FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

RNDr. Tomáš Masařík

Variants of graph labeling problems

Department of Applied Mathematics

Supervisor of the doctoral thesis:	doc. RNDr. Jiří Fiala, Ph.D.
Study programme:	Informatics
Study branch:	Discrete Models and Algorithms (4I4)

Prague 2019

Dedication. My first thanks go to Jiří Fiala, my supervisor and mentor. He has always encouraged me to work independently and he gave me a lot of freedom which was essential for me. Moreover, he was always keen on helping me whenever I needed and he gave me several opportunities to meet and work with great researchers in my field.

At the beginning of my PhD studies, I collaborated mostly with my friends who were or still are students. We had lots of fun together not only during work. I wrote many papers together with Dušan Knop, including my first paper that was based on my diploma thesis. Tomáš Toufar, even though he suddenly decided not to contact any of us again (maybe any more), influenced me a lot and I am glad I can mention him here. I have in mind nice memories spent together with Pavel "Koblich" Dvořák. Jana Novotná, who is the only person mentioned here twice, was very nice to work with. Also, I enjoyed working with all the others that time students from Prague group: Radek Hušek, Martin Koutecký, Jan Musílek, Veronika Slívová, Pavel Veselý, and many others. I always like a friendly working environment at Charles in Prague. I took pleasure in working with Milan Hladík, Terka Klimošová, Ondra Pangrác, and Andreas Feldmann. It was an invaluable experience to work with Zdeněk Dvořák who influenced my future research directions significantly.

Later in my studies, I met and worked with several important researchers abroad that formed my research greatly. Large impact on me had Mike Fellows, who showed me that one should focus on a great story and new ideas and not just on problem-solving. It was a great pleasure to collaborate with him and his group in Bergen: Fran Rosamond, Lars Jaffke, and Mateus Oliveira de Oliveira, during my half-year stay in Norway. I also met there Geevarghese Philip, who was always in good mood. Later, I met Marcin Pilipczuk, who offered me an internship in Poland which I enjoyed so much that I stayed there for over a year in a very friendly atmosphere. It is a pleasure to work with him, Manuel Sorge, and Irene Muzi in Warsaw. I am very happy that Paweł Rzążewski in not only a great collaborator but also a good friend. I am glad that I had an opportunity to work with Daniël Paulusma and to enjoy his hospitality in Durham. Great impact on me had my "academic brother" Bernard Lidický who always gave me good advice whenever I needed it. I wish to mention a pleasant working and kind atmosphere of GRWC workshop. In particular, a friendly approach of all the organizers.

I thank also to all of my other collaborators and all the other researchers I met and have not mentioned here. I appreciate our discussions, research, and also fun we had together.

Last but not least, I would like to thank my family for their support. Most importantly to Jana Novotná, who helped me and supported me in my research even though it was sometimes difficult. It is an interesting experience to do research together in a relationship and I am very glad about it.

In memoriam of our friend Petr Bradshaw, who died suddenly at our home during the time I was finishing this thesis.

Title: Variants of graph labeling problems

Author: RNDr. Tomáš Masařík

Department: Department of Applied Mathematics

Supervisor: doc. RNDr. Jiří Fiala, Ph.D., Department of Applied Mathematics

Abstract: This thesis consists of three parts devoted to graph labeling, hereditary graph classes, and parameterized complexity.

Packing coloring, originally Broadcasting Chromatic number, assigns natural numbers to vertices such that vertices with the same label are in distance at least the value of the label. This problem is motivated by the assignment of frequencies to the transmitters. We improve hardness on chordal graphs. We proof that packing coloring on chordal graphs with diameter 3 is very hard to approximate. Moreover, we discuss several positive results on interval graphs and on related structural graph parameters.

Hereditary graph classes are preserved under vertex deletion. We study graphs that do not contain an induced subgraph H. We prove that 3-coloring is polynomial-time solvable for $(P_3 + P_4)$-free and $(P_2 + P_5)$-free graphs and thus we have solved the last open cases for the problem on H-free graphs where H has up to 7 vertices.

Fair problems are a modification of graph deletion problems, where, instead of minimizing the size of the solution, the aim is to minimize the maximum number of neighbors in the deleted set. We show that those problems can be solved in FPT time for an MSO_1 formula parameterized by the size of the formula and the twin cover of the graph. Moreover, we define a basic fair problem, Fair Vertex Cover, and we show that it is $\mathsf{W}[1]$-hard parameterized by both treedepth and feedback vertex set. On the other hand, the problem is in FPT parameterized by modular-width. In addition, we provide a few more hardness proofs and we formulate several promising and interesting future research directions.

Keywords: Graph Coloring, Parameterized Complexity, Fair Problems, H-free Graphs.

Contents

Introduction **3**
- 1.1 Graph Problems and Decision Problems . . . 3
- 1.2 Graph Colorings and Labelings . . . 3
  - 1.2.1 Flexibility . . . 4
- 1.3 Hereditary Graph Classes . . . 4
- 1.4 Parameterized Complexity . . . 5
  - 1.4.1 Parameterized Reduction . . . 6
  - 1.4.2 Kernelization . . . 7
  - 1.4.3 Structural Graph Parameters . . . 7
- 1.5 Graph Logics . . . 9
  - 1.5.1 Graph Metatheorems . . . 10
- 1.6 Organization of Thesis . . . 10

2 Complexity of Packing Coloring **13**
- 2.1 Introduction . . . 13
  - 2.1.1 Known Results . . . 14
  - 2.1.2 Our Results and Structure of the Paper . . . 14
- 2.2 Chordal and Interval graphs . . . 15
  - 2.2.1 Structural Parameters . . . 19
- 2.3 Conclusion . . . 20

3 Coloring of H-free Graphs **21**
- 3.1 Introduction . . . 21
  - 3.1.1 Our Results . . . 23
  - 3.1.2 Preliminaries . . . 23
- 3.2 The Proof of Theorem 3.1 . . . 24
  - 3.2.1 Phase 1. Preprocessing $(\mathbf{G}, \mathbf{L})$ . . . 27
  - 3.2.2 Phase 2. Reduce the Number of Distinct Sets $\mathbf{A}(\mathbf{i}, \mathbf{j})$ . . . 29
  - 3.2.3 Phase 3. Reduce to the Case Where Vertices of $\mathbf{A_1}$ Have the Same List . . . 30
  - 3.2.4 Phase 4. Reduce to a Set of Instances of 2-List Colouring . 34
- 3.3 The Proof of Corollary 3.2 . . . 39
- 3.4 Conclusions . . . 39

4 Survey on Fair problems **41**
- 4.1 Definition of Fair Problems . . . 41
- 4.2 Specific Problems . . . 42
  - 4.2.1 Edge Problems . . . 42
  - 4.2.2 Vertex Problems . . . 44
- 4.3 Metatheorems . . . 44
  - 4.3.1 FPT Results . . . 44
- 4.4 Open Problems and Further Research Direction . . . 44

5 Methatheorems for Fair Problems **47**
5.1 Introduction 47
5.1.1 Our Results 50
5.1.2 Preliminaries 52
5.2 Metatheorems for Fair Evaluation 53
5.2.1 Model Checking 55
5.2.2 Finding a Fair Solution 56
5.3 The Fair VC Problem 60
5.3.1 Hardness for Treedepth and Feedback Vertex Set 60
5.3.2 FPT Algorithm for Modular Width 63
5.4 Hardness of Possible Extensions 65
5.5 Conclusions 68

Bibliography **71**

List of Figures **85**

List of Tables **87**

List of Publications **89**

Introduction

In this thesis, I present three selected results obtained during my Ph.D. studies. In addition, I attach a short annotated bibliography about fair problems, the topic of the third result.

My thesis is centered around graph labeling problems. Those are variants of the famous graph coloring problem that was first mentioned as early as in the 19th century. The original task is to assign colors to vertices of the graph such that the adjacent vertices obtain different colors. This problem is sometimes referred to as *proper coloring*. The graph coloring problem emerges in countless practical application varying from the scheduling to the register assignment. Moreover, it is an important theoretical tool, and therefore, many classical results are described in the language of coloring and its modifications. Graph labeling usually means that we exchange the set of indistinguishable colors for a set of natural numbers so that some additional special meaning can be tied to them.

1.1 Graph Problems and Decision Problems

In this work, we study problems on graphs. A graph is a basic structure consisting of the set of vertices and of the set of edges where each edge represents a pair of vertices. We refer to textbooks and monographs [45, 14, 12, 141] for the usual concepts and notation of graph theory used in this thesis.

A decision problem, or generally, an algorithmic problem, is a language over a finite alphabet. We give a similar formal definition for a parameterized problem (see Definition 1.1). In the case of graphs, the language is just a set of graphs that satisfy the problems, i.e., the set of graphs such that the answer to the decisional problem is yes. Loosely speaking, we are usually interested in problems that can be solved by computers in asymptotically reasonable time.

This vague idea leads to well-studied models of Turing machines and definitions of P and NP classes. This is a well-established theory, described in many books, e.g., in the monograph by Garey and Johnson [75], or the original papers by Cook [31] and Karp [96] from the '70s. We assume that the reader is familiar with these concepts, and refer to [75] for formal definitions.

1.2 Graph Colorings and Labelings

The area of graph coloring, labelings, and related research is one of the most studied among the whole graph theory. There are several books dedicated to graph coloring e.g., [93] and [125]. Also general surveys, for instance [62], as well as several specialized surveys, e.g., [77, 22, 140], were published.

In Chapter 3, we stay in the classical setting where we use a small extension of the notion of proper coloring. For a graph G let L be a function $L : V(G) \to 2^{\{1,\dots,c\}}$, where L is usually called a *list assignment*, then graph G is *L-colorable* (or *list-colorable*) if there exists a proper coloring $\varphi : V(G) \to \{1,\dots,c\}$ such that $\varphi(v) \in L(v)$ for all vertices v. Note that if for all vertices the list is the same and contains all colors $\{1,\dots,c\}$, then this is just the proper coloring. We define a special variant called *list k-coloring* where for all vertices the lists are subsets of $\{1,\dots,k\}$. In this language, the *precoloring extension* problem allows for all vertices only list $\{1,\dots,k\}$ or just a singleton containing only one color. On the other hand, *k-list coloring* or *k-choosability* bound only the size of the lists (each list has size at least k) but not the number of allowed colors.

Chapter 2 is devoted to packing coloring. This is one of the labeling problems with the original motivation in the frequency assignment to the transmitters. Chapters 4 and 5 contain a broadly studied defective coloring problem which is a very natural particular fair edge deletion problem. All the necessary definitions are within respective chapters.

Besides a part of my work, I present in detail in the main part of the thesis, I was during my PhD studies involved in another project closely related to the graph coloring problem. The following subsection serves as a small excursion to the concept of flexibility in the graph coloring.

1.2.1 Flexibility

In certain applications of the classical coloring it is common that some vertices have preferred resource(s). This idea motivated precoloring extension questions. However, unfortunately, it is not usually possible to satisfy all such preferences. The notion of ε-Flexibility was first defined by Dvořák, Norin, and Postle in [54]. Instead of satisfying all the preferences, the aim is to satisfy at least a constant fraction of any request.

More formally, we are given a graph together with its list assignment. By *request* we mean a set of pairs, vertex and its preferred color. Note that not necessarily every vertex has to make a request. Graph G is *ε-flexible*, if it is possible for any list assignment L and any request, L-color the graph G such that at least a constant fraction of the request is satisfied. As it turns out, this question is trivial in the ordinary proper coloring setting with a bounded number of colors (where all the lists are of the same size and they consist of colors ranging from 1 to k). The answer is always positive there, since we can permute the colors according to the request, and therefore, satisfy at least $\frac{1}{k^2}$ fraction of any request. On the other hand, flexibility brought about a number of interesting problems in the list coloring setting. The main target we aimed for is the following statement. There exists an absolute constant ε such that for any graph in the studied graphs class with any lists assignment of size k, and for any request (preferred colors of some of the vertices) there exists a list coloring of the graph such that it satisfies at least ε fraction of the request. A well-studied notion of choosability forms a trivial lower-bound on the value of k from the previous statement. We subsequently derived a couple of theorems of the mentioned form on various subclasses of planar graphs [53, 52, 123]. Table 1.1 summarizes known results and provides a comparison with the choosability on planar graphs.

1.3 Hereditary Graph Classes

Hereditary property is one of the main themes in the study of mathematical structures. *Hereditary graphs* are graph classes such that are closed under vertex deletion. In particular, pattern-free graphs are characterized by some forbidden

Planar Graphs	General	Triangle-free	C_4-free	Girth 5	Girth 6
Choosability	5 [138]	4	4 [110]	3 [139]	3
List size in ε-Flexibility	6 [54]	4 [53]	5 [123]	4	3 [52]

Table 1.1: A summary and a comparisons of choosability and respective results in the flexibility setting on subclases of planar graphs. Non-implied bounds are accompagned with the respective citation.

pattern. This notion captures a large number of well-studied graph classes that are hereditary. For example, bipartite graphs, chordal graphs, and many others. It has a connection with minor-closed graph classes (for example, planar graphs) that can be viewed as even a stronger notion of excluding some patterns.

In particular, we consider *H-free graphs*, i.e., graphs without an induced copy of graph H. Those graphs are obviously hereditary. Despite the easy description, such graphs are quite difficult to analyze and it seems that a deeper understanding of these graph classes or new tools should be developed. We focus on the coloring problem which is very well-studied on such graphs, see [77] and also Section 3.1.

1.4 Parameterized Complexity

The central concept of the thesis is parameterized complexity, which is one of the main tools in the study of algorithms and complexity, nowadays.

As we discussed, many classical problems are NP-complete [75], and therefore, they are not well-scalable even for modern computers. In the run to overcome this problem, many approaches are considered. We can either sacrifice the optimality of the solution, and therefore, aim for approximation algorithms. Or we can find an additional structure in the input data and measure the time complexity in both, the size of the input and the chosen parameters. Those algorithms are called *Parameterized Algorithms.* The history of this field started by a series of paper by Downey and Fellows [46, 48, 49, 47, 1] with the very first paper that appeared on FOCS conference in 1989 [2]. Since the publication of a seminal book [50] by Downey and Fellows in 1999, Parameterized Complexity became one of the most important fields in algorithm study with plenty of essential publications each year and a handful of books, e.g., [51, 60] among others. The very recent development in the field is covered in a book by Cygan et al. [41].

First, we define a parameterized problem.

Definition 1.1. *A* parameterized problem *is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed alphabet and Σ^* is an arbitrary string over the alphabet. Then for an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the* parameter*.*

The class of the effective parameterized algorithms is called *Fixed Parameter Tractable*, in short *FPT*. It contains all the algorithms with running time $f(k)n^c$, where f is an arbitrary computable function, k represents the parameter, n is the size of the input, and c is a constant. Less effective is the class of XP algorithms with running time $n^{f(k)}$. However, for problems where an FPT algorithm probably does not exist (W-hard problems, see Subsection 1.4.1), XP algorithms might still

k/n	50 (1 day)	100 (3.2T yrs)	500 (10^{133} yrs)	1000 (10^{283} yrs)
5	133 ns | 26 ms	266 ns | 0.8 s	1.3 μs | 43 mins	2.7 μs | 1 day
10	4 μs | 94 days	8 μs| 264 yrs	40 μs | 2.5G yrs	80 μs | 2.6T yrs
25	0.1 s | 10^{24} yrs	0.2 s | 10^{32} yrs	1 s | 10^{49} yrs	2 s | 10^{57} yrs

Table 1.2: Comparisons of running times between naive exponential (2^n), XP (n^k) and FPT ($2^k n$) algorithms on a current computer (4 cores, 3GHz). Naive running times are in the first row in brackets, FPT | XP running times are within the inner-fields of the table. Shortcut yrs stands for years.

be a very good choice. There exists a similar hierarchy of hard problems as in the cla

To support the importance of the field in practical computations, we provide Table 1.2 that sums up the differences in the running time of the efficient algorithms—a comparison of both FPT and XP with a naive single exponential algorithm.

1.4.1 Parameterized Reduction

It is usually useful to derive hardness results alongside with the development of algorithms. On the first glance, it seems that whenever we have a parameter, such that the problem is NP-hard even when the parameter is constant, then the problem is hard for parameterization. More formally, if the problem is NP-hard for all values of a parameter larger than some constant, then it is called *para*-NP*-hard*. This reasoning rules out the existence of not only FPT algorithms but also XP algorithms (unless P = NP). For example, check Theorem 2.6.

We define the parameterized reduction that is useful to show evidences that some problems would unlikely allow an FPT algorithm.

Definition 1.2. *Let A, B be two parameterized problems.* Parameterized reduction *is an algorithm that given an instance (x, k) of A outputs an instance (x', k') of B such that:*

- *(x, k) is yes-instance of A if and only if (x', k') is yes-instance of B,*
- *$k' \leq g(k)$, for a computable function g,*
- *the reduction algorithm runs in time $f(k) \cdot |x|^{\mathcal{O}(1)}$, for a computable function f.*

The parameterized reduction is similar to the classical polynomial-time reduction that is used to show NP-hardness. However, in fact, both reductions are incomparable, although many existing reductions fit both contexts. It can be observed that the parameterized reduction preserves containment in FPT class, i.e., if there is a parameterized reduction from A to B and problem B is in FPT then also problem A is in the FPT class.

Since even P $\neq$ NP is far from being proven then also a classification of problems hard for parameterized algorithms has to be based on some assumption. Usually, even a bit stronger assumption than P $\neq$ NP is used. There is a nested hierarchy of problems, where all the problems in one class are equivalent under a parameterized reduction:

$$\text{FPT} = \mathsf{W[0]} \subseteq \mathsf{W[1]} \dots$$

Problems in classes $\mathsf{W[t]}$ for $t \geq 1$ are unlikely to admit an FPT algorithm. The proof of Theorem 5.2 is an example of parameterized reduction showing a $\mathsf{W[1]}$-hard problem. Details of the definition of W-hierarchy is omitted s for our purposes it suffices to refer to problems that have been shown $\mathsf{W[t]}$-hard elsewhere. For a more detailed introduction to the topic consult the book [41] or the original paper [48].

1.4.2 Kernelization

Kernelization plays an important role in many results from the parameterized algorithms field.

Definition 1.3. *Let L be parameterized problem. A* kernelization algorithm *or* kernelization *is an algorithm such that for any instance (x, k) it outputs in time polynomial in $|(x, k)|$ a string $x' \in \Sigma^*$ and an integer $k' \in \mathbb{N}$ such that*

$$(x, k) \in L \Leftrightarrow \Big((x', k') \in L \text{ and } |x'|, k' \leq h(k)\Big)$$

where h is an arbitrary computable function. Function h is called the size *of the kernel.*

Definition 1.3 is sometimes strenghtened in a way that $k' \leq k$. It is well-known that the problem is in FPT if and only if it has a kernel [21]. One implication is easy since the kernelization algorithm takes polynomial time and then the problem can be solved exhaustively on the kernel in time independent on the original size of the input and dependent only on the parameter(s).

In particular, a big effort was spent in obtaining polynomial kernels or in negative result—refuting the existence of a polynomial kernel. Kernelization algorithm is described in the proof the main theorem in Section 5.1. Refuting the polynomial kernel is briefly shown in Lemma 5.4. Very recently (2019), the entire book devoted to Kernelization has appeared [61]. It is a good source of further details.

1.4.3 Structural Graph Parameters

In this work, we considered structural parameters in Chapters 2, 4, and 5. See Figure 1.1 for an overview of the parameters.

We define structural parameters for a graph $G = (V, E)$. The most famous structural parameter is treewidth. The history of this parameter reaches back to Bertelè and Briochy in 1972 [9]. A modern equivalent defintion works with a term called *tree decomposition* of graph G. It is defined as a tree T with nodes $X_i \subseteq V(G)$, satisfying the following properties:

- $V(G) = \cup_i X_i$.
- For each $\{u, v\} \in E(G)$ exists $t \in V(T)$ such that $u, v \in X_t$.

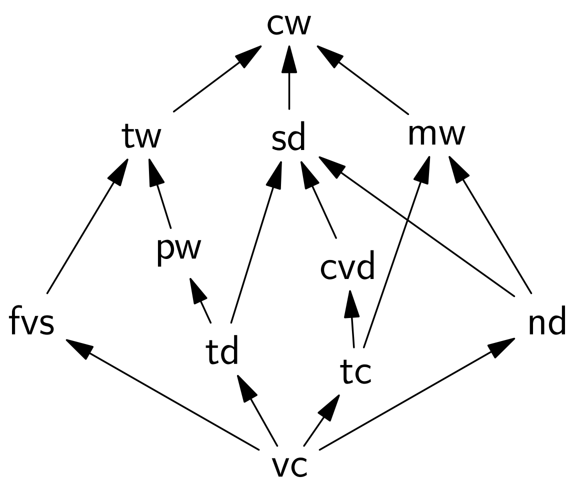

Figure 1.1: Hierarchy of graph parameters considered in the thesis. An arrow indicates that a graph parameter upper-bounds the other.

- For each $v \in V(G)$, nodes that contain the vertex v induce a connected subgraph in T.

The *size* of a tree decomposition is the maximum size of the node. Then the *treewidth* ($\mathsf{tw}(G)$) is the minimum size of a tree decompostion taken over all tree decompositions of the graph G. Computing the exact value the treewidth is NP-complete [5], however, it can be computed in FPT time parameterized by its size [10]. We note for curiosity that the two years of an annual PACE challenge competition were devoted to computing the treewidth as good as possible [43, 44].

The *pathwidth* ($\mathsf{pw}(G)$) is the minimum size of the tree decompositions of graph G, where the decomposition is restricted only to a simple path. This gives a trivial inclusion. Even weaker graph parameter is the *treedepth* of a graph G ($\mathsf{td}(G)$). It is defined as the minimum height of a rooted forest whose transitive closure contains the graph G [126]. The *feedback vertex set* ($\mathsf{fvs}(G)$) is the minimum number of vertices of a graph G whose removal leaves a graph without cycles. The simplest considered parameter is *vertex cover* ($\mathsf{vc}(G)$), whih is the minimum number of vertices of a graph G, whose removal leaves an edgeless graph. All the above-mentioned parameters form so-called sparse parameters.

Dense graph parameters follow—as oposed to sparse, cliques has the following parameters bounded. The *neighborhood diversity* ($\mathsf{nd}(G)$) is the smallest integer r, such that the graph can be partitioned into r sets, where each set is either complete graph or an independent set and each pair of sets forms either a complete bipartite graph or there is no edge between them.

A more complicated conept is the *modular width* of a graph G ($\mathsf{mw}(G)$), whih is the smallest positive integer r such that G can be obtained from an algebraic expression of width at most r, defined as follows. The *width of an expression* A is the maximum number of operands used by any occurrence of the substitution operation in A, where A is an algebraic expression that uses the following operations:

1. Create an isolated vertex.

2. The *substitution operation* with respect to a template graph T with vertex set $[r]$ and graphs $G_1, \ldots, G_r$ created by algebraic expression. The substitution

operation, denoted by $T(G_1, \ldots, G_r)$, results in the graph on vertex set $V = V_1 \cup \cdots \cup V_r$ and edge set $E = E_1 \cup \cdots \cup E_r \cup \bigcup_{\{i,j\} \in E(T)} \Big\{ \{u, v\} : u \in V_i, v \in V_j \Big\}$, where $G_i = (V_i, E_i)$ for all $i \in [r]$.

An algebraic expression of width $\mathsf{mw}(G)$ can be computed in linear time [137]. The *twin cover* $\mathsf{tc}(G)$ (introduced by Ganian [69]) is one possible generalization of vertex cover. It is defined as the number of vertices needed to cover all edges of graph G that are not twin-edges; an edge $\{u, v\}$ is a *twin-edge* if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. Here, we measure the number of vertices needed to cover all edges that are not twin-edges; an edge $\{u, v\}$ is a *twin-edge* if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. A more general parameter than twin cover is *cluster vertex deletion* ($\mathsf{cvd}(G)$), that is, the smallest number of vertices one has to delete from a graph in order to get a collection of (disjoint) cliques. The *shrub dept* ($\mathsf{sd}(G)$) is defined in [72] and since it is here only for completeness of the picture, we omit here the formal definition. The last presented and the most powerfull is the *cliquewidth* ($\mathsf{cw}(G)$) that upper-bounds all the studied parameters. This concept has been defined in [35] already in 1993. It is defined as the minimum number of colors used in the following process. Starting with an empty graph, the given graph G is created by the following operations.

- Create a vertex of a certain color.
- Make the disjoint union of two colored graphs.
- Add all the edges between vertices of two different colors.
- Change color of all the vertices of a certain color to a different color.

1.5 Graph Logics

Graph properties can be formally specified and modeled by a suitable logic. This concept is heavily exploited in Chapters 4 and 5. We provide here only a brief and rather an informal introduction to the subject. For more extensive description We recommend book [116] which explores the use of logic from graphs to finite models. In the basic setting, a logic has access to the graph and it can make queries, whether two vertices are adjacent or not. Standard and well-examined logics for graphs are MSO_2, MSO_1, and FO. The simplest among them is the FO logic. There the only allowed quantification is over elements (vertices or edges) of the graph. A strictly more powerful is MSO_1 logic, where in addition the quantification over the sets of vertices is possible. The most powerful out of them is MSO_2, where in addition even the quantification over the set of edges is allowed. For example, the hamiltonicity (the existence of a path visiting all the vertices of a given graph) can be expressed in MSO_2 but not in MSO_1 [34, 116]. The connectivity in the graph is an example of a property, that can be expressed in MSO_1 but not in FO (not even in existential MSO_2, where only existential quantification is allowed) [57].

1.5.1 Graph Metatheorems

Undoubtedly, Courcelle's Theorem [33] for graph properties expressible in the monadic second-order logic (MSO_2) on graphs of bounded treewidth, as well as an MSO_1 algorithm on graphs of bounded clique-width, play a prime role among model checking algorithms. In particular, Courcelle's Theorem provides for an MSO_2 sentence φ an algorithm, that given an n-vertex graph G with treewidth k decides, whether φ holds in G in time $f(k, |\varphi|)n$, where f is some computable function and $|\varphi|$ is the quantifier depth of φ. In other words, we obtain an FPT algorithm parameterized by treewidth and quantifier depth of the formula. Similar results are known for a weaker MSO_1 logic and clique-width. We cannot hope for much more on dense graph classes since MSO_2 model checking is not even in XP on graphs as simple as cliques unless E = NE [112].

There are many more FPT model checking specialized algorithms, e.g., an algorithm for (existential counting) modal logic model checking on graphs of bounded treewidth [129], MSO model checking on graphs of bounded neighborhood diversity [111], or MSO model checking on graphs of bounded shrubdepth [73] (generalizing the previous result). First order logic (FO) model checking received recently quite some attention as well and algorithms for graphs with bounded degree [133], nowhere dense graphs [81], and some dense graph classes [67] were given. Chapter 5 or paper [100] contains more discussion on the topic.

1.6 Organization of Thesis

The thesis is composed of 3 papers and of one problem summary with annotated bibliography. They are presented in the original journal form with only a small adjustment of their layout and few additional references. The first result appeared in Information Processing Letters journal. The second was published on ISAAC 2018 conference and is currently under reviews in a journal. The third has just been accepted to MFCS 2019 conference and we will submit the full version to a journal shortly. In addition, a short summary of fair problems is provided in Chapter 4.

Chapter 2 of the thesis is devoted to the packing coloring problem. A graph labeling problem that has its motivation in assigning frequencies to transmitters, under its original name, Broadcasting Chromatic Number. It was first formulated by Goddard, Hedetniemi, Hedetniemi, Harris, and Rall [76]. The presented results were published as *Minki Kim, Bernard Lidický, Tomáš Masařík, and Florian Pfender. Notes on the complexity of packing coloring. Information Processing Letters, 137:6–10, 2018. doi:10.1016/j.ipl.2018.04.012.* [97]. We slightly improved the hardness result on chordal graphs given by Fiala and Golovach [59]. Moreover, several positive results were developed mostly for interval graphs and in terms of parameterized complexity.

Chapter 3 describes a result in a clasical 3-coloring setting. We examined H-free graphs, that are graphs without an induced copy of H. In *Tereza Klimošová, Josef Malík, Tomáš Masařík, Jana Novotná, Daniël Paulusma, and Veronika Slívová. Colouring (P_r+P_s)-free graphs. In 29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16–19, 2018, Jiaoxi, Yilan, Taiwan, pages 5:1–5:13, 2018. doi:10.4230/LIPIcs.ISAAC.2018.5.* [98] we showed that list 3-coloring is polynomial time solvable on $P_2 + P_5$-free and $P_3 + P_4$-free graphs where P_i represents a path on i vertices and symbol + denotes disjoint union. By this result, we completed the characterization for 3-coloring of H-free graphs for any H up to seven vertices. The submitted journal version with full proofs is also available on arxiv [99].

Both Packing coloring and Coloring of H-free graphs are very heavily studied and several overview papers were published, see e.g., [77]. However, Fair problems have not received much attention yet, therefore we add a short annotated summary of known results from this field in Chapter 4. This summary also contains connections with a closely related topic of defective coloring.

Chapter 5 contains a paper *Dušan Knop, Tomáš Masařík, and Tomáš Toufar. Parameterized Complexity of Fair Vertex Evaluation Problems. In 44rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26–30 2019, Aachen, Germany, pages 8:1–8:16, 2019.* [102]. The full version we are about to submit to journal containing full proofs is also available on arxiv [101]. It is a part of a larger project on studying fair problems and their extensions. They have been mostly studied from the metatheorem point of view in papers [100, 122, 102]. The main presented result is a metatheorem for the twin cover. Moreover, the notion of Fair Vertex cover is formulated. For the Fair Vertex Cover problem, we derive an algorithm parameterized by modular-width and a W[1]-hardness for a combination of tree depth and feedback vertex set. The paper presents a few additional hardness results and further research directions.

Instead of one common Conclusion section at the end of the thesis, each of Chapters 2, 3, 4, and 5 has its own Conclusions (see 2.3, 3.4, and 5.5) with a brief summary, open problems and several new directions for the future research. Chapter 4.4 is devoted to an extended discussion about open problems and interesting directions for future research.

2. Complexity of Packing Coloring

2.1 Introduction

Given a graph $G = (V, E)$ and an integer k, a *packing k-coloring* is a mapping $\varphi : V \to \{1, \ldots, k\}$ such that any two vertices u, v of color $\varphi(u) = \varphi(v)$ are in distance at least $\varphi(u) + 1$. An equivalent way of defining the packing k-coloring of G is that it is a partition of V into sets $V_1, \ldots, V_k$ such that for all k and any $u, v \in V_k$, the distance between u and v is at least $k + 1$. The *packing chromatic number* of G, denoted $\chi_P(G)$, is the smallest k such there exists a packing k-coloring of G.

The definition of packing k-coloring is motivated by frequency assignment problems. It emphasizes the fact that the signal on different frequencies can travel different distances. In particular, lower frequencies, modeled by higher colors, travel further so they may be used less often than higher frequencies. The packing coloring problem was introduced by Goddard et al. [76] under the name *broadcasting chromatic number.* The term packing coloring was introduced by Brešar, Klavžar, and Rall [17].

Determining the packing chromatic number is often difficult. For example, Sloper [134] showed that the packing chromatic number of the infinite 3-regular tree is 7 but the infinite 4-regular tree does not admit any packing coloring by a finite number of colors. Results of Brešar, Klavžar, and Rall [17] and Fiala, Klavžar, and Lidický [58] imply that the packing chromatic number of the infinite hexagonal lattice is 7.

