List coloring in the absence of a linear forest

Jean-François Couturier[1], Petr A. Golovach[2],
Dieter Kratsch[1], and Daniël Paulusma[2] *

[1]Laboratoire d'Informatique Théorique et Appliquée,
Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France
{kratsch,couturier}@univ-metz.fr
[2]School of Engineering and Computing Sciences, Durham University,
Science Laboratories, South Road, Durham DH1 3LE, United Kingdom
{petr.golovach,daniel.paulusma}@durham.ac.uk

Abstract. The k-COLORING problem is to decide whether a graph can be colored with at most k colors such that no two adjacent vertices receive the same color. The LIST k-COLORING problem requires in addition that every vertex u must receive a color from some given set $L(u) \subseteq \{1, \ldots, k\}$. Let P_n denote the path on n vertices, and $G + H$ and rH the disjoint union of two graphs G and H and r copies of H, respectively. For any two fixed integers k and r, we show that LIST k-COLORING can be solved in polynomial time for graphs with no induced $rP_1 + P_5$, hereby extending the result of Hoàng, Kamiński, Lozin, Sawada and Shu for graphs with no induced P_5. Our result is tight; we prove that for any graph H that is a supergraph of $P_1 + P_5$ with at least 5 edges, already LIST 5-COLORING is NP-complete for graphs with no induced H. We then initiate a parameterized complexity study. We show that LIST k-COLORING is fixed parameter tractable in $k+r$ on graphs with no induced rP_1+P_2, and that k-COLORING restricted to such graphs allows a polynomial kernel when parameterized by k. We also show that LIST k-COLORING is fixed parameter tractable in k for graphs with no induced $P_1 + P_3$.

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 the problem 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. Without doubt one of the most well-known results in this respect is due to Grötschel, Lovász, and Schrijver [11] who show that k-COLORING is polynomially 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 [23, 25] on this topic.

We continue the study of the computational complexity of the k-COLORING problem and related problems, in particular LIST k-COLORING when restricted to graph

* This work has been supported by EPSRC (EP/G043434/1) and ANR Blanc AGAPE (ANR-09-BLAN-0159-03).

classes defined by one or more forbidden induced subgraphs. Such problems have been studied in many papers by different groups of researchers [3–7, 12, 15–19, 22, 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 $G = (V, E)$ without loops and multiple edges. We sometimes denote the vertex set of G by V_G. The subgraph of $G = (V, E)$ induced by $U \subseteq V$ is denoted by $G[U]$. We refer to the textbook by Bondy and Murty [2] for any undefined graph terminology.

The graph P_n denotes the path on n vertices. 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 $\phi : V \to \{1, 2, \ldots\}$ such that $\phi(u) \neq \phi(v)$ whenever $uv \in E$. Here, $\phi(u)$ is referred to as the *color* of u. An k*-coloring* of G is a coloring ϕ of G with $\phi(V) \subseteq \{1, \ldots, k\}$. Here, we used the notation $\phi(U) = \{\phi(u) \mid u \in U\}$ for $U \subseteq V$. If G has an k-coloring, then G is called k*-colorable*. For a graph G we let $\chi(G)$ denote its *chromatic number*, i.e., the smallest k such that G has an k-coloring. We recall that the problem k-COLORING is to decide whether a given graph admits an k-coloring. Here, k is *fixed*, i.e., not part of the input. The related VERTEX COLORING problem is the problem of determining the chromatic number of a given graph.

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 $\phi : V \to \{1, 2, \ldots\}$ *respects* L if $\phi(u) \in L(u)$ for all $u \in V$. The LIST k-COLORING problem has as input a graph G with a k-list-assignment L and asks if 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 $u \in V \setminus W$ then we obtain the k-PRECOLORING EXTENSION problem.

For background on parameterized complexity we refer to Downey and Fellows [8] and Niedermeier [21]. Some fundamental notions can be found in Appendix A.

1.2 Related work

Král', Kratochvíl, Tuza and Woeginger [17] completely determined the computational complexity of VERTEX COLORING for graph classes characterized by a forbidden induced subgraph and 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* VERTEX COLORING *can be solved in polynomial time for H-free graphs; otherwise it is* NP*-hard for H-free graphs.*

Theorem 1 can be extended in various ways. One way of doing this is to consider the computational complexity of VERTEX COLORING for $\mathcal{H}$-free graphs where $\mathcal{H}$ is a family of two (or more) graphs. Some initial results have been obtained by Král' et al. [17], Schindl [24] and a number of authors studying the $\mathcal{H}$-free graphs, in which one of the two graphs in $\mathcal{H}$ is the triangle [5, 7, 15, 20].

Another way is to apply parameterized complexity to establish a more subtle classification of those problems being NP-complete. Theorem 1 can also be extended to classify the computational complexity of k-COLORING and other variants of coloring for H-free graphs where k is a fixed integer and H is a fixed graph. The complexity classifications in all three directions is far from being finished. In this paper we focus on the second and third direction.

Our focus on the case when H is a linear forest can be justified as follows. Kamiński and Lozin [15] showed that 3-COLORING is NP-complete for the class of graphs of girth (the length of a shortest induced cycle) at least p for any fixed $p \geq 3$, and Holyer [13] showed that 3-COLORING is NP-complete for claw-free graphs (graphs with no induced $K_{1,3}$). The first result implies that 3-COLORING is NP-complete for the class of H-free graphs if H contains a cycle. The second result implies that 3-COLORING is NP-complete for the class of H-free graphs if H is a forest that contains a vertex with degree at least 3. Hence, only the case in which H is a linear forest remains.

It is 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]. On the contrary, Randerath and Schiermeyer [22] showed that 3-COLORING can be solved in polynomial time for P_6-free graphs. A result which was generalized by Broersma et al. [3] who showed that 3-PRECOLORING EXTENSION can be solved in polynomial time for P_6-free graphs. Broersma et al. [4] extended this result by showing that 3-PRECOLORING EXTENSION, or as a matter of fact LIST 3-COLORING, can be solved in polynomial time for H-free graphs if H is a linear forest on at most 6 vertices. Their proof methods can directly be applied to show exactly the same result for LIST 3- COLORING. For P_5-free graphs, Hoàng et al. [12] could show a stronger result; note that VERTEX COLORING is NP-hard for P_5-free graphs due to Theorem 1.

Theorem 2 ([12]). *For any fixed integer k, the* LIST k-COLORING *problem can be solved in polynomial time for P_5-free graphs.*

1.3 Our new results

The first aim of our paper is to generalize Theorem 2 as much as possible. We prove that for any fixed integers k and r, the LIST k-COLORING problem is polynomial-time solvable for $(rP_1 + P_5)$-free graphs. In order to prove our result, we show that our input graphs allow a dominating set of small size. We precolor such a set in every possible way. Afterwards, we use the technique of "separating the color lists of independent sets" of Hoáng et al. [12] on each resulting instance. They successfully applied this technique for coloring P_5-free graphs. Our result for $(rP_1 + P_5)$-free graphs can be seen as a second example of the usefulness of the technique of Hoáng et al. [12] that we present in Section 2 (in a more generic way). However, we emphasize that in order to obtain it we must prove a number of additional structural results. This is done in Section 3. In the same section we also show that our result is *tight* by proving that already LIST 5-COLORING is NP-complete for the class of H-free graphs whenever H has at least 5 edges and contains $P_1 + P_5$ as a subgraph.

