4-Coloring H-Free Graphs When H Is Small *

Petr A. Golovach, Daniël Paulusma, and Jian Song

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

Abstract. The k-Coloring problem is to test whether a graph can be colored with at most k colors such that no two adjacent vertices receive the same color. If a graph G does not contain a graph H as an induced subgraph, then G is called H-free. For any fixed graph H on at most 6 vertices, it is known that 3-Coloring is polynomial-time solvable on H-free graphs whenever H is a linear forest and NP-complete otherwise. By solving the missing case $P_2 + P_3$, we prove the same result for 4-Coloring provided that H is a fixed graph on at most 5 vertices.

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 k-Coloring problem is to decide whether a graph can be colored with at most k colors. Due to the fact that k-Coloring is NP-complete for any fixed $k \geq 3$, there has been considerable interest in studying its complexity when restricted to certain graph classes. One of the most well-known results in this respect is due to Grötschel, Lovász, and Schrijver [12] who show that k-Coloring is polynomial-time solvable for perfect graphs. More information on this classic result and on the general motivation, background and related work on coloring problems restricted to special graph classes can be found in several surveys [22, 25] on this topic.

We continue the study of the computational complexity of the k-Coloring problem restricted to graph classes defined by one or more forbidden induced subgraphs. This problem has been studied in many papers by different groups of researchers [3–9, 11, 13, 15–18, 20, 21, 26]. Before we summarize these results and explain our new results, we first state the necessary terminology and notations.

1.1 Terminology

We only consider finite undirected graphs G with no loops and no multiple edges. We refer to the textbook by Bondy and Murty [2] for any undefined graph terminology. We write $G[U]$ to denote the subgraph of G *induced by* the

* This work has been supported by EPSRC (EP/G043434/1) and an extended abstract of it has been accepted for SOFSEM 2011.

vertices in U, i.e., the subgraph of G with vertex set U and an edge between two vertices $u, v \in U$ if and only if $uv \in E$.

The graph P_n denotes the *path* on n vertices, i.e., $V(P_n) = \{u_1, \ldots, u_n\}$ and $E(P_n) = \{u_i u_{i+1} \mid 1 \leq i \leq n-1\}$. The graph C_n denotes the *cycle* on n vertices, i.e., $V(C_n) = \{u_1, \ldots, u_n\}$ and $E(C_n) = \{u_i u_{i+1} \mid 1 \leq i \leq n-1\} \cup \{u_n u_1\}$. The graph $K_{1,n}$ denotes the *star* on $n+1$ vertices, i.e., $V(K_{1,n}) = \{u_1, \ldots, u_{n+1}\}$ and $E(K_{1,n}) = \{u_1 u_i \mid 2 \leq i \leq n+1\}$. 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. A *linear forest* is the disjoint union of a collection of paths. Let $\{H_1, \ldots, H_p\}$ be a set of graphs. We say that a graph 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.

A *(vertex) coloring* of a graph $G = (V, E)$ is a mapping $c : V \to \{1, 2, \ldots\}$ such that $c(u) \neq c(v)$ whenever $uv \in E$. Here, $c(u)$ is referred to as the *color* of u. A k*-coloring* of G is a coloring c of G with $c(V) \subseteq \{1, \ldots, k\}$. Here, we used the notation $c(U) = \{c(u) \mid u \in U\}$ for $U \subseteq V$. If G has a k-coloring, then G is called k*-colorable*. Recall that 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. The optimization version of this problem is to determine the *chromatic number* of a graph G, i.e., the smallest k such that G has a k-coloring.

1.2 Related Work

Král', Kratochvíl, Tuza and Woeginger [17] completely determined the computational complexity of Coloring for graph classes characterized by one forbidden induced subgraph. They achieved 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.*

The computational complexity of Coloring for $\mathcal{H}$-free graphs where $\mathcal{H}$ is a family of two graphs is still open, although several partial results are known, e.g., Král' et al. [17] also showed that Coloring is NP-complete for (C_3, H)-free graphs whenever H is a fixed graph containing at least one cycle. This work has been extended by Schindl [23]. Maffray and Preissmann [20] showed that Coloring is NP-complete for $(C_3, K_{1,5})$-free graphs. Broersma et al. [5] showed that Coloring is polynomial-time solvable for $(C_3, 2P_3)$-free graphs, hereby completing a study of Dabrowski et al. [9] who considered the Coloring problem restricted to (C_3, H)-free graphs for graphs H on at most six vertices.

We focus on the computational complexity of the k-Coloring problem for H-free graphs. Kamiński and Lozin [15] 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 has the following immediate consequence.

Theorem 2. *For any $k \geq 3$, the k-Coloring problem is NP-complete for the class of H-free graphs whenever H contains a cycle.*

Holyer [14] showed that 3-Coloring is NP-complete on line graphs. Later, Leven and Galil [19] extended this result by showing that k-Coloring is also NP-complete on line graphs for $k \geq 4$. Because line graphs are claw-free, i.e., they have no induced $K_{1,3}$, these two results together have the following consequence.

Theorem 3. *For any $k \geq 3$, the k-Coloring problem is NP-complete for the class of H-free graphs whenever H is a forest with a vertex of degree at least 3.*

Due to Theorems 2 and 3, only the case in which H is a linear forest remains. We first consider the case when H is a path. Hoàng et al. [13] 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 [21] showed that 3-Coloring can be solved in polynomial time for P_6-free graphs. It is also known that 4-Coloring is NP-complete for P_8-free graphs [4] and that 6-Coloring is NP-complete for P_7-free graphs [3]. Combining these results leads us to Table 1. In this table, "P" means that the combination of k and ℓ is polynomial-time solvable and "NP-c" means that it is NP-complete; all open cases are denoted "?".