Looking at these examples, researchers asked the question if there exists a constant p such that every subcubic graph has packing chromatic number bounded by p. A very recent result of Balogh, Kostochka and Liu [7] shows that there is no such p in quite a strong sense. They show that for every fixed k and $g \geq 2k + 2$, almost every n-vertex cubic graph of girth at least g has packing chromatic number greater than k. It is still open if a constant bound holds for planar subcubic graphs, and no deterministic construction of subcubic graphs with arbitrarily high packing chromatic number is known.

Despite a lot of effort [58, 76, 119, 135], the packing chromatic number of the square grid is still not determined. It is known to be between 13 and 15 due to Barnaby, Franco, Taolue, and Jos [119], who use state of the art SAT-solvers to tackle the problem. In this paper, we consider the packing coloring problem from the computational complexity point of view. In particular, we study the following problem.

Packing k-coloring of a graph
Instance: A graph G and a positive integer k.
Question: Does G allow a packing k-coloring?

2.1.1 Known Results

We characterize our algorithmic parameterized results in terms of FPT (running time $f(k)\text{poly}(n)$) and XP (running time $n^{f(k)}$) where n is the size of the input, k is the parameter and f is any computable function. The investigation of computational complexity of packing coloring was started by Goddard et al. [76] in 2008. They showed that PACKING k-COLORING is NP-complete for general graphs and $k = 4$ and it is polynomial time solvable for $k \leq 3$. Fiala and Golovach [59] showed that PACKING k-COLORING is NP-complete for trees for large k (dependent on the number of vertices).

For a fixed k, PACKING k-COLORING is expressible in MSO_1 logic. Thus, due to Courcelle's theorem [33], it admits a fixed parameter tractable (FPT) algorithm parameterized by the tree-width or clique-width [36] of the graph. Moreover, it is solvable in polynomial time if both the tree-width and the diameter are bounded [59]. The problem remains in FPT even if we fix the number of colors that can be used more than once by the extended framework of Courcelle, Makowsky and Rotics [36], see Theorem 2.11. On the other hand, the problem is NP-complete for chordal graphs of diameter exactly 5 [59], and it is polynomial time solvable for split graphs [76]. Note that split graphs are chordal and have diameter at most 3. However, PACKING k-COLORING admits an FPT algorithm on chordal graphs parameterized by k [59].

2.1.2 Our Results and Structure of the Paper

In Section 2.2, we describe new complexity results on chordal, interval and proper interval graphs. We improve a result by Fiala and Golovach [59] in Theorem 2.5, where we show that computing packing chromatic number of chordal graphs of any diameter greater or equal than three is NP-complete. Moreover, the reduction implies an inapproximability result based on the inapproximability of the size of the largest independent set. Proposition 2.3 shows that calculating the packing chromatic number of chordal graphs of diameter less than three is can be done in polynomial time.

We complement these results by several FPT and XP algorithms for calculating the packing chromatic number on interval and proper interval graphs. We use dynamic programming to get an XP algorithm for interval graphs of bounded diameter, see Theorem 2.6. For unit interval graphs, there is an FPT algorithm parameterized by the size of the largest clique, see Theorem 2.9. Note that the existence of an FPT algorithm for calculating the packing chromatic number parameterized by path-width would imply an FPT algorithm for general interval graphs parameterized by the size of the largest clique, but the existence of such algorithm remains an open question. We also provide an XP algorithm calculating the packing chromatic number for interval graphs parameterized by the number of colors that can be used more than once, see Theorem 2.10.

In Subsection 2.2.1, we describe complexity results and algorithms parameterized by structural parameters. We design FPT algorithms for them. For standard notation and terminology, we refer to the recent book about parameterized complexity [41].

The packing coloring problem is interesting only when the number of colors is not bounded. Otherwise, we can easily model the problem by a fixed MSO_1 formula and use the FPT algorithm by Courcelle [33] parameterized by the clique-width of the graph. We show that we can do similar modeling even when we fix only the number of colors that can be used more than once and then use a stronger result by Courcelle, Makowski and Rotics [36] that gives an FPT algorithm parameterized by clique-width of the graph (Theorem 2.11).

If the number of such colors is part of the input, then we can solve the problem on several graph classes. If they have a bounded diameter, then we can use Theorem 2.11 due to the following easy observation.

Observation 2.1. *Let G be a graph of bounded diameter. Then G has a bounded number of colors that can be used more than once.*

This observation together with Theorem 2.11 implies that the problem is FPT for any class of graphs of bounded shrub-depth. Any class of graphs that has bounded shrub-depth has a bounded length of induced paths ([71], Theorem 3.7) and thus bounded diameter. The same holds for graphs of bounded modular-width as they have bounded diameter according to Observation 2.2. On the other hand, the problem was shown to be hard on graphs of bounded tree-width [59], in fact the problem is NP-hard even on trees. There seems to be a big gap and thus interesting question about parameterized complexity with respect to path-width of the graph. It still remains open (Question 2.14). Note that the original hardness reduction by Fiala and Golovach [59] has unbounded path-width. See Figure 2.2 for an overview of the results with respect to the structural parameters. We refer [68] for the definition of modular-width and its construction operations.

Observation 2.2. *Let G be a graph of modular-width k. Then G has diameter at most* $\max(k, 2)$.

Proof. We look at the last step of the decomposition. It has to create a connected graph and thus it is either a join operation or a template operation. If it is the join operation then the diameter is at most 2 and if it is the template operation the longest path between any two vertices in different operands is at most k and if they are in the same operand their distance is at most 2. □

2.2 Chordal and Interval graphs

Proposition 2.3. *Packing chromatic number is in P for chordal graphs of diameter 2.*

Proof. Let G be a chordal graph of diameter 2. Notice that in graphs of diameter 2, the only color that can be used more than once is color 1. Hence, determining the packing chromatic number of G is equivalent to finding a largest independent set in G. In chordal graphs, the largest independent set can be found in polynomial time. Hence $\chi_P(G)$ can be found in polynomial time. □

For larger diameters, we use a similar reduction as Fiala and Golovach [59] to finding a largest independent set in a general graph. ZPP is a complexity class of problems which can be solved in expected polynomial time by a probabilistic algorithm that never makes an error. It lies between P and NP (P $\subseteq$ ZPP $\subseteq$ NP). It is strongly believed that ZPP $\neq$ NP. Håstad [91] showed that finding a largest independent set is hard to approximate.

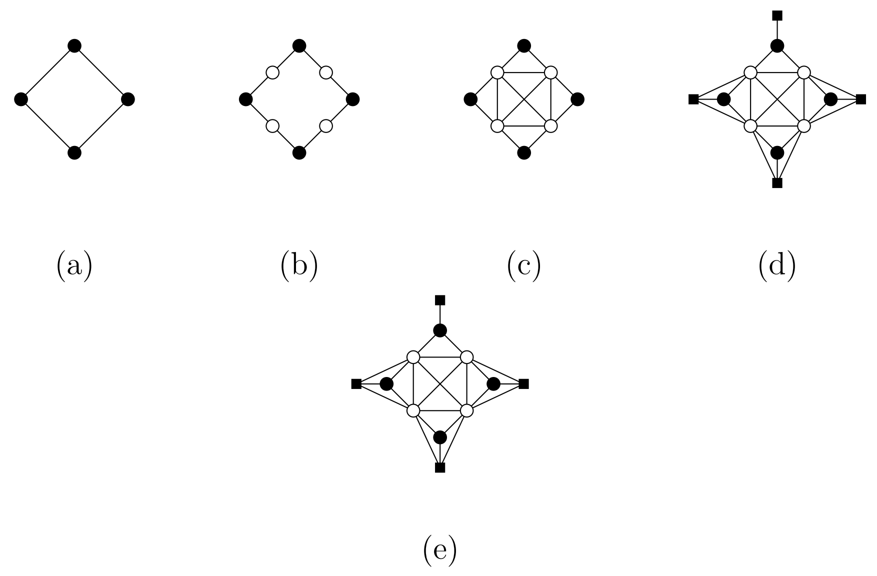

Figure 2.1: The reduction from Theorem 2.5 on a 4-cycle.

Theorem 2.4 (Håstad [91])**.** *Unless* NP = ZPP*, Independent set cannot be approximated within* $n^{1-\varepsilon}$ *for any* $\varepsilon > 0$ *on graphs on* n *vertices.*

Together with our reduction, this implies that the packing chromatic number is hard to approximate.

Theorem 2.5. *Packing chromatic number is* NP*-complete on chordal graphs of any diameter at least 3 on* n_H *vertices. Moreover, it is hard to approximate within* ${n_H}^{\frac{1}{2}-\varepsilon}$ *for any* $\varepsilon > 0$ *and any fixed diameter at least 3, unless* NP = ZPP*.*

Proof. We use a reduction to the independent set problem. Let G be any connected graph on n vertices. We construct a chordal graph H of diameter $d \geq 3$ from G by the following sequence of operations:

(a) start with G, denote the set of its vertices by V,

(b) subdivide every edge once, denote the set of new vertices by S,

(c) add all possible edges between vertices in S,

(d) for every $v \in V$ add a duplicate vertex v' and the edge vv'; denote the set of new duplicate vertices by D,

(e) to increase the diameter to $d > 3$, add a path P of length $d - 2$ starting in one vertex in S.

See Figure 2.1 for an example of the construction.

We will choose a packing coloring φ of H with $\chi_P(H)$ colors. Notice that the graph induced by $V \cup S \cup D$ has diameter at most three. Hence, only colors 1 and 2 can be used more than once on $V \cup S \cup D$. We call colors other than 1 and 2 *unique*. Notice that we can freely permute the unique colors. Pick φ in a way to maximize the number of unique colors among vertices in S, and subject to that, to maximize the number of vertices in D colored 1. We will show that S has only vertices of unique colors and all vertices in D are colored 1.

Suppose for the sake of contradiction that there is a vertex $s \in S$ colored 1 or 2. Since S is a clique, s is the only vertex in S with this color. Let $u \in D \cup V$ be a neighbor of s with a unique color. Such a vertex must exist since s has four neighbors in $D \cup V$, and at most two can be colored by 1 and 2. Observe that by the construction of H, the closed neighborhood $N[u] \subseteq N[s]$. Thus, for every vertex $w \neq u$, the distance $d(w,u) \leq d(w,s)$. Hence, we can swap the colors on s and u, contradicting the choice of φ. Therefore, all vertices in S have unique colors. Now let $x \in D$ and let v be its unique neighbor in V. If v has color 1, we can swap the colors on x and v, contradicting our choice of φ. Therefore, no vertices in $N(x)$ have color 1, and thus x has color 1 by our choice of φ.

Since all vertices in D are colored 1, no vertex in V can be colored 1. Minimizing the number of unique colors on V is the same as maximizing the number of vertices colored 2. By the distance constraints in H, a subset of V can be colored 2 in H if and only if it is an independent set in G. Therefore, the vertices colored 2 in V form a largest independent set in G. For the other implication, if we have a maximum independent set in G then we can color its vertices by 2. All the other vertices in V and in S are colored by a unique color and vertices in D are colored 1. This coloring is by the above discussion optimal.

Recall that in order to increase the diameter of H, we added the path P with one endpoint $s \in S$ in step (e). Notice that P can be colored by a pattern of four colors starting in s: $\varphi(s), 1, 2, 1, 3, 1, 2, 1, 3, 1 \ldots$. The existence of the path neither increases $\chi_P(H)$ nor influences the coloring φ in $V \cup D \cup S$.

Finally, notice that H has at most $n_H := \binom{n}{2} + 2n + d - 2$ vertices, where n is the number of vertices of the original graph G.

Recall that $\chi_P(H) = n_H - (n-1) - (\alpha(G)-1) - (d-2)$, where $n-1$ correspond to n vertices colored by color 1, $\alpha(G) - 1$ are vertices colored by color 2 and $d-2$ are extra vertices used in the diameter increasing path, which does not change $\chi_P(H)$. Therefore, $\chi_P(H) = \frac{n^2}{2} + \frac{n}{2} + 2 - \alpha(G)$. Hence, if we could approximate $\chi_P(H)$ with factor $(n_H)^{\frac{1}{2}-\varepsilon}$ for some $\varepsilon > 0$ and the fixed diameter d, we could approximate size of a largest independent set in G with the same factor, that can be bounded as

$$(n_H)^{\frac{1}{2}-\varepsilon} \leq c \cdot (n^2)^{\frac{1}{2}-\varepsilon} = c \cdot n^{1-2\varepsilon}$$

for some positive constant c, which contradicts Theorem 2.4. □

Theorem 2.6. *Packing chromatic number for interval graphs of diameter d can be solved in time $O(n^{d\ln(5d)})$.*

Proof. Let φ be a packing coloring of an interval graph G with diameter d, and let P be a diameter path in G. Note that every interval corresponding to a vertex of G intersects an interval corresponding to an internal vertex of P. Suppose X is a set colored by color $c \geq 2$ in φ. Internal vertices of P that are neighbors of X are in distance at least $c-1$. Since there is at most $d-2$ of them, $|X| \leq \frac{d-2}{c-1} + 1$.

Therefore, only colors $1, \ldots, d-1$ can be used more than once by φ. Notice that the number of vertices colored by $2, \ldots, d-1$ is upper bounded by

$$f(d) = \sum_{2 \leq c \leq d-1} \left(\frac{d-2}{c-1} + 1 \right) = (d-2)(1 + H(d-2)) < d\ln(5d) - 1,$$

where $H(n)$ is the harmonic number. There are at most $n^{f(d)}$ such partial colorings of G by colors $2, \dots, d-1$. Finally, vertices colored by 1 form an independent set. Therefore, the following is an algorithm to find the packing chromatic number of G.

Enumerate all $n^{f(d)}$ partial colorings by colors $2, \dots, d-1$. For each partial coloring, find a maximum independent set in the remaining graph, which takes time $O(n)$ and color the remaining vertices with unique colors. The whole algorithm runs in time $O(n^{f(d)+1}) = O(n^{d \ln(5d)})$. □

Theorem 2.6 can be restated also for AT-free graphs as an anonymous reviewer suggested. In fact, it holds for any graph class where for every vertex of a graph holds that it is a neighbor of a vertex of a diameter path and where maximum independent set is polynomial-time solvable. For AT-free graphs the first trivially holds and the second was shown in [20].

When restricting the class of graphs to unit interval graphs, we can find an FPT algorithm parametrized by the size of the largest clique, independent of diameter. We need the following two results.

Lemma 2.7 (Goddard et al. [76])**.** *For every $s \in \mathbb{N}$, the infinite path can be colored by colors $s, s+1, \dots, 3s+2$.*

Proposition 2.8 (Fiala and Golovach [59])**.** *Chordal graphs admit an* FPT *algorithm parameterized by the number of colors used in the solution.*

Theorem 2.9. *Packing chromatic number for unit interval graphs with a largest clique of size at most k is* FPT *in k.*

Proof. Let G be a unit interval graph. As G is perfect, we can find a partition of its vertex set into k independent sets $X_1, \dots, X_k$ in polynomial time. Let $X_\ell = \{v_1, v_2, \dots, v_{|X_\ell|}\}$, where the v_i are ordered corresponding to their interval representation. Note that for all $i < j$, the distance of v_i and v_j in G is at least $j - i$. This implies that any packing coloring of a path on $|X_\ell|$ vertices can be used to packing color the set X_ℓ without conflicts.

Use Lemma 2.7 to color each X_ℓ with colors $\{\frac{5}{2}(3^{\ell-1}-1)+1, \dots, \frac{5}{2}(3^\ell - 1)\}$, and notice that these color sets are disjoint. This yields a packing coloring of G with at most $\frac{5}{2}(3^k - 1)$ colors. Therefore, the number of colors we need is bounded in terms of k, and we can apply Theorem 2.8 to conclude the proof. □

In the previous argument, we saw that restricting the number of colors makes the problem simpler. While we obviously do not have such a restriction for all interval graphs, we can still achieve a result about partial packing colorings with a bounded number of colors along similar ideas.

Theorem 2.10. *Let k be fixed and G be an interval graph. Finding a partial coloring by colors $1, \dots, k$ that is maximizing the number of colored vertices can be solved in time $O(n^{k+1})$.*

Proof. We compute a function $H(u_1, \dots, u_k) \to \mathbb{N}$, which counts the maximum number of colored vertices such that vertex u_i has its interval with the right end-point most to the right among all vertices colored by color i. The domain of H is $(V \cup \{N\})^k$, where N is a symbol representing that a particular color was not used at all. There are $(n+1)^k$ such functions, so we can pick the optimal one.

We show how to compute H using dynamic programming in time $O(n^{k+1})$. We order ($\prec$) vertices by the right-most endpoint of their interval. We compute H based on the order on vertices, obviously, $H(N, \dots, N) = 0$. We try all colorings of vertex v, that means we compute $H(\dots, v_\ell, \dots)$ for every $\ell \in [k]$ where $\dots$ stands for any combination of previous vertices and N. If vertex v is colored by ℓ in H we find the largest value among $H(\dots, u_\ell, \dots)$ for any u in distance greater than ℓ and $u \prec v$. There are at most n such vertices. So we set

$$H(\dots, v_\ell, \dots) = \max_{\mathrm{dist}(v,u)>\ell \ \wedge\ u \prec v} H(\dots, u_\ell, \dots) + 1. \qquad \square$$

Notice that Theorem 2.10 implies Theorem 2.6 with a smaller exponent in the running time.

2.2.1 Structural Parameters

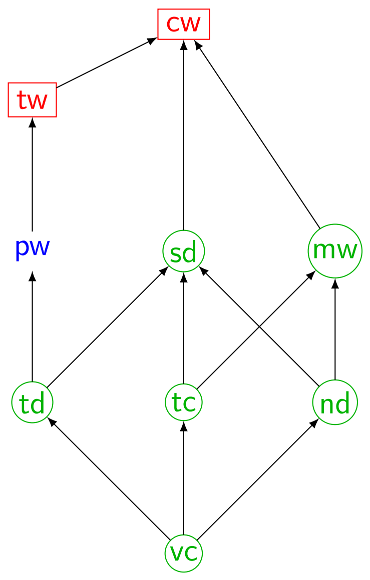

Figure 2.2: Hierarchy of graph parameters. An arrow indicates that a graph parameter upper-bounds the other. Thus, hardness results are implied in the direction of arrows and algorithms are implied in the reverse direction. Green circles and red rectangle colors distinguish between hardness results and FPT algorithms provided. Blue color without boundary denotes that the hardness is unknown. (cw is clique-width, nd is neighborhood diversity, mw is modular-width, pw is path-width, sd is shrub-depth, tc is twin cover, td is tree-depth, tw is tree-width, vc is vertex cover. See [41] for definitions.)

Theorem 2.11. *Let k be fixed and $G = (V, E)$ be a graph of clique-width q. Finding a partial packing coloring by colors $1, \dots, k$ that is maximizing the number of colored vertices can be solved in* FPT *time parameterized by q.*

Proof. We model the problem as an extended formulation in MSO_1 logic with one free variable X that represents the large colors. We use a result by Courcelle, Makowski and Rotics [36] to solve this formula $\varphi(X)$ on graphs of clique-width q in FPT time such that it minimizes the size of the set X.

$$d(x,y) \geq i \models \nexists z_1, \dots, z_{i-1} \in V \text{ such that}$$
$$x = z_1 \wedge y = z_{i-1} \wedge \cup_{j=1}^{i-2} (\mathrm{edge}(z_j, z_{j+1}) \vee (z_j = z_{j+1})).$$

$$i\text{-independent}(X) \models \forall x, y \in X \ d(x,y) \geq i.$$

$$\varphi(X) \models \exists X_1, \dots X_k \subseteq V \text{ s.t. } \forall i \ i\text{-independent}(X_i) \wedge V = X \dot\cup X_1 \dot\cup \cdots \dot\cup X_k.$$

$\square$

2.3 Conclusion

Although the diameter is a widely investigated structural parameter we found that in some cases a related parameter better captures the problem, namely the number of colors that can be used more than once, as we show in Theorem 2.10. We close with a few open questions.

Question 2.12. *Is determining the packing chromatic number for (unit) interval graphs in* P *or is it* NP*-hard?*

Question 2.13. *Is determining the packing chromatic number for interval graphs* FPT *when parametrized by the largest clique size?*

One can think of graphs of bounded path-width as a generalization of interval graphs with bounded clique size. This leads to the following question. Notice that Theorem 2.9 could be modified to work on graphs of bounded path-width that have a decomposition such that every vertex is in a bounded number of bags.

Question 2.14. *Is determining the packing chromatic number* FPT *or* XP *when parametrized by the path-width?*

3. Coloring of H-free Graphs

3.1 Introduction

Graph colouring is a popular concept in Computer Science and Mathematics due to a wide range of practical and theoretical applications, as evidenced by numerous surveys and books on graph colouring and many of its variants (see, for example, [4, 25, 77, 93, 108, 128, 131, 140]). Formally, a *colouring* of a graph $G = (V, E)$ is a mapping $c : V \to \{1, 2, \ldots\}$ that assigns each vertex $u \in V$ a *colour* $c(u)$ in such a way that $c(u) \neq c(v)$ whenever $uv \in E$. If $1 \leq c(u) \leq k$, then c is also called a *k-colouring* of G and G is said to be *k-colourable*. The Colouring problem is to decide if a given graph G has a k-colouring for some given integer k.

It is well known that Colouring is NP-complete even if $k = 3$ [118]. To pinpoint the reason behind the computational hardness of Colouring one may impose restrictions on the input. This led to an extensive study of Colouring for special graph classes, particularly hereditary graph classes. A graph class is *hereditary* if it is closed under vertex deletion. As this is a natural property, hereditary graph classes capture a very large collection of well-studied graph classes. A classical result in this area is due to Grötschel, Lovász, and Schrijver [82], who proved that Colouring is polynomial-time solvable for perfect graphs.

It is readily seen that a graph class $\mathcal{G}$ is hereditary if and only if $\mathcal{G}$ can be characterized by a unique set $\mathcal{H}_{\mathcal{G}}$ of minimal forbidden induced subgraphs. If $\mathcal{H}_{\mathcal{G}} = \{H\}$, then a graph $G \in \mathcal{G}$ is called *H-free*. Hence, for a graph H, the class of H-free graphs consists of all graphs with no induced subgraph isomorphic to H.

Král', Kratochvíl, Tuza, and Woeginger [107] started a systematic study into the complexity of Colouring on $\mathcal{H}$-free graphs for sets $\mathcal{H}$ of size at most 2. They showed polynomial-time solvability if H is an induced subgraph of P_4 or $P_1 + P_3$ and NP-completeness for all other graphs H. The classification for the case where $\mathcal{H}$ has size 2 is far from finished; see the summary in [77] or an updated partial overview in [42] for further details. Instead of considering sets $\mathcal{H}$ of size 2, we consider H-free graphs and follow another well-studied direction, in which the number of colours k is *fixed*, that is, k no longer belongs to the input. This leads to the following decision problem:

k-Colouring	
Instance:	A graph G.
Question:	Does there exist a k-colouring of G?

A *k-list assignment* of G is a function L with domain V such that the *list of admissible colours* $L(u)$ of each $u \in V$ is a subset of $\{1, 2, \ldots, k\}$. A colouring c *respects* L if $c(u) \in L(u)$ for every $u \in V$. If k is fixed, then we obtain the following generalization of k-Colouring:

List k-Colouring	
Instance:	A graph G and a k-list assignment L.
Question:	Does there exist a colouring of G that respects L?

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

Table 3.1: Summary for P_t-free graphs.

For every $k \geq 3$, k-Colouring on H-free graphs is NP-complete if H contains a cycle [56] or an induced claw [89, 115]. Hence, it remains to consider the case where H is a *linear forest* (a disjoint union of paths). The situation is far from settled yet, although many partial results are known [15, 18, 19, 27, 28, 29, 30, 38, 78, 88, 90, 113, 130, 132, 142]. Particularly, the case where H is the t-vertex path P_t has been well studied. The cases $k = 4$, $t = 7$ and $k = 5$, $t = 6$ are NP-complete [90]. For $k \geq 1$, $t = 5$ [88] and $k = 3$, $t = 7$ [15], even List k-Colouring on P_t-free graphs is polynomial-time solvable (see also [77]).

For a fixed integer k, the k-Precolouring Extension problem is to decide if a given k-colouring c' defined on an induced subgraph G' of a graph G can be extended to a k-colouring c of G. Note that k-Colouring is a special case of k-Precolouring Extension, whereas the latter problem can be formulated as a special case of List k-Colouring by assigning list $\{c'(u)\}$ to every vertex u of G' and list $\{1, \ldots, k\}$ to every other vertex of G. Recently, it was shown in [28, 29] that 4-Precolouring Extension, and therefore 4-Colouring, is polynomial-time solvable for P_6-free graphs. In contrast, the more general problem List 4-Colouring is NP-complete for P_6-free graphs [78]. See Table 3.1 for a summary of all these results.

From Table 3.1 we see that only the cases $k = 3$, $t \geq 8$ are still open, although some partial results are known for k-Colouring for the case $k = 3$, $t = 8$ [30]. The situation when H is a disconnected linear forest $\bigcup P_i$ is less clear. It is known that for every $s \geq 1$, List 3-Colouring is polynomial-time solvable for sP_3-free graphs [19, 77]. For every graph H, List 3-Colouring is polynomial-time solvable for $(H + P_1)$-free graphs if it is polynomially solvable for H-free graphs [19, 77]. If $H = rP_1 + P_5$ ($r \geq 0$), then for every integer k, List k-Colouring is polynomial-time solvable on $(rP_1 + P_5)$-free graphs [38]. This result cannot be extended to larger linear forests H, as List 4-Colouring is NP-complete for P_6-free graphs [78] and List 5-Colouring is NP-complete for $(P_2 + P_4)$-free graphs [38].

A way of making progress is to complete a classification by bounding the size of H. It follows from the above results and the ones in Table 3.1 that for a graph H with $|V(H)| \leq 6$, 3-Colouring and List 3-Colouring (and consequently, 3-Precolouring Extension) are polynomial-time solvable on H-free graphs if H is a linear forest, and NP-complete otherwise (see also [77]). There are two open cases [77] that must be solved in order to obtain the same statement for graphs H with $|V(H)| \leq 7$. These cases are

- $H = P_2 + P_5$
- $H = P_3 + P_4$.

3.1.1 Our Results

In Section 3.2 we address the two missing cases listed above by proving the following theorem.

Theorem 3.1. List 3-Colouring *is polynomial-time solvable for* $(P_2 + P_5)$*-free graphs and for* $(P_3 + P_4)$*-free graphs.*

We prove Theorem 3.1 as follows. If the graph G of an instance (G, L) of List 3-Colouring is P_7-free, then we can use the aforementioned result of Bonomo et al. [15]. Hence we may assume that G contains an induced P_7. We consider every possibility of colouring the vertices of this P_7 and try to reduce each resulting instance to a polynomial number of smaller instances of 2-Satisfiability. As the latter problem can be solved in polynomial time, the total running time of the algorithm will be polynomial. The crucial proof ingredient is that we partition the set of vertices of G that do not belong to the P_7 into subsets of vertices that are of the same distance to the P_7. This leads to several "layers" of G. We analyse how the vertices of each layer are connected to each other and to vertices of adjacent layers so as to use this information in the design of our algorithm.

Combining Theorem 3.1 with the known results yields the following complexity classifications for graphs H up to seven vertices; see Section 3.3 for its proof.

Corollary 3.2. *Let H be a graph with $|V(H)| \leq 7$. If H is a linear forest, then* List 3-Colouring *is polynomial-time solvable for H-free graphs; otherwise already* 3-Colouring *is* NP*-complete for H-free graphs.*

3.1.2 Preliminaries

Let $G = (V, E)$ be a graph. For a vertex $v \in V$, we denote its *neighbourhood* by $N(v) = \{u \mid uv \in E\}$, its *closed neighbourhood* by $N[v] = N(v) \cup \{v\}$ and its degree by $\deg(v) = |N(v)|$. For a set $S \subseteq V$, we write $N(S) = \bigcup_{v \in S} N(v) \setminus S$ and $N[S] = N(S) \cup S$, and we let $G[S] = (S, \{uv \mid u, v \in S\})$ be the subgraph of G induced by S. The *contraction* of an edge $e = uv$ removes u and v from G and introduces a new vertex which is made adjacent to every vertex in $N(u) \cup N(v)$. The *identification* of a set $S \subseteq V$ by a vertex w removes all vertices of S from G, introduces w as a new vertex and makes w adjacent to every vertex in $N(S)$. The *length* of a path is its number of edges. The *distance* $\text{dist}_G(u, v)$ between two vertices u and v is the length of a shortest path between them in G. The *distance* $\text{dist}_G(u, S)$ between a vertex $u \in V$ and a set $S \subseteq V \setminus \{v\}$ is defined as $\min\{\text{dist}(u, v) \mid v \in S\}$.

For two graphs G and H, we use $G + H$ to denote the disjoint union of G and H, and we write rG to denote the disjoint union of r copies of G. Let (G, L) be an instance of List 3-Colouring. For $S \subseteq V(G)$, we write $L(S) = \bigcup_{u \in S} L(u)$. We let P_n and K_n denote the path and complete graph on n vertices, respectively. The *diamond* is the graph obtained from K_4 after removing an edge.

We say that an instance (G', L') is *smaller* than some other instance (G, L) of List 3-Colouring if either G' is an induced subgraph of G with $|V(G')| < |V(G)|$; or $G' = G$ and $L'(u) \subseteq L(u)$ for each $u \in V(G)$, such that there exists at least one vertex u^* with $L'(u^*) \subset L(u^*)$.

3.2 The Proof of Theorem 3.1

In this section we show that List 3-Colouring problem is polynomial-time solvable for $(P_2 + P_5)$-free graphs and for $(P_3 + P_4)$-free graphs. As arguments for these two graph classes are overlapping, we prove both cases simultaneously. Our proof uses the following two results.

Theorem 3.3 ([15]). List 3-Colouring *is polynomial-time solvable for P_7-free graphs.*

If we cannot apply Theorem 3.3, our strategy is to reduce, in polynomial time, an instance (G, L) of List 3-Colouring to a polynomial number of smaller instances of 2-List Colouring. We use the following well-known result due to Edwards.

Theorem 3.4 ([55]). *The* 2-List Colouring *problem is linear-time solvable.*

We are now ready to prove our main result , namely that List 3-Colouring is polynomial-time solvable for $(P_2 + P_5)$-free graphs and for $(P_3 + P_4)$-free graphs. As arguments for these two graph classes are overlapping, we prove both cases simultaneously. We start with an outline followed by a formal proof.

Outline of the proof of Theorem 3.1. Our goal is to reduce, in polynomial time, a given instance (G, L) of List 3-Colouring, where G is $(P_2 + P_5)$-free or $(P_3 + P_4)$-free, to a polynomial number of smaller instances of 2-List-Colouring in such a way that (G, L) is a yes-instance if and only if at least one of the new instances is a yes-instance. As for each of the smaller instances, we can apply Theorem 3.4, the total running time of our algorithm will be polynomial.

