar{x} and to the clauses containing it. It is not difficult to see that every independent dominating set I in G contains *exactly* one vertex in each pair x, \bar{x} . If I dominates (satisfies) each clause, the formula is satisfiable. Determining if G contains an independent dominating set satisfying all clauses is equivalent to (coincides with) SATISFIABILITY. Therefore, INDEPENDENT DOMINATION is NP-hard for graphs in the class \mathcal{P} .

Similarly to split graphs, the class \mathcal{P} can be characterized by *finitely many* forbidden induced subgraphs [91]. As we mentioned in the introduction, we call classes of graphs possessing this property *finitely defined*. Several sufficient conditions for the NP-hardness of INDEPENDENT DOMINATION in finitely defined classes have been identified in [51]. To describe some of them, let us recall the class \mathcal{S} , which is the class of forests of degree at most 3, in which every connected component contains at most one vertex of degree 3. Also, by \mathcal{T} we denote the class of line graphs of graphs in \mathcal{S} . In other words, \mathcal{S} is the class of graphs in which every connected component has the form $S_{i,j,k}$ represented in Figure 6.1(left) and \mathcal{T} is the class of graphs in which every connected component has the form $T_{i,j,k}$ represented in Figure 6.1(right). Observe that, by the definition of \mathcal{S} , i, j and k may take the value 0 for both of these graphs. For example, $S_{0,j,k}$ is a path on $j + k + 1$ vertices and its line graph, $T_{0,j,k}$, is a path on $j + k$ vertices.

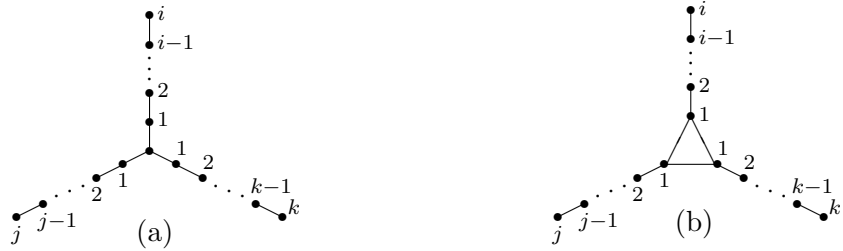


Figure 6.1: The graphs $S_{i,j,k}$ (a) and $T_{i,j,k}$ (b)

Theorem 49. [51] *Let M be a finite set. If $M \cap \mathcal{S} = \emptyset$ or $M \cap \mathcal{T} = \emptyset$, then the INDEPENDENT DOMINATING SET problem is NP-hard in the the class of graphs containing no graph in M as an induced subgraph.*

In the family of finitely defined classes, of particular interest are *monogenic* classes, i.e. classes defined by a single forbidden induced subgraph H . Under the assumption that $P \neq NP$, Theorem 49 implies that INDEPENDENT DOMINATION is polynomial-time solvable in the class of H -free graphs *only if* $H \in \mathcal{S} \cap \mathcal{T}$. It is not difficult to see that the intersection $\mathcal{S} \cap \mathcal{T}$ consists of linear forests, i.e. graphs every connected component of which is a path. Let us denote this class by \mathcal{L} and let us discuss the complexity of the problem in H -free graphs for some small graphs $H \in \mathcal{L}$.

For $H = P_3$, the problem is trivial, since P_3 -free graphs are precisely the graphs every connected component of which is a clique. In this class, the problem

can be solved in linear time even for weighted graphs (by choosing a vertex of minimum weight in each connected component).

The class of P_4 -free graphs is much richer but still simple enough to solve the WEIGHTED INDEPENDENT DOMINATING SET problem. In particular, the clique-width of P_4 -free graphs is at most 2, and hence the problem can be solved in the class of P_4 -free graphs in linear time [33].

A polynomial-time solution to WEIGHTED INDEPENDENT DOMINATION in the class of $2K_2$ -free graphs follows from the fact that these graphs contain only polynomially many maximal independent sets, which was proved by Farber in [97].

In the class of $2P_3$ -free graphs the problem is NP-hard, because $2P_3$ does not belong to the class \mathcal{P} defined above. The only monogenic class between $2K_2$ -free and $2P_3$ -free graphs is the class of $P_2 + P_3$ -free graphs. The complexity status of the problem in this class was open for a long time. In [87], Lozin and Mosca claimed that graphs in this class admit a polynomial-time solution for the weighted version of the problem. However, the proof contains a mistake which is crucial even for unweighted graphs. In this section, we partly correct this mistake by showing that the problem can be solved in polynomial time for unweighted $P_2 + P_3$ -free graphs. The complexity of WEIGHTED INDEPENDENT DOMINATION in the class of $P_2 + P_3$ -free graphs remains an open problem.

In addition to the results mentioned above, let us also observe that in the case of a single forbidden induced subgraph H we may assume without loss of generality that H does not contain isolated vertices. Indeed, let G be an H -free graph. Clearly, an optimal solution to the WEIGHTED INDEPENDENT DOMINATING SET problem in G can be found by first solving the problem in the antineighbourhood of each vertex $x \in V(G)$, then combining the result with x , and finally choosing a set of minimum weight. If H contains an isolated vertex v , then the antineighbourhood of each vertex in G induces an $(H - v)$ -free graph. Therefore, the complexity of the problem in the class of H -free graphs is polynomially equivalent to its complexity in the class of $(H - v)$ -free graphs.

This observation together with the above discussion show that the complexity of INDEPENDENT DOMINATION in the class of H -free graphs is known for every graph H with at most 5 vertices, except for $H = P_5$. The exceptional role of the P_5 -free graphs is also suggested by the fact that the situation is similar for the closely related problem of finding an independent set of maximum size. The complexity status of this problem in the class of P_5 -free graphs is unknown and this question remains open in spite of multiple attempts to answer it. Recently, it was shown in [98] that there is a sub-exponential algorithm to find a maximum

independent set in a P_5 -free graph. Also, numerous results have been obtained for the maximum independent set problem in subclasses of P_5 -free graphs (see e.g. [99, 100, 101, 102, 103, 104]). These results exploit various techniques, such as modular decomposition, augmenting graphs, generation of maximal independent sets. In the present chapter, we employ or adapt these techniques to deal with the INDEPENDENT DOMINATING SET problem in subclasses of P_5 -free graphs. The main part consists of three subsections devoted to the three techniques mentioned above. In the concluding subsection, we summarize the new results and list a number of open problems.

6.3 Generation of maximal independent sets

In [97], Farber proved that $2K_2$ -free graphs have polynomially many maximal independent sets. Inspired by his proof, one can develop the following algorithm for the generation of all maximal independent sets in a $2K_2$ -free graph. In the description of the algorithm we use the following notation. Given a graph G with vertex set $V(G) = \{v_1, v_2, \dots, v_n\}$, we denote by G_i the subgraph of G induced by vertices v_1, v_2, \dots, v_i .

Algorithm GENERATION-0

Input: a graph G with vertex set $V(G) = \{v_1, v_2, \dots, v_n\}$.

Output: a family S of subsets of $V(G)$.

```

 $S := \{\emptyset\}$ 
For  $i = 1, \dots, n$ , do
begin
  1. [Extension of some members of  $S$ ]
    For each  $H \in S$ ,
      If  $N_{G_i}(v_i) \cap H = \emptyset$ ,
        then  $H := H \cup \{v_i\}$ .
  2. [Addition of new members to  $S$ ]
    For each  $u \in N_{G_i}(v_i)$ ,
       $H := \{v_i\} \cup A_{G_i}(\{v_i, u\})$ 
       $S := S \cup \{H\}$ .
end

```

From the description of Algorithm GENERATION-0 it follows that if G is a $2K_2$ -free graph, then every member of the family S produced by the algorithm is a

maximal independent set in G . Moreover, according to Farber's argumentation, S contains all maximal independent sets of G , which can be proved by induction on i . Also, it is not difficult to see that the algorithm runs in $O(n^2)$ time, which is also a bound on the number of maximal independent sets in a $2K_2$ -free graph with n vertices. Therefore, the WEIGHTED INDEPENDENT DOMINATING SET problem can be solved for $2K_2$ -free graphs in polynomial time. In the next three subsections, we use the basic idea suggested by Algorithm GENERATION-0 to extend polynomial-time solvability of the (WEIGHTED) INDEPENDENT DOMINATING SET problem to larger classes of graphs.

6.3.1 Independent domination in $P_2 + P_3$ -free graphs

The class of $P_2 + P_3$ -free graphs is an extension of the class of $2K_2$ -free graphs. The structure of $P_2 + P_3$ -free graphs is more complex than that of $2K_2$ -free graphs. In particular, $P_2 + P_3$ -free graphs may have exponentially many maximal independent sets. This makes the problem much harder and we solve it only for unweighted graphs. Our solution is based on a tricky elaboration of Algorithm GENERATION-0. This elaboration is presented as Algorithm GENERATION-1 below.

Similarly to Algorithm GENERATION-0, the new algorithm also generates a family S of vertex subsets of the input graph. However, this time to each set $H \in S$ the algorithm also assigns a special vertex $u(H)$ which does not belong to H . As before, given a graph G with vertex set $V(G) = \{v_1, v_2, \dots, v_n\}$, we denote by G_i the subgraph of G induced by vertices v_1, v_2, \dots, v_i . Also, for a subset $U \subset V(G)$, we denote by U_0 the set of isolated vertices in $G[U]$. Without loss of generality we assume that the input graph G has no isolated vertices, because all these vertices must belong to any optimal solution. We also assume that the vertices of G are ordered so that v_1 is adjacent to v_2 . In the beginning of the algorithm, the family S includes only the set $\{v_1\}$ and the special vertex assigned to this set is $u(\{v_1\}) = v_2$.

Algorithm GENERATION-1

Input: a graph G with vertex set $V(G) = \{v_1, v_2, \dots, v_n\}$.

Output: a family S of subsets of $V(G)$.

$S := \{\{v_1\}\}$ with $u(\{v_1\}) = v_2$.

For $i = 2, \dots, n$, do

begin

1. [Extension of some members of S]

For each $H \in S$,

If $(N_{G_i}(v_i) \cap H = \emptyset)$ OR $((N_{G_i}(v_i) \cap H_0 = \emptyset$ AND $(u(H), v_i) \notin E(G))$,
then $H := H \cup \{v_i\}$.

2. [Addition of new members to S]

2.1. For each pair of vertices v_i, u inducing in G_i a P_2 ,

$$H := \{v_i\} \cup A_{G_i}(\{v_i, u\}), u(H) := u, S := S \cup \{H\}.$$

2.2. For each triple of vertices v_i, u, w inducing in G_i a P_3 with u being the center,

$$H := \{v_i, w\} \cup A_{G_i}(\{v_i, u, w\}), u(H) := u, S := S \cup \{H\}.$$

end

Lemma 50. *Let G be a $P_2 + P_3$ -free graph with vertex set $V(G) = \{v_1, v_2, \dots, v_n\}$ and let S be the family of subsets of $V(G)$ produced by Algorithm GENERATION-1. Then*

(i) *for each $H \in S$, there is an edge vu of G such that $H - H_0 \subseteq A_G(\{v, u\})$,*

(ii) *for each maximal independent set I in G , there is a set $H \in S$ such that $I \subseteq H$.*

Proof. To prove (i), let us observe that when a set H is created in Step 2 of loop i , it obviously satisfies the inclusion $H - H_0 \subseteq A_G(\{v, u\})$ with $v = v_i$ and $u = u(H)$. After the creation, the set H can be extended in Step 1 of further loops by adding new vertices, but we emphasize that once a vertex appears in H as an isolated vertex, it always remains isolated since by definition of Step 1 no new vertex can be adjacent to an isolated vertex of $G[H]$. In particular, $v = v_i$ is always isolated. Therefore, when a new vertex x is added to H , it is either isolated (if x has no neighbours in H) or belongs to the antineighbourhood of vu (if x has no neighbours in H_0 and x is not adjacent to $u = u(H)$).

To prove (ii), let us denote by S_i the content of the family S at the end of loop i . The proof will be given by induction on $i = 2, \dots, n$. For $i = 2$, the proposition is trivial.

Let I be a maximal independent set in the subgraph G_i . If $v_i \notin I$, then, by the induction hypothesis, there is a set $H \in S_{i-1}$ such that $I \subseteq H$. According to the algorithm, S_i contains either H or $H \cup \{v_i\}$. In either case, the proposition is true.

