Determining the chromatic number of triangle-free $2P_3$-free graphs in polynomial time*

Hajo Broersma, Petr A. Golovach, Daniël Paulusma, and Jian Song **

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

Abstract. Let $2P_3$ denote the disjoint union of two paths on three vertices. A graph G that has no subgraph isomorphic to a graph H is called H-free. The Vertex Coloring problem is the problem to determine the chromatic number of a graph. Its computational complexity for triangle-free H-free graphs has been classified for every fixed graph H on at most 6 vertices except for the case $H = 2P_3$. This remaining case is posed as an open problem by Dabrowski, Lozin, Raman and Ries. We solve their open problem by showing polynomial-time solvability.

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 ℓ-Coloring problem is the problem to decide whether a graph can be colored with at most ℓ colors. The related Vertex Coloring problem is to determine the smallest number of colors a graph can be colored with. Due to the fact that ℓ-Coloring is NP-complete for any fixed $\ell \geq 3$, 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 that ℓ-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 ℓ-Coloring and Vertex Coloring problem restricted to graphs characterized by one or more forbidden induced subgraphs. This problem has been studied in many papers by different groups of researchers [4–6, 8, 12–14, 16–18, 22, 25, 26].

If a graph G does not contain an induced subgraph that is isomorphic to a graph H, then G is called *H-free*. By combining several results from the literature with a number of new results we obtained the following result that even holds for the precoloring extension version of 3-Coloring [5]. Here, a *linear forest* is the disjoint union of a collection of paths.

* A preliminary and shortened version of the results in this paper will appear in the proceedings of ISAAC 2010.

** This work has been supported by EPSRC (EP/G043434/1).

Theorem 1 ([5]). *Let H be a fixed graph on at most 6 vertices. Then the* 3-Coloring *problem for H-free graphs is polynomial-time solvable if H is a linear forest; otherwise it is* NP*-complete.*

The complexity status of the 3-Coloring problem restricted to H-free graphs is open for many graphs H on seven or more vertices, in particular for paths. It is even unknown whether there exists a fixed integer $k \geq 7$ such that 3-Coloring is NP-complete for P_k-free graphs. Here, P_k denotes the path on k vertices.

For larger values of ℓ, more is known on the complexity status of the ℓ-Coloring problem restricted to P_k-free graphs. The currently sharpest known results are that 4-Coloring is NP-complete for P_8-free graphs [5] and that 6-Coloring is NP-complete for P_7-free graphs [4]. It is unknown whether there exists an integer ℓ such that ℓ-Coloring is NP-complete for P_6-free graphs.

Hoàng et al. [12] showed that ℓ-Coloring for any fixed integer ℓ is polynomial-time solvable for P_5-free graphs. In contrast, Král' et al. [16] proved that Vertex Coloring is NP-hard on P_5-free graphs. In fact, they give a complete complexity classification of the Vertex Coloring problem restricted to graphs in which one fixed graph is forbidden as an induced subgraph. In particular, this problem is NP-hard for triangle-free graphs. These graphs are also called K_3-free graphs.

The result of Král' et al. [16] motivated a study by Kamiński and Lozin [13] on the computational complexity of the Vertex Coloring problem on triangle-free graphs with one extra forbidden subgraph H. They showed that Vertex Coloring is NP-hard for triangle-free H-free graphs for any fixed graph H that is not a forest. Let $K_{1,5}$ denote the 6-vertex star. Maffray and Preissmann [21] showed that Vertex Coloring is NP-hard for triangle-free $K_{1,5}$-free graphs.

A very recent paper of Dabrowski et al. [8] deals with the computational complexity of the Vertex Coloring problem for triangle-free H-free graphs, where H is a forest on at most 6 vertices not isomorphic to $K_{1,5}$. They prove that the problem is polynomial-time solvable for all such cases, except when $H = 2P_3$, i.e., the disjoint union of two paths on 3 vertices. For a number of cases they use a generic approach, which we describe below.

Let H be a forest on at most 6 vertices. First show that the class of triangle-free H-free graphs has bounded clique-width. This means that for any fixed integer k the k-Coloring problem can be solved in polynomial time for triangle-free H-free graphs by applying the result of Courcelle, Makowsky and Rotics [7]. This does not immediately lead to the desired result for Vertex Coloring for this graph class. However, it does so, if one can determine a constant c that is an upper bound on the chromatic number of every triangle-free H-free graph. This is the final part of their approach.