If G is P_7-free, then we do not have to do the above and may apply Theorem 3.3 instead. Hence, we assume that G contains an induced P_7. We put the vertices of the P_7 in a set N_0 and define sets N_i ($i \geq 1$) of vertices of the same distance i from N_0; we say that the sets N_i are the layers of G. We then analyse the structure of these layers using the fact that G is $(P_2 + P_5)$-free or $(P_3 + P_4)$-free. The first phase of our algorithm is about preprocessing (G, L) after colouring the seven vertices of N_0 and applying a number of propagation rules. We consider every possible colouring of the vertices of N_0. In each branch, we may have to deal with vertices u that still have a list $L(u)$ of size 3. We call such vertices active and prove that they all belong to N_2. We then enter the second phase of our algorithm. In this phase we show, via some further branching, that N_1-neighbours of active vertices either all have a list from $\{\{h, i\}, \{h, j\}\}$, where $\{h, i, j\} = \{1, 2, 3\}$, or they all have the same list $\{h, i\}$. In the third phase, we reduce, again via some branching, to the situation where only the latter option applies: N_1-neighbours of active vertices all have the same list. Then in the fourth and final phase of our algorithm, we know so much structure of the instance that we can reduce to a polynomial number of smaller instances of 2-List-Colouring via a new propagation rule identifying common neighbourhoods of two vertices by a single vertex.

Theorem 3.1 (restated). List 3-Colouring *is polynomial-time solvable for $(P_2 + P_5)$-free graphs and for $(P_3 + P_4)$-free graphs.*

Proof. Let (G, L) be an instance of List 3-Colouring, where $G = (V, E)$ is an H-free graph for $H \in \{P_2 + P_5, P_3 + P_4\}$. Note that G is $(P_3 + P_5)$-free. Since the problem can be solved component-wise, we may assume that G is connected. If G contains a K_4, then G is not 3-colourable, and thus (G, L) is a no-instance. As we can decide if G contains a K_4 in $\mathcal{O}(n^4)$ time by brute force, we assume that from now on G is K_4-free. By brute force, we either deduce in $\mathcal{O}(n^7)$ time that G is P_7-free or we find an induced P_7 on vertices $v_1, \ldots, v_7$ in that order. In the first case, we use Theorem 3.3. It remains to deal with the second case.

Definition 3.5 (Layers). *Let $N_0 = \{v_1, \ldots, v_7\}$. For $i \geq 1$, we define $N_i = \{u \mid \operatorname{dist}(u, N_0) = i\}$. We call the sets N_i $(i \geq 0)$ the layers of G.*

In the remainder, we consider N_0 to be a fixed set of vertices. That is, we will update (G, L) by applying a number of propagation rules and doing some (polynomial) branching, but we will never delete the vertices of N_0. This will enable us to exploit the H-freeness of G.

We show the following two claims about layers.

Claim 3.6. *$V = N_0 \cup N_1 \cup N_2 \cup N_3$.*

Proof of Claim. Suppose $N_i \neq \emptyset$ for some $i \geq 4$. As G is connected, we may assume that $i = 4$. Let $u_4 \in N_4$. By definition, there exists two vertices $u_3 \in N_3$ and $u_2 \in N_2$ such that u_2 is adjacent to u_3 and u_3 is adjacent to u_4. Then G has an induced $P_3 + P_5$ on vertices $u_2, u_3, u_4, v_1, v_2, v_3, v_4, v_5$, a contradiction. $\diamondsuit$

Claim 3.7. *$G[N_2 \cup N_3]$ is the disjoint union of complete graphs of size at most 3, each containing at least one vertex of N_2 (and thus at most two vertices of N_3).*

Proof of Claim. First assume that $G[N_2 \cup N_3]$ has a connected component D that is not a clique. Then D contains an induced P_3, which together with the subgraph $G[\{v_1, \ldots, v_5\}]$ forms an induced $P_3 + P_5$, a contradiction. Then the claim follows after recalling that G is K_4-free and connected. $\diamondsuit$

We will now introduce a number of propagation rules, which run in polynomial time. We are going to apply these rules on G *exhaustively*, that is, until none of the rules can be applied anymore. Note that during this process some vertices of G may be deleted (due to Rules 4 and 10), but as mentioned we will ensure that we keep the vertices of N_0, while we may update the other sets N_i $(i \geq 1)$. We say that a propagation rule is *safe* if the new instance is a yes-instance of List 3-Colouring if and only if the original instance is so.

Rule 1. (no empty lists) If $L(u) = \emptyset$ for some $u \in V$, then return `no`.

Rule 2. (some lists of size 3) If $|L(u)| \leq 2$ for every $u \in V$, then apply Theorem 3.4.

Rule 3. (connected graph) If G is disconnected, then solve List 3-Colouring on each instance (D, L_D), where D is a connected component of G that does not contain N_0 and L_D is the restriction of L to D. If D has no colouring respecting L_D, then return `no`; otherwise remove the vertices of D from G.

Rule 4. **(no coloured vertices)** If $u \notin N_0$, $|L(u)| = 1$ and $L(u) \cap L(v) = \emptyset$ for all $v \in N(u)$, then remove u from G.

Rule 5. **(single colour propagation)** If u and v are adjacent, $|L(u)| = 1$, and $L(u) \subseteq L(v)$, then set $L(v) := L(v) \setminus L(u)$.

Rule 6. **(diamond colour propagation)** If u and v are adjacent and share two common neighbours x and y with $L(x) \neq L(y)$, then set $L(x) := L(x) \cap L(y)$ and $L(y) := L(x) \cap L(y)$.

Rule 7. **(twin colour propagation)** If u and v are non-adjacent, $N(u) \subseteq N(v)$, and $L(v) \subset L(u)$, then set $L(u) := L(v)$.

Rule 8. **(triangle colour propagation)** If u, v, w form a triangle, $|L(u) \cup L(v)| = 2$ and $|L(w)| \geq 2$, then set $L(w) := L(w) \setminus (L(u) \cup L(v))$, so $|L(w)| \leq 1$.

Rule 9. **(no free colours)** If $|L(u) \setminus L(N(u))| \geq 1$ and $|L(u)| \geq 2$ for some $u \in V$, then set $L(u) := \{c\}$ for some $c \in L(u) \setminus L(N(u))$.

Rule 10. **(no small degrees)** If $|L(u)| > |\deg(u)|$ for some $u \in V \setminus N_0$, then remove u from G.

As mentioned, our algorithm will branch at several stages to create a number of new but smaller instances, such that the original instance is a yes-instance if and only if at least one of the new instances is a yes-instance. Unless we explicitly state otherwise, we *implicitly* assume that Rules 1–10 are applied exhaustively immediately after we branch (the reason why we may do this is shown in Claim 3.8). If we apply Rule 1 or 2 on a new instance, then a no-answer means that we will discard the branch. So our algorithm will only return a no-answer for the original instance (G, L) if we discarded all branches. On the other hand, if we can apply Rule 2 on some new instance and obtain a yes-answer, then we can extend the obtained colouring to a colouring of G that respects L, simply by restoring all the already coloured vertices that were removed from the graph due to the rules. We will now state Claim 3.8.

Claim 3.8. *Rules 1–10 are safe and their exhaustive application takes polynomial time. Moreover, if we have not obtained a yes- or no-answer, then afterwards G is a connected (H, K_4)-free graph, such that $V = N_0 \cup N_1 \cup N_2 \cup N_3$ and $2 \leq |L(u)| \leq 3$ for every $u \in V \setminus N_0$.*

Proof of Claim. It is readily seen that Rules 1–5 are safe. For Rule 6, this follows from the fact that any 3-colouring assigns x and y the same colour. For Rule 7, this follows from the fact that u can always be recoloured with the same colour as v. For Rule 8, this follows from the fact that the colours from $L(u) \cup L(v)$ must be used on u and v. For Rule 9, this follows from the fact that no colour from $L(u) \setminus L(N(u))$ will be assigned to a vertex in $N(u)$. For Rule 10, this follows from the fact that we always have a colour available for u.

It is readily seen that applying Rules 1, 2 and 4–10 take polynomial time. Applying Rule 3 takes polynomial time, as each connected component of G that does not contain N_0 is a complete graph on at most three vertices due to the

(H, K_4)-freeness of G (recall that $H = P_2 + P_3$ or $H = P_3 + P_4$). Each application of a rule either results in a no-answer, a yes-answer, reduces the list size of at least one vertex or reduces G by at least one vertex. Thus exhaustive application of the rules takes polynomial time.

Suppose exhaustive application does not yield a no-answer or a yes-answer. By Rule 3, G is connected. As no vertex of N_0 was removed, G contains N_0. Hence, we can define $V = N_0 \cup N_1 \cup N_2 \cup N_3$ by Claim 3.6. By Rules 4 and 5, we find that $2 \leq |L(u)| \leq 3$ for every $u \in V \setminus N_0$. It is readily seen that Rules 1–10 preserve (H, K_4)-freeness of G. ♢

3.2.1 Phase 1. Preprocessing $(\mathbf{G}, \mathbf{L})$

In Phase 1 we will preprocess (G, L) using the above propagation rules. To start off the preprocessing we will branch via colouring the vertices of N_0 in every possible way. To start off the preprocessing we will branch via colouring the vertices of N_0 in every possible way. By colouring a vertex u, we mean reducing the list of permissible colours to size exactly one. (When $L(u) = \{c\}$, we consider vertex coloured by colour c.) Thus, when we colour some vertex u, we always give u a colour from its list $L(u)$, moreover, when we colour more than one vertex we will always assign distinct colours to adjacent vertices.

Branching I ($\mathcal{O}(1)$ branches). We now consider all possible combinations of colours that can be assigned to the vertices in N_0. That is, we branch into at most 3^7 cases, in which $v_1, \ldots, v_7$ each received a colour from their list. We note that each branch leads to a smaller instance and that (G, L) is a yes-instance if and only if at least one of the new instances is a yes-instance. Hence, if we applied Rule 1 in some branch, then we discard the branch. If we applied Rule 2 and obtained a no-answer, then we discard the branch as well. If we obtained a yes-answer, then we are done. Otherwise we continue by considering each remaining branch separately. For each remaining branch, we denote the resulting smaller instance by (G, L) again.

We will now introduce a new rule, namely Rule 11. We apply Rule 11 together with the other rules. That is, we now apply Rules 1–11 exhaustively. However, each time we apply Rule 11 we first ensure that Rules 1–10 have been applied exhaustively.

Rule 11 **($\mathbf{N_3}$-reduction)** If u and v are in N_3 and are adjacent, then remove u and v from G.

Claim 3.9. *Rule 11, applied after exhaustive application of Rules 1–10, is safe and takes polynomial time. Moreover, afterwards G is a connected (H, K_4)-free graph, such that $V = N_0 \cup N_1 \cup N_2 \cup N_3$ and $2 \leq |L(u)| \leq 3$ for every $u \in V \setminus N_0$.*

Proof of Claim. Assume that we applied Rules 1–10 exhaustively and that N_3 contains two adjacent vertices u and v. By Claim 3.7, we find that u and v have a common neighbour $w \in N_2$ and no other neighbours. By Rules 4, 5 and 10, we then find that $|L(u)| = |L(v)| = 2$. First suppose that $L(u) = L(v)$, say $L(u) = L(v) = \{1, 2\}$. Then, by Rule 8, we find that $L(w) = \{3\}$, contradicting

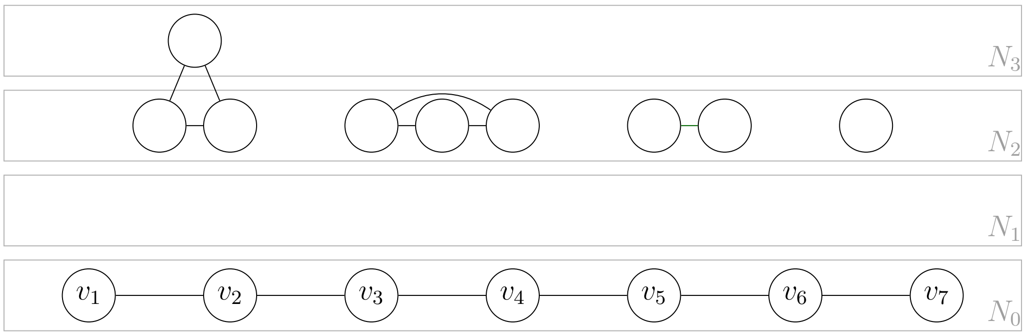

Figure 3.1: All possible connected components in $G[N_2 \cup N_3]$.

Rule 4. Hence $L(u) \neq L(v)$, say $L(u) = \{1, 2\}$ and $L(v) = \{1, 3\}$. By Rule 8, we find that $L(w) = \{2, 3\}$ or $L(w) = \{1, 2, 3\}$. If w gets colour 1, we can give u colour 2 and v colour 3. If w gets colour 2, we can give u colour 1 and v colour 3. Finally, if w gets colour 3, then we can give u colour 2 and v colour 1. Hence we may set $V := V \setminus \{u_1, u_2\}$. This does not destroy the connectivity or (H, K_4)-freeness of G. ♢

We now show the following claim.

Claim 3.10. *The set N_3 is independent, and moreover, each vertex $u \in N_3$ has $|L(u)| = 2$ and exactly two neighbours in N_2 which are adjacent.*

Proof. By Rule 11, we find that N_3 is independent. As every vertex of N_3 has at most two neighbours in N_2 due to Claim 3.7, the claim follows from Rules 4, 5, 10 and the fact that N_3 is independent. The two neighbours of a vertex in N_3 must be mutually adjacent, otherwise, they form an induced P_3 that is not connected with N_0. □

The following claim follows immediately from Claims 3.7 and 3.10 and gives a complete description of the second and third layer, see also Figure 3.1.

Claim 3.11. *Every connected component D of Graph$[N_2 \cup N_3]$ is a complete graph with either $|D| \leq 2$ and $D \subseteq N_2$, or $|D| = 3$ and $|D \cap N_3| \leq 1$.*

The following claim describes the location of the vertices with list of size 3 in G.

Claim 3.12. *For every $u \in V$, if $|L(u)| = 3$, then $u \in N_2$.*

Proof of Claim. As the vertices in N_0 have lists of size 1, the vertices in N_1 have lists of size 2. By Claim 3.10, the same holds for vertices in N_3. ♢

In the remainder of the proof we will show how to branch in order to reduce the lists of the vertices $u \in N_2$ with $|L(u)| = 3$ by at least one colour. We formalize this approach in the following definition.

Definition 3.13 (Active vertices)**.** *A vertex $u \in N_2$ and its neighbours in N_1 are called* active *if $|L(u)| = 3$. Let A be the set of all active vertices. Let $A_1 = A \cap N_1$ and $A_2 = A \cap N_2$. We* deactivate *a vertex $u \in A_2$ if we reduce the list $L(u)$ by at* *least one colour. We* deactivate *a vertex $w \in A_1$ by deactivating all its neighbours in A_2.*

Note that every vertex $w \in A_1$ has $|L(w)| = 2$ by Rule 5 applied on the vertices of N_0. Hence, if we reduce $L(w)$ by one colour, all neighbours of w in A_2 become deactivated by Rule 5, and w is removed by Rule 4.

For $1 \leq i < j \leq 7$, we let $A(i,j) \subseteq A_1$ be the set of active neighbours of v_i that are not adjacent to v_j and similarly, we let $A(j,i) \subseteq A_1$ be the set of active neighbours of v_j that are not adjacent to v_i.

3.2.2 Phase 2. Reduce the Number of Distinct Sets $\mathbf{A(i,j)}$

We will now branch into $\mathcal{O}((n^{45})$ smaller instances such that (G,L) is a yes-instance of List 3-Colouring if and only if at least one of these new instances is a yes-instance. Each new instance will have the following property:

(P) for $1 \leq i \leq j \leq 7$ with $j - i \geq 2$, either $A(i,j) = \emptyset$ or $A(j,i) = \emptyset$.

Branching II ($\mathcal{O}\Big(n^{\left(3\cdot\left(\binom{7}{2}-6\right)\right)}\Big) = \mathcal{O}(n^{45})$ branches. Consider two vertices v_i and v_j with $1 \leq i \leq j \leq 7$ and $j - i \geq 2$. Assume without loss of generality that v_i is coloured 3 and that v_j is coloured either 1 or 3. Hence, every $w \in A(i,j)$ has $L(w) = \{1,2\}$, whereas every $w \in A(j,i)$ has $L(w) = \{2,q\}$ for $q \in \{1,3\}$. We branch as follows. We consider all possibilities where at most one vertex of $A(i,j)$ receives colour 2 (and all other vertices of $A(i,j)$ receive colour 1) and all possibilities where we choose two vertices from $A(i,j)$ to receive colour 2. This leads to $\mathcal{O}(n) + \mathcal{O}(n^2) = \mathcal{O}(n^2)$ branches. In the branches where at most one vertex of $A(i,j)$ receives colour 2, every vertex of $A(i,j)$ will be deactivated. So Property **(P)** is satisfied for i and j.

Now consider the branches where two vertices x_1, x_2 of $A(i,j)$ both received colour 2. We update $A(j,i)$ accordingly. In particular, afterwards no vertex in $A(j,i)$ is adjacent to x_1 or x_2, as 2 is a colour in the list of each vertex of $A(j,i)$. We now do some further branching for those branches where $A(j,i) \neq \emptyset$. We consider the possibility where each vertex of $N(A(j,i)) \cap A_2$ is given the colour of v_j and all possibilities where we choose one vertex in $N(A(j,i)) \cap A_2$ to receive a colour different from the colour of v_j (we consider both options to colour such a vertex). This leads to $\mathcal{O}(n)$ branches. In the first branch, every vertex of $A(j,i)$ will be deactivated. So Property **(P)** is satisfied for i and j.

Now consider a branch where a vertex $u \in N(A(j,i)) \cap A_2$ receives a colour different from the colour of v_j. We will show that also, in this case, every vertex of $A(j,i)$ will be deactivated. For contradiction, assume that $A(j,i)$ contains a vertex w that is not deactivated after colouring u. As u was in $N(A(j,i)) \cap A_2$, we find that u had a neighbour $w' \in A(j,i)$. As u is coloured with a colour different from the colour of v_j, the size of $L(w')$ is reduced by one (due to Rule 4). Hence w' got deactivated after colouring u, and thus $w' \neq w$. As w is still active, w has a neighbour $u' \in A_2$. As u' and w are still active, u' and w are not adjacent to w' or u. Hence, u, w', v_j, w, u' induce a P_5 in G. As x_1 and x_2 both received colour 2, we find that x_1 and x_2 are not adjacent to each other. Hence, x_1, v_i, x_2 induce a P_3 in G. Recall that all vertices of $A(j,i)$, so also w and w', are not

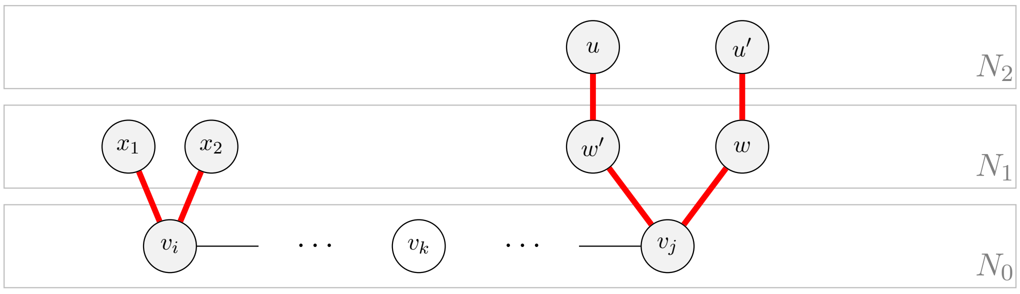

Figure 3.2: The situation in Branching II.

adjacent to x_1 or x_2. As u and u' were still active after colouring x_1 and x_2, we find that u and u' are not adjacent to x_1 or x_2 either. By definition of $A(j, i)$, w and w' are not adjacent to v_i. By definition of $A(i, j)$, x_1 and x_2 are not adjacent to v_j. Moreover, v_i and v_j are non-adjacent, as $j - i \geq 2$. We conclude that G contains an induced $P_3 + P_5$, namely with vertex set $\{x_1, v_i, x_2\} \cup \{u, w', v_j, w, u'\}$, a contradiction (see Figure 3.2 for an example of such a situation). Hence, every vertex of $A(j, i)$ is deactivated. So Property **(P)** is satisfied for i and j also for these branches.

Finally by recursive application of the above described procedure for all pairs v_i, v_j such that $1 \leq i \leq j \leq 7$ and $j-i \geq 2$ we get a graph satisfying Property **(P)**, which together leads to $O\left(n^{\left(3\cdot\left(\binom{7}{2}-6\right)\right)}\right) = \mathcal{O}(n^{45})$ branches.

We now consider each resulting instance from Branching II. We denote such an instance by (G, L) again. Note that vertices from N_2 may now belong to N_3, as their neighbours in N_1 may have been removed due to the branching. The exhaustive application of Rules 1– 11 preserves **(P)** (where we apply Rule 11 only after applying Rules 1–10 exhaustively). Hence (G, L) satisfies **(P)**.

We observe that if two vertices in A_1 have a different list, then they must be adjacent to different vertices of N_0. Hence, by Property **(P)**, at most two lists of $\{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$ can occur as lists of vertices of A_1. Without loss of generality this leads to two cases: either every vertex of A_1 has list $\{1, 2\}$ or $\{1, 3\}$ and both lists occur on A_1; or every vertex of A_1 has list $\{1, 2\}$ only. In the next phase of our algorithm we reduce, via some further branching, every instance of the first case to a polynomial number of smaller instances of the second case.

3.2.3 Phase 3. Reduce to the Case Where Vertices of $\mathbf{A_1}$ Have the Same List

Recall that we assume that every vertex of A_1 has list $\{1, 2\}$ or $\{1, 3\}$. In this phase we deal with the case when both types of lists occur in A_1. We first prove the following claim.

Claim 3.14. *Let $i \in \{1, 3, 5, 7\}$. Then every vertex from $A_1 \cap N(v_i)$ is adjacent to some vertex v_j with $j \notin \{i - 1, i, i + 1\}$.*

Proof of Claim. We may assume without loss of generality that $i = 1$ or $i = 3$. For contradiction suppose there exists a vertex $w \in A_1 \cap N(v_i)$ that is non-adjacent to all v_j with $j \notin \{i - 1, i, i + 1\}$. As two consecutive vertices in N_0 have different colours, no vertex in A_1 has two consecutive neighbours in N_0 due to Rules 4 and 5. Hence $N(w) \cap N_0 = \{v_i\}$. By definition, w has a neighbour $u \in A_2$. If $i = 1$, then $\{u, w, v_1, v_2, v_3\} \cup \{v_5, v_6, v_7\}$ induces a $P_3 + P_5$ in G. If $i = 3$, then $\{v_1, v_2, v_3, w, u\} \cup \{v_5, v_6, v_7\}$ induces a $P_3 + P_5$ in G. $\diamondsuit$

Claim 3.15. *It holds that $N(A_1) \cap N_0 = \{v_{i-1}, v_i, v_{i+1}\}$ for some $2 \leq i \leq 6$. Moreover, we may assume without loss of generality that v_{i-1} and v_{i+1} have colour 3 and both are adjacent to all vertices of A_1 with list $\{1, 2\}$, whereas v_i has colour 2 and is adjacent to all vertices of A_1 with list $\{1, 3\}$.*

Proof of Claim. Recall that lists $\{1, 2\}$ and $\{1, 3\}$ both occur on A_1. By Property **(P)**, this means that either $N(A_1) \cap N_0 = \{v_{i-1}, v_i\}$ for some $2 \leq i \leq 7$ or $N(A_1) \cap N_0 = \{v_{i-1}, v_i, v_{i+1}\}$ for some $2 \leq i \leq 6$. The case where $N(A_1) \cap N_0 = \{v_{i-1}, v_i\}$ for some $2 \leq i \leq 7$ is not possible due to Claim 3.14. It follows that $N(A_1) \cap N_0 = \{v_{i-1}, v_i, v_{i+1}\}$ for some $2 \leq i \leq 6$. We may assume without loss of generality that v_i has colour 2, meaning that v_{i-1} and v_{i+1} must have colour 3. It follows that every vertex of A_1 with list $\{1, 3\}$ is adjacent to v_i but not to v_{i-1} or v_{i+1}, whereas every vertex of A_1 with list $\{1, 2\}$ is adjacent to at least one vertex of $\{v_{i-1}, v_{i+1}\}$ but not to v_i. As a vertex of A_1 with list $\{1, 3\}$ has v_i as its only neighbour in N_0, it follows from Claim 3.14 that i is an even number. This means that $i - 1$ is odd. Hence, every vertex of A_1 with list $\{1, 2\}$ is in fact adjacent to both v_{i-1} and v_{i+1} due to Claim 3.14. $\diamondsuit$

By Claim 3.15, we can partition the set A_1 into two (non-empty) sets $X_{1,2}$ and $X_{1,3}$, where $X_{1,2}$ is the set of vertices in A_1 with list $\{1, 2\}$ whose only neighbours in N_0 are v_{i-1} and v_{i+1} (which both have colour 3) and $X_{1,3}$ is the set of vertices in A_1 with list $\{1, 3\}$ whose only neighbour in N_0 is v_i (which has colour 2).

Our goal is to show that we can branch into at most $\mathcal{O}(n^2)$ smaller instances, in which either $X_{1,2} = \emptyset$ or $X_{1,3} = \emptyset$, such that (G, L) is a yes-instance of List 3-Colouring if and only if at least one of these smaller instances is a yes-instance. Then afterwards it suffices to show how to deal with the case where all vertices in A_1 have the same list in polynomial time; this will be done in Phase 4 of the algorithm. We start with the following $\mathcal{O}(n)$ branching procedure (in each of the branches we may do some further $\mathcal{O}(n)$ branching later on).

Branching III ($\mathcal{O}(n)$ branches). We branch by considering the possibility of giving each vertex in $X_{1,2}$ colour 2 and all possibilities of choosing a vertex in $X_{1,2}$ and giving it colour 1. This leads to $\mathcal{O}(n)$ branches. In the first branch we obtain $X_{1,2} = \emptyset$. Hence we can start Phase 4 for this branch. We now consider every branch in which $X_{1,2}$ and $X_{1,3}$ are both nonempty. For each such branch we will create $\mathcal{O}(n)$ smaller instances of List 3-Colouring, where $X_{1,3} = \emptyset$, such that (G, L) is a yes-instance of List 3-Colouring if and only if at least one of the new instances is a yes-instance.

Let $w \in X_{1,2}$ be the vertex that was given colour 1 in such a branch. Although by Rule 4 vertex w will need to be removed from G, we make an exception by temporarily keeping w after we coloured it. The reason is that the presence of w will be helpful for analysing the structure of (G, L) after Rules 1–11 have been applied exhaustively (where we apply Rule 11 only after applying Rules 1–10 exhaustively). In order to do this, we first show the following three claims.

Claim 3.16. *Vertex w is not adjacent to any vertex in $A_2 \cup X_{1,2} \cup X_{1,3}$.*

Proof of Claim. By giving w colour 1, the list of every neighbour of w in A_2 has been reduced by one due to Rule 5. Hence, all neighbours of w in A_2 are deactivated. For the same reason all neighbours of w in $X_{1,2}$, which have list $\{1, 2\}$, are coloured 2, and all neighbours of w in $X_{1,3}$, which have list $\{1, 3\}$, are coloured 3. These vertices were removed from the graph by Rule 4. This proves the claim. $\diamondsuit$

Claim 3.17. *The graph $G[X_{1,3} \cup (N(X_{1,3}) \cap A_2) \cup N_3]$ is the disjoint union of one or more complete graphs, each of which consists of either one vertex of $X_{1,3}$ and at most two vertices of A_2, or one vertex of N_3.*

Proof of Claim. We write $G^* = G[X_{1,3} \cup (N(X_{1,3}) \cap A_2) \cup N_3]$ and first show that G^* is the disjoint union of one or more complete graphs. For contradiction, assume that G^* is not such a graph. Then G^* contains an induced P_3, say on vertices u_1, u_2, u_3 in that order. As $w \in X_{1,2} \subseteq N_1$, we find that w is not adjacent to any vertex of N_3. By Claim 3.16, we find that w is not adjacent to any vertex of $A_2 \cup X_{1,3}$. Recall that v_{i-1} and v_{i+1} are the only neighbours of w in N_0, whereas v_i is the only neighbour of the vertices of $X_{1,3}$ in N_0. Hence, $\{u_1, u_2, u_3\} \cup \{v_1, \ldots, v_{i-1}, w, v_{i+1}, \ldots, v_7\}$ induces a $P_3 + P_7$. This contradicts the $(P_3 + P_5)$-freeness of G. We conclude that G^* is the disjoint union of one or more complete graphs.

As G is K_4-free, the above means that every connected component of G^* is a complete graph on at most three vertices. No vertex of N_3 is adjacent to a vertex in $X_{1,3} \subseteq N_1$. Moreover, by definition, every vertex of $N(X_{1,3}) \cap A_2$ is adjacent to at least one vertex of $X_{1,3}$. As every connected component of G^* is a complete graph, this means that no vertex of N_3 is adjacent to a vertex of $N(X_{1,3}) \cap A_2$ either. We conclude that the vertices of N_3 are isolated vertices of G^*.

Let D be a connected component of G^* that does not contain a vertex of N_3. From the above we find that D is a complete graph on at most three vertices. By definition, every vertex in $X_{1,3}$ has a neighbour in A_2 and every vertex of $N(X_{1,3}) \cap A_2$ has a neighbour in $X_{1,3}$. This means that D either consists of one vertex in $X_{1,3}$ and at most two vertices of A_2, or D consists of two vertices of $X_{1,3}$ and one vertex of A_2. We claim that the latter case is not possible. For contradiction, assume that D is a triangle that consists of three vertices s, u_1, u_2, where $s \in A_2$ and $u_1, u_2 \in X_{1,3}$. However, as $L(u_1) = L(u_2) = \{1, 3\}$, we find that $|L(s)| = 1$ by Rule 8, contradicting the fact that s belongs to A_2. This completes the proof of the claim. $\diamondsuit$

Claim 3.18. *For every pair of adjacent vertices s, t with $s \in A_2$ and $t \in N_2$, either t is adjacent to w, or $N(s) \cap X_{1,3} \subseteq N(t)$.*

Proof of Claim. For contradiction, assume that t is not adjacent to w and that there is a vertex $r \in X_{1,3}$ that is adjacent to s but not to t. By Claim 3.16, we find that w is not adjacent to r or s. Just as in the proof of Claim 3.17, we find that $\{r, s, t\}$ together with $\{v_1, , \ldots, v_{i-1}, w, v_{i+1}, \ldots, v_7\}$ induces a $P_3 + P_7$ in G, a contradiction. $\diamondsuit$

We now continue as follows. Recall that $X_{1,3} \neq \emptyset$. Hence there exists a vertex $s \in A_2$ that has a neighbour $r \in X_{1,3}$. As $s \in A_2$, we have that $|L(s)| = 3$. Then, by Rule 10, we find that s has at least two neighbours t and t' not equal to r. By Claim 3.17, we find that neither t nor t' belongs to $X_{1,3} \cup N_3$. We are going to fix an induced 3-vertex path P^s of G, over which we will branch, in the following way.

If t and t' are not adjacent, then we let P^s be the induced path in G with vertices t, s, t' in that order. Suppose that t and t' are adjacent. As G is K_4-free and s is adjacent to r, t, t', at least one of t, t' is not adjacent to r. We may assume without loss of generality that t is not adjacent to r.

