Closing Complexity Gaps for Coloring Problems on H-Free Graphs ⋆

Petr A. Golovach[1], Daniël Paulusma[2], and Jian Song[2]

[1] Department of Informatics, Bergen University,
PB 7803, 5020 Bergen, Norway
petr.golovach@ii.uib.no

[2] School of Engineering and Computing Sciences, Durham University,
Science Laboratories, South Road, Durham DH1 3LE, United Kingdom
{daniel.paulusma,jian.song}@durham.ac.uk

Abstract. If a graph G contains no subgraph isomorphic to some graph H, then G is called H-free. A coloring of a graph $G = (V, E)$ is a mapping $c : V \to \{1, 2, \ldots\}$ such that no two adjacent vertices have the same color, i.e., $c(u) \neq c(v)$ if $uv \in E$; if $|c(V)| \leq k$ then c is a k-coloring. The Coloring problem is to test whether a graph has a coloring with at most k colors for some integer k. The Precoloring Extension problem is to decide whether a partial k-coloring of a graph can be extended to a k-coloring of the whole graph for some integer k. The List Coloring problem is to decide whether a graph allows a coloring, such that every vertex u receives a color from some given set $L(u)$. By imposing an upper bound ℓ on the size of each $L(u)$ we obtain the ℓ-List Coloring problem. We first classify the Precoloring Extension problem and the ℓ-List Coloring problem for H-free graphs. We then show that 3-List Coloring is NP-complete for n-vertex graphs of minimum degree $n-2$, i.e., for complete graphs minus a matching, whereas List Coloring is fixed-parameter tractable for this graph class when parameterized by the number of vertices of degree $n-2$. Finally, for a fixed integer $k > 0$, the List k-Coloring problem is to decide whether a graph allows a coloring, such that every vertex u receives a color from some given set $L(u)$ that must be a subset of $\{1, \ldots, k\}$. We show that List 4-Coloring is NP-complete for P_6-free graphs, where P_6 is the path on six vertices. This completes the classification of List k-Coloring for P_6-free graphs.

Keywords. graph coloring, precoloring extension, list coloring, forbidden induced subgraph.

1 Introduction

Graph coloring involves the labeling of the vertices of some given graph by integers called colors such that no two adjacent vertices receive the same color. The corresponding decision problem is called Coloring and is to decide whether a

⋆ This work has been supported by EPSRC (EP/G043434/1) and ERC (267959); an extended abstract of it has been accepted for ISAAC 2012.

graph can be colored with at most k colors for some given integer k. Because Coloring is NP-complete for any fixed $k \geq 3$, its computational complexity has been widely studied for special graph classes, see e.g. the surveys of Randerath and Schiermeyer [24] and Tuza [27]. In this paper, we consider the Coloring problem together with two natural and well-studied variants, namely Precoloring Extension and List Coloring for graphs characterized by one or more forbidden induced subgraphs. Before we summarize related results and explain our new results, we first state the necessary terminology.

1.1 Terminology.

We only consider finite undirected graphs without loops and multiple edges. We refer to the textbook of Bondy and Murty [1] for any undefined graph terminology.

A *coloring* of a graph $G = (V, E)$ is a mapping $c : V \to \{1, 2, \ldots\}$ such that $c(u) \neq c(v)$ whenever $uv \in E$. We call $c(u)$ the *color* of u. A *k-coloring* of G is a coloring c of G with $1 \leq c(u) \leq k$ for all $u \in V$. The problem k-Coloring is to decide whether a given graph admits a k-coloring. Here, k is *fixed*, i.e., not part of the input. If k is part of the input, then we denote the problem as Coloring. A *list assignment* of a graph $G = (V, E)$ is a function L that assigns a list $L(u)$ of so-called *admissible* colors to each $u \in V$. If $L(u) \subseteq \{1, \ldots, k\}$ for each $u \in V$, then L is also called a *k-list assignment*. The *size* of a list assignment L is the maximum list size $|L(u)|$ over all vertices $u \in V$. We say that a coloring $c\colon V \to \{1, 2, \ldots\}$ *respects* L if $c(u) \in L(u)$ for all $u \in V$.

The List Coloring problem is to test whether a given graph has a coloring that respects some given list assignment. For a fixed integer k, the List k-Coloring problem has as input a graph G with a k-list assignment L and asks whether G has a coloring that respects L. For a fixed integer ℓ, the ℓ-List Coloring problem has as input a graph G with a list assignment L of size at most ℓ and asks whether G has a coloring that respects L. In *precoloring extension* we assume that a (possibly empty) subset $W \subseteq V$ of G is precolored by a *precoloring* $c_W : W \to \{1, 2, \ldots k\}$ for some integer k, and the question is whether we can extend c_W to a k-coloring of G. For a fixed integer k, we denote this problem as k-Precoloring Extension. If k is part of the input, then we denote this problem as Precoloring Extension.

Note that k-Coloring can be viewed as a special case of k-Precoloring Extension by choosing $W = \emptyset$, and that k-Precoloring Extension can be viewed as a special case of List k-Coloring by choosing $L(u) = \{c_W(u)\}$ if $u \in W$ and $L(u) = \{1, \ldots, k\}$ if $u \in W \setminus V$. Moreover, List k-Coloring can be readily seen as a special case of k-List Coloring. Hence, we can make the following two observations for a graph class $\mathcal{G}$. If k-Coloring is NP-complete for $\mathcal{G}$, then k-Precoloring Extension is NP-complete for $\mathcal{G}$, and consequently, List k-Coloring and hence k-List Coloring are NP-complete for $\mathcal{G}$. Conversely, if k-List Coloring is polynomial-time solvable on $\mathcal{G}$, then List k-Coloring is polynomial-time solvable on $\mathcal{G}$, and consequently, k-Precoloring Extension is polynomial-time solvable on $\mathcal{G}$, and then also k-Coloring is polynomial-time solvable on $\mathcal{G}$.