Now let $v_i \in I$ and let I' denote $I - \{v_i\}$. By the induction hypothesis, there is a set $H \in S_{i-1}$ such that $I' \subseteq H$. If v_i has no neighbours in H , then $H \cup \{v_i\} \in S_i$ (see Step 1 of the algorithm) and $I \subseteq H \cup \{v_i\}$.

Suppose now that v_i has a neighbour $u \in H$. If u has no neighbours in I' ,

then I is a subset of the set $\{v_i\} \cup A_{G_i}(\{v_i, u\})$ produced in Step 2.1 of the algorithm at loop i . Assume now that u has a neighbour w in I' . We claim that

(*) w is the *only* neighbour of u in I' . To show this, let us observe that according to the algorithm, vertex $u(H)$ does not belong to H and has at least one neighbour in H , say x . Moreover, every neighbour of $u(H)$ in H , including x , is an isolated vertex in the subgraph $G[H]$. Therefore, neither $u(H)$ nor x is adjacent to u or to any neighbour of u in H . This implies that if u has two neighbours in I' , then these two neighbours together with u , $u(H)$ and x create an induced $P_2 + P_3$ in G . Since this is impossible, we conclude that w is the only neighbour of u in I' .

According to this claim, I is a subset of the set $\{v_i, w\} \cup A_{G_i}(\{v_i, u, w\})$ produced in Step 2.2 of the algorithm at loop i . \square

Lemma 51. *For a graph G with n vertices, Algorithm GENERATION-1 runs in time $O(n^5)$ and the family S produced by this algorithm contains $O(n^3)$ subsets of $V(G)$.*

Proof. Algorithm GENERATION-1 creates new members of S by inspecting vertex subsets of G of size at most 3 and for each such a subset it adds at most one new member to S . Therefore, $|S| = O(n^3)$.

Each new member of S can be computed in Step 2 in $O(n)$ time. Therefore, the total complexity of Step 2 (i.e. complexity computed over all iterations of the algorithm) is $O(n^4)$. Steps 1, collectively, can be executed in $O(m|S|)$ time, where m is the number of edges of G . Therefore, the total time complexity of the algorithm can be estimated as $O(n^5)$. \square

According to Lemma 50, if G is a $P_2 + P_3$ -free graph, then every set H in the family S induces a P_3 -free graph, i.e. a graph every connected component of which is a clique. Also, for every maximal independent set I in G there is a set $H \in S$ containing I . Clearly, I must be maximal in $G[H]$ as well. However, not every independent set which is maximal in $G[H]$ is also maximal in G . Theoretically, we need to check *all* maximal independent sets in $G[H]$ in order to identify those of them which are maximal in G and then to choose a smallest one, but unfortunately this task cannot be implemented in polynomial time even for P_3 -free graphs, because these graphs can contain exponentially many maximal independent sets. On the other hand, fortunately, all maximal independent sets in $G[H]$ have the same size which is equal to the number of connected components of $G[H]$. Therefore, to solve the INDEPENDENT DOMINATING SET problem we need to find at most one maximal

independent set in $G[H]$ which is also maximal in G . The peculiar structure of the generated sets in the family S allows us to implement this task in polynomial time.

Let us observe that the family S contains sets of two types. If a set $H \in S$ was created in Step 2.2, then H is an independent set both at the time of its creation and throughout the algorithm (since no vertex adjacent to an isolated vertex of $G[H]$ can be added to H). Verifying if H is a maximal independent set in G is a trivial task.

From now on, we assume that H was created in Step 2.1, i.e. it has the form $H := \{v\} \cup A_{G_i}(\{v, u\})$ with $v = v_i$ and $u = u(H)$. We repeat that by H_0 we denote the set of isolated vertices of H . Also, we introduce the following notation: $A = H - H_0$. Clearly, every vertex of H_0 , including v , belongs to every maximal independent set in $G[H]$, and by Lemma 50, $A \subseteq A_G(\{v, u\})$. Thus, every connected component of $G[A]$ is clique, and hence every maximal independent set in $G[H]$ contains exactly one vertex from each component of $G[A]$. Clearly, every maximal independent set in $G[H]$ dominates all vertices of G except possibly some neighbours of u . We denote by U the set of neighbours of u each of which is dominated by *every* maximal independent set in $G[H]$, and by W the set of the remaining neighbours of u . In other words, a neighbour x of u belongs to W if and only if x has no neighbours in H_0 and has a non-neighbour in each connected component of $G[A]$. The above discussion leads to the following conclusion.

Proposition 52. *A set H created in Step 2.1 contains an independent set dominating G if and only if A contains an independent set dominating W .*

In what follows, we show that the problem of determining if A contains an independent set dominating W can be solved in polynomial time. We start by showing that any two adjacent vertices of A dominate W .

Lemma 53. *If ab is an edge in $G[A]$, then $W \subseteq N(a) \cup N(b)$.*

Proof. If a vertex $x \in W$ is adjacent neither to a nor to b , then a, b, x, u, v induce a $P_2 + P_3$. □

Lemma 53 shows that if $Q = \{q_1, \dots, q_p\}$ is a component (clique) in $G[A]$ and $W_i = W \cap A_G(q_i)$, then $\{W_1, \dots, W_p\}$ is a partition of W , i.e. a collection of pairwise disjoint subsets of W the union of which coincides with W . We denote this partition by $\mathcal{P}(Q)$.

Let us denote the components of $G[A]$ by Q_1, \dots, Q_t . Then an element of $\mathcal{P}(Q_1) \times \dots \times \mathcal{P}(Q_t)$ is an ordered list (Y_1, \dots, Y_t) of subsets of W such that $Y_i \in \mathcal{P}(Q_i)$.

Lemma 54. *The set A contains a maximal independent set dominating W if and only if there is an element $(Y_1, \dots, Y_t) \in \mathcal{P}(Q_1) \times \dots \times \mathcal{P}(Q_t)$ such that*

$$Y_1 \cap \dots \cap Y_t = \emptyset.$$

Proof. Let I be a maximal independent set in $G[A]$, and let Y_i be the set of vertices of W that are non-adjacent to the only vertex of $I \cap Q_i$. For an arbitrary subset $X \subseteq W$, we denote $\bar{X} = W \setminus X$. Then \bar{Y}_i is the set of vertices of W that are adjacent to the only vertex of $I \cap Q_i$. Therefore, I dominates W if and only if $\bar{Y}_1 \cup \dots \cup \bar{Y}_t = W$. By De Morgan's law, this holds if and only if $Y_1 \cap \dots \cap Y_t = \emptyset$. \square

Lemma 55. *Given a set W of n elements and a number of partitions $\mathcal{P}_1, \dots, \mathcal{P}_t$ of W , one can check if there is an element $(Y_1, \dots, Y_t) \in \mathcal{P}_1 \times \dots \times \mathcal{P}_t$ such that $Y_1 \cap \dots \cap Y_t = \emptyset$ in $O(n^2)$ time.*

Proof. Let us describe the set W and the t partitions $\mathcal{P}_1, \dots, \mathcal{P}_t$ of W as a rooted tree T . Each node of T will represent a subset of W and we describe T inductively as follows. The root of T represents W and the children of the root represent the subsets in the partition \mathcal{P}_1 . Now let v be a node of distance $0 < i < t$ from the root, and let $\mathcal{P}_{i+1} = \{W_1, \dots, W_p\}$. If v represents the empty set, then it has no children. If v represents a non-empty subset X of W , then v has p children representing the sets $X \cap W_1, \dots, X \cap W_p$.

From the definition of T it follows that if v is a node of distance i from the root, then the unique path connecting v to the root corresponds to a unique list of sets $Y_1 \in \mathcal{P}_1, \dots, Y_i \in \mathcal{P}_i$ such that $Y_1 \cap \dots \cap Y_i$ is the set represented by v . Therefore, the following claim holds.

- *There is an element $(Y_1, \dots, Y_t) \in \mathcal{P}_1 \times \dots \times \mathcal{P}_t$ such that $Y_1 \cap \dots \cap Y_t = \emptyset$ if and only if T has a node representing the empty set.*

From this claim it follows that the problem in question can be solved by recursively constructing the tree T starting from the root and finishing as soon as a node representing the empty set appears, or all t levels of the tree are constructed. Now let us estimate the worst time complexity of this approach.

For $i \geq 1$, let v be a node of T of distance $i - 1$ from the root and let p_i be the number of subsets in the partition \mathcal{P}_i . Without loss of generality we may assume that \mathcal{P}_i is a non-trivial partition, i.e. $p_i \geq 2$, since otherwise it can be omitted. Denote by X the subset of W represented by v . If X is non-empty, then the children of v partition X into smaller subsets X_1, \dots, X_{p_i} . If none of them is empty, then partitioning of X is equivalent to placing $p_i - 1$ separators between

the elements of X . Therefore, if no node of T represents the empty set, then the total number of nodes in T is at most twice the number of separators that can be placed between the elements of W , i.e. at most $2(n-1)$. Thus, in the worst case the algorithm generates $O(n)$ subsets of W . Since each subset can be generated in $O(n)$ time, the algorithm can be executed in $O(n^2)$ time. \square

Theorem 56. *Given a $P_2 + P_3$ -free graph G with n vertices, one can find an independent dominating set of minimum cardinality in G in $O(n^5)$ time.*

Proof. By Lemma 51, in $O(n^5)$ time Algorithm GENERATION-1 generates a family S of $O(n^3)$ subsets of $V(G)$. By Lemma 50, for each maximal independent set I in G there is a set $H \in S$ containing I . By the same lemma, $H \in S$ induces a P_3 -free graph, and hence, all maximal independent sets in H have the same cardinality. By Proposition 52, Lemma 54 and Lemma 55 the problem of determining if $H \in S$ contains a maximal independent set dominating G can be solved in $O(n^2)$ time. Therefore, an independent dominating set of minimum cardinality in G can be found in $O(n^5)$ time. \square

6.3.2 Weighted independent domination in $(P_5, 2P_3)$ -free graphs

Similarly to $P_2 + P_3$ -free graphs, the class of $(P_5, 2P_3)$ -free graphs extends the class of $2K_2$ -free graphs. However, the structure of graphs in this extension is more specific and allows us to solve the weighted version of the problem. Again, we solve it by elaborating Algorithm GENERATION-0. This elaboration is presented as Algorithm GENERATION-2 below. As before, given a graph G with vertex set $V(G) = \{v_1, v_2, \dots, v_n\}$, we denote by G_i the subgraph of G induced by vertices v_1, v_2, \dots, v_i .

Algorithm GENERATION-2

Input: a graph G with vertex set $V(G) = \{v_1, v_2, \dots, v_n\}$. **Output:** a family S of subsets of $V(G)$.

$S := \{\emptyset\}$

For $i = 1, \dots, n$, do

begin

1. [Extension of some members of S]

 For each $H \in S$,

 If $H \cup \{v_i\}$ induces a P_3 -free graph,

then $H := H \cup \{v_i\}$.

2. [Addition of new members to S]

2.1. For each triple v_i, u, w of vertices inducing in G_i a P_3 with edges $v_i u, uw$,

$$H := \{v_i, w\} \cup A_{G_i}(\{v_i, u, w\}),$$

$$S := S \cup \{H\}.$$

2.2. For each triple v_i, u, w of vertices inducing in G_i a P_3 with edges $uv_i, v_i w$,

$$H := \{v_i\} \cup A_{G_i}(\{v_i, u, w\}),$$

$$S := S \cup \{H\}.$$

end

Lemma 57. *Let G be a $2P_3$ -free graph and S be the family of subsets of $V(G)$ produced by Algorithm GENERATION-2. Then:*

(i) *each member of S induces a P_3 -free subgraph of G ;*

(ii) *each maximal independent set of G is contained in some member of S .*

Proof. A member of S is created either by the initialization step as the empty set or in Step 2 of some loop. By definition of Step 2, since G is $2P_3$ -free, each member created in Step 2 induces a P_3 -free subgraph of G . Then, by definition of Step 1, a member of S is extended only if this extension preserves its P_3 -freeness. This proves the first part of the lemma.

To prove the second part, let us denote by S_i the content of the family S after i loops of the algorithm. We will show that for any maximal independent set I of G_i , there is a member $H \in S_i$ such that $I \subseteq H$. The proof is by induction on $i = 1, \dots, n$. For $i = 1$, the family S_i consists of the single set $\{v_1\}$, and this is obviously the only maximal independent set in the graph G_1 . Now let us assume that the statement holds for $i - 1$ and prove that it holds for i .