First assume that $t \in N_2$. Recall that s has a neighbour in $X_{1,3}$, namely r, and that r is not adjacent to t. We then find that t must be adjacent to w by Claim 3.18. As $s \in A_2$, we find that s is not adjacent to w by Claim 3.16. In this case we let P^s be the induced path in G with vertices s, t, w in that order.

Now assume that $t \notin N_2$. Recall that $t \notin N_3$. Hence, t must be in N_1. Then, as $t \notin X_{1,3}$ but t is adjacent to a vertex in A_2, namely s, we find that $t \in X_{1,2}$. Recall that $t' \notin X_{1,3}$. If $t' \in N_1$ then the fact that $t' \notin X_{1,3}$, combined with the fact that t' is adjacent to $s \in A_2$, implies that $t' \in X_{1,2}$. However, by Rule 8 applied on s, t, t', vertex s would have a list of size 1 instead of size 3, a contradiction. Hence, $t' \notin N_1$. As $t' \notin N_3$, this means that $t' \in N_2$. If t' is adjacent to r, then $t \in X_{1,2}$ with $L(t) = \{1, 2\}$ and $r \in X_{1,3}$ with $L(r) = \{1, 3\}$ would have the same lists by Rule 6 applied on r, s, t, t', a contradiction. Hence t' is not adjacent to r. Then, by Claim 3.18, we find that t' must be adjacent to w. Note that s is not adjacent to w due to Claim 3.16. In this case we let P^s be the induced path in G with vertices s, t', w in that order.

We conclude that either $P^s = tst'$ or $P^s = stw$ or $P^s = st'w$. We are now ready to apply another round of branching.

Branching IV ($\mathcal{O}(n)$ branches). We branch by considering the possibility of removing colour 2 from the list of each vertex in $N(X_{1,3}) \cap A_2$ and all possibilities of choosing a vertex in $N(X_{1,3}) \cap A_2$ and giving it colour 2. In the branch where we removed colour 2 from the list of every vertex in $N(X_{1,3}) \cap A_2$, we obtain that $X_{1,3} = \emptyset$. Hence for that branch we can enter Phase 4. Now consider a branch where we gave some vertex $s \in N(X_{1,3}) \cap A_2$ colour 2. Let $P^s = tst'$ or $P^s = stw$ or $P^s = st'w$. We do some further branching by considering all possibilities of colouring the vertices of P^s that are not equal to the already coloured vertices s and w (should w be a vertex of P^s) and all possibilities of giving a colour to the vertex from $N(s) \cap X_{1,3}$ (recall that by Claim 3.17, $|N(s) \cap X_{1,3}| = 1$). This leads to a total of $\mathcal{O}(n)$ branches. We claim that in each of these branches, the size of $X_{1,3}$ has reduced to at most 1.

For contradiction, assume that there exists a branch where $X_{1,3}$ contains two vertices y and y'. Let s_a and s_b be the neighbours of y and y' in A_2, respectively. By Claim 3.17, the graph induced by $\{y, y', s_a, s_b\}$ is isomorphic to $2P_2$. Hence, the set $\{s_a, y, v_i, y', s_b\}$ induces a P_5 in G. Recall that $P^s = tst'$ or $P^s = stw$ or $P^s = st'w$. As s_a and s_b have a list of size 3, neither s_a nor s_b is adjacent to a vertex of P^s due to rule 5. By Claims 3.16 and 3.17, neither y nor y' is adjacent to w or s, respectively. By colouring $N(s) \cap X_{1,3}$ neither y nor y' is adjacent to s, too. As s received colour 2, vertices t and t' have received colour 1 or 3 should

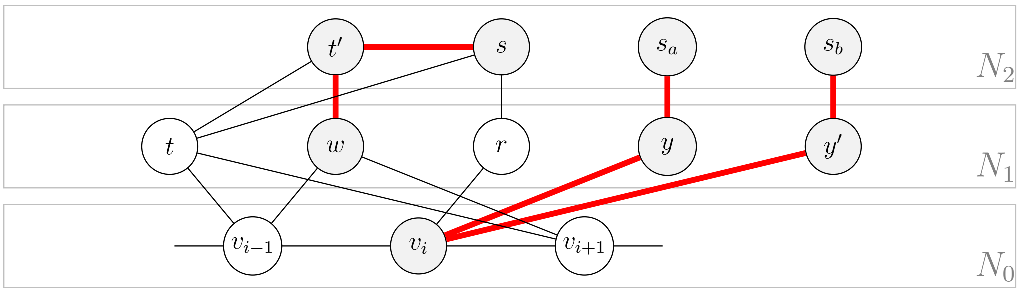

Figure 3.3: The situation in Branching IV if $t_1 \in N_1$ and if vertices s_a and s_b exist.

they belong to P^s. In that case neither t nor t' can be adjacent to y or y', as $L(y) = L(y') = \{1, 3\}$. By definition, v_i is not adjacent to s or w. Moreover, v_i can only be adjacent to a vertex from $\{t_j, t'_j\}$ if that vertex belonged to N_1. However, recall that t and t' were not in $X_{1,3}$ while s was an active vertex. Hence if t or t' belonged to N_1, they must have been in $X_{1,2}$ and thus not adjacent to v_i. This means that the vertices of P^s, together with $\{s_a, y, v_i, y', s_b\}$, induce a $P_3 + P_5$ in G, a contradiction (see Figure 3.3 for an example of such a situation). Thus $X_{1,3}$ must contain at most one vertex.

Branching V ($\mathcal{O}(1)$ branches). We branch by considering both possibilities of colouring the unique vertex of $X_{1,3}$. This leads to two new but smaller instances of List 3-Colouring, in each of which the set $X_{1,3} = \emptyset$. Hence, our algorithm can enter Phase 4.

3.2.4 Phase 4. Reduce to a Set of Instances of 2-List Colouring

Recall that in this stage of our algorithm we have an instance (G, L) in which every vertex of A_1 has the same list, say $\{1, 2\}$. We deal with this case as follows. First suppose that $H = P_2 + P_5$. Then $G[N_2 \cup N_3]$ is an independent set, as otherwise two adjacent vertices of $N_2 \cup N_3$ form, together with $v_1, \ldots, v_5$, an induced $P_2 + P_5$. Hence, we can safely colour each vertex in A_2 with colour 3, and afterwards we may apply Theorem 3.4.

Now suppose that $H = P_3 + P_4$. We first introduce two new rules, which turn (G, L) into a smaller instance. In Claims 3.19 and 3.21 we show that we may include those rules in our set of propagation rules that we apply implicitly every time we modify the instance (G, L).

Rule 12 (neighbourhood identification) If u and v are adjacent, $N(v) \subseteq N[u]$, and $|L(v)| = 3$, then identify $N(u) \cap N(v)$ by w, set $L(w) := \bigcap\{L(x) \mid x \in N(u) \cap N(v)\}$ and remove v from G. If G contains a K_4, then return `no`.

Claim 3.19. *Rule 12 is safe for K_4-free input, takes polynomial time and does not affect any vertex of N_0. Moreover, if we have not obtained a no-answer, then afterwards G is a connected (H, K_4)-free graph, in which we can define sets N_1, N_2, N_3, A_1, A_2 as before.*

Proof of Claim. Note that by Claim 3.8, G is K_4-free before the application of Rule 12. Hence $N(u) \cap N(v)$ is an independent set. Let w be the new vertex obtained from identifying $N(u) \cap N(v)$. Observe that every vertex in the common neighbourhood of two adjacent vertices must receive the same colour. Hence w can be given the same colour as any vertex of $N(u) \cap N(v)$, which belongs to $\bigcap\{L(x) \mid x \in N(u) \cap N(v)\}$. For the reverse direction, we give each vertex $x \in N(u) \cap N(v)$ the colour of w, which belongs to $L(x)$ by definition. As $|L(v)| = 3$ and $N(v) \setminus N(u) = \{u\}$, we have a colour available for v. The above means that (G, L) is a no-instance if a K_4 is created. We conclude that Rule 12 is safe and either yields a no-instance if a K_4 was created, or afterwards we have again that G is K_4-free.

It is readily seen that applying Rule 12 takes polynomial time and that afterwards G is still connected. As $|L(v)| = 3$, Claim 3.12 tells us that $v \in N_2$, and thus $N(v) \subseteq N_1 \cup N_2 \cup N_3$. Thus Rule 12 does not involve any vertex of N_0. Hence, as G is connected, we can define $V = N_0 \cup N_1 \cup N_2 \cup N_3$ by Claim 3.6.

It remains to prove that G is H-free after applying Rule 12. For contradiction, assume that G has an induced subgraph $P + P'$ isomorphic to H. Then we find that $w \in V(P) \cup V(P')$, say $w \in V(P)$, as otherwise $P + P'$ was already an induced subgraph of G before Rule 12 was applied. By the same argument, we find that w is incident with two edges wx and wy in P that correspond to edges sx and ty with $s \neq t$ in G before Rule 12 was applied. However, then we can replace P by the path $xsuty$ to find again that G already contained an induced copy of H before Rule 12 was applied, a contradiction. $\diamondsuit$

Let $u \in A_2$. We let $B(u)$ be the set of neighbours of u that have colour 3 in their list. By Rule 9, there is a vertex $v \in N(u)$ such that $3 \in L(v)$. Vertex v cannot be in N_1; otherwise the edge uv implies that $v \in A_1$ and thus v would have list $\{1, 2\}$. This means that v must be in $N_2 \cup N_3$. Hence we have proven the following claim.

Claim 3.20. *For every $u \in A_2$, it holds that $B(u) \neq \emptyset$ and $B(u) \subseteq N_2 \cup N_3$.*

We will use the following rule (in Claim 3.21 we show that the colour q is unique).

Rule 13 ($\mathbf{A_2}$ list-reduction) If a vertex $v \in B(u)$ for some $u \in A_2$ has no neighbour outside $N[u]$, then remove colour q from $L(u)$ for $q \in L(v) \setminus \{3\}$.

Claim 3.21. *Rule 13 is safe, takes polynomial time and does not affect any vertex of N_0. Moreover, afterwards G is a connected (H, K_4)-free graph, in which we can define sets N_1, N_2, N_3, A_1, A_2 as before.*

Proof of Claim. Let u be a vertex in A_2 for which there exists a vertex $v \in B(u)$ with no neighbour outside $N[u]$. It is readily seen that Rule 13 applied on u takes polynomial time, does not affect any vertex of N_0, and afterwards we can define sets N_1, N_2, N_3, A_1, A_2 as before.

We recall from above that $v \in N_2 \cup N_3$. As $N(v) \setminus N[u] = \emptyset$, we find by Rule 12 that $|L(v)| \neq 3$. Then, by Rule 4, it holds that $|L(v)| = 2$. Thus vertex v has $L(v) = \{q, 3\}$ for some $q \in \{1, 2\}$. If there exists a colouring c of G with $c(u) = q$ that respects L, then $c(v) = 3$, and so c colours each vertex in $N(v) \cap N(u)$ with a colour from $\{1, 2\}$.

We define a colouring c' by setting $c'(u) = 3$, $c'(v) = q$ and $c' = c$ for $V(G) \setminus \{u, v\}$. We claim that c' also respects L. As $N(v) \setminus N[u] = \emptyset$, every neighbour $w \neq u$ of v is a neighbour of u as well and thus received a colour $c'(w) = c(w)$ that is not equal to colour q (and colour 3). As $v \in N_2 \cup N_3$ by Claim 3.20, all vertices in $N(u) \setminus N[v]$ are in N_1 by Claim 3.7. As $u \in A_2$, these vertices all belong to A_1 and thus their lists are equal to $\{1, 2\}$, so do not contain colour 3. Hence, c' respects L indeed.

The above means that we can avoid assigning colour q to u. We may therefore remove q from $L(u)$. This completes the proof of the claim. $\diamondsuit$

We note that if a colour q is removed from the list of some vertex $u \in A_2$ due to Rule 13, then u is no longer active.

Assume that Rules 1–13 have been applied exhaustively. By Rule 2, we find that $A_2 \neq \emptyset$. Then we continue as follows. Let $u \in A_2$ and $v \in B(u)$ (recall that $B(u)$ is nonempty due to Claim 3.20). Let $A(u, v) \subseteq N_1$ be the set of (active) neighbours of u that are not adjacent to v. Note that $A(u, v) \subseteq A_1$ by definition. Let $A(v, u) \subseteq N_1$ be the set of neighbours of v that are not adjacent to u. We claim that both $A(u, v)$ and $A(v, u)$ are nonempty. By Rule 13, we find that $A(v, u) \neq \emptyset$. By Rule 12, vertex u has a neighbour $t \notin N(v)$. As $v \in N_2 \cup N_3$ due to Claim 3.20, we find by Claim 3.7 that t belongs to N_1, thus $t \in A(u, v)$, and consequently, $A(u, v) \neq \emptyset$. We have the following three disjoint situations:

1. $A(v, u)$ contains a vertex w with $L(w) = \{1, 2\}$ that is not adjacent to some vertex $t \in A(u, v)$;

2. $A(v, u)$ contain at least one vertex w that is not adjacent to some vertex $t \in A(u, v)$, but for all such vertices w it holds that $L(w) \neq \{1, 2\}$.

3. Every vertex in $A(v, u)$ is adjacent to every vertex of $A(u, v)$.

Now we construct a triple $(Q, P, x) = (Q(u), P(u), x(u))$ such that Q is a set which contains u, $P \subseteq Q$ is an induced P_4 and x is a vertex of Q. In situation 1, we let $Q = \{w, t, u, v\}$. We say that Q is of type 1. We let $x = u$. As P we can take the path on vertices t, u, v, w in that order. In situation 2, we let $Q = \{w, t, u, v\}$ for some $w \in A(v, u)$ that is not adjacent to some $t \in A(u, v)$. We say that Q is of type 2. We let $x = v$. As P we can take the path on vertices t, u, v, w in that order.

Finally we consider situation 3. By Rule 7 applied on u and a vertex of $A(v, u)$ we find that $N(u) \cap N(v)$ contains at least one vertex s. We let $Q = \{s, t, w, u, v\}$ for some $w \in A(v, u)$ and $t \in A(u, v)$. We let $x = v$. We claim that the vertices s, u, t, w induce a P_4 in that order. By definition, u is not adjacent to w. If $sw \in E(G)$, then $L(u) = L(w)$ due to Rule 6. As w has a list of size 2, u has also a list of size 2. If $st \in E(G)$, then $L(v) = L(t)$ due to Rule 6. In that case it even holds that $|L(v)| = |L(t)| = 1$, which means that u has a list of size 2 due to Rule 5. In both cases u is not an active vertex, a contradiction. Hence, as P we can take the path on vertices s, u, t, w in that order.

We slightly try to extend Q as follows. If $A(u, v)$ contains more vertices than only vertex t, we pick an arbitrary vertex t' of $A(u, v) \setminus \{t\}$ and put t' to Q. We first observe that if $c(x) = 3$ no other vertex of Q can be coloured with colour 3; in particular recall that t and t' (if t' exists) both belong to A_1, and as such have list $\{1,2\}$. Moreover, if Q is of type 2, then any vertex in $A(v,u)$ with list $\{1,2\}$ is adjacent to t and t', as otherwise Q is of type 1.

Branching VI ($\mathcal{O}(n)$ branches). We choose a vertex $u \in A_2$ such that $|N(u) \cap N_1|$ is minimal and create (Q,P,x). We branch by considering all possibilities of colouring Q such that $c(x)=3$ and the possibility where we remove colour 3 from $L(x)$. The first case leads to $\mathcal{O}(1)$ branches, since $|Q| \leq 6$. We will prove that we either terminate by Rule 2 or branch in Branching VII. In the second case we deactivate u directly or by applying Rule 13 and Rule 5. This is the only recursive branch and the depth of the recursion is $|A_2| \in \mathcal{O}(n)$.

Now consider a branch where Q is coloured. Although by Rule 4 vertices in Q will need to be removed from G, we make an exception by temporarily keeping Q in the graph after we coloured it until the end of Branching VII. The reason is that this will be helpful for analysing the structure of (G,L). We run only Rules 2, 5 and 8 to prevent changes in the size of neighbourhood of vertices in A_2 for the purposes of the next claim (Claim 3.22). Observe that Rules 2, 5 and 8 do not decrease the degree of any vertex. By Rule 2, $A_2 \neq \emptyset$. We prove the following claim for vertices in A_2.

Claim 3.22. *There is no vertex in A_2 with more than one neighbour in A_1.*

Proof of Claim. For contradiction, assume that r is a vertex in A_2 with two neighbours s and s' in A_1. By Rule 8, s and s' are not adjacent. Hence the set $\{s,r,s'\}$ induces a P_3, which we denote by P'. As every vertex in A_1 has list $\{1,2\}$, the only possible edges between Q and P' are those between $\{s,s'\}$ and vertex x, the only vertex in Q which has colour 3.

First suppose Q is of type 1. Recall that $x=u$. If $|N(u) \cap A_1| = 1$, then u cannot have any other vertex in $A_1 \setminus Q$ as a neighbour. If $|N(u) \cap A_1| \geq 2$, then there has to be an edge to all but one vertex in $N(r) \cap A_1$. This together with at least two coloured vertices in $Q \cap N(u) \cap N_1$ gives a contradiction with the minimality of $|N(u) \cap N_1|$.

Now suppose that Q is of type 2. Recall that $x=v$. Recall also that if v is adjacent to a vertex in $A_1 \setminus Q$, then this vertex must be adjacent to another vertex from Q as well, since otherwise Q would be of type 1. This is not possible since all of them are already coloured by colour in $\{1,2\}$.

Finally, suppose that Q is of type 3. Recall that x is not in P, thus there is no vertex with list $\{1,2\}$ adjacent to P.

We conclude that in all three cases $Q \cup V(P')$, and thus G, contains an induced P_3+P_4, a contradiction. $\diamondsuit$

We now run reduction Rules 1–13 exhaustively (and in the right order). Recall, however, that we make exception by not erasing Q. We continue with Branching VII.

Branching VII ($\mathcal{O}(n)$ branches). We branch by considering the possibility of removing colour 3 from the list of each vertex in A_2, and all possibilities of choosing one vertex in A_2, to which we give colour 3, and all possibilities of colouring its neighbour in A_1 (recall that this neighbour is unique due to Claim 3.22). This

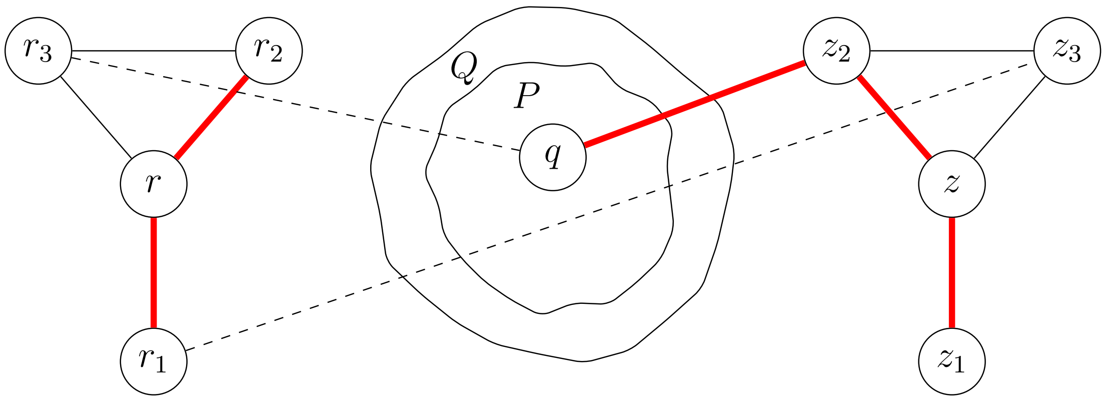

Figure 3.4: The situation in Branching VII. Dashed lines denote edges that might or might not be there.

leads to $\mathcal{O}(n)$ branches. We show that all of them are instances with no vertex with list of size three and thus Rule 2 can be applied on them.

In the first branch, all lists have size at most 2 directly by the construction.

Now consider a branch where a vertex $r \in A_2$ and its unique neighbour in A_1 were coloured (where r is given colour 3). We make an exception to Rule 4 and temporarily keep vertex r and all its neighbours in G, even if they need to be removed from G due to our rules.

Denote the vertex in $N(r) \cap A_1$ by r_1. Recall that $L(r_1) = \{1, 2\}$ and that every vertex in A_2 has exactly one neighbour in A_1. Note that $|N(u) \cap A_1|$ was equal to 1 before u was coloured. Before assigning a colour to r, vertex r had two other neighbours r_2 and r_3 by Rule 10, which were in $N_2 \cap N_3$. Vertices r_1, r, r_2 as well as r_1, r, r_3 induce a P_3, otherwise, there is either a K_4 or we use Rule 12 on vertex r. As G is $P_3 + P_4$-free, there must be at least one edge between P and $\{r_1, r, r_2\}$ and between P and $\{r_1, r, r_3\}$. We first show that such an edge is not incident to r_1.

If there exists an edge between r_1 and a vertex from P, then this vertex must be x (as $L(r_1) = \{1, 2\}$). First suppose Q is of type 1. Then $x = u$. However, u had only one neighbour in A_1, which is in Q, a contradiction. Now suppose Q is of type 2. Then $x = v$. If r_1 is adjacent to v, then r_1 is adjacent to another vertex in Q, a contradiction. Finally suppose that Q is of type 3. Then x in not in P. Thus r_1 is not adjacent to P.

We conclude from the above that there must exist an edge between r_2 and a vertex of P, and an edge between r_3 and a vertex of P. By Rule 6, these neighbours of r_2 and r_3 must be different. We show that vertices r_2 and r_3 received a colour, since r_2 and r_3 have a different neighbour in Q and thus at least one neighbour has obtained a colour different from 3. Since r is coloured by 3, the lists of r_2 and r_3 are reduced by Rule 5 to size 1 or the instance is a no-instance.

For sake of contradiction assume that there exists a vertex z with list of size three, i.e., $z \in A_2$. Note that $|N(z) \cap A_1| = 1$. The same observations for neighbours of z hold by the same arguments as above. Namely, vertex $z_1 \in N(z) \cap A_1$ does not have a neighbour in P and vertices z_2, z_3 are in $N_2 \cup N_3$ and they induce two P_3s: z_1, z, z_2 and z_1, z, z_3. Therefore, z_2, z_3 each have different neighbours in P, too. Moreover, at least one edge between r_1 and z_2, z_3 is missing by Rule 6. Without loss of generality $\{r_1, z_2\} \notin E$. Then vertices z_1, z, z_2, q, where q is in $N(z_2) \cap V(P)$, induce a new P_4. Again at least one vertex from r_2, r_3 is not adjacent to q, say $r_2q \notin E$. As r_1 and r_2 are coloured by 1 or 2, they have no edge to z_1 and to z; otherwise z and z_1 are not active by Rule 5. Recall that r_1, z_1 have no neighbour in P and that r had only one neighbour in A_1, thus r is not adjacent to z_1. By Claim 3.11 there are no edges between r_2, r_3 and z_2, z_3. Hence r_1, r, r_2 together with z_1, z, z_2, q induce a $P_3 + P_4$ in G, a contradiction (see Figure 3.4 for an example of such a situation).

The correctness of our algorithm follows from the above description. It remains to analyse its running time. The branching is done in seven stages (Branching I-VII) yielding a total number of $\mathcal{O}(n^{49})$ branches. It is readily seen that processing each branch created in Branching I-VII takes polynomial time. Hence the total running time of our algorithm is polynomial. □

Remark. Except for Phase 4 of our algorithm, all arguments in our proof hold for $(P_3 + P_5)$-free graphs. The difficulty in Phase 4 is that in contrary to the previous phases we cannot use the vertices from N_0 to find an induced $P_3 + P_5$ and therefore obtain the contradiction similarly to the previous phases.

3.3 The Proof of Corollary 3.2

By combining our new results from Section 3.2 with known results from the literature we can now prove Corollary 3.2.

Corollary 3.2 (restated). *Let H be a graph with $|V(H)| \leq 7$. If H is a linear forest, then* List 3-Colouring *is polynomial-time solvable for H-free graphs; otherwise already* 3-Colouring *is* NP*-complete for H-free graphs.*

Proof. If H is not a linear forest, then H contains an induced claw or a cycle, which means that 3-Colouring is NP-complete due to results in [56, 89, 115]. Suppose H is a linear forest. We first recall that List 3-Colouring is polynomial-time solvable for P_7-free graphs [27] and thus for $(rP_1 + P_7)$-free graphs for every integer $r \geq 0$ [19, 77]. Now suppose that H is not an induced subgraph of $rP_1 + P_7$ for any $r \geq 0$. If $H = P_1 + 3P_2$, then the class of H-free graphs is a subclass of $4P_3$-free graphs, for which List 3-Colouring is polynomial-time solvable [19, 77]. Otherwise, H has at least two connected components, all of which containing at least one edge. This means that $H \in \{2P_2 + P_3, P_2 + P_5, P_3 + P_4\}$. If $H = 2P_2 + P_3$, then the class of H-free graphs is a subclass of $4P_3$-free graphs, for which we just recalled that List 3-Colouring is polynomial-time solvable. The cases where $H = P_2 + P_5$ and $H = P_3 + P_4$ follow from Theorem 3.1. □

3.4 Conclusions

By solving two new cases we completed the complexity classifications of 3-Colouring and List 3-Colouring on H-free graphs for graphs H up to seven vertices. We showed that both problems become polynomial-time solvable if H is a linear forest, while they stay NP-complete in all other cases. Chudnovsky et al. improved our results in a recent arXiv paper [26] that appeared after our paper by showing that List 3-Colouring is polynomial-time solvable on (rP_3+P_6)-free graphs for any $r \geq 0$. In the same paper, they also proved that 5-Colouring is NP-complete for (P_2+P_5)-free graphs. Recall that k-Colouring ($k \geq 3$) is NP-complete on H-free graphs whenever H is not a linear forest. For the case where H is a linear forest, the NP-hardness result of [26] for 5-Colouring for (P_2+P_5)-free graphs, together with the known NP-hardness results of [90] for 4-Colouring for P_7-free graphs and 5-Colouring for P_6-free graphs, bounds the number of open cases of k-Colouring from above.

For future research, we remark that it is still not known if there exists a linear forest H such that 3-Colouring is NP-complete for H-free graphs. This is a notorious open problem studied in many papers; for a recent discussion see [79]. It is also open for List 3-Colouring, where an affirmative answer to one of the two problems yields an affirmative answer to the other one [78]. In the line of our proof method, we pose the question if 3-Colouring is polynomial-time solvable on (P_2+P_{t-2})-free graphs for some $t \geq 3$ whenever 3-Colouring is polynomial-time solvable for P_t-free graphs.

For $k \geq 4$, we emphasize that all open cases involve linear forests H whose connected components are small. For instance, if H has at most six vertices, then the polynomial-time algorithm for 4-Precolouring Extension on P_6-free graphs [28, 29] implies that there are only three graphs H with $|V(H)| \leq 6$ for which we do not know the complexity of 4-Colouring on H-free graphs, namely $H \in \{P_1+P_2+P_3, P_2+P_4, 2P_3\}$ (see [77]).

The main difficulty to extend the known complexity results is that hereditary graph classes characterized by a forbidden induced linear forest are still not sufficiently well understood due to their rich structure (proofs of algorithmic results for these graph classes are therefore often long and technical; see also, for example, [15, 28, 29]). We need a better understanding of these graph classes in order to make further progress. This is not only the case for the two colouring problems in this paper. For example, the Independent Set problem is known to be polynomial-time solvable for P_6-free graphs [83], but it is not known if there exists a linear forest H such that it is NP-complete for H-free graphs. A similar situation holds for Odd Cycle Transversal and Feedback Vertex Set and a whole range of other problems; see [13] for a survey.

Acknowledgments. We thank Karel Král for pointing out a mistake in a preliminary version of our paper.

4. Survey on Fair problems

In 1978 Yannakakis defined the notion of Deletion problems [145]. The task is to find a set of vertices (or edges) such that the graph after the removal of the found set satisfies a given property. Many classical graph problems can be reformulated in this way. For example, Vertex Cover is a set of vertices such that the rest of the graph is edgeless. An example of an edge deletion problem is feedback edge set, where the graph has to be a forest after the removal of edges. Usually, the size of the deleted set is minimized in such problems.

Fair problems have been first defined and studied by Lin and Sahni in 1989 [117]. They reused a definition of deletion problems but instead of minimizing the size of the deleted set F, they proposed a different optimization function, where $\max_{v \in V(G)} \deg_F(v)$ is minimized. We call them fair edge deletion problems. They examined several specific fair edge deletion problems. Since then fair problems have not been studied for several years. The notion was briefly investigated by Kolman, Lidický and Sereni in years 2009–10 [104][1] and [103]. They show an interesting connection with improper coloring. Recently, a systematic study of metatheorems involving fair problems [122, 102, 100] have been started, (See also Chapter 5 that contains all the results from [102]). They also proposed to start a study of some particular fair vertex deletion problems. In [102] they started a study of the fair vertex cover problem.

4.1 Definition of Fair Problems

Several different variants have been studied. Let $G = (V, E)$ be a graph.

The *fair cost* of a set $W \subseteq V$ is defined as $\max_{v \in V} |N(v) \cap W|$. The *fair cost* of a set $F \subseteq E$ is defined as $\max_{v \in V} \deg_F(v)$, where $\deg_F(v)$ is the degree of the vertex v in graph $G' = (V, F)$.

We present two different definitions of fair problems: the deletion and the evaluation variant. The evaluation is more powerful as it uses formulas with free variables. Also, we need to distinguish the vertex and edge version. In each, the appropriate fair cost is used. Therefore, we present four separate template definitions, each of them is modified by a graph logic L.

Fair Vertex L Deletion

Instance: An undirected graph $G = (V, E)$ and a positive integer k, an L sentence φ.

Question: Is there a set $W \subseteq V$ of fair cost at most k such that $G \setminus W \models \varphi$?

Fair Edge L Deletion

Instance: An undirected graph $G = (V, E)$ and a positive integer k, an L sentence φ.

Question: Is there a set $F \subseteq E$ of fair cost at most k such that $G \setminus F \models \varphi$?

[1] Paper [104] appeared also under a slightly different name [105].

To increase the power of the logic we can add free variables. Let $\varphi(X_1, \ldots, X_\ell)$ be a formula with one free variable and let $G = (V, E)$ be a graph. For sets $W_1, \ldots, W_\ell \subseteq V$, by $\varphi(W_1, \ldots, W_\ell)$ we mean that we substitute W_i for X_i in φ.We do it analogically for sets of edges.

ℓ-Fair Vertex L Evaluation

Instance: An undirected graph $G = (V, E)$ and a positive integer k, an L formula $\varphi(X_1, \ldots, X_\ell)$ with ℓ free variables.

Question: Are there sets $W_1, \ldots, W_\ell$ of vertices each of fair cost at most k such that $G \models \varphi(W_1, \ldots, W_\ell)$?

ℓ-Fair Edge L Evaluation

Instance: An undirected graph $G = (V, E)$ and a positive integer k, an L formula $\varphi(X_1, \ldots, X_\ell)$ with ℓ free variables.

Question: Are there sets $F_1, \ldots, F_\ell$ of edges each of fair cost at most k such that $G \models \varphi(F_1, \ldots, F_\ell)$?

We shortcut Fair Vertex as FV and Fair Edge as FE. When we talk about both in general then a shortcut FX is used. Naturally, a special evaluation variant with only one free variable, 1-FX L Evaluation, has been extensively studied. There the number of free variables in the name of the metatheorem is omitted in the rest of the paper, therefore they are denoted as FX L Evaluation. This problem is called Generalized Fair L Vertex-Deletion in [122] and in its conference version [121], however, we believe that evaluation is a more suitable expression and moreover, it coincides with standard terminology in logic.