	$k \rightarrow$			
P_ℓ-free	3	4	5	≥ 6
$\ell \leq 5$	P	P	P	P
$\ell = 6$	P	?	?	?
$\ell = 7$	?	?	?	NP-c
$\ell \geq 8$	?	NP-c	NP-c	NP-c

Table 1. The complexity of k-Coloring for P_ℓ-free graphs for combinations of fixed k and ℓ.

We now discuss the case when H is a linear forest that is the disjoint union of two or more paths. Combining a result from Balas and Yu [1] on the maximal number of independent sets in an sP_2-free graph and a result from Tsukiyama et al. [24] on the enumeration of such sets leads to the known result that k-Coloring is polynomial-time solvable on sP_2-free graphs for any two integers k and s. Broersma et al. [4] extended the aforementioned result of Randerath and Schiermeyer [21] by showing that 3-Coloring is polynomial-time solvable for H-free graphs if H is a linear forest with $|V_H| \leq 6$ or $H = sP_3$ for any integer s. They also observed that 3-Coloring is polynomial-time solvable for $(P_1 + H)$-free graphs whenever this problem is polynomial-time solvable for H-free graphs. Couturier et al. [7] extended the aforementioned result of Hoàng et al. [13] by proving that for any fixed integers k and r, the k-Coloring problem can be solved in polynomial time for $(rP_1 + P_5)$-free graphs. All these positive results are summarized in Theorems 4 and 5 taking into account that k-Coloring is polynomial-time solvable on H'-free graphs whenever it is so on H-free graphs for some graph H containing H' as an induced subgraph.

Theorem 4. *The* 3-Coloring *problem can be solved in polynomial time for H-free graphs if*

- $H = rP_1 + P_2 + P_4$ *for all* $r \geq 0$
- $H = rP_1 + P_6$ *for all* $r \geq 0$
- $H = sP_3$ *for all* $s \geq 0$.

Theorem 5. *For any $k \geq 4$, the* k-Coloring *problem can be solved in polynomial time for H-free graphs if*

- $H = rP_1 + P_5$ *for all* $r \geq 0$
- $H = sP_2$ *for all* $s \geq 0$.

1.3 Our new result

Theorems 2–4 imply that for any fixed graph H on at most 6 vertices, 3-Coloring is polynomial-time solvable on H-free graphs whenever H is a linear forest and NP-complete otherwise. We prove the following result.

Theorem 6. *For any fixed graph H on at most 5 vertices,* 4-Coloring *is polynomial-time solvable on H-free graphs whenever H is a linear forest and* NP*-complete otherwise.*

Theorems 2, 3 and 5 imply that the only missing case is when $H = P_2 + P_3$. We present a polynomial-time algorithm for this case in Section 3. The correctness proof of this algorithm uses some known structural and algorithmic results stated in Section 2.

2 Preliminaries

In order to proceed we must slightly generalize the coloring concept as follows. 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 $u \in V$, then L is also called a *k-list assignment*. Equivalently, L is a k-list assignment if $|\bigcup_{u \in V} L(u)| \leq k$. We say that a coloring $c : V \to \{1, 2, \ldots\}$ *respects* L if $c(u) \in L(u)$ for all $u \in V$. 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. If $|L(u)| = 1$ for every vertex u of some subset $W \subseteq V$ and $L(u) = \{1, \ldots, k\}$ for all $u \in V \setminus W$, then we obtain the k-Precoloring Extension problem. In that case we also say that we want to *extend* the precoloring of W to a k-coloring of G.

We will frequently use the following observation, the proof of which follows from the fact that the problem in this case can be modeled and solved as an instance of the 2-Satisfiability problem. This approach has been introduced by Edwards [10] and is folklore now.

Lemma 1 ([10]). *Let G be a graph in which every vertex has a list of admissible colors of size at most 2. Then checking whether G has a coloring respecting these lists is solvable in polynomial time.*

Let $G = (V, E)$ be a graph. For a subset $U \subseteq V$ we define $N_G(U) = \{v \in V \setminus U \mid uv \in E \text{ for some } u \in U\}$; note that $N(\emptyset) = \emptyset$. A set $D \subseteq V$ *dominates* a set $S \subseteq V$ if $S \subseteq D \cup N_G(D)$; if $S = V$ then we say that D is a *dominating set* of G.

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

Lemma 2 ([4]). *Let $s \geq 1$ and $G = (V, E)$ be an sP_3-free graph with a set $W \subseteq V$ such that each vertex in W is precolored with a color from $\{1, \ldots, k\}$ and every vertex in $V \setminus W$ has degree at least k for some integer k. If G has a k-coloring extending the precoloring of W, then G contains a set D of size at most $k \cdot\ r(s, k+1) + (k^2+3) \cdot (s-1)$ that dominates $V \setminus W$.*

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

Lemma 3. *Let $G = (V, E)$ be a (P_2+P_3)-free graph of minimum degree at least 4. If G has a 4-coloring, then G contains a dominating set D of size at most 39.*

Note that Lemma 3 involves a minimum degree condition. However, we can easily get around this by applying the following well-known procedure on a graph $G = (V, E)$. Remove all vertices of V with degree at most 3 from G. Propagate this until we obtain a graph G^* of minimum degree at least 4; note that G^* may be the empty graph. We observe the following straightforward result, see e.g. Broersma et al. [4] for a proof.

Lemma 4. *Let G be an H-free graph for some graph H. Then G has a 4-coloring if and only if G^* has a 4-coloring. Moreover, G^* is H-free and can be obtained in polynomial time.*

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

Lemma 5. *The* List 3-Coloring *problem can be solved in polynomial time for the class of $(P_2 + P_3)$-free graphs.*

We also need the following lemma, which follows immediately from Lemma 5.

Lemma 6. *Let $G = (V, E)$ be a $(P_2 + P_3)$-free graph. Then a partition of V into three (possibly empty) independent sets I_1, I_2, I_3 can be found in polynomial time if it exists.*

3 The Algorithm

Let G be a (P_2+P_3)-free graph that is an instance of 4-Coloring. By Lemma 4 we may assume that G has minimum degree at least 4. We also assume that each vertex u has been assigned an initial list $L_0(u) = \{1, 2, 3, 4\}$ of admissible colors.

