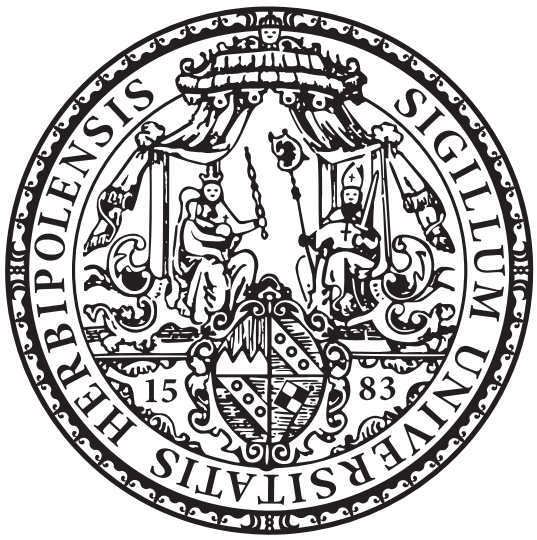
Master Thesis

Coloring Mixed Graphs

Antonio Lauerbach

Date of Submission:	September 3, 2025
Advisors:	Prof. Dr. Alexander Wolff
	Prof. Dr. Konstanty Junosza-Szaniawski



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen und Komplexität

Abstract

Coloring graphs is an old and well-known graph problem with applications in several areas, such as scheduling, frequency assignment, and graph drawing. Mixed graphs extend the classes of directed and undirected (simple) graphs by allowing for directed and undirected edges in the same graph. Combining these concepts, mixed graph colorings generalize classical colorings of undirected graphs by having vertex colors adhere to strict partial orders imposed by directed edges. Therefore, mixed graph colorings can be used to model more complex problems, such as scheduling with precedence constraints. In this thesis, we focus on developing exact single-exponential-time algorithms for mixed graph coloring. We adapt existing algorithms and techniques for coloring undirected graphs to mixed graphs, and use a novel approach to obtain a polynomial-space algorithm. Furthermore, we explore the parameterized complexity of mixed graph coloring, with a focus on treewidth and related parameters. We also provide bounds on the chromatic number of mixed graphs in terms of the length of directed paths.

Zusammenfassung

Graphenfärben ist ein altbekanntes Problem aus der Graphentheorie mit vielen Anwendungsbereichen, wie beispielsweise in der Ablaufplanung, der Frequenzzuteilung, und dem Graphenzeichnen. Gemischte Graphen erweitern das Konzept gerichteter und ungerichteter (einfacher) Graphen dadurch, dass in einem gemischten Graphen sowohl gerichtete als auch ungerichtete Kanten vorkommen können. Das Färben gemischter Graphen erweitert das Färben ungerichteter Graphen dadurch, dass die Farben der Knoten sich an eine, durch die gerichteten Kanten auferlegte, strenge Halbordnung halten müssen. Dementsprechend können Färbungen gemischter Graphen zum Modellieren komplexerer Sachverhalte benutzt werden, wie zum Beispiel für die Ablaufplanung mit Präzedenz-Einschränkungen. In dieser Arbeit entwickeln wir exakte Einexponentialzeit-Algorithmen fürs Färben gemischter Graphen. Dazu adaptieren wir bestehende Algorithmen und Techniken zum Färben ungerichteter Graphen. Weiterhin entwickeln wir einen neuartigen Ansatz um einen Algorithmus mit polynomiellem Platzbedarf zu erhalten. Außerdem erkunden wir die parametrisierte Komplexität des Färbungsproblems auf gemischten Graphen, mit einem Fokus auf Baumweite und verwandte Parameter. Wir zeigen auch obere Schranken für die chromatische Zahl gemischter Graphen in Abhängigkeit von der Länge gerichteter Pfade.

Contents

1 Introduction

Graph coloring has been frequently used to solve special types of scheduling problems, where different items have to be scheduled while avoiding conflicts. An example would be timetabling, where courses (vertices) have to be scheduled in certain time slots (colors) while avoiding conflicts (edges), e.g., due to two courses having the same teacher or students. These types of problems are referred to as chromatic scheduling problems. In real word applications, there are usually several additional constraints, such as certain courses having to be scheduled in specific time slots, certain teachers not being available at certain times, courses using multiple consecutive time slots, or a lecture having to be held before the corresponding exercises. Thus, there exist several extensions of coloring to accommodate these additional constraints, for example, list coloring can be used to restrict courses to certain slots [dW97], or interval coloring can be used to enforce consecutive time slots [Kub89]. To account for precedence constraints, i.e., one course having to be scheduled before another, Hansen et al. [HKdW97] added arcs to the given graph, with an arc (u, v) meaning that u has to be scheduled before v. The resulting graph, consisting of vertices, edges, and arcs, is called a *mixed graph*. In a *proper coloring* of a mixed graph, vertices connected by an edge must have different colors, fulfilling the conflict constraints, and for each arc (u, v), the color of u must be less than the color of v, fulfilling the precedence constraints. The problem of deciding whether a mixed graph can be properly colored with k colors is called MixedColoring. The problem for undirected graphs is called Coloring.

Apart from scheduling, mixed graph colorings have also been applied in graph drawing in order to compact layered orthogonal drawings [GMR+22, GJK+23].

1.1 Related Work

In the following, we give a brief overview of the history of mixed graph coloring as well as graph coloring algorithms.

Mixed Graph Coloring

The study of mixed graph colorings was initialized in 1976 by Sotskov and Tanaev [ST76], albeit in a slightly different version, requiring for each arc (u, v) that the color of u is at most the color of v. The version of mixed graph coloring studied in this thesis was introduced by Hansen et al. [HKdW97] in 1997. They showed that a proper coloring of a mixed graph exists only if the graph contains no directed cycle and provided several lower bounds on the chromatic number. Further, they generalized the Gallai-Hasse-Roy-Vitaver theorem to mixed graphs, which states that the chromatic number is the

minimum length of the longest path over all orientations without directed cycles of a graph. This theorem gives rise to a simple FPT-algorithm parameterized by the number of edges m. As the number of orientations without directed cycles is bounded by 2^m, this results in a runtime of $\mathcal{O}((a+m)2^m)$, where a is the number of arcs. This algorithm was later improved for small m to $\mathcal{O}(m^2 \cdot 2^m + ma)$ by Damaschke [Dam19].

There has also been some work on the complexity of mixed coloring on specific graph classes, such as on mixed interval graphs by Gutowski et al. [GJK$^+$23]. Furthermore, different problem variants of mixed graph coloring have been considered, such as counting or enumerating all different k-colorings. Algorithmic approaches studied to solve mixed graph coloring include ILP, branch-and-bound, and heuristics. See Sotskov [Sot20] for a survey on the state of mixed graph coloring.

While there have been some parameterized algorithms for mixed graph coloring, such as the aforementioned algorithm by Damaschke [Dam19], as well as an XP-algorithm parameterized by treewidth developed by Ries and de Werra [RdW08], to our knowledge, there have not been any *single-exponential-time algorithms*, i.e., algorithms with a runtime of $2^{\mathcal{O}(n)}$, where n is the number of vertices.

In this thesis, we therefore focus on developing single-exponential-time algorithms for the (general) mixed graph coloring problem.

Exact Algorithms for Graph Coloring

As mixed graph coloring is a generalization of graph coloring, it seems natural to generalize existing algorithms for graph coloring. Exact algorithms for graph coloring have been studied for a long time, with Christofides [Chr71] being the first to provide a non-trivial algorithm. He used the fact that there is an optimal coloring where one color class is a maximal independent set. Lawler [Law76] improved on this by using a dynamic program based on the same property, achieving a runtime of $\mathcal{O}(2.4423^n)$. This algorithm was improved by Eppstein [Epp01] to $\mathcal{O}(2.4151^n)$ by initializing the table with all 3-colorable subgraphs, using a 3-coloring algorithm by Beigel and Eppstein [BE05], as well as by using the insight that only small maximal independent sets have to be considered. Byskov [Bys04] further improved on this by initializing the table with all 4-colorable subgraphs, with a subsequent runtime of $\mathcal{O}(2.4023^n)$.

Using an inclusion–exclusion approach, Björklund et al. [BHK09] obtained an algorithm with a runtime of $\mathcal{O}^*(2^n)$, which to this day is the fastest algorithm for graph coloring. Recently, Björklund et al. [BCH$^+$25] showed that under Strassen's [Str94] Asymptotic Rank Conjecture, a conjecture about the complexity of matrix multiplication, there exists an $\mathcal{O}(1.9999^n)$-time algorithm for graph coloring. As the conjecture is yet unproven, the $\mathcal{O}^*(2^n)$-time inclusion–exclusion algorithm remains the fastest known.

For small (fixed) numbers of colors, faster algorithms are known. For 3-Coloring, the first algorithm was an $\mathcal{O}(1.4425^n)$-time algorithm by Lawler [Law76] based on maximal independent sets. Beigel and Eppstein [BE05] used the Constraint Satisfaction Problem to obtain an $\mathcal{O}(1.3289^n)$-time algorithm for 3-Coloring and an $\mathcal{O}(1.8073^n)$-time algorithm for 4-Coloring. Meijer [Mei23] recently improved the 3-coloring algorithm to $\mathcal{O}(1.3218^n)$, thereby also improving the, at the time fastest, 4-coloring algorithm

by Fomin et al. [FGS07], which used a 3-coloring algorithm as a subroutine. The currently fastest algorithm for 4-Coloring is by Wu et al. [WGJ+24], achieving a runtime of $\mathcal{O}(1.7159^n)$ using only polynomial space. Byskov [Bys04] developed polynomial-space algorithms for 5- and 6-Coloring, using two branching techniques that use algorithms for 3- and 4-Coloring. Their runtimes, however, are worse than the $\mathcal{O}^*(2^n)$-runtime from the inclusion–exclusion algorithm. For some time, for more than four colors, no faster algorithms than the inclusion–exclusion algorithm were known until Zamir [Zam21] showed that 5- and 6-Coloring can be solved in $\mathcal{O}((2-\varepsilon)^n)$ time for some $\varepsilon > 0$. He used an inclusion–exclusion approach based on that of Björkelund et al. [BHK09] to color graphs where few vertices have high degree, and made use of a 4-list-coloring algorithm by Beigel and Eppstein [BE05] for the remaining cases. For more than six colors, no algorithms with a runtime faster than $\mathcal{O}^*(2^n)$, that is, faster than the inclusion–exclusion algorithm, are currently known. See Table 1.1 for a summary of the fastest known algorithms for coloring (undirected) graphs.

Problem	Runtime	Reference
3-Coloring	$\mathcal{O}(1.3218^n)$	[Mei23]
4-Coloring	$\mathcal{O}(1.7159^n)$	[WGJ+24]
5-Coloring	$\mathcal{O}((2-\varepsilon)^n)$	[Zam21]
6-Coloring	$\mathcal{O}((2-\varepsilon)^n)$	[Zam21]
Coloring	$\mathcal{O}^*(2^n)$	[BHK09]

Tab. 1.1: Fastest known exact algorithms for coloring (undirected) graphs

Polynomial Space Most of the previously mentioned algorithms use exponential space. In practice, this can be undesirable, as computers have limited memory which limits the size of computable instances. Thus, it is of interest to consider more space-efficient algorithms even if they are asymptotically slower. Bodlaender and Kratsch [BK06] developed an $\mathcal{O}(5.283^n)$-time algorithm based on the fact that either there is an optimal coloring with a large independent set, and the remaining graph can be colored recursively, or the graph can be partitioned into two, similarly sized, halves which are colored recursively. The inclusion–exclusion algorithm by Björklund et al. [BHK09] also yielded a polynomial space variant with a runtime of $\mathcal{O}(2.2461^n)$, which relies on a fast algorithm to compute the number of independent sets in a graph. The computation of the number of independent sets was subsequently improved, with the latest improvement being by Gaspers and Lee [GL23]. Their algorithm for this problem yields an $\mathcal{O}(2.2356^n)$-time algorithm for coloring in polynomial space. See Table 1.2 for an overview of the fastest known algorithms for graph coloring in polynomial space.

Parameterization As Coloring is NP-hard, the parameterized complexity of Coloring has been studied in order to obtain algorithms that are efficient parameterized by certain parameters. For example, it is well-known that Coloring can be solved

Problem	Runtime	Reference
3-Coloring	$\mathcal{O}(1.3218^n)$	[Mei23]
4-Coloring	$\mathcal{O}(1.7159^n)$	[WGJ+24]
5-Coloring	$\mathcal{O}(2.1523^n)$	[Bys04]
Coloring	$\mathcal{O}(2.2356^n)$	[GL23]

Tab. 1.2: Fastest known exact polynomial-space algorithms for coloring (undirected) graphs

in $\mathcal{O}^*(k^{\mathrm{tw}})$ time [CFK+15], with k being the number of colors and tw the treewidth of the given graph. As in undirected graphs the chromatic number is bounded by treewidth, it follows that Coloring is FPT parameterized by treewidth. Furthermore, Lokshtanov et al. [LMS18] showed that, under the Strong Exponential Time Hypothesis (SETH), a hypothesis about the complexity of SAT with at most k literals per clause, there is no coloring algorithm with a runtime of $\mathcal{O}^*((k-\varepsilon)^{\mathrm{tw}})$ for any $\varepsilon > 0$, i.e., the $\mathcal{O}^*(k^{\mathrm{tw}})$-time algorithm is optimal. However, there are other parameters for which Coloring is not FPT. The most prominent example is the number of colors, as Coloring is already NP-hard for three colors. Further, Coloring is W[1]-hard parameterized by cliquewidth, as shown by Fomin et al. [FGLS10]. However, Kobler and Rotics [KR03] showed that Coloring is still XP parameterized by cliquewidth. Furthermore, it can be shown using Courcelle's theorem [Cou90] that Coloring is FPT parameterized by cliquewidth plus the number of colors.

1.2 Contribution

The remainder of this thesis is structured as follows. We first introduce the concepts and notations used throughout this thesis; see Chapter 2. Section 2.3 contains a more precise definition of mixed graph colorings together with some basic properties.

Following this, we arrive at the heart of this thesis, Chapter 3. We begin by generalizing a property commonly used in coloring algorithms; see Section 3.1. We use this property to adapt Lawler's [Law76] algorithm to mixed graphs; see Section 3.2. Following this, we briefly discuss why we were not able to adapt faster algorithms. Then we present a new polynomial-space algorithm; see Section 3.3. We conclude the chapter by analyzing the complexity of mixed coloring with fixed number of colors; see Section 3.4. The algorithmic results of this chapter are summarized in Table 1.3. For more detailed results on polynomial-space algorithms for fixed number of colors, see Table 3.2.

Apart from single-exponential-time algorithms, we also consider parameterized algorithms for MixedColoring in Chapter 4. We begin, in Section 4.1, by using Courcelle's theorem to obtain that MixedColoring is FPT parameterized by cliquewidth plus the number of colors. Following this, Section 4.2, we generalize the well-known $\mathcal{O}^*(k^{\mathrm{tw}})$-time and -space dynamic program for Coloring to MixedColoring, which yields that MixedColoring is XP parameterized solely by treewidth. We then generalize and tighten a bound on the chromatic number of mixed interval graphs by Gutowski

Problem	Runtime	Space	Reference
2-MixedColoring	`Linear`	`Linear`	Theorem 2.9
3-MixedColoring	$\mathcal{O}(1.3218^n)$	`Polynomial`	Theorem 3.16
4-MixedColoring	$\mathcal{O}(1.7433^n)$	`Polynomial`	Theorem 3.19
5-MixedColoring	$\mathcal{O}(2.1523^n)$	`Polynomial`	Theorem 3.19
6-MixedColoring	$\mathcal{O}(2.3218^n)$	`Polynomial`	Theorem 3.18
MixedColoring	$\mathcal{O}(2.4423^n)$	`Exponential`	Theorem 3.3
MixedColoring	$\mathcal{O}(4.8107^n)$	`Polynomial`	Theorem 3.5

Tab. 1.3: Fastest algorithms to color mixed graphs

et al. [GJK+23] to (general) mixed graphs; see Section 4.3. We show that, unlike on undirected graphs, the chromatic number is no longer bounded by the treewidth of the graph. On the positive side, we use this bound to obtain FPT-algorithms for further parameters, such as treedepth. Lastly we show that MixedColoring is W[1]-hard for several parameters, such as pathwidth or the size of a feedback vertex set; see Section 4.4. An overview of the parameterized complexity of MixedColoring is given in Figure 1.1.

We conclude the thesis with an outlook on open problems; see Chapter 5.

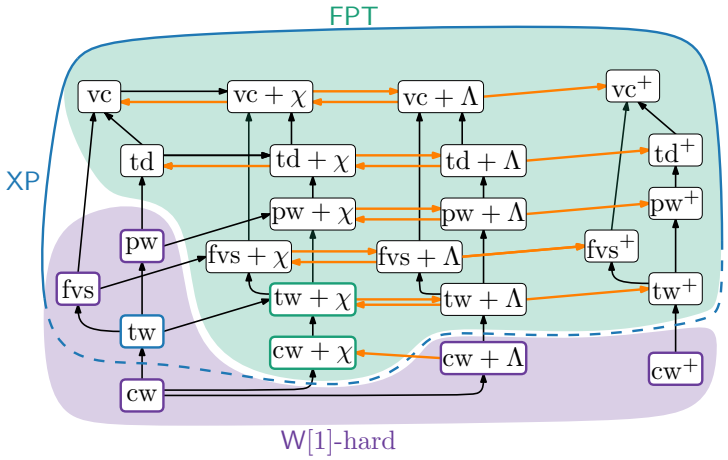


Fig. 1.1: Overview of the parameterized complexity of MixedColoring. The bold bordered parameters are the ones for which we directly show results, with the complexity of the other parameters propagating from them. The color of the rim represents the complexity class for which the result is shown, i.e., purple for W[1]-hardness, blue for XP, and green for FPT. The bold (orange) arcs represent relationships shown in this thesis. The boundary of XP is dashed at the bottom as it is yet unclear if the remaining parameterized problems are in XP. For an explanation of the parameters, see Section 2.4.

2 Preliminaries

In this chapter, we introduce some concepts of algorithmic graph theory as well as the notation used throughout this thesis. We also define mixed graph colorings and show some basic properties in Section 2.3.

2.1 Mathematical Basics

Sets We denote with $\mathbb{N}$ the set of natural numbers, $\{1, 2, \ldots\}$, and with $\mathbb{N}_0$ the set of natural numbers including 0. For the first k natural numbers, $\{1, \ldots, k\}$ we write $[k]$. For a set S, we denote with $\binom{S}{k}$ the set of all k-element subsets of S, i.e. $\{X \subseteq S \colon |X| = k\}$. We denote with S^k the set of all k-tuples of elements of S, i.e. $\{(s_1, \ldots, s_k) \mid s_i \in S\}$.

Throughout this thesis, we need to bound the number of subsets of a given size, i.e., the binomial coefficient, with an exponential function. To this end, we use the following well-know bound, where $B(\alpha) := \frac{1}{\alpha^\alpha (1-\alpha)^{(1-\alpha)}}$.

Lemma 2.1. *For any $n \in \mathbb{N}$ and $\alpha \in (0, 1)$, it holds that $\binom{n}{n\alpha} \leq B(\alpha)^n$.*

Proof. It is well-known, as seen for example in the textbook by Fomin and Kratsch [FK10, Lemma 3.13][1], that for any $n \in \mathbb{N}$ and $\alpha \in (0, 1)$ it holds that $\binom{n}{n\alpha} \leq (2^{H(\alpha)})^n$ where $H(\alpha)$ is the binary entropy function $-\alpha \log_2(\alpha) - (1 - a) \log_2(1 - \alpha)$. As it holds that $B(\alpha) = 2^{H(\alpha)}$, we obtain the desired bound. □

Functions To simplify notation, we consider functions with a finite domain, such as colorings of graphs, as sets of mappings. Thus, $\{u \mapsto 1, v \mapsto 2\}$ denotes a function $c \colon \{u, v\} \to [2]$ with $c(u) = 1$ and $c(v) = 2$. Given two functions $f \colon A \to B$ and $g \colon C \to D$, we denote with $f \cup g$ the *union of the two functions*, that is, $(f \cup g) \colon A \cup C \to B \cup D$ with $(f \cup g)(a) = f(a)$ for all $a \in A$ and $(f \cup g)(c) = g(c)$ for all $c \in C$. As long as $f(x) = g(x)$ for all $x \in A \cap C$, the union is well-defined. This notation allows us to easily augment a function. For example, given the previous function c, we can augment the function to color a new vertex w with color 1 by writing $c \cup \{w \mapsto 1\}$. As a consequence, we denote, for arbitrary sets X, with $\emptyset$ the *empty function* $f \colon \emptyset \to X$. When restricting a function $f \colon X \to Y$ to a subset X' of X, we denote the resulting function as $f|_{X'}$.

[1] The textbook seems to contain a typo as their inequality only holds for $\alpha \in (0, \frac{1}{2}]$. However, their proof suffices for the bound used here as $\binom{n}{n\alpha} = \binom{n}{n(1-\alpha)}$ and $B(\alpha) = B(1 - \alpha)$.

$\mathcal{O}^*$-Notation As we deal with exponential-time algorithms, we are mainly interested in the exponential factor, with the polynomial factor being less relevant. Therefore, we use the *$\mathcal{O}^*$-notation* to suppress polynomial factors, similar to how the $\mathcal{O}$-notation suppresses constant factors. Formally $g(n) \in \mathcal{O}^*(f(n))$ if there exists a polynomial $p(n)$ such that $g(n) \in \mathcal{O}(f(n) \cdot p(n))$. For example, $2^n \cdot n^2 \in \mathcal{O}^*(2^n)$. Note that, for exponential runtimes, it holds that $\mathcal{O}^*(c^n) \subset \mathcal{O}((c+\varepsilon)^n)$ for any $\varepsilon > 0$. Therefore, it holds that $\mathcal{O}^*(2.4422496^n) \subset \mathcal{O}(2.44225^n)$.

2.2 Mixed Graphs

An *undirected graph* G consists of a *set of vertices* $V(G)$ and a *set of edges* $E(G)$, with $E(G) \subseteq \binom{V(G)}{2}$. A *directed graph* G consists of a vertex set $V(G)$ and a *set of arcs* $A(G)$, with $A(G) \subseteq V(G)^2$. A *mixed graph* G combines the two types of graphs, consisting of a set of vertices $V(G)$, a set of edges $E(G)$, and a set of arcs $A(G)$. We require (mixed) graphs to be *simple*, i.e., for every pair of vertices u, v there is at most one edge or arc between them and there are no loops (edges or arcs from a vertex to itself). If a mixed graph consists solely of edges or solely of arcs, we may treat it as an undirected, respectively, directed graph. Conversely, we may treat an undirected graph as a mixed graph with an empty arc set, and a directed graph as a mixed graph with an empty edge set. We denote with $n(G)$ the *number of vertices*, with $m(G)$ the *number of edges*, and with $a(G)$ the *number of arcs* in G. The *empty (mixed) graph*, containing no vertices, and thus no edges or arcs, is denoted by $\emptyset$.

Throughout this thesis, we explicitly mention when a graph is mixed, while we usually refer to undirected graphs simply as graphs.

Algorithmically, we represent a mixed graph as a triple of vertices, edges, and arcs. Therefore, the *size of a mixed graph* is the number of vertices plus the number of edges and arcs. As we require mixed graphs to be simple, the size is at most quadratic in the number of vertices. Note that we can obtain the adjacency list of each vertex from the triple representation in linear time.

Incidences An arc (u, v) is directed from the *tail* u to the *head* v. The tail and the head of an arc form its *endpoints*. A vertex is *incident* to an arc of which it is an endpoint. For a vertex v, we classify the incident arcs into *incoming* arcs, where v is the head, and *outgoing* arcs, where v is the tail. In other words, an arc (u, v) is an outgoing arc for u and an incoming arc for v. For an edge $\{u, v\}$, we say that u and v are its *endpoints*. As with arcs, a vertex is *incident* to edges of which it is an endpoint.