Let I be a maximal independent set in G_i . If $v_i \notin I$, then by the induction assumption I is contained in some member of S_{i-1} and thus of S_i , since each member of S_{i-1} is contained (properly or not) in some member of S_i .

Assume now that $v_i \in I$. Then by the induction assumption $I \setminus \{v_i\}$ is contained in some member H of S_{i-1} . We know (by part (i) of the lemma) that H induces a P_3 -free graph in G . We split the rest of the proof into two cases depending whether or not $H \cup \{v_i\}$ induces a P_3 -free graph.

Case 1: $H \cup \{v_i\}$ induces a P_3 -free graph. Then I is contained in $H \cup \{v_i\}$ which is a member of S_i obtained by extending the set H in Step 1 of the algorithm at loop i .

Case 2: $H \cup \{v_i\}$ does not induce a P_3 -free graph. Then an induced P_3 can appear in $G[H \cup \{v_i\}]$ in one of the following two ways.

2.1: v_i has a neighbour u in a component (clique) K of $G[H]$ but does not dominate K . Then, since $G[H]$ is P_3 -free, I is contained in one of the subsets of the family $\{\{v_i, w\} \cup A_{G_i}(v_i, u, w) : w \in K \setminus N(v_i)\}$. Each subset of this family is generated in Step 2.1 of the algorithm at loop i and hence each of them belongs to S_i .

2.2: v_i dominates at least two components (cliques) K, K' of $G[H]$. Then, since $G[H]$ is P_3 -free, I is contained in one of the subsets of the family $\{\{v_i\} \cup A_{G_i}(u, v_i, w) : u \in K, w \in K'\}$. Each subset of this family is generated in Step 2.2 of the algorithm at loop i and hence each of them belongs to S_i .

The lemma is proved. □

Lemma 58. *Let G be a $2P_3$ -free graph with n vertices and let S be the family of subsets of $V(G)$ produced by Algorithm GENERATION-2. Then S contains $O(n^3)$ subsets and this family can be computed in time $O(n^5)$.*

Proof. New members of the family S are created in Step 2 of the algorithm. This step inspects triples of vertices, and each triple can create at most 3 different induced P_3 's. Therefore, S contains $O(n^3)$ members.

Each new member of S can be computed in Step 2 in $O(n)$ time. Therefore, the total complexity of Step 2 (i.e. complexity computed over all iterations of the algorithm) is $O(n^4)$. Steps 1, collectively, can be executed in $O(m|S|)$ time, where m is the number of edges of G . Therefore, the total time complexity of the algorithm can be estimated as $O(n^5)$. □

We now proceed to the second phase of the procedure to solve the WEIGHTED INDEPENDENT DOMINATING SET problem in the class of $(P_5, 2P_3)$ -free graphs. In this phase, we solve the problem separately for each member H of the family S produced in Phase 1.

From Lemma 57 we know that for every maximal independent set I in G there is a set $H \in S$ containing I , and clearly, I is also maximal in $G[H]$. However, the converse is not necessarily true, i.e. not every maximal independent set in $G[H]$ is also maximal in G . Since $G[H]$ is P_3 -free, all maximal independent sets in $G[H]$ have the same cardinality and contain exactly one vertex in each connected component of H . In what follows we show that the problem of determining if there is a maximal independent set in $G[H]$ dominating G can be solved for $(P_5, 2P_3)$ -free graphs in polynomial time.

Lemma 59. *Let G be a vertex-weighted $(P_5, 2P_3)$ -free graph with n vertices and let H be a member of the family S produced by Algorithm GENERATION-2. One can determine if $G[H]$ contains a maximal independent set dominating G in $O(n^2)$ time. Moreover, if such a set exists, one can find a maximal independent set of minimum weight in $G[H]$ dominating G in the same time.*

Proof. Let us partition the vertices outside H into three subsets as follows:

D is the set of vertices not in H each of which is dominated by *any* maximal independent set in $G[H]$. Clearly, a vertex $v \notin H$ belongs to D if and only if there is a component (clique) in $G[H]$ each vertex of which is adjacent to v .

Z is the set of vertices not in H none of which is dominated by *any* maximal independent set in $G[H]$. Clearly, a vertex $v \notin H$ belongs to Z if and only if it has no neighbours in H .

T is the set of the remaining vertices of $V(G) \setminus H$, i.e. $T = V(G) \setminus (H \cup D \cup Z)$.

This partitioning can be implemented in $O(n^2)$ time in an obvious way. By definition, there is a maximal independent set in $G[H]$ dominating G if and only if $Z = \emptyset$ and there is a maximal independent set in $G[H]$ dominating T . So, we assume $Z = \emptyset$. In order to determine whether there is a maximal independent set in $G[H]$ dominating T , we observe the following:

- every vertex of T has neighbours in exactly one connected component of $G[H]$. Indeed, if a vertex $v \in T$ has a neighbour x in a component (clique) K of $G[H]$, then it also must have a non-neighbour y in K , since otherwise v belongs to D . Therefore, if v has a neighbour x' in one more component of $G[H]$, then it also has a non-neighbour y' in that component, but then x, y, v, x', y' induced a P_5 in G .

According to this observation the set T can be partitioned into subsets T_1, \dots, T_k , each containing neighbours in exactly one component of $G[H]$. Let us denote the component of $G[H]$ containing neighbours of T_i by Q_i . Then there is a maximal independent set in $G[H]$ dominating T if and only if there is a vertex in Q_i dominating T_i for each $i = 1, \dots, k$. Moreover, if such a vertex exists for each i , one can choose a vertex of Q_i of minimum weight dominating T_i in linear time. \square

Theorem 60. *Let G be a vertex-weighted $(P_5, 2P_3)$ -free graph with n vertices. One can find a minimum weight independent dominating set in G in $O(n^5)$ time.*

Proof. By Lemma 58, in $O(n^5)$ time Algorithm GENERATION-2 generates a family S of $O(n^3)$ subsets of $V(G)$. By Lemma 57, for each maximal independent set I in G there is a set $H \in S$ containing I . By Lemma 59, for each $H \in S$, the problem of determining if H contains a maximal independent set dominating G and finding such a set minimum weight can be solved in $O(n^2)$ time. Therefore, a minimum weight independent dominating set in G can be found in $O(n^5)$ time. \square

6.3.3 More subclasses P_5 -free graphs

Taking into account the exceptional role of P_5 -free graphs for the (WEIGHTED) INDEPENDENT DOMINATING SET problem, in this subsection we develop a tool that allows us to extend polynomial-time solvability of the problem from smaller classes to larger ones.

Theorem 61. *Let F be a graph and p a natural number. If the WEIGHTED INDEPENDENT DOMINATING SET problem can be solved in polynomial time for (P_5, F) -free graphs, then it can also be solved in polynomial time for $(P_5, F + pK_2)$ -free graphs.*

Proof. We prove the theorem by induction on p . For $p = 0$, there is nothing to prove. Therefore, we assume the problem is solvable in polynomial time for $(P_5, F + (p - 1)K_2)$ -free graphs and consider a $(P_5, F + pK_2)$ -free graph G with n vertices.

To solve the problem for G , we first apply to G Algorithm GENERATION-1 without Step 2.2. By analogy with Lemma 51, one can show that this simplification of Algorithm GENERATION-1 runs in time $O(n^4)$ and produces a family S containing $O(n^2)$ subsets of $V(G)$. Also, by analogy with Lemma 50 we show that

- (i) for each $H \in S$, there is an edge vu of G such that $H - H_0 \subseteq A_G(\{v, u\})$,
- (ii) for each maximal independent set I in G , there is a set $H \in S$ such that $I \subseteq H$.

Up to Claim (*), the proof is identical to that of Lemma 50, since up to this point the proof does not assume any specific structure of the input graph. To prove the rest, we assume that u has a neighbour $w \in I'$. According to the algorithm, vertex $u(H)$ does not belong to H and has exactly one neighbour in H , say x , and this neighbour is an isolated vertex in the subgraph $G[H]$. Therefore, x is different from u and w , and neither $u(H)$ nor x is adjacent to u or to w . We know that v_i has no neighbours among isolated vertices of $G[H]$, including x , since otherwise I is a subset of $\{v_i\} \cup A_{G_i}(\{v_i, x\})$ produced in Step 2.1 at loop i . Also, v_i is not adjacent to $u(H)$, since otherwise vertices $w, u, v_i, u(H), x$ induce a P_5 in G . But then the

algorithm adds v_i to H in Step 1, and hence $I \subset H$. This completes the proof of the first stage of the procedure for solving the problem for G , i.e. of the stage that generates a family S of subsets of $V(G)$ satisfying (i) and (ii).

Now we turn to the second stage of the procedure. At this stage, we determine for each $H \in S$ if H contains an independent set dominating G , and if so, find such a set of minimum weight.

Let H be a member of S , let $u = u(H)$ be the unique vertex associated with H which does not belong to H , and let v be a neighbour of u in H_0 . If H does not dominate G , then clearly no independent set in H dominates G . Now let us show that

(**) *if H dominates G , then any maximal independent set in $G[H]$ dominates G .*

Indeed, let I be a maximal independent set in $G[H]$, and assume by contradiction that there is a vertex $x \notin H$ which has no neighbours in I . Then x has no neighbours in H_0 , since every isolated vertex of $G[H]$ belongs to every maximal independent set in $G[H]$. In particular, x is not adjacent to v and hence $x \neq u$. Then x is adjacent to u , since otherwise $x \in A_G(\{v, u\})$, in which case x must be a member of H . By assumption, H dominates G and hence x has a neighbour $y \in H$. Let us denote by Y the connected component of $G[H]$ containing y and let z be any vertex of $I \cap Y$ (clearly any maximal independent set in $G[H]$ must have at least one vertex in any connected component of $G[H]$). Since x is adjacent to y and is non-adjacent to z , then any path connecting y to z in Y contains a pair of adjacent vertices exactly one of which is adjacent to x . But then these two vertices together with x, u, v induce a P_5 in G .

This claim reduces the problem to $G[H]$. All non-isolated vertices of $G[H]$ belong to $A_G(\{v, u\})$ and this set must induce a $(P_5, F + (p-1)K_2)$ -free graph, since G is $(P_5, F + pK_2)$ -free. By inductive assumption, the problem is polynomial-time solvable for $(P_5, F + (p-1)K_2)$ -free graphs. Therefore, it is solvable for each $G[H]$ in polynomial time. Since the number of sets in S is $O(n^2)$, we obtain a polynomial bound on the total complexity time of the outlined procedure. \square

6.4 Modular Decomposition

The idea of modular decomposition has been first described in the 1960s by Galilai [105], and also appeared in the literature under various other names such as

prime tree decomposition [106], *X-join decomposition* [107], or *substitution decomposition* [108]. This technique allows one to reduce many graph problems from arbitrary graphs to so-called *prime* graphs. In this section, we introduce terminology related to this technique, state basic properties of modules in graphs, and describe a procedure implementing the reduction to prime graphs for the WEIGHTED INDEPENDENT DOMINATING SET problem. Then we apply modular decomposition to solve the problem in some subclasses of P_5 -free graphs.

Let $G = (V, E)$ be a graph, U a subset of its vertices and v a vertex of G outside U . We say that v *distinguishes* U if v has both a neighbour and a non-neighbour in U . If v does not distinguish U , then v is adjacent either to each vertex of U , in which case we say that v is *complete* to U , or to no vertex of U , in which case we say that v is *anticomplete* to U . Similarly, given two disjoint subsets $U \subset V(G)$ and $W \subset V(G)$, we say that U is complete or anticomplete to W depending on whether each vertex of U is complete or anticomplete to W .

A set of vertices $U \subseteq V(G)$ is called a *module* of G if it is indistinguishable by the vertices outside of U . Trivially, every vertex of a graph is a module and the set of all vertices of the graph is a module. Not surprisingly such modules are called *trivial*. A graph every module of which is trivial is called *prime*. A path P_k with at least $k \geq 4$ vertices is an example of a prime graph. Also, P_1 and P_2 are obviously prime graphs. However, P_3 is not prime, since the vertices of degree 1 in P_3 form a non-trivial module.

We say that a module is *proper* if it is different from the set of all vertices of the graph. A *maximal proper* module is a proper module which is not contained in any other proper module.

Modules enjoy many nice properties. We state the two most important ones in the following lemma. To make the section self-contained we provide the lemma with a short proof.