Outline. Our algorithm is a branching algorithm. The main idea is to obtain in polynomial time a polynomial-bounded set $\mathcal{L}$ of list assignments for G that have the following two properties. First, G has a 4-coloring if and only if G has a coloring that respects at least one list assignment in $\mathcal{L}$. Second, for every list assignment in $\mathcal{L}$, we either have that all its lists have size at most two or else that the union of its lists that contain at least 2 colors has size 3; in the first case we can use Lemma 1, and in the second case we can use Lemma 5 after removing all vertices with a single color in their list from G. Because we obtain $\mathcal{L}$ in polynomial time and its size is bounded by a polynomial, this means that the total running time of our algorithm is polynomial.

Our algorithm consists of two phases. In Phase 1 we first check for a "small" dominating set D. Such a set D must exist in the case that G is 4-colorable, as we prove later. Because D has small size, the total number of 4-colorings of $G[D]$ will be "small" as well. The algorithm considers every 4-coloring of $G[D]$. Given such a 4-coloring of D, it partitions the remaining vertices of G in four different ways using Lemma 6. We use these partitions, together with further structural properties of $(P_2 + P_3)$-free graphs, for a branching procedure. At the end of Phase 1, we either have found that G has no 4-coloring or we have obtained a set $\mathcal{L}$ of list assignments, for which we will prove the desired properties specified in the outline. In Phase 2 we consider the list assignments of $\mathcal{L}$ one by one to determine whether G has a coloring respecting at least one of them.

We now describe Phases 1 and 2 in detail. Here, we use the following terminology. If we say that we "color the vertices of a set U according to their lists", then we mean that we assign every vertex $u \in U$ a color that is in the list of u, and moreover, such that two adjacent vertices in U do not get the same color. Afterwards, for every $u \in U$, we may remove the color of u from the list of every neighbor of u in $N_G(U)$. This is what we call *updating* the list assignment. Also, when coloring a vertex, say with color i, then we set its list of admissible colors to $\{i\}$. After proving a number of lemmas, we show in Theorem 7 that our algorithm is correct and that it runs in polynomial time.

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

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

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

Assume that $G[D]$ is 4-colorable and set $\mathcal{L} = \emptyset$. Perform Steps 3-9 for every 4-coloring c_D of $G[D]$.

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

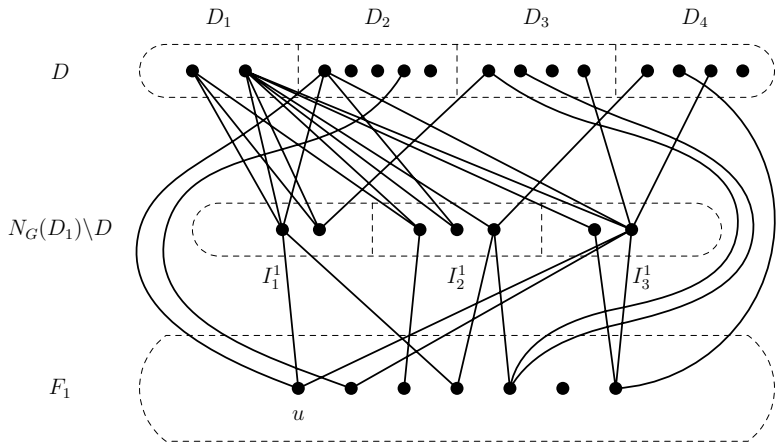

Fig. 1. A graph G decomposed as $V_G = D \cup (N_G(D_1) \setminus D) \cup V_{F_1}$, where edges inside the different parts are not displayed; note that vertex u belongs to $F_1 \cap (N_G(D_2) \setminus D) \cap F_3 \cap F_4$ in this particular example.

$V_{F_h} \cap V_{F_i} \neq \emptyset$ is possible for $h \neq i$. For $i = 1, \ldots, 4$ check whether $N_G(D_i) \setminus D$ can be partitioned into three independent sets, where one or more of such sets are allowed to be empty; in particular, all three sets are empty if $N_G(D_i) \setminus D = \emptyset$. If such a partition does not exist for some i, then stop considering c_D. Otherwise, let I_1^i, I_2^i, I_3^i be such a partition for $i = 1, \ldots, 4$. Figures 1 and 2 illustrate that

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

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

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

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

Step 5. Check if there exists a graph F_i' that has at most 2 vertices. If so, then pick an arbitrary such F_i' and do as follows. Color the vertices of Q_i with color i. Consider every possible coloring of the vertices of F_i' according to their lists. Each time, update the list assignment and put the resulting list assignment in $\mathcal{L}$. Stop considering c_D.

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

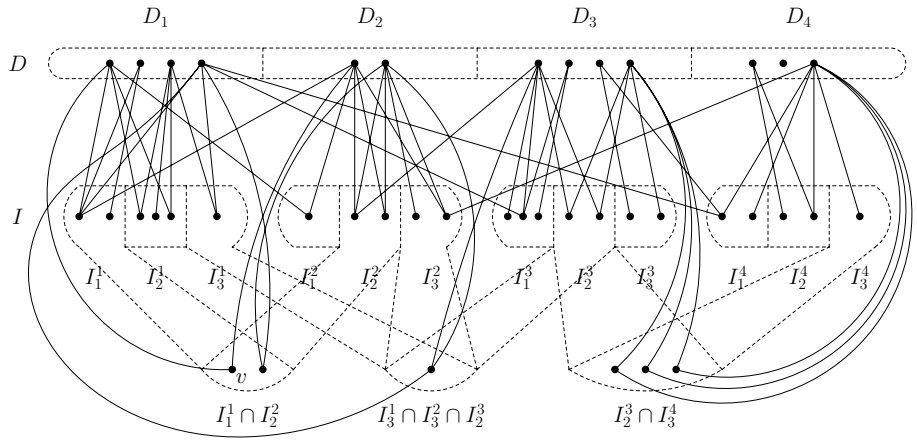

Fig. 2. A graph G decomposed as $V_G = D \cup \bigcup_{i,j} I_j^i$, where edges inside the different parts are not displayed; note that $I_1^1 \cap I_2^2 \neq \emptyset$, $I_3^1 \cap I_3^2 \cap I_2^3 \neq \emptyset$ and $I_2^3 \cap I_3^4 \neq \emptyset$, whereas all other sets I_j^i do not intersect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

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

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

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

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

7a. If F_i' is connected and bipartite, then do as follows. Give all the vertices of one partition class color i. Consider both possibilities. In both cases, color the vertices of Q_i with color i, update the list assignment, put the resulting list assignment in $\mathcal{L}$ and restore all lists to the situation at the end of Step 6.