For vertex sets, we define incidences similarly. Given a vertex set S, we say that an edge or arc is *incident to S* if exactly one of its endpoints is in S. The incident arcs are also classified into *incoming* arcs, which have their head in S, and *outgoing* arcs, which have their tail in S.

Adjacencies & Neighbors In a mixed graph G, a vertex v is *adjacent* to a vertex u if it is connected to u by an edge or an arc. The *neighborhood* of v, denoted by $N_G(v)$, is

the set of all adjacent vertices. The *outgoing neighbors* of v, denoted by $N_G^+(v)$, are all vertices u to which v has an outgoing arc (v, u). Analogously, the *incoming neighbors* of v, denoted by $N_G^-(v)$, are all vertices u from which v has an incoming arc (u, v).

Degrees The *degree* of a vertex v in a mixed graph G, denoted by $d_G(v)$, is the number of edges and arcs incident to v. The *outdegree* of a vertex v, denoted by $d_G^+(v)$, is the number of outgoing arcs. The *indegree*, denoted by $d_G^-(v)$, is the number of incoming arcs. Note that, as we require mixed graphs to be simple, the degree, outdegree, and indegree of a vertex v corresponds to the number of neighbors, outgoing neighbors, and incoming neighbors, respectively. The *maximum degree* of G, denoted by $\Delta(G)$, is the maximum degree of any vertex in G.

For a vertex set S we define the degree, $d_G(S)$, outdegree, $d_G^+(S)$, and indegree, $d_G^-(S)$, as the number of incident edges and arcs, outgoing arcs, and incoming arcs, respectively.

Paths & Cycles A *walk* $\langle v_0, \ldots, v_k \rangle$ of length k in a mixed graph G is a sequence of vertices such that $\{v_{i-1}, v_i\} \in E(G)$ or $(v_{i-1}, v_i) \in A(G)$ for each $i \in [k]$. We say that a walk is *directed*, if it consists solely of arcs, and *undirected*, if it consists solely of edges. Otherwise, it is *mixed*. A walk is *closed* if $v_0 = v_k$, otherwise it is *open*. A *path* is a walk without repeated vertices, except for the possibility of $v_0 = v_k$, i.e., the path being closed. A closed path is also called a *cycle*. We denote for $k \in \mathbb{N}$ with P_k the graph consisting of a single path of length k.

Ranks The *inrank* of a vertex v in a mixed graph G without directed cycles, denoted by $\rho_G^-(v)$, is the length of the longest directed path ending at v. The *outrank*, $\rho_G^+(v)$, is the length of the longest directed path starting at v. Combining both, the *rank*, $\rho_G(v)$, is the length of the longest directed path containing v. It holds that $\rho_G(v) = \rho_G^-(v) + \rho_G^+(v)$.

Therefore, the length of the longest directed path in G equals the *maximum rank* of any vertex in G and is denoted by $\Lambda(G)$.

Note that, since we consider mixed graphs without directed cycles, the arcs impose a strict partial order on the in- and outrank of vertices. This is formalized in the following lemma.

Lemma 2.2. *For a mixed graph G without directed cycles and an arc (u, v) of G, it holds that $\rho_G^-(u) < \rho_G^-(v)$ and $\rho_G^+(u) > \rho_G^+(v)$.*

Subgraphs We say that G' is a *subgraph* of a mixed graph G, denoted $G' \subseteq G$, if it holds that $V(G') \subseteq V(G)$, $E(G') \subseteq E(G)$, and $A(G') \subseteq A(G)$. For a subset of vertices $S \subseteq V$, we denote the *subgraph induced by* S, $\left(S, E \cap \binom{S}{2}, A \cap (S \times S)\right)$, as $G[S]$. For a set of vertices S, we write $G - S$ instead of $G[V(G) \setminus S]$. Similarly, for a set T of edges and arcs, we write $G - T$ instead of $(V(G), E(G) \setminus T, A(G) \setminus T)$. Furthermore, for a single vertex, edge, or arc x we write $G - x$ instead of $G - \{x\}$.

We generalize the notion of subgraphs by not only allowing for vertices, edges, and arcs to be removed, but also allowing arcs to be replaced by edges, resulting in *relaxed*

subgraphs. A particular relaxed subgraph is the *underlying undirected graph*, which results from replacing all arcs with edges.

The inverse of the subgraph is the supergraph. A mixed graph G' is a *supergraph* of G if G is a subgraph of G'. *Relaxed supergraphs* are defined analogously.

Orientations An *orientation* of a mixed graph G is a directed graph G' resulting from G by *orienting* each edge, i.e., turning each edge into an arc. Therefore, for every edge $\{u, v\}$ in G there is either the arc (u, v) or the arc (v, u) in G'. Thus, a mixed graph is a relaxed subgraph of its orientations.

Partitions A *proper partition* $\langle V_1, \dots, V_k \rangle$ of (the vertices of) a mixed graph G is a partition of the vertex set such that there is no arc going from a vertex in a set with a higher index to a vertex in a set with a lower index, i.e., there is no $(u, v) \in A(G)$ with $u \in V_i$, $v \in V_j$, and $i > j$. It can be proven by a simple induction that the following are equivalent characterizations:

- $d^-_G(V_1 \cup \dots \cup V_i) = 0$ for all $i \in [k]$
- $d^-_{G-(V_1 \cup \dots \cup V_{i-1})}(V_i) = 0$ for all $i \in [k]$.
- $d^-_{G[V_i \cup \dots \cup V_k]}(V_i) = 0$ for all $i \in [k]$.

See Figure 2.1 for an example of a proper and an improper partition.

(a) (b)

Fig. 2.1: A proper (a) and improper (b) partition of a mixed graph.

As we show in the following, we can *refine* proper partitions, i.e., we can substitute any set V_i of a proper partition of a mixed graph G with a proper partition of $G[V_i]$ and obtain a proper partition of G.

Lemma 2.3. *Let G be a mixed graph and $\langle V_1, \dots, V_k \rangle$ a proper partition of G. Let $i \in [k]$ and $\langle V_{i,1}, \dots, V_{i,\ell} \rangle$ be a proper partition of $G[V_i]$. Substituting V_i by $\langle V_{i,1}, \dots, V_{i,\ell} \rangle$ in $\langle V_1, \dots, V_k \rangle$ yields a proper partition of G.*

Proof. Suppose that $\langle V_1, \dots, V_{i-1}, V_{i,1}, \dots, V_{i,\ell}, V_{i+1}, \dots, V_k \rangle$ is not a proper partition of G. As we know $\langle V_1, \dots, V_k \rangle$ to be a proper partition of G, there cannot be an arc (u, v) with $u \in V_x$, $v \in V_y$, and $x > y$. Thus, there must be an arc (u, v) with $u \in V_{i,x}$, $v \in V_{i,y}$ and $x > y$. It follows that $u, v \in V_i$ and $(u, v) \in A(G[V_i])$. This contradicts $\langle V_{i,1}, \dots, V_{i,\ell} \rangle$ being a proper partition of $G[V_i]$. Therefore, $\langle V_1, \dots, V_{i-1}, V_{i,1}, \dots, V_{i,\ell}, V_{i+1}, \dots, V_k \rangle$ is a proper partition of G. □

Properties of Underlying Undirected Graphs

While the previous properties are defined for mixed graphs, the following properties are (mostly) defined for undirected graphs. We apply them to mixed graphs by considering them w.r.t. the underlying undirected graph, where all arcs are replaced by edges.

Connectivity A graph is *connected*, if there is a path between every pair of vertices. A graph is *disconnected* otherwise. A *connected component* is a maximal connected subgraph.

Cliques & Independent Sets A *clique* is a set of vertices such that there is an edge between every pair of vertices. We denote with $\omega(G)$ the size of the largest clique in a graph G. A graph is *complete* if its vertex set forms a clique. We denote with K_n the complete graph with n vertices.

The opposite of a clique is an *independent set*, i.e., a set of vertices where there is no edge between any pair of vertices. We denote with $\mathcal{I}(G)$ the family of independent sets of a graph G. An independent set is *maximal*, if it is not a proper subset of another independent set, i.e., we cannot add any vertex to the set such that the resulting set is still independent. We denote with $\mathrm{MIS}(G)$ the family of maximal independent sets of G. For a mixed graph G, we denote with $\mathcal{I}^\circ(G)$ the family of independent sets of indegree 0, i.e., $\mathcal{I}^\circ(G) = \{I \in \mathcal{I}(G) \mid d_G^-(I) = 0\}$. We denote with $\mathrm{MIS}^\circ(G)$ the family of maximal independent sets of indegree 0, i.e., $\mathrm{MIS}^\circ(G) = \{I \in \mathcal{I}^\circ(G) \mid \forall I' \in \mathcal{I}^\circ(G) \colon I \subsetneq I'\}$. Clearly, it holds that $\mathcal{I}^\circ(G) = \mathcal{I}(G[\mathcal{V}^\circ(G)])$ where $\mathcal{V}^\circ(G)$ is the set of vertices with indegree 0 in G. It follows that $\mathrm{MIS}^\circ(G) = \mathrm{MIS}(G[\mathcal{V}^\circ(G)])$, i.e., the maximal independent sets with indegree 0 are exactly the maximal independent sets in the (undirected) subgraph induced by the vertices with indegree 0.

It was shown by Moon and Moser [MM65] that the number of maximal independent sets in a graph with n vertices is at most $3^{n/3}$. Clearly, as $\mathrm{MIS}^\circ(G) = \mathrm{MIS}(G[\mathcal{V}^\circ(G)])$ for every mixed graph G, it follows that $|\mathrm{MIS}^\circ(G)| \leq 3^{|\mathcal{V}^\circ(G)|/3} \leq 3^{n(G)/3}$. Furthermore, this bound is tight, as seen when given a graph which is a collection of disjoint triangles. However, this bound does not take the sizes of the maximal independent sets into consideration. For the number of maximal independent sets of a given size, Eppstein [Epp01] provided the following bound.

Lemma 2.4 (Eppstein [Epp01, Theorem 2])**.** *The maximal independent sets of size less than ℓ in a graph with n vertices can be enumerated in $\mathcal{O}(3^{4\ell-n}4^{n-3\ell})$ time.*

In particular, there are at most $3^{4\ell-n}4^{n-3\ell}$ maximal independent sets of size less than ℓ in a graph with n vertices. For $n/4 \leq \ell \leq n/3$, this bound is tight. For other sizes of maximal independent sets, Byskov [Bys04] proved the following stricter bound.

Lemma 2.5 (Byskov [Bys04, Theorem 5])**.** *For every $d \in \mathbb{N}$, every graph with n vertices contains at most $d^{(d+1)\ell-n}(d+1)^{n-d\ell}$ maximal independent sets of size ℓ. All of these sets can be found within a polynomial factor of this bound. The bound is tight if $d = \lfloor n/\ell \rfloor$.*

Bipartite We say that a graph G is *bipartite* if there is a partition of the vertex set into two independent sets.

Forests & Trees[2] A graph without cycles is a *forest*. If it is connected, it is a *tree*. Vertices of degree 1 in a forest are called *leaves*.

A *rooted tree* is a tree T with a designated *root* $r \in V(T)$, inducing an ancestral relation on the vertices of T. For each non-root vertex $v \in V(T) \setminus \{r\}$, the vertex p adjacent to v in the (unique) path from r to v is called the *parent* of v, and $N_T(v) \setminus \{p\}$ the *children* of v. A vertex without children is called a *leaf*. Note that, as long as the root has a degree of at least 2 (i.e., is not a leaf itself), the leaves correspond to the leaves of a non-rooted tree. The *subtree of T rooted at v*, denoted T_v, is the subgraph of T induced by v and its descendants. The *height* of a rooted tree is the maximum distance from the root to any vertex. A *rooted forest* is a collection of rooted trees.

2.3 Colorings

A *coloring* of a mixed graph is a function that assigns a *color*, a natural number, to each vertex. A coloring c of a mixed graph G is *proper* if

- for every edge $\{u, v\} \in E(G)$ it holds that $c(u) \neq c(v)$ and
- for every arc $(u, v) \in A(G)$ it holds that $c(u) < c(v)$.

An example of a proper and improper mixed coloring can be seen in Figure 2.2. A *k-coloring* is a coloring that uses only k colors. We say that G is *k-colorable* if there exists a proper k-coloring of G. The *chromatic number*, denoted by $\chi(G)$, is the smallest k such that G is k-colorable. We say that a proper coloring consisting of $\chi(G)$ colors is *optimal*.

(a) (b)

Fig. 2.2: A proper (a) and improper (b) coloring of a mixed graph. Violated constraints, in this case two arcs and one edge, are highlighted in red.

We call the problem of deciding whether a given mixed graph is k-colorable for a given k MixedColoring. If the number of colors k is part of the problem, we obtain the k-MixedColoring problem. Note that these problems generalize the well-known Coloring and k-Coloring problems for undirected graphs, as our definition of mixed graph coloring is equivalent to the coloring definition on undirected graphs.

[2]The following definitions and notation are derived from Jaffke [Jaf20].

As a coloring is proper if it adheres to the constraints imposed by edges and arcs, it follows that a proper coloring of a mixed graph is also proper for all subgraphs. Furthermore, as the constraint imposed by an arc (u, v), i.e., that $c(u) < c(v)$, is stronger than the constraint imposed by the edge $\{u, v\}$, which only requires that $c(u) \neq c(v)$, it follows that a proper coloring is even proper for all relaxed subgraphs. We formalize this insight in the following lemma.

Lemma 2.6. *Let G be a mixed graph, and let c be a proper coloring of G. If G' is a relaxed subgraph of G, then $c|_{V(G')}$ is a proper coloring of G'.*

Thus, the chromatic number of a mixed graph is lower bounded by the chromatic number of its relaxed subgraphs, such as by the chromatic number of the underlying undirected graph.

As an arc requires the head to receive a higher color than the tail, it follows that vertices in a directed path must have increasing colors in a proper coloring. Therefore, we can use the in- and outrank to bound the colors a vertex can receive, as formalized in the following lemma.

Lemma 2.7. *Given a proper k-coloring c of a mixed graph G, it holds for every vertex v of G that $\rho_G^-(v) + 1 \leq c(v) \leq k - \rho_G^+(v)$.*

Using the same argument, it follows that a mixed graph does not admit a proper coloring if it contains a directed cycle. A mixed graph without directed cycles always admits a proper coloring, as observed by Hansen et al. [HKdW97, Proposition 1].

Since we can test in linear time whether a given mixed graph contains a directed cycle, e.g., using depth-first search, we assume for the remainder of this thesis that we are given mixed graphs without directed cycles. Furthermore, as we can color connected components separately, we also assume mixed graphs to be connected.

Given a k-coloring c of a mixed graph G, the *color classes* of c are $\langle V_1, \ldots, V_k \rangle$ where $V_i = \{v \in V(G) \mid c(v) = i\}$ for $i \in [k]$. As shown in the following, the color classes of a proper coloring correspond to a proper partition into independent sets.

Lemma 2.8. *A proper k-coloring c of a mixed graph G is equivalent to a proper partition of G into independent sets $\langle V_1, \ldots, V_k \rangle$.*

Proof. Let c be a proper k-coloring of G and $\langle V_1, \ldots, V_k \rangle$ be its color classes. Since c is proper, for each edge $\{u, v\} \in E(G)$ it holds that $c(u) \neq c(v)$. Thus, each color class is an independent set. Further, we know that each arcs points towards the endpoint with the higher color, which is therefore in a higher color class. Thus, $\langle V_1, \ldots, V_k \rangle$ is a proper partition.

Given a proper partition of G into independent sets $\langle V_1, \ldots, V_k \rangle$, we construct a coloring c by setting $c(v) = i$ for all $v \in V_i$. Since each color class is an independent set, for each edge $\{u, v\} \in E(G)$ it holds that $c(u) \neq c(v)$. Further, as the partition is proper, for each arc $(u, v) \in A(G)$ it holds that $u \in V_i$, $v \in V_j$ with $i < j$. Thus, it is $c(u) = i < j = c(v)$. Therefore, c is a proper k-coloring of G. □

Proper colorings being equivalent to proper partitions into independent sets highlights the fact that, contrary to colorings of undirected graphs, we cannot simply permute the colors of a proper coloring, as this may violate the constraints imposed by arcs.

Decision vs. Optimization vs. Search As with other computational problems, mixed graph colorings can be considered as a decision, optimization, or search problem. So far, we mentioned only the decision problem, where we ask whether a mixed graph is k-colorable, and the optimization problem, where we ask for the chromatic number. As an algorithm that determines the chromatic number also solves the decision problem, most of our algorithms will compute the chromatic number. The exceptions are algorithms that directly exploit the number of possible colors, such as the algorithms in Section 3.4 and the treewidth algorithm in Section 4.2. However, an algorithm for the decision problem can be used to compute the chromatic number via a linear search, i.e., by iteratively increasing the number of available colors k until the graph is k-colorable. This does not work, however, for the algorithms in Section 3.4 as they are designed for fixed k (and not for all k). Note that the runtime of a chromatic-number algorithm repurposed to solve the decision problem could be improved by making use of the number of colors k, e.g., by bounding the depth of the recursion (Algorithm 3) or the number of iterations (Algorithm 2). However, for small k, the dedicated algorithms from Section 3.4 are more efficient, and for larger k, the (asymptotic) improvements are not substantial.

The search problem asks for a proper k-coloring of a mixed graph, not just whether such a coloring exists. For the sake of simplicity, our algorithms do not return a coloring. However, except for the algorithms in Section 3.4, we explain how our algorithms can be modified to return a coloring without altering the time or space requirements.

List Coloring

While there are many variants of (undirected) colorings, for this thesis we only make use of one of them, namely ListColoring. Given an (undirected) graph G and a list of colors $L(v)$ for each vertex v of G, the ListColoring problem asks whether there is a proper coloring c of G with $c(v) \in L(v)$ for each $v \in V(G)$. This problem generalizes Coloring, as we can give each vertex the same color list $[k]$, where k is the number of colors in the Coloring instance.

In the k-ListColoring problem, the color lists are restricted to be subsets of $[k]$. For ListColoring, we can assume w.l.o.g. that given a graph with n vertices as input, all color lists are subsets of $[n^2]$. Indeed, we can assume w.l.o.g. that each color list contains at most n colors, as a vertex with n possible colors can always be properly colored. Thus, we can assume w.l.o.g. that the number of colors contained in the color lists is at most n^2, and, since we can remove unused colors, that the color lists are subsets of $[n^2]$.

Note that ListColoring generalizes naturally to mixed graphs by taking a mixed graph as input and requiring the coloring to be a proper coloring of the mixed graph.

Polynomial Cases of MixedColoring

It is well-known that Coloring is polynomially solvable for at most two colors, and NP-hard for at least three colors. Due to MixedColoring being a generalization of Coloring, it is thus also NP-hard for at least three colors. We now show that for at most two colors, MixedColoring is, like Coloring, solvable in polynomial time.

Theorem 2.9. *For every mixed graph G it holds that:*

- *G is 0-colorable if and only if G is the empty graph,*
- *G is 1-colorable if and only if G has no edges or arcs, and*
- *G is 2-colorable if and only if there is a proper partition of G into two independent sets.*

Each of these conditions can be checked in linear time, and thus k-colorability of mixed graphs can be decided in linear time for $k \leq 2$.

Proof. The empty graph contains no vertices and thus does not need any colors. Furthermore, each graph with at least one vertex needs to color said vertex and thus needs at least one color. Checking whether a graph is empty is done in constant time.

A graph without edges and arcs is an independent set, and can thus be colored with a single color. An edge or arc necessitates a second color, as the endpoints of said edge or arc need to be colored differently from each other. Checking whether a graph has no edges or arcs can be done in constant time.

As shown in Lemma 2.8, G is 2-colorable if and only if it has a proper partition into two independent sets. It remains to show that we can check the existence of such a partition in linear time. For G to have a proper partition into two independent sets, it needs to be bipartite. It is well-known that we can check whether a graph is bipartite in linear time using algorithms such as depth-first search. Furthermore, it is well-known that a connected graph has only one bipartite partition (up to permutation). Thus, for a proper partition into two independent sets to exist, all arcs have to be oriented equally w.r.t. the two sets of the bipartite partition, i.e., one set has to have indegree 0 and the other outdegree 0. Checking the in- and outdegree of two sets is possible in linear time. □

2.4 Parameterization

Parameterization is a technique to analyze the complexity of (NP-hard) problems w.r.t. certain parameters with the goal of solving the problems efficiently for small values of the parameters. In this section, we first introduce the basic concepts of parameterized complexity. Following this, we define the parameters used throughout this thesis. More details on parameterized complexity can be found in standard textbooks, such as by Downey and Fellows [DF13].

Complexity

An instance (I, k) of a *parameterized problem* consist of an instance I of a decision problem and a parameter $k \in \mathbb{N}_0$. The size of such an instance is $|I| + k$, i.e., the parameter is encoded in unary.

A parameterized problem is *fixed-parameter tractable* (FPT) if there is an algorithm that solves the problem in $f(k) \cdot (|I| + k)^{\mathcal{O}(1)}$ time, where f is a computable function. We refer to such a runtime as FPT w.r.t. k. Expressed using the $\mathcal{O}^*$-notation, a runtime is FPT w.r.t. k if it is in $\mathcal{O}^*(f(k))$ for a computable function f.

A parameterized problem is *slice-wise polynomial* (XP) if there is an algorithm that solves the problem in $\mathcal{O}(n^{f(k)})$ time, where f is a computable function. Clearly, if a problem is FPT w.r.t. a parameter it is also XP w.r.t. the same parameter.

In order to show that a problem is in XP or FPT, it suffices to show that there is an algorithm that solves the problem in the respective time. To show that a problem is (likely) not FPT or XP, one can show that the problem is already NP-hard for a small values of the parameter, such as Coloring being already NP-hard for three colors. However, some problems admit an XP-algorithm while likely not having an FPT-algorithm, such as Coloring w.r.t. cliquewidth. In order to show that a problem likely does not admit an FPT-algorithm, we can show that it is W[1]-hard. For this thesis, it is sufficient to know that W[1] lies between FPT and XP, and that it is assumed that FPT $\neq$ W[1] (similar to the assumption that P $\neq$ NP). Thus, a parameterized problem that is W[1]-hard is assumed to not be in FPT.

To show that a parameterized problem is W[1]-hard, one can perform a *parameterized reduction* from a known W[1]-hard problem. A parameterized reduction transforms an instance (I, k) of a parameterized problem into an equivalent instance (I', k') of another parameterized problem such that $k' \leq g(k)$, g is a computable function, and the reduction is performed in FPT-time.

Parameters

The natural parameter of MixedColoring is the number of colors. However, as MixedColoring is already NP-hard for three colors, it is not XP (nor FPT) w.r.t. the number of colors. Thus, we consider the complexity of MixedColoring w.r.t. other parameters. Unless stated otherwise, the following parameters are defined for undirected graphs, but can be applied to mixed graphs by considering the underlying undirected graph. An overview of the already established relations between the parameters can be found in the work of Tran [Tra22].

Vertex Cover A *vertex cover* of a graph G is a set of vertices C, such that every edge $\{u, v\} \in E(G)$ has at least one endpoint in C. The *vertex cover number* of G, denoted by $\mathrm{vc}(G)$, is the size of the smallest vertex cover of G.

Tree- and Pathwidth[3] A *tree decomposition* of a graph G is a pair $\mathcal{T} = (T, \mathcal{B})$ with $\mathcal{B} = \{B_t\}_{t \in V(T)}$, where T is a tree whose every node t is assigned a subset of vertices B_t, called *bag*, such that the following conditions hold:

- $\bigcup_{t \in V(T)} B_t = V(G)$, i.e., each vertex is in at least one bag.
- For every edge $\{u, v\} \in E(G)$ there is a node $t \in V(T)$ such that $u, v \in B_t$.
- For every vertex $v \in V(G)$, the set of nodes containing v, $\{t \in V(T) \mid v \in B_t\}$, induces a connected subtree of T.

The *width of a tree decomposition* $\mathcal{T} = (T, \mathcal{B})$ is $\max_{t \in V(T)} |B_t| - 1$, i.e., the maximum size of its bags minus 1. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum possible width of a tree decomposition of G.