Lemma 62. *Let G be a graph.*

- (1) *Any two disjoint modules in G are either complete or anticomplete to each other.*
- (2) *If both G and \overline{G} are connected, then the maximal proper modules of G are pairwise disjoint.*

Proof. To prove (1), consider two disjoint modules U and W , and let $x \in U$. By definition, x is either complete or anticomplete to W . If x is complete to W , then every vertex of W is complete to U (since it has a neighbour in U), in which case

W and U are complete to each other. Similarly, if x is anticomplete to W , then W and U are anticomplete to each other.

To prove (2), let U and W be two maximal proper modules in G . Assume by contradiction they have a non-empty intersection.

Suppose there is a vertex $x \notin U \cup W$. Then by definition x must be either complete or anticomplete to U . If x is anticomplete to U , then x is anticomplete to the intersection $U \cap W$, and therefore to W , and hence to $U \cup W$. Similarly, if x is complete to U , then it is complete to $U \cup W$. Thus $U \cup W$ is a proper module, which contradicts the maximality of U and W . As a result we conclude that $U \cup W = V(G)$.

By the maximality of U and W we know that neither $U - W$ nor $W - U$ is empty. Let $x \in U - W$ and assume x is anticomplete to W . Then $W - U$ must be anticomplete to U , since otherwise any vertex of $W - U$ which has a neighbour in U distinguishes U . But then G is disconnected. Similarly, if x is complete to W , then \overline{G} is disconnected. This contradiction shows that U and W are disjoint. \square

The properties of modules stated in Lemma 62 allow a reduction of the WEIGHTED INDEPENDENT DOMINATING SET problem from general graphs to prime graphs. Informally, the reduction can be described as follows. If the input graph G is disconnected, then clearly an optimal solution in G is the union of optimal solutions in the connected components of G . Similarly, if the complement of G is disconnected, then solving the problem separately for each co-component of G (a connected component of \overline{G}) and taking a minimum solution we obtain an optimal solution for G . If both G and \overline{G} are connected, by Lemma 62 we partition G into maximal proper modules M_1, \dots, M_k and assume inductively that an optimal solution is available for each module. By contracting each module M_i to a single vertex m_i , we obtain a new graph G^0 which must be prime due to the maximality of the modules. To vertex m_i of this graph we assign the weight $\omega(m_i)$ equal to the weight of an optimal solution in the subgraph of G induced by M_i . Then solving the problem for G is equivalent to solving it for G^0 . A formal description of the procedure is presented as Algorithm WID below.

Algorithm WID(G)

Input: a vertex-weighted graph G .

Output: an independent dominating set S of minimum weight in G .

1. If $|V(G)| = 1$, set $S = V(G)$. Go to 7.

2. If G is disconnected, partition it into connected components M_1, \dots, M_k .
3. If \bar{G} is disconnected, partition G into co-components M_1, \dots, M_k .
4. If G and $\text{co-}G$ are connected, partition G into maximal proper modules M_1, \dots, M_k .
5. Construct a weighted graph G^0 from G by contracting each M_j ($j = 1, \dots, k$) to a single vertex m_j and set $\omega(m_j) := \omega(\text{WID}(G[M_j]))$.
6. Find in G^0 an independent dominating set S^0 of minimum weight and set

$$S = \bigcup_{m_j \in S^0} \text{WID}(G[M_j])$$

7. Return S .

Theorem 63. *Let G be a graph with n vertices and m edges. If WEIGHTED INDEPENDENT DOMINATION can be solved for prime induced subgraphs of G in time $O(n^c)$ for a constant $c \geq 1$, then the problem can be solved for G in time $O(n^c + m)$.*

Proof. Let G be a graph with n vertices and m edges. The recursive decomposition of G produced by algorithm WID can be implemented in time $O(n + m)$ [109]. This decomposition associates with G a tree, $T(G)$, whose leaves correspond to the vertices of G , and whose internal nodes represent induced subgraphs of G of size at least 2.

Consider an internal node U of $T(G)$, corresponding to an induced subgraph G_U of G . Then the children of U correspond to the subgraphs $G[M_1], \dots, G[M_k]$, where M_1, \dots, M_k are defined in steps 2-4 of the algorithm. If G_U or \bar{G}_U is disconnected, then G_U^0 is empty (edgeless) or complete respectively, and the problem can be trivially solved for G_U^0 in time $O(|V(G_U^0)|)$. If both are connected, then G_U^0 is a prime induced subgraph of G and the problem can be solved for G_U^0 in time $O(|V(G_U^0)|^c)$ by assumption. Summing over all the internal nodes of $T(G)$, we find that the total time complexity of solving the problem for G is at most $O(\sum_U |V(G_U^0)|^c)$.

It is easy to see that the total number of vertices in all graphs G_U^0 corresponding to internal nodes is equal to the number of edges of $T(G)$, i.e. $|V(T(G))| - 1$. Since the number of leaves of $T(G)$ is n and the number of internal nodes is at most

$n - 1$, we conclude that

$$\sum_U |V(G_U^0)|^c \leq \left(\sum_U |V(G_U^0)| \right)^c \leq (2n - 2)^c = O(n^c).$$

Adding the term $O(n + m)$ needed to obtain the decomposition tree, we obtain the desired time complexity. \square

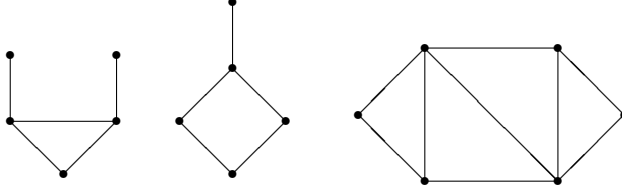


Figure 6.2: The graphs *bull*, *banner* and *A* (listed from left to right).

Let us now apply Theorem 63 to some subclasses P_5 -free graphs. In the description of the subclasses we introduce the *bull*, the *banner* and the graph *A*, depicted in Figure 6.2.

Theorem 64. *The WEIGHTED INDEPENDENT DOMINATING SET problem can be solved in polynomial time for the following subclasses of P_5 -free graphs: (P_5, A) -free graphs, (P_5, bull) -free graphs, $(P_5, K_{2,3})$ -free graphs, (P_5, banner) -free graphs.*

Proof. In [102], it was shown that a prime graph containing a $P_2 + P_3$ contains either a P_5 or the graph *A*. Therefore, every prime (P_5, A) -free graph is $(P_2 + P_3)$ -free. From Theorem 60, we know that WEIGHTED INDEPENDENT DOMINATION is solvable in polynomial time for $(P_5, P_2 + P_3)$ -free graphs. Therefore, by Theorem 63, the problem is also solvable in polynomial time for (P_5, A) -free graphs.

From a result proved in [110] it follows that a prime (P_5, bull) -free graph is either K_3 -free or $K_2 + 2K_1$ -free. It is not difficult to see that (P_5, K_3) -free graphs form a subclass of (P_5, A) -free graphs, while $(P_5, K_2 + 2K_1)$ -free graphs form a subclass of $(P_5, 2P_3)$ -free graphs. We know that WEIGHTED INDEPENDENT DOMINATION is solvable in polynomial time for (P_5, A) -free graphs and for $(P_5, 2P_3)$ -free graphs. Therefore, by Theorem 63, the problem is also solvable in polynomial time for (P_5, bull) -free graphs.

To solve the problem for $(P_5, K_{2,3})$ -free graphs, we use the following obvious observation: if the problem is solvable for the antineighbourhood of each vertex of a graph G in time T , then it is solvable for G in time nT , where $n = |V(G)|$. In [111], it was shown that if G is a prime $(P_5, K_{2,3})$ -free graph, then the antineighbourhood of each vertex of G induces a graph whose vertices can be partitioned into at most

3 sets W_1, W_2, W_3 in such a way that W_1 and W_2 are clique and $G[W_3]$ is P_3 -free and no element of $W_1 \cup W_2$ distinguishes any component of $G[W_3]$. Let H be a graph admitting such a partition, and $n = |V(H)|$. Let us show that WEIGHTED INDEPENDENT DOMINATION is solvable in polynomial time for H . Clearly, any solution to the problem contains at most one vertex in W_1 , at most one vertex in W_2 , and at most one vertex in each component of $G[W_3]$. In $O(n^2)$ time, one can inspect all solutions containing exactly one vertex in each of W_1 and W_2 . If a pair of chosen vertices from $W_1 \cup W_2$ does not dominate W_3 , what is left is choosing a vertex of minimum weight (not dominated by the pair) in each component of $G[W_3]$. Also, in time $O(n^2)$ one can inspect all solutions containing at most one vertex x in $W_1 \cup W_2$. Assume without loss of generality that x belongs to W_1 , and let W'_2 and W'_3 be the vertices of W_2 and W_3 not dominated by x . If W'_2 contains a vertex which is anticomplete to each component of $G[W'_3]$, then clearly there is no solution containing x . If each vertex of W'_2 is complete to at least one component of $G[W'_3]$, then an optimal solution containing x can be obtained by choosing a vertex of minimum weight in each component of $G[W'_3]$. Similarly, an optimal solution containing no vertex in $W_1 \cup W_2$ can be obtained by choosing a vertex of minimum weight in each component of $G[W_3]$ and by checking if the chosen set dominates $W_1 \cup W_2$.

Finally, in [112] it was shown that every prime *banner*-free graph is $K_{2,3}$ -free. Together with the above discussion and Theorem 63 this implies polynomial-time solvability of the problem for (P_5, \textit{banner}) -free graphs. \square

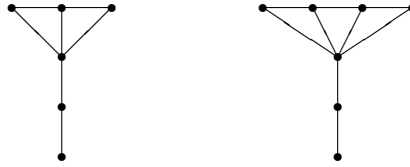


Figure 6.3: The graphs P_3^* (left) and P_4^* (right)

Now we prove a result that allows further extensions of subclasses of P_5 -free graphs with polynomial-time solvable WEIGHTED INDEPENDENT DOMINATING set problem. Given a graph F , we denote by F^* the graph obtained by adding to F three new vertices inducing a P_3 so that one of the endpoints of the P_3 dominates F . For instance, P_3^* is the graph represented in Figure 6.3 (right).

Theorem 65. *Let F be any connected graph. If WEIGHTED INDEPENDENT DOMINATION can be solved in polynomial time for (P_5, F) -free graphs, then this problem can also be solved in polynomial time for (P_5, F^*) -free graphs.*

Proof. To prove the theorem, we will show that *prime* (P_5, F^*) -free graphs are $(K_2 + F)$ -free, in which case the problem is solvable in polynomial time by Theorems 61 and 63.

Assume to the contrary that G is a prime (P_5, F^*) -free graph containing an induced $K_2 + F$. Denote by a and b the vertices that induce K_2 in the graph $K_2 + F$. Let $P = (p_0, p_1, \dots, p_k)$ be a shortest path with $p_0 \in \{a, b\}$ and $p_k \in V(F)$ (such a path must exist, since G is prime and hence is connected). Obviously $k \geq 2$, and because of P_5 -freeness of G , $k \leq 3$.

Let $k = 3$. Then p_1 dominates $\{a, b\}$ and p_2 dominates the $V(F)$, since otherwise a P_5 arises. But now a, p_1, p_2 together with the vertices of F induce an F^* in G .

Let $k = 2$. Denote by X the subset of vertices of G that dominate $\{a, b\}$ and have a neighbour in F , and let Y be the component of $G[V \setminus X]$ containing a, b .

Claim 1. $Y \cap V(F) = \emptyset$ and no vertex of Y has a neighbour in $V(F)$.

Proof. By contradiction, assume that $V(F)$ and Y have a vertex in common or a vertex of Y has a neighbour in $V(F)$. Then by definition of Y , all vertices of $V(F)$ must belong to Y , since $G[F]$ is connected. Therefore, Y must contain a path $P' = (p'_0, p'_1, \dots, p'_l)$ of length $l = 2$ from $\{a, b\}$ to F (the case of $l = 3$ has been analysed before). By definition, p'_1 does not belong to X and therefore it has exactly one neighbour in $\{a, b\}$, say a . To avoid an induced P_5 , we conclude that p'_1 dominates $V(F)$. But now a, b, p'_1 together with the vertices of F induce an F^* in G . This contradiction completes the proof of Claim 1.

Claim 2. Each vertex of Y is adjacent to each vertex of X .