7b. If F_i' is disconnected and bipartite, then do as follows for every j with $\tilde{I}_j^i \neq \emptyset$. Consider every edge in F_i' that has no end-vertex with list $\{i\}$ already. If both end-vertices are adjacent to all but at most three vertices of $\tilde{I}_j^i$, then arbitrarily pick one of these end-vertices and color it with i. If exactly one end-vertex is adjacent to all but at most three vertices of $\tilde{I}_j^i$, then color that end-vertex with color i. Afterwards, let S_j^i be the set of edges in F_i', both end-vertices

of which are not colored i. Consider every possible coloring of the end-vertices of the edges in S_j^i according to their lists. Each time, color the vertices of Q_i with color i, update the list assignment, put the resulting list assignment in $\mathcal{L}$, and restore the lists to the situation at the end of Step 6.

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

After considering all $1 \leq i \leq 4$, let M be the set that consists of the following vertices: the vertices u^i, v^i for every connected F_i' and the vertices u_1^i, u_2^i, u_3^i for every disconnected F_i'; note that $|M| \leq 12$. Check whether there exists a coloring of $G[M]$ that respects the lists of the vertices in M, and moreover, that neither colors u^i nor v^i with color i for each connected F_i', and that colors none of u_1^i, u_2^i, u_3^i with color i for each disconnected F_i'. If so, then call such a coloring *suitable* and M a *suitable branch set*, and continue as described below.

For each connected F_i', let $\tilde{I}^i(\bar{e}^i)$ be the set of vertices in $(\tilde{I}_1^i \cup \tilde{I}_2^i \cup \tilde{I}_3^i) \setminus M$ that are adjacent neither to u^i nor to v^i. For each disconnected F_i', let $\tilde{I}_j^i(\bar{u}_j^i)$ be the set of vertices in $\tilde{I}_j^i \setminus M$ that are not adjacent to u_j^i. Color M with a suitable coloring. For $i = 1, \ldots, 4$, if F_i' is connected, then color all vertices in $\tilde{I}^i(\bar{e}^i)$ according to their lists, and if F_i' is disconnected, then color all vertices in every $\tilde{I}_j^i(\bar{u}_j^i)$ according to their lists. Afterwards, color all remaining uncolored vertices in I^* according to their lists. Only then update the resulting list assignment, put it in $\mathcal{L}$ and restore all lists to the situation at the end of Step 7.

Repeat the above procedure until all suitable branch sets, all their suitable colorings, all colorings of the vertices in the sets $\tilde{I}^i(\bar{e}^i)$, all colorings of the vertices in the sets $\tilde{I}_j^i(\bar{u}_j^i)$ and all colorings of any remaining uncolored vertices in I^* have been considered.

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

Do as follows for every $L \in \mathcal{L}$. Determine the set U_L of vertices of G that have a list of size 1. Color every vertex in U_L with the (unique) color from its list. If there are two adjacent vertices in U_L colored alike, then stop considering L. If such vertices do not exist, then update L and remove U_L from G. Denote the resulting graph and list assignment by G' and L', respectively. If all lists in L' have size at most 2, then apply Lemma 1. If the union of all lists in L' has size 3, then apply Lemma 5. If this leads to a coloring of G' respecting L', then return `Yes`. If after considering all $L \in \mathcal{L}$ no `Yes`-answer has been returned, then return `No`.

We prove the correctness of our algorithm and analyze its running time in Theorem 7. For doing this, we need the following lemmas.

Lemma 7. *For $i = 1, \ldots, 4$ and $j = 1, \ldots, 3$, the number of vertices of I_j^i that is not adjacent to a_j^i in Step 6 of Phase 1 is at most 38.*

Proof. In order to obtain a contradiction, suppose that there exists a pair of indices (i, j) such that $a_0 = a_j^i$ is not adjacent to 39 vertices $b_1, \ldots, b_{39}$ in I_j^i. Consider a vertex b_h for some $1 \leq h \leq 39$. Because b_h is in $N_G(D_i)$, it has a neighbor $a_h \in D_i$; note that $a_h \neq a_0$ by definition. Suppose that a_h is not adjacent to two vertices c and c' of I_j^i that are neighbors of a_0. Then G contains an induced $P_2 + P_3$, where $P_2 = a_h b_h$ and $P_3 = ca_0c'$. This is not possible. By our choice of a_0, we find that a_h cannot be adjacent to all neighbors of a_0 in I_j^i; otherwise a_h has more neighbors in I_j^i than a_0. We conclude that a_h is adjacent to all but one neighbor of a_0 in I_j^i. By our choice of a_0, this implies that a_h cannot be adjacent to a vertex $b_{h'}$ with $h' \neq h$. Hence, we found that D_i contains vertices $a_1, \ldots, a_{39}$ (where each a_i is adjacent to b_i and to all but one neighbor of a_0 in I_j^i). However, then $|D| \geq |D_i| \geq 40$, which is not possible as D has size at most 39 according to Step 1 of Phase 1. This completes the proof of Lemma 7. □