A *path decomposition* is a special case of a tree decomposition $\mathcal{T} = (T, \mathcal{B})$, where we restrict T to be a path. The *pathwidth* of a graph G, denoted by $\mathrm{pw}(G)$, is the minimum possible width of a path decomposition of G.

A common approach to show that a problem is FPT w.r.t. treewidth is to provide a dynamic program that uses the tree decomposition to solve the problem. To simplify such a program, it is useful to use the following special form of tree decompositions. A *nice tree decomposition* of a graph G is a tree decomposition $(T, \mathcal{B})$ such that T is a rooted tree with the following node types:

Leaf Each leaf t of T is a *leaf node* with empty bag, i.e. $B_t = \emptyset$.

Introduce A node t of T is an *introduce node* if it has exactly one child t' and it holds that $B_t = B_{t'} \cup \{v\}$ for some vertex v of G. We say that *v is introduced at t.*

Forget A node t of T is a *forget node* if it has exactly one child t' and it holds that $B_t = B_{t'} \setminus \{v\}$ for some vertex v of G. We say that *v is forgotten at t.*

Join A node t of T is a *join node* if it has exactly two children t_1, t_2 and it holds that $B_t = B_{t_1} = B_{t_2}$.

We further require the root to have an empty bag.

It is well-known [CFK+15, Lemma 7.4], that every tree decomposition can be transformed into a nice tree decomposition in polynomial time, and that the size of the resulting nice tree decomposition is polynomial in the size of the graph G, more precisely the size is in $\mathcal{O}(\mathrm{tw}(G) \cdot n(G))$. It is long known that the treewidth and a corresponding tree decomposition of a graph can be computed in FPT-time w.r.t. treewidth, with Bodlaender [Bod96] being the first to provide such an algorithm, with recent improvements by Korhonen and Lokshtanov [KL23]. When constructing an FPT-algorithm w.r.t. treewidth, we can therefore assume that we are given a nice tree decomposition of the graph.

For each node $t \in V(T)$, let V_t be the set of vertices contained in bags of the subtree rooted at t, i.e., $V_t = \bigcup_{t \in V(T_t)} B_t$, and let $G_t = G[V_t]$.

[3] The following definitions and notation are taken from Cygan et al. [CFK+15].

Nice tree decompositions have some well-known properties, which we summarize in the following.

Lemma 2.10. *Let $(T, \mathcal{B})$ be a nice tree decomposition of a graph G. The following holds:*

- *For an introduce node t with child t', the edges contained in G_t but not $G_{t'}$ are incident to the vertex v introduced at t. Thus, all edges in G_t are contained in $G_{t'}$ or $G[B_t]$, and there is no edge between v and a vertex in $V_t \setminus B_t$.*
- *For a join node t with children t_1 and t_2, there is no edge between a vertex in $V_{t_1} \setminus B_t$ and a vertex in $V_{t_2} \setminus B_t$. That is, each edge in G_t is contained in G_{t_1} or G_{t_2}.*

While many problems are FPT w.r.t. treewidth, like Coloring, treewidth has the drawback of only being small for sparse graphs. Thus, more general measures have been introduced, which can also be small for dense graphs. One such measure is cliquewidth, for which Coloring is known to be XP.

Cliquewidth Courcelle and Olariu [CO00] define cliquewidth for directed and undirected graphs. Combining their definitions, we obtain the following definition for cliquewidth on mixed graphs. The *cliquewidth* of a mixed graph G, denoted by $\mathrm{cw}(G)$, is the minimum number of distinct labels needed to construct G using the following operations:

Introduce: creation of a new vertex with label i, denoted i

Union: disjoint union of two graphs, denoted $\dot\cup$

Relabel: changing all labels i to j, denoted $\rho_{i\to j}$

Join: connect all vertices labeled i to all vertices labeled j with edges, denoted $\eta_{i,j}$

Arc: connect all vertices labeled i to all vertices labeled j with arcs, denoted $\alpha_{i,j}$

For example, the directed path P_3 has cliquewidth 3. It is constructed by the expression $\alpha_{2,3}(\rho_{3\to 2}(\rho_{2\to 1}(\alpha_{2,3}(\alpha_{1,2}(1\dot\cup 2)\dot\cup 3)))\dot\cup 3)$. Meanwhile, the oriented K_4 without directed cycles, constructed by $\alpha_{1,2}(\rho_{2\to 1}(\alpha_{1,2}(\rho_{2\to 1}(\alpha_{1,2}(1\dot\cup 2))\dot\cup 2))\dot\cup 2)$, has cliquewidth 2. Thus, adding arcs can decrease the cliquewidth of a graph.

To more accurately determine the boundary between FPT and W[1]-hard parameters for MixedColoring, we also consider the following three parameters.

Feedback Vertex Set A *feedback vertex set* of a graph G is a set of vertices F such that $G - F$ contains no cycle. The size of the smallest feedback vertex set of G is denoted by $\mathrm{fvs}(G)$.

Treedepth The *treedepth* of a graph G, denoted by $\mathrm{td}(G)$, is the minimum height of a rooted forest such that for every edge $\{u, v\} \in E(G)$, the vertices u and v have an ancestor–descendant relationship in the forest.

Recall that MixedColoring can be used to model scheduling problems with precedence constraints. Therefore, it is plausible that, when given a mixed graph modeling a scheduling problem, the graph is *transitively closed*. That is, for each pair of vertices u, v with a directed path from u to v there is the arc (u, v). As we show, the parameterized complexity of MixedColoring is different for transitively closed graphs. Furthermore, as we show in the following, we can relate mixed graphs to transitively closed graphs.

Transitive Closure The *transitive closure* of a mixed graph G, denoted by G^+, is the smallest transitively closed relaxed supergraph of G. This graph is obtained by adding the arc (u, v) for every pair of vertices $u, v \in V(G)$ that has a directed path from u to v in G. If G already contains the edge $\{u, v\}$, the arc (u, v) replaces this edge, as we require mixed graphs to be simple. Hence, G^+ is a relaxed supergraph of G. Therefore, as we assume that G contains no directed cycles, the transitive closure of G is a simple mixed graph without directed cycles. Thus, G^+ admits a proper coloring. Further, as a directed path from u to v in G implies that u receives a smaller color than v in any proper coloring of G, it holds that a coloring that is proper for G is also proper for G^+. The inverse holds due Lemma 2.6 and G being a relaxed subgraph of G^+. Thus, G and G^+ have the same chromatic number. We can therefore study the parameterized complexity of MixedColoring on the transitive closure of mixed graphs.

Note that, except for cliquewidth, the value of the parameters defined in this section do not increase when considering a relaxed subgraph, as stated in the following.

Lemma 2.11. *For every parameter $\alpha \in \{\mathrm{vc}, \mathrm{td}, \mathrm{fvs}, \mathrm{pw}, \mathrm{tw}\}$, it holds that $\alpha(G') \leq \alpha(G)$ for every mixed graph G and relaxed subgraph G' of G. In particular, it holds that $\alpha(G) \leq \alpha(G^+)$.*

The reason this does not hold for cliquewidth is that adding arcs or edges can decrease the cliquewidth. Recall that a directed P_3 has a cliquewidth of 3, while its transitive closure, the oriented K_4 without directed cycles, has a cliquewidth of 2.

Note that, except for cliquewidth, the value of these parameters in the transitive closure can be arbitrarily bigger than in the original graph, as seen for example with the family of oriented stars where the center vertex has half incoming and half outgoing arcs.

For a mixed graph G, we denote with $\mathrm{vc}^+(G)$ the vertex cover number of G^+, i.e., $\mathrm{vc}(G^+)$. The parameters $\mathrm{td}^+(G)$, $\mathrm{fvs}^+(G)$, $\mathrm{pw}^+(G)$, $\mathrm{tw}^+(G)$, and $\mathrm{cw}^+(G)$ are defined analogously.

3 Coloring Mixed Graphs

In this chapter, we consider various algorithms for solving MixedColoring. Several of these algorithms use the same property, which we show in the following section.

3.1 Canonical Colorings

A property used by several coloring algorithms, such as the ones by Christofides [Chr71], by Lawler [Law76], or by Byskov [Bys04], is that there exists an optimal coloring where one color class is a maximal independent set. This property, however, does not hold for mixed graphs, as shown in Figure 3.1.

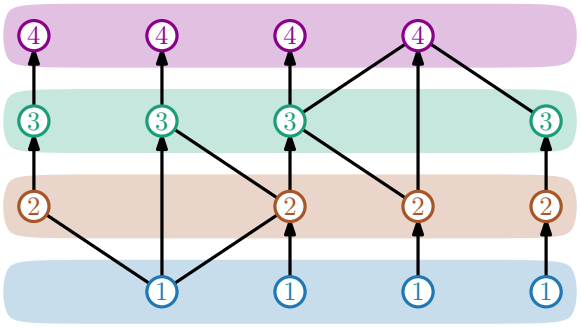


Fig. 3.1: A uniquely 4-colorable mixed graph, where no color class is a maximal independent set.

However, when we restrict ourselves to maximal independent sets with indegree 0, the property does holds for mixed graphs. As in undirected graphs all vertices have indegree 0, this is even a generalization of the property for undirected graphs. In fact, we can further strengthen the property by requiring the first color class to be a maximal independent set with indegree zero. This results in the following recursive formula for the chromatic number of mixed graphs.

Lemma 3.1. *For a mixed graph G, it holds that*

$$\chi(G) = \begin{cases} 0 & \text{if } G = \emptyset, \\ 1 + \min_{I \in \mathrm{MIS}^\circ(G)} \chi(G - I) & \text{otherwise.} \end{cases}$$

Proof. The empty graph $\emptyset$ can be colored with zero colors. If G is not empty, it contains at least one vertex, and thus at least one maximal independent set with indegree 0.

We first show that $\chi(G) \leq 1+\chi(G-I)$ for every $I \in \mathrm{MIS}^\circ(G)$. Let $k = \chi(G-I)$, and let $\langle V_2, \ldots, V_{k+1}\rangle$ be the color classes of an optimal coloring of $G-I$. As $I \in \mathrm{MIS}^\circ(G)$, I has no incoming arcs, and thus $\langle I, V_2, \ldots, V_{k+1}\rangle$ forms the color classes of a proper $(k+1)$-coloring of G.

It remains to show that there is an $I \in \mathrm{MIS}^\circ(G)$ such that $\chi(G) \geq 1+\chi(G-I)$. Let $\langle V_1, \ldots, V_k\rangle$ be the color classes of an optimal coloring of G. As $V_1 \in \mathcal{I}^\circ(G)$, there is a superset $V_1' \supseteq V_1$ with $V_1' \in \mathrm{MIS}^\circ(G)$. As $G - V_1' \subseteq G - V_1$, it follows from Lemma 2.6 that $G - V_1'$ is $(k-1)$-colorable, and thus $\chi(G) = k \geq 1 + \chi(G - V_1')$. □

The colorings obtained using this formula are of a special form: we say that a proper coloring with color classes $\langle V_1, \ldots, V_k\rangle$ is *canonical* if it holds for each color class that $V_i \in \mathrm{MIS}^\circ(G-(V_1 \cup \cdots \cup V_{i-1}))$. It follows from the formula that there always exists an optimal canonical coloring. However, note that a canonical coloring is not necessarily optimal.

A Simple Algorithm

Implementing the formula directly as a branching algorithm, we obtain a polynomial-space algorithm, Algorithm 1, which computes the chromatic number by enumerating all canonical colorings.

Algorithm 1: branching algorithm enumerating canonical colorings

```
EnumerateCanonical(G):
    if G = ∅ then
        return 0
    return 1 + max_{I∈MIS°(G)} EnumerateCanonical(G − I)
```

Unfortunately, a mixed graph can have a lot of canonical colorings, as we show in the following.

Observation 3.2. *A mixed graph with n vertices can have $n!$ canonical colorings. Therefore, Algorithm 1 has a worst-case runtime of $\Omega(n!)$.*

Proof. We show that the complete graph K_n has $n!$ canonical colorings. First, notice that, since K_n is undirected, $|\mathrm{MIS}^\circ(K_n)| = |\mathrm{MIS}(K_n)| = n$, with each maximal independent set being a single vertex. Thus, when the K_n is the input of Algorithm 1, each recursive call gets the K_{n-1} as input. The total number of canonical colorings is thus $T(n) = n \cdot T(n-1)$ with $T(0) = 1$, which results in $T(n) = \prod_{i=1}^{n} i = n!$. □

The fact that the K_n has $n!$ canonical colorings may come as a surprise, as the K_n has, up to permutation, only one coloring. However, recall that in mixed colorings, we also consider the order of the colors, and thus arrive at $n!$ canonical colorings. While this distinction is not relevant for undirected graphs, it becomes relevant as soon as arcs are present. We can, in fact, construct a mixed graph with $2n$ vertices and $n!$ canonical colorings by stacking two K_n on top of each other, with each vertex in the lower K_n having an arc to the corresponding vertex in the upper K_n. Permuting the color classes of a canonical coloring of this graph would lead to an improper coloring.

3.2 Exponential Space

With the canonical colorings in hand, we can generalize the algorithm by Lawler [Law76] to mixed graphs. Our implementation differs from the one by Lawler in that we do not iterate over all induced subgraphs, instead incrementally constructing i-colorable induced subgraphs for increasing i. This allows us to stop as soon as we encounter an optimal coloring.

Algorithm 2: dynamic program based on canonical colorings

```
LawlerColoring(G):
    L_0 = {∅}
    for i = 1 to n do
        L_i = ∅
        foreach S ∈ L_{i-1} do
            foreach I ∈ MIS°(G − S) do
                L_i ← L_i ∪ {S ∪ I}
                if S ∪ I = V(G) then
                    return i
    return 0
```

Theorem 3.3. *The chromatic number of mixed graphs can be computed in $\mathcal{O}(2.44225^n)$ time and $\mathcal{O}(2^n n)$ space.*

Proof. We show that Algorithm 2 has the desired properties. Let G be a mixed graph with n vertices. We begin by showing the correctness of the algorithm, followed by the time and space complexity.

Correctness If the graph is empty, the algorithm returns 0. Otherwise, let k be the value returned by the algorithm. The algorithm stops and returns k as soon as it creates the set $V(G)$. Further, in this case it holds that $V(G) \in L_k$. As we show in the following, it follows that $G[V]$ is k-colorable, and thus that $\chi(G) \leq k$.

Claim. For each $S \in L_i$, the graph $G[S]$ is i-colorable.

Proof. We prove this claim by induction over i, showing that for each $S \in L_i$, the graph $G[S]$ is i-colorable and $d_G^-(S) = 0$.

Base Case If $i = 0$, the empty graph $G[\emptyset]$ is 0-colorable and $d_G^-(\emptyset) = 0$.

Induction Step Let $i > 0$ and $S \in L_i$. Assume that for each $S' \in L_{i-1}$ it holds that $G[S']$ is $(i-1)$-colorable and $d_G^-(S') = 0$. By construction of L_i, there is an $S' \in L_{i-1}$ and $I \in \mathrm{MIS}^\circ(G - S')$ such that $S = S' \cup I$. As $d_G^-(S') = 0$, there is no arc from I to S'. We can thus color S' with the $(i-1)$ smallest colors, and I with the i-th color, obtaining a proper i-coloring of $G[S' \cup I] = G[S]$. Furthermore, as $d_{G-S'}^-(I) = 0$, it follows that there is no arc from a vertex in $V(G) \setminus S$ to a vertex in S, i.e. $d_G^-(S) = 0$. ◁

In the following, we show that $V(G) \in L_{\chi(G)}$, which implies that $k \leq \chi(G)$. For this, we use the fact from Lemma 3.1 that G has an optimal canonical coloring.

Claim. Let $\langle V_1, \ldots, V_{\chi(G)} \rangle$ be the color classes of an optimal canonical coloring of G. For each $i \in [\chi(G)]$, it holds that $V_1 \cup \cdots \cup V_i \in L_i$.

Proof. We show that $V_1 \cup \cdots \cup V_i \in L_i$ by induction over i.

Base Case For $i = 0$, the empty union (i.e. the empty set) is in L_0.

Induction Step Let $i > 0$ and assume that $V_1 \cup \cdots \cup V_{i-1} \in L_{i-1}$. As it holds that $V_i \in \mathrm{MIS}^\circ(G - (V_1 \cup \cdots \cup V_{i-1}))$ by the definition of canonical colorings, the algorithms adds the set $(V_1 \cup \cdots \cup V_{i-1}) \cup V_i$ to L_i. ◁

As we have shown that $\chi(G) \leq k$ and $k \leq \chi(G)$, it follows that the algorithm computes the chromatic number.

Runtime To save runtime, we implement each L_i as a balanced binary search tree with each set S being a bitvector of length n, where the j-th bit is set if the j-th vertex is contained in S. Thus, remembering which bits were already compared when descending the binary tree, inserting a set S into L_i takes $\mathcal{O}(n)$ time.

Let $T(i)$ be the runtime of the i-th iteration of the outer for-loop. The runtime of the algorithm is then $\sum_{i=1}^{n} T(i)$. In the following, we show that $T(i) \in \mathcal{O}^*((1 + \sqrt[3]{3})^n)$ for $i \in [n]$.

As the innermost for-loop has a linear runtime, it suffices to count its iterations. For each $S \in L_{i-1}$, we know that $|\mathrm{MIS}^\circ(G - S)| \leq 3^{(n-|S|)/3}$. Thus, it holds for $T(i)$ that:

$$\begin{aligned} T(i) &= \sum_{S \in L_{i-1}} |\mathrm{MIS}^\circ(G - S)| \leq \sum_{S \in 2^V} 3^{(n-|S|)/3} = \sum_{i=0}^{n} \binom{n}{i} 3^{(n-i)/3} \\ &= \sum_{i=0}^{n} \binom{n}{i} 3^{i/3} = (1 + \sqrt[3]{3})^n \end{aligned}$$

Therefore, the runtime of the algorithm is in $\mathcal{O}^*((1 + \sqrt[3]{3})^n) \subset \mathcal{O}(2.44225^n)$.

Space It holds for each $i \in [n]$ that $L_i \subset 2^{V(G)}$. Further, each entry of L_i is a bitvector of length n. Thus, the space consumption of L_i is at most $2^n n$. Furthermore, in the i-th iteration of the outer for-loop we only need L_i and L_{i-1}. Therefore, the total space consumption is $\mathcal{O}(2^n n)$. □

In order to obtain an optimal coloring, we can modify the algorithm by having each entry $S \in L_i$ be a vector of numbers instead of a bitvector, where the j-th entry is the color of the j-th vertex, with 0 being used to indicate that the j-th vertex is not contained in the set S. Assuming that our numbers can be stored and compared in constant space and time, this modification does not change the runtime or space requirements of the algorithm. Furthermore, the coloring returned by the algorithms is even an optimal canonical coloring.

Problems Generalizing Faster Algorithms While we were able to generalize the algorithm by Lawler, we did not manage to generalize faster algorithms. The algorithm by Eppstein [Epp01] is a modification of Lawler's algorithm, where the dynamic program is initialized with all 3-colorable subgraphs, and only maximally independent sets of size at most $|S|/(i-1)$ are considered. The initialization of the dynamic program is performed using a fast algorithm for 3-Coloring, which we were able to generalize to mixed graphs; see Theorem 3.16. However, we were not able to generalize the property that it suffices to consider maximally independent sets of size at most $|S|/(i-1)$, as this property relies on the fact that the color classes of a proper coloring of an undirected graph can be ordered arbitrarily, in this case by decreasing size. The color classes of a proper coloring of a mixed graph however, cannot be ordered arbitrarily, due to arcs. As Byskov [Bys04] just modified the algorithm of Eppstein by initializing the dynamic program with all 4-colorable subgraphs, we were also not able to generalize this algorithm.

The inclusion–exclusion algorithm by Björkelund et al. [BHK09] uses the fact that coloring the vertices corresponds to covering the vertices with, potentially overlapping, independent sets. However, mixed coloring, again, has to account for the order of the independent sets, and counting the number of covers adhering to a certain order seems far more difficult. Our attempts lead to runtimes far worse than the algorithm by Lawler and are thus omitted. Therefore, we were also not able to generalize algorithms based on inclusion–exclusion, such as Zamir's algorithms [Zam21] for 5- and 6-Coloring.

3.3 Polynomial Space

As the algorithm in the previous section uses exponential space, which can be undesirable, we now present a divide-and-conquer branching algorithm that uses only polynomial space. For that, we need the following lemma, which allows us to properly partition a mixed graph into two subgraphs of bounded size that can be colored recursively.

Lemma 3.4. *For an $\alpha \in (0,1)$, a non-empty mixed graph G with n vertices, and each proper partition $\langle S, I, T\rangle$ of G such that $|S| < n\alpha$, $|T| \leq n(1-\alpha)$, and $I \in \mathrm{MIS}^\circ(G-S)$, it holds that $\chi(G) \leq \chi(G[S]) + 1 + \chi(G[T])$. Furthermore, for at least one of these partitions, it holds that $\chi(G) = \chi(G[S]) + 1 + \chi(G[T])$.*

Proof. First off, we show that $\chi(G) \leq \chi(G[S]) + 1 + \chi(G[T])$ for each such partition $\langle S, I, T\rangle$. Let $\langle S_1, \ldots, S_{\chi(G[S])}\rangle$ be the color classes of an optimal coloring of $G[S]$ and $\langle T_1, \ldots, T_{\chi(G[T])}\rangle$ be the color classes of an optimal coloring of $G[T]$. As I is an independent set, it follows by Lemma 2.3 that $\langle S_1, \ldots, S_{\chi(G[S])}, I, T_1, \ldots, T_{\chi(G[T])}\rangle$ is a proper partition of G into $\chi(G[S]) + 1 + \chi(G[T])$ independent sets, and thus that $\chi(G) \leq \chi(G[S]) + 1 + \chi(G[T])$.

We now show that $\chi(G) = \chi(G[S]) + 1 + \chi(G[T])$ holds for at least one of these partitions. Let $\langle V_1, \ldots, V_k\rangle$ be the color classes of an optimal canonical coloring of G. We know due to Lemma 3.1 that an optimal canonical coloring exists. Let i be biggest such that $|\bigcup_{j=1}^{i-1} V_j| < n\alpha$. We then set $S = \bigcup_{j=1}^{i-1} V_j$, $I = V_i$, and $T = \bigcup_{j=i+1}^{k} V_j$.

By construction, we have $|S| < n\alpha$ and $|S \cup I| = |\bigcup_{j=1}^{i} V_j| \geq n\alpha$. Thus, it holds that $|T| \leq n - |S \cup I| \leq n(1-\alpha)$. Furthermore, as $I = V_i$ for an $i \in [\chi(G)]$, it follows by the definition of canonical colorings that $I \in \mathrm{MIS}^\circ(G - (V_1 \cup \cdots \cup V_{i-1})) = \mathrm{MIS}^\circ(G - S)$. Thus, $\chi(G[S]) \leq i-1$, $\chi(G[T]) \leq \chi(G) - i$, and $\chi(G) \geq \chi(G[S]) + 1 + \chi(G[T])$. With the upper bound from the previous paragraph, it follows that $\chi(G) = \chi(G[S]) + 1 + \chi(G[T])$. □

Algorithm 3: polynomial-space divide-and-conquer algorithm for MixedColoring

```
PolySpaceColor(G):
    if G = ∅ then
        return 0
    x ← ∞
    foreach S ⊂ V(G) with |S| ≤ ⌈nα − 1⌉ and d⁻_G(S) = 0 do
        x_1 = PolySpaceColor(G[S])
        foreach I ∈ MIS°(G − S) with |I| ≥ ⌈nα − |S|⌉ do
            T = V(G) \ (S ∪ I)
            x_2 = PolySpaceColor(G[T])
            x = min{x, x_1 + 1 + x_2}
    return x
```

Implementing Lemma 3.4 yields a polynomial-space divide-and-conquer algorithm, Algorithm 3. Note that, as $|T| \leq n(1-\alpha)$, it holds that $|I| = n - |T| - |S| \geq \lceil n\alpha - |S| \rceil$. Using this algorithm, we obtain the following theorem.