The second aim of our paper is to initiate a parameterized complexity study for the k-COLORING and LIST k-COLORING problem restricted to H-free graphs, when H is some fixed linear forest. In Section 4 we prove the following three results: (i) LIST k-COLORING is fixed parameter tractable in $k + r$ for $(rP_1 + P_2)$-free graphs; (ii) k-COLORING restricted to $(rP_1 + P_2)$-free graphs allows a polynomial kernel when parameterized by k; and (iii) LIST k-COLORING is fixed parameter tractable in k for $(P_1 + P_3)$-free graphs.

2 A generic approach for coloring H-free graphs

We generalize the technique of Hoáng et al. [12] for proving Theorem 2, i.e., for showing that LIST k-COLORING is in XP for P_5-free graphs. Given a graph G with a k-list-assignment L, we use the following terminology.

Two adjacent vertices u and v are *essential* if $L(u) \cap L(v) \neq \emptyset$; otherwise u and v are *non-essential*. We observe that u is an essential neighbor of v if and only if v is an essential neighbor of u. Two disjoint sets of vertices are *separated* for L if no vertex in one of them has an essential neighbor in the other. Let $\mathcal{L}$ be a set of k-list-assignments of G. Then L and $\mathcal{L}$ are *compatible* if the following holds: G has a coloring respecting L if and only if G has a coloring respecting L' for some $L' \in \mathcal{L}$.

Assigning an admissible color to a vertex u does not influence the choice of admissible colors for its non-essential neighbors. Hence, in our coloring algorithm, we would like to branch in such a way that we obtain a compatible set of list-assignments for which disjoint sets of vertices become separated. Then we can apply the algorithm recursively on smaller graphs induced by these disjoint sets. This idea has been applied more often but usually leads to a huge case analysis. However, Hoáng et al. [12] developed an elegant technique, which works well for P_5-free graphs. We present it in a more generic way below.

A subset $D \subseteq V$ is a *dominating* set of G if every vertex in G belongs to D or is adjacent to a vertex of D. In that case we also say that $G[D]$ is *dominating*. Suppose that we have ordered the vertices of D as $d_1, \ldots, d_p$. Then we can define (possibly empty) sets F_i for $i = 1, \ldots, p$ as follows. Let F_1 be the set of vertices in $V \setminus D$ adjacent to d_1, and for $i = 2, \ldots, p$, let F_i be the set of vertices in $V \setminus D$ adjacent to d_i but not to any d_h with $h \leq i - 1$. The sets $F_1, \ldots, F_p$ are called *fixed* sets for D. By this definition and because D is dominating, every vertex in $V \setminus D$ belongs to exactly one fixed set F_i. We note, however, that D can have several collections of fixed sets, depending on the ordering of the vertices of D. A subset $X \subseteq V$ is *independent* if there is no edge between any two vertices of X.

We call a graph H a *dominator-separator* graph if every connected H-free graph G satisfies the following two properties.

(i) If G is k-colorable for some integer k, then G has a dominating set D of at most $f(k)$ vertices, where f is a function that only depends on k.

(ii) For any two independent sets X and Y belonging to two different fixed sets of a dominating set of G and for any k-list-assignment L of G, there exists a polynomial-time algorithm that outputs a set $\mathcal{L}$ of k-list-assignments for G such that the following hold. First, $\mathcal{L}$ is compatible with L. Second, $|\mathcal{L}| = O(h(k)n^{g(k)})$ for some functions $h(k)$ and $g(k)$ that only depend on k. Third, X and Y are separated for every $L' \in \mathcal{L}$.

By a straightforward translation of the proof of Hoàng et al. [12] one finds that for P_5-free graphs $f(k) = k$ satisfies property (i), and $h(k) = k^k$ and $g(k) = k$ satisfy property (ii). Hence, P_5 is a dominator-separator. The following theorem generalizes their approach. Its proof can be found in Appendix B.

Theorem 3. *Let H be a dominator-separator graph, and let k be a fixed integer. Then* LIST k-COLORING *can be solved in polynomial time for H-free graphs.*

3 Coloring $(rP_1 + P_5)$-free graphs

In this section, we present a new class of dominator-separator graphs, namely the class of $(rP_1 + P_5)$-free graphs. In order to apply Theorem 3 on $(rP_1 + P_5)$-free graphs we first prove that $rP_1 + P_5$ is indeed a dominator-separator graph for any fixed r. We start with the following more general lemma that we use in Section 4 as well.

Lemma 1. *Let G be an $(rP_1 + P_\ell)$-free graph for integers r and ℓ. If G contains an induced P_ℓ then G contains a dominating induced $sP_1 + P_\ell$ for some $s < r$.*

Proof. Let P be an induced P_ℓ in G. Let U consist of all vertices in G that are neither on P nor a neighbor of a vertex of P. We choose a maximal independent set S in the subgraph of G induced by U. Because G is $(rP_1 + P_\ell)$-free, we find that $|S| < r$. By maximality of S, all vertices in U are dominated by S. Hence $V_P \cup S$ is a dominating set in G. We define $s = |S|$ and observe that $V_P \cup S$ induces an $sP_1 + P_\ell$ in G. Hence, G contains a dominating induced $sP_1 + P_\ell$. □

A vertex subset K in a graph G is called a *clique* of G if there is an edge between any two vertices of K. Just as Hoáng et al. [12], we need the following result of Bacsó and Tuza [1] for the class of connected P_5-free graphs.

Theorem 4 ([1]). *Every connected P_5-free graph G has a dominating P_3 or a dominating clique.*

We are now ready to prove the following two lemmas which together show that $rP_1 + P_5$ is a dominator-separator for any fixed integer r.

Lemma 2. *Every connected $(rP_1 + P_5)$-free graph satisfies property (i).*

Proof. Let G be a connected $(rP_1 + P_5)$-free graph that is k-colorable for some integer k. We must show that G has a dominating set of size at most $f(k)$ for some function f that only depends on k. If G is P_5-free, then G has a dominating P_3 or a dominating clique due to Theorem 4. Because G is k-colorable, any clique in G has at most k vertices. Hence, we let $f(k) = \max\{3, k\}$. Suppose G is not P_5-free. By Lemma 1, G has a dominating induced $sP_1 + P_5$ for some $s < r$. Hence, we let $f(k) = s + 5$. Since $r > s$ is fixed, $f(k)$ is a constant. We conclude that G satisfies property (i). □

Lemma 3. *Every connected $(rP_1 + P_5)$-free graph satisfies property (ii).*

Proof. Let G be a connected (rP_1+P_5)-free graph on n vertices with a k-list-assignment L. Let $D = \{d_1, \ldots, d_p\}$ be a dominating set of G, and let $F_1, \ldots, F_p$ be a collection of fixed sets for D. For some $1 \leq i < j \leq n$, let $X \subseteq F_i$ and $Y \subseteq F_j$ be two independent sets of G. Note that $i < j$ implies that d_i is not adjacent to any vertex in F_j, whereas d_j might be adjacent to one or more vertices of F_i.

Let the set C consist of every color c for which there exist two adjacent vertices $x \in X$ and $y \in Y$ such that $c \in L(x) \cap L(y)$. By definition, such x and y are essential neighbors of each other. If $C = \emptyset$, then X and Y are separated.

Suppose that $C \neq \emptyset$. Our goal is to branch in such a way that the size of C decreases. For this purpose, we define X' as the set of all vertices in X that have an essential neighbor in Y, and Y' is the set of all vertices in Y that have an essential neighbor in X'. Because $C \neq \emptyset$, we find that both X' and Y' are nonempty.