As we shall see later on, every triangle-free $2P_3$-free graph can be colored with at most 5 colors, so the chromatic number of such graphs is bounded by a constant. However, Lozin and Volz [20] showed that already bipartite $2P_3$-free graphs can have arbitrarily large clique-width. Hence, the above approach cannot be used. Dabrowski et al. [8] leave the case $H = 2P_3$ as an open problem and mention that it could be NP-hard. We disprove this by presenting a polynomial-time algorithm that solves Vertex Coloring for triangle-free $2P_3$-free graphs.

Our result together with all the aforementioned results leads to the following theorem.

Theorem 2. *Let H be a fixed graph on at most 6 vertices. Then the* Vertex Coloring *problem for triangle-free H-free graphs is polynomial-time solvable if H is a forest not isomorphic to $K_{1,5}$; otherwise it is* NP*-hard.*

Our polynomial-time algorithm that solves the Vertex Coloring problem for triangle-free $2P_3$-free graphs tests if the input graph can be colored with ℓ colors for increasing value of ℓ. As we mentioned above, a triangle-free $2P_3$-free graph can always be colored with at most 5 colors; we will prove this in in Section 4. Hence, our algorithm terminates. It runs in polynomial time because of the following reasons. Firstly, the 2-Coloring problem is trivial. Secondly, by Theorem 1, we can solve the 3-Coloring problem for this graph class (even without requiring triangle-freeness). Thirdly, we can solve the 4-Coloring problem in polynomial time for triangle-free $2P_3$-free graphs; we will prove this in Section 3. We start by stating some basic terminology and observations in Section 2.

2 Preliminaries

We only consider finite undirected graphs without loops and without multiple edges. Let $G = (V, E)$ be a graph. For $u \in V$ let $N_G(u) = \{v \mid uv \in E\}$ denote the *neighborhood* of u, and let $d_G(u) = |N_G(u)|$ denote the *degree* of u. Let U be a subset of V. Then we define $N_G(U) = \{v \in V \setminus U \mid uv \in E \text{ for some } u \in U\}$. We write $G[U]$ to denote the subgraph of G *induced by* the vertices in U, i.e., the subgraph of G with vertex set U and an edge between two vertices $u, v \in U$ whenever $uv \in E$. Furthermore, U is called a *dominating set* of G if every vertex of G is in U or adjacent to a vertex of U, and U is called an *independent set* if there is no edge between any two vertices in U. If $G[U]$ is a *complete* graph, i.e., if there is an edge between any two vertices of U, then U is called a *clique*. The complete graph on n vertices is denoted K_n.

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 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. An ℓ-*coloring* of G is a coloring c of G with $c(V) \subseteq \{1, \ldots, \ell\}$. Here we use the notation $c(U) = \{c(u) \mid u \in U\}$ for $U \subseteq V$. We let $\chi(G)$ denote the *chromatic number* of G, i.e., the smallest ℓ such that G has an ℓ-coloring. We say that a graph G is ℓ-*chromatic* if $\chi(G) = \ell$ and ℓ-*colorable* if $\chi(G) \leq \ell$. We recall that the problem ℓ-Coloring is to decide whether a given graph admits an ℓ-coloring, and that the Vertex Coloring problem is to determine 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 $u \in V$. We say that a coloring $c : V \to \{1, 2, \ldots\}$ *respects* L if $c(u) \in L(u)$ for all $u \in V$. In this case we

also call c a *list-coloring*. The problem of finding such a coloring of a graph is relevant for us in the following sense. We begin our coloring algorithm in Section 3 by assigning a list $\{1, 2, 3, 4\}$ of colors to each vertex of the input graph $G = (V, E)$. Then, in order to start some branching, we sometimes color the vertices of a subset $W \subseteq V$ (we *precolor* W) in every possible way. Afterwards we does as follows for each $u \in W$. If u got precolored by color i we remove this color from the list of every neighbor of u that is not in W. In that case we say that we *update* the lists, and then we must find a coloring that respects the new lists.

The coloring algorithm in Section 3 makes frequently use of the following well-known observation, the proof of which follows from the fact that the decision problem in this case can be modeled and solved as an instance of the 2-SATISFIABILITY problem. This approach has been introduced by Edwards [9] and is folklore now.

Observation 1 ([9]) *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.*

3 Coloring $(K_3, 2P_3)$-free graphs with at most four colors

Our polynomial-time algorithm for solving 4-COLORING for $(K_3, 2P_3)$-free graphs heavily relies on a number of structural properties of 4-colorable $(K_3, 2P_3)$-free graphs. We present these properties together with some other useful observations in Section 3.1. Then in Sections 3.2–3.4 we present our algorithm.