The graph P_r denotes the *path* on r vertices, i.e., $V(P_r) = \{u_1, \ldots, u_r\}$ and $E(P_r) = \{u_i u_{i+1} \mid 1 \leq i \leq r-1\}$. The graph K_r denotes the *complete* graph on r vertices, i.e., $V(K_r) = \{u_1, \ldots, u_r\}$ and $E(K_r) = \{u_i u_j \mid 1 \leq i < j \leq r\}$. The vertex set of a complete graph is called a *clique*. The disjoint union of two graphs G and H is denoted $G + H$, and the disjoint union of r copies of G is denoted rG.

Let G be a graph and $\{H_1, \ldots, H_p\}$ be a set of graphs. We say that G is $(H_1, \ldots, H_p)$*-free* if G has no induced subgraph isomorphic to a graph in $\{H_1, \ldots, H_p\}$; if $p = 1$, we sometimes write H_1-free instead of (H_1)-free.

Let G be a graph. The *complement* of G denoted by $\overline{G}$ has vertex set $V(G)$ and an edge between two distinct vertices if and only if these vertices are not adjacent in G. We denote the *neighborhood* of a vertex u in G by $N(u) = \{v \in V(G) \mid uv \in E(G)\}$. For a set $S \subseteq V(G)$, we let $G[S] = (S, \{uv \in E(G) \mid u, v \in S\})$ denote the subgraph of G *induced by* S.

1.2 Related and New Results.

Kráľ, Kratochvíl, Tuza and Woeginger [17] completely determined the computational complexity of Coloring for H-free graphs by showing the following dichotomy.

Theorem 1 ([17]). *Let H be a fixed graph. If H is a (not necessarily proper) induced subgraph of P_4 or of P_1+P_3, then* Coloring *can be solved in polynomial time for H-free graphs; otherwise it is* NP*-complete for H-free graphs.*

In Section 2 we use Theorem 1 and a number of other results from the literature to obtain the following two dichotomies, which complement Theorem 1. Theorem 3 shows amongst others that Precoloring Extension is polynomial-time solvable on (P_1+P_3)-free graphs, which contain the class of $3P_1$-free graphs, i.e., complements of triangle-free graphs. As such, this theorem also generalizes a result of Hujter and Tuza [13] who showed that Precoloring Extension is polynomial-time solvable on complements of bipartite graphs.

Theorem 2. *Let ℓ be a fixed integer, and let H be a fixed graph. If $\ell \leq 2$ or H is a (not necessarily proper) induced subgraph of P_3, then ℓ-*List Coloring *is polynomial-time solvable on H-free graphs; otherwise ℓ-*List Coloring *is* NP*-complete for H-free graphs.*

Theorem 3. *Let H be a fixed graph. If H is a (not necessarily proper) induced subgraph of P_4 or of $P_1 + P_3$, then* Precoloring Extension *can be solved in polynomial time for H-free graphs; otherwise it is* NP*-complete for H-free graphs.*

In Section 3 we consider the List Coloring problem for graphs that are obtained from a complete graph after removing the edges of some matching. We call such a graph a *complete graph minus a matching.* Note that a graph G on n vertices is a complete graph minus a matching if and only if G is $(3P_1, P_1 + P_2)$-free if and only if G has minimum degree at least $n-2$. Our motivation to study complete graphs minus a matching comes from the fact that List Coloring is NP-complete on almost all non-trivial graph classes, such as can be deduced from Theorem 2 and from other results known in the literature. For example, List Coloring is NP-complete for complete split graphs [16], for line graphs of complete graphs [20], and moreover, even for complete bipartite graphs [16], which are $(P_1 + P_2)$-free and for (not necessarily vertex-disjoint) unions of two complete graphs [15], which are $3P_1$-free. We refer to Table 1 in the paper by Bonomo, Durán and Marenco [2] for an overview. It is known that List Coloring can be solved in polynomial time for block graphs [15], which contain the class of complete graphs and trees. Our aim was to extend this positive result. However, as we show, already 3-List Coloring is NP-complete even on graphs that are $(P_1 + P_2)$-free *and* $3P_1$-free, i.e., complete graphs minus a matching.

On the other hand, we can use parameterized complexity to get a more positive result. In parameterized complexity theory, we consider the problem input as a pair (I, p), where I is the main part and p is the parameter. A problem is *fixed-parameter tractable* if an instance (I, p) can be solved in time $f(p)|I|^c$, where f is a computable function that only depends on p, and c is a constant independent of p. As a positive result, we show that List Coloring is fixed-parameter tractable for complete graphs minus a matching when parameterized by the number of matching edges removed. In fact we show a slightly stronger result, namely that List Coloring is fixed-parameter tractable with parameter p for *complete*$-$*pe* graphs, which are those graphs that can be modified into complete graphs by adding at most p edges.

The above result can be placed in a study of parameterized coloring problems for graph classes $\mathcal{F} + pe$, $\mathcal{F} - pe$, $\mathcal{F} + pv$ and $\mathcal{F} - pv$, which consist of all graphs that can be transformed into a graph from a class $\mathcal{F}$ by deleting at most p edges, adding at most p edges, deleting at most p vertices and adding at most p vertices, respectively. This research was initiated by Cai [6] who showed that Coloring with parameter p is fixed-parameter tractable on split$+pe$ graphs, and that this is unlikely, i.e., W[1]-hard for split$+pv$ graphs. Cai [6] also showed that whenever Coloring is polynomial-time solvable on a graph class $\mathcal{F}$ that is closed under edge contraction, then Coloring is fixed-parameter tractable on $\mathcal{F} - pe$ when parameterized by p. This yields fixed-parameter tractability of Coloring with parameter p for split$-pe$ graphs, and also for example, for interval$-pe$ graphs and chordal$-pe$ graphs. The same result is obtained for split$-pv$, interval$-pv$, and chordal$-pv$, because split, interval and chordal graphs are closed under vertex deletion (and for such graph classes $\mathcal{F}$ we obtain $\mathcal{F} - pv = \mathcal{F}$). Marx [21] extended these results by showing that Coloring is fixed-parameter tractable on interval$+pe$ graphs and chordal$+pe$ graphs but W[1]-hard for interval$+pv$ graphs and chordal$+pv$ graphs. Jansen and Kratsch [14] considered the k-Coloring problem for various graph classes $\mathcal{F} + pv$ in order to obtain polynomial kernels.