We start our proof with the following claim.

For a vertex $x \in X'$ we let $N_{Y'}(x)$ denote its sets of neighbors in Y'. We say that x is *maximal* if there is no vertex $x' \in X'$ with $N_{Y'}(x) \subsetneq N_{Y'}(x')$. We say that a vertex $z \in X'$ is an *associate* of x if at least $|Y'| - r + 1$ vertices in Y' are adjacent to x or z.

Claim 1. Let $x \in X'$ be a maximal vertex. Then either x is adjacent to all vertices of Y', or any vertex $z \in X'$ adjacent to a non-neighbor of x in Y' is an associate of x.

We prove Claim 1 as follows. Suppose that $x \in X'$ is maximal and that x is not adjacent to all vertices of Y'. Let z be adjacent to a vertex $y \in Y'$ with $xy \notin E$.

In order to derive at a contradiction, suppose that Y' contains r vertices $y_1, \ldots, y_r$ neither adjacent to x nor to z. Because x is maximal and z is adjacent to a vertex in Y', namely y, that is not adjacent to x, there exists a vertex $y' \in Y'$ adjacent to x but not to z. Recall that d_i is not adjacent to any vertex of Y'. Hence, we have found an induced $P_5 = y'xd_izy$ that together with $y_1, \ldots, y_r$ forms an induced $rP_1 + P_5$ in G. This is not possible, because G is $(rP_1 + P_5)$-free. Hence, we have proven Claim 1.

We are now ready to describe the branching algorithm; we refer to Fomin and Kratsch [9] for any undefined algorithmic notions. The goal is to reduce the size of X'; if X' is empty, then C must be empty as well, and consequently X and Y will be separated. Hence, we branch on vertices of X'. Because X' may have a large size, we cannot branch by arbitrarily assigning colors to vertices of X'. Therefore, we do as follows as long as $X' \neq \emptyset$: we determine a maximal vertex $x \in X'$ and start to branch on x.

Our algorithm branches as follows: it either assigns to such a vertex x a specific color c from C, creating a number of subproblems, or no color from C at all, yet another subproblem. To obtain a subproblem of the first type we cannot only remove x from X'

but we will also ensure that we may remove c from C; this is crucial for the running time analysis which we do afterwards. To obtain the subproblem of the second type, we remove every color in C from the list of x. Consequently, x can be removed from X' as desired (but we might not have decreased the size of C during this step). We distinguish between two possible cases in which the above described branching will be applied, depending on whether or not x is adjacent to all vertices in Y'; in the latter case we must refine the branching to involve the associates of x.

Case 1. x is adjacent to all vertices of Y'.

For each color $c \in C$ we do as follows. We check if c is in the list of x. If not we stop considering c. Otherwise, we color x by c and remove c from the list of neighbors of x. Afterwards, c is not in the list of any vertex of Y', because x is adjacent to every vertex in Y'. Hence, C reduces in size by at least one color. Because x has received a color and we removed this color from the list of its neighbors, we can remove x from X'. So, X' reduces in size by at least one vertex. Thus one obtains at most $|C|$ subproblems.

If the above branching does not lead to a coloring of G that respects L, then x will not have a color from C in any coloring of G that respects L. In the corresponding subproblem we remove every color in C from the list of x. After updating the list of x in this way, we find that x is no longer in X'. So, also in this case, X' reduces in size by at least one vertex. We observe that the total number of subproblems obtained by branching on x is at most $|C| + 1$.

Case 2. x is not adjacent to all vertices of Y'.

Let $Z \subseteq X'$ consist of those vertices of X' that are adjacent to some vertex in Y' that is not adjacent to x. We observe that every $z \in Z$ is an associate of x due to Claim 1.

For each color $c \in C$ we do as follows. We check if c is in the list of x. If not we stop considering c. Otherwise, we color x by c and remove c from the list of neighbors of x. For each vertex $z \in Z$ we then do as follows. We check if c is in the list of z. If not we stop considering z. Otherwise we color z by c and remove c from the list of neighbors of z. Because z is an associate of x, there at most $r - 1$ vertices in Y' that might still have color c in their list. We assign each of these vertices a color from their list and remove them from Y'. Consequently, the lists of the remaining vertices of Y' do not contain c anymore. Hence, C reduces in size by at least one color. Because x and z received a color, and we removed this color from the lists of their neighbors, we can remove x and z from X'. So, X' reduces in size by at least two vertices. In this way we branch into at most $|C||Z|$ subproblems.

Now we branch into the subproblem where (x gets color c and) no vertex $z \in Z$ gets color c. Hence, we remove c from the lists of the vertices of Z. We claim that c is not in C anymore, which can be seen as follows. Suppose that $c \in C$. Then there are two adjacent vertices $x^* \in X'$ and $y^* \in Y'$ that each have c in their list. Because x received color c and we removed c from the lists of its neighbors, we find that y^* is no neighbor of x. However, then x^* must be in Z. This is not possible either, because we removed c from the list of every vertex in Z. We conclude that $c \notin C$, and hence, C reduces in size by at least one color. As before, we can remove x from X'. So, X' reduces in size by at least one vertex. This creates at most $|C|$ subproblems.

Finally the algorithm branches into the subproblem in which one does not assign a color from C to x. Hence, we remove every color in C from the list of x. This means that x is no longer in X'. So, in this subproblem, X' reduced in size by one vertex. This leads to yet another subproblem; and thus the total number of subproblems obtained by branching on x is at most $|C|(|Z|+1)+1$.

Having completed the overall description of our branching algorithm we prove that G satisfies property (ii) as follows. From the description of our algorithm, we conclude that each time we process a maximal vertex $x \in X'$, the size of X' reduces by at least one vertex. Hence, our algorithm will terminate. Let $\mathcal{L}$ be its output. Then $\mathcal{L}$ is a set of k-list-assignments of G for which X and Y are separated, because X', and consequently, C are empty for each of the k-list-assignments of $\mathcal{L}$. Below we show that L and $\mathcal{L}$ are compatible.

Suppose that G has a coloring ϕ respecting a list-assignment of $\mathcal{L}$. Our algorithm only reduces lists of vertices of G. This means that every list-assignment $L' \in \mathcal{L}$ has the property that $L'(u) \subseteq L(u)$ for all $u \in V$. Hence, ϕ respects L as well.

Suppose that G has a coloring ϕ respecting L. Then, in the search tree that represents our algorithm, those branches that assign each maximal vertex its color prescribed by ϕ lead to a k-list-assignment $L' \in \mathcal{L}$ that is respected by ϕ.

We are left to prove that our algorithm runs in polynomial time and that $|\mathcal{L}| = O(h(k)n^{g(k)})$ for some functions $h(k)$ and $g(k)$ that only depend on k. We note that the leaves in the search tree T are in 1-to-1 correspondence with the k-list-assignments of $\mathcal{L}$. Also, finding a maximal vertex, assigning it a color and updating its list and the lists of its neighbors takes polynomial time. This means that it suffices to show that the number of leaves in the search tree is $O(h(k)n^{g(k)})$ for some functions $h(k)$ and $g(k)$ that only depend on k. Let ℓ be a leaf of the search tree T. Then there exist q vertices $x_1, \ldots, x_q$ of X', on which we branched in order to arrive at ℓ. Each of these vertices were a maximal vertex at the moment our algorithm considered it. We call these vertices the *ℓ-vertices*.