We would like to point out one crucial difference between deletion and evaluation problems, namely that in evaluation problems we have access to the variable that represents the solution. This enables evaluation problems to impose some conditions on the solution, e.g., we can ensure that the graph induced by the solution has a diameter at most 2 or that it is triangle-free.

Recall, in dense graph classes, one cannot obtain an MSO_2 model checking algorithm running in FPT-time, in fact, MSO_2 on cliques is not even in XP unless E = NE [36, 112]. That is a reason why parameterized algorithms for edge evaluation problems might be problematic or probably even impossible in the dense context.

4.2 Specific Problems

Here, we discuss the studied specific fair problems together with corresponding results.

4.2.1 Edge Problems

Fair Feedback Edge Set Lin and Sahni [117] showed that the Fair Feedback Edge Set problem, the resulting graph needs to be a forrest[2], is NP-complete.

[2]In fact, in the original paper only fair deletion into a tree was mention. However, this more classical formulation does not change the complexity.

Interestingly, they also show that a variant of this problem, fair deletion into a weakly-connected directed acyclic graph, is solvable in linear time on directed graphs.

Fair Odd Cycle Transversal, (k, d)-coloring (defective coloring) Kolman, Lidický, and Sereni [104] studied the Fair Odd Cycle Transversal problem. There, the resulting graph needs to be bipartite. They provide $\Theta(\sqrt{(n)})$-approximation algorithm and a polynomial-time algorithm whenever the input graph has bounded tree-width.

Kolman, Lidický and Sereni [104, 103] brought up the connection of the Fair Odd Cycle Transversal (Fair OCT) with $(2, d)$-coloring. *Defective colorings* (also known as improper colorings), here denoted as (k, d)*-coloring* of a graph $G = (V, E)$ is defined as a partition of V into k parts such that each part induces a subgraph of G with maximum degree at most d. Note that the standard proper k-coloring is exactly $(k, 0)$-coloring. Observe that a graph has a $(2, d)$-coloring if and only if it has a Fair OCT of the fair cost at most d. More generally, (k, d)-coloring is exactly Fair partition of Vertices into the independent set with the fair edge cost at most d. In this light, (k, d)-coloring can be expressed in the framework of Fair Edge MSO_1 Deletion. For $G = (V, E)$ we express the problem by the following formula and by bounding the fair cost by d.

$$\exists X_1, \ldots, X_k \subseteq V \quad \text{s.t.} \quad \forall i \in \{1, \ldots, k\} \forall u, v \in X_i \quad \{u, v\} \notin E.$$

Defective colorings have been introduced in 1986 [40], but the history originated in as far as 1974 [94]. It developed into a very extensive area of research studied both as a theoretical notion [87, 147, 144] as well as a practical tool [86, 95]. For the details cf. a survey by Frick [64] from 1993, or a very recent survey by Wood [143] from 2018. Plenty of related results are already known and described. In the spirit of complexity we mention that $(2, d)$-coloring problem for $d \geq 1$ is NP-complete on planar graphs [32, 39]. Recentlay (2018–19), two papers about parameterized complexity of defective colorings appered [8] and [84][3].

Paper [84] defines a variant of defective coloring called Weighted Improper Coloring. For an arc-weighted directed graph $D = (V, A)$ with weight function $w : A \to [0, 1]$ Weighted Improper k-coloring is defined as a mapping $c : V \to k$ such that for each vertex v of color a it holds:

$$\sum_{c(u)=a \wedge (u,v) \in A} w(u, v) < 1.$$

They show that (k, d)-coloring can be reduced in polynomial time to weighted improper k-coloring. This reduction only multiplies by 2 the number of edges so their FPT results can be translated into the setting of (k, d)-coloring. They claim an FPT algorithm parameterized by tree-width and maximum degree.

Paper [8] shows W-hardness of (k, d)-coloring parameterized by treedepth for any fixed $k \geq 2$. The problem is also W-hard for $k = 2$ parameterized by feedback vertex set. Surprisingly, the problem turns out to be in FPT for larger fixed $k > 2$ when parameterized by feedback vertex set. They also show that whenever all three k, d and treewidth are parameters then the problem is in FPT. Moreover,

[3]Paper [84] is an extended version of the published paper [85].

the authors show various approximation algorithms in this setting. Last, they show that the problem is in FPT parameterized by vertex cover. This is also implied by the vertex cover metatheorem in [122] and the observation that the number k is always bounded by vertex cover since otherwise it is a yes-instance.

Minimum Fair Cut Kolman, Lidický, and Sereni [104] propose a study of the FAIR s, t-CUT problem. There, the resulting graph has vertex s in a different component than vertex t. They also provide $\Theta(\sqrt{(n)})$-approximation algorithm and a polynomial-time algorithm whenever the input graph is series-parallel. Those graphs have treewidth at most 2. Moreover, they hinted how to extend such algorithm to graphs of bounded tree-width.

4.2.2 Vertex Problems

In general, the study of the Fair Vertex Deletion problems appeared explicitly much later. Up to our best knowledge, it was first formulated in 2017 [121].

Fair Vertex Cover The study of one particular problem Fair Vertex Deletion problem, Fair Vertex Cover, was proposed in [102]. The authors show that this problem is W[1]-hard parameterized by treedepth and feedback vertex set combined. On the other hand, they propose an FPT algorithm parameterized by modular-width.

4.3 Metatheorems

In [104] the authors proposed a search for a general metatheorem involving fair problems. Soon after, the same authors provide the first result in this direction in [103]. They show that every Fair Edge MSO_2 Deletion problem is solvable in XP time parameterized by tree-width of the graph. According to to [122] their approach can be straightforwardly transformed to Fair Vertex MSO_2 Deletion problems. This result was later extended to even more powerful logic in [100] using an interesting method of CSP reformulations.

4.3.1 FPT Results

Table 4.1 summarizes FPT results. In this Table, we are focused on the combination of the expressive power of logic and the structural class. That means that ETH-based lower bounds are not discussed here, despite they might matter, see [122] for details. There a stronger lower bound was derived for a more powerful logic and a slightly stronger parameter. Refer to Figure 4.1 to compare the results between edge and vertex variant on the hierarchy of parameters.

4.4 Open Problems and Further Research Direction

There are many possible directions for future research.

Left part of the table	vc	fvs + td	tc
FAIR VERTEX COVER	✓	[102]	✓
FV L DELETION	MSO_2 ✓	$\mathsf{L}_\emptyset$ ✓	MSO_1 ✓
FV L EVALUATION	MSO_2 [122]	$\mathsf{L}_\emptyset$ ✓	MSO_1 [102]
ℓ-FV L EVALUATION	MSO_1 ✓	$\mathsf{L}_\emptyset$ ✓	MSO_1 [102]
FE L DELETION	MSO_2 ✓	FO [122]	?
FE L EVALUATION	MSO_2 [122]	FO ✓	?

Right part of the table	nd	cvd	mw
FAIR VERTEX COVER	✓	?	[102]
FV L DELETION	MSO_1 ✓	?	?
FV L EVALUATION	MSO_1 ✓	?	?
ℓ-FV L EVALUATION	MSO_1 [100]	MSO_1 ✓	MSO_1 ✓
FE L DELETION	?	FO [102]	?
FE L EVALUATION	?	FO ✓	?

Table 4.1: The table summarize metatheorem results (with a citation), in terms of FPT complexity and used logic, parameterized by structural parameters. Green cells denote FPT results and red cells represent hardness results. Logic L in metatheorems is specified by a logic used in the respective theorem. Symbol $\mathsf{L}_\emptyset$ denotes any logic that can express an edgeless graph. Symbol ✓denotes implied results. A question mark (?) indicates that the complexity is unknown. Fair edge problems are delimited from fair vertex problems since there are no apparent relations between them.

Metatheorems—Fair Edge problems In terms of methatheorems, the largest yet unsolved group are edge problems. Compare the known results in Table 4.1. In particular, there are no results (except for implied harndnesses) for the ℓ-FE L EVALUATION problem. The systematic step is to determine the complexity of any fair edge problem on neighborhood diversity. Last but not least, it would be interesting to formulate a "basic" edge deletion problem that is by itself NP-hard. In a similar spirit as Fair Vertex Cover is a somewhat basic problem for vertex deletion problems.

Metatheorems—Fair Vertex problems The logical next step in the investigation of Fair Vertex metatheorems is a study of parameterization by modular-width and cluster vertex deletion. In hardness results, it might be useful to explore Marx and Pilipczuk subgraph isomorphism problem [120] to obtain a better ETH-based lower bound.

Future extensions of Fair Framework Previously, an FPT algorithm for evaluation of a fair objective was given for parameter neighborhood diversity [122]. That algorithm was extended [100] to a so-called *local linear constraints* which are defined for every vertex $v \in V(G)$. Each vertex gets two positive integers $\ell(v), u(v)$, the lower and the upper bound, and the task is to find a set X that not only $G \models \varphi(X)$ but for each $v \in V(G)$ it holds that $\ell(v) \leq |N(v) \cap X| \leq u(v)$. Note that this is a generalization as fair cost t can be tested in this framework by setting $\ell(v) = 0$ and $u(v) = t$ for every $v \in V(G)$. Is this extension possible for

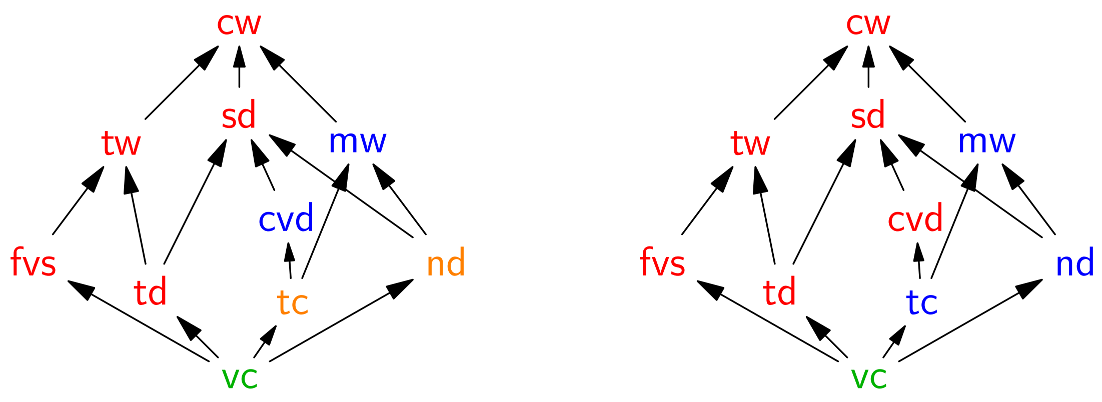

Figure 4.1: Hierarchy of graph parameters with depicted complexity of the FAIR L VERTEX EVALUATION problem on the left side and the FAIR L VERTEX EVALUATION problem on the right side. An arrow indicates that a graph parameter upper-bounds the other. Thus, hardness results are implied in the direction of arrows, and FPT algorithms are implied in the reverse direction. Green colors indicate FPT results for MSO_2, orange are FPT for MSO_1, blue are open, and red are hardness results. Descriptions of the relations and parameteres are in Section 2.2.1.

other graph parameters, for example, the twin cover number?

Specific fair Vertex problems Besides Fair Vertex Cover nothing more has been considered in terms of specific Fair Vertex Deletion problems. We propose a variant of well-studied problems, for instance, Fair Dominating Set or Fair Feedback Vertex Set.

Parametrization by the fair cost It will be interesting to explore the complexity for parameterization by the “natural parameter”. Our proposal is to explore the parameterization by the fair cost in addition to the size of the formula. This can be seen as a natural parameter since it plays a similar to the size of the solution in the classical problems.

5. Methatheorems for Fair Problems

5.1 Introduction

A prototypical graph problem is centered around a fixed property for a set of vertices. A solution is any set of vertices for which the given property holds. In a similar manner, one can define the solution as a set of vertices such that the given property holds when we remove this set of vertices from the input graph. This leads to the introduction of deletion problems—a standard reformulation of some classical problems in combinatorial optimization introduced by Yannakakis [146]. Formally, for a graph property π we formulate a *vertex deletion problem.* That is, given a graph $G = (V, E)$, find the smallest possible set of vertices W such that $G \setminus W$ satisfies the property π. Many classical problems can be formulated in this way such as Minimum Vertex Cover (π describes an edgeless graph) or Minimum Feedback Vertex Set (π is valid for forests). Of course, the complexity is determined by the desired property π but most of these problems are NP-complete [145, 3, 109].

Clearly, the complexity of a graph problem is governed by the associated predicate π. We can either study one particular problem or a broader class of problems with related graph-theoretic properties. One such relation comes from logic, for example, two properties are related if it is possible to express both by a first order (FO) formula. Then, it is possible to design a *model checking algorithm* that for any property π expressible in the fixed logic decides whether the input graph with the vertices from W removed satisfies π or not.

Undoubtedly, Courcelle's Theorem [33] for graph properties expressible in the monadic second-order logic (MSO) on graphs of bounded treewidth plays a prime role among model checking algorithms. In particular, Courcelle's Theorem provides for an MSO sentence φ an algorithm that given an n-vertex graph G with treewidth k decides whether φ holds in G in time $f(k, |\varphi|)n$, where f is some computable function and $|\varphi|$ is the quantifier depth of φ. In terms of parameterized complexity, such an algorithm is called *fixed-parameter tractable* (the problem belongs to the class FPT for the combined parameter $k + |\varphi|$). We refer the reader to monographs [41, 51] for background on parameterized complexity theory and algorithm design. There are many more FPT model-checking algorithms, e.g., an algorithm for (existential counting) modal logic model checking on graphs of bounded treewidth [129], MSO model checking on graphs of bounded neighborhood diversity [111], or MSO model checking on graphs of bounded shrubdepth [73] (generalizing the previous result). First order logic (FO) model checking received recently quite some attention as well and algorithms for graphs with bounded degree [133], nowhere dense graphs [81], and some dense graph classes [67] were given.

Kernelization is one of the most prominent techniques for designing FPT algorithms [61]. It is a preprocessing routine that in polynomial time reduces the input instance to an equivalent one whose size and parameter value can be upper-bounded in terms of the input (i.e., original) parameter value. It should be noted that kernelization is not really common for model checking algorithms.[1] There are few notable exceptions—the result of Lampis [111] (see Proposition 5.7) was, up to our knowledge, the first result of this kind. Lampis [111] presented a kernel for MSO_1 model checking in graphs of bounded neighborhood diversity (of exponential size in quantifier depth). The aforementioned result has been recently followed by Gajarský and Hliněný [66] who showed a kernel for MSO_1 model checking in graphs of bounded shrubdepth (a parameter generalizing neighborhood diversity, twin cover, and treedepth). It is worth noting that it was possible to use the kernel of [111] to extend his model checking algorithm for so-called fair objectives and their generalizations [122, 100]. In this work, we continue this line of research.

Fair Objective. Fair deletion problems, introduced by Lin and Sahni [117], are such modifications of deletion problems where instead of minimizing the size of the deleted set we change the objective. The Fair Vertex Deletion problem is defined as follows. For a given graph $G = (V, E)$ and a property π, the task is to find a set $W \subseteq V$ which minimizes the maximum number of neighbors in the set W over all vertices, such that the property π holds in $G \setminus W$. Intuitively, the solution should not be concentrated in a neighborhood of any vertex. In order to classify (fair) vertex deletion problems we study the associated decision version, that is, we are interested in finding a set W of size at most k, for a given number k. Note that this can introduce only polynomial slowdown, as the value of our objective is bounded by 0 from below and by the number of vertices of the input graph from above. Since we are about to use a formula with a free variable X to express the desired property π, we naturally use X to represent the set of deleted vertices in the formula. The *fair cost of a set* $W \subseteq V$ is defined as $\max_{v \in V} |N(v) \cap W|$. We refer to the function that assigns each set $W \subseteq V$ its fair cost as to the *fair objective function*. Here, we give a formal definition of the computational problem when the property π is defined in some logic L.

Fair L Vertex Deletion

Instance: An undirected graph $G = (V, E)$ and a positive integer k, an L sentence φ.

Question: Is there a set $W \subseteq V$ of fair cost at most k such that $G \setminus W \models \varphi$?

Let $\varphi(X)$ be a formula with one free variable and let $G = (V, E)$ be a graph. For a set $W \subseteq V$ by $\varphi(W)$ we mean that we substitute W for X in φ. The definition bellow can be naturally generalized to contain ℓ free variables. We would like to point out one crucial difference between deletion and evaluation problems, namely that in evaluation problems we have access to the variable that represents the solution. This enables evaluation problems to impose some conditions on the solution, e.g., we can ensure that the graph induced by the solution has diameter at most 2 or is triangle-free.

[1]We are aware of the equivalence of FPT and kernels (cf. [41]), however, we would like to point out that the difference between a directly applicable set of rules and a theoretical bound is large.

Fair L Vertex Evaluation[2]	
Instance:	An undirected graph $G = (V, E)$ and a positive integer k, an L formula $\varphi(X)$ with one free variable.
Question:	Is there a set $W \subseteq V$ of fair cost at most k such that $G \models \varphi(W)$?

One can define *edge deletion problems* in a similar way as vertex deletion problems.

Fair L edge deletion	
Instance:	An undirected graph $G = (V, E)$, an L sentence φ, and a positive integer k.
Question:	Is there a set $F \subseteq E$ such that $G \setminus F \models \varphi$ and for every vertex v of G, it holds that $\lvert\{e \in F : e \ni v\}\rvert \le k$?

The evaluation variant is analogical. Recall, in dense graph classes, one cannot obtain an MSO_2 model checking algorithm running in FPT-time [112]. That is the reason why evaluation problems do not make sense in this context. In sparse graph classes, this problem was studied in [122] where $\mathsf{W}[1]$-hardness was obtained for Fair FO Edge Deletion on graphs of bounded treedepth and FPT algorithm was derived for Fair MSO_2 Edge Evaluation on graphs of bounded vertex cover.

Minimizing the fair cost arises naturally for edge problems in many situations as well, e.g., in defective coloring [40], which has been substantially studied from the practical network communication point of view [86, 95]. A graph $G = (V, E)$ is (k, d)-colorable if every vertex can be assigned a color from the set $[k]$ in such a way that every vertex has at most d neighbors of the same color. This problem can be reformulated in terms of fair deletion; we aim to find a set of edges F such that graph $G' = (V, F)$ has maximum degree d and $G \setminus F$ can be partitioned into k independent sets.

Related results. There are several possible research directions once a model checking algorithm is known. One possibility is to broaden the result either by enlarging the class of graphs it works for or by extending the expressive power of the concerned logic, e.g., by introducing a new predicate [106]. Another obvious possibility is to find the exact complexity of the general model checking problem by providing better algorithms (e.g., for subclasses [111]) and/or lower bounds for the problem [65, 112]. Finally, one may instead of deciding a sentence turn attention to finding a set satisfying a formula with a free variable that is optimal with respect to some objective function [6, 37, 74]. In this work, we take the last presented approach—extending a previous work on MSO model checking for the fair objective function.

When extending a model checking result to incorporate an objective function or a predicate, we face two substantial difficulties. On the one hand, we are trying to introduce as strong extension as possible, while on the other, we try not to worsen the running time too much. Of course, these two are in a clash. One evident

[2]This problem is called Generalized Fair L Vertex-Deletion in [122] and in the respective conference version [121]. However, we believe that evaluation is a more suitable expression and coincides with standard terminology in logic.

possibility is to sacrifice the running time and obtain an XP algorithm, that is an algorithm running in time $f(k)|G|^{g(k)}$. As an example there is an XP algorithm for the Fair MSO_2 Vertex Evaluation problems parameterized by the treewidth (and the formula) by Kolman et al. [103] running in time $f(|\varphi|, \mathsf{tw}(G))|G|^{g(\mathsf{tw}(G))}$. An orthogonal extension is due to Szeider [136] for the so-called local cardinality constraints (MSO-LCC) who gave an XP algorithm parameterized by treewidth. If we decide to keep the FPT running time, such result is not possible for treedepth (consequently for treewidth) as we give W[1]-hardness result for a very basic Fair $\mathsf{L}_\emptyset$ Vertex Deletion problem[3] in this paper. A weaker form of this hardness was already known for FO logic [122]. However, Ganian and Obdržálek [74] study CardMSO and provide an FPT algorithm parameterized by neighborhood diversity. Recently, Masařík and Toufar [122] examined fair objective and provide an FPT algorithm for the Fair MSO_1 Vertex Evaluation problem parameterized by neighborhood diversity. See also [100] for a comprehensive discussion of various extensions of the MSO.

Since classical Courcelle's theorem [33] have an exponential tower dependence on the quantifier depth of the MSO formula, it could be interesting to find a more efficient algorithm. However, Frick and Grohe [65] proved that this dependence is unavoidable, unless $\mathsf{P} = \mathrm{NP}$. On the other hand, there are classes where MSO model checking can be done in asymptotically faster time, e.g., the neighborhood diversity admits a single-exponential dependence on the quantifier depth of the formula [111]. For a recent survey of algorithmic metatheorems consult the article by Grohe and Kreutzer [80].

We want to turn particular attention to the Fair Vertex Cover (Fair VC) problem which, besides its natural connection with Vertex Cover, has some further interesting properties. For example, we can think about classical vertex cover as having several crossroads (vertices) and roads (edges) that we need to monitor. However, people could get uneasy if they will see too many policemen from one single crossroad. In contrast, if the vertex cover has low fair cost, it covers all roads while keeping a low number of policemen in the neighborhood of every single crossroad.

5.1.1 Our Results

We give a metatheorem for graphs having bounded twin cover. Twin cover (introduced by Ganian [69]; see also [70]) is one possible generalization of the vertex cover number. Here, we measure the number of vertices needed to cover all edges that are not twin-edges; an edge $\{u, v\}$ is a *twin-edge* if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. Ganian introduced twin cover in the hope that it should be possible to extend algorithms designed for parameterization by the vertex cover number to a broader class of graphs.

Theorem 5.1. *The* Fair MSO_1 Vertex Evaluation *problem parameterized by the twin cover number and the quantifier depth of the formula admits an* FPT *algorithm.*

We want to point out here that in order to obtain this result we have to reprove the original result of Ganian [69] for MSO_1 model checking on graphs of bounded

[3]Here, $\mathsf{L}_\emptyset$ stands for any logic that can express an edgeless graph.

twin cover. For this, we extend arguments given by Lampis [111] in the design of an FPT algorithm for MSO_1 model checking on graphs of bounded neighborhood diversity. We do this to obtain better insight into the structure of the graph (a kernel) on which model checking is performed (its size is bounded by a function of the parameter). This, in turn, allows us to find a solution minimizing the fair cost and satisfying the MSO_1 formula. The result by Ganian in version [69] is based on the fact that graphs of bounded twin cover have bounded shrubdepth and so MSO_1 model checking algorithm on shrubdepth ([73, 66]) can be used.

When proving hardness results it is convenient to show the hardness result for a concrete problem that is expressible by an MSO_1 formula, yet as simple as possible. Therefore, we introduce a key problem for Fair Vertex Deletion—the Fair VC problem.

Fair Vertex Cover (Fair VC)	
Instance:	An undirected graph $G = (V, E)$, and a positive integer k.
Question:	Is there a set $W \subseteq V$ of fair cost at most k such that $G \setminus W$ is an independent set?

Fair VC problem can be expressed in any logic that can express an edgeless graph (we denote such logic $\mathsf{L}_\emptyset$) which is way weaker even than FO. Therefore, we propose a systematic study of Fair VC problem which, up to our knowledge, have not been considered before.

Theorem 5.2. *The* Fair VC *problem parameterized by treedepth* $\mathsf{td}(G)$ *and feedback vertex set* $\mathsf{fvs}(G)$ *combined is* $\mathsf{W}[1]$*-hard. Moreover, if there exists an algorithm running in time* $f(w)n^{o(\sqrt{w})}$*, where* $w := \mathsf{td}(G) + \mathsf{fvs}(G)$*, then Exponential Time Hypothesis fails.*

Note that this immediately yields $\mathsf{W}[1]$-hardness and $f(w)n^{o(\sqrt{w})}$ lower bound for Fair $\mathsf{L}_\emptyset$ Vertex Evaluation. Previously, an $f(w)n^{o(w^{1/3})}$ lower bound was given for FO logic by Masařík and Toufar [122]. Thus our result is stronger in both directions, i.e., the lower bound is stronger, and the logic is less powerful. On the other hand, we show that Fair VC can be solved efficiently in dense graph models.

Theorem 5.3. *The* Fair VC *problem parameterized by modular width* $\mathsf{mw}(G)$ *admits an* FPT *algorithm with running time* $2^{\mathsf{mw}(G)}\mathsf{mw}(G)n^3$*, where* n *is the number of vertices in* G*.*

We point out that the Fair VC problem is (rather trivially) AND-compositional and thus it does not admit a polynomial kernel for parameterization by modular width.

Lemma 5.4. *The* Fair VC *problem parameterized by the modular width of the input graph does not admit a polynomial kernel, unless* NP $\subseteq$ $\mathsf{coNP}/$poly*.*

Moreover, an analog to Theorem 5.3 cannot hold for parameterization by shrubdepth of the input graph. This is a consequence of Theorem 5.2 and [73, Proposition 3.4].

Another limitation in a rush for extensions of Theorem 5.1 is given when aiming for more free variables.

ℓ-FAIR L VERTEX EVALUATION	
Instance:	An undirected graph $G = (V, E)$ and a positive integer k, an L formula $\varphi(X_1, \ldots, X_\ell)$ with ℓ free variables.
Question:	Are there sets $W_1, \ldots, W_\ell \subseteq V$ such that $G \models \varphi(W_1, \ldots, W_\ell)$ having fair cost at most k?

The *fair cost* of $W_1, \ldots, W_\ell$ is defined as $\max_{v \in V} \max_{i \in [\ell]} |N(v) \cap W_i|$.

It is very surprising that such a generalization is not possible for parameterization by twin cover since the same extension is possible for parameterization by neighborhood diversity [100]. In fact, they prove something even stronger, i.e., an FPT algorithm parameterized by neighborhood diversity in the context of $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$ is given in [100]. In $\mathsf{MSO}^{\mathsf{L}}_{\mathsf{lin}}$ one can specify both lower- and upper-bound for each vertex and each free variable (i.e., a feasibility interval is given for every vertex).

Theorem 5.5. *The* ℓ-FAIR FO VERTEX EVALUATION *problem is* $\mathsf{W[1]}$*-hard for parameter ℓ even on graphs with twin cover of size one. Moreover, if there exists an algorithm running in time $f(\ell)n^{O(\ell/\log \ell)}$, then Exponential Time Hypothesis fails.*

Furthermore, we obtain a hardness result for the Fair FO Edge Deletion problem parameterized by the cluster vertex deletion number. Observe that for any graph its cluster vertex deletion number is upper bounded by the size of its twin cover.

Theorem 5.6. *The* FAIR FO EDGE DELETION *problem is* $\mathsf{W[1]}$*-hard when parameterized by the cluster vertex deletion number of the input graph.*

For an overview of the results, please refer to Table 5.1[4] and to Figure 5.1 for the hierarchy of classes.

5.1.2 Preliminaries

We denote the set $\{1, \ldots, n\}$ by $[n]$. We deal with simple undirected graphs, for further standard notation we refer to monographs: graph theory [124] and parameterized complexity [41]. For a vertex v by $N(v)$ we denote the neighborhood of v and by $N[v]$ we denote the closed neighborhood of vertex v, i.e., $N(v) \cup \{v\}$.

A parameter closely related to twin cover is *cluster vertex deletion* ($\mathsf{cvd}(G)$), that is, the smallest number of vertices one has to delete from a graph in order to get a collection of (disjoint) cliques. *Treedepth* of a graph G ($\mathsf{td}(G)$) is the minimum height of a rooted forest whose transitive closure contains the graph G [126]. *Feedback vertex set* ($\mathsf{fvs}(G)$) is the minimum number of vertices of a graph G whose removal leaves a graph without cycles. *Neighborhood diversity* ($\mathsf{nd}(G)$) is the smallest integer r such that the graph can be partitioned into r sets where each set is either complete graph or independent set and each pair of sets forms either a complete bipartite graph or there is no edge between them. *Modular width* of a graph G ($\mathsf{mw}(G)$), is the smallest positive integer r such that G can be obtained from an algebraic expression of width at most r, defined as

[4]A similar table (Table 4.1) was presented in Chapter 4. Compared to it Table 5.1 is tailored to highlight the results specific for this paper.

Left part of the table	vc	fvs + td	tc
Fair Vertex Cover	🟩 $*$	🟥 T5.2	🟩 ✓
FV L Deleletion	🟩 MSO_2 $*$	🟥 $\mathsf{L}_\emptyset$ ✓	🟩 MSO_1 ✓
FV L Evaluation	🟩 MSO_2 [122]	🟥 $\mathsf{L}_\emptyset$ ✓	🟩 MSO_1 T5.1
ℓ-FV L Evaluation	🟩 MSO_1 $*$	🟥 $\mathsf{L}_\emptyset$ ✓	🟥 MSO_1 T5.5
FE L Deleletion	🟩 MSO_2 [122]	🟥 FO [122]	?

Right part of the table	nd	cvd	mw
Fair Vertex Cover	🟩 $*$	?	🟩 T5.3
FV L Deletion	🟩 MSO_1 $*$	?	?
FV L Evaluation	🟩 MSO_1 $*$	?	?
ℓ-FV L Evalution	🟩 MSO_1 [100]	🟥 MSO_1 ✓	🟥 MSO_1 ✓
FE L Deleletion	?	🟥 FO T5.6	?

Table 5.1: The table summarizes some related (with a citation) and all the presented (with a reference) results on the studied parameters. Green cells denote FPT results, and red cells represent hardness results. Logic L in metatheorems is specified by a logic used in the respective theorem. Symbol $*$ denotes implied results from previous research and symbol ✓denotes new implied results. A question mark (?) indicates that the complexity is unknown. The Fair Edge L Deletion problem is delimited from Vertex problems since there are no apparent relations between them.

follows. The *width of an expression* A is the maximum number of operands used by any occurrence of the substitution operation in A, where A is an algebraic expression that uses the following operations:

1. Create an isolated vertex.

2. The *substitution operation* with respect to a template graph T with vertex set $[r]$ and graphs $G_1, \dots, G_r$ created by algebraic expression. The substitution operation, denoted by $T(G_1, \dots, G_r)$, results in the graph on vertex set $V = V_1 \cup \dots \cup V_r$ and edge set $E = E_1 \cup \dots \cup E_r \cup \bigcup_{\{i,j\} \in E(T)} \Big\{ \{u, v\} : u \in V_i, v \in V_j \Big\}$, where $G_i = (V_i, E_i)$ for all $i \in [r]$.

An algebraic expression of width $\mathsf{mw}(G)$ can be computed in linear time [137].

We stick with standard definitions and notation in logic. For a comprehensive summary, please consult a book by Libkin [116].

5.2 Metatheorems for Fair Evaluation

First we show an FPT algorithm for the fair MSO_1 vertex evaluation problem parameterized by the twin cover number as it is stated in Theorem 5.1.

We give a more detailed statement that implies the promised result straigthforwardly. We split the proof into two parts. First, we show an algorithm for MSO_1 model checking parameterized by twin cover of the graph (Proposition 5.9). In the second part, we prove that we can even compute the optimal fair cost (Proposition 5.13) and so derive the promised result.