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

Table 1. The complexity of k-Coloring, k-Precoloring Extension and List k-Coloring on P_r-free graphs for fixed k and r. The bold entry is our new result.

In Section 4, we consider the List k-Coloring problem. As we explained in Section 1.1, this problem is closely related to the problems k-Coloring and k-Precoloring Extension. In contrast to Coloring and Precoloring Extension (cf. Theorems 1 and 3), the complexity classifications of k-Coloring and k-Precoloring Extension for H-free graphs are yet to be completed, even when H is a path. Hoàng et al. [10] showed that for any $k \geq 1$, the k-Coloring problem can be solved in polynomial time for P_5-free graphs. Randerath and Schiermeyer [23] showed that 3-Coloring can be solved in polynomial time for P_6-free graphs. These results are complemented by the following hardness results of Huang [12]: 4-Coloring is NP-complete for P_7-free graphs and 5-Coloring is NP-complete for P_6-free graphs. Also the computational complexity of the List k-Coloring problem is still open for P_r-free graphs. Hoàng et al. [10] showed that their polynomial-time result on k-Coloring for P_5-free graphs is in fact valid for List k-Coloring for any fixed $k \geq 1$. Broersma et al. [3] generalized the polynomial-time result of Randerath and Schiermeyer [23] for 3-Coloring on P_6-free graphs to List 3-Coloring on P_6-free graphs. Table 1 summarizes all existing results for these three problems restricted to P_r-free graphs. We prove that List 4-Coloring is NP-complete for P_6-free graphs. Because List 3-Coloring is polynomial-time solvable on P_6-free graphs [3], we completely characterized the computational complexity of List k-Coloring for P_6-free graphs. In Table 1 we indicate this result in bold. All cases marked by "?" in Table 1 are still open.

2 Classifying Precoloring Extension and 3-List Coloring

For the proof of Theorem 2 we need the following lemma, which is well-known (cf. [2]). We give its proof in order to explain the bound on the running time stated in this lemma; we need this bound for our fixed-parameter tractability result in Section 3.

Lemma 1. List Coloring *can be solved in* $O((n+k)^{\frac{5}{2}})$ *time on* n*-vertex complete graphs with a* k*-list assignment.*

Proof. Let $G = (V, E)$ be a complete graph on n vertices $u_1, \ldots, u_n$ with some k-list assignment L. Let $V(G) = \{u_1, \ldots, u_n\}$. Then we construct a bipartite graph B as follows. One partition class of B consists of n vertices $u_1, \ldots, u_n$, whereas the other partition class consists of vertices $1, \ldots, k$. We add an edge between two vertices u_i and j if and only if $j \in L(u_i)$. Now G has a coloring that respects L if and only if B has a matching that contains an edge with u_i as one of its end-vertices for $i = 1, \ldots, n$. We can solve the latter problem in $O((n+k)^{\frac{5}{2}})$ time by using the Hopcroft-Karp algorithm [11]. □

We are now ready to give the proof of Theorem 2; we restate this theorem below.

Theorem 2. *Let ℓ be a fixed integer, and let H be a fixed graph. If $\ell \leq 2$ or H is a (not necessarily proper) induced subgraph of P_3, then ℓ-*List Coloring *is polynomial-time solvable on H-free graphs; otherwise ℓ-*List Coloring *is* NP*-complete for H-free graphs.*

Proof. Early papers by Erdös, Rubin and Taylor [8] and Vizing [28] already observed that 2-List Coloring is polynomial-time solvable on general graphs. Hence, we can focus on the case $\ell \geq 3$. Because the ℓ-Coloring problem is a special case of the ℓ-List Coloring problem, the following results are useful. Kamiński and Lozin [19] showed that for any $k \geq 3$, the k-Coloring problem is NP-complete for the class of graphs of girth (the length of a shortest induced cycle) at least p for any fixed $p \geq 3$. Their result implies that for any $\ell \geq 3$, the ℓ-Coloring problem, and consequently, the ℓ-List Coloring problem is NP-complete for the class of H-free graphs whenever H contains a cycle.

The proof of Theorem 4.5 in the paper by Jansen and Scheffler [16] is to show that 3-List Coloring is NP-complete on P_4-free graphs but as a matter of fact shows that 3-List Coloring is NP-complete on complete bipartite graphs, which are $(P_1 + P_2)$-free. The proof of Theorem 11 in the paper by Jansen [15] is to show that List Coloring is NP-complete for (not necessarily vertex-disjoint) unions of two complete graphs but as a matter of fact shows that 3-List Coloring is NP-complete for these graphs. As the union of two complete graphs is $3P_1$-free, this means that 3-List Coloring is NP-complete for $3P_1$-free graphs.

The above results leave us with the case when H is a (not necessarily proper) induced subgraph of P_3. By Lemma 1 we can solve List Coloring in polynomial time on complete graphs. This means that we can solve ℓ-List Coloring in polynomial time on P_3-free graphs for any $\ell \geq 1$. Hence we have proven Theorem 2. □

We are now ready to prove Theorem 3; we restate this theorem below.