As we have shown during the description of the algorithm, we only assign a color from C to a vertex in X' if we can remove this color from C afterwards. Maintaining this property has the following two consequences. First, the number of ℓ-vertices that received a color from C is at most $|C|$; all other ℓ-vertices got their list reduced by removing the colors of C. Second, no two ℓ-vertices received the same color from C.

For a leaf ℓ of T, we let C_ℓ denote the set of colors from C used on the ℓ-vertices. Using the above observations, we can bound the number of leaves of T as follows.

1. We fix a set $C_\ell \subseteq C$. There are at most $2^{|C|} \leq 2^k$ such sets.
2. We fix a set X'_ℓ of $|C_\ell|$ vertices in X', which correspond to the ℓ-vertices. There are at most $|X'|^{|C_\ell|} \leq n^{|C_\ell|} \leq n^{|C|} \leq n^k$ such sets.
3. We fix the order in which we assign the colors of C_ℓ to the vertices of X'_ℓ during the branching. There are at most $|C_\ell|! \leq |C|! \leq k!$ such orderings.
4. For each $x_i \in X'_\ell$ we fix an associate z_i from X'. We allow that $z_i = x_i$ in order to take into account that x_i might be adjacent to all vertices of Y', or that none of its associates get the same color. This leads to at most $|X'|^{|X'_\ell|} \leq n^{|X'_\ell|} = n^{|C_\ell|} \leq n^k$ sets of associates.
5. For each $x_i \in X'_\ell$ we choose a set R_i of at most r vertices from Y', each of which we color with a color from their lists. Because there are at most $r|Y'|^r \leq rn^r$

choices for each R_i, this leads to at most $(rn^r)^{|X'_\ell|} = (rn^r)^{|C_\ell|} \leq (rn^r)^{|C|} \leq (rn^r)^k = r^k n^{rk}$ collections of such sets, and each such set can be colored in at most k^r ways.

From the above, we find that the number of leaves, and consequently the number of k-list-assignments of $\mathcal{L}$ is at most $2^k \cdot n^k \cdot k! \cdot n^k \cdot r^k n^{rk} \cdot k^r$. Hence, we can set $h(k) = 2^k k! r^k k^r$ and $g(k) = 2k + rk$. This completes the proof of Lemma 3. $\square$

The proof of Lemma 3 differs from the proof of Hoàng et al. [12] in the following way. They define $C = L(Y')$ and show that X' contains a dominating vertex; this suffices for the case $H = P_5$ but it does not work for the case $H = rP_1 + P_5$.

Due to Lemmas 2 and 3, the graph $rP_1 + P_5$ is a dominator-separator for every fixed integer r. Hence we can apply Theorem 3 and obtain a main result of our paper.

Theorem 5. *For any fixed integers k and r, the* LIST k-COLORING *problem can be solved in polynomial time for $(rP_1 + P_5)$-free graphs.*

Theorem 5 is best possible in the sense that LIST k-COLORING becomes NP-complete for some integer k on H-free graphs, whenever H is a supergraph of $P_1 + P_5$ with at least 5 edges. In order to prove this we need the following two results. The first result is due to Broersma et al. [4].

Theorem 6 ([4]). *The* 5-PRECOLORING EXTENSION *problem is* NP*-complete for P_6-free graphs.*

We prove the second result in Appendix C.

Theorem 7. *The* LIST 5-COLORING *problem is* NP*-complete for (P_2+P_4)-free graphs.*

As explained in Section 1, the 3-COLORING problem is NP-complete for H-free graphs, whenever H is not a linear forest, due to results of Kamiński and Lozin [15] and Holyer [13]. Consequently, LIST 5-COLORING is NP-complete for such graph classes. This means that P_6 and $P_2 + P_5$ are the two smallest remaining supergraphs of $P_1 + P_5$ with exactly 5 edges. We observe that Theorems 6 and 7 imply that LIST 5-COLORING is NP-complete for P_6-free graphs and for $(P_2 + P_5)$-free graphs, respectively. This yields our desired result.

Theorem 8. *Let H be a supergraph of $P_1 + P_5$ with at least 5 edges. Then* LIST 5-COLORING *is* NP*-compete for H-free graphs.*

4 Parameterized complexity results

By Theorem 5, LIST k-COLORING is in XP for $(rP_1 + P_5)$-free graphs when k is the parameter and r is fixed. In this section we show that LIST k-COLORING is in FPT for graph classes defined by smaller linear forests as forbidden induced subgraph.

First we consider $(rP_1 + P_2)$-free graphs. For a graph $G = (V, E)$, we let $N(u) = \{v \in V \mid uv \in E\}$ denote the set of neighbors of a vertex $u \in V$, $N(S) = \{v \in V \setminus S \mid uv \in E \text{ for some } u \in S\}$ denotes the set of neighbors of a set $S \subseteq V$, and $N[S] = N(S) \cup S$.

We need the following lemma which is proved is in Appendix D.

Lemma 4. *Let $G = (V, E)$ be an $(rP_1 + P_2)$-free graph for some $r \geq 0$. If S is an independent set with $|S| \geq r$, then $X = V \setminus N(S)$ is a maximal independent set. Moreover, X is the unique maximal independent set containing S.*

Let G be a graph with a k-list assignment L. Let $\mathcal{G} = \{G_1, \ldots, G_p\}$ be a set of graphs, where each G_i has a $(k-1)$-list assignment L_i. Then we say that G and $\mathcal{G}$ are $(k-1)$-*compatible* if the following holds: G has a coloring respecting L if and only if there exists a graph $G_i \in \mathcal{G}$ that has a coloring respecting L_i. We can prove the following lemma.

Lemma 5. *Let $k \geq 2$ and $r \geq 1$. Let $G = (V, E)$ be an $(rP_1 + P_2)$-free graph on n vertices with a k-list assignment L. If G has a maximal independent set X with at least $(r-1)k + 1$ vertices, then it is possible to find in $O(k^2 n)$ time a $(k-1)$-compatible set $\mathcal{G}$ that consists of at most k induced subgraphs of G.*

Proof. Let X be a maximal independent set with at least $(r-1)k + 1$ vertices. We perform the following procedure for every color $1 \leq i \leq k$.

1 For each vertex $v \in X$ we check whether $i \in L(v)$. If so, then we color v by i and delete v afterwards. If not, we set $L_i(v) = L(v)$.
2 For each vertex $v \in V \setminus X$, we set $L_i(v) = L(v) \setminus \{i\}$.

In this way one computes a set $\mathcal{G} = \{G_1, \ldots, G_k\}$ of at most k induced subgraphs of G, where each G_i has a $(k-1)$-list assignment L_i. The running time of this procedure is $O(k^2 n)$. We are left to show that G and $\mathcal{G}$ are $(k-1)$-compatible.

First suppose that G has a coloring respecting L. Then at least r vertices in X must have the same color. Suppose that this color is i, and let S be the set of all vertices of X colored by i. Because S is an independent set with at least r vertices and X is a maximal independent set containing S, we find that $X = V \setminus N(S)$ due to Lemma 4. Because every vertex in S has color i, every vertex in $N(S)$ cannot be colored with color i. This means that we can remove color i from the list of every vertex in $N(S)$. Then every vertex $v \in X \setminus N[S]$ with $i \in L(v)$ can safely be recolored by i if it was not colored by i already. So, in the end, every vertex in X with color i in its list gets color i, and we have removed i from the list of every vertex not in X. This means that we obtain the subgraph G_i after deleting all vertices with color i from G.