3.1 Structural properties

Let $G = (V, E)$ be a $2P_3$-free graph. Let I be an independent set in G, and let X be a subset of $V \backslash I$. We write $I(X) := N_G(X) \cap I$ and $I(\overline{X}) := I \backslash N_G(X)$, so $I = I(X) \cup I(\overline{X})$ and $I(X) \cap I(\overline{X}) = \emptyset$. If every vertex in $N_G(I) \backslash X$ has at most one neighbor in $I(\overline{X})$ then we say that X *pseudo-dominates* I. An example of a set X that pseudo-dominates a set I is illustrated in Figure 1.

We need the following two lemmas. Lemma 1 is an improvement of a similar lemma for sP_3-free graphs from our previous paper [5] for the case $s = 2$. Lemma 2 is the direct translation of a lemma for sP_3-free graphs from our previous paper [5] for the case $s = 2$.

Lemma 1. *Let I be an independent set in a $2P_3$-free graph $G = (V, E)$. Then $G[V \backslash I]$ contains a clique X that pseudo-dominates I.*

Proof. Let $G = (V, E)$ be a $2P_3$-free graph, and let I be an independent set in G. Let X be a clique in $G[V \backslash I]$ such that there is no clique X' in $G[V \backslash I]$ with $|I(X')| > |I(X)|$.

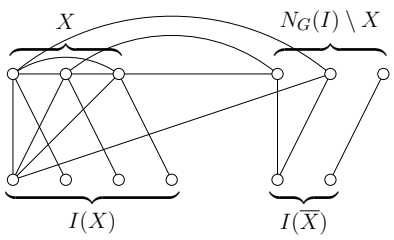

Fig. 1. A set X that pseudo-dominates a set I.

We will prove that X pseudo-dominates I. In order to derive a contradiction assume that X does not pseudo-dominate I. Then there is a vertex $v \in V \setminus (I \cup X)$ with at least two neighbors z, z' in $I(\overline{X})$. The following claim is trivially true for any $x \in X$ with exactly one neighbor in I. Because G is $2P_3$-free, the claim also holds if such an x has more than one neighbor in I.

Claim 1. Let $x \in X$. If $vx \notin E$, then v is adjacent to at least $|N(x) \cap I| - 1$ neighbors of x in I.

Let $X_1 = N(v) \cap X$ and let $X_2 = X \setminus X_1$. If $X_1 = X$ then $X \cup \{v\}$ contradicts our choice of X, because $X \cup \{v\}$ is a clique with $|I(X \cup \{v\})| > |I(X)|$. Hence, $X_1 \neq X$, so $X_2 \neq \emptyset$.

Let $I^*(X_2)$ consist of all vertices in $I(X_2)$ that are adjacent to every vertex in X_2. We claim that v is adjacent to every vertex in $I(X_2) \setminus I^*(X_2)$. In order to see this, suppose there exist two vertices $x \in X_2$ and $w \in I(X_2)$ such that $vw \notin E$ and $wx \notin E$. By definition of $I(X_2)$, there exists a vertex $x' \in X_2$ with $wx' \in E$, so $x' \neq x$. However, then $wx'x$ and zvz' are two induced paths on three vertices that form an induced $2P_3$ in G. This is not possible, because G is $2P_3$-free. Hence, indeed v is adjacent to every vertex in $I(X_2) \setminus I^*(X_2)$. By Claim 1, v is adjacent to at least $|N(x) \cap I| - 1$ neighbors of any $x \in X_2$ in I. Hence, we find that v is adjacent to every vertex in $I(X_2)$ except perhaps one. However, then $X_1 \cup \{v\}$ is a clique with $|I(X_1 \cup \{v\})| > |I(X)|$. This contradicts our choice of X. We conclude that X pseudo-dominates I. □

Lemma 2 ([5]). *Let G be a $2P_3$-free graph that contains a set X and an independent set I, such that X pseudo-dominates I. Let $k \geq 1$. If $I(\overline{X})$ contains more than k vertices with degree at least k in G, then G is not k-colorable.*

The following lemma states a useful relationship between k-colorability of $(K_3, 2P_3)$-free graphs with minimum degree at least k and the existence of a dominating set, the size of which is bounded by a linear function in k. Its proof uses Lemmas 1 and 2.

Lemma 3. *Let G be a $(K_3, 2P_3)$-free graph with minimum degree at least k for some integer k. If G is k-colorable, then G contains a dominating set D of size at most $2k + 7$, such that the size of a maximum independent set in $G[D]$ is at most $2k + 4$.*

Proof. Let G be a $(K_3, 2P_3)$-free graph with minimum degree at least k for some integer k. Assume that G is k-colorable. If G has a component that is a complete graph, then this component must have at least $k+1$ vertices due to our minimum degree assumption. However, then G is not k-colorable. Hence, such a component does not exist. Then G contains an induced path uvw. If $\{u, v, w\}$ is a dominating set of G, then the statement of the lemma holds. Suppose $\{u, v, w\}$ is not a dominating set of G.