Theorem 3. *Let H be a fixed graph. If H is a (not necessarily proper) induced subgraph of P_4 or of $P_1 + P_3$, then* Precoloring Extension *can be solved in polynomial time for H-free graphs; otherwise it is* NP*-complete for H-free graphs.*

Proof. Let H be a fixed graph. If H is not an induced subgraph of P_4 or of $P_1 + P_3$, then Theorem 1 tells us that Coloring, and consequently, Precoloring Extension is NP-complete for H-free graphs. Jansen and Scheffler [16] showed that Precoloring Extension is polynomial-time solvable for P_4-free graphs. Hence, we are left with the case $H = P_1 + P_3$.

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

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

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

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

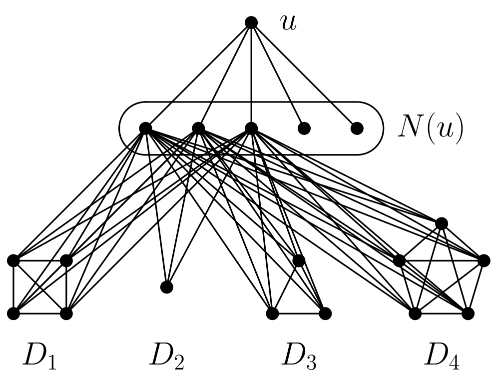

Fig. 1. An example of a graph G that shows a vertex u that belongs to a set of (at least) three independent vertices and the corresponding complete graphs $D_1, \dots, D_p$ In this example, $p = 4$ and edges between vertices in $N(u)$ have not been displayed. Also note that the vertices in $V(D_1) \cup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

Let $S = V(G) \setminus (\{u\} \cup N(u))$. Because G is $(P_1 + P_3)$-free, $G[S]$ is the disjoint union of a set of complete graphs $D_1, \dots, D_p$ for some $p \geq 2$; note that $p \geq 2$ holds, because the other two vertices of T must be in different graphs D_i and D_j. We refer to Figure 1 for an example. We will use the following claim.

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

We prove Claim 1 as follows. First suppose that w and w' are two vertices in two different graphs D_i and D_j, such that w is adjacent to some vertex $v \in N(u)$. Then w' is adjacent to v, as otherwise w' and u, v, w form an induced $P_1 + P_3$ in

G, which is not possible. Now suppose that w and w' are two vertices in the same graph D_i, say D_1, such that w is adjacent to some vertex $v \in N(u)$. Because $p \geq 2$, the graph D_2 is nonempty. Let w^* be in D_2. As we just showed, the fact that w is adjacent to v implies that w^* is adjacent to v as well. By repeating this argument with respect to w^* and w', we then find that w' is adjacent to v. Hence, we have proven Claim 1.

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

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

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

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

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

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

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

We now apply the algorithm of Hujter and Tuza [13] for solving Precoloring Extension on complements of bipartite graphs. Because G is $3P_1$-free, G has no three mutually nonadjacent vertices. Suppose that u and v are two non-adjacent vertices in W. Then every vertex of $V(G) \setminus \{u, v\}$ is adjacent to at least one of $\{u, v\}$. This means that we can remove u, v if they are both colored alike by c_W in order to obtain a new instance $(G - \{u, v\}, k-1, c_{W \setminus \{u,v\}})$ that is a yes-instance of Precoloring Extension if and only if (G, k, c_W) is a yes-instance. If u and v are colored differently by c_W, then we add an edge between them. We perform this step for any pair of non-adjacent vertices in W. Afterward, we have found in polynomial time a new instance (G^*, k^*, c_{W^*}) with the following properties. First, $|V(G^*)| \leq |V(G)|$, $k^* \leq k$ and $c_{W^*} : W^* \to \{1, \ldots, k\}$ is a precoloring defined on some clique W^* of G^*. Second, (G^*, k^*, c_{W^*}) is a yes-instance if and only if (G, k, c_W) is a yes-instance. Hence, we may consider (G^*, k^*, c_{W^*}) instead. Because W^* is a clique, we find that (G^*, k^*, c_{W^*}) is a yes-instance if and only if G^* is k^*-colorable. Because G is $3P_1$-free and G^* is obtained by only removing vertices from G, we find that G^* is $3P_1$-free as well. This means that we can solve the Coloring problem with input (G^*, k^*) by using Theorem 1 (which in this particular case comes down to computing the size of a maximum matching in the complement of G^*). This completes the proof for the case $H = P_1 + P_3$, and we have proven Theorem 3. □

3 List Coloring for Complete Graphs Minus a Matching

We prove that 3-List Coloring is NP-complete for complete graphs minus a matching. In order to do this we use a reduction from a variant of Not-All-Equal 3-Satisfiability with positive literals only, which we denote as Not-All-Equal $(\leq 3, 2/3)$-Satisfiability with positive literals. The Not-All-Equal 3-Satisfiability problem is NP-complete [25] and is defined as follows. Given a set $X = \{x_1, x_2, ..., x_n\}$ of logical variables, and a set $C = \{C_1, C_2, ..., C_m\}$ of three-literal clauses over X in which all literals are positive, does there exist a truth assignment for X such that each clause contains at least one true literal and at least one false literal? The variant Not-All-Equal $(\leq 3, 2/3)$-Satisfiability with positive literals asks the same question but takes as input an instance I that has a set of variables $\{x_1, \ldots, x_n\}$ and a set of literal clauses $\{C_1, \ldots, C_m\}$ over X with the following properties. Each C_i contains either 2 or 3 literals, and these literals are all positive. Moreover, each literal occurs in at most three different clauses. One can prove that Not-All-Equal $(\leq 3, 2/3)$-Satisfiability is NP-complete by a reduction from Not-All-Equal-3-Satisfiability via a well-known folklore trick.[3]

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

We construct G_I and L in four steps.

1. We represent every variable x_i by a vertex with $L(x_i) = \{1_i, 2_i\}$ in G_I. We say that these vertices are of x-type and these colors are of 1-type and 2-type, respectively.