To prove the reverse implication, suppose that $\mathcal{G}$ contains a graph G_i that has a coloring respecting L_i. By construction of G_i, there is no vertex of G_i that has color i in its list. Hence, color i is not used on G_i. Because we only deleted vertices from G that were independent and that had color i in their list, we can safely color these deleted vertices by color i. In this way we obtain a coloring of G that respects L. □

The proof of the following lemma is given in Appendix D.

Lemma 6. *Let $k \geq 2$ and $r \geq 1$. Let G be an $(rP_1 + P_2)$-free graph with $n \geq (r+1)^{k-1}((r-1)k+1) + (r+1)\frac{(r+1)^{k-1}-1}{r}$ vertices and m edges. Then either G has a clique of size $k+1$ or a maximal independent set X of size at least $(r-1)k+1$. Moreover, it is possible to find such a clique or independent set in $O(k(n+m))$ time.*

Now we are ready to prove the following result.

Theorem 9. *The* LIST k-COLORING *problem is in* FPT *for* $(rP_1 + P_2)$*-free graphs when parameterized by* k *and* r.

Proof. Let G be an $(rP_1 + P_2)$-free graph on n vertices that has a k-list assignment L. If $k \leq 2$, then we can solve the problem in polynomial time. If $n < f(k,r) = (r+1)^{k-1}((r-1)k+1) + (r+1)\frac{(r+1)^{k-1}-1}{r}$, then we can solve it in $O(f(k,r)^k)$ time by brute force. Otherwise, by Lemma 6, we either find a clique of size $k+1$ or a maximal independent set of size at least $(r-1)k+1$ in $O(k(n+m))$ time. In the first case, G has no coloring respecting L. In the second case, we construct in $O(k^2 n)$ time a $(k-1)$-compatible set $\mathcal{G}$ of at most k subgraphs of G by using Lemma 5. We branch on each of them and repeat the same steps. Since the depth of the search tree is bounded by k, the desired result follows. □

If we only choose k as the parameter, then we can improve our result for the k-COLORING problem as follows; see Appendix E for its proof.

Theorem 10. *For any fixed integer* $r \geq 2$*, the* k-COLORING *problem restricted to* $(rP_1 + P_2)$*-free graphs has a kernel of size* $k^2(r-1)$ *when parameterized by* k.

We recall that VERTEX COLORING is polynomial-time solvable for these graphs due to Theorem 1. However, Jansen and Scheffler [14] showed that LIST k-COLORING is NP-complete when k is part of the input, already for complete bipartite graphs which form a subclass of the class of $(P_1 + P_3)$-free graphs. We conclude this section by the following result. Its proof uses a different technique to separate fixed sets and can be found in Appendix F.

Theorem 11. *The* LIST k-COLORING *problem is in* FPT *on* $(P_1 + P_3)$*-free graphs when parameterized by* k.

5 Future Work

The following questions are relevant and open.

1. Does there exist an integer k such that k-COLORING is NP-complete for P_6-free graphs?
2. Is LIST k-COLORING parameterized by k in FPT for $(rP_1 + P_3)$-free graphs for any fixed integer $r \geq 3$?
3. Is LIST k-COLORING parameterized by k in FPT for $2P_2$-free graphs?

References

1. G. Bacsó and Zs. Tuza, Dominating cliques in P_5-free graphs, Periodica Mathematica Hungarica 21, 303–308 (1990).
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, Determining the chromatic number of triangle-free $2P_3$-free graphs in polynomial time, manuscript, http://www.dur.ac.uk/daniel.paulusma/Papers/Submitted/2p3.pdf.
5. H.J. Broersma, P.A. Golovach, D. Paulusma and J. Song, Updating the complexity status of coloring graphs without a fixed induced linear forest, manuscript, http://www.dur.ac.uk/daniel.paulusma/Papers/Submitted/updating.pdf.
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. K. Dabrowski, V. Lozin, R. Raman and B. Ries, Colouring vertices of triangle-free graphs, Proceedings of WG 2010, LNCS 6410, 184-195 (2010).
8. R.G. Downey and M.R. Fellows, Parameterized Complexity, Springer, 1999.
9. F.V. Fomin and D. Kratsch, Exact Exponential Algorithms, Springer, 2010.
10. M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco (1979).
11. M. Grötschel, L. Lovász, and A. Schrijver, Polynomial algorithms for perfect graphs, Ann. Discrete Math., Topics on Perfect Graphs 21, 325–356 (1984).
12. 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).
13. I. Holyer, The NP-completeness of edge-coloring, SIAM J. Comput. 10, 718–720 (1981).
14. K. Jansen and P. Scheffler, Generalized coloring for tree-like graphs, Discrete Appl. Math. 75 (1997), 135–155.
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. J. Kratochvíl, Precoloring extension with fixed color bound, Acta Math. Univ. Comen. 62, 139–153 (1993).
19. 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).
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. R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, 2006.
22. B. Randerath and I. Schiermeyer, 3-Colorability $\in$ P for P_6-free graphs, Discrete Appl. Math. 136, 299–313 (2004).
23. B. Randerath and I. Schiermeyer, Vertex colouring and forbidden subgraphs - a survey, Graphs Combin. 20, 1–40 (2004).
24. D. Schindl, Some new hereditary classes where graph coloring remains NP-hard, Discrete Math. 295, 197–202 (2005).
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).

A Notions in parameterized complexity

In parameterized complexity theory, we consider the problem input as a pair (I, k), where I is the main part and k the parameter. The complexity class XP consists of parameterized decision problems Π such that for each instance (I, k) it can be decided in $\mathcal{O}(f(k)|I|^{g(k)})$ time whether $(I, k) \in \Pi$, where f and g are computable functions depending only on the *parameter* k, and $|I|$ denotes the size of I. So XP consists of parameterized decision problems which can be solved in polynomial time if the parameter is a constant. A problem is *fixed parameter tractable* if an instance (I, k) can be solved in time $O(f(k)n^c)$, where f denotes a computable function and c a constant independent of k. The class $\mathsf{FPT} \subseteq \mathsf{XP}$ is the class of all fixed-parameter tractable decision problems.

A well-known technique to show that a parameterized problem Π is fixed-parameter tractable is to find a *reduction to a problem kernel*. This technique replaces an instance (I, k) of Π with a reduced instance (I', k') of Π (called *problem kernel*) such that (i) $k' \leq k$ and $|I'| \leq g(k)$ for some computable function g, (ii) the reduction from (I, k) to (I', k') is computable in polynomial time, and (iii) (I, k) is a `Yes`-instance of Π if and only if (I', k') is a `Yes`-instance of Π. An upper bound $g(k)$ of $|I'|$ is called the *kernel size*, and a kernel is called *polynomial* if the kernel size is polynomial in k. It is well known that a parameterized problem is fixed-parameter tractable if and only if it is kernelizable (cf. [21]).

B The proof of Theorem 3

Theorem 3. *Let H be a dominator-separator graph, and let k be a fixed integer. Then* LIST k-COLORING *can be solved in polynomial time for H-free graphs.*

Proof. Suppose that H is a dominator-separator and that k is some fixed integer. Let G be an H-free graph with k-list-assignment L. Let $|V_G| = n$. If G is not connected, we apply the algorithm on each connected component of G. Hence, we may assume that G is connected.