Theorem 3.5. *The chromatic number of a mixed graph can be computed in $\mathcal{O}(4.81069^n)$ time and $\mathcal{O}(n^2)$ space.*

Proof. As Algorithm 3 implements Lemma 3.4, it computes the chromatic number of a mixed graph. It remains to show that it does so in $\mathcal{O}(4.81069^n)$ time and $\mathcal{O}(n^2)$ space.

Runtime For a mixed graph G, let $L(G)$ be the number of leaves in the recursion tree of Algorithm 3 with G as input. It is $L(\emptyset) = 1$ and for $G \neq \emptyset$ we know that:

$$\begin{aligned} L(G) &= \sum_{S \subset V(G)\colon |S| \leq \lceil n\alpha - 1 \rceil, d_G^-(S) = 0} \left[L(G[S]) + \sum_{I \in \mathrm{MIS}^\circ(G-S)\colon |I| \geq \lceil n\alpha - |S| \rceil} L(G - S - I) \right] \\ &= \sum_{i=0}^{\lceil n\alpha - 1 \rceil} \sum_{S \in \binom{V(G)}{i}\colon d_G^-(S) = 0} \left[L(G[S]) + \sum_{\ell = \lceil n\alpha - i \rceil}^{n-i} \sum_{I \in \mathrm{MIS}^\circ(G-S)\colon |I| = \ell} L(G - S - I) \right] \end{aligned}$$

Let $L(n)$ be the worst-case number of leaves in the recursion tree of Algorithm 3 on mixed graphs with n vertices. It holds that $L(0) = 1$, and for $n \geq 1$ we can upper bound $L(n)$

as follows, where $\mu(n-i,\ell)$ denotes the maximum number of maximal independent sets of size ℓ in a graph with $n-i$ vertices:

$$L(n) \leq \sum_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} \cdot \left[L(i) + \sum_{\ell=\lceil n\alpha-i\rceil}^{n-i} (\mu(n-i,\ell) \cdot L(n-i-\ell)) \right]$$

In order to simplify the bound, we split the sum:

$$\begin{aligned} L(n) &\leq \sum_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} L(i) + \sum_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} \sum_{\ell=\lceil n\alpha-i\rceil}^{n-i} (\mu(n-i,\ell) \cdot L(n-i-\ell)) \\ &\leq \sum_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} L(i) + \sum_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{n-i} \sum_{\ell=\lceil n\alpha-i\rceil}^{n-i} (\mu(n-i,\ell) \cdot L(n-i-\ell)) \\ &\leq \sum_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} L(i) + \sum_{i=\lfloor n(1-\alpha)+1\rfloor}^{n} \binom{n}{i} \sum_{\ell=\lceil i-n(1-\alpha)\rceil}^{i} (\mu(i,\ell) \cdot L(i-\ell)) \end{aligned}$$

Next, we use the easily observed fact that for any $m \in \mathbb{N}_0$ and $f : \mathbb{N}_0 \to \mathbb{R}$ it holds that $\sum_{i=0}^{m} f(i) \leq (m+1)\max_{i=0}^{m} f(i)$:

$$\begin{aligned} L(n) &\leq \lceil n\alpha\rceil \max_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} L(i) + \\ &\quad \lceil n\alpha\rceil \max_{i=\lfloor n(1-\alpha)+1\rfloor}^{n} \binom{n}{i} \cdot \lfloor n(1-\alpha)+1\rfloor \max_{\ell=\lceil i-n(1-\alpha)\rceil}^{i} (\mu(i,\ell) \cdot L(i-\ell)) \\ &\leq n \max_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} L(i) + n \max_{i=\lfloor n(1-\alpha)+1\rfloor}^{n} \binom{n}{i} \cdot n \max_{\ell=\lceil i-n(1-\alpha)\rceil}^{i} (\mu(i,\ell) \cdot L(i-\ell)) \\ &\leq n^2 \left[\max_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} L(i) + \max_{i=\lfloor n(1-\alpha)+1\rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha)\rceil}^{i} (\mu(i,\ell) \cdot L(i-\ell)) \right] \end{aligned}$$

Our goal is to find a non-recursive upper bound for $L(n)$, which we expect to be of the form c^n for a $c > 1$. To account for a factor of $2n^2$ introduced by the analysis on each level of the recursion, we add a factor of $(2n^2)^{D(n)}$, where $D(n)$ is the worst-case depth of the recursion tree on a mixed graph with n vertices. Clearly, it holds that $D(0) = 0$ and $D(n) = 1 + \max\{D(\lceil n\alpha - 1\rceil), D(\lfloor n(1-\alpha)\rfloor)\}$, as $\lceil n\alpha - 1\rceil$ and $\lfloor n(1-\alpha)\rfloor$ are upper bounds on the size of S and T, respectively.

We show inductively that $L(n) \leq c^n(2n^2)^{D(n)}$ holds for some constant $c > 1$ and every $n \in \mathbb{N}_0$. The bound holds for the base case, $n = 0$, as $L(0) = 0 \leq c^0(2 \cdot 0^2)^{D(0)}$.

Assuming that $L(m) \leq c^m (2m^2)^{D(m)}$ holds for all $0 \leq m < n$, we obtain:

$$\begin{aligned}
L(n) &\leq n^2 \left[\max_{i=0}^{\lceil n\alpha - 1 \rceil} \binom{n}{i} L(i) + \max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(\mu(i,\ell) \cdot L(i-\ell) \right) \right] \\
&\leq n^2 \left[\max_{i=0}^{\lceil n\alpha - 1 \rceil} \binom{n}{i} c^i (2i^2)^{D(i)} \right. \\
&\qquad \left. + \max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(\mu(i,\ell) \cdot c^{i-\ell} (2(i-\ell)^2)^{D(i-\ell)} \right) \right] \\
&\leq n^2 \left[\max_{i=0}^{\lceil n\alpha - 1 \rceil} \binom{n}{i} c^i (2n^2)^{D(n)-1} \right. \\
&\qquad \left. + \max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(\mu(i,\ell) \cdot c^{i-\ell} (2n^2)^{D(n)-1} \right) \right] \\
&\leq n^2 (2n^2)^{D(n)-1} \left[\max_{i=0}^{\lceil n\alpha - 1 \rceil} \binom{n}{i} c^i + \max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(\mu(i,\ell) \cdot c^{i-\ell} \right) \right]
\end{aligned}$$

Let c be chosen such that for all $n \in \mathbb{N}$ the following holds:

$$\max_{i=0}^{\lceil n\alpha - 1 \rceil} \binom{n}{i} c^i \leq c^n \tag{3.1}$$

$$\max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(\mu(i,\ell) \cdot c^{i-\ell} \right) \leq c^n \tag{3.2}$$

Then, we obtain that our bound also holds for $L(n)$:

$$\begin{aligned}
L(n) &\leq n^2 (2n^2)^{D(n)-1} \left[\max_{i=0}^{\lceil n\alpha - 1 \rceil} \binom{n}{i} c^i + \max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(\mu(i,\ell) \cdot c^{i-\ell} \right) \right] \\
&\leq n^2 (2n^2)^{D(n)-1} \left[c^n + c^n \right] \\
&\leq n^2 (2n^2)^{D(n)-1} 2c^n \\
&\leq c^n (2n^2)^{D(n)}
\end{aligned}$$

For given α, it remains to determine the smallest possible c that fulfills both inequalities. Up to this point, we assumed α to be a fixed constant in $(0, 1)$. To improve the runtime of the algorithm, we choose an α such that the c can become as small as possible.

For $\alpha \leq 0.5$, it holds that:

$$\begin{aligned}\max_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} c^i &\leq \max_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} c^{\lceil n\alpha-1\rceil} \\ &\leq \max_{i=\lfloor n(1-\alpha)+1\rfloor}^{n} \binom{n}{i} c^{\lfloor n(1-\alpha)\rfloor} \\ &\leq \max_{i=\lfloor n(1-\alpha)+1\rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha)\rceil}^{i} \left(\mu(i,\ell)\cdot c^{i-\ell}\right)\end{aligned}$$

In the following, we therefore only considers $\alpha \geq 0.5$.

In order to easier determine a small c such that both inequalities hold, we strengthen both inequalities, making them independent of n. The first inequality, Equation (3.1), can be strengthened using the following:

$$\frac{1}{\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)} \leq c \Longrightarrow \max_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} c^i \leq c^n$$

We prove this implication in two steps. First, we show that we can replace Equation (3.1) with an inequality independent of n.

Claim 3.5.1. For $n \in \mathbb{N}$ and $\alpha \in (0,1)$, it holds that:

$$\max_{\beta\in[0,\alpha]} B(\beta)c^\beta \leq c \Longrightarrow \max_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} c^i \leq c^n$$

Recall that $B(\beta) = \frac{1}{\beta^\beta(1-\beta)^{1-\beta}}$.

Proof. We show this by upper bounding $\max_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} c^i$ with $\left[\max_{\beta\in[0,\alpha]} B(\beta)c^\beta\right]^n$. To achieve this, we use the well-known upper bound from Lemma 2.1, $\binom{n}{n\beta} \leq B(\beta)^n$. We obtain:

$$\begin{aligned}&\max_{i=0}^{\lceil n\alpha-1\rceil} \binom{n}{i} c^i \\ =&\max_{\beta\in\{i/n \mid 0\leq i\leq \lceil n\alpha-1\rceil\}} \binom{n}{n\beta} c^{n\beta} \\ \leq&\max_{\beta\in\{i/n \mid 0\leq i\leq \lceil n\alpha-1\rceil\}} B(\beta)^n c^{n\beta} \\ \leq&\left[\max_{\beta\in[0,\alpha]} B(\beta)c^\beta\right]^n\end{aligned}$$

◁

Next, we show the following claim that helps us to resolve the maximum.

Claim 3.5.2. The function $B(\beta)c^\beta$ is strictly monotonically increasing for $\beta \in [0, \frac{c}{c+1})$ and strictly monotonically decreasing for $\beta \in (\frac{c}{c+1}, 1]$. Its maximum for $\beta \in [0,1]$ is at $\beta = \frac{c}{c+1}$ with value $(c+1)$.

Proof. Since $B(\beta)c^\beta$ is continuous for $\beta \in [0,1]$ with $B(0)c^0 = 1$ and $B(1)c^1 = c$, it suffices to consider its behavior for $\beta \in (0,1)$. Let $h_c(\beta) = B(\beta)c^\beta$. The derivative of $h_c(\beta)$, denoted $h_c'(\beta)$, is $\ln(\beta/((1-\beta)\cdot c)) \cdot c^\beta(1-\beta)^\beta/((\beta-1)\beta^\beta)$. The only zero point of $h_c'(\beta)$ for $\beta \in (0,1)$ is if $\ln(\beta/((1-\beta)\cdot c)) = 0$, which is equivalent to $\beta = \frac{c}{c+1}$. Note that for $\beta \in (0,1)$ it holds that $c^\beta(1-\beta)^\beta/((\beta-1)\beta^\beta) < 0$. Thus, $h_c'(\beta) > 0$ if $\ln(\beta/((1-\beta)\cdot c)) < 0$ and $h_c'(\beta) < 0$ if $\ln(\beta/((1-\beta)\cdot c)) > 0$. Rewriting the inequalities, we obtain that $h_c'(\beta) > 0$ if $\beta < \frac{c}{c+1}$ and $h_c'(\beta) < 0$ if $\beta > \frac{c}{c+1}$. Therefore, the function $h_c(\beta)$ has its only maximum in the interval $(0,1)$ at $\frac{c}{c+1}$ with value $c+1$. ◁

Using this claim, we can now show the second half of the implication.

Claim 3.5.3. For $\alpha \in (0,1)$ and $n \in \mathbb{N}$, the following holds:

$$\frac{1}{\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)} \leq c \Longrightarrow \max_{\beta\in[0,\alpha]} B(\beta)c^\beta \leq c$$

Proof. Recall that $B(\alpha) = \frac{1}{\alpha^\alpha(1-\alpha)^{1-\alpha}}$. It therefore holds that:

$$B(\alpha)c^\alpha \leq c \Leftrightarrow \frac{1}{\alpha^\alpha(1-\alpha)^{1-\alpha}}c^\alpha \leq c \Leftrightarrow \frac{1}{\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)} \leq c$$

To prove the claim, we therefore show in the following that $\frac{1}{\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)} \leq c$ implies $\max_{\beta\in[0,\alpha]} B(\beta)c^\beta = B(\alpha)c^\alpha$.

We know from Claim 3.5.2 that the function $B(\beta)c^\beta$ is strictly monotonically increasing for $\beta \in [0, \frac{c}{c+1})$. Thus, it holds that $\max_{\beta\in[0,\alpha]} B(\beta)c^\beta = B(\alpha)c^\alpha$ if $\alpha \leq \frac{c}{c+1}$.

It therefore suffices to show that $1/(\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)) \leq c$ implies $\alpha \leq \frac{c}{c+1}$. Note that $\frac{c}{c+1} = \frac{1}{c^{-1}+1}$, which is monotonically increasing in c. It therefore holds that:

$$\left(\alpha \leq \frac{1}{\left(\frac{1}{\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)}\right)^{-1}+1}\right) \wedge \left(\frac{1}{\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)} \leq c\right) \Rightarrow \alpha \leq \frac{1}{c^{-1}+1}$$

It therefore remains to show that $\alpha \leq \frac{1}{\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)+1}$ holds. It holds that:

$$\begin{aligned}
\alpha \leq \frac{1}{\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)+1} &\Leftrightarrow \alpha(\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)+1) \leq 1 \\
&\Leftrightarrow \alpha^{\frac{1}{1-\alpha}}(1-\alpha)+\alpha \leq 1 \\
&\Leftrightarrow \alpha^{\frac{1}{1-\alpha}}+\alpha(1-\alpha^{\frac{1}{1-\alpha}}) \leq 1
\end{aligned}$$

As $\alpha < 1$, it holds that $\alpha^{\frac{1}{1-\alpha}} + \alpha(1-\alpha^{\frac{1}{1-\alpha}}) < \alpha^{\frac{1}{1-\alpha}} + (1-\alpha^{\frac{1}{1-\alpha}}) = 1$. Thus, the claim holds. $\triangleleft$

Combining Claims 3.5.1 and 3.5.3, we obtain that if $1/(\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha)) \leq c$, then Equation (3.1) holds. Note that $1/(\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha))$ is monotonically increasing for $\alpha \in (0,1)$, as its derivative is $\ln(\alpha)/((\alpha-1)^3\alpha^{\frac{\alpha}{1-\alpha}})$, which is positive for $\alpha \in (0,1)$. As we only consider $\alpha \geq 0.5$, it follows that we only need to consider $c \geq 4$.

Next, we deal with the second inequality, Equation (3.2).

Claim 3.5.4. For $\frac{16}{43} \leq \alpha < 1$ and $c \geq \frac{81}{64}$ it holds that:

$$\frac{81}{64} \cdot \frac{172}{81}^{\frac{1}{\alpha}} \leq c \Longrightarrow \max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(\mu(i,\ell) \cdot c^{i-\ell}\right) \leq c^n$$

Proof. We prove this implication by showing that if $c \geq \frac{81}{64}$ and $\frac{16}{43} \leq \alpha < 1$, it holds that:

$$\max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(\mu(i,\ell) \cdot c^{i-\ell}\right) \leq \left[\frac{64c}{81}^{1-\alpha}\frac{43}{16}\right]^n$$

This bound proves the implication, as $\frac{64c}{81}^{1-\alpha}\frac{43}{16} \leq c$ is equivalent to $\frac{81}{64} \cdot \frac{172}{81}^{\frac{1}{\alpha}} \leq c$.

To show the bound, we begin by using the bound on the number of maximal independent sets of a given size by Eppstein [Epp01]; see Lemma 2.4:

$$\begin{aligned}
&\max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(\mu(i,\ell) \cdot c^{i-\ell}\right) \\
\leq &\max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(3^{4\ell-i}4^{i-3\ell} \cdot c^{i-\ell}\right) \\
= &\max_{i=\lfloor n(1-\alpha)+1 \rfloor}^{n} \binom{n}{i} \left(\frac{4c}{3}\right)^i \max_{\ell=\lceil i-n(1-\alpha) \rceil}^{i} \left(\frac{3^4}{4^3c}\right)^\ell
\end{aligned}$$

As $c \geq \frac{81}{64}$, it follows that $\frac{3^4}{4^3c} \leq 1$, and thus that the maximum of $\left(\frac{3^4}{4^3c}\right)^\ell$ is attained for the smallest ℓ, in this case for $\ell = \lceil i - n(1-\alpha) \rceil$. As we are constructing an upper bound, we can omit the ceilings, setting $\ell = i - n(1-\alpha)$ instead. We simplify the bound accordingly, using also the bound on the binomial coefficient from Lemma 2.1:

$$
\begin{aligned}
&\max_{i=\lfloor n(1-\alpha)+1\rfloor}^{n} \binom{n}{i} \left(\frac{4c}{3}\right)^i \max_{\ell=\lceil i-n(1-\alpha)\rceil}^{i} \left(\frac{3^4}{4^3c}\right)^\ell \\
&\leq \max_{i=\lfloor n(1-\alpha)+1\rfloor}^{n} \binom{n}{i} \left(\frac{4c}{3}\right)^i \left(\frac{3^4}{4^3c}\right)^{i-n(1-\alpha)} \\
&= \max_{\beta\in\{i/n \mid \lfloor n(1-\alpha)+1\rfloor \leq i \leq n\}}^{n} \binom{n}{n\beta} \left(\frac{4c}{3}\right)^{n\beta} \left(\frac{3^4}{4^3c}\right)^{n(\beta-(1-\alpha))} \\
&= \left[\left(\frac{4^3c}{3^4}\right)^{1-\alpha} \max_{\beta\in\{i/n \mid \lfloor n(1-\alpha)+1\rfloor \leq i \leq n\}}^{n} B(\beta) \left(\frac{3^3}{4^2}\right)^\beta\right]^n \\
&\leq \left[\left(\frac{4^3c}{3^4}\right)^{1-\alpha} \max_{\beta\in[1-\alpha,1]} B(\beta) \left(\frac{3^3}{4^2}\right)^\beta\right]^n \\
&= \left[\left(\frac{64c}{81}\right)^{1-\alpha} \max_{\beta\in[1-\alpha,1]} B(\beta) \left(\frac{27}{16}\right)^\beta\right]^n
\end{aligned}
$$

From Claim 3.5.2, we know that the maximum of $B(\beta)\frac{27}{16}^\beta$ for $\beta \in [0,1]$ is attained at $\beta = \frac{27/16}{27/16+1} = \frac{27}{43}$. As $\alpha \geq \frac{16}{43}$, it follows that $\frac{27}{43} \in [1-\alpha, 1]$. Thus, we obtain that $\max_{\beta\in[1-\alpha,1]} B(\beta)\frac{27}{16}^\beta = \frac{27}{16} + 1 = \frac{43}{16}$, proving the bound and therefore the claim. $\lhd$

As we only consider $\alpha \geq 0.5$, and know that we only have to consider $c \geq 4$, we can use this claim to strengthen the second inequality, Equation (3.2). Note that $\frac{81}{64} \cdot \frac{172}{81}^{\frac{1}{\alpha}}$ is monotonically decreasing for $\alpha \in (0,1)$.

Thus, for $\alpha \geq 0.5$, the smallest c we obtained such that both inequalities holds is $\max\{(\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha))^{-1}, \frac{81}{64} \cdot \frac{172}{81}^{\frac{1}{\alpha}}\}$. A plot of these two functions is shown in Figure 3.2. It remains to determine the best value for α. As for $\alpha \in [0.5, 1)$ it holds that $(\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha))^{-1}$ is monotonically increasing from 4 to ∞ and $\frac{81}{64} \cdot \frac{172}{81}^{\frac{1}{\alpha}}$ is monotonically decreasing from about 5.7 to 2.6875, it follows that the best value for α is obtained when $(\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha))^{-1} = \frac{81}{64} \cdot \frac{172}{81}^{\frac{1}{\alpha}}$. Solving this equation numerically we obtain $\alpha \approx 0.563964$ with $c \approx 4.81068$. The Mathematica [Wol] code used to compute these values can be seen in the appendix (Code A.1).

Thus, by setting the α in the algorithm to 0.563964, we obtain that for all $n \in \mathbb{N}_0$ it holds that $L(n) \leq 4.81068^n(2n^2)^{D(n)}$. We now show that the runtime of the algorithm is within a polynomial factor of that bound. First off, the number of nodes in the recursion tree is at most $L(n) \cdot (D(n)+1)$. Second, the for-loops over the sets S and I run within a polynomial factor of their respective bounds. For S, this is achieved via enumeration using a bitvector. For I, this follows from Lemma 2.4. The remaining operations of the algorithm are polynomial. As we show in the following, $D(n) \in \mathcal{O}(\log n)$, which implies that the runtime is in $\mathcal{O}^*(4.81068^n(2n^2)^{\log n})$.

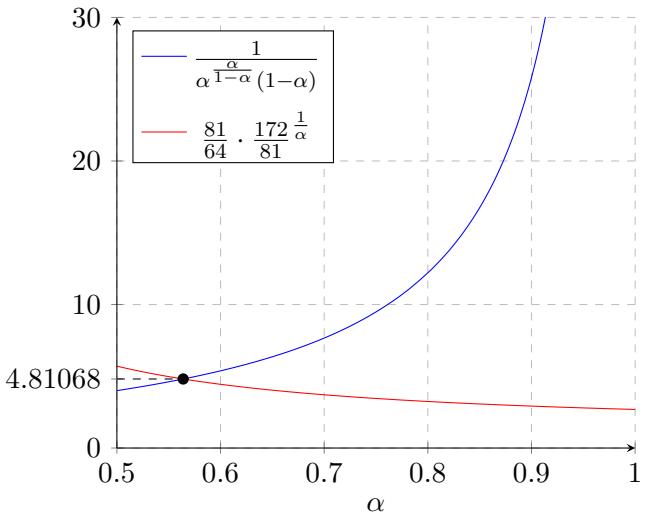


Fig. 3.2: Plot of the functions $(\alpha^{\frac{\alpha}{1-\alpha}}(1-\alpha))^{-1}$ and $\frac{81}{64} \cdot \frac{172}{81}^{\frac{1}{\alpha}}$ for $\alpha \in [0.5, 1)$. The intersection point $(0.563964, 4.81068)$ gives the best values for α and c.

Recall that $D(0) = 0$ and $D(n) = 1 + \max\{D(\lceil n\alpha - 1 \rceil), D(\lfloor n(1-\alpha) \rfloor)\}$. Thus, it holds that $D(1) = 1$ and $D(n) \leq 1 + D(\lfloor n\alpha \rfloor)$ for $n \geq 2$ as $\alpha \geq 1/2$. Therefore, it holds that $D(n) \leq \sum_{i=0}^{\lceil \log_{1/\alpha} n \rceil} 1 = \lceil \log_{1/\alpha} n \rceil + 1$. As α is fixed, it follows that $D(n) \in \mathcal{O}(\log n)$.

We can further simplify the runtime of $\mathcal{O}^*(4.81068^n (2n^2)^{\log n})$ by noting that $(2n^2)^{\log n}$ is quasi-polynomial. It follows that $\mathcal{O}^*(4.81068^n (2n^2)^{\log n}) \subset \mathcal{O}((4.81068+\varepsilon)^n)$ for $\varepsilon > 0$. Therefore, the runtime of the algorithm is in $\mathcal{O}(4.81069^n)$.

Space As the algorithm is a branching algorithm, it needs to store the following in each node: the sets S and I together with the current state of the algorithms generating S and I as well as the current bound on the chromatic number. As the algorithms generating S and I only use polynomial space, the space consumption of a node is polynomial. Furthermore, due to the algorithm being a branching algorithm, it only needs to store all nodes on the current path in the recursion tree, which is at most $D(n) + 1$ nodes. As we have shown previously that $D(n) \in \mathcal{O}(\log n)$, it follows that the algorithm only needs polynomial space.