Proof. To the contrary, let Y^0 be the subset of vertices of Y that have a non-neighbour in X and $Y^1 = Y - Y^0$. Denote by y a vertex in Y^0 of minimum distance from $\{a, b\}$ and let $x \in X$ be a non-neighbour of y . Since $G[Y]$ is connected, y has a neighbour $y' \in Y^1$. From Claim 1 we know that vertices of Y have no neighbours in F , and by definition x has a neighbour in F . Then either x dominates $V(F)$, in which case y, y', x together with the vertices of F induce an F^* in G , or x has a non-neighbour in F . In the latter case, x must distinguish two adjacent vertices of F , since F is connected. But then y, y', x and two adjacent vertices of F distinguished by x induced a P_5 . This contradiction completes the proof of Claim 2.

From Claims 1 and 2 it follows that Y is a module, since no vertex outside Y distinguishes it. Moreover, this is a non-trivial module, since it contains a, b and does not contain any vertex of F . This contradicts the fact that G is prime and shows that G contains a P_5 . \square

6.5 Decreasing graphs

The idea of decreasing graphs is an adaptation of the augmenting graph technique used to solve the maximum independent set problem in various classes of graphs. In particular, this technique was used by Edmonds to solve the maximum matching problem, which is equivalent to the maximum independent set problem in the class of line graphs. See [113, 100, 114] for more applications of this technique. The idea of augmenting graphs consists in step-by-step increasing a current solution until a maximum independent set is obtained. In the case of the INDEPENDENT DOMINATING SET problem we iteratively decrease a current solution. The main idea of this approach can be described as follows.

Let G be a vertex-weighted graph and I an independent dominating set in G . We denote the weight of a set $U \subseteq V(G)$ by $\omega(U)$. If G contains a bipartite induced subgraph $B = (B_1, B_2, F)$ such that

- (i) $B_1 \cap I = \emptyset$ and $B_2 \subseteq I$,
- (ii) $\omega(B_1) < \omega(B_2)$,
- (iii) $B_1 \cup (I \setminus B_2)$ is an independent dominating set,

then I is not a minimum weight independent dominating set in G , since $B_1 \cup (I \setminus B_2)$ is an independent dominating set of smaller weight. On the other hand, if I is not a minimum weight independent dominating set and if J is an arbitrary minimum weight independent dominating set in G , then $B_1 := J \setminus I$ and $B_2 := I \setminus J$ induce a bipartite graph satisfying (i), (ii), (iii). We call a bipartite graph B satisfying (i), (ii), (iii) a *decreasing graph* for I , and we call the difference $\omega(B_2) - \omega(B_1)$ the *decrement* of B . Observe that by definition the decrement is a strictly positive number. If there is a decreasing graph for I , we also say that I admits a decreasing graph.

According to the above discussion, an independent dominating set in a graph G is of minimum weight if and only if I admits no decreasing graph. Let us observe that a decreasing graph may, in general, be disconnected. However, if we deal with P_5 -free graphs, we may restrict ourselves to *connected* decreasing graphs, as we show in the following lemma.

Lemma 66. *Let G be a P_5 -free vertex-weighted graph and I an independent dominating set in G . Then I is an independent dominating set of minimum weight if and only if it admits no CONNECTED decreasing graph.*

Proof. If I is a minimum weight independent dominating set, then it admits *no* decreasing graphs (including connected decreasing graphs), which proves the lemma in one direction. To prove it in the other direction, assume I admits no connected decreasing graph, and suppose by contradiction that I is not a minimum weight independent dominating set. Then there must exist a decreasing graph $B = (B_1, B_2, F)$ with at least two connected components. Each component of B must contain at least two vertices, since otherwise either I or $B_1 \cup (I \setminus B_2)$ is not a dominating set. Among the components of B there must exist at least one component with strictly positive decrement. Let $B' = (B'_1, B'_2, F')$ be such a component. By assumption, $B'_1 \cup (I \setminus B'_2)$ is not a dominating set in G , since otherwise B' would be a connected decreasing graph. Therefore, there must exist a vertex u that does not belong to $B'_1 \cup (I \setminus B'_2)$ and has no neighbour in this set. We observe that

- u does not belong to the set B'_2 (and hence to I), since every vertex of B'_2 has a neighbour in B'_1 due to the connectedness of B' ,
- u has a neighbour b'_2 in B'_2 , since I is dominating,
- u does not belong to B_1 , since the only vertices of B_1 that have neighbours in B'_2 are those in the set B'_1 , but u does not belong B'_1 ,
- u has a neighbour b''_1 in $B_1 \setminus B'_1$, since $B_1 \cup (I \setminus B_2)$ is a dominating set.

Since each connected component of B has at least two vertices, b'_2 must have a neighbour $b'_1 \in B'_1$, while b''_1 must have a neighbour $b''_2 \in B_2 \setminus B'_2$. But then vertices $b'_1, b'_2, b''_1, b''_2, u$ induce a P_5 in G . This contradiction completes the proof of the lemma. \square

Let us now apply Lemma 66 to solve WEIGHTED INDEPENDENT DOMINATION in the class of $(P_5, K_{1,p})$ -free graphs for any fixed value of p . We emphasize that in the current version the solution applies only to graphs of polynomial weight, i.e. we assume that the total weight of the input graph is bounded by a polynomial in the number of vertices.

Theorem 67. *The WEIGHTED INDEPENDENT DOMINATING SET problem for $(P_5, K_{1,p})$ -free graphs of polynomial weight can be solved in polynomial time for each fixed value of p .*

Proof. Let G be a $(P_5, K_{1,p})$ -free graph and I an arbitrary independent dominating set in G . By Lemma 66, I is a minimum weight independent dominating set if and only if it admits no connected decreasing graph. It is well-known (and not difficult

to see) that a connected P_5 -free bipartite graph is $2K_2$ -free. Therefore, the vertices in each part of a connected P_5 -free bipartite graph can be ordered under inclusion of their neighbourhoods, and hence any vertex with a maximal neighbourhood is adjacent to all vertices in the opposite part. This implies that if a connected P_5 -free bipartite graph is $K_{1,p}$ -free, then each part of the graph has at most $p - 1$ vertices. Thus, I is a minimum weight independent dominating set if and only if it admits no connected decreasing graph with at most $2p - 2$ vertices. If such a graph exists for I , we obtained an independent dominating set of smaller weight by exchanging the parts of the graph. Since the weights of the vertices are bounded by a polynomial, in polynomially many iterations we obtain an independent dominating set of minimum weight. Each iteration can be done in $O(n^{2p})$ time, and hence the overall complexity of the procedure is bounded by a polynomial. \square

6.6 Conclusion

In this chapter we first gave an algorithm which generates maximal independent sets in $2K_2$ -free graphs, inspired by Farber's proof that such graphs have only polynomially many maximal independent sets and went on to extend the algorithm to larger classes of graphs. In particular, we solved the problem in the class of $P_2 + P_3$ -free graphs, correcting a mistake in [87], as well as in the class of $(P_5, 2P_3)$ -free graphs. We considered a number of subclasses of P_5 -free graphs obtained by forbidding one additional induced subgraph, making use of modular decomposition and so-called decreasing graphs to give a collection of polynomial time results. These results are extended from solutions in (P_5, F) -free graphs to $(P_5, F + pK_2)$ -free graphs and (P_5, F^*) -free graphs, where F^* is obtained from F by adding three vertices inducing a P_3 such that one end vertex dominates the vertices of F .

Notice that all results in this chapter deal with classes defined by finitely many forbidden induced subgraphs. A useful tool to study complexity of algorithmic problems on finitely defined classes is the notion of boundary properties of graphs. This notion was introduced by Alekseev in [62] for the MAXIMUM INDEPENDENT SET problem and then was extended in [113, 115, 15] to other problems. The importance of this notion is due to the fact that an NP-hard algorithmic graph problem Π can be solved in polynomial time in a finitely defined class X if and only if X contains none of the boundary classes for Π .

At present, only two boundary classes are known for INDEPENDENT DOMINATION: the class \mathcal{S} and the class \mathcal{T} of line graphs of graphs in \mathcal{S} . Also, it is known that the class of SAT-graphs is finitely defined and the problem is NP-hard

in this class. Therefore, the class of SAT-graphs must contain a boundary class for the problem. However, neither \mathcal{S} nor \mathcal{T} is a subclass of SAT-graphs, and hence, there must exist at least one more boundary class for the problem. We discover this class in the concluding part of the thesis by exploiting the relationship between INDEPENDENT DOMINATION in SAT-graphs and SATISFIABILITY.

Part III

Satisfiability
and
Related Problems

Chapter 7

Boundary Properties of the Satisfiability Problem

7.1 Introduction

SATISFIABILITY (SAT for short) is the central problem of theoretical computer science, because it is the first problem which has been shown to be NP-complete. Moreover, the problem remains NP-complete under substantial restrictions. For instance, Cook [116] proved that 3-SAT, i.e. the SATISFIABILITY problem restricted to instances with at most three literals per clause, is NP-complete. This restriction is tight in the sense that 2-SAT is polynomial-time solvable [117, 118]. However, it is not a tightest possible restriction in the sense that 3-SAT remains NP-complete even for instances where each variable appears in at most five clauses, as was shown by Papadimitriou in [119]. This tightening allowed Papadimitriou to show that the Euclidean travelling salesman problem is NP-complete.

Tovey [120] strengthened this line of restrictions further by showing that 3-SAT remains NP-complete for instances where each variable appears in at most three clauses. Moreover, he also showed that this restriction is best possible in the sense that 3-SAT where each variable appears in at most two clauses is polynomial-time solvable. However, this restriction is not best possible if we allow other types of restrictions.

In order to introduce more restricted versions of the SATISFIABILITY problem, let us associate with each instance F of the problem a bipartite graph G_F one part of which corresponds to the variables and the other part to the clauses. The edges of G_F connect variables to the clauses containing them. We call G_F the *formula graph* of F . See Figure 7.1 for an example of a formula graph.

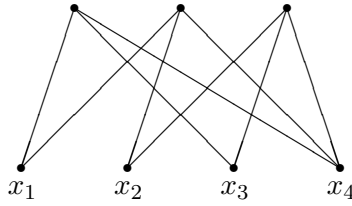


Figure 7.1: The formula graph of $(x_1 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee x_3 \vee x_4)$

The notion of the formula graph was introduced in [121] by Lichtenstein. He called a CNF formula *planar* if its formula graph is planar and proved that 3-SAT is NP-complete even if restricted to planar formulas. The main motivation for the introduction of this type of restriction was to simplify reductions in the proof of NP-completeness of various problems. For instance, Lichtenstein [121] used PLANAR 3-SAT to prove the NP-completeness of geometric connected dominating set and Mansfield [122] used PLANAR 3-SAT to prove the NP-completeness of deciding whether a graph has thickness two.

Kratochvíl [123] introduced more restrictions on PLANAR 3-SAT by requiring the formula graph to be vertex 3-connected and each variable to appear in at most four clauses. Again, this additional restriction was motivated by the study of specific problems and allowed the author to prove their NP-completeness.

With a closer look at the Tovey's transformation [120], one can easily see that it also works in the planar case, and therefore PLANAR 3-SAT remains NP-complete for instances where each variable appears in at most three clauses. Is this an end of the story, i.e. is this restriction best possible? In the present chapter, we answer this question negatively. Moreover, we identify an *infinite* sequence of restricted NP-complete satisfiability problems converging to a particular version of the problem.

Finding the strongest possible restrictions under which a problem remains NP-complete is important for at least two reasons. First, this can make it easier to establish the NP-completeness of new problems by allowing easier transformations, as the above discussion shows. Second, this can help clarify the interesting boundary between tractable and intractable instances of the problem. In the present chapter, we address the second issue, i.e. our goal is to find an end to the story about restricted versions of the SATISFIABILITY problem. In this endeavour, we use the terminology and techniques of graph theory. To justify this approach, we observe that not only planarity but also the other two types of restrictions mentioned above can be described in terms of the formula graphs. Indeed, by bounding the number of

variables per clause we bound the degree of the clause vertices in the formula graph and by bounding the number of appearances of each variable we bound the degree of the variable vertices. In the language of graph theory, the result of Tovey [120] can be formulated as follows: the SATISFIABILITY problem restricted to formula graphs of vertex degree at most 3 is NP-complete.