Lemma 8. *For each edge uv in each F_i', there exists at most one vertex in each $\tilde{I}_j^i$ that is adjacent neither to u nor to v.*

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

Lemma 9. *For all $1 \leq i \leq 4$ and all $1 \leq j \leq 3$, if F_i' is a disjoint union of at least two edges, then all but at most one edge of F_i' have at least one end-vertex that is adjacent to all but at most three vertices of $\tilde{I}_j^i$.*

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

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

By the same argument as above, we find that s is adjacent to all but at most one neighbor of v in $\tilde{I}_j^i$, or else that v is adjacent to all but at most one neighbor of s in $\tilde{I}_j^i$. Lemma 8 tells us that $\{u, v\}$ dominates all but at most one vertex of $\tilde{I}_j^i$. Consequently, in the first case, s is adjacent to all but at most three vertices of $\tilde{I}_j^i$, and in the second case v is adjacent to all but at most three vertices of $\tilde{I}_j^i$. Hence, in both cases we find a vertex of $\{s, t, u, v\}$ that is adjacent to all

but at most three vertices of $\tilde{I}^i_j$. This is in contradiction with our assumption on $\{s,t,u,v\}$. Hence, we have proven Lemma 9. □

Lemma 10. *In Step 7b of Phase 1, each S^i_j contains at most one edge.*

Proof. In Step 7b of Phase 1, a graph F'_i is disconnected and bipartite and has at least three vertices. Then, because G is (P_2+P_3)-free, F'_i is a disjoint union of at least two edges. Then Lemma 9 tells us that all but at most one edge of F'_i have at least one end-vertex adjacent to all but at most three vertices of $\tilde{I}^i_j$. Hence, a set S^i_j contains at most one edge. □

Lemma 11. *In Step 8 of Phase 1, each $\tilde{I}^i(\bar{e}^i)$ contains at most one vertex, and each $\tilde{I}^i_j(\bar{u}^i_j)$ contains at most three vertices.*

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

Lemma 12. *Let L be a list assignment in the set $\mathcal{L}$ in Phase 2. Then either all lists in L have size at most 2, or the union of the lists in L that contain at least two colors has size 3.*

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

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

$$V_G = D \cup (N_G(D_i) \setminus D) \cup Q_i \cup V_{F'_i}.$$

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

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

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

Let $w \in V_G \setminus D$. If $w \in N_G(D_i) \setminus D$, then $i \notin L(w)$. If $w \in Q_i$, then $L(w) = \{i\}$ due to Step 7a, hence $|L(w)| = 1$. If $w \in B_i^1$, then $L(w) = \{i\}$ due to Step 7a, hence $|L(w)| = 1$. If $w \in B_i^2$, then $i \notin L(w)$, because the algorithm updates the list assignment in Step 7a after coloring each vertex of B_i^1 with color i, and each vertex of B_i^2 is adjacent to at least one vertex of B_i^1 as F_i' is connected. Hence, the union of the lists that contain at least 2 colors does not contain color i, and consequently, has size at most 3.

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

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

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

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

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

Let $w \in V_G \setminus D$. If $w \in I^*$, then $|L(w)| = 1$ due to Step 8. If $w \in \tilde{I}^i(\bar{e}^i) \cup \{u^i, v^i\}$ for some edge $e^i = u^i v^i$ with $u^i, v^i \in M$, then $|L(w)| = 1$ due to Step 8. If $w \in \tilde{I}_j^i(\bar{u}_j^i) \cup \{u_j^i\}$ for some vertex $u_j^i \in M$, then $|L(w)| = 1$ due to Step 8 as well. In all other cases, w belongs to a set $\tilde{I}_j^i$ for at least one $1 \leq i \leq 4$ and some $1 \leq j \leq 3$. If F_i' is connected or non-bipartite, then w is adjacent to a vertex in M, which is an end-vertex of some chosen edge $e^i = u^i v^i$. If F_i' is disconnected and bipartite, then w is adjacent to a vertex in M, which is some chosen vertex u_j^i. In both cases, this neighbor of w is colored with a color not equal to i, because the coloring of M is assumed to be suitable. Because $i \notin L(w)$ by definition, this means that $|L(w)| \leq 2$. Hence, all lists of L have size at most 2. This completes the proof of Lemma 12. □

Lemma 13. *If $\mathcal{L}$ contains a list assignment that is respected by some coloring of G, then the algorithm returns* `Yes`.

Proof. Let $L \in \mathcal{L}$ be a list assignment that is respected by some coloring c of G. Because L is respected by c, coloring every vertex in U_L with the (unique) color

from its list does not result in two adjacent vertices with the same color; each $u \in U_L$ receives color $c(u)$.

We remove all vertices in U_L from G and denote the resulting graph and list assignment by G' and L', respectively. Let c' be the restriction of c to $V_{G'}$. Then L' is respected by c'.

Lemma 12 tells us that either all lists in L have size at most 2, or the union of the lists in L that contain at least two colors has size 3. Consequently, either all lists in L' have size at most 2, or the union of the lists in L' has size 3. In the first case the algorithm applies Lemma 1. In the second case the algorithm applies Lemma 5. In both cases the algorithm will conclude that G' has a coloring that respects L' (because c' is such a coloring). Hence, it will return `Yes`. This completes the proof of Lemma 13. □

Theorem 7. *The* 4-Coloring *problem can be solved in polynomial time for* $(P_2 + P_3)$*-free graphs.*

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

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

First suppose that the output of our algorithm is `Yes`. Note that such an output only occurs in Phase 2. Hence, a graph G' has a coloring respecting a list assignment L', where G' and L' are obtained by removing all vertices from G that have a list of size 1, i.e., belong to a set U_L for some $L \in \mathcal{L}$. Coloring the vertices in U_L with the (unique) color from their list does not yield two adjacent vertices colored alike, as otherwise the algorithm would have stopped considering L and thus would not have modified L into L'. Because of this and because the algorithm updates L before removing U_L, we can extend the coloring of G' to a coloring of G by assigning every vertex that is not in G', i.e., that belongs to U_L, the unique color in its list. Because every list in every list assignment of $\mathcal{L}$ is a subset of $\{1, 2, 3, 4\}$, the resulting coloring is a 4-coloring of G.