First we check if G has a dominating set of at most $f(k)$ vertices. Recall that f is a function that only depends on k and that is given to us, because G satisfies property (i). This property also tells us that G is not k-colorable if we do not find such a dominating set. A brute force search for this dominating set takes $O(n^{f(k)})$ time, which is polynomial because k is fixed.

Suppose that we find a dominating set D with $|D| \leq f(k)$. Then we fix an order $d_1, \ldots, d_{|D|}$ of the vertices of D and compute the corresponding fixed sets F_i for $i = 1, \ldots, |D|$ in polynomial time. Let G_i denote the subgraph of G induced by F_i for $i = 1, \ldots, |D|$. By definition, every vertex of each F_i is adjacent to d_i for $i = 1, \ldots, |D|$. Consequently, each G_i must have a coloring using at most $k - 1$ colors should G have a list-coloring respecting L. For $i = 1, \ldots, |D|$, we define a $(k-1)$-list-assignment L_i by assigning the list $L_i(u) = \{1, \ldots, k-1\}$ to every $u \in F_i$. Then we apply the algorithm on every G_i with list-assignment L_i in order to determine if G_i allows a $(k-1)$-coloring. If there is a subgraph G_i that has no $(k-1)$-coloring, then G has no list-coloring respecting L. Otherwise, we have found a $(k-1)$-coloring ϕ_i for every G_i. The color classes of each ϕ_i form a partition $\mathcal{X}_i$ of F_i in at most $k-1$ independent sets. Obtaining this partition was exactly the purpose of constructing these colorings. Afterwards they does not play a role anymore.

We now precolor the vertices of D in every possible way, while respecting L. In other words, the new lists of the vertices in D have size 1 in every precoloring of D. If $u \in D$ got precolored by color i, then we remove i from the list of every neighbor of u that is not in D. This gives us a set $\mathcal{L}_D$ of k-list-assignments of G that is compatible with L and that has cardinality $|\mathcal{L}_D| \leq k^{|D|} \leq k^{f(k)}$; the latter is a constant because k is fixed.

We consider each k-list-assignment $L' \in \mathcal{L}_D$. We apply property (ii) on two independent sets $X \in \mathcal{X}_i$ and $Y \in \mathcal{X}_j$ for some $1 \leq i < j \leq |D|$. This yields a set of k-list-assignments that is compatible with L', and for which X and Y are separated. Property (ii) also guarantees that the size of this set is at most $O(h(k)n^{g(k)})$ for some functions $h(k)$ and $g(k)$ that only depend on k.

Starting with each newly created list-assignment, we repeatedly apply property (ii) until all other pairs of independent sets that consist of one set from $\mathcal{X}_i$ and one from $\mathcal{X}_j$ are separated as well. The resulting set of k-list-assignments $\mathcal{L}^*$ is compatible with L'. Because the number of pairs X, Y with $X \in \mathcal{X}_i$ and $Y \in \mathcal{X}_j$ is $|\mathcal{X}_i| \cdot |\mathcal{X}_j| \leq (k-1) \cdot (k-1) \leq k^2$, the set $\mathcal{L}^*$ is obtained in polynomial time and contains $O((h(k)n^{g(k)})^{k^2})$ list-assignments. We find that F_i and F_j are separated for each list-assignment $L^* \in \mathcal{L}^*$, because every pair (X, Y) with $X \in \mathcal{X}_i$ and $Y \in \mathcal{X}_j$ is separated for L^*, and $\mathcal{X}_i$ and $\mathcal{X}_j$ are partitions of F_i and F_j, respectively.

Starting with each list-assignment of $\mathcal{L}^*$, we repeatedly apply the above procedure until all other pairs of fixed sets are separated as well. Because there are $\frac{1}{2}|D| \cdot (|D|-1) \leq \frac{1}{2}f(k) \cdot (f(k)-1) \leq f(k)^2$ such pairs, the total time that we use is polynomial and the total number of k-list-assignments that we create in this way form a set $\mathcal{L}'$ that is compatible with the list-assignment $L' \in L_D$ and that has size $O(((h(k)n^{g(k)})^{k^2})^{f(k)^2}) = O(h'(k)n^{g'(k)})$ where $h'(k) = h(k)^{k^2 f(k)^2}$ and $g'(k) = g(k)^{k^2 f(k)^2}$ are functions that only depend on k.

Recall that $|\mathcal{L}_D| \leq k^{f(k)}$. After processing all list-assignments of $\mathcal{L}_D$ as described above, we obtain a set $\mathcal{L}$ that is compatible with L and that has size $O(k^{f(k)}h'(k)n^{g'(k)})$; this number is polynomial in n because k is fixed.

We consider each k-list-assignment $L'' \in \mathcal{L}$. For $i = 1, \ldots, |D|$, let L''_i denote the restriction of L'' to F_i. By construction, every two fixed sets are separated for L''. We also recall that every vertex in D already received a color. Then, G has a coloring respecting L'' if and only if every G_i has a coloring respecting L''_i. Hence, we can apply the algorithm on each G_i. Because we updated the list of every vertex in every F_i after precoloring the vertices of D, we find that $L''_i(F_i)$ contains at most $k-1$ colors for $i = 1, \ldots, |D|$, i.e., the number of colors decreases. This means that the algorithm runs in polynomial time. Hence, we have proven Theorem 3. □

C The proof of Theorem 7

Theorem 7. *The* LIST 5-COLORING *problem is* NP*-complete for* (P_2+P_4)*-free graphs.*

Proof. We use a reduction from NOT-ALL-EQUAL 3-SATISFIABILITY with positive literals only, which we denote as NAE 3SATPL. This NP-complete problem [10], also known as HYPERGRAPH 2-COLORABILITY and SET SPLITTING, is defined as follows. Given a set $X = \{x_1, x_2, \ldots, x_n\}$ of logical variables, and a set $C = \{C_1, C_2, \ldots, C_m\}$ 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?

We consider an arbitrary instance I of NAE 3SATPL and define a graph G_I and a 5-list assignment for the vertices of G_I, and next we show that G_I is (P_2+P_4)-free and that G_I has a coloring respecting the list-assignment if and only if I has a satisfying truth assignment in which each clause contains at least one true literal and at least one false literal.

Construction of G_I.
Let I be an instance of NAE 3SATPL with variables $\{x_1, x_2, \ldots, x_n\}$ and clauses $\{C_1, C_2, \ldots, C_m\}$. We define a graph G_I corresponding to I and lists of admissible colors for its vertices based on the following construction.

- We introduce one new vertex for each of the clauses, and use the same labels $C_1, C_2, \ldots, C_m$ for these m vertices; we assume that for each of these vertices there is a list $\{1, 2, 3\}$ of admissible colors. We say that these vertices are of *C-type*.
- We introduce one new vertex for each of the variables, and use the same labels $x_1, x_2, \ldots, x_n$ for these n vertices; we assume that for each of these vertices there is a list $\{4, 5\}$ of admissible colors. We say that these vertices are of *x-type*.
- For each clause C_j, we fix an arbitrary order of its variables x_i, x_k, and x_r, and we introduce three pairs of new vertices $\{a_{i,j}, b_{i,j}\}$, $\{a_{k,j}, b_{k,j}\}$, $\{a_{r,j}, b_{r,j}\}$; we assume the following lists of admissible colors for these three pairs, respectively: $\{\{1,4\},\{2,5\}\}$, $\{\{2,4\},\{3,5\}\}$, $\{\{3,4\},\{1,5\}\}$. We say that these vertices are *auxiliary*. We add edges between x-type and auxiliary vertices whenever the first index of the auxiliary vertex is the same as of the x-type vertex. We add edges between C-type and auxiliary vertices whenever the second index of the auxiliary vertex is the same as the index of the C-type vertex. Hence each clause with three variables is represented by three 4-cycles that have one C-type vertex in common. This part of our construction is shown in Fig. 1
- We join all C-type vertices to all x-type vertices to form a complete bipartite graph with nm edges.