More precisely, Eppstein's algorithm to find all maximal independent sets uses $\mathcal{O}(n^2)$ space, while the algorithm to generate all subsets S of a certain size uses linear space. Since a node at depth i contains a graph with at most n/α^i vertices, this results in $\mathcal{O}(n^2)$ total space. □

To obtain an optimal coloring, we can store the current best coloring (e.g., as color classes) additionally to the chromatic number and return it at the end. Assuming that numbers can be stored in constant space, this does not change the space or runtime of the algorithm.

3.4 Fixed Number of Colors

We have already seen in Section 2.3 that for $k \leq 2$ the k-MixedColoring problem can be solved in linear time. For fixed $k \geq 3$ the problem is NP-hard, due to the NP-hardness of the k-Coloring problem for $k \geq 3$. In this section, we consider (polynomial-space) algorithms for k-MixedColoring for $k \geq 3$, with the goal of finding faster algorithms than the exponential-space algorithm from Theorem 3.3 or the polynomial-space algorithm from Theorem 3.5.

CSP-based Algorithms The currently fastest known algorithm for 3-Coloring is based on a generalization of satisfiability and list coloring, the *(a, b)-Constraint Satisfaction Problem (CSP)*. As defined by Beigel and Eppstein [BE05], an instance of (a, b)-CSP consists of n *variables*, each with a list of at most a colors, and m *constraints*, each being a set of up to b variables together with a color for each variable. A *constraint is satisfied* by a coloring of the variables, if there is at least one variable in the constraint that has a different color than specified in the constraint. For example, the constraint $\{(v_1, 1), (v_2, 1)\}$ is satisfied by the coloring $\{v_1 \mapsto 1, v_2 \mapsto 2\}$ but not by the coloring $\{v_1 \mapsto 1, v_2 \mapsto 1\}$. The goal is to decide whether there exists a list coloring of the variables that satisfies all constraints. See Figure 3.3a for a visualization of a $(3, 2)$-CSP instance.

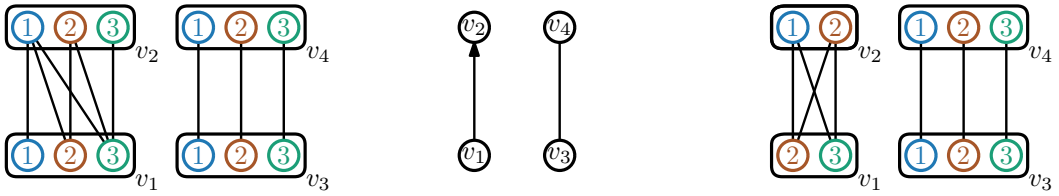


Fig. 3.3: (a) A (3,2)-CSP instance with four variables, each with three possible colors. The variables are represented by black nodes, containing the colors. The constraints are represented by lines connecting incompatible variable colors. A coloring satisfying all constraints would be $\{v_1 \mapsto 1, v_2 \mapsto 2, v_3 \mapsto 3, v_4 \mapsto 2\}$.
(b) A 3-MixedColoring instance equivalent to the CSP instance (a). The variables in the CSP instance correspond to the vertices, and the constraints ensure for the edge $\{v_3, v_4\}$ that the variables receive different colors, and for the arc (v_1, v_2) that v_2 receives a greater color than v_1.
(c) The (3,2)-CSP instance resulting from applying the transformation from Lemma 3.6 to the 3-MixedColoring instance (b). This CSP instance differs from (a) as the rank bounds from Lemma 2.7 have been applied to the variables.

Beigel and Eppstein showed that CSP is a generalization of Coloring. In the following, we show that CSP even generalizes MixedColoring by providing a translation from k-MixedColoring to $(k, 2)$-CSP, with the number of variables corresponding to the number of vertices.

Lemma 3.6. *Given a k-MixedColoring instance with n vertices, we obtain in polynomial time an equivalent instance of $(k, 2)$-CSP with n variables.*

Proof. Given a mixed graph, we construct an equivalent $(k, 2)$-CSP instance as follows: we create a variable v for each vertex v, with its possible colors being the colors $[k]$. For every edge $\{u, v\}$ and every $i \in [k]$ we add the constraint $\{(u, i), (v, i)\}$, ensuring that adjacent vertices receive different colors. For every arc (u, v) and every $i, j \in [k]$ with $i \geq j$ we add the constraint $\{(u, i), (v, j)\}$, ensuring that v receives a greater color than u. See Figure 3.3a for an example of the CSP instance resulting from the 3-MixedColoring instance in Figure 3.3b.

We can further improve the CSP instance by using the property from Lemma 2.7, that the colors of a vertex are restricted by its in- and outrank. Instead of giving each variable v the colors $[k]$, we restrict its colors to $\{\rho^-(v) + 1, \ldots, k - \rho^+(v)\}$, resulting in each variable v having $k - \rho(v)$ possible colors. See Figure 3.3c for an example of the resulting $(3, 2)$-CSP instance with restricted color lists. □

Beigel and Eppstein's [BE05] main result is an algorithm to solve $(4, 2)$-CSP. Making use of several reduction rules, they were able to have the runtime depend only on the number of variables with at least three possible colors. The following lemma formalizes their result.

Lemma 3.7 (Beigel & Eppstein [BE05, Chapter 5])**.** *A $(4, 2)$-CSP instance can be solved in $\mathcal{O}(1.36443^\ell)$ time and polynomial space with $\ell = n_3 + (2-\epsilon)n_4$ and $\epsilon \approx 0.095543$ where n_i is the number of variables with i possible colors.*

Note that the parameters a and b of the (a, b)-CSP problem only restrict the instances, but do not alter the problem itself. Thus, an algorithm for $(4, 2)$-CSP also solves $(3, 2)$-CSP. Therefore, combining the reduction from MixedColoring to CSP with the algorithm for $(4, 2)$-CSP yields an algorithm for 3-MixedColoring, as shown in the following.

Proposition 3.8. *The problem 3-MixedColoring can be solved in $\mathcal{O}(1.36443^{n_0})$ time and polynomial space, where n_0 is the number of vertices with rank 0, i.e., not incident to arcs.*

Proof. A vertex incident to an arc has rank at least 1, since its in- or outrank is at least 1. Using the translation form Lemma 3.6, a variable corresponding to a vertex v has $3 - \rho(v)$ colors. Therefore, the number of variables with 3 possible colors corresponds to the number of vertices with rank 0. Using Lemma 3.7 on the resulting CSP-instance yields the desired runtime. □

Analogously, we obtain an algorithm for 4-MixedColoring.

Proposition 3.9. *The problem 4-MixedColoring can be solved in polynomial space and $\mathcal{O}(1.36443^\ell) \subset \mathcal{O}(1.80723^n)$ time, where $\ell = (2 - 0.095543)n_0 + n_1$ and n_i is the number of vertices of rank i.*

The algorithm for $(4,2)$-CSP can be extended to $(a,2)$-CSP by transforming instances of $(a,2)$-CSP into equivalent instances of $(4,2)$-CSP in polynomial time. We achieve this by generalizing a reduction by Beigel and Eppstein from $(4,2)$-CSP to $(3,2)$-CSP, which replaces a variable with four colors with two variables of three colors.

Lemma 3.10. *The problem $(a,2)$-CSP can be solved in polynomial space and $\mathcal{O}(1.36443^{\ell})$ time where $\ell = \sum_{i=3}^{a}(\lfloor (i-2)/2 \rfloor (2-0.095543) + (i \bmod 2)) \cdot n_i$ with n_i being the number of variables with i possible colors.*

Proof. We show this inductively over a. We know from Lemma 3.7 that $(3,2)$-CSP can be solved in $\mathcal{O}(1.36443^{n_3})$ time and that $(4,2)$-CSP can be solved in $\mathcal{O}(1.36443^{\ell})$ time with $\ell = n_3 + (2-0.095543)n_4$.

For $a \geq 5$, we reduce an instance of $(a,2)$-CSP to an instance of $(a-1,2)$-CSP by replacing each variable of a colors with two variables, one of 4 and one of $a-2$ colors. Let v be a variable with a colors and assume w.l.o.g. that the colors of v are $[a]$. We then create two new variables v_1 and v_2, where v_1 gets the first three colors while v_2 gets the remaining $a-3$ colors. The constraints containing v and a color i are adapted accordingly, using now v_1 if $i \in \{1,2,3\}$ and v_2 otherwise. For example, a constraint $\{(v,2),(u,3)\}$ becomes $\{(v_1,2),(u,3)\}$ while a constraint $\{(v,6),(u,1)\}$ becomes $\{(v_2,6),(u,1)\}$. It remains to ensure that not more than one of the colors of v has to be used. This is achieved by adding a new color, $a+1$, to both variables. Additionally, the constraint $\{(v_1,a+1),(v_2,a+1)\}$ is added to ensure that at most one variable uses the color $a+1$ and thus that one of the colors of v must be used. An example of this reduction is shown in Figure 3.4.

Fig. 3.4: A variable with seven possible colors (a) is replaced by two variables (b), one with four colors and one with five colors.

By the induction hypothesis, we know that the resulting $(a-1,2)$-CSP instance can be solved in $\mathcal{O}(1.36443^{\ell'})$ time with $\ell' = \sum_{i=3}^{a-1}(\lfloor (i-2)/2 \rfloor (2-0.095543) + (i \bmod 2)) \cdot n'_i$ where n'_i is the number of variables with i possible colors in the resulting instance. As we replaced each variable of a colors with one variable of four colors and one variable of $a-2$ colors, it holds that $n'_4 = n_4 + n_a$ and $n'_{a-2} = n_{a-2} + n_a$, with $n'_i = n_i$ for the remaining numbers of colors i. It follows that:

$$
\begin{aligned}
\ell' &= \sum_{i=3}^{a-1} \left(\left\lfloor \frac{i-2}{2} \right\rfloor (2 - 0.095543) + (i \bmod 2) \right) \cdot n'_i \\
&= \sum_{i=3}^{a-1} \left(\left\lfloor \frac{i-2}{2} \right\rfloor (2 - 0.095543) + (i \bmod 2) \right) \cdot n_i \\
&\quad + (2 - 0.095543)\, n_a + \left(\left\lfloor \frac{a-4}{2} \right\rfloor (2 - 0.095543) + (a-2 \bmod 2) \right) n_a \\
&= \sum_{i=3}^{a} \left(\left\lfloor \frac{i-2}{2} \right\rfloor (2 - 0.095543) + (i \bmod 2) \right) \cdot n_i.
\end{aligned}
$$

Therefore, it holds that the problem $(a, 2)$-CSP can be solved in $\mathcal{O}(1.36443^{\ell})$ time where $\ell = \sum_{i=3}^{a} (\lfloor (i-2)/2 \rfloor (2 - 0.095543) + (i \bmod 2)) \cdot n_i$. □

As with 4-MixedColoring, we can use the algorithm for $(a, 2)$-CSP to solve k-MixedColoring, generalizing the result from Proposition 3.9.

Theorem 3.11. *The problem k-MixedColoring can be solved in polynomial space and $\mathcal{O}(1.36443^{\ell})$ time with $\ell = \sum_{i=3}^{k} (\lfloor (i-2)/2 \rfloor (2-0.095543) + (i \bmod 2)) \cdot n_{k-i}$, where n_j denotes the number of vertices of rank j.*

Proof. Follows from Lemma 3.6 and Lemma 3.10, as the number of colors a vertex v can have is $k - \rho(v)$ and thus a vertex with i possible colors has rank $k - i$. □

Using the result from Theorem 3.11, we obtain the runtimes of CSP-based algorithms for k-MixedColoring. The runtimes for the first few values of k are shown in Table 3.1.

k	**Runtime** $\mathcal{O}(\dots)$
3	1.36443^n
4	1.80723^n
5	2.46587^n
6	3.26611^n
7	4.45641^n
8	5.90262^n

Tab. 3.1: Runtimes of CSP-based algorithms for k-MixedColoring.

As seen, this approach is only faster than the exponential-space algorithm from Theorem 3.3 for $k \leq 4$. However, it is faster than the polynomial-space algorithm from Theorem 3.5 for $k \leq 7$. Furthermore, for graphs with few vertices of low rank, the algorithm is significantly faster.

Beigel and Eppstein showed in the same paper that 3-Coloring can be solved in $\mathcal{O}(1.32886^n)$ time, exploiting some graph properties to further reduce the size of the instance. This was later improved to $\mathcal{O}(1.32173^n)$ by Meijer [Mei23].

Lemma 3.12 (Meijer [Mei23])**.** *3-Coloring can be solved in $\mathcal{O}(1.32173^n)$ time and polynomial space.*

Unfortunately, due to our additional restrictions through arcs, these results do not directly translate to mixed graphs. However, we can still use these algorithms by reducing 3-MixedColoring to 3-Coloring via 3-ListColoring, while only increasing the number of vertices by 3. We begin with the reduction from 3-MixedColoring to 3-ListColoring.

Lemma 3.13. *Let G be a 3-MixedColoring instance. We can construct in polynomial time an equivalent 3-ListColoring instance (G', L') with the same number of vertices.*

Proof. We set G' to be the underlying undirected graph of G and ensure via the lists L', that a coloring c is a proper list coloring for G' if and only if it is a proper coloring for G. Since each arc (u, v) in G is an edge in G', it remains to ensure that $c(u) < c(v)$ in each proper list coloring of G'. Since we only have three colors, there is only a proper coloring for G if $\Lambda(G) \leq 2$. Thus, if there is a directed path of length at least 3, we can return a negative 3-ListColoring instance, e.g., by setting all lists to only contain the color 1 (since there are at least two adjacent vertices in G', due to the path, the resulting instance admits no proper list coloring). Due to Lemma 2.7, we have bounds $\rho_G^-(v) + 1 \leq c(v) \leq 3 - \rho_G^+(v)$ on the color a vertex v can receive in a proper coloring c. We thus set the color list as $L'(v) = [\rho_G^-(v) + 1, \ldots, 3 - \rho_G^+(v)]$. The resulting instance can be constructed in polynomial time, as we only need to compute the in- and outrank of each vertex, as well as the underlying undirected graph.

It remains to show that these instances are equivalent. Let c be a proper coloring of G. From Lemma 2.7, we know that for each v it is $\rho_G^-(v) + 1 \leq c(v) \leq 3 - \rho_G^+(v)$. By definition, it follows that $c(v) \in L'$. By Lemma 2.6, c is a proper coloring of the underlying undirected graph G', and thus a proper list coloring of (G', L'). Now let c be a proper list coloring of (G', L'). Suppose that c is not a proper coloring of G. Since G' is the underlying undirected graph of G, this must mean that an arc is violated, i.e., there is an $(u, v) \in A(G)$ with $c(u) > c(v)$. Since $\rho_G^-(v) \geq 1$ and there are only three colors, it follows that $c(v) = 2$ and $c(u) = 3$. However, as $\rho_G^+(u) \geq 1$, it follows that $c(u) \notin L'(u)$, a contradiction to c being a proper list coloring of (G', L'). □

We now show the second part of the reduction, a well-known reduction from k-ListColoring to k-Coloring.

Lemma 3.14. *Let (G, L) be an instance of k-ListColoring. We can construct in polynomial time an equivalent instance of k-Coloring with k additional vertices.*

Proof. We modify the graph G by adding the K_k, consisting of the vertices $\{v_1, \ldots, v_k\}$, resulting in a graph G'. In any proper k-coloring of G', each vertex v_i must receive a different color, and, since we can permute the colors in a coloring of an undirected graph, we may assume w.l.o.g. that in a proper coloring c of G' it holds that $c(v_i) = i$ for each $i \in [k]$. In order to enforce the color list of each $v \in V(G)$ in G', we add for each color i that v cannot have, i.e., for each $i \in [k] \setminus L(v)$, the edge $\{v, v_i\}$ to G'. This ensures

that in a proper coloring of G' the vertex v cannot be assigned a color outside $L(v)$. An example of the resulting instance can be seen in Figure 3.5.

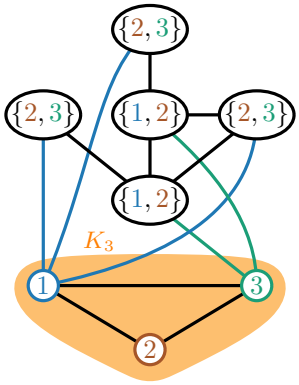


Fig. 3.5: The 3-Coloring instance resulting from a 3-ListColoring instance by addition of the clique K_3. The color lists of the original vertices are depicted inside the vertices.

As we add k new vertices and up to $\mathcal{O}(k^2 + nk)$ edges, we can perform this reduction in polynomial time. □

Joining these two reductions, we arrive at the following.

Lemma 3.15. *An instance of 3-MixedColoring can be transformed in polynomial time into an equivalent instance of 3-Coloring with three additional vertices.*

This reduction allows us to use all algorithms for 3-Coloring on 3-MixedColoring. As the three additional vertices do not influence the asymptotic runtime, using the 3-coloring algorithm from Lemma 3.12 yields the following result.

Theorem 3.16. *3-MixedColoring can be solved in $\mathcal{O}(1.32173^n)$ time and polynomial space.*

Byskov's Techniques We have seen that for $k \leq 7$, CSP-based algorithms are faster than the polynomial space algorithm from Theorem 3.5. In the following, we generalize two techniques by Byskov [Bys04] to solve k-MixedColoring in polynomial space, which results in faster runtimes. We begin with the second technique.

The idea of Byskov's technique 2 is to partition the vertices into two sets and to color the two induced subgraphs independently. To generalize this technique to mixed graphs, we require the partition to be proper. The following lemma shows the correctness of this approach.

Lemma 3.17. *Let $k \geq 2$ and let $k_L, k_R \geq 1$ such that $k_L + k_R = k$. A given mixed graph G is k-colorable if and only if there is a proper partition $\langle V_L, V_R \rangle$ of G such that $G[V_L]$ is k_L-colorable and $G[V_R]$ is k_R-colorable.*

Proof. Let $\langle V_1, \ldots, V_{k_\mathrm{L}}, V_{k_{\mathrm{L}+1}}, \ldots, V_k \rangle$ be the color classes of a proper k-coloring of G. Then, $\langle V_\mathrm{L}, V_\mathrm{R} \rangle$ with $V_\mathrm{L} = V_1 \cup \cdots \cup V_{k_\mathrm{L}}$ and $V_\mathrm{R} = V_{k_{\mathrm{L}+1}} \cup \cdots \cup V_k$ is a proper partition of G. Furthermore, $G[V_\mathrm{L}]$ is k_L-colorable with color classes $\langle V_1, \ldots, V_{k_\mathrm{L}} \rangle$ and $G[V_\mathrm{R}]$ is k_R-colorable with color classes $\langle V_{k_{\mathrm{L}+1}}, \ldots, V_k \rangle$ as $k = k_\mathrm{L} + k_\mathrm{R}$.

On the other hand, if there is such a partition $\langle V_\mathrm{L}, V_\mathrm{R} \rangle$ with $\langle V_1^\mathrm{L}, \ldots, V_{k_\mathrm{L}}^\mathrm{L} \rangle$ being the color classes of a proper k_L-coloring of $G[V_\mathrm{L}]$ and $\langle V_1^\mathrm{R}, \ldots, V_{k_\mathrm{R}}^\mathrm{R} \rangle$ being the color classes of a proper k_R-coloring of $G[V_\mathrm{R}]$, it follows from Lemma 2.3 that $\langle V_1^\mathrm{L}, \ldots, V_{k_\mathrm{L}}^\mathrm{L}, V_1^\mathrm{R}, \ldots, V_{k_\mathrm{R}}^\mathrm{R} \rangle$ is a proper k-coloring of G. □

As shown by this lemma, we can freely choose how to distribute the number of colors between the two induced subgraphs. As we show in the following, the best choice is to distribute the colors evenly.

Theorem 3.18. *The k-MixedColoring problem can be solved in $\mathcal{O}^*((c_{\lceil k/2 \rceil}+1)^n)$ time and polynomial space if $\lceil k/2 \rceil$-MixedColoring can be solved in $\mathcal{O}^*((c_{\lceil k/2 \rceil})^n)$ time and polynomial space.*

Proof. Assume that we can solve j-MixedColoring for $j < k$ in polynomial space and $\mathcal{O}^*(c_j^n)$ time, where $c_j \geq 1$ is a constant. As we can reduce j-MixedColoring to $(j+1)$-MixedColoring by adding a single vertex with edges to all other vertices, we can assume w.l.o.g. that the c_j are nondecreasing for increasing j. Solving k-MixedColoring using the idea from Lemma 3.17 requires creating all proper partitions of the vertices of G into two sets $\langle V_\mathrm{L}, V_\mathrm{R} \rangle$ and checking for each partition whether $G[V_\mathrm{L}]$ is k_L-colorable and $G[V_\mathrm{R}]$ is k_R-colorable. This results in a runtime of $\sum_{i=0}^{n} \binom{n}{i} (\mathcal{O}^*(c_{k_\mathrm{L}}^i) + \mathcal{O}^*(c_{k_\mathrm{R}}^{n-i})) = \mathcal{O}^*((c_{k_\mathrm{L}} + 1)^n) + \mathcal{O}^*((c_{k_\mathrm{R}} + 1)^n)$. The best runtime is therefore obtained if the number of colors is split evenly, i.e., $k_\mathrm{L} = \lceil k/2 \rceil$ and $k_\mathrm{R} = \lfloor k/2 \rfloor$. □

The idea behind Byskov's technique 1 is to find a maximal independent set and to color the remaining graph with one less color. It uses the property that every undirected graph has an optimal coloring where one color class is a maximal independent set. As we have shown in Section 3.1, the generalization of this property is that every mixed graph has an optimal coloring where one color class is an independent set with indegree 0. We can therefore generalize Byskov's technique 1 and obtain the following theorem.

Theorem 3.19. *Let k-MixedColoring be solvable in $\mathcal{O}^*(c^n)$ time and polynomial space for some constant $c \geq 1$. Then, $(k+1)$-MixedColoring is solvable with polynomial space and a runtime of, if $(\lfloor ce \rfloor^{\lfloor ce+1 \rfloor})/(\lfloor ce+1 \rfloor^{\lfloor ce \rfloor} c) \leq 1$, $\mathcal{O}^*((\lceil ce \rceil^{\frac{1}{\lceil ce \rceil}} c^{1-\frac{1}{\lceil ce \rceil}})^n)$, otherwise, $\mathcal{O}^*((\lfloor ce \rfloor^{\frac{1}{\lfloor ce \rfloor}} c^{1-\frac{1}{\lfloor ce \rfloor}})^n)$.*

Proof. If k-MixedColoring can be solved in $\mathcal{O}^*(c^n)$ time and polynomial space for some constant $c \geq 1$, we can solve $(k+1)$-MixedColoring using the generalization of Byskov's technique 1 in $\mathcal{O}^*(\sum_{\ell=1}^{n} \mu(n,\ell) c^{n-\ell})$ time and polynomial space, where $\mu(n,\ell)$ denotes the maximum number of maximal independent sets of size ℓ in a graph with n vertices. We use Byskov's [Bys04] bound on the number of maximal independent sets

of a given size, see Lemma 2.5, as well as the easily observed fact that for any $m \in \mathbb{N}$ and $f\colon \mathbb{N} \to \mathbb{R}$ it holds that $\sum_{i=1}^{m} f(i) \leq m \max_{i=1}^{m} f(i)$ to obtain:

$$
\begin{aligned}
&\mathcal{O}^*\left(\sum_{\ell=1}^{n} \mu(n,\ell)c^{n-\ell}\right)\\
&\subset \mathcal{O}^*\left(n \max_{\ell=1}^{n} \mu(n,\ell)c^{n-\ell}\right)\\
&\subset \mathcal{O}^*\left(\max_{\ell=1}^{n} \lfloor n/\ell \rfloor^{(\lfloor n/\ell \rfloor+1)\ell-n}(\lfloor n/\ell \rfloor+1)^{n-\lfloor n/\ell \rfloor \ell}c^{n-\ell}\right)\\
&= \mathcal{O}^*\left(\max_{d\in\{n/\ell \mid \ell\in[n]\}} \left(\lfloor d \rfloor^{(\lfloor d \rfloor+1)/d-1}(\lfloor d \rfloor+1)^{1-\lfloor d \rfloor/d}c^{1-1/d}\right)^n\right)\\
&\subset \mathcal{O}^*\left(\left(\max_{d\geq 1} \lfloor d \rfloor^{(\lfloor d \rfloor+1)/d-1}(\lfloor d \rfloor+1)^{1-\lfloor d \rfloor/d}c^{1-1/d}\right)^n\right)
\end{aligned}
$$

For convenience, we denote the function $\lfloor d \rfloor^{(\lfloor d \rfloor+1)/d-1}(\lfloor d \rfloor+1)^{1-\lfloor d \rfloor/d}c^{1-1/d}$ as $F(c,d)$. In the following, we determine the maximum of $F(c,d)$ for $d \geq 1$.