Now suppose that G has a 4-coloring c. Lemma 3 tells us that G has a dominating set of size at most 39. Consequently, our algorithm will find such a dominating set in Step 1. Because G is 4-colorable, $G[D]$ is 4-colorable. Hence, the algorithm does not return `No` in Step 2. Instead it considers each 4-coloring of $G[D]$ including the 4-coloring c_D of $G[D]$ with $c_D(a) = c(a)$ for all $a \in D$.

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

proof we only need the algorithm to find *some* partition I_1^i, I_2^i, I_3^i of $N_G(D_i) \setminus D$ for $i = 1, \ldots, 4$, and the fact that the restriction of c to $N_G(D_i) \setminus D$ is a 3-coloring ensures that this is going to happen.

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

In Step 5, the algorithm checks whether there exists a graph F_i' that has at most 2 vertices for some $1 \leq i \leq 4$. If so, then the algorithm considers each possible coloring of these vertices, so including the coloring of F_i' that corresponds to c. In addition, it colors each vertex of Q_i with color i. We may assume without loss of generality that $c(u) = i$ for all $u \in Q_i$. If $c(u) \neq i$ for some $u \in Q_i$, then we may redefine c by setting $c(u) := i$ for the following reason. All neighbors of u in G belong to $N_G(D_i) \setminus D$, i.e., are adjacent to a vertex in D_i, which c has assigned color i, and as such, no neighbor of u is assigned color i by c. Hence, the algorithm goes to Phase 2 with a set $\mathcal{L}$ of list assignments that include a list assignment L that is respected by c. Then it will return `Yes` due to Lemma 13.

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

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

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

The first case is in Step 7a, when F_i' is connected and bipartite for some $1 \leq i \leq 4$, such that $c(u) = i$ for every vertex u in one partition class of F_i'. Because the algorithm considers both partition classes of F_i', one of the two created list assignments that are to be placed in $\mathcal{L}$ is respected by c. Consequently, the algorithm will return `Yes` due to Lemma 13.

The second case *may* be in Step 7b. We first note that in this step the algorithm colors a vertex of all but at most one edge with color i due to Lemma 10. Hence, if F_i' is a disconnected and bipartite graph, in which all but at most one edge contain a vertex colored i by c, and moreover, such that the algorithm picks exactly those vertices u with $c(u) = i$ to get color i for some $1 \leq j \leq 3$ with $\tilde{I}_j^i \neq \emptyset$, then the resulting list assignment is respected by c. In that case, the algorithm will return `Yes` due to Lemma 13. We emphasize that in this step the algorithm considers at most three possible assignments of color i to vertices in F_i', namely one assignment for each nonempty $\tilde{I}_j^i$. If in each case the algorithm assigns color i to one or more different vertices than the ones that are colored i by c, then the resulting list assignment that is placed in $\mathcal{L}$ will not be respected by c. We take this into account when analyzing Step 8.

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

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

Case 1. Suppose that F_i' is connected or non-bipartite.
If c colors an end-vertex of every edge in F_i' with color i, then F_i' must be a bipartite graph; the set of vertices colored i and the set of vertices not colored i form the two partition classes. In that case, the algorithm has already placed, namely in Step 7a or 7b, a list assignment in $\mathcal{L}$ that is respected by c. This is in contradiction with our assumption that the algorithm has not yet done this. Hence, F_i' contains at least one edge $e = uv$ with $c(u) \neq i$ and $c(v) \neq i$.

Case 2. Suppose that F_i' is disconnected and bipartite.
Then, because G is $(P_2 + P_3)$-free and F_i' contains no isolated vertices, F_i' is a disjoint union of edges. Because F_i' has at least three vertices, this means that F_i' has at least two edges. The algorithm considers the sets $\tilde{I}_j^i$ for $j = 1, \ldots, 3$ in sequential order. Let $1 \leq j \leq 3$ and consider a set $\tilde{I}_j^i$. Let Z_i be the set of vertices in F_i' that are adjacent to all but at most three vertices of $\tilde{I}_j^i$. By Lemma 9 we find that $Z_i \neq \emptyset$, because all but at most one edge in F_i' contains a vertex adjacent to all but at most three vertices of $\tilde{I}_j^i$, and F_i' has at least two edges. Suppose that every vertex in Z_i is colored i by c. Then every one of those edges that contains a vertex adjacent to all but at most three vertices of $\tilde{I}_j^i$ contains exactly one vertex of Z_i, because two vertices that are both colored with color i cannot be adjacent. However, in that case, the algorithm would already have placed a list assignment in $\mathcal{L}$ that is respected by c, namely in Step 7b. Hence, Z_i contains a vertex u with $c(u) \neq i$.

By our case analysis we find that there exists a suitable branch set M, such that the restriction of c to M is a suitable coloring of $G[M]$. Our algorithm will detect this in one of the branches in Step 8. At some point it will also color the vertices in each $\tilde{I}^i(\bar{e}^i)$, the vertices in each $\tilde{I}_j^i(\bar{u}_j^i)$ and all remaining uncolored vertices in I^* according to c, because it considers all possibilities exhaustively. We conclude that after Step 8 has finished, the algorithm has put a list assignment L in $\mathcal{L}$ that is respected by c. Then, by Lemma 13, it will return `Yes`. This completes our correctness proof.

Running time analysis. We prove that Phase 1 can be performed in polynomial time and leads to a set $\mathcal{L}$ of polynomial size.

The algorithm performs Step 1 in $O(n^{39})$ time by brute force. In Step 2 we find at most $4^{|D|} \leq 4^{39}$ different 4-colorings of $G[D]$. The algorithm performs Step 3 in polynomial time by applying Lemma 6 at most four times. It performs Step 4 in linear time, because it only has to detect the isolated vertices in each F_i. Afterwards, it has immediately obtained the graphs F_i'.