We also observe that several important results in the study of satisfiability have been obtained with the help of graph theory. For instance, to show polynomial-time solvability of 2-SAT, the authors of [117] reduce the problem to identifying strong components in a directed graph. In [120], it was proved that for each r , every CNF formula with *exactly* r variables per clause and at most r occurrences per variable is satisfiable by showing that in this case the formula graph necessarily has a perfect matching. In [124], the authors proved that SATISFIABILITY restricted to instances whose formula graphs are chordal bipartite can be solved in polynomial time.

Let us observe that all classes of graphs mentioned so far (planar, bipartite, chordal bipartite, graphs of bounded vertex degree) belong to the family of so called hereditary properties. Hereditary classes allow a uniform description in terms of forbidden induced subgraphs, which in turn leads to a systematic approach to study computational complexity of various problems. Based on this approach, we identify an *infinite* sequence of “difficult” classes of formula graphs converging to a particular property. Following [115, 15] we call it a *limit property*. A minimal limit property is called *boundary*. The main result of this chapter is the identification of the first boundary property of formula graphs.

The organization of the chapter is as follows. In the rest of this section we recall basic terminology of satisfiability and graph theory. In Section 7.2, we introduce the notions of limit and boundary properties of graphs. In Section 7.3 we identify the first limit property of formula graphs, and in Section 7.4, we prove its minimality. In Section 7.5 we discuss the question whether this is a unique boundary class.

An instance of the satisfiability problem is given by a Boolean formula in *Conjunctive Normal Form* (CNF). A CNF formula consists of conjunctions of *clauses*, each clause is the disjunction of *literals* and each literal is either a variable or its negation. We denote the set of clauses by \mathcal{C} and the set of variable by $\mathcal{X} = \{x_1, \dots, x_n\}$. We say that a variable *appears* in a clause if the clause contains either the variable itself or its negation. If we want to specify that the clause contains negation of the variable, we say that the variable appears in the clause *negatively*, otherwise it appears *positively*.

A *truth assignment* is a mapping $\gamma : \mathcal{X} \rightarrow \{0, 1\}$ which assigns to each variable one of the two values 0 or 1. A truth assignment *satisfies a clause* $C \in \mathcal{C}$ if C contains at least one literal whose value is 1.

A truth assignment *satisfies a CNF formula* if it satisfies each of its clauses. Given a CNF formula F , the SATISFIABILITY problem asks to determine if there is a truth assignment satisfying F .

7.2 Hereditary, limit and boundary properties of graphs

In order to give a formal definition of a boundary property, let us recall that a graph property is *hereditary* if it is closed under taking induced subgraphs. In other words, a class of graphs is hereditary if deletion of a vertex from a graph in the class results in a graph in the same class. It is well-known and not difficult to see that a class of graphs is hereditary if and only if it can be characterized in terms of forbidden induced subgraphs. More formally, for a set M of graphs, we denote by $Free(M)$ the class of all graphs containing no induced subgraphs isomorphic to graphs in the set M . Then a class Y of graphs is hereditary if and only if $Y = Free(M)$ for a set M . Moreover, for every hereditary class, the set of minimal (or the minimal set of) forbidden induced subgraphs is unique.

The family of hereditary properties contains two important subfamilies: monotone properties and minor-closed properties. A graph property is *monotone* if it is closed under vertex deletions and edges deletions. A graph property is *minor-closed* if it is closed under vertex deletions, edges deletions, and edge contractions. For instance, graphs of degree at most k (for a fixed k) form a monotone, but not minor-closed, property, since vertex and edge deletions cannot increase the degree, while edge contractions can. In contrast, planarity is a minor-closed property.

Similarly to the description of hereditary properties by means of minimal forbidden induced subgraphs, monotone properties can be described by minimal forbidden subgraphs, not necessarily induced. We will denote the monotone property containing no subgraphs from a set M by $Free_m(M)$.

To simplify our discussion, let us call any hereditary property of formula graphs with polynomial-time solvable SATISFIABILITY problem *good* and all other hereditary properties of formula graphs *bad*.

A hereditary property Z of formula graphs will be called a *limit property* if there is a sequence $Y_1 \supseteq Y_2 \supseteq \dots$ of bad classes such that $Z = \bigcap_{i \geq 1} Y_i$. We will say that the sequence $\{Y_i\}_{i \geq 1}$ converges to Z .

A minimal limit property will be called a *boundary property*. A helpful minimality criterion is given in the following lemma.

Lemma 68. *A limit class $Y = \text{Free}(M)$ is minimal (i.e. boundary) if and only if for every graph $G \in Y$ there is a finite set $T \subseteq M$ such that $\text{Free}(\{G\} \cup T)$ is good.*

Proof. Suppose Y is a boundary class, and assume for contradiction that there is a graph $G \in Y$ such that for every finite set $T \subseteq M$ the class $\text{Free}(\{G\} \cup T)$ is bad. Let $M := \{m_1, m_2, \dots\}$ and $Z_i := \text{Free}(G, m_1, m_2, \dots, m_i)$. Then, according to our assumption, Z_i is bad for each i . Therefore, $Z := \bigcap_i Z_i$ is a limit class. It contains no element from M and it does not contain G . Therefore, it is a proper subset of Y , contradicting the minimality of Y .

Conversely, assume that for every graph $G \in Y$ there is a finite set $T \subseteq M$ such that $\text{Free}(\{G\} \cup T)$ is good, and suppose for contradiction that there exists a limit class Z which is properly contained in X . Since Z is a limit class, there exists a sequence $Z_1 \supseteq Z_2 \supseteq \dots$ of bad classes converging to Z . Pick any graph $G \in Y \setminus Z$ and a finite set $T \subseteq M$ such that $\text{Free}(\{G\} \cup T)$ is good. Then, since T is finite, there must exist an n such that Z_n is $(\{G\} \cup T)$ -free, in which case Z_n must be good. This contradiction finishes the proof. \square

7.3 A limit property of satisfiability

Let F be a CNF formula, x a variable and C a clause containing it. We denote the literal of x contained in C by x^α , where $x^\alpha = x$ if $\alpha = 1$, and $x^\alpha = \bar{x}$ if $\alpha = 0$. Let us denote by $F(x, C)$ the formula obtained from F as follows: add a new variable y , add a new clause $x^\alpha \vee \bar{y}$, replace x^α by y in C . We make the following obvious but useful observation:

Observation 69. *Formula F is satisfiable if and only if $F(x, C)$ is.*

In terms of graphs, the transformation of F into $F(x, C)$ is equivalent to a double subdivision of an edge in the formula graph G_F . This observation leads to the following conclusion, where H_n denotes the graph represented on the left of Figure 7.2.

Lemma 70. *For any fixed k , the SATISFIABILITY problem restricted to instances whose formula graphs belong to the class $\text{Free}(K_{1,4}, C_3, C_4, \dots, C_k, H_1, H_2, \dots, H_k)$ is NP-complete.*

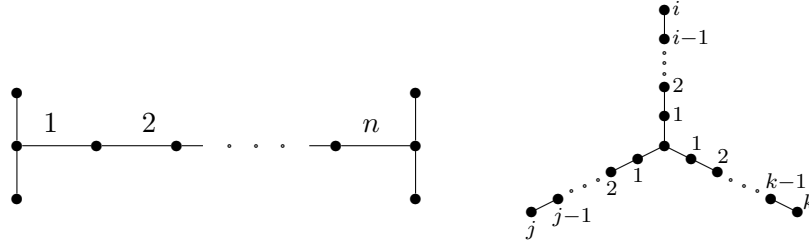


Figure 7.2: Graphs H_n (left) and $S_{i,j,k}$ (right)

Proof. We know that SAT is NP-complete when restricted to formula graphs of vertex degree at most three [120]. Given a graph G_F in this class, we subdivide each edge of G twice and denote the resulting graph by $G_{F'}$. Observe that by means of this transformation we destroy all small induced copies of cycles and graphs of the form H_i . According to Observation 69, F is satisfiable if and only if F' is. Applying the above transformation sufficiently many times, we can transform G_F into a graph $G_{F''}$ in the class $Free(C_3, \dots, C_k, H_1, \dots, H_k)$. As before, F is satisfiable if and only if F'' is, and for a fixed k , this transformation is obviously polynomial. Finally, since G_F is of degree at most 3, then so is $G_{F''}$. Therefore, $G_{F''}$ is $K_{1,4}$ -free. \square

Let us denote the class $Free(K_{1,4}, C_3, C_4, \dots, C_k, H_1, H_2, \dots, H_k)$ by S_k . Clearly, $S_3 \supset S_4 \supset S_5 \dots$ and therefore, by Lemma 70, the intersection $\bigcap_k S_k$ is a limit class. This intersection contains no cycles. Therefore, it is a class of forests. Since the intersection does not contain $K_{1,4}$, it is a class of forest of vertex degree at most 3. Finally, since the intersection contains no graphs of the form H_n , every connected component of any graph in this class has at most one vertex of degree 3. In other words, the intersection $\bigcap_k S_k$ consists of graphs in which every connected component has the form $S_{i,j,k}$ (with some values of i, j, k) represented on the right of Figure 7.2. Throughout the chapter, we denote this class by \mathcal{S} . The above discussion leads to the following conclusion.

Theorem 71. *\mathcal{S} is a limit property of the SATISFIABILITY problem.*

7.4 Minimality of the limit property

The proof of minimality of the property \mathcal{S} is based on a number of auxiliary results involving the notion of tree-width. The first of them can be found in [125].

Theorem 72. [125] *The SATISFIABILITY problem restricted to any class of formula graphs of bounded tree-width is polynomial-time solvable.*

The next two results deal with monotone classes, i.e. classes closed under taking subgraphs, not necessarily induced.

Lemma 73. [126] *For any fixed k , the class $Free_m(\{C_j \mid j \geq k\})$ is of bounded tree-width.*

Theorem 74. *Let Y be a monotone class of graphs. If at least one graph of \mathcal{S} does not belong to Y , then the tree-width of Y is bounded.*

Proof. Let us denote by $F_{t,k}$ the graph in \mathcal{S} with exactly t connected components each of which is of the form $S_{k,k,k}$. Clearly, every graph in \mathcal{S} is an induced subgraph of $F_{t,k}$ for some fixed values of t and k . Therefore, it suffices to prove the theorem for monotone classes excluding (i.e. not containing) $F_{t,k}$ for some t and k . We prove it by induction on t .

First, assume $t = 1$. We will show that graphs in $Free_m(S_{k,k,k})$ have bounded tree-width. Let G be a connected graph in this class. Consider a path P of length $2k - 2$, and a cycle C of length at least $2k + 1$ in G . If G does not contain such P or C , then the tree-width of G is bounded by Lemma 73. Assume P and C are vertex disjoint. Since G is connected, there must be a path P' whose endpoints belong to C and P , and the remaining vertices are outside C and P . Then the union of C , P and P' contains a subgraph isomorphic to $S_{k,k,k}$. This contradiction shows that P intersects every cycle of length at least $2k + 1$. Therefore, by deleting from G the vertices of P we obtain a graph with no cycles of length at least $2k + 1$, i.e. a graph of bounded tree-width (Lemma 73). The deletion of P decreases the tree-width of G by at most $2k - 2$, therefore the tree-width of G is bounded as well.

Now let $t > 1$ and G be a graph in $Free_m(F_{t,k})$. If G contains no $S_{k,k,k}$ as a subgraph (not necessarily induced), then the tree-width of G is bounded by the previous paragraph (the basis of the induction). If G contains a copy of $S_{k,k,k}$, then the deletion of this copy results in a graph in $Free_m(F_{t-1,k})$. By the induction hypothesis, this graph is of bounded tree-width. Therefore, the tree-width of G is bounded too, since we deleted only constantly many vertices (namely, $3k + 1$). \square

With the help of the above results we now prove the main result of this chapter.

Theorem 75. *\mathcal{S} is a boundary property of the SATISFIABILITY problem.*

Proof. We use Lemma 68. Let G be a graph from \mathcal{S} . Without loss of generality, we may assume that every connected component of G has the form $S_{k,k,k}$ for some finite value k , since every graph in \mathcal{S} is an induced subgraph of a graph of this form.