Claim. For $d \geq 1$ and fixed $c \geq 1$, the maximum of $F(c,d)$ is attained at $d = \lceil ce \rceil$ if $\frac{\lfloor ce \rfloor^{\lfloor ce+1 \rfloor}}{\lfloor ce+1 \rfloor^{\lfloor ce \rfloor} c} \leq 1$ and at $d = \lfloor ce \rfloor$ if $\frac{\lfloor ce \rfloor^{\lfloor ce+1 \rfloor}}{\lfloor ce+1 \rfloor^{\lfloor ce \rfloor} c} \geq 1$.

Proof. To obtain the maximum of $F(c,d)$ w.r.t. d, we can analyze the zero points of its derivative w.r.t. d. However, the derivative of the floor function is zero (or undefined for integer values). Therefore, we first ignore the floors when analyzing the function, i.e., we analyze $d^{(d+1)/d-1}(d+1)^{1-d/d}c^{1-1/d}$, which we denote as $f_c(d)$, instead. Note that, since $f_c(d)$ has no floors, we can simplify it to $d^{1/d}c^{1-1/d}$.

It holds that $f_c'(d) = c^{(d-1)/d} \cdot d^{1/d-2} \cdot \ln(ce/d)$. Since $c^{(d-1)/d} \cdot d^{1/d-2}$ is positive for $d \geq 1$, it follows that $f_c'(d)$ is positive if $\ln(ce/d) > 0$ and negative if $\ln(ce/d) < 0$. Rewriting this, we obtain that $f_c'(d)$ is positive for $d < ce$ and negative for $d > ce$. Thus, the maximum of $f_c(d)$ for $d \geq 1$ is attained at $d = ce$.

Since $F(c,d)$ uses floors, we cannot immediately deduce its maximum from the maximum of $f_c(d)$. However, we know that for integer d, it holds that $F(c,d) = f_c(d)$. Thus, if we show that the maximum of $F(c,d)$ is always at an integer point, it follows that the maximum of $F(c,d)$ for $d \geq 1$ is at $d = \lfloor ce \rfloor$ or $d = \lceil ce \rceil$, since $f_c(d)$ has only one extreme point for $d \geq 1$. In order to show that the maximum of $F(c,d)$ is attained for integer d, we first show that the function is continuous.

Note that $F(c,d)$ can also be written as $\left(\frac{\lfloor d \rfloor+1}{\lfloor d \rfloor} \cdot c\right) \cdot \left(\frac{\lfloor d \rfloor^{(\lfloor d \rfloor+1)}}{(\lfloor d \rfloor+1)^{\lfloor d \rfloor}} \cdot \frac{1}{c}\right)^{1/d}$. Thus, for every $y \in \mathbb{N}$ it holds that $F(c,d)$ is continuous in the interval $[y, y+1)$, as for $d \in [y, y+1)$ it holds that $F(c,d) = \left(\frac{y+1}{y} \cdot c\right) \cdot \left(\frac{y^{(y+1)}}{(y+1)^y} \cdot \frac{1}{c}\right)^{1/d}$. To show that $F(c,d)$ is continuous for all $d \geq 1$, it remains to show that $\lim_{d\to(y+1)^-} F(c,d) = F(c,y+1)$ for every $y \in \mathbb{N}$. Since y is integer, it holds that $F(c,y+1) = f_c(y+1)$. Thus, we obtain:

$$\begin{aligned}
F(c, y+1) &= f_c(y+1) \\
&= (y+1)^{1/(y+1)} c^{1-1/(y+1)} \\
&= y^{(y+1)/(y+1)-1} (y+1)^{1-y/(y+1)} c^{1-1/(y+1)} \\
&= \lim_{d \to (y+1)^-} \lfloor d \rfloor^{(\lfloor d \rfloor+1)/d-1} (\lfloor d \rfloor + 1)^{1-\lfloor d \rfloor/d} c^{1-1/d} \\
&= \lim_{d \to (y+1)^-} F(c, d)
\end{aligned}$$

In order to show that the maximum is at an integer point, we now show that the function is monotone in each interval $[y, y+1)$ for $y \in \mathbb{N}$. We already know that for given $y \in \mathbb{N}$ and $d \in [y, y+1)$ it holds that $F(c,d) = \left(\frac{y+1}{y} \cdot c\right) \cdot \left(\frac{y^{(y+1)}}{(y+1)^y} \cdot \frac{1}{c}\right)^{1/d}$. Since $F(c,d)$ is continuous, it even holds that $F(c,d) = \left(\frac{y+1}{y} \cdot c\right) \cdot \left(\frac{y^{(y+1)}}{(y+1)^y} \cdot \frac{1}{c}\right)^{1/d}$ for $d \in [y, y+1]$. Thus, if $\left(\frac{y^{(y+1)}}{(y+1)^y} \cdot \frac{1}{c}\right) < 1$, then $F(c,d)$ is strictly increasing in the interval $[y, y+1]$, and if $\left(\frac{y^{(y+1)}}{(y+1)^y} \cdot \frac{1}{c}\right) > 1$, then $F(c,d)$ is strictly decreasing in the interval $[y, y+1]$, with the function being constant if $\left(\frac{y^{(y+1)}}{(y+1)^y} \cdot \frac{1}{c}\right) = 1$. Therefore, the maximum of $F(c,d)$ in the interval $[y, y+1]$ is at $d = y+1$ if $\left(\frac{y^{(y+1)}}{(y+1)^y} \cdot \frac{1}{c}\right) \leq 1$ and at $d = y$ if $\left(\frac{y^{(y+1)}}{(y+1)^y} \cdot \frac{1}{c}\right) \geq 1$. As in each interval $[y, y+1]$ the maximum is attained at one of the interval's endpoints, it follows that the maximum of $F(c,d)$ for $d \geq 1$ is at an integer point.

Therefore, the maximum of $F(c,d)$ for $d \geq 1$ is at $d = \lfloor ce \rfloor$ or $d = \lceil ce \rceil$. Thus, the maximum of $F(c,d)$ lies in the interval $[\lfloor ce \rfloor, \lceil ce \rceil]$ and, using the above analysis of the maximum in an interval $[y, y+1]$ with $y = \lfloor ce \rfloor$, we obtain the stated result. ◁

We have therefore found the maximum of $F(c,d)$ for $d \geq 1$. For integer d, we can simplify $F(c,d)$ to $d^{1/d} c^{1-1/d}$, resulting in the stated runtime. □

Using these two techniques, we can now determine the runtimes of polynomial-space algorithms for k-MixedColoring, which are summarized in Table 3.2. As seen, we obtained algorithms faster than the algorithm from Theorem 3.5 for $k \leq 32$. Note that, as it currently stands, all faster algorithms obtained through these two techniques use the 3-coloring algorithm from Theorem 3.16 as base case. Therefore, any improvement of 3-coloring algorithms result in an improvement of all these runtimes.

$k \leq$	**Runtime** $\mathcal{O}(\dots)$	**Approach**
2	`Linear`	Thm. 2.9
3	1.32173^n	Thm. 3.16
4	1.74330^n	MIS° (3)
5	2.15225^n	MIS° (4)
6	2.32173^n	D&C (3)
7	2.71979^n	MIS° (6)
8	2.74330^n	D&C (4)
9	3.13611^n	MIS° (8)
10	3.15225^n	D&C (5)
12	3.32173^n	D&C (6)
13	3.71075^n	MIS° (12)
14	3.71979^n	D&C (7)
16	3.74330^n	D&C (8)
17	4.12981^n	MIS° (16)
18	4.13611^n	D&C (9)
20	4.15225^n	D&C (10)
24	4.32173^n	D&C (12)
25	4.70563^n	MIS° (24)
26	4.71075^n	D&C (13)
28	4.71979^n	D&C (14)
32	4.74330^n	D&C (16)
∞	4.81069^n	Thm. 3.5

Tab. 3.2: Runtimes of polynomial-space algorithms for k-MixedColoring. D&C refers to the technique from Theorem 3.18 and MIS° refers to the technique from Theorem 3.19. The number in brackets refers to the number of colors in the subproblem used by these techniques.

4 Parameterized Colorings

In this chapter, we study the parameterized complexity of MixedColoring. We already know that MixedColoring is NP-hard for three or more colors. Therefore, there cannot be any FPT or XP algorithms for MixedColoring parameterized by the number of colors unless P=NP. To obtain FPT and XP algorithms, we therefore take other parameters into consideration, such as cliquewidth and treewidth.

4.1 Logic

A simple way to obtain FPT-algorithms parameterized by tree- or cliquewidth is to use the algorithmic meta theorem by Courcelle [Cou90]. This theorem yields FPT-algorithms for problems expressible as formulas in the monadic second-order logic of graphs. In the following, we show how to apply Courcelle's Theorem to MixedColoring.

Logic of Mixed Graphs In order to evaluate logical formulas, we need to define a logical structure, which consists of a domain, containing the possible values of the variables, as well as predicates. For a given mixed graph G, we define the logical structure to have domain $V(G)$, i.e., the possible values for variables are the vertices. Furthermore, the logical structure also contains two predicates, $\mathbf{E}$ and $\mathbf{A}$, where $\mathbf{E}(x, y)$ holds if $\{x, y\} \in E(G)$ and $\mathbf{A}(x, y)$ holds if $(x, y) \in A(G)$.

In *first order logic* we can write expressions containing variables, the binary relation $=$, as well as standard logical operators, such as $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. Furthermore, we can quantify variables using $\forall$ and $\exists$. Semantically, the logical operators behave as usual. This allows us to write formulas for several graph properties, such as an (undirected) graph being a clique,

$$\forall x \forall y[(x \neq y) \rightarrow \mathbf{E}(x, y)]$$

or a graph being transitively closed:

$$\forall x \forall y \forall z[(\mathbf{A}(x, y) \wedge \mathbf{A}(y, z)) \rightarrow \mathbf{A}(x, z)]$$

The *monadic second-order logic (MSO_1)* of graphs extends the first order logic by introducing sets of vertices as separate variables together with the membership predicate $\in$, which has its usual meaning.[1] Furthermore, we can quantify over these new variables,

[1] Actually, monadic second-order logic introduces monadic predicates as additional variables. Monadic means that the predicates take a single argument. Thus, they can be seen as sets with the predicate being the check for membership (using $\in$).

i.e., we can quantify over sets of vertices. This enables us to express more complex properties which cannot be expressed in first-order logic, like an (undirected) graph being bipartite:

$$\exists X \forall x \forall y [E(x,y) \to (x \in X \leftrightarrow y \notin X)]$$

We denote variables representing vertices with small letters (x, y) and variables representing sets of vertices as capital letters (X).

Courcelle's Theorem Courcelle [Cou90] showed that every graph property expressed as a monadic second-order formula in the logic of graphs, allowing for quantification over sets of edges, can be decided in FPT time parameterized by the treewidth of the graph plus the size of the formula. Later, Courcelle et al. [CMR00] extended this result to cliquewidth, although without quantification of edge sets. Courcelle's Theorem has also been extended to mixed graphs by Arnsborg et al. [ALS91], although they only showed the result for treewidth, not cliquewidth.

As we show in the following, we can also generalize the cliquewidth result of Courcelle's Theorem to mixed graphs.

Lemma 4.1 (Courcelle's Theorem)**.** *Solving a problem on a mixed graph G expressed in MSO_1 via a formula φ is possible in $\mathcal{O}(f(\mathrm{cw}(G) + |\varphi|) \cdot n(G)^3)$ time with f being computable.*

Proof. We show how to apply the version of Courcelle's Theorem given by Courcelle and Engelfriet [CE12, Theorem 1.25(2)] to mixed graphs. Courcelle and Engelfriet defined the logical structure for directed graphs, with the predicate $\mathbf{arc}(x, y)$ indicating whether there is an arc from x to y. They incorporate undirected graphs by treating them as directed graphs, where each edge $\{x, y\}$ is replaced by the two opposing arcs (x, y) and (y, x). We use the same idea for mixed graphs, which results in a directed graph whose cliquewidth is at most the cliquewidth of the mixed graph, as we can replace each join operation with two opposing arc operations without increasing the number of labels and thus the cliquewidth. As we require mixed graphs to have no parallel edges or arcs, we can express our predicates $\mathbf{A}(x, y)$ and $\mathbf{E}(x, y)$ over the mixed graph with the predicate $\mathbf{arc}(x, y)$ of the directed graph as follows:

$$\mathbf{A}(x,y) := \mathbf{arc}(x,y) \wedge \neg \mathbf{arc}(y,x)$$
$$\mathbf{E}(x,y) := \mathbf{arc}(x,y) \wedge \mathbf{arc}(y,x)$$

Therefore, Courcelle's Theorem can be applied to mixed graphs as well. □

MixedColoring as a Formula As shown in Lemma 2.8, the color classes of a proper coloring of a mixed graph form a proper partition of the vertices into independent sets. In order to obtain a formula for MixedColoring, we first construct a formula that checks whether a given collection of vertex sets $X_1, \ldots, X_k$ corresponds to the color classes of a proper coloring, i.e., whether they form a proper partition into independent

sets. We begin by providing a formula that checks whether a given collection forms a partition. To achieve this, the formula has to check that each vertex is contained in one set, and that there is no vertex contained in multiple sets. As mentioned e.g. by Barr [Bar20], we can express the formula as follows:

$$\mathsf{partition}(X_1, \ldots, X_k) := \forall x \left[\bigvee_{i=1}^{k} x \in X_i \right] \wedge \neg \exists x \left[\bigvee_{i \neq j}^{k} (x \in X_i \wedge x \in X_j) \right]$$

Now that we can check whether a collection of sets is a partition, we need formulas to ensure that a given partition adheres to the constraints imposed by arcs and edges, i.e., that the partition is proper and consists of independent sets. This is achieved by the following formulas:

$$\mathsf{checkArcs}(X_1, \ldots, X_k) := \forall x, y \left[A(x,y) \rightarrow \neg \left(\bigvee_{i \geq j}^{k} (x \in X_i \wedge y \in X_j) \right) \right]$$

$$\mathsf{checkEdges}(X_1, \ldots, X_k) := \forall x, y \left[E(x,y) \rightarrow \neg \left(\bigvee_{i=1}^{k} (x \in X_i \wedge y \in X_i) \right) \right]$$

Combining these three formulas, we can check whether a given collection of sets corresponds to a proper coloring and from that whether a mixed graph is k-colorable:

$$\begin{aligned} \mathsf{properColoring}(X_1, \ldots, X_k) := \mathsf{partition}(X_1, \ldots, X_k) \wedge \mathsf{checkEdges}(X_1, \ldots, X_k) \\ \wedge \mathsf{checkArcs}(X_1, \ldots, X_k) \\ \mathsf{colorable}(k) := \exists X_1, \ldots, X_k \colon \mathsf{properColoring}(X_1, \ldots, X_k) \end{aligned}$$

We have thus expressed MixedColoring as a MSO_1 formula. The length of this formula is in $\mathcal{O}(k^2)$, due to the length of the partition and checkArcs formulas. Using Lemma 4.1 we obtain the following result.

Theorem 4.2. *The problem MixedColoring is FPT parameterized by cliquewidth plus the number of colors.*

While Courcelle's Theorem yields an FPT-algorithm for MixedColoring, the resulting algorithm is quite impractical. For example, the runtime contains huge constants depending on the length of the formula, i.e., the number of colors, which cannot be bounded by an elementary function unless P = NP, as shown by Frick and Grohe [FG04]. The algorithm does therefore not yield an XP-algorithm parameterized solely by tree- or cliquewidth, as the running time depends non-polynomially on the number of colors k, since polynomials are elementary functions.

4.2 Dynamic Program

It is well known [CFK+15, Theorem 7.9] that Coloring, given a tree decomposition of width tw, can be solved in $\mathcal{O}^*(k^{\mathrm{tw}})$ time via a dynamic program. In the following, we

show that this algorithm easily generalizes to MixedColoring. Assume that we are given a nice tree decomposition $\mathcal{T} = (T, \mathcal{B})$ of our mixed graph G. For each node $t \in T$ and coloring c_t of B_t we create a boolean table entry $D[t, c_t]$ that indicates whether there exists a proper k-coloring c of the subgraph G_t with $c|_{B_t} = c_t$. The entry of the root $D[r, \emptyset]$, recall that $\emptyset$ denotes the empty function $\emptyset \to [k]$, then indicates whether there exists a proper k-coloring of the whole graph G. We fill the table bottom-up, starting with the leaves, as follows:

- For each leaf node t, we set $D[t, \emptyset] = \texttt{true}$.
- For each introduce node t with child t' that introduces a vertex v, we set $D[t, c_t] = \texttt{false}$ if c_t is not a proper coloring of $G[B_t]$. Otherwise, we set $D[t, c_t] = D[t', c_t|_{B_{t'}}]$.
- For each forget node t with child t' that forgets a vertex v, we set $D[t, c_t] = \bigvee_{\ell \in [k]} D[t', c_{t'} \cup \{v \mapsto \ell\}]$,
- For a join node t with children t_1 and t_2, we set $D[t, c_t] = D[t_1, c_t] \wedge D[t_2, c_t]$.

Theorem 4.3. *The problem MixedColoring, given a tree decomposition of width* tw*, can be solved in* $\mathcal{O}^*(k^{\mathrm{tw}})$ *time.*

Proof. We show the correctness of the above algorithm inductively. Clearly, the entries for the leaves are computed correctly, as an empty graph has a proper k-coloring. Assume that for a vertex of the tree decomposition t the entries of its children have been computed correctly.

If t is an introduce node, it holds that G_t has a proper k-coloring c with $c|_{B_t} = c_t$ if and only if c_t is a proper coloring of $G[B_t]$ and $G_{t'}$ has a proper k-coloring c' with $c'|_{B_{t'}} = c_t|_{B_{t'}}$. Indeed, if G_t has a proper k-coloring c with $c|_{B_t} = c_t$, it follows that c_t is a proper coloring of $G[B_t]$. Furthermore, $c' = c|_{V_t'}$ is a proper coloring of $G[V_t'] = G_{t'}$ and, since $B_{t'} \subset B_t$, it follows that $c'|_{B_{t'}} = c_t|_{B_{t'}}$. On the other hand, if c_t is a proper coloring of $G[B_t]$ and there exists a proper k-coloring c' of $G_{t'}$ with $c'|_{B_{t'}} = c_t|_{B_{t'}}$, we can construct the following proper k-coloring c of G_t. We set $c = c' \cup c_t$. This is well-defined, as c' and c_t overlap only on $B_{t'}$, and we required that $c'|_{B_{t'}} = c_t|_{B_{t'}}$. Clearly, c is a proper coloring of $G_{t'}$ and $G[B_t]$. Thus, as there is no edge or arc in G_t not already in $G_{t'}$ or $G[B_t]$, due to the properties of the tree decomposition, c is a proper k-coloring of G_t with $c|_{B_t} = c_t$.

If t is a forget node, it holds that there is a proper k-coloring c of G_t with $c|_{B_t} = c_t$ if and only if there is a k-coloring c' of $G_{t'}$ with $c'|_{B_t'} = c_t \cup \{v \mapsto \ell\}$. Indeed, assume there is such a coloring c of G_t. As c colors v, the vertex forgotten by t, there must be a color $\ell \in [k]$ such that $c(v) = \ell$. Thus, as $G_{t'} = G_t$ c is a proper k-coloring of $G_{t'}$ with $c|_{B_{t'}} = c_t \cup \{v \mapsto \ell\}$ (as $B_{t'} = B_t \cup \{v\}$). Conversely, if there is a proper k-coloring c' of $G_{t'}$ with $c'|_{B_{t'}} = c_t \cup \{v \mapsto \ell\}$ for an $\ell \in [k]$ it is also a proper k-coloring of G_t with $c'|_{B_t} = c_t$. There being a color $\ell \in [k]$ such that there is a proper k-coloring c' of $G_{t'}$ with $c'|_{B_{t'}} = c_t \cup \{v \mapsto \ell\}$ is precisely checked by the expression $\bigvee_{\ell \in [k]} D[t', c_t \cup \{v \mapsto \ell\}]$.

If t is a join node with children t_1 and t_2 it holds that there is a proper k-coloring c of G_t with $c|_{B_t} = c_t$ if and only if there is a proper k-coloring c of G_{t_1} and G_{t_2} with $c|_{B_{t_1}} = c|_{B_{t_2}} = c_t$. Clearly, if there is such a coloring c of G_t it is also a coloring of G_{t_1} and G_{t_2} (as $G_t = G_{t_1} \cup G_{t_2}$) with $c|_{B_{t_1}} = c|_{B_{t_2}} = c_t$ (as $B_t = B_{t_1} = B_{t_2}$). Conversely, the reverse direction also holds as $G_t = G_{t_1} \cup G_{t_2}$ (i.e., there is no edge or arc in G_t not already in G_{t_1} or G_{t_2}). There being a proper k-coloring c of G_{t_1} and G_{t_2} such that $c|_{B_{t_1}} = c|_{B_{t_2}} = c_t$ is equivalent to there being proper k-colorings c_1 of G_{t_1} and c_2 of G_{t_2} such that $c_1|_{B_{t_1}} = c_t$ and $c_2|_{B_{t_2}} = c_t$. This follows from the fact that G_{t_1} and G_{t_2} overlap only on B_t, and thus $c = c_1 \cup c_2$ is well-defined and a proper coloring of both graphs. Therefore, it holds that $D[t, c_t]$ if and only if $D[t_1, c_t] \wedge D[t_2, c_t]$.

Computing each entry takes polynomial time, as we access at most k table entries and check whether a coloring is proper. As $|B_t| \leq \mathrm{tw} + 1$, the size of the table is in $\mathcal{O}^*(k^{\mathrm{tw}})$, as a nice tree decomposition has polynomially many nodes. Thus, the overall runtime is $\mathcal{O}^*(k^{\mathrm{tw}})$. This algorithm is quite similar to the tree decomposition based coloring algorithm for undirected graphs, with the only difference being that when checking whether a coloring is proper we have to account for arcs. □

By using backtracking, we can even obtain a proper k-coloring, if it exists. As shown in the following, this algorithm yields FPT and XP runtimes, depending on the parameter.

Corollary 4.4. *MixedColoring is FPT parameterized by treewidth plus the number of colors and XP parameterized solely by treewidth.*

Proof. Recall that we can compute a nice tree decomposition of minimum width in FPT time parameterized by treewidth. Thus, the dynamic program from Theorem 4.3 yields an FPT algorithm for MixedColoring parameterized by treewidth plus the number of colors.

By implementing a preprocessing step that returns `True` immediately if $k \geq n$, as a graph is always k-colorable for such k, the runtime becomes $\mathcal{O}^*(n^{\mathrm{tw}})$, i.e., XP parameterized solely by treewidth. □

Note that this algorithm is also an improvement over the XP-algorithm by Ries and de Werra [RdW08], which runs in $\mathcal{O}(n^{2\mathrm{tw}+4}m^{\mathrm{tw}+2})$ time, where m is the number of edges.