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

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

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

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

3. For each variable x_j that occurs in three clauses we fix an arbitrary order of the clauses C_p, C_q, C_r, in which it occurs. Then we do as follows. First, we modify the lists of $a_{p,j}$, $a_{q,j}$, $b_{p,j}$ and $b_{q,j}$. In $L(a_{p,j})$ we replace color 1_j with a new color $1'_j$. In $L(a_{q,j})$ we replace color 1_j with a new color $1''_j$. In $L(b_{p,j})$ we replace color 2_j with a new color $2'_j$. In $L(b_{q,j})$ we replace color 2_j with a new color $2''_j$. Next we introduce four vertices $a'_{p,j}, a'_{q,j}, b'_{p,j}, b'_{q,j}$ with lists of admissible colors $\{1_j, 1'_j\}, \{1'_j, 1''_j\}, \{2_j, 2'_j\}, \{2'_j, 2''_j\}$, respectively. We say that these vertices are of a'-type or b'-type, respectively. We say that the new colors are also of 1-type or 2-type, respectively. We add edges $a_{p,j}a'_{p,j}$, $a'_{p,j}a'_{q,j}$, $a'_{p,j}x_j$, $a_{q,j}a'_{q,j}$, $b_{p,j}b'_{p,j}$, $b'_{p,j}b'_{q,j}$, $b'_{p,j}x_j$, $b_{q,j}b'_{q,j}$.
4. We add an edge between any two not yet adjacent vertices of G_I whenever they have no common color in their lists.

In Figure 2 we give an example, where in order to increase the visibility we display the complement graph $\overline{G_I}$ of G_I instead of G_I itself.

As can be seen from Figure 2, the graph $\overline{G_I}$ is isomorphic to the disjoint union of a number of P_1s and P_2s. This means that G_I is a complete graph minus a matching. We formally prove this statement in Lemma 2, whereas the hardness reduction is stated in Lemma 3.

Lemma 2. *The graph G_I is a complete graph minus a matching.*

Proof. Let $z \in V(G_|)$. We obtain the following from the construction of G_I.

Suppose that z is of x-type. Let $L(z) = \{1_j, 2_j\}$. By step 4, there is an edge between z and any vertex that has neither color 1_j nor color 2_j in its list. The only vertices that have color 1_j or 2_j in their lists are vertices of a-type, a'-type, b-type or b'-type that correspond to clauses in which z occurs. Of those vertices, the vertices of a-type and b-type are made adjacent to z in step 2, whereas the vertices of a'-type and b'-type are made adjacent to z in step 3 (if they exist). Hence, z is adjacent to all vertices of $V(G_I)\setminus\{z\}$, that is, has degree $|V(G_I)|-1$.

Suppose that z is of C-type. Let $L(z) = \{3_p, 4_p, 5_p\}$. By step 4, there is an edge between z and any vertex that has neither color 3_p nor color 4_p nor color 5_p in its list. The only vertices that have color 3_p or 4_p or 5_p in their lists are vertices of a-type or b-type that correspond to literals contained in z. These vertices are made adjacent to z in step 2. Just as in the previous case, we conclude that z is adjacent to all vertices of $V(G_I) \setminus \{z\}$, that is, has degree $|V(G_I)| - 1$.

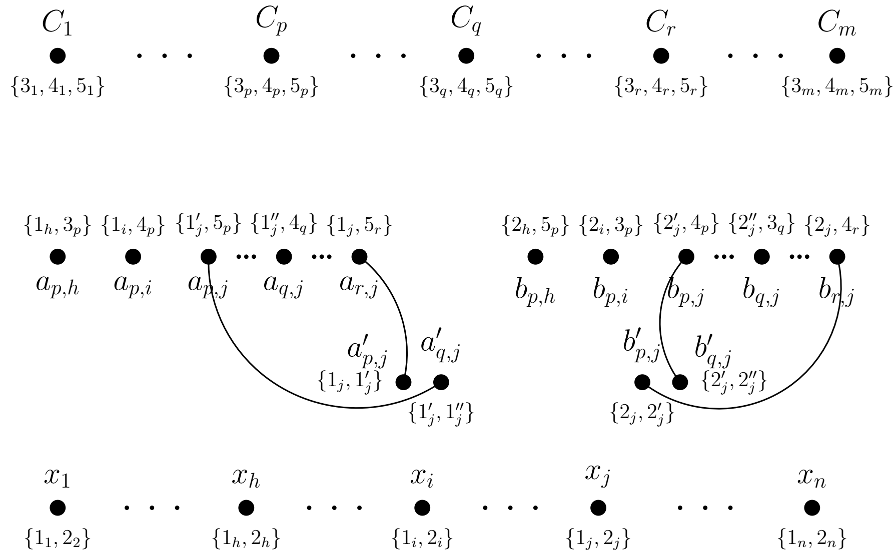

Fig. 2. An example of a graph $\overline{G_I}$ in which a clause C_p and a variable x_j are highlighted. Note that in this example C_p is a clause with ordered variables x_h, x_i, x_j, and that x_j is a variable contained in ordered clauses C_p, C_q and C_r.

Suppose that z is of a-type or b-type. We already deduced that z is adjacent to all vertices that are of x-type and of C-type. By steps 3 and 4, we find that z is adjacent to all vertices of G_I except to perhaps one vertex, which is of a'-type if z is of a-type, and which is of b'-type if z is of b-type (also see Figure 2). This means that z has degree $|V(G_I)|-2$ in this case.

Finally, if z is of a'-type or b'-type, then z is adjacent to all vertices of G_I except to perhaps one vertex, which is of a-type if z is of a'-type, and which is of b-type if z is of b'-type. This means that z has degree $|V(G_I)|-2$ in this case.