Theorem 7 directly follows from the following three claims. In Claim 1 we prove that the graph G_I is $P_2 + P_4$-free. In Claim 2 we prove that the G_I has a list coloring if I has a truth assignment in which each clause contains at least one true and at least one false literal. Finally, in Claim 3, we prove the converse.

Claim 1. The graph G_I is $(P_2 + P_4)$-free.

We prove Claim 1 as follows. Consider an arbitrary edge uv of G_I and denote by H the graph obtained from G_I by the removal of u, v and all the vertices adjacent with them. To prove the lemma, it is sufficient to show that H is P_4-free. Observe that the

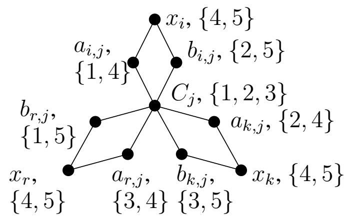

Fig. 1. Construction of G_I.

set $\{u,v\}$ contains either an x-type vertex or a C-type vertex. In the first case H has no C-type vertices. Since the graph obtained from G_I by the removal of all C-type vertices is a disjoint union of stars, H has no induced P_4. In the second case H has no x-type vertices, and by the same arguments we conclude that H is P_4-free. This completes the proof of Claim 1.

Claim 2. If I has a truth assignment in which each clause contains at least one true and at least one false literal, then the graph G_I has a coloring that respects the list-assignment.

We prove Claim 2 as follows. Suppose I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal. We use color 4 to color the x-type vertices representing the true literals and color 5 for the false literals. Now consider the lists assigned to the auxiliary vertices that come in pairs chosen from $\{\{1,4\},\{2,5\}\}$, $\{\{2,4\},\{3,5\}\}$, $\{\{3,4\},\{1,5\}\}$. If the adjacent x-type vertex has color 4, color 1, 2 or 3 is forced on one of the adjacent auxiliary vertices, respectively, while on the other one we can use color 5; similarly, if the adjacent x-type vertex has color 5, color 2, 3 or 1 is forced on one of the adjacent auxiliary vertices, respectively, while on the other one we can use color 4. Since precisely two of the three x-type vertices of one clause gadget have the same color, this leaves at least one of the colors 1, 2 and 3 admissible for the C-type vertex representing the clause. This completes the proof of Claim 2.

Claim 3. If G_I has a coloring that respects the list-assignment, then I has a satisfying truth assignment in which each clause contains at least one true and at least one false literal.

We prove Claim 3 as follows. Suppose we have a coloring of the graph G_I that respects the lists-assignment. Then each of the x-type vertices has color 4 or 5, and each of the C-type vertices has color 1, 2 or 3. We define a truth assignment that sets a variable to TRUE if the corresponding x-type vertex has color 4, and to FALSE otherwise. Suppose one of the clauses contains only true literals. Then the three x-type vertices in the corresponding clause gadget of G_I all have color 4. Now consider the lists assigned to the auxiliary vertices of this gadget that come in pairs chosen from $\{\{1,4\},\{2,5\}\}$, $\{\{2,4\},\{3,5\}\}$, $\{\{3,4\},\{1,5\}\}$. Since the adjacent x-type vertices all have color 4, colors 1, 2 and 3 are forced on three of the auxiliary vertices adjacent to the C-type vertex of this gadget, a contradiction, since the C-type vertex has color 1, 2 or 3. This proves that every clause contains at least one false literal. Analogously, it is easy to show that every clause contains at least one true literal. This completes the proof of Claim 3.

Due to Claims 1-3, Theorem 7 is valid. □

D The proof of Lemmas 4 and 6

Lemma 4. *Let $G = (V, E)$ be an $(rP_1 + P_2)$-free graph for some $r \geq 0$. If S is an independent set with $|S| \geq r$, then $X = V \setminus N(S)$ is a maximal independent set. Moreover, X is the unique maximal independent set containing S.*

Proof. Suppose that S is an independent set with at least r vertices. Let $X = V \setminus N(S)$. Because G is $(rP_1 + P_2)$-free and S is independent, $V \setminus N[S]$ is independent as well. Because a neighbor of a vertex of S does not belong to any independent set that contains S, we find that X is the unique maximal independent set containing S. This completes the proof of Lemma 4. □

Lemma 6. *Let $k \geq 2$ and $r \geq 1$. Let G be an $(rP_1 + P_2)$-free graph with $n \geq (r+1)^{k-1}((r-1)k+1) + (r+1)\frac{(r+1)^{k-1}-1}{r}$ vertices and m edges. Then either G has a clique of size $k+1$ or a maximal independent set X of size at least $(r-1)k+1$. Moreover, it is possible to find such a clique or independent set in $O(k(n+m))$ time.*

Proof. The case $m = 0$ is trivial. Suppose that $m \geq 1$. We apply Lemma 1 for $\ell = 2$. This yields a dominating $sP_1 + P_2$ of G for some $s < r$. The vertex set of this subgraph is a dominating set of G that has size $s + 2 \leq r + 1$. Hence, it contains a vertex v of degree at least $\frac{n-(r+1)}{r+1} \geq (r+1)^{k-2}((r-1)k+1) + (r+1)\frac{(r+1)^{k-2}-1}{r}$. We can apply the same arguments inductively for the subgraph of G induced by $N(v)$. Then after at most $k-1$ steps we either obtain a clique of size $k+1$, or else we find that we cannot apply Lemma 1 any longer. The latter case means that the graph under consideration has no edges. Then its vertices form an independent set Y of size at least $(r-1)k+1 \geq r$, as $k \geq 2$. Hence, $X = V \setminus N(Y)$ is a maximal independent set due to Lemma 4.

Because a vertex of maximum degree can be found in $O(n+m)$ time, the total running time of our procedure is $O(k(n+m))$. This completes the proof of Lemma 6. □

E The proof of Theorem 10

Theorem 10. *For any fixed integer $r \geq 2$, the k-*COLORING *problem restricted to $(rP_1 + P_2)$-free graphs has a kernel of size $k^2(r-1)$ when parameterized by k.*

Proof. Let k be a positive integer, and let $G = (V, E)$ be an $(rP_1 + P_2)$-free graph for some fixed integer $r \geq 2$. If $k \leq 2$ then we can solve k-COLORING in polynomial time. Suppose that $k \geq 3$. If G has at most $k^2(r-1)$ vertices, then we are done. Suppose that G has at least $k^2(r-1)+1$ vertices. We check if G has an independent set S of r vertices such that $X = V \setminus N(S)$ contains at least $k(r-1)+1$ vertices. If not then we output $\texttt{No}$. Otherwise we give every vertex in X color k. We then delete X from G and check if the resulting graph G' has a $(k-1)$-coloring recursively. Finally, we either solve the problem or get an instance (G', k') of k'-COLORING for $k' \leq k$ and the graph G' with at most $k'^2(r-1)$ vertices.