4.3 Bounds

We have seen in the two previous sections that MixedColoring is FPT parameterized by cliquewidth plus the number of colors, and that it is XP parameterized solely by treewidth. In this section, we extend these results to other graph parameters, including vertex cover number, treedepth, and maximum rank. Note that, to extend FPT or XP results from a parameter a to a parameter b, it suffices to show that a is bounded by b, i.e., that there is a computable function f such that $a \leq f(b)$. For example, it is

well-know that treewidth is bounded by vertex cover number, which results in MixedColoring being XP parameterized by vertex cover number. Apart from the number of colors, all considered parameters depend on the provided graph. As the number of colors is, as part of the input, independent of the provided graph, it cannot be bounded by the other graph parameters. However, MixedColoring is also FPT parameterized by cliquewidth plus chromatic number. Indeed, we achieve this by performing a linear search for the chromatic number using the algorithm from Theorem 4.2, which has a runtime of $\mathcal{O}^*(f(\mathrm{cw}+k))$ for a computable, w.l.o.g. monotonically increasing, f. This results in a runtime of $\mathcal{O}^*(f(\mathrm{cw}+\chi))$ to compute the chromatic number of mixed graphs. As we can use the chromatic number to immediately decide MixedColoring, we obtain that MixedColoring is FPT parameterized by cliquewidth plus chromatic number. We can apply the same idea to the algorithm from Theorem 4.3 to obtain that MixedColoring is solvable in $\mathcal{O}^*(\chi^{\mathrm{tw}})$ time. The current state of the parameterized complexity of MixedColoring is shown in Figure 4.1.

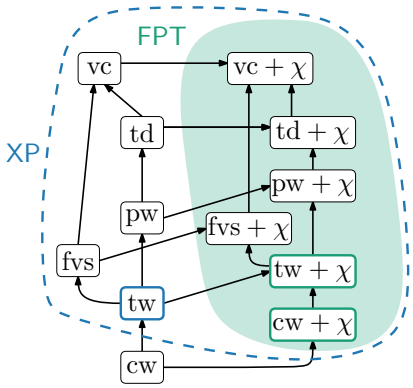


Fig. 4.1: Current state of the parameterized complexity of MixedColoring. An arc from parameter α to parameter β indicates that parameter α is bounded by parameter β.

Treewidth & Maximum Rank For undirected graphs, it is well known that $\chi \leq \mathrm{tw}+1$. As Coloring is FPT parameterized by treewidth plus chromatic number, it follows that Coloring is FPT parameterized solely by treewidth. However, for mixed graphs the treewidth is no longer an upper bound on the chromatic number. Gutowski et al. [GJK+23, Theorem 9 & Proposition 10] showed that for mixed interval graphs it holds that $\chi \leq (\Lambda+1)\cdot\omega$ and that this bound is asymptotically tight. As $\omega = \mathrm{tw}+1$ on (mixed) interval graphs, this implies that the chromatic number of mixed graphs is not bounded by treewidth. In the following, we generalize the bound by Gutowski et al. to mixed graphs, while also tightening it.

To obtain the bound, Gutowski et al. introduced the concept of layering. A *layering* of a mixed graph G is a partition of the vertex set into $\Lambda(G)+1$ layers $\langle L_0,\ldots,L_{\Lambda(G)}\rangle$, such that the *layer* L_i contains all vertices with inrank i. Layerings have some nice properties. First off, each arc is oriented towards the higher layer, i.e. for each $(u,v)\in A(G)$ with $u\in L_i$ and $v\in L_j$ it holds that $i<j$. This stems from the fact that an

arc (u, v) implies that $i = \rho_G^-(u) < \rho_G^-(v) = j$, as stated in Lemma 2.2. Thus, a layering forms a proper partition. Furthermore, each induced subgraph $G[L_i]$ is undirected. We are now equipped to show the bound on the chromatic number of mixed graphs.

Theorem 4.5. *For every mixed graph G and its corresponding layering $\langle L_0, \dots, L_{\Lambda(G)} \rangle$ it holds that $\chi(G) \leq \sum_{i=0}^{\Lambda(G)} (\mathrm{tw}(G[L_i]) + 1)$.*

Proof. We know that $\chi(G[L_i]) \leq \mathrm{tw}(G[L_i]) + 1$ for each layer L_i, as each induced subgraph $G[L_i]$ is undirected. As the layering $\langle L_0, \dots, L_{\Lambda(G)} \rangle$ of G is a proper partition, replacing each layer L_i with the color classes of a $(\mathrm{tw}(G[L_i])+1)$-coloring of $G[L_i]$ yields, according to Lemma 2.3 and 2.8, a proper coloring of G that uses $\sum_{i=0}^{\Lambda(G)} (\mathrm{tw}(G[L_i]) + 1)$ colors. See Figure 4.2 for an example of the resulting coloring. □

(a) (b)

Fig. 4.2: (a) A mixed graph with its layering $\langle L_0, L_1, L_2 \rangle$ of three layers, with $\mathrm{tw}(G[L_0]) = 1$, $\mathrm{tw}(G[L_1]) = 2$, and $\mathrm{tw}(G[L_2]) = 0$. The proof of Theorem 4.5 yields a 6-coloring. (b) An optimal 4-coloring, showing that the coloring obtained via Theorem 4.5 is not optimal.

Note that the only property of treewidth used in the proof of the bound is the fact that treewidth is an upper bound on the chromatic number of undirected graphs. Thus, substituting treewidth with any other bound on the chromatic number of undirected graphs, such as maximum degree, yields an analogous bound.

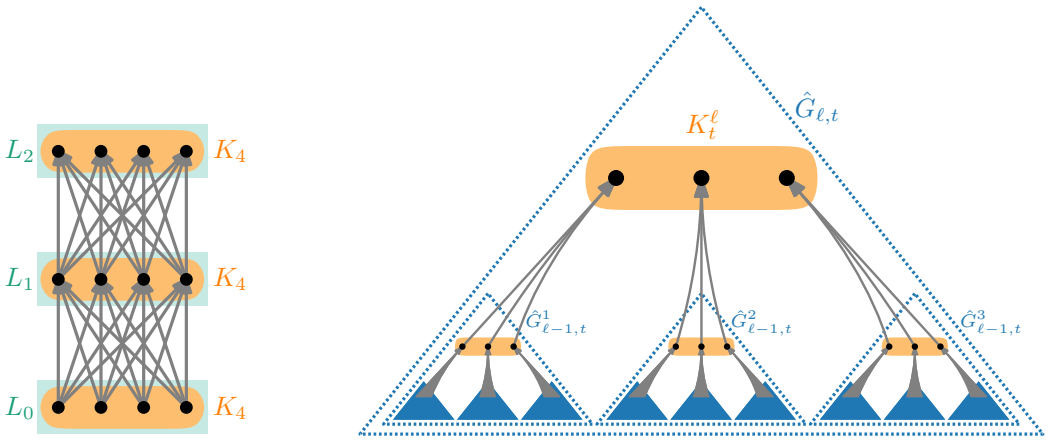
For undirected graphs, it is further well-known that the bound $\chi \leq \mathrm{tw} + 1$ is tight, with the family of complete graphs yielding an equality. By placing a complete graph in each layer and fully connecting adjacent layers via arcs, we obtain for each combination of treewidth and maximum rank a graph for which the bound from Theorem 4.5 is tight. We formalize this in the following.

Proposition 4.6. *There is a family $\mathcal{G}$ consisting of mixed graphs $G_{\ell,t}$ for each $\ell, t \in \mathbb{N}_0$ with $\Lambda(G_{\ell,t}) = \ell$ and $\mathrm{tw}(G_{\ell,t}[L_i]) = t$ for each layer L_i of $G_{\ell,t}$ such that $\chi(G_{\ell,t}) = \sum_{i=0}^{\ell} (t+1)$.*

Proof. For each $\ell \in \mathbb{N}_0$ and $t \in \mathbb{N}_0$, we construct a graph $G_{\ell,t}$ such that $\Lambda(G_{\ell,t}) = \ell$ and $\mathrm{tw}(G_{\ell,t}[L_i]) = t$ for each layer L_i. This is done by taking $\ell + 1$ copies of the graph K_{t+1},

with each vertex in the i-th copy having an incoming arc from each vertex in the $(i-1)$-th copy, and an outgoing arc to each vertex in the $(i+1)$-th copy, with the first and last copies having no incoming and outgoing arcs, respectively. See Figure 4.3a for a visualization of $G_{2,3}$. We now show that $(\ell+1)\cdot(t+1)$ colors are necessary and sufficient to properly color this graph. Note that in the transitive closure $G^+_{\ell,t}$, each vertex in layer i has an incoming arc from each vertex in a lower layer, and an outgoing arc to each vertex in a higher layer. Each vertex must therefore receive a higher color than all vertices in layers below, and a lower color than all vertices in layers above. Thus, in an optimal coloring, the colors used in a layer are successive, with no other layer using these colors. Since each layer, being a copy of the K_{t+1}, needs $t+1$ colors, we need $(\ell+1)\cdot(t+1)$ colors in total. □

Note that in order to show the tightness of the bound, we just relied on the fact that there is an undirected graph (in this case the K_n) for which the bound on undirected graphs (in this case $\chi \leq \mathrm{tw}+1$) is tight. Thus, for any other tight bound on the chromatic number of undirected graphs, such as $\chi \leq \Delta+1$, it follows that their bound on mixed graphs analogous to Theorem 4.5 is also tight, with the corresponding graph family being constructed analogously to Proposition 4.6.



(a) (b)

Fig. 4.3: (a) A visualization of $G_{2,3}$ from Proposition 4.6, consisting of three layers, each with a treewidth of 3.
(b) A schematic representation of the graphs $\hat{G}_{\ell,t}$ from Proposition 4.8, where each vertex in the K_t^ℓ is preceded by its own copy of $\hat{G}_{\ell-1,t}$.
In both figures, the edges of the complete graphs are omitted for visual clarity.

As for each layer L_i of a mixed graph G it holds that $\mathrm{tw}(G[L_i]) \leq \mathrm{tw}(G)$, using Theorem 4.5, we obtain the following bound.

Corollary 4.7. *For every mixed graph G it holds that $\chi(G) \leq (\Lambda(G)+1)\cdot(\mathrm{tw}(G)+1)$.*

While the graph family $\mathcal{G}$ from Proposition 4.6 shows the tightness of the bound from Theorem 4.5, it does not show the tightness of the bound from Corollary 4.7. This is

because it holds for these graphs that $\mathrm{tw}(G_{\ell,t}) = 2t+1$, as adjacent layers form a K_{2t+2}, leading to $\chi(G_{\ell,t}) = (\ell+1)(t+1) = (\Lambda(G_{\ell,t})+1)(\mathrm{tw}(G_{\ell,t})+1)/2$. As we show in the following, we can construct a family of mixed graphs that avoids creating large cliques over adjacent layers, thereby getting closer to the bound from Corollary 4.7.

Proposition 4.8. *There is a family $\hat{\mathcal{G}}$ consisting of mixed graphs $\hat{G}_{\ell,t}$ for each $\ell \in \mathbb{N}$ and $t \in \mathbb{N}$ such that $\Lambda(\hat{G}_{\ell,t}) = \ell$, $\mathrm{tw}(\hat{G}_{\ell,t}) = t$ and $\chi(\hat{G}_{\ell,t}) = (\Lambda(\hat{G}_{\ell,t})+1) \cdot \mathrm{tw}(\hat{G}_{\ell,t})$.*

Proof. We construct the family inductively over ℓ. For simplicity, we start with $\ell = 0$, even though $\hat{G}_{0,t}$ is not part of the family. The graph $\hat{G}_{0,t}$ is just a copy of the K_t, which we denote K_t^0. It holds that $\Lambda(G_{0,t}) = 0$ and $\mathrm{tw}(G_{0,t}) = t-1$ and $\chi(G_{0,t}) = t$. Furthermore, $\hat{G}_{0,t}$ has a tree decomposition of minimum width that consists of a single bag containing all vertices. Also, each vertex in $\hat{G}_{0,t}$ has inrank 0.

For $\ell \geq 1$, assume that we have constructed $\hat{G}_{\ell-1,t}$ with $\Lambda(\hat{G}_{\ell-1,t}) = \ell-1$, $\mathrm{tw}(\hat{G}_{\ell-1,t}) \leq t$, $\chi(\hat{G}_{\ell-1,t}) = \ell \cdot t$, and that $\hat{G}_{\ell-1,t}$ has a tree decomposition of minimum width such that the bag of the root node contains all t vertices of $K_t^{\ell-1}$. Further, assume that each vertex in $K_t^{\ell-1}$ has inrank $\ell-1$.

The graph $\hat{G}_{\ell,t}$ consists of a copy of the K_t, denoted K_t^ℓ, and t copies of the $\hat{G}_{\ell-1,t}$. Let $\{v_1, \ldots, v_t\}$ be the vertices of K_t^ℓ. For each $i \in [t]$ we link v_i to its own copy $\hat{G}^i_{\ell-1,t}$ of $\hat{G}_{\ell-1,t}$, with v_i having an incoming arc from every vertex in $K_t^{\ell-1,i}$. A schematic representation of this graph is shown in Figure 4.3b. Therefore, for each vertex v_i in K_t^ℓ there is a directed path from each vertex in $\hat{G}^i_{\ell-1,t}$ to v_i. Since $\chi(\hat{G}^i_{\ell-1,t}) = \ell \cdot t$, it follows that v_i has to be colored with a color larger than $\ell \cdot t$. As each vertex in K_t^ℓ needs a distinct color, we therefore need at least $(\ell+1) \cdot t$ colors. Furthermore, as we can color each $\hat{G}^i_{\ell-1,t}$ with the first $\ell \cdot t$ colors, and K_t^ℓ with the next t colors, it follows that $(\ell+1) \cdot t$ colors suffice. Thus, it holds that $\chi(\hat{G}_{\ell,t}) = (\ell+1) \cdot t$.

Since each vertex v_i in K_t^ℓ is preceded by vertices from $K_t^{\ell-1,i}$, which have inrank $\ell-1$, it follows that each vertex in K_t^ℓ has inrank ℓ. As we added no other arcs, it follows that $\Lambda(\hat{G}_{\ell,t}) = \ell$.

Further, we construct a tree decomposition of $\hat{G}_{\ell,t}$ of width t as follows: we create the root node r with the bag B_r containing all vertices of K_t^ℓ. For each $i \in [t]$, we add a child node t_i to r, with the bag B_{t_i} containing v_i and all vertices in $K_t^{\ell-1,i}$. We then attach the tree decomposition of $\hat{G}^i_{\ell-1,t}$ with its root node r_i to t_i. Since B_{r_i} contains all vertices in $K_t^{\ell-1,i}$, we obtain a valid tree decomposition. The size of the bag B_{t_i} is $t+1$, the size of the bag B_r is t, and we know from all other bags that, since they are part of tree decompositions of width at most t, they have size at most $t+1$. Thus, the width of this tree decomposition is t, with the root bag containing all vertices of K_t^ℓ. Since each vertex v_i in K_t^ℓ forms a $(t+1)$-clique with the vertices in $K_t^{\ell-1,i}$, it follows that $\mathrm{tw}(\hat{G}_{\ell,t}) = t$. □

So, as treewidth alone does not bound the chromatic number of mixed graphs, the treewidth dynamic program from Theorem 4.3 does not yield an FPT-algorithm parameterized solely by treewidth.

However, we know from Corollary 4.7 that the chromatic number is bounded by treewidth plus maximum rank. It follows that MixedColoring is FPT parameterized by treewidth plus maximum rank. In fact, it holds that treewidth plus chromatic number and treewidth plus maximum rank are equivalent parameters on mixed graphs. This is due to maximum rank being a lower bound on the chromatic number, according to Lemma 2.7. For any parameter p that upper bounds treewidth, it holds therefore that $p + \chi$ and $p + \Lambda$ are equivalent parameters on mixed graphs.

Therefore, if we show that a parameter upper bounds treewidth and maximum rank, it follows that MixedColoring is FPT w.r.t. said parameter. In the following, we show that this is the case for vertex cover number and treedepth, beginning with vertex cover number.

Vertex Cover Number It is well-known that $\mathrm{tw} \leq \mathrm{vc} + 1$ and that for a path P_ℓ it holds that $\mathrm{vc}(P_\ell) = \lceil \ell/2 \rceil$. Thus, it holds for mixed graphs that $\Lambda + 1 \leq 2\mathrm{vc} + 1$. Therefore, MixedColoring is FPT parameterized solely by vertex cover. Furthermore, it follows from Corollary 4.7 that $\chi \leq (2\mathrm{vc} + 1)(\mathrm{vc} + 1)$. However, this bound on the chromatic number is quite large, especially considering that on undirected graphs it holds that $\chi \leq \mathrm{vc} + 1$. As we show in the following, there is a much better bound.

Theorem 4.9. *For every mixed graph G it holds that $\chi(G) \leq 2\mathrm{vc}(G) + 1$. This bound is tight.*

Proof. We construct a proper $2\mathrm{vc}(G) + 1$ coloring c of G given a vertex cover C of size $\mathrm{vc}(G)$ as follows: We first sort the vertices of the cover C topologically (according to the arcs between them in the transitive closure G^+, i.e., we sort $G^+[C]$ topologically), resulting in an ordering $\langle v_1, \ldots, v_{\mathrm{vc}(G)} \rangle$. We then color these vertices with distinct colors, with each vertex v_i obtaining the color $2i$. This yields a proper coloring of $G^+[C]$. For any of the remaining uncolored vertices $v \in V(G) \setminus C$ it holds that v only has neighbors in C, and thus that the neighbors of v are already colored. Let $c^-(v) = \max_{u \in N^-(v)} c(u)$ and $c^+(v) = \min_{u \in N^+(v)} c(u)$ be the biggest and smallest color of the incoming and outgoing neighbors of v, respectively. In the case that v has no incoming or outgoing neighbors, we set $c^-(v) = 0$ and $c^+(v) = 2\mathrm{vc}(G) + 2$, respectively. It holds that $c^-(v) < c^+(v)$. Indeed, suppose that $c^+(v) \geq c^-(v)$. Then there is a $u^+ \in N^+(v)$ with $c(u^+) = c^+(v)$ and a $u^- \in N^-(v)$ with $c(u^-) = c^-(v)$. It follows, that there are arcs (u^-, v) and (v, u^+) in $A(G)$, and thus there is the arc $(u^-, u^+) \in A(G^+)$. As $u^-, u^+ \in C$, it follows that $c^-(v) = c(u^-) < c(u^+) = c^+(v)$ due to our initial coloring being according to the topological order of $G^+[C]$, a contradiction to $c^-(v) \geq c^+(v)$. Thus, it holds that $c^+(v) - c^-(v) \geq 2$, due to all colors of vertices in C being even. We therefore set $c(v) = c^-(v) + 1$. Now all incoming neighbors have a smaller color, all outgoing neighbors have a bigger color, and all remaining neighbors (adjacent via edges) are in C and have thus an even color, which cannot be $c(v)$, as $c(v)$ is uneven. Thus, the coloring remains proper. By coloring all vertices in $V(G) \setminus C$ in this manner, we obtain a proper coloring of G with $2\mathrm{vc}(G) + 1$ colors. See Figure 4.4 for an example of the obtained coloring.

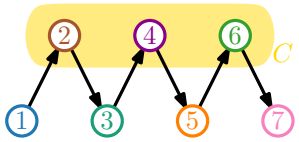


Fig. 4.4: The path P_6 with a smallest vertex cover C of size 3 and resulting optimal 7-coloring.

It remains to show that the bound is tight. For any $\ell \in \mathbb{N}$, the directed path $P_{2\ell}$ has a chromatic number of $2\ell + 1$. Furthermore, the smallest vertex cover contains every second vertex and has thus a size of ℓ. □

Treedepth For undirected graphs, it is well-known [NO12, Section 6.4 & 6.2], that $\mathrm{tw} \leq \mathrm{td} - 1$ and that $\mathrm{td}(P_\ell) = \lceil \log_2(\ell + 1) \rceil$. It follows that $\Lambda + 1 \leq 2^{\mathrm{td}}$. Thus, using Corollary 4.7, it holds for mixed graphs that $\chi \leq 2^{\mathrm{td}} \cdot \mathrm{td}$ and therefore MixedColoring is FPT parameterized by treedepth.

Transitive Closure Lastly, we consider the parameters on the transitive closure. In the following, we show that, except for cliquewidth, all considered parameters on the transitive closure upper bound the maximum rank.

Theorem 4.10. *For every parameter $\alpha \in \{\mathrm{vc}, \mathrm{td}, \mathrm{fvs}, \mathrm{pw}, \mathrm{tw}\}$, it holds on mixed graphs that $\alpha + \Lambda$ is bounded by α^+.*

Proof. We know from Lemma 2.11 that for each of these parameters it holds on every mixed graph G that $\alpha(G) \leq \alpha(G^+)$. It remains to show that all of these parameters bound the maximum rank. If a mixed graph G has maximum rank ℓ, it must contain a directed path P_ℓ. As the underlying undirected graph of the transitive closure of P_ℓ is the complete graph $K_{\ell+1}$, G^+ must contain a $K_{\ell+1}$ as a relaxed subgraph. Thus, it follows from Lemma 2.11 that $\alpha(K_{\ell+1}) \leq \alpha(G^+)$. Further, it is well-known that $\mathrm{vc}(K_{\ell+1}) = \ell$, $\mathrm{td}(K_{\ell+1}) = \ell$, $\mathrm{fvs}(K_{\ell+1}) = \ell - 1$, $\mathrm{pw}(K_{\ell+1}) = \ell$, and $\mathrm{tw}(K_{\ell+1}) = \ell$. Thus, all of these parameters on the transitive closure upper bound the maximum rank. □

Therefore, MixedColoring is FPT parameterized by any of these parameters on the transitive closure. This results in the parameterized complexity seen in Figure 4.5.

4.4 Hardness

Lokshtanov et al. [LMS18] showed that under SETH, there is no algorithm that solves Coloring faster than the treewidth dynamic program, i.e., no algorithm with a runtime of $\mathcal{O}^*((k - \varepsilon)^{\mathrm{tw}})$ for any $\varepsilon > 0$. Together with the result that treewidth is not an upper bound for the chromatic number of mixed graphs, this seems to imply that under SETH there is no FPT-algorithm for MixedColoring parameterized solely by treewidth. However, for the graphs constructed by Lokshtanov et al. to show the lower bound it holds that $k \leq \mathrm{tw} + 1$. Thus, the lower bound under SETH does not exclude the

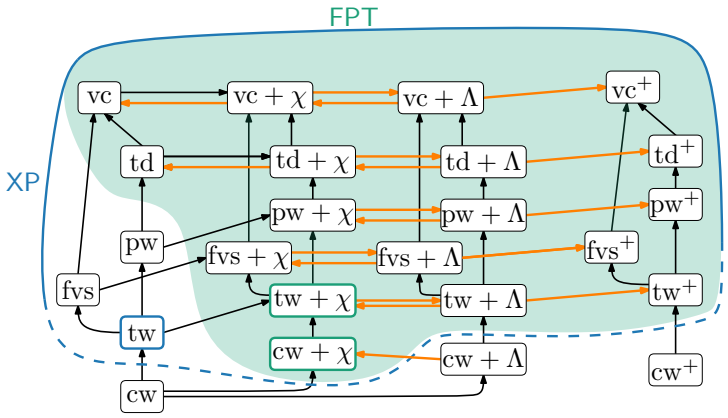


Fig. 4.5: State of parameterized complexity of MixedColoring after Section 4.3. The bold (orange) arcs indicate bounds shown in this section.

possibility of an $\mathcal{O}^*(\mathrm{tw}^{\mathrm{tw}})$ time algorithm for MixedColoring. In this section, we exclude this possibility (unless W[1]=FPT) by showing that MixedColoring is W[1]-hard w.r.t. pathwidth, feedback vertex set, and cliquewidth.

It was shown by Fomin et al. [FGLS10] that Coloring is W[1]-hard w.r.t. cliquewidth. It follows that MixedColoring is also W[1]-hard w.r.t. cliquewidth. Furthermore, as Coloring is a special case of MixedColoring where $\Lambda = 0$ and $\mathrm{cw}^+ = \mathrm{cw}$, as $G^+ = G$ for undirected graphs G, it follows that MixedColoring is W[1]-hard w.r.t. cliquewidth plus maximum rank, as well as w.r.t. the cliquewidth of the transitive closure.