The algorithm performs Step 5 in linear time; in addition to considering at most one coloring of some set Q_i of isolated vertices, it needs to consider at most

3^2 colorings of a set of at most 2 vertices that can be detected in linear time; note that each vertex of such a set has indeed a list of size at most 3, because it is adjacent to an already colored vertex in D and the algorithm updated the list assignment in Step 3. If the algorithm starts Phase 2 directly after Step 5, then we have a set $\mathcal{L}$ of size at most $4^{39} \cdot 3^2$, which is a constant. Otherwise, we must continue our running time analysis with Step 6.

For $i = 1, \ldots, 4$ and $j = 1, \ldots, 3$, the algorithm determines a required vertex $a_j^i \in D_i$ in Step 6 in polynomial time. By Lemma 7, each a_j^i is adjacent to all but at most 38 vertices of I_j^i, hence the set I^* has at most $4 \cdot 3 \cdot 38 = 456$ vertices.

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

The algorithm performs Step 7b in polynomial time. This can be seen as follows. The algorithm checks in polynomial time whether a graph F_i' is disconnected and bipartite, i.e., whether F_i' is a disjoint union of at least two edges. If so, then it places at most $4 \cdot 3 \cdot 3^2 = 108$ different list assignments in $\mathcal{L}$, because each set S_j^i contains at most one edge according to Lemma 10, and the vertices in Q_i are colored in only one way for each $1 \leq j \leq 3$ with nonempty $\tilde{I}_j^i$.

In Step 8 the algorithm considers in worst case all edges e^i in every F_i' that is connected or non-bipartite, all colorings of their end-vertices u^i and v^i, all colorings of the vertices in $\tilde{I}^i(\bar{e}^i)$, all possible triples of vertices u_1^i, u_2^i, u_3^i in every F_i' that is a disjoint union of at least two edges, all the colorings of these triples, all colorings of the vertices in $\tilde{I}_1^i(\bar{u}_1^i) \cup \tilde{I}_2^i(\bar{u}_2^i) \cup \tilde{I}_3^i(\bar{u}_3^i)$ and all colorings of the remaining uncolored vertices in I^*. The number of edges e^i is at most n^2. The number of colorings to be considered for the two end-vertices of an edge e^i is at most 3^2. The number of colorings to be considered for the vertices in a set $\tilde{I}^i(\bar{e}^i)$ is at most 3, because $\tilde{I}^i(\bar{e}^i)$ has size at most 1 due to Lemma 11. The number of triples of vertices u_1^i, u_2^i, u_3^i is at most n^3. The number of colorings to be considered for each such triple is at most 3^3. The number of colorings to be considered for the vertices in $\tilde{I}_1^i(\bar{u}_1^i) \cup \tilde{I}_2^i(\bar{u}_2^i) \cup \tilde{I}_3^i(\bar{u}_3^i)$ is at most 9^3, because each $\tilde{I}_j^i(\bar{u}_j^i)$ has size at most 3 due to Lemma 11. Recall that I^* has at most 152 vertices. Hence, the set of remaining uncolored vertices of I^* in some branch has at most 3^{152} colorings. This means that the total number of list assignments obtained in Step 9 is at most $p(n) = n^2 \cdot 3^2 \cdot 3 \cdot n^3 \cdot 3^3 \cdot 9^3 \cdot 3^{152}$. Hence, if we start Phase 2 after Step 9, then we have a set $\mathcal{L}$ of size at most $4^{39}(8+108+p(n))$, which is polynomial. We also conclude that Phase 1 can be performed in polynomial time.

In Phase 2, the algorithm preprocesses each $L \in \mathcal{L}$ in quadratic time; first it colors every vertex in U_L with the unique color in its list, then it checks whether there exist two vertices in U_L that are colored alike, and if not, it updates L and then removes all vertices in U_L from G. Afterwards, Lemma 12 implies that the algorithm can apply (in polynomial time) Lemma 1 or else Lemma 5 for

every resulting graph G' and list assignment L'. Because $\mathcal{L}$ has polynomial size as we deduced above, this means that our algorithm can perform Phase 2 in polynomial time. This completes the proof of Theorem 7. □

We can generalize Theorem 7 in the following way.

Theorem 8. *The* 4-Precoloring Extension *problem can be solved in polynomial time for* $(P_2 + P_3)$*-free graphs.*

Proof. Let $G = (V, E)$ be a $(P_2 + P_3)$-free graph with a set $W \subseteq V$ such that each vertex in W is precolored with a color from $\{1, 2, 3, 4\}$. We may assume without loss of generality that every vertex in $V \setminus W$ has degree at least 4. This can be seen as follows. We consecutively remove vertices of $V \setminus W$ with degree at most 3 from G until this is no longer possible. We denote the remaining graph, which we obtain in polynomial time, by G^*. Because we only removed vertices, G^* is $(P_2 + P_3)$-free. Then G has a 4-coloring extending the precoloring of W if and only if G^* has a 4-coloring extending the precoloring of W (cf. [4]). Because G^* is also $2P_3$-free, we may apply Lemma 2 for $k = 4$ and $s = 2$ to find a set D of at most 39 vertices that dominates $V \setminus W$ in the case that G^* has a 4-coloring extending the precoloring of W. We put all vertices of W in D and run the remainder of the algorithm of Theorem 7 for graph $G[V_G^* \cup W]$ under the additional condition that we let the algorithm only consider 4-colorings of D that do not change the colors of the vertices of W as prescribed by the given precoloring of W. As such, the number of different 4-colorings of D considered by the algorithm is still at most 4^{39}. Hence, the correctness proof and running time analysis are exactly the same as in the proof of Theorem 7. □

Couturier et al. [7] prove that the List k-Coloring problem, and consequently, the k-Precoloring Extension problem is polynomial-time solvable for $(rP_1 + P_5)$-free graphs for any fixed $r \geq 0$. The NP-completeness results in Theorems 2 and 3 carry over to the k-Precoloring Extension problem. These three results together with Theorem 8 imply the following theorem.