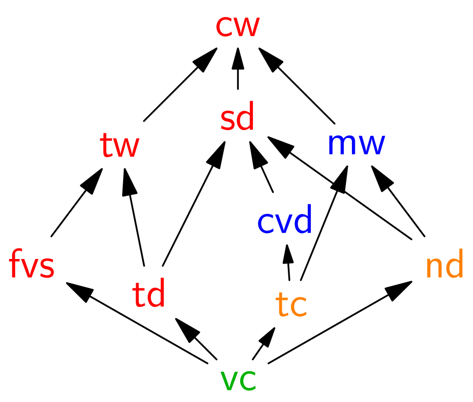

Figure 5.1: Hierarchy of graph parameters with depicted complexity of the FAIR L VERTEX EVALUATION problem. An arrow indicates that a graph parameter upper-bounds the other. Thus, hardness results are implied in the direction of arrows, and FPT algorithms are implied in the reverse direction. Green colors indicate FPT results for MSO_2, orange are FPT for MSO_1, blue are open, and red are hardness results. We denote treewidth by tw, shrubdepth by sd, and clique-width by cw. We refer to book [41] for definitions. Other parameters and their respective abbreviations are defined in Subsection 5.1.2.

Overview of the Algorithm. For the model checking algorithm, we closely follow the approach of Lampis [111]. The idea is that if there is a large set of vertices with the same closed neighborhood, then some of them are irrelevant, i.e., we can delete them without affecting the truthfulness of the given formula φ. For graphs of bounded neighborhood diversity using this rule alone can reduce the number of vertices below a bound that depends on $\mathsf{nd}(G)$ and $|\varphi|$ only, thus providing an FPT model checking algorithm. For the graphs of bounded twin cover, this approach can be used to reduce the size of all (twin) cliques, yet their number can still be large. We take the approach one step further and describe the deletion of irrelevant cliques in a similar manner; these rules together yield a model checking algorithm for graphs of bounded twin cover.

The reduction rules also lead to a notion of *shape* of a set $W \subseteq V$. The motivation behind shapes is to partition all subsets of V such that if two sets W, W' have the same shape, then $G \models \varphi(W)$ if and only if $G \models \varphi(W')$. This allows us to consider only one set of each shape for the purposes of model checking. Since the number of all distinct shapes is bounded by some function of parameters, we can essentially brute force through all possible shapes.

A final ingredient is an algorithm that for a given shape outputs a subset of vertices with this shape that minimizes the fair cost. This algorithm uses ILP techniques, in particular minimizing quasiconvex function subject to linear constraints.

Notation. In what follows $G = (V, E)$ is a graph and K is its twin cover of size k. An MSO_1 formula φ contains q_S set quantifiers and q_v element (vertex) quantifiers. Given a twin cover K and $A \subseteq K$, we say that A is the *cover set* of a set $S \subseteq V \setminus K$ if every $v \in S$ has $N(v) \cap K = A$. Note that, by the definition of twin cover, for all $u, v \in V \setminus K$ with $\{u, v\} \in E$ we have that A is a cover set for u if and only if A is a cover set for v. We say that two cliques have the same *type* if they have the same size and the same cover set. Clearly, if the cover set is fixed, two cliques agree on type if and only if their sizes are the same. We define a *labeled graph*, that is, a graph and a collection of labels on the vertices. We say that two cliques have the *same labeled type* if all of them have the same size, the same cover set and the same labels on vertices.

5.2.1 Model Checking

We give a reformulated combination of Lemma 5 and Theorem 4 by Lampis [111].

Proposition 5.7 ([111, Lemma 5 and Theorem 4])**.** *Let φ be an MSO_1 formula and let G be a labeled graph. If there is a set S of more than $2^{q_S} q_v$ vertices having the same closed neighborhood and the same labels, then for any $v \in S$ we have $G \models \varphi$ if and only if $G \setminus v \models \varphi$.*

In particular, if G is a graph with just one label, then for any clique C where each vertex has exactly the same closed neighborhood in G the following holds. Either there is a vertex $v \in C$ such that $G \models \varphi$ if and only if $G \setminus v \models \varphi$ or the size of C is bounded by

$$2^{q_S+1} q_v.$$

Proposition 5.7 bounds the size of a maximum clique in $G \setminus K$ because we can apply it repeatedly for each clique that is bigger than the threshold $2^{q_S+1} q_v$. Now, we need to bound the number of cliques of each type. For this, we establish the following technical lemma.

Lemma 5.8. *Let G be a labeled graph with twin cover K. Let φ be an MSO_1 formula with q_v element quantifiers and q_S set quantifiers. Suppose the size of a maximum clique in $G \setminus K$ is bounded by r. If there are strictly more than*

$$\alpha(q_S, q_v) = 2^{r q_S}(q_v + 1)$$

cliques of the same labeled type $\mathcal{T}$, then there exists a clique C of the labeled type $\mathcal{T}$ such that $G \models \varphi$ if and only if $G \setminus C \models \varphi$.

Proof. We prove the lemma by induction on $q_S + q_v$. Without loss of generality, all the quantifiers are assumed to be existential.

The base case of the induction is a quantifier-free formula. If there is at least one labeled type with at least two cliques C_1, C_2 then the following holds. If $G \models \varphi$ then $G \setminus C_1 \models \varphi$ clearly holds as well since clique C_2 has the same cover set and the same labels and a quantifier-free formula can only examine the labeled vertices. If $(G \setminus C_1) \models \varphi$ and since C_1, C_2 have the same labels the same size and the same cover set so $G \models \varphi$.

For the induction case, we consider the first quantified variable in φ and we split the proof whether it is set or vertex variable. Suppose there is at least one labeled type which contains strictly more than $\alpha(q_S, q_v)$ cliques.

If it is a set variable then we try all possible assignments of the variable to cliques of the chosen labeled type. There are at most 2^r of possible assignments to single clique and so from cliques of one labeled type emerge at most 2^r different labeled (sub)types of cliques. We can compute that at least one of them has strictly more than $\alpha(q_S - 1, q_v)$:

$$\left\lceil \frac{\alpha(q_S, q_v) + 1}{2^r} \right\rceil \geq \left\lceil 2^{r(q_S-1)}(q_v + 1) + \frac{1}{2^r} \right\rceil \geq \alpha(q_S - 1, q_v) + 1.$$

So, by the induction assumption we know that there is a clique C in the newly created labeled (sub)type of the promised properties and so of the larger labeled type.

If it is a vertex variable then only one more different labeled type can be created and importantly at most one single clique may contain the new label. We can compute:

$$\alpha(q_s, q_v) + 1 - 1 \geq 2^{rq_S} q_v + 2^{rq_S} \geq \alpha(q_s, q_v - 1) + 1.$$

The argument follows from the induction assumption by the same reasoning as in the previous case. □

From this, we can derive a model checking algorithm.

Proposition 5.9 (Model checking on graphs of bounded twin cover)**.** *Let G be a graph with twin cover K of size k and the size of the maximum clique in $G \setminus K$ bounded by $2^{q_S} q_v$ and φ is an MSO_1 sentence then either there exists a clique $C \in G \setminus K$ such that $G \models \varphi$ if and only if $G \setminus C \models \varphi$ or the size of G is bounded by*

$$k + (q_v + 1)q_v^2 2^{k+2q_S+2^{q_S}q_S q_v} = 2^{\mathcal{O}(k+2^{q_S}q_S q_v)}.$$

Proof. There are k vertices in the cover and $2^k r$ types of cliques and each of them (by Lemma 5.8) is repeated at most $\alpha(q_S, q_v) = 2^{rq_S}(q_v + 1)$ otherwise one clique of that type cannot be distinguished by formula φ. The maximal size of the clique is $r = 2^{q_S} q_v$ from Proposition 5.7 and this gives us the desired bound.

$$k + 2^k r^2 \alpha(q_S, q_v) = k + 2^k r^2 2^{rq_S}(q_v + 1) = k + 2^k (2^{q_S} q_v)^2 2^{2^{q_S} q_v q_S}(q_v + 1) =$$

$$= \quad k + (q_v + 1)q_v^2 2^{k+2q_S+2^{q_S}q_v q_S}.$$ □

5.2.2 Finding a Fair Solution

In the upcoming proof, we follow the ideas of Masařík and Toufar [122]. They define, for a given formula $\varphi(X)$, a so-called shape of a set $W \subseteq V$ in G. The idea behind a shape is that in order to do the model checking we have deleted some vertices from G that cannot change the outcome of $\varphi(X)$, however, we have to derive a solution of minimal cost in the whole graph G. Thus the shape characterizes a set under which $\varphi(X)$ holds and we have to be able to find a set $W \subseteq V(G)$ for which $\varphi(W)$ holds and W minimizes the fair cost among sets having this shape.

Shape. Let $G = (V, E)$ be a graph, $\varphi(X)$ an MSO formula, $K \subseteq V$ a twin cover of G, $A \subseteq K$, and let $r = 2^{q_S+2} q_v$ and $\alpha = 2^{r(q_S+1)}(q_v + 1)$. Let $\mathcal{C}$ be the collection of all cliques in G such that A is their cover set. We define an A-*shape*. An A-shape of size r is a two dimensional table S_A of dimension $(r + 2) \times (r + 2)$ indexed by $\{0, 1, \ldots, r + 1\} \times \{0, 1, \ldots, r + 1\}$. Each entry $S_A(i, j) \in \{0, \ldots, \alpha + 1\}$. The interpretation of $S_A(i, j)$ is the minimum of $\alpha + 1$ and of the number of cliques C with $N(C) = A$ such that

$$\min(\alpha + 1, |C \cap W|) = i \text{ and } \min(\alpha + 1, |C \setminus W|) = j.$$

Finally, the shape of X in G is a collection of A-shapes for all $A \subseteq K$.

A solution for $\mathcal{C}$ with cover set A can be formally described by a function $\mathrm{sol}\colon \mathcal{C} \to \mathbb{N} \times \mathbb{N}$. The solution sol is *valid* if for every $C \in \mathcal{C}$ with $\mathrm{sol}(C) = (i, j)$ either $i + j = |C|$ or $|C| \geq r$, $i = r + 1$ (or equivalently $j = r + 1$), and $i + j < |C|$. For an illustration of a valid assignment please refer to Figure 5.2. We say that a valid solution sol is *compatible* with the shape S_A if $S(i, j) = \left|\mathrm{sol}^{-1}(i, j)\right|$, whenever $S(i, j) \leq \alpha$ and $\left|\mathrm{sol}^{-1}(i, j)\right| \geq \alpha + 1$ if $S(i, j) = \alpha + 1$. The A-shape S_A is said to be valid if there exists a valid solution for S_A. Note that such a solution does not exist if the shape specifies too many (or too few) cliques of certain sizes. The shape S is *valid* if all its A-shapes are valid.

The following lemma is a key observation about shapes.

Lemma 5.10. *Let φ be an MSO_1 formula with one free variable, G a graph and W, W' two subsets of vertices having the same shape. Then $G \models \varphi(W)$ if and only if $G \models \varphi(W')$.*

Proof. The proof follows using Proposition 5.7 and Lemma 5.8. Indeed, if we take the graph G with one label corresponding to set W and apply the reduction rules given by Proposition 5.7 and Lemma 5.8 and repeat the same process with W', we obtain two isomorphic labeled graphs. $\square$

Lemma 5.10 allows us to say that a formula with one free variable holds under a shape since it is irrelevant which subset of vertices of this particular shape is assigned to the free variable. Also note that deciding whether the formula holds under the shape can be done in FPT time by simply picking arbitrary assignment of the given shape and running a model checking algorithm. Lemma 5.12 computes a solution of minimal cost for an A-shape. We do this by reducing the task to integer linear programming (ILP) while using non-linear objective. A fuction $f\colon \mathbb{R}^p \to \mathbb{R}$ is *separable convex* if there exist convex functions $f_i\colon \mathbb{R} \to \mathbb{R}$ for $i \in [p]$ such that $f(x_1, \ldots, x_p) = \sum_{i=1}^{p} f_i(x_i)$.

Theorem 5.11 ([127] – simplified)**.** *Integer linear programming with objective minimization of a separable convex function in dimension p is* FPT *with respect to p and space exponential in L the length of encoding of the ILP instance.*

Lemma 5.12. *Let $G = (V, E)$ be a graph, K be a twin cover of G, and $\emptyset \neq A \subseteq K$. There is an algorithm that given an A-shape S_A of size r computes a valid solution for S_A of minimal cost in time $f(|K|, r) \cdot |G|^{O(1)}$ or reports that S_A is not valid.*

Proof. Let $\mathcal{C}$ be the collection of all cliques such that A is their cover set. We split the task of finding a minimal solution to S_A into two independent parts depending on the size of cliques assigned in the phase.

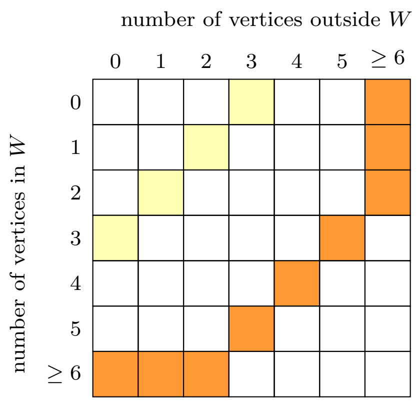

Figure 5.2: Example of a 7×7 A-shape. All cliques of size 3 will be assigned to yellow (light gray) fields, while cliques of size 8 will be assigned to orange (darker gray) fields.

number of vertices outside W

number of vertices in W	0	1	2	3	4	5	≥ 6
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
≥ 6	?	?	?	?	?	?	6

Figure 5.3: An example of uncertainty in computation of objective function. The value in the last row depends on the size of the clique we are assigning to those cells. The value in the cell is how much we pay for any compatible clique assigned to this cell.

The first phase is for cliques in $\mathcal{C}$ with sizes at most r. Observe that these can be assigned deterministically in a greedy way. This is because no cell of S_A is shared by two sizes and we can see that if there are more cells with value α on the corresponding diagonal we can always prefer the top one as this minimizes the cost (see Figure 5.3). However, this is not possible for larger cliques as they may in general share some cells of S_A and thus we defer them to the second phase.

Now observe that the most important vertices for computing the cost are the vertices constituting the set A. To see this just note that all other vertices see only A and their neighborhood (a clique) which is at least as large as for the vertices in A. It follows that we should only care about the number of selected vertices such that A is their cover set. Thus if the size of all cliques in $\mathcal{C}$ is bounded in terms of k we are done. Alas, this is not the case.

We split the set $\mathcal{C}$ into sets $\mathcal{C}_1, \ldots, \mathcal{C}_{2r}$, and $\mathcal{C}_{\max}$. A clique $C \in \mathcal{C}$ belongs to $\mathcal{C}_{|C|}$ if $1 \leq |C| \leq 2r$ and belongs to $\mathcal{C}_{\max}$ otherwise. Note that cliques from $\mathcal{C}_{\max}$ may be assigned only to cells having at least one index $r+1$. As mentioned we are about to design an ILP with a non-linear objective function. This ILP has variables $x^q_{i,j}$ that express the number of cliques from the set $\mathcal{C}_q$ assigned to the cell (i,j) of S_A (that is $1 \leq i,j \leq r+1$ and $q \in Q = \{1, \ldots, 2r\} \cup \{\max\}$). The obvious conditions are the following (the $\trianglerighteq$ symbol translates to $=$ if $S(i,j) \leq \alpha$ while it translates to $\geq$ if $S(i,j) = \alpha + 1$).

$$\sum_{q \in Q} x^q_{i,j} \trianglerighteq S_A(i,j) \qquad 0 \leq i,j \leq r+1$$

$$\sum_{0 \leq i,j \leq r+1} x^q_{i,j} = |\mathcal{C}_q| \qquad \forall q \in Q$$

$$x^q_{i,j} \geq 0 \qquad 0 \leq i,j \leq r+1,\ \forall q \in Q$$

We are about to minimize the following objective

$$\sum_{0\le i\le r+1; 0\le j\le r} \sum_{1\le q\le 2r} (q-j)x^q_{i,j} + \sum_{0\le i\le r+1} \sum_{\forall q} i\cdot x^q_{i,r+1} + g\left(x^{\max}_{r+1,0},\dots,x^{\max}_{r+1,r}\right) ,$$

where $g\colon \mathbb{N}^r \to \mathbb{N}$ is a function that has access to sizes of all cliques in $\mathcal{C}_{\max}$ and computes the minimum possible assignment. We claim that the function g is a separable convex function in variables $x^{\max}_{r+1,0},\dots,x^{\max}_{r+1,r}$. The first summand of the objective function describes the cliques of size at most $2r$. Their price corresponds to the number of vertices in the clique q minus the number of vertices that are not selected j. The second summand corresponds to the last row, where the cheapest price is always the number of selected vertices i. The last summand, discussed in the following paragraph, describes the assignment to the last row. The result then follows from Theorem 5.11 as the number of integral variables is $O(r^3)$.

Observe that the value of $g\left(x^{\max}_{r+1,0},\dots,x^{\max}_{r+1,r}\right)$ is equal to sum of sizes of cliques "assigned to the last row" minus $\sum_{j=0}^r j\cdot x^{\max}_{r+1,j}$. Now, $g\left(x^{\max}_{r+1,0},\dots,x^{\max}_{r+1,r}\right) = g'\left(\sum_{j=0}^r x^{\max}_{r+1,j}\right) - \sum_{j=0}^r j\cdot x^{\max}_{r+1,j}$. Since all cliques in $\mathcal{C}_{\max}$ are eligible candidates to be assigned to the last row and since it is always cheaper to assign there those of the smallest size among them we can define g' based only on the number of cliques that are assigned to the last row. This finishes the proof since g' is a convex function. See Section 5.2.2 for details on polynomial space version. □

Now we are ready to prove the main result of this section. It essentially follows by the exhaustive search among all possible shapes S such that φ is true under S and application of Lemma 5.12.

Proposition 5.13. *Let $G=(V,E)$ be a graph with twin cover K of size k. For an MSO_1 formula $\varphi(X)$ with one free variable that represents the set to be deleted it is possible to find a set $W\subseteq V$ such that*

- *$\varphi(W)$ holds in G and*
- *the cost of W is minimized among all subset of V satisfying $\varphi(X)$*

in time $f(k,|\varphi|)|V|^{O(1)}$ for some computable function f.

Proof. We proceed as follows. For every possible selection of $K\cap W$ we generate all possible shapes and check whether $\varphi(X)$ evaluates to true under shape S and if so, we compute W for S having the minimal fair-cost. Finally, we return the set X minimizing the cost.

We stress that Lemma 5.8 applies only in cases where the cover set of the cliques at hand is not empty. Thus, these cliques are exceptional and we have to handle them separately. Note that in this case, the objective is different—we want to minimize the maximum number of selected vertices in any clique. It is not hard, however, to do this via standard tricks.

In order to summarize the running time we have

- at most 2^k possible selections for $W\cap K$ and
- at most $2^k\cdot(\alpha+1)^{(r+1)}$ possible shapes (for each such selection).

For every shape S we label the graph according to S. Afterwards, apply Lemma 5.8 and proposition 5.7 exhaustively on the labeled graph to resolve whether $\varphi(X)$ evaluates to true under S. If $\varphi(X)$ was evaluated to true, then using Lemma 5.12 on every possible A we obtain a set W_A minimizing the cost for vertices in A and put $W = \cup_{A \subseteq K} W_A$. Observe that this union gives the optimal cost for the selected shape S. Finally, we return the set W minimizing the cost for any true evaluated shape. Clearly, this routine runs in FPT time with respect to k and $|\varphi|$ as parameters. $\square$

Polynomial Space Version of Proposition 5.13

We now argue that it is possible to implement Lemma 5.12 in polynomial space via reducing it to integer linear programming. We note that similar application of this technique is recently presented by Bredereck et al. [16].

Theorem 5.14 (Lenstra & Frank, Tardós [114, 63])**.** *There is an algorithm that given an ILP with p variables finds an optimal solution to it using $O(p^{2.5p}\operatorname{poly}(L))$ arithmetic operations and space polynomial in L, where L is the bitsize of the ILP.*

Remark 5.15. *Let $G = (V, E)$ be a graph, K be a twin cover of G, and $A \subseteq K$. There is an algorithm that given an A-shape S_A of size r computes a valid solution for S_A of minimal cost in time $f(|K|, r) \cdot |G|^{O(1)}$ and space polynomial in $|G|$ or reports that S_A is not valid.*

Proof. Here the only difference is that we have to rewrite the function

$$g(x^{\max}_{r+1,1}, \dots, x^{\max}_{r+1,r})$$

using a new variable y (representing its value) into constraints of ILP and thus obtain a linear objective.

We do this by adding a variable $y \geq 0$ with constraint $y = \sum_{j=1}^{r} x^{\max}_{r+1,j}$ and work with univariate function $g(y)$ instead. We order cliques in $\mathcal{C}_{\max}$ according to their size, that is, $|C_1| \leq |C_2| \leq \cdots \leq |C_t|$, where t denotes the size of $\mathcal{C}_{\max}$. By c_i we denote the sum $\sum_{j=1}^{i} |C_i|$ and note that $g(y) = c_y$. Finally we introduce a variable g_y representing the value of $g(y)$ and add constraints

$$g_y \geq (y - i)c_i \qquad \forall 1 \leq i \leq t\,.$$

Our result then follows as $t \leq |V|$ and thus we add at most $|V|$ new constraints. $\square$

5.3 The Fair VC Problem

5.3.1 Hardness for Treedepth and Feedback Vertex Set

We begin with several simple observations about the fair objective value k in the Fair VC problem.

Observation 5.16. *Let $G = (V, E)$ be a graph and $U \subseteq V$ be a vertex cover of G with fair objective k, that is, $\forall v \in V$ it holds that $|N(v) \cap U| \leq k$. If $v \in V$ has* $\deg(v) \geq k$*, then $v \in U$.*

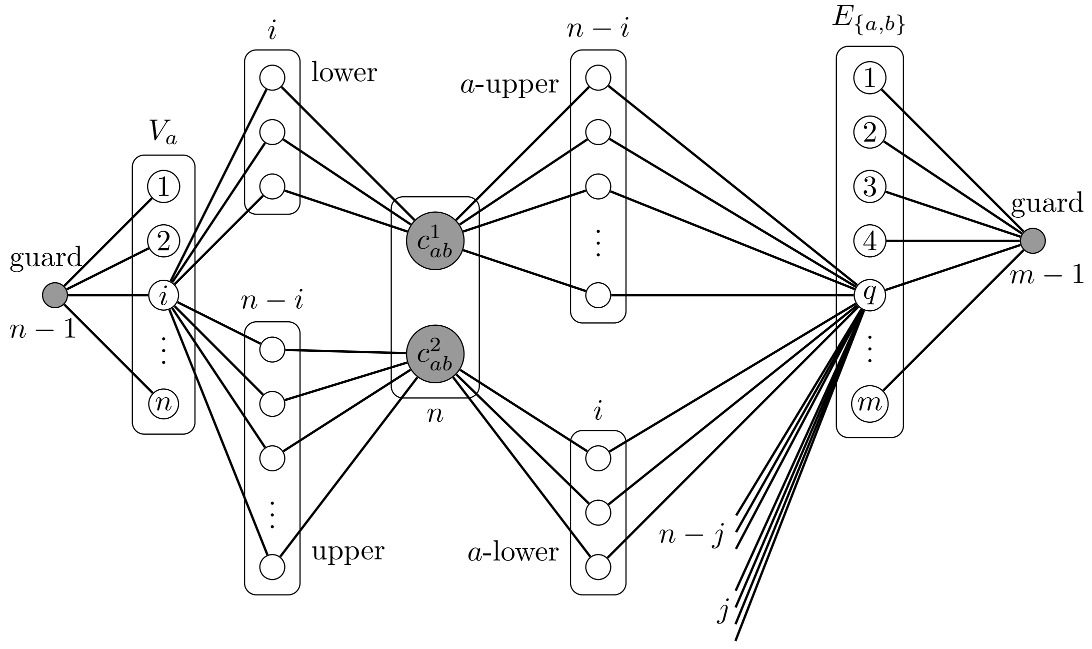

Figure 5.4: An overview of the reduction in the proof of Theorem 5.2. The gray vertices are enforced to be a part the fair vertex cover. If a vertex fair objective was lowered, then the resulting threshold is beneath the vertex (the group of vertices).

Note that we can use Observation 5.16 to enforce a vertex v to be a part of the fair vertex cover by attaching $k+1$ degree 1 vertices to v. Observe further that we may adjust (lower) the global budget k for individual vertex v by attaching vertices to v and then attaching k new leaves to the newly added vertices. To this end if the above operations are applied to a graph G resulting in a graph G', then $\mathsf{td}(G') \leq \mathsf{td}(G) + 2$ and $\mathsf{fvs}(G') = \mathsf{fvs}(G)$.

We observe substantial connection between Fair VC and Target Set Selection (TSS). It is worth mentioning that Vertex Cover can be formulated in the language of TSS by setting the threshold to $\deg(v)$ for every vertex v. As a result, our reduction given here is, in certain sense, dual to the one given by Chopin et al. [24] for the TSS problem. However, we will show that the structure of the solution for Fair VC is, in fact, the complement of the structure of the solution for TSS given therein. The archetypal $\mathsf{W[1]}$-hard problem is the ℓ-Multicolored Clique problem [41]:

ℓ-Multicolored Clique	*Parameter: ℓ*
Instance:	An ℓ-partite graph $G = (V_1 \cup \cdots \cup V_\ell, E)$, where V_c is an independent set for every $c \in [\ell]$ and they are pairwise disjoint.
Question:	Is there a clique of the size ℓ in G?

Proof of Theorem 5.2. Let $G = (V_1 \cup \cdots \cup V_\ell, E)$ be an instance of the ℓ-Multicolored Clique problem and let $n = |V_i|$ for all $i \in [\ell]$. We denote by $E_{\{a,b\}}$ the set of edges between V_a and V_b and by $m = |E_{\{a,b\}}|$. We will describe graph

$H = (W, F)$ that together with $k = \max(m-1, 2n)$ will form an equivalent instance of the Fair VC problem. The reduction has the following properties:

- $|W| = \text{poly}(n, k)$ and $|F| = \text{poly}(m, k)$,
- $\mathsf{td}(H) = \mathcal{O}(\ell^2)$, and $\mathsf{fvs}(H) = \mathcal{O}(\ell^2)$.

For an overview of the reduction please refer to Figure 5.4. There are three types of gadgets in our reduction, namely the vertex selection gadget (one for each vertex), the edge selection gadget (one for each edge), and the incidence check gadget (one for each vertex–edge incidence). We start by enumerating the vertices in each color class by numbers from $[n]$ and edges by numbers in $[m]$. Throughout the proof a, b are distinct numbers from $[\ell]$.

The V_a selection gadget consists of n *choice* vertices (representing the color class V_a), a special vertex called *guard*, and a group of n^2 *enumeration* vertices. The guard vertex is connected to all choice vertices, it is enforced to be a part of the fair vertex cover, and its budget is lowered so that at most $n-1$ choice vertices can be in any fair vertex cover. The i-th choice vertex is connected to n enumeration vertices. For each choice vertex, there are n such vertices and so these are private for vertex i. We further divide these vertices into two parts – the *lower part* consists of i vertices and the *upper part* consists of $n-i$ vertices.

The $E_{\{a,b\}}$ selection gadget consists of m *choice* vertices, a special vertex called *guard*, and a group of $2nm$ *enumeration* vertices and is constructed analogously to the vertex selection gadget. If the q-th edge of $E_{\{a,b\}}$ connects i-th vertex in V_a and j-th vertex in V_b, there are (private) $2n$ numeration vertices are connected to the q-th choice vertex. These are split into four groups – *lower a-part* consisting of i vertices, *upper a-part* consisting of $n-i$ vertices, and similarly *lower* and *upper b-parts*.

The ab-incidence check gadget consist of two vertices c^1_{ab} and c^2_{ab}. Both c^1_{ab} and c^2_{ab} are enforced to be a part of the solution and with a lowered budget in a way that at most n vertices in the neighborhood of each of them can be part of any fair vertex cover. The vertex c^1_{ab} is connected to every lower part vertex in the selection gadget for V_a and to every upper a-part vertex in the selection gadget for $E_{\{a,b\}}$. The vertex c^2_{ab} is connected to every upper part vertex in the selection gadget for V_a and to every lower a-part vertex in the selection gadget for $E_{\{a,b\}}$.

This finishes the construction of H. Now observe that if we remove vertices c^1_{ab}, c^2_{ab} from H, then each component of the resulting graph is a tree (rooted in its guard vertex) of depth at most 3. It follows that $\mathsf{td}(H) = \mathcal{O}(\ell^2)$ and $\mathsf{fvs}(G) = \mathcal{O}(\ell^2)$, as so is the size of the removed set of vertices. We finish the proof by showing that the two instances are equivalent.

Suppose (G, ℓ) is a yes-instance which is witnessed by a set $K \subseteq V_1 \times \cdots \times V_\ell$. We now construct a vertex cover C_K of H having $|N(w) \cap C_K| \leq k$ for all $w \in W$. The set C_K contains the following:

- all enforced vertices (including all guard and check vertices),
- if $v \in V_a \cap K$ is the i-th vertex of V_a, then all selection vertices of V_a but the vertex i are in C_K and lower and upper enumeration vertices of i are in C_K,
- if $v \in V_a \cap K$ and $u \in V_b \cap K$ are adjacent through q-th edge of $E_{\{a,b\}}$, then all selection vertices of $E_{\{a,b\}}$ but the vertex q are in C_K and q's enumeration vertices are in C_K.

For the other direction we prove that a vertex cover C in H fulfils $|N(w) \cap C| \leq k$ for all $w \in W$ if it corresponds to a clique in G. For the other direction suppose that there is a vertex cover C in H such that $|N(w) \cap C| \leq k$ for all $w \in W$. Recall that C has to contain all enforced vertices. This implies that at least 1 choice vertex for V_a is not in U; we will show that exactly 1 such choice vertex is in U. The same holds for the edge choice vertices. To see this suppose for contradiction that 2 choice vertices for V_a (vertex i and j) are not in U. Because U is a vertex cover of H it follows that their enumeration vertices must belong to U. But now take vertices c^1_{ab}, c^2_{ab}. In their neighborhood U have at least $3n$ vertices ($2n$ from the V_a's numeration part and n from the a-numeration part of $E_{\{a,b\}}$). This is absurd as these vertices (due to the lowered budget) can have at most $2n$ vertices in their neighborhood and thus at least one of them exceeds its budget. Thus the selection gadgets actually encode some selection of vertices v_a and edges $e_{a,b}$. To finish the proof we have to observe that both c^1_{ab} and c^2_{ab} have at most (in fact, exactly) n neighbors in U if and only if the vertex v_a is incident to the edge $e_{a,b}$. If this holds for all possible combinations of a, b, then we have selected a clique in the graph G.