It remains to show that MixedColoring is W[1]-hard w.r.t. pathwidth and feedback vertex set. We achieve through a parameterized reduction from ListColoring, which was shown to be W[1]-hard w.r.t. treewidth by Fellows et al. [FFL+11]. To show W[1]-hardness w.r.t. treewidth, Fellows et al. used a parameterized reduction from MulticoloredClique, which was shown to be W[1]-hard w.r.t. to the size of the clique by Fellows et al. [FHRV09]. In the following, we restate their proof and show that it even yields W[1]-hardness w.r.t. vertex cover number.

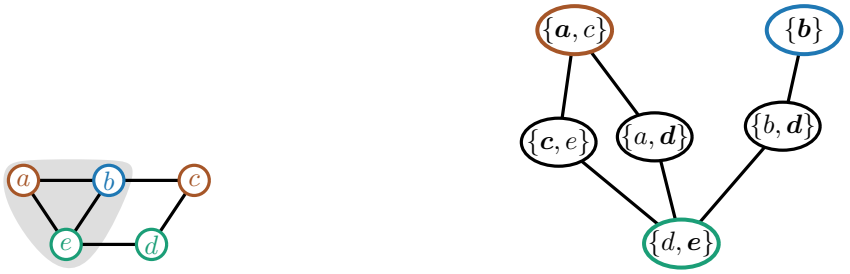
Definition 4.11 (MulticoloredClique)**.** Given a graph G together with an ℓ-coloring, is there an ℓ-clique containing exactly one vertex of each color?

Lemma 4.12 (Fellows et al. [FFL+11, Section 2.1])**.** *ListColoring parameterized by vertex cover number is W[1]-hard.*

Proof. Given an instance (G, c) of MulticoloredClique, Fellows et al. [FFL+11] construct an instance (G', L') of ListColoring as follows: For each color class V_i of c, they create a *color-class-vertex* v_i in G' and set its list $L(v_i)$ to V_i.[2] For every pair of differently colored non-adjacent vertices x, y, i.e., for $x \in V_i$ and $y \in V_j$ such that $i \neq j$ and $\{x, y\} \notin E(G)$, they add an *edge-vertex* v_{xy} to G' and set it adjacent to v_i and v_j

[2]The vertices of G can be treated as colors by numbering them arbitrarily. For simplicity, we use the vertices as colors, without explicit numbering.

with $L(v_{xy}) = \{x, y\}$. Since the color-class-vertices v_i form a vertex cover of G', the vertex cover number is at most ℓ. An example of the reduction can be seen in Figure 4.6.



(a) (b)

Fig. 4.6: (a) An instance of MULTICOLOREDCLIQUE with three colors and five vertices. It has a multicolored 3-clique consisting of the vertices a, b, and e. (b) The instance of LISTCOLORING resulting from the MULTICOLOREDCLIQUE instance. The vertices with a bold (colored) rim are the color-class-vertices corresponding to the color classes of the MULTICOLOREDCLIQUE instance. Since the MULTICOLOREDCLIQUE instance has a multicolored 3-clique, the LISTCOLORING instance has a proper list coloring (highlighted in bold).

We now show that a multicolored ℓ-clique in G corresponds to a proper list coloring of G'. Given a multicolored ℓ-clique C, let $\langle w_1, \ldots, w_\ell \rangle$ be the vertices in C and assume w.l.o.g. that $c(w_i) = i$. We color each color-class-vertex $v_i \in G'$ with the color w_i. For each edge-vertex v_{xy} it holds that $\{x, y\} \notin E(G)$. Thus, x and y cannot be both contained in C. Therefore, as we treat vertices from G as colors in G', at least one of these two colors can be used to color v_{xy}, resulting in a proper list coloring of G'.

Given a proper list coloring c' of G', the set $C = \{c'(v_i) \mid i \in [\ell]\}$ is a multicolored ℓ-clique. Indeed, suppose that there are $x, y \in C$ with $\{x, y\} \notin E(G)$. Let $c(x) = i$ and $c(y) = j$. Then, v_{xy} is adjacent to v_i and v_j which are colored x and y, respectively. As the color of v_{xy} must be either x or y, the coloring c' would be improper, a contradiction. □

To reduce LISTCOLORING to MIXEDCOLORING, we use a similar idea as in Lemma 3.14, using directed paths instead of a clique to enforce the color-list of each vertex.

Lemma 4.13. *There exists a parameterized reduction w.r.t. pathwidth as well as w.r.t. feedback vertex set from* LISTCOLORING *to* MIXEDCOLORING.

Proof. Given an instance (G, L) of LISTCOLORING with $L(v) \subset [\ell]$ for each vertex v of G, we construct the following mixed graph G': For each vertex v of G and $j \in [\ell] - L(v)$, we add the directed path $\langle w_1^{v,j}, \ldots, w_\ell^{v,j} \rangle$, denoted $P_{\ell-1}^{v,j}$. For each color $j \in [\ell] - L(v)$, we add the edge $\{v, w_j^{v,j}\}$. See Figure 4.7 for an example of the additions for each vertex. In any proper ℓ-coloring of G', a vertex $w_i^{v,j}$ must be colored with the color i, as it is part of a directed path of length $\ell - 1$. As v is adjacent to vertices $w_j^{v,j}$ for $j \in [\ell] - L(v)$, it must subsequently be colored with a color in $L(v)$. Thus, a proper ℓ-coloring of G' corresponds to a proper list coloring of G and vice versa.

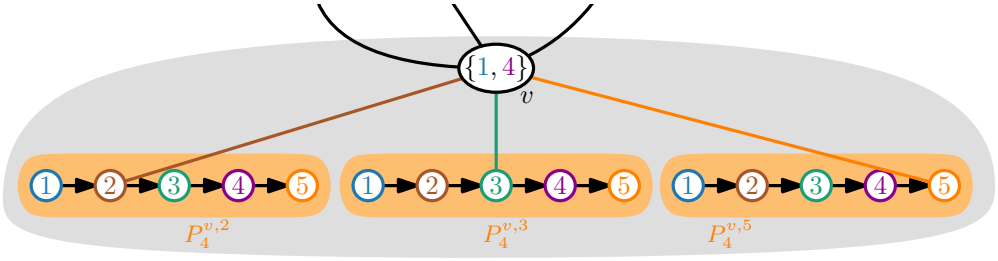


Fig. 4.7: An example of the additions for a vertex v in an instance of ListColoring with $\ell = 5$ and $L(v) = \{1, 4\}$. Each added path $P_4^{v,j}$ prevents v from being colored with the color j in a proper 5-coloring.

As we can assume w.l.o.g. that $\ell \leq n^2$, where n is the number of vertices in G, the reduction is possible in polynomial time. It remains to show that the parameters remain bounded. We only modified G by adding paths, each connected via a single edge to a vertex in G. Thus, no new cycles were created, and feedback vertex set remains the same.

To show that pathwidth remains bounded using this reduction, we first show that we can perform a slightly different reduction that gives a simpler argument for pathwidth remaining bounded. Instead of adding the path $P_{\ell-1}^{v,j}$ for each forbidden color $j \in [\ell] - L(v)$, we add the path $P_{\ell-1}^{v}$ once for each vertex v, and connect it to v via the edges $\{v, w_j^v\}$ for each forbidden color $j \in [\ell] - L(v)$. This still enforces the color list of v, and is thus a valid reduction. In the following, we show that this reduction increases the pathwidth by at most 2. Given a path decomposition $\mathcal{T}$ of G of minimal width $\mathrm{tw}(G)$, we construct a path decomposition $\mathcal{T}'$ of G' as follows: For each vertex v of G, let B_t be a bag containing v that contains at most $\mathrm{tw}(G) + 1$ vertices. To obtain $\mathcal{T}'$, we replace the node t with a path $\langle t_0, t_1, \ldots, t_{\ell-1}, t_\ell \rangle$. Let $\langle t_1^v, \ldots, t_{\ell-1}^v \rangle$ be the nodes of the path decomposition of $P_{\ell-1}^v$ with bags $B_{t_i^v} = \{w_i^v, w_{i+1}^v\}$ for $i \in [\ell - 1]$. We set $B_{t_0} = B_t$, $B_{t_i} = B_t \cup B_{t_i^v}$ for $i \in [\ell - 1]$, and $B_{t_\ell} = B_t$. As $P_{\ell-1}^v$ is only connected to v, which is contained in B_t, this results in a valid path decomposition of G'. As the new bags contain at most two additional vertices, the width of the path decomposition of G' is at most the width of the path decomposition of G plus 2. For the original reduction where for each vertex we added several adjacent paths, we can apply the same argument to each added path, resulting in an overall increase of pathwidth by at most 2. □

Combining the W[1]-hardness of ListColoring with the parameterized reduction to MixedColoring we obtain the following theorem.

Theorem 4.14. *MixedColoring is W[1]-hard w.r.t. feedback vertex set as well as w.r.t. pathwidth.*

The W[1]-hardness w.r.t. treewidth follows from the fact that we showed W[1]-hardness for parameters that upper bound treewidth, in this case pathwidth and feedback vertex set. The resulting parameterized complexity of MixedColoring is shown in Figure 1.1.

5 Conclusion

In this thesis, we have developed exact single-exponential time algorithms to color mixed graphs. In the following, we take a look at open problems and directions for future research.

General Algorithms (Sections 3.1 to 3.3) We were able to generalize Lawler's [Law76] algorithm to mixed graphs, but were not able to generalize faster algorithms [Epp01, Bys04, BHK09], due to arcs imposing a strict partial order on the colors. It would be interesting to see if there are ways to generalize these faster algorithms. This could be done, for example, by finding a way to quickly count the number of proper covers of mixed graphs with independent sets, as this would allow for the usage of the inclusion-exclusion approach. Another aspect would be to find new properties that can be exploited algorithmically, as we did for the polynomial-space algorithm in Lemma 3.4.

Fixed Number of Colors (Section 3.4) For a fixed number of colors, we have shown that MixedColoring can be reduced to CSP, which yields fast algorithms if there are few vertices of small rank. Furthermore, we have reduced 3-MixedColoring to 3-Coloring, which allows us to use existing algorithms for 3-Coloring on mixed graphs. For more than three colors, however, we were not able to find a reduction that maintains the size of the graph. It is an intriguing question whether reductions that add a constant number of vertices exist, or if such reductions are impossible. We have also generalized two coloring techniques of Byskov [Bys04] to mixed graphs, which allowed us to obtain algorithms for k-MixedColoring from algorithms for $(k-1)$-MixedColoring and $\lceil k/2 \rceil$-MixedColoring, respectively. As all of our algorithms for k-MixedColoring are ultimately based on 3-Coloring, any improvement for 3-Coloring would also improve our k-MixedColoring algorithms.

Parameterization (Chapter 4) On the parameterized side, we applied Courcelle's Theorem and generalized a treewidth dynamic program to obtain FPT and XP runtimes. We were able to propagate the FPT and XP results to further parameters, especially to parameters on the transitive closure of mixed graphs, by generalizing a bound on the chromatic number of mixed interval graphs by Gutowski et al. [GJK$^+$23]. In doing so, we have developed a framework to obtain tight bounds on the chromatic number of mixed graphs from tight bounds on the chromatic number of undirected graphs. However, we were not able to show tightness of the bound $\chi \leq (\Delta + 1)(\mathrm{tw} + 1)$. It remains to be seen whether this bound can be improved, or if it is already tight. Another open question is whether the treewidth dynamic program, Theorem 4.3, is optimal under SETH,

as the current lower bound has only been shown for undirected graphs. It is also still unclear whether there exists an XP algorithm parameterized by cliquewidth. Note that, for undirected graphs, such an algorithm exists. Furthermore, a large part of the graph parameter hierarchy remains unexplored.

Variants Other aspects which have not received much consideration are variants of MixedColoring. For example, ListColoring, which we have used throughout this thesis, generalizes naturally to mixed graphs. Furthermore, it seems that, Lemma 3.1, the property used by several of our algorithms and thus the algorithms themselves, can be generalized to ListColoring. Other results, such as the treewidth, the MSO_1-formula, and, in particular, the reduction to CSP, also seem to generalize to ListColoring. This raises the question whether the algorithms and results of this thesis can also be applied to other coloring variants, such as DefectiveColoring, CliqueColoring, or BoundedColoring; see [CCW86, DSSW91, KGS95] for the respective definitions. However, for some of these problems it is not clear how they would generalize to mixed graphs, such as in the case of DefectiveColoring or CliqueColoring. Noth of these problems rely on improper colorings, i.e., colorings where some adjacent vertices may have the same color. For these problems it would be necessary to define how arcs should be treated, i.e., whether the constraints imposed by them may be violated or not.

Bibliography

[ALS91] Stefan Arnborg, Jens Lagergren, and Detlef Seese: Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991, 10.1016/0196-6774(91)90006-K.

[Bar20] Samuel Frederic Barr: Courcelle's theorem: Overview and applications. Bachelor's thesis, Oberlin College, 2020. `https://digitalcommons.oberlin.edu/honors/679/`.

[BCH+25] Andreas Björklund, Radu Curticapean, Thore Husfeldt, Petteri Kaski, and Kevin Pratt: Fast deterministic chromatic number under the asymptotic rank conjecture. In Yossi Azar and Debmalya Panigrahi (editors): *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2804–2818. SIAM, 2025, 10.1137/1.9781611978322.91.

[BE05] Richard Beigel and David Eppstein: 3-coloring in time $O(1.3289^n)$. *J. Algorithms*, 54(2):168–204, 2005, 10.1016/J.JALGOR.2004.06.008.

[BHK09] Andreas Björklund, Thore Husfeldt, and Mikko Koivisto: Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009, 10.1137/070683933.

[BK06] Hans L. Bodlaender and Dieter Kratsch: An exact algorithm for graph coloring with polynomial memory. Technical report, Department of Information and Computing Sciences, Utrecht University, 2006. `https://core.ac.uk/outputs/39718818/`.

[Bod96] Hans L. Bodlaender: A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996, 10.1137/S0097539793251219.

[Bys04] Jesper Makholm Byskov: Enumerating maximal independent sets with applications to graph colouring. *Oper. Res. Lett.*, 32(6):547–556, 2004, 10.1016/J.ORL.2004.03.002.

[CCW86] Lenore J. Cowen, Robert Cowen, and Douglas R. Woodall: Defective colorings of graphs in surfaces: Partitions into subgraphs of bounded valency. *J. Graph Theory*, 10(2):187–195, 1986, 10.1002/JGT.3190100207.

[CE12] Bruno Courcelle and Joost Engelfriet: *Graph Structure and Monadic Second-Order Logic – A Language-Theoretic Approach*, volume 138 of *Encyclopedia*

of Mathematics and Its Applications. Cambridge University Press, 2012, 10.1017/cbo9780511977619.

[CFK+15] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh: *Parameterized Algorithms*. Springer, 2015, 10.1007/978-3-319-21275-3.

[Chr71] Nicos Christofides: An algorithm for the chromatic number of a graph. *Comput. J.*, 14(1):38–39, 1971, 10.1093/COMJNL/14.1.38.

[CMR00] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics: Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000, 10.1016/S0166-218X(00)00221-3.

[CO00] Bruno Courcelle and Stephan Olariu: Upper bounds to the clique width of graphs. *Discret. Appl. Math.*, 101(1–3):77–114, 2000, 10.1016/S0166-218X(99)00184-5.

[Cou90] Bruno Courcelle: The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990, 10.1016/0890-5401(90)90043-H.

[Dam19] Peter Damaschke: Parameterized mixed graph coloring. *J. Comb. Optim.*, 38(2):362–374, 2019, 10.1007/S10878-019-00388-Z.

[DF13] Rodney G. Downey and Michael R. Fellows: *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013, 10.1007/978-1-4471-5559-1.

[DSSW91] Dwight Duffus, Bill Sands, Norbert Sauer, and Robert E. Woodrow: Two-colouring all two-element maximal antichains. *J. Comb. Theory A*, 57(1):109–116, 1991, 10.1016/0097-3165(91)90009-6.

[dW97] Dominique de Werra: Restricted coloring models for timetabling. *Discrete Math.*, 165–166:161–170, 1997, 10.1016/S0012-365X(96)00208-7.

[Epp01] David Eppstein: Small maximal independent sets and faster exact graph coloring. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Roberto Tamassia (editors): *Workshop on Algorithms and Data Structure (WADS)*, volume 2125 of *LNCS*, pages 462–470. Springer, 2001, 10.1007/3-540-44634-6_42.

[FFL+11] Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen: On the complexity of some colorful problems parameterized by treewidth. *Inf. Comput.*, 209(2):143–153, 2011, 10.1016/J.IC.2010.11.026.

[FG04] Markus Frick and Martin Grohe: The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Log.*, 130(1–3):3–31, 2004, 10.1016/J.APAL.2004.01.007.

[FGLS10] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh: Intractability of clique-width parameterizations. *SIAM J. Comput.*, 39(5):1941–1956, 2010, 10.1137/080742270.

[FGS07] Fedor V. Fomin, Serge Gaspers, and Saket Saurabh: Improved exact algorithms for counting 3- and 4-colorings. In Guohui Lin (editor): *13th Annual International Conference on Computing and Combinatorics (COCOON)*, volume 4598 of *LNCS*, pages 65–74. Springer, 2007, 10.1007/978-3-540-73545-8_9.

[FHRV09] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette: On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009, 10.1016/J.TCS.2008.09.065.

[FK10] Fedor V. Fomin and Dieter Kratsch: *Exact Exponential Algorithms*, volume 138 of *Texts in Theoretical Computer Science*. Springer Berlin, Heidelberg, 2010, 10.1007/978-3-642-16533-7.

[GJK+23] Grzegorz Gutowski, Konstanty Junosza-Szaniawski, Felix Klesen, Pawel Rzazewski, Alexander Wolff, and Johannes Zink: Coloring and recognizing mixed interval graphs. In Satoru Iwata and Naonori Kakimura (editors): *34th International Symposium on Algorithms and Computation (ISAAC)*, volume 283 of *LIPIcs*, pages 36:1–36:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 10.4230/LIPICS.ISAAC.2023.36.

[GL23] Serge Gaspers and Edward J. Lee: Faster graph coloring in polynomial space. *Algorithmica*, 85(2):584–609, 2023, 10.1007/S00453-022-01034-7.

[GMR+22] Grzegorz Gutowski, Florian Mittelstädt, Ignaz Rutter, Joachim Spoerhase, Alexander Wolff, and Johannes Zink: Coloring mixed and directional interval graphs. In Patrizio Angelini and Reinhard von Hanxleden (editors): *30th International Symposium on Graph Drawing and Network Visualization (GD)*, volume 13764 of *LNCS*, pages 418–431. Springer, 2022, 10.1007/978-3-031-22203-0_30.

[HKdW97] Pierre Hansen, Julio Kuplinsky, and Dominique de Werra: Mixed graph colorings. *Math. Methods Oper. Res.*, 45(1):145–160, 1997, 10.1007/BF01194253.

[Jaf20] Lars Jaffke: Bounded width graph classes in parameterized algorithms. PhD thesis, University of Bergen, 2020. `https://lars-jaffke.github.io/pdfs/Jaffke-thesis.pdf`.

[KGS95] Damon Kaller, Arvind Gupta, and Thomas C. Shermer: The χ_t-coloring problem. In Ernst W. Mayr and Claude Puech (editors): *12th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 900 of *LNCS*, pages 409–420. Springer, 1995, 10.1007/3-540-59042-0_92.

[KL23] Tuukka Korhonen and Daniel Lokshtanov: An improved parameterized algorithm for treewidth. In Barna Saha and Rocco A. Servedio (editors): *55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 528–541. ACM, 2023, 10.1145/3564246.3585245.

[KR03] Daniel Kobler and Udi Rotics: Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Appl. Math.*, 126(2–3):197–221, 2003, 10.1016/S0166-218X(02)00198-1.

[Kub89] Marek Kubale: Interval vertex-coloring of a graph with forbidden colors. *Discrete Math.*, 74(1–2):125–136, 1989, 10.1016/0012-365X(89)90204-5.

[Law76] Eugene L. Lawler: A note on the complexity of the chromatic number problem. *Inf. Process. Lett.*, 5(3):66–67, 1976, 10.1016/0020-0190(76)90065-x.

[LMS18] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh: Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018, 10.1145/3170442.

[Mei23] Lucas Meijer: 3-coloring in time $O(1.3217^n)$. *CoRR*, abs/2302.13644, 2023, 10.48550/ARXIV.2302.13644.

[MM65] John W. Moon and Leo Moser: On cliques in graphs. *Isr. J. Math.*, 3:23–28, 1965, 10.1007/bf02760024.

[NO12] Jaroslav Nešetřil and Patrice Ossona de Mendez: *Sparsity – Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and Combinatorics*. Springer, 2012, 10.1007/978-3-642-27875-4.

[RdW08] Bernard Ries and Dominique de Werra: On two coloring problems in mixed graphs. *Eur. J. Comb.*, 29(3):712–725, 2008, 10.1016/J.EJC.2007.03.006.

[Sot20] Yuri N. Sotskov: Mixed graph colorings: A historical review. *Mathematics*, 8(3):385, 2020, 10.3390/math8030385.

[ST76] Yuri N. Sotskov and V.S. Tanaev: Chromatic polynomial of a mixed graph. *Vestsi Akademii Navuk BSSR, Ser. Fiz. Mat. Navuk*, 6:20–23, 1976.

[Str94] Volker Strassen: Algebra and complexity. In Anthony Joseph, Fulbert Mignot, François Murat, Bernard Prum, and Rudolf Rentschler (editors): *First European Congress of Mathematics*, volume 120 of *Progress in Mathematics*, pages 429–446. Birkhäuser Basel, 1994, 10.1007/978-3-0348-9112-7_18.

[Tra22] Duc Long Tran: Expanding the graph parameter hierarchy. Bachelor's thesis, Institute of Software Engineering and Theoretical Computer Science, Technische Universität Berlin, 2022. `https://fpt.akt.tu-berlin.de/publications/theses/BA-Duc-Long-Tran.pdf`.

[WGJ+24] Pu Wu, Huanyu Gu, Huiqin Jiang, Zehui Shao, and Jin Xu: A faster algorithm for the 4-coloring problem. In Timothy M. Chan, Johannes Fischer, John Iacono, and Grzegorz Herman (editors): *32nd Annual European Symposium on Algorithms (ESA)*, volume 308 of *LIPIcs*, pages 103:1–103:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 10.4230/LIPICS.ESA.2024.103.

[Wol] Wolfram Research, Inc.: Mathematica, Version 14.0. `https://www.wolfram.com/mathematica`, Champaign, IL, 2024.

[Zam21] Or Zamir: Breaking the 2^n barrier for 5-coloring and 6-coloring. In Nikhil Bansal, Emanuela Merelli, and James Worrell (editors): *48th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 198 of *LIPIcs*, pages 113:1–113:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 10.4230/LIPICS.ICALP.2021.113.

A Code

```
In[1]:= N[Reduce[1/(a^(a/(1 - a))*(1 - a)) == 81/64*(172/81)^(1/a)
          && 1/2 <= a < 1, a, Reals]]

Out[1]= a == 0.563964

In[2]:= 81/64*(172/81)^(1/0.563964)

Out[2]= 4.81068
```

Code A.1: Mathematica code used to compute good values for α and c in the runtime analysis of the polynomial-space algorithm from Theorem 3.5.

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Die benutzte Literatur sowie sonstige Hilfsquellen sind vollständig angegeben. Wörtlich oder dem Sinne nach dem Schrifttum oder dem Internet entnommene Stellen sind unter Angabe der Quelle kenntlich gemacht.

Weitere Personen waren an der geistigen Leistung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich nicht die Hilfe eines Ghostwriters oder einer Ghostwriting-Agentur in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar Geld oder geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Arbeit stehen.

Der Durchführung einer elektronischen Plagiatsprüfung stimme ich hiermit zu. Die eingereichte elektronische Fassung der Arbeit ist vollständig. Mir ist bewusst, dass nachträgliche Ergänzungen ausgeschlossen sind.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung zur Versicherung der selbstständigen Leistungserbringung rechtliche Folgen haben kann.

Würzburg, den 3. September 2025

.....

Antonio Lauerbach