Consider the class $Z := \text{Free}(G, K_{1,4}, C_3, \dots, C_{2k+1}, H_1, \dots, H_{2k+1})$. By definition, no graph in Z contains an induced copy of G . We can show even more: no graph in Z contains G as a subgraph, not necessarily induced. Indeed, assume to the contrary that a graph from Z contains a copy of G as a subgraph. We know that this copy is not induced, therefore it must contain an edge which does not belong to G . If this edge connects two vertices of the same connected component of G , then a chordless cycle of length at most $2k + 1$ arises, which is impossible since such cycles are forbidden. Similarly, a small chordless cycle can be found if two different connected components of G are connected by at least 2 edges. Finally, if two different connected components of G are connected by a single edge, then an induced copy of a graph H_i with $i \leq 2k + 1$ arises, which is also impossible. Therefore, no graph in Z contains G as a subgraph. In other words, $Z \subseteq \text{Free}_m(G)$. By Theorem 74 this implies that the tree-width of graphs in Z is bounded. As a result, Z is a good class (by Theorem 72) and hence \mathcal{S} is a minimal limit class (by Lemma 68). \square

7.5 Is \mathcal{S} a unique boundary property?

We conjecture that \mathcal{S} is unique, i.e. that if a graph property \mathcal{P} is a boundary property for SATISFIABILITY then \mathcal{P} is in fact \mathcal{S} . The class \mathcal{S} is a unique boundary class if and only if for every graph $G \in \mathcal{S}$ the SATISFIABILITY problem is polynomial-time solvable for instances whose formula graphs are G -free. For some small graphs G in \mathcal{S} answering this question is an easy task. For instance, if $G = S_{1,1,1}$, then the problem is polynomial-time solvable, because any $S_{1,1,1}$ -free bipartite graph is of vertex degree at most 2, in which case we deal with an instance of 2-SAT. Also, if $G = S_{0,1,2}$, then the problem is polynomial-time solvable again, because any $S_{0,1,2}$ -free bipartite graph is chordal bipartite, in which case the problem is solvable due to the result in [124]. Below we generalize both of these observations and show that the problem is polynomial-time solvable for instances whose formula graphs are $S_{1,1,2}$ -free. For larger graphs in \mathcal{S} , determining the complexity status of the problem remains a challenging open question.

Theorem 76. *The SATISFIABILITY problem restricted to instances whose formula graphs are $S_{1,1,2}$ -free is solvable in polynomial time.*

Proof. First, we observe that every connected $S_{1,1,2}$ -free bipartite graph is either of degree at most 2 or an almost complete bipartite graph, i.e. a bipartite graph in which every vertex has at most one non-neighbour in the opposite part (see e.g. [62]). For graphs of degree at most 2, the problem is simple. To solve the problem for

almost complete bipartite graphs, we employ the notion of Davis-Putnam resolution defined as follows.

For a formula F containing a variable x , let C, D be two clauses such that one of them contains x positively, the other one negatively, and x is the only variable on which C and D ‘disagree’. Then the clause $C \cup D \setminus \{x, \bar{x}\}$ is called the x -resolvent of C and D . Let $DP_x(F)$ be the formula obtained from F by adding all possible x -resolvents and removing all clauses containing x . This operation is called Davis-Putnam resolution. It is known [127] (and not difficult to see) that F is satisfiable if and only if $DP_x(F)$ is satisfiable.

Claim 77. *If F is an instance whose formula graph is almost complete bipartite and let x be an arbitrary variable, then the formula graph of $DP_x(F)$ is complete bipartite.*

Proof. Let G be the almost complete bipartite graph representing F , and let G' be the graph representing $DP_x(F)$. The graph G' is obtained from G by deleting all clauses containing x , introducing all x -resolvents and deleting x .

Let C be a vertex representing a clause in G' . Since x has at most one non-neighbour in G , C is either an x -resolvent or the only non-neighbour of x in G . In the latter case, C must be adjacent to all the vertices in the opposite part of G' , since x has been deleted.

Assume now that C is an x -resolvent of two clauses $D_1, D_2 \in F$. By assumption, every vertex y representing a variable in G is adjacent to at least one of D_1 and D_2 , since otherwise G is not almost complete bipartite. Therefore, y must be adjacent to C in G' , since by definition $C = D_1 \cup D_2 \setminus \{x, \bar{x}\}$. As a result, C is adjacent all the vertices in the opposite part of G' , and hence G' is complete bipartite. \square

As we mentioned earlier, complete bipartite graphs are chordal bipartite. Therefore, the above claim together with the result in [124] which solves the problem for instances representable by chordal bipartite graphs completes the proof of the theorem. \square

7.6 Conclusion

In this chapter, we revealed the first boundary property for the SATISFIABILITY problem. The relationship between this problem and INDEPENDENT DOMINATION in SAT-graphs allows us to identify the respective boundary class for INDEPENDENT DOMINATION.

Let us repeat that a SAT-graph is a graph representing an instance of the SATISFIABILITY problem. Its vertices can be partitioned into a clique and an induced matching. The vertices in the clique part of the graph represent clauses and the vertices in the other part represent literals (i.e. variables and their negations). Each literal vertex x is connected to its negation \bar{x} and to the clauses containing it.

Given a SAT-graph G , let us denote by G^* the graph obtained from G by deleting the edges from its clique part and by contracting the edges of the other part. It is not difficult to see that if G represents an instance F of the SATISFIABILITY problem, then G^* coincides with G_F , i.e. with the formula graph of F . Taking into account this relationship between SAT-graphs and formula graphs and the fact that \mathcal{S} is a boundary class for SATISFIABILITY, the respective boundary class for INDEPENDENT DOMINATION can be described as follows

Corollary 78. *Let $\mathcal{H}\text{-SAT}$ be the hereditary closure of the set of all SAT-graphs, i.e. the class containing all SAT-graphs and all their induced subgraphs, and let $\mathcal{H}\text{-S}$ denote the subclass of $\mathcal{H}\text{-SAT}$ containing graphs G such that $G^* \in \mathcal{S}$. Then $\mathcal{H}\text{-S}$ is a boundary class for the INDEPENDENT DOMINATING SET problem.*

Bibliography

- [1] V. V. Lozin and C. Purcell. Coloring vertices of claw-free graphs in three colors. *Journal of Combinatorial Optimization*, accepted.
- [2] N. Korpelainen, V. V. Lozin, and C. Purcell. Dominating induced matchings in graphs without a skew star. submitted.
- [3] N. Korpelainen. *Boundary properties of graphs*. PhD thesis, University of Warwick, 2012.
- [4] V. V. Lozin, M. Milanič, and C. Purcell. Graphs without large apples and the maximum weight independent set problem. *Graphs and Combinatorics*, accepted.
- [5] V. V. Lozin, R. Mosca, and C. Purcell. Sparse regular induced subgraphs in $2P_3$ -free graphs. *Discrete Optimization*, accepted.
- [6] V. V. Lozin, R. Mosca, and C. Purcell. Independent domination in finitely defined classes of graphs: polynomial algorithms. *Discrete Applied Mathematics*, accepted.
- [7] V. V. Lozin and C. Purcell. Boundary properties of the satisfiability problems. *Information Processing Letters*, 113:313–317, 2013.
- [8] David S. Johnson and Michael A. Trick, editors. *Cliques, coloring, and satisfiability*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 26. American Mathematical Society, Providence, RI, 1996. Papers from the workshop held as part of the 2nd DIMACS Implementation Challenge in New Brunswick, NJ, October 11–13, 1993.
- [9] E. J. Gardiner, P. J. Artymiuk, and P. Willett. Clique-detection algorithms for matching three-dimensional molecular structures. *Journal of Molecular Graphics and Modeling*, 15:245–253, 1997.

- [10] M. Pelillo, K. Siddiqi, and S. W. Zucker. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:1105–1120, 1999.
- [11] S. Khurshid and D. Marinov. Testera: Specification-based testing of java programs using SAT. *Automated Software Engineering*, 11:403–434, 2004.
- [12] M. Jean. An interval graph is a comparability graph. *Journal of Combinatorial Theory*, 7:189–190, 1969.
- [13] P. C. Fishburn. An interval graph is not a comparability graph. *Journal of Combinatorial Theory*, 8:442–443, 1970.
- [14] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. The strong perfect graph theorem. *Annals of Mathematics*, 164(1):51–229, 2006.
- [15] N. Korpelainen, V. V. Lozin, D. S. Malyshev, and A. Tiskin. Boundary properties of graphs for algorithmic graph problems. *Theoretical Computer Science*, 412:3545–3554, 2011.
- [16] H. L. Bodlaender. A partial k -arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209:1–45, 1998.
- [17] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. Freeman, San Francisco, CA, 1979.
- [18] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.
- [19] M. Kochol, V. V. Lozin, and B. Randerath. The 3-colourability problem on graphs with maximum degree 4. *SIAM Journal on Computing*, 32:1128–1139, 2003.
- [20] M. Kochol. 3-colouring and 3-clique-ordering of locally connected graphs. *Journal of Algorithms*, 54:122–125, 2005.
- [21] F. Maffray and M. Preissmann. On the NP-completeness of the k -colourability problem for triangle-free graphs. *Discrete Mathematics*, 162:313–317, 1996.
- [22] B. Randerath. 3-colourability and forbidden subgraphs I: characterizing pairs. *Discrete Mathematics*, 276:313–325, 2004.

- [23] B. Randerath, I. Schiermeyer, and M. Tewes. 3-colourability and forbidden subgraphs II: polynomial algorithms. *Discrete Mathematics*, 251:137–153, 2002.
- [24] A. Tucker. A reduction procedure for coloring perfect K_4 -free graphs. *Journal of Combinatorial Theory, Series B*, 43(2):151–172, 1987.
- [25] M. Chudnovsky and P. Seymour. Claw-free graphs V – global structure. *Journal of Combinatorial Theory, Series B*, 98:1373–1410, 2008.
- [26] O. Favaron. Independence and upper irredundance in claw-free graphs. *Discrete Applied Mathematics*, 132:85–95, 2003.
- [27] V. B. Le, R. Mosca, and H. Müller. On stable cutsets in claw-free graphs and planar graphs. *Journal of Discrete Algorithms*, 6:256–276, 2008.
- [28] G. J. Minty. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory Series B*, 28:284–304, 1980.
- [29] I. Holyer. The NP-completeness of edge-colouring. *SIAM Journal on Computing*, 10:718–720, 1981.
- [30] Frank Harary. *Graph theory*. Addison-Wesley Publishing Co., Reading, Mass.-Menlo Park, Calif.-London, 1969.
- [31] M. Kaminski and V. V. Lozin. Coloring edges and vertices of graphs without short or long cycles. *Contributions to Discrete Mathematics*, 2:61–66, 2007.
- [32] H. L. Bodlaender and D. M. Thilikos. Treewidth for graphs with small chordality. *Discrete Applied Mathematics*, 79:45–61, 1997.
- [33] 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.
- [34] V. V. Lozin and D. Rautenbach. On the band-, tree- and clique-width of graphs with bounded vertex degree. *SIAM Journal of Discrete Mathematics*, 18:195–206, 2004.
- [35] M. Kaminski and V. V. Lozin. Vertex 3-colourability of claw-free graphs. *Algorithmic Operations Research*, 2:15–21, 2007.

- [36] A. Brandstädt, C. Hundt, and R. Nevries. Efficient edge domination on hole-free graphs in polynomial time. *Lecture Notes in Computer Science*, 6034:650–661, 2010.
- [37] D. M. Cardoso, J. Cerdeira, C. Delorme, and P. C. Silva. Efficient edge domination in regular graphs. *Discrete Applied Mathematics*, 156:3060–3065, 2008.
- [38] D. L. Grinstead, P. J. Slater, N. A. Sherwani, and N. D. Holmes. Efficient edge domination problems in graphs. *Information Processing Letters*, 48:221–228, 1993.
- [39] C. L. Lu, M-T. Ko, and C. Y. Tang. Perfect edge domination and efficient edge domination in graphs. *Discrete Applied Mathematics*, 119:227–250, 2002.
- [40] C. L. Lu and C. Y. Tang. Solving the weighted efficient edge domination problem on bipartite permutation graphs. *Discrete Applied Mathematics*, 87:203–211, 1998.
- [41] A. Brandstädt and R. Mosca. Dominating induced matchings for P_7 -free graphs in linear time. *Lecture Notes in Computer Science*, 7074:100–109, 2011.
- [42] D. Cardoso and V. V. Lozin. Dominating induced matchings. *Lecture Notes in Computer Science*, 5420:77–86, 2009.
- [43] D. M. Cardoso, N. Korpelainen, and V. V. Lozin. On the complexity of the dominating induced matching problem in hereditary classes of graphs. *Discrete Applied Mathematics*, 159:521–531, 2011.
- [44] N. Korpelainen. A polynomial-time algorithm for the dominating induced matching problem in the class of convex graphs. *Electronic Notes in Discrete Mathematics*, 32:133–140, 2009.
- [45] M. Livingston and Q. F. Stout. Distributing resources in hypercube computers. In *Proceedings of the third conference on Hypercube concurrent computers and applications: Architecture, software, computer systems, and general issues*, volume 1, pages 222–231, New York, NY, USA, 1988. ACM.
- [46] J. Kratochvíl. Regular codes in regular graphs are difficult. *Discrete Mathematics*, 133:191–205, 1994.