From the above we conclude that every vertex in G_I has degree at least $|V(G_I)|-2$. Hence G_I is a complete graph minus a matching. □

Lemma 3. *The graph G_I has a coloring that respects L if and only if I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal.*

Proof. First suppose that G_I has a coloring that respects L. Consider a variable x_j contained in ordered clauses C_p, C_q, and C_r. If the color of x_j is 1_j then the color of $a'_{p,j}$ is $1'_j$. Consequently, the color of $a'_{q,j}$ is $1''_j$. We conclude that the colors of $a_{p,j}$, $a_{q,j}$, $a_{r,j}$ are all not of 1-type. Similarly, if the color of x_j is 2_j, then the colors of $b_{p,j}$, $b_{q,j}$, $b_{r,j}$ are all not of 2-type. We use this observation as follows. Consider a clause C_p with three literals ordered as x_h, x_i, x_j. If x_h, x_i and x_j all have a 1-type color, then $a_{p,h}$, $a_{p,i}$ and $a_{p,j}$ have colors 3_p, 4_p and 5_p, respectively. Then there is no color available for C_p. A similar argument can be made for clauses that contain only two literals. Hence, we find that at least one literal in every clause is colored with a 2-type color. Analogously, we find that at least one literal in every clause is colored with a 1-type color. This means that the truth assignment that sets a literal to `true` if the corresponding x-type vertex has a 1-type color, and to `false` otherwise, is a satisfying truth assignment in which each clause contains at least one true and one false literal.

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

Recall that complete graphs minus a matching are exactly those graphs that are $(3P_1, P_1+P_2)$-free, or equivalently, graphs of minimum degree at least $n-2$, where n is the number of vertices. By observing that 3-List Coloring belongs to NP and using Lemmas 2 and 3, we have proven Theorem 4.

Theorem 4. *The* 3-List Coloring *problem is* NP*-complete for complete graphs minus a matching.*

To complement Theorem 4, we show that List Coloring is fixed-parameter tractable on complete graphs minus a matching when parameterized by the number of removed matching edges, or equivalently, for n-vertex graphs G of minimum degree at least $n-2$ when parameterized by the number of vertices of degree $n-2$. In fact we prove that List Coloring is fixed-parameter tractable on complete$-pe$ graphs when parameterized by p. Our proof is following the same approach as the one that Cai used for showing that whenever Coloring is polynomial-time solvable on a graph class $\mathcal{F}$ that is closed under edge contraction, then Coloring is fixed-parameter tractable on $\mathcal{F}-pe$ when parameterized by p (Theorem 3.1 in [6]).

Theorem 5. *The* List Coloring *problem can be solved in* $O(2^p(n+k)^{\frac{5}{2}})$ *time on pairs* (G, L) *where* G *is an* n*-vertex graph with* p *pairs of non-adjacent vertices and* L *is a* k*-list assignment.*

Proof. Let G be an n-vertex graph with p pairs of non-adjacent vertices. Let L be a k-list assignment of G. We branch as follows. Consider two non-adjacent vertices u and v. Then we either choose to color them alike (provided u and v have overlapping lists) or we choose to color them differently. We stress that we do not assign any color to u or v; we only made a choice to color them alike or differently. In the first case we identify u and v and assign the new vertex the list $L(u) \cap L(v)$ as its list of admissible colors. In the second case we place an edge between u and v (and leave $L(u)$ and $L(v)$ unaltered). We repeat this step in the resulting graph as long as possible. Afterward, we have obtained a complete graph K_{n-r} with a k'-list assignment L'; here, r is the number of times we chose two non-adjacent vertices to be colored alike, and $k' \leq k$.

Because the number of pairs of non-adjacent vertices is p, we have created at most 2^p new instances of List Coloring that each consist of a complete graph with some list assignment. By Lemma 1 we can test in $O((n+k)^{\frac{5}{2}})$ whether a complete graph on $n - r \leq n$ vertices has a coloring that respects a given k'-list assignment for some $k' \leq k$. Hence, we have proven Theorem 5. □

4 List 4-Coloring for P_6-Free Graphs

We prove that List 4-Coloring is NP-complete for P_6-free graphs. We use a reduction from the Not-All-Equal 3-Satisfiability problem with positive literals; recall that this is an NP-complete problem [25]. We consider an arbitrary instance I of Not-All-Equal 3-Satisfiability with variables $x_1, x_2, \ldots, x_n$ and 3-literal clauses $C_1, C_2, \ldots, C_m$ that all contain positive literals only. From I we construct a graph G_I with a 4-list assignment L. Next we show that G_I is P_6-free and that G_I has a coloring that respects L if and only if I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal.

To obtain the graph G_I with its 4-list assignment L we modify the construction of the (P_7-free but not P_6-free) graph used to prove that 4-Precoloring Extension is NP-complete for P_7-free graphs [4]. We do this as follows.

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

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

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

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

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

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

In Figure 3 we illustrate an example in which C_j is a clause with ordered variables $x_{i_1}, x_{i_2}, x_{i_3}$. The thick edges indicate the connection between these vertices and the a-type vertices of the two copies of the clause gadget. We omitted the indices from the labels of the clause gadget vertices to increase the visibility.

We now prove two lemmas. Lemma 4 shows that the graph G_I is P_6-free. In Lemma 5 we prove that G_I has a coloring that respects L if and only if I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal.

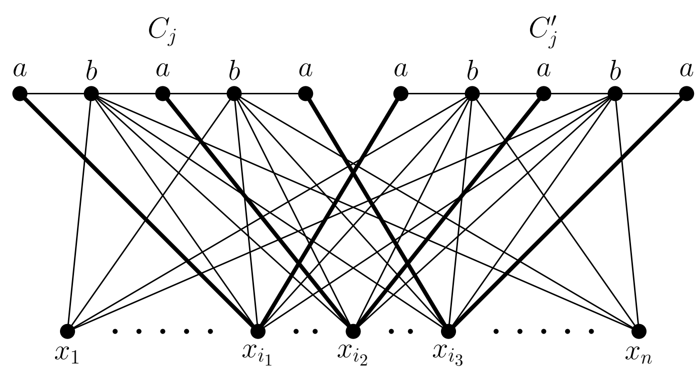

Fig. 3. The graph G_I for the clause $C_j = \{x_{i_1}, x_{i_2}, x_{i_3}\}$.