Let G' be the graph obtained from G after removing u, v, w, and all vertices in $N_G(\{u, v, w\})$. Because G is $(K_3, 2P_3)$-free and u, v, w form an induced P_3, we find that G' is (K_3, P_3)-free. Hence, every component of G' is isomorphic to K_1 or to K_2.

We partition the vertices of G' into at most 2 independent sets I_1 and I_2 as follows. First we form I_1 by taking exactly one vertex from each component of G'. We remove I_1 from G' and repeat the above step to obtain I_2 if there were any vertices of G' left. This is indeed a partition of $V(G')$, because every component of G' has at most 2 vertices at the start of this procedure.

We apply Lemma 1 to each I_h in order to find a clique X_h in G that pseudo-dominates I_h. Because G is K_3-free, $|X_h| \leq 2$ for $h = 1, 2$.

We apply Lemma 2 to G and each I_h in order to find that $I_h(\overline{X_h})$ has size at most k for $h = 1, 2$. Then $D = \{u, v, w\} \cup X_1 \cup X_2 \cup I_1(\overline{X_1}) \cup I_2(\overline{X_2})$ is a dominating set of G that has at most $3 + 2 + 2 + k + k = 2k + 7$ vertices. We observe that the size of a maximum independent set in $G[D]$ is at most $2 + 1 + 1 + k + k = 2k + 4$. This completes the proof of Lemma 3. □

We observe that the graph in Lemma 3 is required to have minimum degree at least k. This is not a problem for our algorithm (which assumes $k = 4$) due to the following well-known procedure. Let G be a graph. Remove all vertices with degree at most $k - 1$ from G. Propagate this until we obtain a graph with minimum degree at least k, denoted as $G_{\geq k}$. We note that $G_{\geq k}$ might be empty and observe the following, see e.g. [5] for a proof.

Observation 2 *Let k be a fixed integer. Then $G_{\geq k}$ can be obtained in polynomial time, and $G_{\geq k}$ is k-colorable if and only if G is k-colorable. Furthermore, for any set $\mathcal{H}$ of graphs, $G_{\geq k}$ is $\mathcal{H}$-free if G is $\mathcal{H}$-free.*

We now present our algorithm that solves the 4-Coloring problem for the class of $(K_3, 2P_3)$-free graphs.

3.2 Outline of the algorithm

Our algorithm first assigns a list with colors $1, 2, 3, 4$ to every vertex of the input graph G. Our goal is to reduce the list of every vertex to a list with at most two admissible colors such that Observation 1 can be used. For this purpose our algorithm first preprocesses G, thereby reducing the lists of admissible colors of every vertex by at least one. This preprocessing heavily relies on Lemma 3

and is explained in detail in Section 3.3. After the preprocessing stage, we either find that G has no 4-coloring, or else we find a constant-bounded number of so-called suitable list-assignments of G. Due to the preprocessing, every list in every suitable list-assignment is a proper subset of $\{1,2,3,4\}$, thus of size at most three. We show the following claim:

G has a 4-coloring if and only if G has a coloring respecting one of the suitable list-assignments of G.

However, for a suitable list-assignment L', we might not be able to apply Observation 1 immediately, because there may exist vertices u with $|L'(u)| = 3$. In Section 3.4 we apply a polynomial-time branching algorithm that reduces the size of such lists, such that Observation 1 may be used.

During the execution of the algorithm some vertices may get an empty list of admissible colors at some moment. In that case our algorithm can immediately output `No`. We do not write this explicitly in the description of our algorithm, because such a case will be spotted anyway, namely at the moment we apply Observation 1. At the end of Section 3.4, we explain a few other directions for decreasing the running time of our polynomial-time algorithm even further.

3.3 The preprocessing

Let G be a $(K_3, 2P_3)$-free graph, every vertex u of which has a list $L(u) = \{1,2,3,4\}$ of admissible colors. By Observation 2, we may assume that G has minimum degree at least 4. We preprocess G in three phases. If at some moment we precolor a vertex u with a certain color, then we remove this color from the list of every neighbor of u. Recall that in that case we say that we update the lists.

Phase 1. Reduce the list sizes by at least 1.
The algorithm checks if G has a dominating set D of size at most $2 \cdot 4 + 7 = 15$. If not, it outputs `No`. Suppose G has such a dominating set D. Then the algorithm precolors every vertex of D with a color from $\{1,2,3,4\}$ and updates the lists of the other vertices afterwards.