We now prove that the above approach leads to a kernel of size $k^2(r-1)$. For every k-coloring of G, there must exist an independent set X with at least $k(r-1)+1$ vertices in G that all get the same color. We may without loss of generality assume that X is maximal. Because $k \geq 2$, we find that $k(r-1)+1 \geq r$. Hence, X contains an independent set S of size r. By Lemma 4, we find that $V \setminus N(S)$ is the unique maximal independent set containing S. Because X is maximal as well and $S \subseteq X$, we deduce that $X = V \setminus N(S)$. Hence, G has no k-coloring if G has no independent set S with $|S| = r$ and $|V \setminus N(S)| \geq k(r-1)+1$. Suppose we do find such a set S. Let $X = V \setminus N(S)$. Then, for any k-coloring of G, there exists a set $S' \subseteq X$ of vertices that get the same color, because X contains at least $k(r-1)+1$ vertices. We may assume without loss of generality that this color is k. By Lemma 4, we find that $V \setminus N(S')$ is the unique maximal independent set containing S'. Because X contains S' as well, we find that $X = V \setminus N(S')$. Because every vertex in S' received color k, no vertex in $N(S')$ will receive color k. This means that we can safely color every vertex in $X \setminus S'$ with color k as well. Consequently the graph G' obtained after deleting X must have a $(k-1)$-coloring, should G have a k-coloring.

We are left to show that the running time of our kernelization algorithm is polynomial. This follows from the observation that there are at most $|V|^r$ sets of size r and r is fixed. Hence, we can determine all independent sets of size r in polynomial time. This completes the proof of Theorem 10. □

F The proof of Theorem 11

Theorem 11. *The* LIST k-COLORING *problem is* FPT *on* $(P_1 + P_3)$*-free graphs when parameterized by* k.

Proof. Clearly, we may restrict ourselves to connected graphs. Let $G = (V, E)$ be a connected $(P_1 + P_3)$-free graph with a k-list assignment L.

First suppose that G has no induced P_3. Then V is a clique. If V has at least $k+1$ vertices, then G has no coloring that satisfies L. If V has at most k vertices, then we try to color V in every possible way by brute force.

Now suppose that G contains an induced $P_3 = d_1d_2d_3$. Because G is $(P_1 + P_3)$-free, $D = \{d_1, d_2, d_3\}$ is a dominating set. We construct fixed sets F_1, F_2, F_3 for D and prove a number of properties of these sets.

Claim 1. If G has a coloring respecting L then F_3 is a clique on at most $k-1$ vertices.

We prove Claim 1 as follows. Suppose F_3 is not a clique. Then there exist two vertices x and y in F_3 that are not adjacent. Consequently, d_1 and the path xd_3y form an induced $P_1 + P_3$, which is not possible. Hence F_3 is a clique. Suppose that $|F_3| \geq k$. Then $F_3 \cup \{d_3\}$ is a clique on at least $k+1$ vertices, and G has no coloring respecting L. Hence $|F_3| \leq k-1$. This completes the proof of Claim 1.

Claim 2. If G has a coloring respecting L then F_2 induces a disjoint union of complete graphs, each of which has as most $k-1$ vertices.

We prove Claim 2 as follows. Suppose $G[F_2]$ contains a connected component with two non-adjacent vertices. Then this component contains an induced P_3. However, d_1 is not adjacent to any vertex of F_2. Consequently, this induced P_3 and d_1 form an induced $P_1 + P_3$, which is not possible. Hence, F_2 induces a disjoint union of complete graphs. Suppose that $G[F_2]$ contains a connected component F of size at least k. Then $F \cup \{d_2\}$ is a clique on at least $k+1$ vertices, and G has no coloring respecting L. This is impossible which completes the proof of Claim 2.

Let $X \subseteq F_1$ be the set of all vertices of F_1 that are not adjacent to any vertex of F_2.

Claim 3. If G has a coloring respecting L and $F_2 \neq \emptyset$ then X is a clique on at most $k-1$ vertices.

We prove Claim 3 as follows. Suppose that $F_2 \neq \emptyset$ and that X contains two non-adjacent vertices x and y. Because $F_2 \neq \emptyset$, there exists a vertex $z \in F_2$. Then z and the path xd_1y form an induced $P_1 + P_3$, which is not possible. Hence, X is a clique. If $|X| \geq k$, then $X \cup \{d_1\}$ is a clique on at least $k+1$ vertices, and G has no coloring respecting L, contradiction. This completes the proof of Claim 3.

Claim 4. If $G[F_2]$ has at least two connected components, then each vertex of F_2 is adjacent to all vertices of $F_1 \setminus X$.

We prove Claim 4 as follows. Suppose that there is a vertex $x \in F_2$ that is not adjacent to a vertex $y \in F_1 \setminus X$. Because $y \notin X$, we find that y is adjacent to a vertex $z \in F_2$. If $xz \notin E$, then the path zyd_1 and the vertex x form an induced $P_1 + P_3$. Consequently, $xz \in E$. So, y can only be adjacent to the vertices of the connected component of $G[F_2]$ that contains x. Let v be a vertex of another connected component of $G[F_2]$. Then v and the path xzy form an induced $P_1 + P_3$, which is not possible. This proves Claim 4.

We are now ready to describe our algorithm. First, we guess a coloring of F_3 and the vertices d_1, d_2, d_3. We consider three cases.

Case 1. $F_2 = \emptyset$.

We update the lists of the vertices of F_1 as follows. For each vertex $v \in F_1$, we delete color $i \in L(v)$ from its list whenever v is adjacent to a vertex in $F_3 \cup \{d_1, d_2, d_3\}$ that was colored by i. We remove all vertices not in F_1 from G.

Case 2. $F_2 \neq \emptyset$ and $G[F_2]$ is connected.

In addition to the guessed colors of F_3 and d_1, d_2, d_3, we guess a coloring of F_2. Then we do the same as we did in Case 1.

Case 3. $G[F_2]$ has at least two components.

We first find the set X and then guess a coloring of X. Then we guess a set C of all colors that will be used for the coloring of F_2. We update the lists of every $v \in F_2$; its new list consists of all colors from $L(v) \cap C$ that are no color of one of its neighbors from $F_3 \cup \{d_2, d_3\}$. We update the lists of every $v \in F_1 \setminus X$; its new list consists of all colors from $L(v) \setminus C$ that are no color of one of its neighbors from $F_3 \cup X \cup \{d_1, d_2, d_3\}$

We observe that F_2 and $V \setminus F_2$ are separated if $F_2 \neq \emptyset$. We also observe that $F_1 \setminus X$ and the set of other vertices of G are separated. Hence we can consider the graphs $G[F_2]$ and $G[F_1 \setminus X]$ independently.

Correctness of the algorithm immediately follows from its description. It remains to show that it runs in FPT time. To prove this, we consider the worst case, namely Case 3. In this case we guess a coloring of d_1, d_2, d_3, F_3, X and a set of colors C. Note that there are at most 2^k different sets C. By Claims 1 and 3 we then find that there are at most $k^{2k+1} \cdot 2^k$ choices to branch on. For each branch we solve the problem for $G[F_2]$ in $O(k^{k-1}|F_2|)$ time, due to Claim 2. Because d_1 is adjacent to all vertices of F_1, at least one color cannot be used on F_1. Hence, we must solve the LIST $(k-1)$-COLORING problem with input $G[F_1 \setminus X]$ in every branch. We conclude that the depth of the search tree is bounded by k, and the desired result follows. $\square$