Lemma 4. *The graph G_I is P_6-free.*

Proof. Let P be an induced path in G_I. We show that P has at most 5 vertices. We distinguish the following cases.

Case 1. P contains no x-type vertex.
This means that P is contained in one clause-component, which is isomorphic to an induced P_5. Consequently, P has at most 5 vertices.

Case 2. P contains exactly one x-type vertex.
Let x_i be this vertex. Then P contains vertices of at most two clause-components. Since x_i is adjacent to all b-type vertices, we then find that P contains at most two vertices of each of the clause-components. Hence P has at most 5 vertices.

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

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

Lemma 5. *The graph G_I has a coloring that respects L if and only if I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal.*

Proof. First suppose that G_I has a coloring that respects L. Consider a clause C_j with literals ordered as x_{i_1}, x_{i_2} and x_{i_3}. Suppose that x_{i_1}, x_{i_2} and x_{i_3} all have color 2. Since the list of $a_{j,1}$ is $\{2,4\}$, we find that $a_{j,1}$ must receive color 4. Consequently, its neighbor $b_{j,1}$ must have color 3. Similarly, $a_{j,4}$ must have color 3 and $b_{j,2}$ must have color 4. This means that $a_{j,2}$ has neighbors, namely $x_{i_2}, b_{j,1}, b_{j,2}$, with colors $2, 3, 4$, respectively. However, $L(a_{j,2}) = \{2,3,4\}$. Hence, this is not possible. We conclude that at least one literal in every clause is colored with color 1. By considering the copy gadgets, we find in a similar way that at least one literal in every clause is colored with color 2. This means that the truth assignment that sets a literal to `false` if the corresponding x-type vertex has color 2, and to `true` otherwise, is a satisfying truth assignment of I in which each clause contains at least one true and at least one false literal.

Now suppose that I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal. We use color 1 to color the x-type vertices representing the true literals and color 2 to color the x-type vertices representing the false literals. Since each clause contains at least one true literal, we can color $a_{j,1}$, $a_{j,2}$ and $a_{j,3}$ respecting their lists. Similarly, since each clause contains at least one false literal, we can color $a'_{j,1}$, $a'_{j,2}$ and $a'_{j,3}$ respecting their lists. We color all other remaining uncolored vertices in a straightforward way. This completes the proof of Lemma 5. □

The observation that List 4-Coloring belongs to NP, together with Lemmas 4 and 5, immediately gives us the main result of this section.

Theorem 6. *The* List 4-Coloring *problem is* NP*-complete for P_6-free graphs.*

5 Concluding Remarks

The main tasks are to determine the computational complexity of Coloring for AT-free graphs and to solve the open cases marked "?" in Table 1. This table shows that so far all three problems k-Coloring, k-Precoloring Extension and List k-Coloring behave similarly on P_r-free graphs. Hence, our new NP-completeness result on List 4-Coloring for P_6-free graphs may be an indication that 4-Coloring for P_6-free graphs is NP-complete, or otherwise at least this result makes clear that new proof techniques not based on subroutines that solve List 4-Coloring are required for proving polynomial-time solvability.

Another question arising from Table 1 is whether there exists an integer $r \geq 7$ such that List 3-Coloring is NP-complete for P_r-free graphs. If so, this leads to an affirmative answer of the corresponding question for 3-Coloring, i.e., then we also find an integer $r' \leq 2r + 1$ such that 3-Coloring is NP-complete for $P_{r'}$-free graphs. This can be seen as follows. We modify a given P_r-free graph G that is an instance of List 3-Coloring by adding three new vertices u_1, u_2, u_3 that form a triangle. We then connect this triangle to the other vertices of G by adding edges in such a way that the required lists of admissible colors are imposed on the vertices of $V(G)$. Because u_1, u_2, u_3 form a triangle, any induced path can only use two of them. Hence, the resulting graph is P_{2r+1}-free.

Table 1 not only shows that we have completed the computational complexity classification of List k-Coloring for P_6-free graphs but also of List 4-Coloring for P_r-free graphs for all $r \geq 1$. However, we point out that there still exist several gaps in the computational complexity classification of List 4-Coloring for H-free graphs. It is known that List k-Coloring is NP-complete for H-free graphs whenever $k \geq 3$ and H is not a linear forest, i.e., a disjoint union of a collection of paths (cf. [4]). Couturier et al. [7] generalized the polynomial-time result of Hoàng et al. [10] on List k-Coloring for P_5-free graphs and fixed $k \geq 1$ to $(P_5 + sP_1)$-free graphs for any fixed integer $s \geq 0$. However, several smaller cases, such as the following, are still unresolved.

What is the computational complexity of List 4-Coloring for $(P_2 + P_3)$-free graphs, for $2P_3$-free graphs and for $(P_2 + P_4)$-free graphs?

We note that List 3-Coloring is polynomial-time solvable on $(P_2 + P_4)$-free graphs and on sP_3-free graphs for any fixed integer $s \geq 1$ [4]. Moreover, both 4-Coloring and 4-Precoloring Extension are polynomial-time solvable on (P_2+P_3)-free graphs [9], whereas List 5-Coloring is known to be NP-complete for $(P_2 + P_4)$-free graphs [7].