It remains to discuss the ETH based lower-bound. This follows straightforwardly from our reduction and the result of Chen et al. [23] who proved that there is no $f(k)n^{o(\ell)}$ algorithm for ℓ-Multicolored Clique unless ETH fails. Since we have $\mathsf{td}(G) + \mathsf{fvs}(G) = O(\ell^2)$ in our reduction, the lower-bound follows. $\square$

5.3.2 FPT Algorithm for Modular Width

Since the algebraic expression A of width $\mathsf{mw}(G)$ can be computed in linear time [137], we can assume that we have A on the input. We construct the rooted ordered tree $\mathcal{T}$ corresponding to A. Each node $t \in \mathcal{T}$ is assigned a graph $G^t \subseteq G$, that is, the graph constructed by the subexpression of A rooted at t. Suppose we are performing substitution operation at node t with respect to template graph T and graphs $G_1, \ldots, G_r$. Denote the resulting graph G^t and denote by n_i the size of $V(G_i)$.

Proof of Theorem 5.3. The edges in G^t between two vertices of G_i will be referred to as *old edges*, the edges between G_i and G_j for $i \neq j$ (i.e., edges introduced by the template operation) will be referred to as *new edges*.

The computation will be carried out recursively from the bottom of the tree $\mathcal{T}$.

We first describe the structure of all vertex covers C in G^t. Observe that if $ij \in E(T)$ then at least one of $V(G_i), V(G_j)$ must be a subset of C; otherwise there would be a new edge not covered by C. From this we can see that the set $C_T := \{i : V(G_i) \subseteq C\}$ is a vertex cover of the template graph T. We call the C_T the *type of the vertex cover* C. Furthermore, every set $C \cap V(G_i)$ must be a vertex cover of G_i (otherwise there would be an old edge uncovered by C).

We now describe the fair cost of the cover C in terms of fair costs and sizes of the sets $C \cap V(G_i)$. Denote by c_i the size $|C \cap V(G_i)|$ and by f_i the fair cost of $C \cap V(G_i)$ in G_i. The *fair cost of* C *in* $W \subseteq V(G)$ is defined as $\max_{v \in W} |C \cap N(v)|$. For $i \in [r]$ the fair cost of C in $V(G_i)$ can be written as $f_i + \sum_{j : ij \in E(T)} c_j$. Clearly, fair cost of C is the maximum of the last expression over $i \in [r]$.

If we know the type C_T of the cover C this can be simplified based on whether i lies in C_T. If $i \in C_T$ then f_i is $\Delta(G_i)$ (the maximal degree of G_i). If on the other hand $i \notin C_T$ then all its neighbors j are in C_T and in this case $c_j = n_j$. Combining those observations, we have

$$\text{fair cost of } C \text{ in } G_i = \begin{cases} \Delta(G_i) + \sum_{j \notin C_T : ij \in E(T)} c_j + \sum_{j \in C_T : ij \in E(T)} n_j & i \in C_T, \\ f_i + \sum_{j : ij \in E(T)} n_j & i \notin C_T. \end{cases}$$

For each node t of the tree $\mathcal{T}$ we keep an $|V(G^t)|$ table Tab^t of integer values from $[n] \cup \infty$. The value at position $\mathrm{Tab}^t[p]$ is the smallest size of a cover in G^t of fair cost p or ∞ if such cover do not exists.

The computation of Tab in leaves of $\mathcal{T}$ is trivial. We describe how to compute value $\mathrm{Tab}^t[p]$ given that we know Tab_i in all children of t. It is enough to determine whether there exists a vertex cover C of G^t of fixed type; we can simply iterate over all types as there are at most 2^r of them.

Fix a type C_T. The cover C of type C_T and fair cost at most p and $\mathrm{Tab}^t[p] \neq \infty$ exists if and only if for every $i \notin C_T$ there is a vertex cover C_i of G_i of fair cost p_i such that $\mathrm{Tab}_i[p_i] \neq \infty$. Moreover, we require that the values p_i satisfy the following inequalities:

$$p \geq \Delta(G_i) + \sum_{j \notin C_T : ij \in E(T)} \mathrm{Tab}_j[p_i] + \sum_{j \in C_T : ij \in E(T)} n_j \qquad \forall i \in C_T, \tag{5.1}$$

$$p \geq p_i + \sum_{j : ij \in E(T)} n_j \qquad \forall i \notin C_T, \tag{5.2}$$

$$\mathrm{Tab}^t[p] \geq \sum_{j \notin C_T} \mathrm{Tab}_j[p_i] + \sum_{j \in C_T} n_j. \tag{5.3}$$

First, for every $i \notin C_T$ we set p_i to the highest possible value without violating the inequality (5.2), that is $p_i := p - \sum_{j : ij \in E(T)} n_j$. Note that this is always a safe choice; p_i does not appear anywhere else in the constraints and choosing the highest possible value to give us the greatest freedom due to the monotonicity of the table. Clearly, if any such p_i is negative we know that given constraints cannot be satisfied and there is no vertex cover C of a given type and fair cost.

If for any i $\mathrm{Tab}_i[p_i] = \infty$, then there is no vertex cover C_i in G_i of fair cost p_i. This means that we cannot find cover C of a given type and fair cost so we set $\mathrm{Tab}_i[p_1] = \infty$. We check whether inequalities (5.1) holds. If not, set $\mathrm{Tab}_i[p_1] = \infty$. Otherwise we set $\mathrm{Tab}^t[p]$ be equal to the expression in (5.3) on the right side. We claim that there is a vertex cover C of a given type and fair cost; we can set $C = \bigcup_{i \in C_T} V(G_i) \cup \bigcup_{i \notin C_T} C_i$, where C_i is any vertex cover of G_i of fair cost p_i and size $\mathrm{Tab}_i[p_i]$ (this is guaranteed to exist because $\mathrm{Tab}_i[p_i]$ was not ∞. It is straightforward to check that C has required properties. Moreover, from our choice of values p_i it follows that a vertex cover of type C_T, fair cost p and size $\mathrm{Tab}^t[p]$ exists if and only if the described procedure finds values of p_i. By iterating over all types C_T we can fill the value $\mathrm{Tab}[p]$ as required.

To complete the description of the algorithm, it is enough to look whether there is not ∞ value in $\mathrm{Tab}_{\mathrm{root}}[k]$, where k was the desired fair cost.

The running time is n for the induction over expression A times 2^r different type of covers in any single node times filling the table of size at most n times nr for determining the correct values p_i and checking other inequalities for every $i \in [r]$. This altogether yields a $2^r rn^3$ time algorithm. $\square$

Proof of Lemma 5.4. First we observe that modular width is trivially compositional, that is, for any two graphs G_1, G_2 it holds that

$$\mathsf{mw}(G_1 \dot\cup G_2) = \max(\mathsf{mw}(G_1), \mathsf{mw}(G_2))$$

, where $\dot\cup$ denotes the disjoint union. Indeed this follows from the fact that disjoin union is one of the operations not affecting modular width. Now, it remains to show that Fair VC is AND-compositional, see [41, Chapter 15]; the rest then follows from the framework of Bodlaender et al. [11]. To this end, observe that if a graph G is not connected, then U is a vertex cover in G if and only if $U \cap C$ is a vertex cover in $G[C]$ for every connected component C of G. $\square$

5.4 Hardness of Possible Extensions

We use the Unary ℓ-Bin Packing problem as the starting point of our hardness reduction. Unary ℓ-Bin Packing is $\mathsf{W}[1]$-hard for parameter ℓ the number of bins to be used [92]. Here, the item sizes are encoded in unary and the task is to assign n items to ℓ bins such that the sum of sizes of items assigned to any bin does not exceed its capacity B. Formally, Unary ℓ-Bin Packing is defined as follows.

Unary ℓ-Bin Packing	*Parameter:* ℓ
Instance:	Positive integers ℓ and B and item sizes $s_1, \ldots, s_n$ encoded in unary.
Question:	Is there a packing of all items into at most ℓ bins of size B?

Proof of Theorem 5.5. We construct a formula $\varphi(X_1, \ldots, X_\ell)$ and an instance (G, k) of Fair Vertex MSO Evaluation with $k = B$ from an instance of Unary ℓ-Bin Packing as follows. The graph G is formed by n disjoint cliques and a universal vertex u. Cliques in G represent the items by their respective sizes, that is, there is a clique with s_i vertices for every $i \in [n]$; denote the clique representing item i by C_i. This finishes the description of the graph; now we turn our attention to the formula. Free variables $X_1, \ldots, X_\ell$ are going to represent an assignment of items to bins. Note that it is possible to recognize the universal vertex u by the following FO formula, since u is the only vertex of G satisfying it:

$$\mathrm{univ}(v) := (\forall w \in V)\Big((w \neq v) \to (wv \in E)\Big).$$

We fix u for the rest of the description of $\varphi(X_1, \ldots, X_\ell)$; this can easily be done by attaching $(\exists u \in V)(\mathrm{univ}(u))$ to it. Let $\mathrm{p}(v)$ be a predicate. For $Q \in \{\exists, \forall\}$ we use the following $Qv \in (V \setminus \{u\})(\psi(v))$ as a short form of the expression

$$(Qv \in V)\Big((v \neq u) \to \psi(v)\Big).$$

Note that this can be straightforwardly extended for more quantifiers.

In order to represent the bin choice, we need to ensure two conditions. First, every item is packed, that is, every non-universal vertex must belong to some X_j. Second, every item is fully packed inside (at least) one bin, that is, vertices belonging to the same clique agree on X_j membership. To do so we first introduce the following predicates (representing these conditions):

$$\mathrm{cover}(X_1,\dots,X_\ell) :=(\forall v \in V \setminus \{u\}) \left(\bigvee_{j=1}^{\ell} v \in X_j\right)$$

and

$$\mathrm{same}(X_1,\dots,X_\ell) := (\forall v, w \in V \setminus \{u\}) \left((vw \in E) \to \bigwedge_{j=1}^{\ell} (v \in X_j \Leftrightarrow w \in X_j)\right).$$

The construction of the new instance is finished by letting

$$\varphi(X_1,\dots,X_\ell) = (\exists u \in V)(\mathrm{univ}(u)) \land \mathrm{cover}(X_1,\dots,X_\ell) \land \mathrm{same}(X_1,\dots,X_\ell).$$

It remains to argue that the instances are indeed equivalent. Note that the bin capacity/fair cost is essentially checked only for u since the fair cost of any vertex cannot exceed its degree. The degree of any other vertex (not u) does not exceed $\max_{i\in[n]} s_i$ and this is always upper-bounded by B.

Let $(B, \ell, s_1, \dots, s_n)$ be a Yes-instance of ℓ-Unary Bin Packing. This is witnessed by a mapping $\sigma\colon [n] \to [\ell]$ assigning items to bins. Now, we put

$$W_j = \cup_{i\in[n]\colon \sigma(i)=j} C_i.$$

Observe that $|N(u) \cap W_j| = \sum_{i\in[n],\sigma(i)=j} s_i \le B$ for every $j \in [\ell]$, since σ represented a valid assignment. Furthermore, $\varphi(W_1,\dots,W_\ell)$ holds, since every clique vertex is covered and vertices of a clique are always in the same W_j.

For the opposite direction assume we have graph $G = (V, E)$ and that there exist sets $W_1,\dots,W_\ell \subseteq V$ such that $\varphi(W_1,\dots,W_\ell)$ holds in G. Recall that u is the universal vertex of G. First, we have that $V = \{u\} \cup W_1 \cup \dots \cup W_\ell$, since the predicate $\mathrm{cover}(W_1,\dots,W_\ell)$ holds if and only if every vertex in a clique C_i is in some W_j. Furthermore, since the predicate $\mathrm{same}(W_1,\dots,W_\ell)$holds, we have that $C_i \cap W_j$ is either an empty set or C_i for every $i \in [n]$ and $j \in [\ell]$. This allows us to construct an assignment $\sigma\colon [n] \to [\ell]$ by setting $\sigma(i) = j$, where $j \in [\ell]$ is the smallest number such that $W_j \cap C_i = C_i$ holds. Now, we have that $\sum_{i\in[n],\sigma(i)=j} s_i \le |W_j| \le B$ by the fair objective. This finishes the proof since we have constructed a valid assignment. □

Proof of Theorem 5.6. We construct a sentence φ, a graph G, and k forming an instance of Fair Edge FO Deletion with $k = B$ from an instance of Unary ℓ-Bin Packing as follows. Each item is represented by clique on $3B$ vertices; denote the clique associated with i-th item by C_i. In addition, there are ℓ vertices $v_1,\dots,v_\ell$ (representing bins) and ℓ guard vertices $g_1,\dots,g_\ell$. For each $j \in [\ell]$ we connect v_j with exactly s_i vertices in C_i (call these vertices *special*) and with g_j. This finishes the description of G.

The sentence φ is constructed using auxiliary predicates which we describe first. A predicate $\mathrm{guard}(v)$ is used to recognize the guard vertices

$$\mathrm{guard}(v) :=(\exists u \in V)((uv \in E) \land (\forall w \in W \setminus \{u, v\})(wv \notin E)).$$

$$\mathrm{bin}(v) :=(\exists u)((uv \in E) \land \mathrm{guard}(u)).$$

$$\text{item-edge} := (\forall v, w \in V : v \neq w)$$
$$\Big((\exists u \in V)(\neg \operatorname{bin}(u) \wedge vu \in E \wedge wu \in E)\Big) \to (vw \in E).$$

Recall that we set the fair cost to B and thus the solution F deletes at most B edges incident to any vertex. Note that a vertex v must be a guard vertex in order to fulfill the guard(v), since the degree of every other vertex in G is at least $3B$. Suppose that the predicate item-edge holds in $G \setminus F$. We claim that $vw \in E$ for any two vertices $v, w \in C_i$, since $|C_i| = 3B$ it follows that v and w have at least $B - 2$ common neighbors in C_i. Thus, the edge vw cannot be deleted, as v and w have a common (non-bin) neighbor (provided $B \geq 3$). We define an auxiliary sentence

$$\psi = (\exists v_1, \dots, v_\ell \in V) \left(\bigwedge_{j \in [\ell]} \operatorname{bin}(v_j) \wedge \bigwedge_{j \neq j' \in [\ell]} v_j \neq v_{j'} \right) \wedge \text{item-edge}$$

and, since ψ implicitly assures existence of ℓ (different) guards, conclude the following claim.

Claim 5.17. *Let F be a set of edges in G. We have that $G \setminus F \models \psi$ if and only if F contains only edges between bins and special vertices.*

The next predicate we introduce is the notable(v), defined as follows

$$\text{notable}(v) := \neg \operatorname{guard}(v) \wedge \Big((\exists u \in V)(\operatorname{bin}(u) \wedge uv \in E)\Big).$$

Claim 5.18.

1. *If $G \setminus F \models \text{notable}(v)$, then v is a special vertex.*
2. *$G \models \text{notable}(v)$ if and only if v is a special vertex.*
3. *If $G \setminus F \not\models \text{notable}(v)$ for a special vertex $v \in C_i$, then $N_{G \setminus F}(v) \subseteq C_i$.*

Proof of Claim. A vertex v is *notable* in a graph if notable(v) holds in that graph. The second part follows immediately from the construction of G, since the only vertices attached to the bin vertex v_j are the special vertices and its guard vertex g_j (which is not notable). On the other hand, the set F may contain edges vv_j for all $j \in [\ell]$. Clearly, such a former special vertex is not notable. Finally, if v is special and $G \setminus F \not\models \text{notable}(v)$, then v lost all of its edges to bin vertices $v_1, \dots, v_\ell$. $\diamondsuit$

The goal is to delete an edge set F that describes a valid assignment of items into bins. We need to ensure two conditions. First, every special vertex is not a neighbor of some bin vertex v_j. Second, if a special vertex $v \in C_i$ is not a neighbor of a bin vertex v_j, then all special special vertices in C_i are not neighbors of v_i. The two conditions correspond to the following two predicates:

$$\text{cover} := (\forall v \in V)(\exists u \in V)\Big(\operatorname{bin}(u) \wedge uv \notin E\Big)$$

$$\text{same} := (\forall v, w \in V)\Bigg((\text{notable}(v) \wedge \text{notable}(w) \wedge vw \in E) \to$$

$$\to (\forall u \in V : \mathrm{bin}(u))(uw \in E \Leftrightarrow vw \in E)\Big)$$

Now, we are ready to give the sentence φ that describes the problem:

$$\varphi = \psi \wedge \text{same} \wedge \text{cover}.$$

This finishes the description of the reduction. We are left with validating that the two instances are equivalent.

Suppose we were given a Yes-instance of ℓ-Unary Bin Packing and let $\sigma\colon [n] \to [\ell]$ be the assignment of items to bins witnessing this fact. The set F contains an edge uv_j for a special vertex $u \in C_i$ and a bin vertex v_j whenever $\sigma(i) = j$. Clearly, we have $|\{e \in F\colon e \ni v_j\}| \le \sum_{i\in[n],\sigma(i)=j} s_i \le B = k$ for every vertex v_j with $j \in [\ell]$, while every other vertex has at most one incident edge in F. Now, we have to verify that $G \setminus F \models \varphi$. We have $G \setminus F \models \text{same} \wedge \text{cover}$, since σ is an assignment. Finally, $G \setminus F \models \psi$, since our F fulfills the condition of Claim 5.17.

Suppose now that we have a set F of edges of G with fair cost B such that $G \setminus F \models \varphi$. By Claim 5.17 we have that F contains only edges between special vertices in G and bin vertices $v_1, \ldots, v_\ell$. We partition the set of special vertices into sets N and R; we put a special vertex v in N if $G \setminus F \models \text{notable}(v)$ and we put it in R otherwise. Note that $|R| \le B$, since a vertex in R contributes to the fair cost of every bin vertex v_j. By Claim 5.18 and the fact that $G \setminus F \models \text{same}$ we have that a bin vertex v_j is either completely attached or non-attached to $C_i \cap N$ for every $j \in [\ell]$ and every $i \in [n]$. Furthermore, there are no edges between a vertex in R and a bin vertex v_j in $G \setminus F$ for every $j \in [\ell]$. We define the assignment $\sigma\colon [n] \to [\ell]$ by defining $\sigma(i)$ to be the smallest integer such that there are no edges between $v_{\sigma(i)}$ and C_i in $G \setminus F$. Since σ is an assignment by $G \setminus F \models \text{cover}$, it remains for verify that the capacity condition is fulfilled. For that we have

$$\sum_{i\in[n],\sigma(i)=j} s_i \le |\{e \in F\colon v_j \in e\}| \le \sum_{i\in[n],\sigma(i)=j} |N \cap C_i| + |R| \le k = B.$$

We conclude that σ is a valid assignment and the theorem follows. $\square$

5.5 Conclusions

Fair Edge L Deletion problems. The crucial open problem is to resolve the parameterized complexity of the Fair FO Edge Deletion problems for parameterization by neighborhood diversity and twin cover. Observe that there is a big difference between vertex and edge deletion problems—in our hardness reduction we use a deletion to an edgeless graph but a fair edge cost, in this case, equals to the maximum degree of the former graph (and thus it is computable in polynomial time).

Generalization of parameters. Another open problem is to resolve the parameterized complexity of the Fair MSO_1 Vertex Evaluation problems with respect to graph parameters generalizing neighborhood diversity or twin cover (e.g., modular width or cluster vertex deletion number respectively).

Cluster Vertex Deletion Number. It seems that a more careful analysis of the model-checking algorithm may yield (again) sufficient insight into the structure of the fair solution and thus lead to an FPT algorithm for this wider class of graphs. Though, it remains open whether this is possible and there is an FPT algorithm for FAIR MSO VERTEX EVALUATION for this parameterization or not.

MSO with Local Linear Constraints. Previously, an FPT algorithm for evaluation of a fair objective was given for parameter neighborhood diversity [122]. That algorithm was extended [100] to a so-called *local linear constraints* again for a formula $\varphi(\cdot)$ with one free variable that is defined as follows. Every vertex $v \in V(G)$ is accompanied with two positive integers $\ell(v), u(v)$, the lower and the upper bound, and the task is to find a set X that not only $G \models \varphi(X)$ but for each $v \in V(G)$ it holds that $\ell(v) \le |N(v) \cap X| \le u(v)$. Note that this is a generalization as fair objective of value t can be tested in this framework by setting $\ell(v) = 0$ and $u(v) = t$ for every $v \in V(G)$. Is this extension possible for parameterization by the twin cover number?

To support this question we note that in the proof of Lemma 5.12 the minimal size of the neighborhood of B for a shape in the exact neighborhood of B is computed. It is not hard to see that through a similar argument we can compute the maximal size of the neighborhood of B for a shape in the exact neighborhood of B. Furthermore, lower and upper bounds for vertices in a clique can be assumed to be nearly the same—each differs by at most 1 [100]. Thus, Lemma 5.12 gives only that if at least one of the computed bounds for a vertex v in the twin cover is within $\ell(v)$ and $u(v)$, then there is a solution with desired properties. However, if on the other hand, it happens that both $\ell(v), u(v)$ are in between the computed values, we do not know whether or not any of the desired values are attainable. To see that not all values in the thus computed range are attainable one can construct a formula that for twin cliques up to a certain size check that the number of selected vertices is even. Then, if the input graph contains only cliques up to this size no twin cover vertex has an odd number of neighbors in the set X (provided the cover vertices form an independent set).

Towards new fair problems. As we proposed the examination of FAIR VC already, we would like to turn an attention to exploring fair versions of other classical and well-studied VERTEX DELETION problems. In contrast, certain FAIR EDGE DELETION problems have got some attention before, namely FAIR FEEDBACK EDGE SET [117] or FAIR EDGE ODD CYCLE TRANSVERSAL [104]. Besides FAIR VC we propose a study of FAIR DOMINATING SET and FAIR FEEDBACK VERTEX SET. In particular, it would be really interesting to know whether fair variants of VERTEX COVER and DOMINATING SET admit a similar behavior as in the classical setting.

Furthermore, We would like to ask whether there is an NP-hard Fair Vertex Deletion problem that admits an FPT algorithm for parameterization by treedepth (and feedback vertex set) of the input graph.

Bibliography

[1] Karl R. Abrahamson, Rodney G. Downey, and Michael R. Fellows. Fixed-parameter tractability and completeness IV: on completeness for W[P] and PSPACE analogues. *Annals of Pure and Applied Logic*, 73(3):235–276, 1995. doi:10.1016/0168-0072(94)00034-Z.

[2] Karl R. Abrahamson, John A. Ellis, Michael R. Fellows, and Manuel E. Mata. On the complexity of fixed parameter problems (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October–1 November 1989*, pages 210–215, 1989. doi:10.1109/SFCS.1989.63480.

[3] Tadashi Ae, Akira Nakamura, and Toshimasa Watanabe. On the NP-hardness of edge-deletion and -contraction problems. *Discrete Applied Mathematics*, 6(1):63–78, 1983. doi:10.1016/0166-218X(83)90101-4.

[4] Noga Alon. Restricted colorings of graphs. In K. Walker, editor, *Surveys in Combinatorics*, pages 1–34. Cambridge University Press, 1993. doi:10.1017/cbo9780511662089.002.

[5] Stefan Arnborg, Derek Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987. doi:10.1137/0608024.

[6] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, June 1991. doi:10.1016/0196-6774(91)90006-k.

[7] József Balogh, Alexandr Kostochka, and Xujun Liu. Packing chromatic number of subcubic graphs, 2017. arXiv:1703.09873.

[8] Rémy Belmonte, Michael Lampis, and Valia Mitsou. Parameterized (approximate) defective coloring. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, pages 10:1–10:15, 2018. doi:10.4230/LIPIcs.STACS.2018.10.

[9] Umberto Bertelè and Francesco Brioschi. On non-serial dynamic programming. *Journal of Combinatorial Theory, Series A*, 14(2):137–148, 1973. doi:10.1016/0097-3165(73)90016-2.

[10] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.

[11] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009. doi:10.1016/j.jcss.2009.04.001.

[12] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 2013. doi:10.1007/978-1-4612-0619-4.

[13] Marthe Bonamy, Konrad K. Dabrowski, Carl Feghali, Matthew Johnson, and Daniël Paulusma. Independent feedback vertex set for P_5-free graphs. *Algorithmica*, 81(4):1342–1369, 2019. doi:10.1007/s00453-018-0474-x.

[14] John A. Bondy and Uppaluri S. R. Murty. *Graph theory with applications*, volume 290. Citeseer, 1976.

[15] Flavia Bonomo, Maria Chudnovsky, Peter Maceli, Oliver Schaudt, Maya Stein, and Mingxian Zhong. Three-coloring and list three-coloring of graphs without induced paths on seven vertices. *Combinatorica*, 38(4):779–801, 2018. doi:10.1007/s00493-017-3553-8.

[16] Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, Piotr Skowron, and Nimrod Talmon. Mixed integer programming with convex/concave constraints: Fixed-parameter tractability and applications to multicovering and voting. *CoRR*, 2017. arXiv:1709.02850.

[17] Boštjan Brešar, Sandi Klavžar, and Douglas F. Rall. On the packing chromatic number of Cartesian products, hexagonal lattice, and trees. *Discrete Applied Mathematics*, 155(17):2303–2311, 2007. doi:10.1016/j.dam.2007.06.008.

[18] Hajo Broersma, Fedor V. Fomin, Petr A. Golovach, and Daniël Paulusma. Three complexity results on coloring P_k-free graphs. *European Journal of Combinatorics*, 34(3):609–619, 2013. doi:10.1016/j.ejc.2011.12.008.

[19] Hajo Broersma, Petr A. Golovach, Daniël Paulusma, and Jian Song. Updating the complexity status of coloring graphs without a fixed induced linear forest. *Theoretical Computer Science*, 414(1):9–19, 2012. doi:10.1016/j.tcs.2011.10.005.

[20] Hajo Broersma, Ton Kloks, Dieter Kratsch, and Haiko Müller. Independent sets in asteroidal triple-free graphs. *SIAM Journal on Discrete Mathematics*, 12(2):276–287, 1999. doi:10.1137/S0895480197326346.

[21] Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, 84(1):119–138, 1997. doi:10.1016/S0168-0072(95)00020-8.

[22] Tiziana Calamoneri. The $L(h, k)$-labelling problem: An updated survey and annotated bibliography. *The Computer Journal*, 54(8):1344–1371, 2011. doi:10.1093/comjnl/bxr037.

[23] Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation*, 201(2):216–231, 2005. doi:10.1016/j.ic.2005.05.001.

[24] Morgan Chopin, André Nichterlein, Rolf Niedermeier, and Mathias Weller. Constant thresholds can make target set selection tractable. *Theory of Computing Systems*, 55(1):61–83, 2014. doi:10.1007/s00224-013-9499-3.

[25] Maria Chudnovsky. Coloring graphs with forbidden induced subgraphs. In *Proceedings of the International Congress of Mathematicians 2014*, volume 4, pages 291–302. 2014. URL: `https://www.mathunion.org/fileadmin/ICM/Proceedings/ICM2014.4/ICM2014.4.pdf`.

[26] Maria Chudnovsky, Shenwei Huang, Sophie Spirkl, and Mingxian Zhong. List-three-coloring graphs with no induced $P_6 + rP_3$. *CoRR*, 2018. `arXiv:1806.11196`.

[27] Maria Chudnovsky, Peter Maceli, Juraj Stacho, and Mingxian Zhong. 4-Coloring P_6-free graphs with no induced 5-cycles. *Journal of Graph Theory*, 84(3):262–285, 2017. `doi:10.1002/jgt.22025`.

[28] Maria Chudnovsky, Sophie Spirkl, and Mingxian Zhong. Four-coloring P_6-free graphs. I. Extending an excellent precoloring. *CoRR*, 2018. `arXiv:1802.02282`.

[29] Maria Chudnovsky, Sophie Spirkl, and Mingxian Zhong. Four-coloring P_6-free graphs. II. Finding an excellent precoloring. *CoRR*, 2018. `arXiv:1802.02283`.

[30] Maria Chudnovsky and Juraj Stacho. 3-colorable subclasses of P_8-free graphs. *SIAM Journal on Discrete Mathematics*, 32(2):1111–1138, 2018. `doi:10.1137/16M1104858`.

[31] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3–5, 1971, Shaker Heights, Ohio, USA*, pages 151–158, 1971. URL: `https://doi.org/10.1145/800157.805047`, `doi:10.1145/800157.805047`.

[32] Ricardo Corrêa, Frédéric Havet, and Jean-Sébastien Sereni. About a brooks-type theorem for improper colouring. *The Australasian Journal of Combinatorics*, 43:219–230, 2009. URL: `https://hal.inria.fr/inria-00223009/`.

[33] Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, March 1990. `doi:10.1016/0890-5401(90)90043-h`.

[34] Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic. A language-theoretic approach.*, volume 138. Cambridge: Cambridge University Press, 2012. `doi:10.1017/CBO9780511977619`.

[35] Bruno Courcelle, Joost Engelfriet, and Grzegorz Rozenberg. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2):218–270, 1993. `doi:10.1016/0022-0000(93)90004-G`.

[36] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000. `doi:10.1007/s002249910009`.

[37] Bruno Courcelle and Mohamed Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, 109(1&2):49–82, 1993. `doi:10.1016/0304-3975(93)90064-z`.

[38] Jean-François Couturier, Petr A. Golovach, Dieter Kratsch, and Daniël Paulusma. List coloring in the absence of a linear forest. *Algorithmica*, 71(1):21–35, 2015. doi:10.1007/s00453-013-9777-0.

[39] Lenore Cowen, Wayne Goddard, and C. Esther Jesurum. Defective coloring revisited. *Journal of Graph Theory*, 24(3):205–219, 1997. doi:10.1002/(SICI)1097-0118(199703)24:3<205::AID-JGT2>3.0.CO;2-T.

[40] Lenore J. Cowen, Robert Cowen, and Douglas R. Woodall. Defective colorings of graphs in surfaces: Partitions into subgraphs of bounded valency. *Journal of Graph Theory*, 10(2):187–195, 1986. doi:10.1002/jgt.3190100207.

[41] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

[42] Konrad K. Dabrowski and Daniël Paulusma. On colouring $(2P_2, H)$-free and $(P5, H)$-free graphs. *Information Processing Letters*, 131:26–32, 2018. doi:10.1016/j.ipl.2018.02.003.

[43] Holger Dell, Thore Husfeldt, Bart M. P. Jansen, Petteri Kaski, Christian Komusiewicz, and Frances A. Rosamond. The First Parameterized Algorithms and Computational Experiments Challenge. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation (IPEC 2016)*, volume 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:9, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.IPEC.2016.30.

[44] Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The PACE 2017 Parameterized Algorithms and Computational Experiments Challenge: The Second Iteration. In Daniel Lokshtanov and Naomi Nishimura, editors, *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, volume 89 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:12, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.IPEC.2017.30.

[45] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[46] Rodney G. Downey and Michael R. Fellows. Fixed parameter tractability and completeness. In *Complexity Theory: Current Research, Dagstuhl Workshop, February 2–8, 1992*, pages 191–225, 1992. URL: http://homepages.msor.vuw.ac.nz/~downey/publications/manitoba.pdf.

[47] Rodney G. Downey and Michael R. Fellows. *Complexity Theory*, chapter Fixed-parameter Tractability and Completeness III: Some Structural Aspects of the W Hierarchy, pages 191–225. Cambridge University Press, New York, NY, USA, 1993. URL: http://dl.acm.org/citation.cfm?id=183589.183729.

[48] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995. `doi:10.1137/S0097539792228228`.

[49] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: on completeness for W[1]. *Theoretical Computer Science*, 141(1&2):109–131, 1995. `doi:10.1016/0304-3975(94)00097-3`.

[50] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer, 1999. `doi:10.1007/978-1-4612-0515-9`.

[51] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer, 2013. `doi:10.1007/978-1-4471-5559-1`.