- [47] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
- [48] V. V. Lozin and M. Milanič. A polynomial algorithm to find an independent set of maximum weight in a fork-free graph. *Journal of Discrete Algorithms*, 6:595–604, 2008.
- [49] D. G. Corneil and U. Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 34:825–847, 2005.
- [50] A. Brandstädt, T. Klemmt, V. V. Lozin, and R. Mosca. Independent sets of maximum weight in apple-free graphs. *Lecture Notes in Computer Science*, 5369:849–859, 2008.
- [51] R. Boliac and V. V. Lozin. Independent domination in finitely defined classes of graphs. *Theoretical Computer Science*, 301:271–284, 2003.
- [52] G. J. Chang. The weighted independent domination problem is NP-complete for chordal graphs. *Discrete Applied Mathematics*, 143:351–352, 2004.
- [53] V. E. Alekseev. On the local restriction effect on the complexity of finding the graph independence number. *Combinatorial-algebraic Methods in Applied Mathematics*, pages 3–13, 1983. Gorkiy University Press, Gorkiy (in Russian).
- [54] M. Chudnovsky and P. Seymour. Claw-free graphs I – orientable prismatic graphs. *Journal of Combinatorial Theory, Series B*, 97:867–901, 2007.
- [55] M. Chudnovsky and P. Seymour. Claw-free graphs II – nonorientable prismatic graphs. *Journal of Combinatorial Theory, Series B*, 98:249–290, 2008.
- [56] M. Chudnovsky and P. Seymour. Claw-free graphs III – circular interval graphs. *Journal of Combinatorial Theory, Series B*, 98:812–834, 2008.
- [57] M. Chudnovsky and P. Seymour. Claw-free graphs IV – decomposition theorem. *Journal of Combinatorial Theory, Series B*, 98:839–938, 2008.
- [58] F. Gavril. Algorithms for minimum colouring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1:180–187, 1972.
- [59] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16:47–56, 1974.

- [60] J. Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards, Section B*, 69B:125–130, 1965.
- [61] L. Lovász and M. D. Plummer. Matching theory. *Annals of Discrete Mathematics*, 29, 1986.
- [62] V. E. Alekseev. A polynomial algorithm for finding the largest independent sets in fork-free graphs. *Discrete Applied Mathematics*, 135:3–16, 2004.
- [63] V. E. Alekseev, V. V. Lozin, D. Malyshev, and M. Milanič. The maximum independent set problem in planar graphs. *Lecture Notes in Computer Science*, 5126:96–107, 2008.
- [64] A. Hertz and D. de Werra. On the stability number of AH-free graphs. *Journal of Graph Theory*, 17:53–63, 1993.
- [65] Y. Faenza, G. Oriolo, and G. Stauffer. An algorithmic decomposition of claw-free graphs leading to an $O(n^3)$ -algorithm for the weighted stable set problem. *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 630–646, 2011.
- [66] D. Nakamura and A. Tamura. A revision of Minty’s algorithm for finding a maximum weight stable set in a claw-free graph. *Journal of the Operations Research Society of Japan*, 44:194–204, 2001.
- [67] G. Oriolo, U. Pietropaoli, and G. Stauffer. A new algorithm for the maximum weighted stable set problem in claw-free graphs. *Lecture Notes in Computer Science*, 5035:96–107, 2008.
- [68] V. E. Alekseev. On the number of maximal independent sets in graphs from hereditary classes. *Combinatorial-algebraic methods in discrete optimization, University of Nizhny Novgorod*, 58, 1991. (in Russian).
- [69] M. Farber, M. Hujter, and Z. Tuza. An upper bound on the number of cliques in a graph. *Networks*, 23:207–210, 1993.
- [70] 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.
- [71] A. Frank. Some polynomial algorithms for certain graphs and hypergraphs. In *Proceedings of the Fifth British Combinatorial Conference (Univ. Aberdeen,*

Aberdeen, 1975), pages 211–226. Congressus Numerantium, No. XV, Winnipeg, Man., 1976. Utilitas Mathematics.

- [72] M. Grötschel, L. Lovász, and A. Schrijver. Polynomial algorithms for perfect graphs. *Annals of Discrete Mathematics*, 21:325–356, 1984.
- [73] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k -trees. *Discrete Applied Mathematics*, 23:11–24, 1989.
- [74] S. H. Whitesides. An algorithm for finding clique cut-sets. *Information Processing Letters*, 12:31–32, 1981.
- [75] R. E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55:221–232, 1985.
- [76] A. Brandstädt and C. Hoàng. On clique separators, nearly chordal graphs, and the maximum weight stable set problem. *Lecture Notes in Computer Science*, 3509:265–275, 2005.
- [77] V. V. Lozin and M. Milanič. Maximum independent sets in graphs of low degree. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 874–880, 2007.
- [78] N. Robertson and P. Seymour. Graph minors XX: Wagner’s conjecture. *Journal of Combinatorial Theory, Series B*, 92:325–357, 2004.
- [79] M. Kamiński, V. V. Lozin, and M. Milanič. Recent developments on graphs of bounded clique-width. *Discrete Applied Mathematics*, 157:2747–2761, 2009.
- [80] N. Robertson and P. D. Seymour. Graph minors V: excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41:92–114, 1986.
- [81] E. D. Demaine and M. T. Hajiaghayi. Diameter and treewidth in minor-closed graph families, revisited. *Algorithmica*, 40:211–215, 2004.
- [82] N. Robertson and P. D. Seymour. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58:22–33, 1993.
- [83] D. M. Cardoso, M. Kaminski, and V. V. Lozin. Maximum k -regular induced subgraphs. *Journal of Combinatorial Optimization*, 14:455–463, 2007.
- [84] G. J. Minty. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B*, 28:284–304, 1980.

- [85] N. Sbihi. Algorithme de recherche d'un indépendant de cardinalité maximum dans un graphe sans étoile. *Discrete Mathematics*, 29:53–76, 1980.
- [86] D. Kobler and U. Rotics. Finding maximum induced matchings in subclasses of claw-free and P_5 -free graphs, and in graphs with matching and induced matching of equal maximum size. *Algorithmica*, 37:327–346, 2003.
- [87] V. V. Lozin and R. Mosca. Independent sets in extensions of $2K_2$ -free graphs. *Discrete Applied Mathematics*, 146:74–80, 2005.
- [88] V. V. Lozin and R. Mosca. Maximum regular induced subgraphs in $2P_3$ -free graphs. *Theoretical Computer Science*, 460:26–33, 2012.
- [89] 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.
- [90] Y. L. Orlovich, V. S. Gordon, and D. de Werra. On the inapproximability of independent domination in $2P_3$ -free graphs. *Theoretical Computer Science*, 410:977–982, 2009.
- [91] I. E. Zverovich. Satgraphs and independent domination, part 1. *Theoretical Computer Science*, 352:47–56, 2006.
- [92] F. R. K. Chung, A. Gyárfás, Z. Tuza, and W. T. Trotter. The maximum number of edges in $2K_2$ -free graphs of bounded degree. *Discrete Mathematics*, 81:129–135, 1990.
- [93] M. S. Chung and D. B. West. Large $2P_3$ -free graphs with bounded bounded degree. *Discrete Mathematics*, 150:69–79, 1996.
- [94] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM Journal of Applied Mathematics*, 38:364–372, 1980.
- [95] P. Damaschke, H. Müller, and D. Kratsch. Domination in convex and chordal bipartite graphs. *Information Processing Letters*, 36:231–236, 1990.
- [96] M. Farber. Independent domination in chordal graphs. *Operations Research Letters*, 1:134–138, 1982.
- [97] M. Farber. On diameters and radii of bridged graphs. *Discrete Mathematics*, 73:249–260, 1989.

- [98] B. Randerath and I. Schiermeyer. On maximum independent sets in P_5 -free graphs. *Discrete Applied Mathematics*, 158:1041–1044, 2010.
- [99] H. L. Bodlaender, A. Brandstädt, D. Kratsch, M. Rao, and J. Spinrad. On algorithms for (P_5, gem) -free graphs. *Theoretical Computer Science*, 349:2–21, 2005.
- [100] R. Boliac and V. V. Lozin. An augmenting graph approach to the stable set problem in P_5 -free graphs. *Discrete Applied Mathematics*, 131:567–575, 2003.
- [101] M. U. Gerber, A. Hertz, and D. Schindl. P_5 -free augmenting graphs and the maximum stable set problem. *Discrete Applied Mathematics*, 132:109–119, 2004.
- [102] V. V. Lozin and R. Mosca. Maximum independent sets in subclasses of P_5 -free graphs. *Information Processing Letters*, 109:319–324, 2009.
- [103] F. Maffray. Stable sets in k -colourable P_5 -free graphs. *Information Processing Letters*, 109:1235–1237, 2009.
- [104] R. Mosca. Some results on maximum stable sets in certain P_5 -free graphs. *Discrete Applied Mathematics*, 132:175–183, 2003.
- [105] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica*, 18:25–66, 1967.
- [106] A. Ehrenfeucht and G. Rozenberg. Primitivity is hereditary for 2-structures. *Theoretical Computer Science*, 70:343–358, 1990.
- [107] M. Habib and M. C. Maurer. On the X -join decomposition for undirected graphs. *Discrete Applied Mathematics*, 1:201–207, 1979.
- [108] Rolf H. Möhring. Algorithmic aspects of comparability graphs and interval graphs. In *Graphs and order (Banff, Alta., 1984)*, volume 147 of *NATO Adv. Sci. Inst. Ser. C Mathematics Phys. Sci.*, pages 41–101. Reidel, Dordrecht, 1985.
- [109] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201:199–241, 1999.
- [110] C. De Simone. On the vertex packing problem. *Graphs and Combinatorics*, 9:19–30, 1993.
- [111] R. Mosca. Some observations on maximum weight stable sets in certain P_5 -free graphs. *European Journal of Operational Research*, 184:849–859, 2008.

- [112] A. Brandstädt, T. Klemmt, V. V. Lozin, and R. Mosca. On independent vertex sets in subclasses of apple-free graphs. *Algorithmica*, 56:383–393, 2010.
- [113] V. E. Alekseev and V. V. Lozin. Augmenting graphs for independent sets. *Discrete Applied Mathematics*, 145:3–10, 2004.
- [114] V. V. Lozin and M. Milanič. On finding augmenting graphs. *Discrete Applied Mathematics*, 156:2517–2529, 2008.
- [115] V. E. Alekseev, R. Boliac, D. V. Korobitsyn, and V. V. Lozin. NP-hard graph problems and boundary classes of graphs. *Theoretical Computer Science*, 389:219–236, 2007.
- [116] S. A. Cook. The complexity of theorem-proving procedures. *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.
- [117] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8:121–123, 1979.
- [118] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multi-commodity flow problems. *SIAM Journal on Computing*, 5:691–703, 1976.
- [119] C. H. Papadimitriou. The Euclidean traveling salesman problem is NP-complete. *Theoretical Computer Science*, 4:237–244, 1977.
- [120] C. A. Tovey. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8:85–89, 1984.
- [121] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11:329–343, 1982.
- [122] A. Mansfield. Determining the thickness of graphs is NP-hard. *Proceedings of the Mathematics Cambridge Philosophical Society*, 39:9–23, 1983.
- [123] J. Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discrete Applied Mathematics*, 52:233–252, 1994.
- [124] S. Ordyniak, D. Paulusma, and S. Szeider. Satisfiability of acyclic and almost acyclic CNF formulas (II). *Lecture Notes in Computer Science*, 6695:47–60, 2011.

- [125] G. Gottlob and S. Szeider. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems. *The Computer Journal*, 51:303–325, 2006.
- [126] M. Kaminski and V. V. Lozin. Coloring edges and vertices of graphs without short or long cycles. *Contributions to Discrete Mathematics*, 2:61–66, 2007.
- [127] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.