Another long-standing open problem, which was posed by Broersma et al. [5], is to determine the computational complexity of the Coloring problem for the class of asteroidal triple-free graphs, also known as *AT-free* graphs. An *asteroidal triple* is a set of three mutually non-adjacent vertices such that each two of them are joined by a path that avoids the neighborhood of the third, and AT-free graphs are exactly those graphs that contain no such triple. We note that unions of two complete graphs are AT-free. Hence NP-completeness of 3-List Coloring for this graph class [15] immediately carries over to AT-free graphs. Stacho [26] showed that 3-Coloring is polynomial-time solvable on AT-free graphs. Recently, Kratsch and Müller [18] extended this result by proving that List k-Coloring is polynomial-time solvable on AT-free graphs for any fixed positive integer k. Marx [22] showed that Precoloring Extension is NP-complete for proper interval graphs, which form a subclass of AT-free graphs. An *asteroidal set* in a graph G is an independent set $S \subseteq V(G)$, such that every triple of vertices of S forms an asteroidal triple. The *asteroidal number* is the size of a largest asteroidal set in G. Note that complete graphs are exactly those graphs that have asteroidal number at most one, and that AT-free graphs are exactly those graphs that have asteroidal number at most two. We observe that Coloring is NP-complete for the class of graphs with asteroidal number at most three, as this class contains the class of $4P_1$-free graphs and for the latter graph class one may apply Theorem 1.

We finish our paper by posing two more open problems. We showed that List Coloring is fixed-parameter tractable for complete$-pe$ graphs when parameterized by p. It is readily seen that this problem is also fixed-parameter tractable for complete$+pe$ graphs and complete$-pv$ graphs when parameterized by p. This leaves us with the remaining case of complete$+pv$ graphs. As vertices in a complete graph must be colored differently, Coloring is fixed-parameter tractable for complete$+pv$ graphs when parameterized by p. What is the parameterized complexity of List Coloring and Precoloring Extension for complete$+pv$ graphs?

References

1. J.A. Bondy and U.S.R. Murty, Graph Theory, Springer Graduate Texts in Mathematics 244 (2008).
2. F. Bonomo, G. Durán and J. Marenco, Exploring the complexity boundary between coloring and list-coloring, Ann. Oper. Res. 169 (2009) 3–16.
3. H.J. Broersma, F.V. Fomin, P.A. Golovach and D. Paulusma, Three complexity results on coloring P_k-free graphs, European Journal of Combinatorics 34 (2013) 609–619.
4. H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song, Updating the complexity status of coloring graphs without a fixed induced linear forest, Theoretical Computer Science 414, 9–19 (2012) .
5. H.J. Broersma, T. Kloks, D. Kratsch and H. Müller, Independent sets in asteroidal triple-free graphs, SIAM Journal on Discrete Mathematics 12 (1999) 276–287.
6. L. Cai, Parameterized complexity of vertex colouring, Discrete Applied Mathematics 127 (2003) 415–429.
7. J.F. Couturier, P.A. Golovach, D. Kratsch and D. Paulusma, List coloring in the absence of a linear forest, Proc. WG 2011, LNCS 6986, 119–130 (2011).
8. P. Erdös, A.L. Rubin and H. Taylor, Choosabilty in graphs, Proc. West Coast Conference on Combinatorics, Graph Theory and Computing, Arcata, California, Congressus Numerantium XXVI (1979) 125–157.
9. P.A. Golovach, D. Paulusma, and J. Song, 4-Coloring H-free graphs when H is small, Proc. SOFSEM 2012, LNCS 7147, 289–300 (2012).
10. C.T. Hoàng, M. Kamiński, V. Lozin, J. Sawada, and X. Shu, Deciding k-colorability of P_5-free graphs in polynomial time, Algorithmica 57, 74–81 (2010).
11. J.E. Hopcroft and R.M. Karp, An $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs, SIAM J. Comput. 2, 225–231 (1973).
12. S. Huang, Improved complexity results on k-coloring P_t-free graphs, Proc. MFCS 2013, LNCS 8087 (2013) 551–558.

13. M. Hujter and Zs. Tuza, Precoloring extension. II. Graph classes related to bipartite graphs, Acta Math. Univ. Comenianae Vol. LXII, 1–11 (1993).
14. B. Jansen and S. Kratsch, Data reduction for graph coloring problems, Proc. FCT 2011, LNCS 6914, 90–101 (2011).
15. K. Jansen, Complexity Results for the Optimum Cost Chromatic Partition Problem, Universität Trier, Mathematik/Informatik, Forschungsbericht 96–41 (1996).
16. K. Jansen and P. Scheffler, Generalized coloring for tree-like graphs, Discrete Appl. Math. 75, 135–155 (1997).
17. D. Král', J. Kratochvíl, Zs. Tuza, and G.J. Woeginger, Complexity of coloring graphs without forbidden induced subgraphs, Proc. WG 2001, LNCS 2204, 254–262 (2001).
18. D. Kratsch and H. Müller, Colouring AT-free graphs, Proc. ESA 2012, LNCS 7501, 707–718 (2012).
19. M. , Coloring edges and vertices of graphs without short or long cycles, Contributions to Discrete Math. 2, 61–66 (2007).
20. M. Kubale, Some results concerning the complexity of restricted colorings of graphs. Discrete Applied Mathematics 36, 35-46 (1992).
21. D. Marx, Paramaterized coloring problems on chordal graphs, Theoretical Computer Science 351, 407-424 (2006).
22. D. Marx, Precoloring extension on unit interval graphs, Discrete Applied Mathematics 154, 995–1002 (2006).
23. B. Randerath and I. Schiermeyer, 3-Colorability $\in$ P for P_6-free graphs, Discrete Appl. Math. 136, 299–313 (2004).
24. B. Randerath and I. Schiermeyer, Vertex colouring and forbidden subgraphs - a survey, Graphs Combin. 20, 1–40 (2004).
25. T. J. Schaefer, The complexity of satisfiability problems, Proc. STOC 1978, 216–226 (1978).
26. J. Stacho. 3-Colouring AT-free graphs in polynomial time, Algorithmica 64, 384–399 (2012).
27. Zs. Tuza, Graph colorings with local restrictions - a survey, Discuss. Math. Graph Theory 17, 161–228 (1997).
28. V.G. Vizing, Coloring the vertices of a graph in prescribed colors, in Diskret. Analiz., no. 29, Metody Diskret. Anal. v. Teorii Kodov i Shem 101, 3–10 (1976).