[52] Zdeněk Dvořák, Tomáš Masařík, Jan Musílek, and Ondřej Pangrác. Flexibility of planar graphs of girth at least six. *CoRR*, 2019. `arXiv:1902.04069`.

[53] Zdeněk Dvořák, Tomáš Masařík, Jan Musílek, and Ondřej Pangrác. Flexibility of triangle-free planar graphs. *CoRR*, 2019. `arXiv:1902.02971`.

[54] Zdeněk Dvořák, Sergey Norin, and Luke Postle. List coloring with requests. *Journal of Graph Theory*, 2019. `doi:10.1002/jgt.22447`.

[55] Keith Edwards. The complexity of colouring problems on dense graphs. *Theoretical Computer Science*, 43:337–343, 1986. `doi:10.1016/0304-3975(86)90184-2`.

[56] Thomas Emden-Weinert, Stefan Hougardy, and Bernd Kreuter. Uniquely colourable graphs and the hardness of colouring graphs of large girth. *Combinatorics, Probability and Computing*, 7(04):375–386, 1998. `doi:10.1017/s0963548398003678`.

[57] Ronald Fagin, Larry J. Stockmeyer, and Moshe Y. Vardi. On monadic NP vs monadic co-NP. *Information and Computation*, 120(1):78–92, 1995. `doi:10.1006/inco.1995.1100`.

[58] Jiří Fiala, Sandi Klavžar, and Bernard Lidický. The packing chromatic number of infinite product graphs. *European Journal of Combinatorics*, 30(5):1101–1113, 2009. `doi:10.1016/j.ejc.2008.09.014`.

[59] Jiří Fiala and Petr A. Golovach. Complexity of the packing coloring problem for trees. *Discrete Applied Mathematics*, 158(7):771–778, 2010. `doi:10.1016/j.dam.2008.09.001`.

[60] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. `doi:10.1007/3-540-29953-X`.

[61] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing.* Cambridge University Press, 2019. `doi:10.1017/9781107415157`.

[62] Piotr Formanowicz and Krzysztof Tanaś. A survey of graph coloring—its types, methods and applications. *Foundations of Computing and Decision Sciences*, 37(3):223–238, 2012. doi:10.2478/v10209-011-0012-y.

[63] András Frank and Eva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987. doi:10.1007/BF02579200.

[64] Marietjie Frick. A survey of (m, k)-colorings. In John Gimbel, John W. Kennedy, and Louis V. Quintas, editors, *Quo Vadis, Graph Theory?*, volume 55 of *Annals of Discrete Mathematics*, pages 45–57. Elsevier, 1993. doi:10.1016/S0167-5060(08)70374-1.

[65] Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Annals of Pure and Applied Logic*, 130(1):3 – 31, 2004. Papers presented at the 2002 IEEE Symposium on Logic in Computer Science (LICS). doi:10.1016/j.apal.2004.01.007.

[66] Jakub Gajarský and Petr Hliněný. Kernelizing MSO properties of trees of fixed height, and some consequences. *Logical Methods in Computer Science*, Volume 11, Issue 1, April 2015. doi:10.2168/LMCS-11(1:19)2015.

[67] Jakub Gajarský, Petr Hliněný, Jan Obdržálek, Daniel Lokshtanov, and M. S. Ramanujan. A new perspective on FO model checking of dense graph classes. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 176–184. ACM, 2016. doi:10.1145/2933575.2935314.

[68] Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In Gregory Gutin and Stefan Szeider, editors, *Parameterized and Exact Computation: 8th International Symposium, IPEC 2013, Sophia Antipolis, France, September 4–6, 2013, Revised Selected Papers*, pages 163–176. Springer International Publishing, Cham, 2013. doi:10.1007/978-3-319-03898-8_15.

[69] Robert Ganian. Twin-cover: Beyond vertex cover in parameterized algorithmics. In Dániel Marx and Peter Rossmanith, editors, *Parameterized and Exact Computation - 6th International Symposium IPEC 2011, Saarbrücken, Germany, September 6–8, 2011. Revised Selected Papers*, volume 7112 of *Lecture Notes in Computer Science*, pages 259–271. Springer, 2011. doi:10.1007/978-3-642-28050-4_21.

[70] Robert Ganian. Improving vertex cover as a graph parameter. *Discrete Mathematics & Theoretical Computer Science*, 17(2):77–100, 2015. URL: http://dmtcs.episciences.org/2136.

[71] Robert Ganian, Petr Hliněný, Jaroslav Nešetřil, Jan Obdržálek, and Patrice Ossona de Mendez. Shrub-depth: Capturing height of dense graphs. *CoRR*, 2017. arXiv:1707.00359.

[72] Robert Ganian, Petr Hliněný, Jaroslav Nešetřil, Jan Obdržálek, Patrice Ossona de Mendez, and Reshma Ramadurai. When trees grow low: Shrubs and fast MSO_1. In *Mathematical Foundations of Computer Science 2012—37th International Symposium, MFCS 2012, Bratislava, Slovakia, August 27–31, 2012. Proceedings*, pages 419–430, 2012. doi:10.1007/978-3-642-32589-2_38.

[73] Robert Ganian, Petr Hliněný, Jaroslav Nešetřil, Jan Obdržálek, and Patrice Ossona de Mendez. Shrub-depth: Capturing height of dense graphs. *Logical Methods in Computer Science*, 15(1), 2019. URL: https://lmcs.episciences.org/5149.

[74] Robert Ganian and Jan Obdržálek. Expanding the expressive power of monadic second-order logic on restricted graph classes. In Thierry Lecroq and Laurent Mouchard, editors, *Combinatorial Algorithms—24th International Workshop, IWOCA 2013, Revised Selected Papers*, volume 8288 of *Lecture Notes in Computer Science*, pages 164–177. Springer, 2013. doi:10.1007/978-3-642-45278-9_15.

[75] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[76] Wayne Goddard, Sandra M. Hedetniemi, Stephen T. Hedetniemi, John M. Harris, and Douglas F. Rall. Broadcast chromatic numbers of graphs. *Ars Combinatoria*, 86:33–49, 2008.

[77] Petr A. Golovach, Matthew Johnson, Daniël Paulusma, and Jian Song. A survey on the computational complexity of colouring graphs with forbidden subgraphs. *Journal of Graph Theory*, 84(4):331–363, 2017. doi:10.1002/jgt.22028.

[78] Petr A. Golovach, Daniël Paulusma, and Jian Song. Closing complexity gaps for coloring problems on H-free graphs. *Information and Computation*, 237:204–214, 2014. doi:10.1016/j.ic.2014.02.004.

[79] Carla Groenland, Karolina Okrasa, Paweł Rzążewski, Alex Scott, Paul Seymour, and Sophie Spirkl. h-colouring P_t-free graphs in subexponential time. *CoRR*, 2018. arXiv:1803.05396.

[80] Martin Grohe and Stephan Kreutzer. Methods for algorithmic meta theorems. *Model Theoretic Methods in Finite Combinatorics*, pages 181–206, 2011. doi:10.1090/conm/558/11051.

[81] Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. *Journal of the ACM*, 64(3):17:1–17:32, 2017. doi:10.1145/3051095.

[82] Martin Grötschel, László Lovász, and Alexander Schrijver. Polynomial algorithms for perfect graphs. *Annals of Discrete Mathematics*, 21:325–356, 1984. doi:10.1016/s0304-0208(08)72943-8.

[83] Andrzej Grzesik, Tereza Klimošová, Marcin Pilipczuk, and Michał Pilipczuk. Polynomial-time algorithm for maximum weight independent set on P_6–free graphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6–9, 2019*, pages 1257–1271. SIAM, 2019. doi:10.1137/1.9781611975482.77.

[84] Bjarki Á. Guðmundsson, Tómas K. Magnússon, and Björn O. Sæmundsson. Bounds and fixed-parameter algorithms for weighted improper coloring (extended version). *CoRR*, pages 1–18, 2015. arXiv:1509.00099.

[85] Bjarki Á. Guðmundsson, Tómas K. Magnússon, and Björn O. Sæmundsson. Bounds and fixed-parameter algorithms for weighted improper coloring. *Electronic Notes in Theoretical Computer Science*, 322:181–195, 2016. doi:10.1016/j.entcs.2016.03.013.

[86] Frédéric Havet, Ross J. Kang, and Jean-Sébastien Sereni. Improper coloring of unit disk graphs. *Networks*, 54(3):150–164, 2009. doi:10.1002/net.20318.

[87] Frédéric Havet and Jean-Sébastien Sereni. Improper choosability of graphs and maximum average degree. *Journal of Graph Theory*, 52(3):181–199, 2006. doi:10.1002/jgt.20155.

[88] Chính T. Hoàng, Marcin Kamiński, Vadim V. Lozin, Joe Sawada, and Xiao Shu. Deciding k-colorability of P_5-free graphs in polynomial time. *Algorithmica*, 57(1):74–81, 2010. doi:10.1007/s00453-008-9197-8.

[89] Ian Holyer. The NP-Completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981. doi:10.1137/0210055.

[90] Shenwei Huang. Improved complexity results on k-coloring P_t-free graphs. *European Journal of Combinatorics*, 51:336–346, 2016. doi:10.1007/978-3-642-40313-2_49.

[91] Johan Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182(1):105–142, 1999. doi:10.1007/BF02392825.

[92] Klaus Jansen, Stefan Kratsch, Dániel Marx, and Ildikó Schlotter. Bin packing with fixed number of bins revisited. *Journal of Computer and System Sciences*, 79(1):39–49, 2013. doi:10.1016/j.jcss.2012.04.004.

[93] Tommy R. Jensen and Bjarne Toft. *Graph Coloring Problems.* Wiley Series in Discrete Mathematics and Optimization. Wiley, 1995. doi:10.1002/9781118032497.

[94] Rhys P. Jones. *Hereditary properties and P-chromatic numbers*, pages 83—-88. London Mathematical Society Lecture Note Series. Cambridge University Press, 1974. doi:10.1017/CBO9780511662072.014.

[95] Ross J. Kang, Tobias Müller, and Jean-Sébastien Sereni. Improper colouring of (random) unit disk graphs. *Discrete Mathematics*, 308(8):1438–1454, 2008. Third European Conference on Combinatorics. doi:https://doi.org/10.1016/j.disc.2007.07.070.

[96] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103, 1972. URL: `http://www.cs.berkeley.edu/%7Eluca/cs172/karp.pdf`.

[97] Minki Kim, Bernard Lidický, Tomáš Masařík, and Florian Pfender. Notes on complexity of packing coloring. *Information Processing Letters*, 137:6–10, 2018. `doi:10.1016/j.ipl.2018.04.012`.

[98] Tereza Klimošová, Josef Malík, Tomáš Masařík, Jana Novotná, Daniël Paulusma, and Veronika Slívová. Colouring $(p_r + p_s)$-free graphs. In *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16–19, 2018, Jiaoxi, Yilan, Taiwan*, pages 5:1–5:13, 2018. `doi:10.4230/LIPIcs.ISAAC.2018.5`.

[99] Tereza Klimošová, Josef Malík, Tomáš Masařík, Jana Novotná, Daniël Paulusma, and Veronika Slívová. Colouring $(p_r + p_s)$-free graphs. *CoRR*, pages 1–21, 2018. `arXiv:1804.11091`.

[100] Dušan Knop, Martin Koutecký, Tomáš Masařík, and Tomáš Toufar. Simplified algorithmic metatheorems beyond MSO: Treewidth and neighborhood diversity. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science: 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*, pages 344–357, Cham, 2017. Springer International Publishing. `doi:10.1007/978-3-319-68705-6_26`.

[101] Dušan Knop, Tomáš Masařík, and Tomáš Toufar. Parameterized complexity of fair vertex evaluation problems. *CoRR*, 2018. `arXiv:1803.06878`.

[102] Dušan Knop, Tomáš Masařík, and Tomáš Toufar. Parameterized complexity of fair vertex evaluation problems. In *44rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26–30, 2019, Aachen, Germany*, pages 8:1–8:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

[103] Petr Kolman, Bernard Lidický, and Jean-Sébastien Sereni. Fair edge deletion problems on treedecomposable graphs and improper colorings, 2010. URL: `http://orion.math.iastate.edu/lidicky/pub/kls10.pdf`.

[104] Petr Kolman, Bernard Lidický, and Jean-Sébastien Sereni. On minimum fair odd cycle transversal. *KAM-DIMATIA Series*, 2010. URL: `https://kam.mff.cuni.cz/~kamserie/serie/clanky/2010/s956.ps`.

[105] Petr Kolman, Bernard Lidický, and Jean-Sébastien Sereni. On Fair Edge Deletion Problems, 2009. URL: `https://kam.mff.cuni.cz/~kolman/papers/kls09.pdf`.

[106] Juha Kontinen and Hannu Niemistö. Extensions of MSO and the monadic counting hierarchy. *Information and Computation*, 209(1):1–19, 2011. `doi:10.1016/j.ic.2010.09.002`.

[107] Daniel Král', Jan Kratochvíl, Zsolt Tuza, and Gerhard J. Woeginger. Complexity of coloring graphs without forbidden induced subgraphs. In *Graph-Theoretic Concepts in Computer Science, 27th International Workshop, WG 2001, Boltenhagen, Germany, June 14–16, 2001, Proceedings*, pages 254–262, 2001. doi:10.1007/3-540-45477-2_23.

[108] Jan Kratochvíl, Zsolt Tuza, and Margit Voigt. New trends in the theory of graph colorings: Choosability and list coloring. In *Contemporary Trends in Discrete Mathematics*, pages 183–197. American Mathematical Society, may 1999. doi:10.1090/dimacs/049/13.

[109] Mukkai S. Krishnamoorthy and Narsingh Deo. Node-deletion NP-complete problems. *SIAM Journal on Computing*, 8(4):619–625, 1979. doi:10.1137/0208049.

[110] Peter C. B. Lam, Baogang Xu, and Jiazhuang Liu. The 4-choosability of plane graphs without 4-cycles. *Journal of Combinatorial Theory, Series B*, 76(1):117–126, 1999. doi:10.1006/jctb.1998.1893.

[111] Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012. doi:10.1007/s00453-011-9554-x.

[112] Michael Lampis. Model checking lower bounds for simple graphs. *Logical Methods in Computer Science*, 10(1), 2014. doi:10.2168/LMCS-10(1:18)2014.

[113] Van Bang Le, Bert Randerath, and Ingo Schiermeyer. On the complexity of 4-coloring graphs without long induced paths. *Theoretical Computer Science*, 389(1–2):330–335, 2007. doi:10.1016/j.tcs.2007.09.009.

[114] Hendrik W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983. doi:10.1287/moor.8.4.538.

[115] Daniel Leven and Zvi Galil. NP completeness of finding the chromatic index of regular graphs. *Journal of Algorithms*, 4(1):35–44, 1983. doi:10.1016/0196-6774(83)90032-9.

[116] Leonid Libkin. *Elements of Finite Model Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004. doi:10.1007/978-3-662-07003-1.

[117] Li-Shin Lin and Sartaj Sahni. Fair edge deletion problems. *IEEE Transactions on Computers*, 38(5):756–761, 1989. doi:10.1109/12.24280.

[118] László Lovász. Coverings and coloring of hypergraphs. In *Proceedings of the 4th Southeastern Conference on Combinatorics, Graph Theory, and Computing (Florida Atlantic University, Boca Raton, March 5–8, 1973)*, volume VIII, pages 3–12. Utilitas Mathematica Publishing Company, Winnipeg, 1973.

[119] Barnaby Martin, Franco Raimondi, Taolue Chen, and Jos Martin. The packing chromatic number of the infinite square lattice is between 13 and 15. *Discrete Applied Mathematics*, 225:136–142, 2017. URL: `https://doi.org/10.1016/j.dam.2017.03.013`.

[120] Dániel Marx and MichałPilipczuk. Everything you always wanted to know about the parameterized complexity of subgraph isomorphism (but were afraid to ask). In *31st International Symposium on Theoretical Aspects of Computer Science, STACS 2014, March 5–8, 2014, Lyon, France*, pages 542–553, 2014. `doi:10.4230/LIPIcs.STACS.2014.542`.

[121] Tomáš Masařík and Tomáš Toufar. Parameterized complexity of fair deletion problems. In T.V. Gopal, Gerhard Jäger, and Silvia Steila, editors, *Theory and Applications of Models of Computation: 14th Annual Conference, TAMC 2017, Bern, Switzerland, April 20-22, 2017, Proceedings*, pages 628–642, Cham, 2017. Springer International Publishing. `doi:10.1007/978-3-319-55911-7\_45`.

[122] Tomáš Masařík and Tomáš Toufar. Parameterized complexity of fair deletion problems. *Discrete Applied Mathematics*, 2019. `doi:10.1016/j.dam.2019.06.001`.

[123] Tomáš Masařík. Flexibility of planar graphs without 4-cycles. In *EUROCOMB 2019, accepted.* 2019. `arXiv:1903.01460`.

[124] Jiří Matoušek and Jaroslav Nešetřil. *Invitation to Discrete Mathematics (2. ed.).* Oxford University Press, 2009.

[125] Michael Molloy and Bruce Reed. *Graph Colouring and the Probabilistic Method.* Springer-Verlag Berlin Heidelberg. Springer, Berlin, Heidelberg, 2002. `doi:10.1007/978-3-642-04016-0`.

[126] Jaroslav Nešetřil and Patrice Ossona De Mendez. *Sparsity: graphs, structures, and algorithms*, volume 28. Springer Science & Business Media, 2012. `doi:10.1007/978-3-642-27875-4`.

[127] Timm Oertel, Christian Wagner, and Robert Weismantel. Integer convex minimization by mixed integer linear optimization. *Operations Research Letters*, 42(6):424 – 428, 2014. `doi:10.1016/j.orl.2014.07.005`.

[128] Daniël Paulusma. Open problems on graph coloring for special graph classes. In *Graph-Theoretic Concepts in Computer Science—41st International Workshop, WG 2015, Garching, Germany, June 17–19, 2015, Revised Papers*, pages 16–30, 2015. `doi:10.1007/978-3-662-53174-7\_2`.

[129] Michał Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2011. `doi:10.1007/978-3-642-22993-0_47`.

[130] Bert Randerath and Ingo Schiermeyer. 3-colorability in P for P_6-free graphs. *Discrete Applied Mathematics*, 136(2–3):299–313, 2004. doi:10.1016/s0166-218x(03)00446-3.

[131] Bert Randerath and Ingo Schiermeyer. Vertex colouring and forbidden subgraphs—a survey. *Graphs and Combinatorics*, 20(1):1–40, 2004. doi:10.1007/s00373-003-0540-1.

[132] Bert Randerath, Ingo Schiermeyer, and Meike Tewes. Three-colourability and forbidden subgraphs. II: polynomial algorithms. *Discrete Mathematics*, 251(1–3):137–153, 2002. doi:10.1016/s0012-365x(01)00335-1.

[133] Detlef Seese. Linear time computable problems and first-order descriptions. *Mathematical Structures in Computer Science*, 6(6):505–526, 1996. doi:10.1017/s0960129500070079.

[134] Christian Sloper. An eccentric coloring of trees. *The Australasian Journal of Combinatorics*, 29:309–321, 2004. URL: http://ajc.maths.uq.edu.au/pdf/29/ajc_v29_p309.pdf.

[135] Roman Soukal and Přemysl Holub. A note on packing chromatic number of the square lattice. *Electronic Journal of Combinatorics*, 17(1):Note 17, 7, 2010. URL: http://www.combinatorics.org/Volume_17/Abstracts/v17i1n17.html.

[136] Stefan Szeider. Monadic second order logic on graphs with local cardinality constraints. *ACM Transactions on Computational Logic*, 12(2):1–21, 2011. doi:10.1145/1877714.1877718.

[137] Marc Tedder, Dereck G. Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *ICALP 2008*, pages 634–645, 2008. doi:10.1007/978-3-540-70575-8_52.

[138] Carsten Thomassen. Every planar graph is 5-choosable. *Journal of Combinatorial Theory, Series B*, 62(1):180–181, 1994. doi:10.1006/jctb.1994.1062.

[139] Carsten Thomassen. 3-list-coloring planar graphs of girth 5. *Journal of Combinatorial Theory, Series B*, 64(1):101–107, 1995. doi:10.1006/jctb.1995.1027.

[140] Zsolt Tuza. Graph colorings with local constraints—a survey. *Discussiones Mathematicae Graph Theory*, 17(2):161–228, 1997. doi:10.7151/dmgt.1049.

[141] Douglas B. West. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, NJ, 1996.

[142] Gerhard J. Woeginger and Jiří Sgall. The complexity of coloring graphs without long induced paths. *Acta Cybernetica*, 15(1):107–117, 2001. URL: http://cyber.bibl.u-szeged.hu/index.php/actcybern/article/view/3566.

[143] David R Wood. Defective and clustered graph colouring. *Electronic Journal of Combinatorics*, Dynamic Surveys:DS23:1–71, 2018. URL: http://www.combinatorics.org/DS23.

[144] Douglas R. Woodall. Defective choosability of graphs with no edge-plus-independent-set minor. *Journal of Graph Theory*, 45(1):51–56, 2004. doi:10.1002/jgt.10153.

[145] Mihalis Yannakakis. Node- and edge-deletion NP-complete problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 253–264, 1978. doi:10.1145/800133.804355.

[146] Mihalis Yannakakis. Edge-deletion problems. *SIAM Journal on Computing*, 10(2):297–309, 1981. doi:10.1137/0210021.

[147] Riste Škrekovski. List improper colourings of planar graphs. *Combinatorics, Probability and Computing*, 8(3):293—299, 1999. doi:10.1017/s0963548399003752.

List of Figures

1.1 Hierarchy of graph parameters considered in the thesis. An arrow indicates that a graph parameter upper-bounds the other. 8

2.1 The reduction from Theorem 2.5 on a 4-cycle. 16

2.2 Hierarchy of graph parameters. An arrow indicates that a graph parameter upper-bounds the other. Thus, hardness results are implied in the direction of arrows and algorithms are implied in the reverse direction. Green circles and red rectangle colors distinguish between hardness results and FPT algorithms provided. Blue color without boundary denotes that the hardness is unknown. (cw is clique-width, nd is neighborhood diversity, mw is modular-width, pw is path-width, sd is shrub-depth, tc is twin cover, td is tree-depth, tw is tree-width, vc is vertex cover. See [41] for definitions.) 19

3.1 All possible connected components in $G[N_2 \cup N_3]$. 28

3.2 The situation in Branching II. 30

3.3 The situation in Branching IV if $t_1 \in N_1$ and if vertices s_a and s_b exist. 34

3.4 The situation in Branching VII. Dashed lines denote edges that might or might not be there. 38

4.1 Hierarchy of graph parameters with depicted complexity of the Fair L Vertex Evaluation problem on the left side and the Fair L Vertex Evaluation problem on the right side. An arrow indicates that a graph parameter upper-bounds the other. Thus, hardness results are implied in the direction of arrows, and FPT algorithms are implied in the reverse direction. Green colors indicate FPT results for MSO_2, orange are FPT for MSO_1, blue are open, and red are hardness results. Descriptions of the relations and parameteres are in Section 2.2.1. 46

5.1 Hierarchy of graph parameters with depicted complexity of the Fair L Vertex Evaluation problem. An arrow indicates that a graph parameter upper-bounds the other. Thus, hardness results are implied in the direction of arrows, and FPT algorithms are implied in the reverse direction. Green colors indicate FPT results for MSO_2, orange are FPT for MSO_1, blue are open, and red are hardness results. We denote treewidth by tw, shrubdepth by sd, and clique-width by cw. We refer to book [41] for definitions. Other parameters and their respective abbreviations are defined in Subsection 5.1.2. 54

5.2 Example of a 7×7 A-shape. All cliques of size 3 will be assigned to yellow (light gray) fields, while cliques of size 8 will be assigned to orange (darker gray) fields. 58

5.3 An example of uncertainty in computation of objective function. The value in the last row depends on the size of the clique we are assigning to those cells. The value in the cell is how much we pay for any compatible clique assigned to this cell. 58
5.4 An overview of the reduction in the proof of Theorem 5.2. The gray vertices are enforced to be a part the fair vertex cover. If a vertex fair objective was lowered, then the resulting threshold is beneath the vertex (the group of vertices). 61

List of Tables

1.1 A summary and a comparisons of choosability and respective results in the flexibility setting on subclases of planar graphs. Non-implied bounds are accompagned with the respective citation. 5

1.2 Comparisons of running times between naive exponential (2^n), XP (n^k) and FPT ($2^k n$) algorithms on a current computer (4 cores, 3GHz). Naive running times are in the first row in brackets, FPT | XP running times are within the inner-fields of the table. Shortcut yrs stands for years. 6

3.1 Summary for P_t-free graphs. 22

4.1 The table summarize metatheorem results (with a citation), in terms of FPT complexity and used logic, parameterized by structural parameters. Green cells denote FPT results and red cells represent hardness results. Logic L in metatheorems is specified by a logic used in the respective theorem. Symbol $\mathsf{L}_\emptyset$ denotes any logic that can express an edgeless graph. Symbol ✓denotes implied results. A question mark (?) indicates that the complexity is unknown. Fair edge problems are delimited from fair vertex problems since there are no apparent relations between them. 45

5.1 The table summarizes some related (with a citation) and all the presented (with a reference) results on the studied parameters. Green cells denote FPT results, and red cells represent hardness results. Logic L in metatheorems is specified by a logic used in the respective theorem. Symbol $*$ denotes implied results from previous research and symbol ✓denotes new implied results. A question mark (?) indicates that the complexity is unknown. The Fair Edge L Deletion problem is delimited from Vertex problems since there are no apparent relations between them. 53

List of Publications

By convention of my research area, author names are in alphabetical order, except papers [13] and [10]. Publications in each category are listed in the chronological order.

Journal Publications

[1] Tomáš Masařík and Tomáš Toufar. Parameterized complexity of fair deletion problems. In *Discrete Applied Mathematics*, available online, 2019. doi:10.1016/j.dam.2019.06.001.
Also in *Theory and Applications of Models of Computation—14th Annual Conference, TAMC 2017, Bern, Switzerland, April 20–22, 2017, Proceedings*, pages 628–642, 2017. doi:10.1007/978-3-319-55911-7_45.

[2] Zdeněk Dvořák, Tomáš Masařík, Jan Musílek, and Ondřej Pangrác. Triangle-free planar graphs with the smallest independence number. *Journal of Graph Theory*, 90(3):443–454, 2019. doi:10.1002/jgt.22406.

[3] Minki Kim, Bernard Lidický, Tomáš Masařík, and Florian Pfender. Notes on complexity of packing coloring. *Information Processing Letters*, 137:6–10, 2018. doi:10.1016/j.ipl.2018.04.012.

[4] Dušan Knop and Tomáš Masařík. Computational complexity of distance edge labeling. *Discrete Applied Mathematics*, 246:80–98, 2018. doi:10.1016/j.dam.2017.01.007.
Also in *Combinatorial Algorithms—26th International Workshop, IWOCA 2015, Verona, Italy, October 5–7, 2015, Revised Selected Papers*, pages 287–298, 2015. doi:10.1007/978-3-319-29516-9_24.

Conference Publications
(excluded those published in journals)

[5] Tomáš Masařík, Irene Muzi, Marcin Pilipczuk, Paweł Rzążewski, and Manuel Sorge. Packing directed circuits quarter-integrally. Accepted to *ESA 2019*, 2019.

[6] Dušan Knop, Tomáš Masařík, and Tomáš Toufar. Parameterized complexity of fair vertex evaluation problems. Accepted to *MFCS 2019*, 2019.
Full version on *CoRR*, pages 1–23, 2018. arXiv:1803.06878.

[7] Tomáš Masařík. Flexibility of planar graphs without 4-cycles. Accepted to *EUROCOMB 2019*, 2019.
Full version on *CoRR*, pages 1–6, 2019. arXiv:1903.01460.

[8] Tereza Klimošová, Josef Malík, Tomáš Masařík, Jana Novotná, Daniël Paulusma, and Veronika Slívová. Colouring $(P_r + P_s)$-free graphs. In *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16–19, 2018, Jiaoxi, Yilan, Taiwan*, pages 5:1–5:13, 2018. doi:10.4230/LIPIcs.ISAAC.2018.5.
Full version on *CoRR*, pages 1–21, 2018. arXiv:1804.11091.

[9] Pavel Dvořák, Andreas Emil Feldmann, Dušan Knop, Tomáš Masařík, Tomáš Toufar, and Pavel Veselý. Parameterized approximation schemes for Steiner trees with small number of steiner vertices. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28–March 3, 2018, Caen, France*, pages 26:1–26:15, 2018.doi:10.4230/LIPIcs.STACS.2018.26.
Full version on *CoRR*, pages 1–24, 2018. arXiv:1710.00668.

[10] Jana Novotná, Milan Hladík, and Tomáš Masařík. Duality gap in interval linear programming. In *14th International Symposium on Operational Research, SOR 2017, September 27–29, 2017, Bled, Slovenia*, pages 501–506, 2017. http://sor17.fov.uni-mb.si/sor-publications/.
Full version on *CoRR*, pages 1–13, 2018. arXiv:1802.05795.

[11] Dušan Knop, Martin Koutecký, Tomáš Masařík, and Tomáš Toufar. Simplified algorithmic metatheorems beyond MSO: treewidth and neighborhood diversity. In *Graph-Theoretic Concepts in Computer Science—43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21–23, 2017, Revised Selected Papers*, pages 344–357, 2017. doi:10.1007/978-3-319-68705-6_26.
Full version on *CoRR*, pages 1–30, 2018. arXiv:1703.00544.

[12] Pavel Dvořák, Dušan Knop, and Tomáš Masařík. Anti-path cover on sparse graph classes. In *Proceedings 11th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2016, Telč, Czech Republic, October 21–23 2016*, pages 82–86, 2016. doi:10.4204/EPTCS.233.8.

Miscellaneous

[13] Radek Hušek, Tomáš Toufar, Dušan Knop, Tomáš Masařík, and Eduard Eiben. Steiner Tree Heuristics for PACE 2018 Challenge Track C. *public repository*, 2018. `https://github.com/goderik01/PACE2018`.

Submitted
(excluded those already published)

[14] Konrad Dabrowski, Tomáš Masařík, Jana Novotná, Daniël Paulusma, and Paweł Rzążewski. Harnessing the power of atoms. *Manuscript*, 2019+.

[15] Radek Hušek, Dušan Knop, and Tomáš Masařík. Approximation Algorithms for Steiner Tree Based on MST and Star Contractions. *Manuscript*, 2019+.

[16] Julien Baste, Michael R. Fellows, Lars Jaffke, Tomáš Masařík, Mateus de Oliveira Oliveira, Geevarghese Philip, and Frances A. Rosamond. Diversity in Combinatorial Optimization. *CoRR*, pages 1–14, 2019. arXiv:1903.07410.

[17] Zdeněk Dvořák, Tomáš Masařík, Jan Musílek, and Ondřej Pangrác. Flexibility of planar graphs of girth at least six. *CoRR*, pages 1–11, 2019. arXiv:1902.04069.

[18] Zdeněk Dvořák, Tomáš Masařík, Jan Musílek, and Ondřej Pangrác. Flexibility of triangle-free planar graphs. *CoRR*, pages 1–28, 2019. arXiv:1902.02971.

[19] Jinha Kim, Ryan R. Martin, Tomáš Masařík, Warren Shull, Heather C. Smith, Andrew Uzzell, and Zhiyu Wang. On difference graphs and the local dimension of posets. *CoRR*, pages 1–13, 2018. arXiv:1803.08641.