Theorem 9. *For any fixed graph* H *on at most* 5 *vertices,* 4-Precoloring Extension *is polynomial-time solvable on* H*-free graphs whenever* H *is a linear forest and* NP*-complete otherwise.*

4 Future work

We note that the running time of our algorithm involves large constants such as 4^{39}. As such it cannot be used to run in practice. We have not tried to minimize the running time, as the motivation for this research is purely theoretical, i.e., to classify the complexity of k-Coloring for H-free graphs further. Hence the following open questions, together with the missing cases of Table 1, are more interesting to us:

1. Is List 4-Coloring polynomial-time solvable for $(P_2 + P_3)$-free graphs?

2. Is 4-Coloring polynomial-time solvable for $(P_1 + P_2 + P_3)$-free graphs?
3. Is 4-Coloring polynomial-time solvable for $2P_3$-free graphs?
4. Is 5-Coloring polynomial-time solvable for $(P_2 + P_3)$-free graphs?

An affirmative answer to Question 1 would be useful for solving the other questions. So far, we only know that the List 5-Coloring problem is NP-complete for P_6-free graphs [3] and for $(P_2 + P_4)$-free graphs [7].

For answering Question 1, it seems difficult to modify the proofs of Theorems 7 and 8. For the variant of list colorings, it is still possible to find a set D that dominates all vertices with list $\{1, 2, 3, 4\}$. The main obstacle is that we do not know how to deal with the vertices that have an initial list of three admissible colors. Also the approach of Hoàng [13] for the k-Coloring problem for P_5-free graphs, which was further generalized by Couturier et al. [7], does not seem applicable here.

Acknowledgments. We thank an anonymous referee for useful comments that helped us to improve the readability of our paper.

References

1. E. Balas and C. S. Yu, On graphs with polynomially solvable maximum-weight clique problem, Networks 19, 247–253 (1989).
2. J.A. Bondy and U.S.R. Murty, Graph Theory, Springer Graduate Texts in Mathematics 244 (2008).
3. H.J. Broersma, F.V. Fomin, P.A. Golovach and D. Paulusma, Three complexity results on coloring P_k-free graphs, Proceedings of IWOCA 2009, LNCS 5874, 95–104 (2009).
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, P.A. Golovach, D. Paulusma and J. Song, Determining the chromatic number of triangle-free $2P_3$-free graphs in polynomial time, Theoretical Computer Science 423, 1–10 (2012).
6. D. Bruce, C.T. Hoàng and J. Sawada, A certifying algorithm for 3-colorability of P_5-free graphs, Proceedings of ISAAC 2009, LNCS 5878, 594-604 (2009).
7. J.F. Couturier, P.A. Golovach, D. Kratsch and D. Paulusma, List coloring in the absence of a linear forest, Proceedings of WG 2011, LNCS 6986, 119–130 (2012).
8. J.F. Couturier, P.A. Golovach, D. Kratsch and D. Paulusma, On the parameterized complexity of coloring graphs in the absence of linear forest, J. Discrete Algorithms 15, 56–62(2012).
9. K. Dabrowski, V. Lozin, R. Raman and B. Ries, Colouring vertices of triangle-free graphs without forests, Discrete Mathematics 312, 1372–1385 (2012).
10. K. Edwards, The complexity of coloring problems on dense graphs. Theoret. Comput. Sci. 43, 337–343 (1986).
11. P.A. Golovach, D. Paulusma and J. Song, Coloring graphs without short cycles and long induced paths, Proceedings of FCT 2011, LNCS 6914, 193–204.
12. M. Grötschel, L. Lovász and A. Schrijver, Polynomial algorithms for perfect graphs, Ann. Discrete Math., Topics on Perfect Graphs 21, 325–356 (1984).

13. 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).
14. I. Holyer, The NP-completeness of edge-coloring, SIAM J. Comput. 10, 718–720 (1981).
15. M. Kamiński and V.V. Lozin, Coloring edges and vertices of graphs without short or long cycles, Contributions to Discrete Math. 2, 61–66 (2007).
16. M. Kamiński and V.V. Lozin, Vertex 3-colorability of Claw-free Graphs. Algorithmic Operations Research 21, (2007).
17. D. Král', J. Kratochvíl, Zs. Tuza and G.J. Woeginger, Complexity of coloring graphs without forbidden induced subgraphs, Proceedings of WG 2001, LNCS 2204, 254–262 (2001).
18. V.B. Le, B. Randerath and I. Schiermeyer, On the complexity of 4-coloring graphs without long induced paths, Theoret. Comput. Sci. 389, 330–335 (2007).
19. D. Leven and Z. Galil, NP completeness of finding the chromatic index of regular graphs, Journal of Algorithms 4, 35–44 (1983).
20. F. Maffray and M. Preissmann, On the NP-completeness of the k-colorability problem for triangle-free graphs, Discrete Math. 162, 313–317 (1996).
21. B. Randerath and I. Schiermeyer, 3-Colorability $\in$ P for P_6-free graphs, Discrete Appl. Math. 136, 299–313 (2004).
22. B. Randerath and I. Schiermeyer, Vertex colouring and forbidden subgraphs - a survey, Graphs Combin. 20, 1–40 (2004).
23. D. Schindl, Some new hereditary classes where graph coloring remains NP-hard, Discrete Math. 295, 197–202 (2005).
24. S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, A new algorithm for generating all the maximal independent sets, SIAM J. Comput. 6, 505–517 (1977).
25. Zs. Tuza, Graph colorings with local restrictions - a survey, Discuss. Math. Graph Theory 17, 161–228 (1997).
26. G.J. Woeginger and J. Sgall, The complexity of coloring graphs without long induced paths, Acta Cybernet. 15, 107–117 (2001).