After Phase 1, we can partition the set of vertices of G into five sets A, B_1, B_2, B_3, B_4, some of which may be empty. They are defined as follows. We let A consist of all vertices with a list of at most two admissible colors. Observe that we have not removed the vertices of D. Because these have been precolored, they have a list of exactly one admissible color. Hence, by definition, $D \subseteq A$. For $i = 1, \ldots, 4$, we let B_i consist of all vertices with list $\{1,2,3,4\} \backslash \{i\}$. We note that each $G[B_i]$ contains at most one component on more than two vertices due to our assumption that G is $(K_3, 2P_3)$-free. We denote this component by H_i if it exists.

Phase 2. Precolor an induced P_3 in each H_i.
For $i = 1, \ldots, 4$ the algorithm acts as follows. It finds three vertices a_i, b_i, c_i that

form an induced P_3 in H_i and precolors these vertices. Afterwards it updates the list of the other vertices.

After Phase 2 we redefine A, B_1, B_2, B_3, B_4 by moving every vertex whose list got reduced to size at most two from $B_1 \cup B_2 \cup B_3 \cup B_4$ to A. We note that for $i = 1, \ldots, 4$ the vertices a_i, b_i, c_i are now in A (if they exist) and so are their neighbors in H_i. Because G is $(K_3, 2P_3)$-free, the remaining vertices of each H_i induce a set of isolated vertices and edges in G. Hence, for $i = 1, \ldots, 4$, every component of $G[B_i]$ is either a vertex or an edge. We write F_i for the subgraph of $G[B_i]$ induced by the vertices of the components isomorphic to K_2. So, each F_i is the disjoint union of a number of edges.

Phase 3. Precolor each F_i.
For $i = 1, \ldots, 4$, the algorithm precolors every vertex of F_i respecting its list of admissible colors. Afterwards it updates the lists of the other vertices.

After Phase 3, we denote the resulting list-assignment by L' and call L' a *suitable* list-assignment of G. We redefine A, B_1, B_2, B_3, B_4 by moving every vertex whose list got reduced to size at most two from $B_1 \cup B_2 \cup B_3 \cup B_4$ to A. In Phase 2 we got rid of any components of $G[B_i]$ on more than 2 vertices, and in Phase 3 we got rid of any components isomorphic to K_2. Hence, each B_i now induces a set of isolated vertices in G.

Before we continue with the description of our algorithm, we need to show the following two lemmas. Lemma 4 shows that we can restrict ourselves to suitable list-assignments of G. Note that G has no suitable list-assignment if our algorithm has outputted `No` in Phase 1, 2 or 3. Otherwise, the number of suitable list-assignments depends on the number of different precolorings in Phase 1, 2 and 3. Hence, G may have many suitable list-assignments. However, Lemma 5 shows that the number of suitable list-assignments is bounded by a constant and that we can find all of them in polynomial time.

Lemma 4. *Let G be a $(K_3, 2P_3)$-free graph with minimum degree at least 4. Then G has a 4-coloring if and only if there exists a suitable list-assignment L' such that G has a coloring respecting L'.*

Proof. Let G be a $(K_3, 2P_3)$-free graph with minimum degree at least four. First suppose G has a 4-coloring c. Consider Phase 1. Because G is $(K_3, 2P_3)$-free and has minimum degree at least 4, we can apply Lemma 3 to find that G has a dominating set of size at most $2 \cdot 4 + 7 = 15$. We precolor D according to c and update the lists of the other vertices. We then color the three vertices a_i, b_i, c_i of an induced P_3 in every nonempty H_i according to c. Afterwards we update the lists of the other vertices. Finally, in Phase 3, we color the two end-vertices of each edge in F_i according to c for $i = 1, \ldots, 4$. Afterwards we update the lists of the other vertices. This leads to our desired suitable list-assignment L' of G.

Now suppose there exists a suitable list-assignment L' such that G has a coloring respecting L'. By definition, this coloring is a 4-coloring of G. This completes the proof of Lemma 4. □

Lemma 5. *Let G be a $(K_3, 2P_3)$-free graph with minimum degree at least 4. Then the number of suitable list-assignments of G is constant-bounded and can be obtained in polynomial time.*

Proof. Let G be a $(K_3, 2P_3)$-free graph on n vertices that has minimum degree at least 4. The running time of Phase 1 is dominated by the time it takes to find the dominating set D of G if it exists. Because $|D| \leq 15$, this takes $O(n^{15})$ time. The running time of Phase 2 is dominated by the time it takes to find the vertices a_i, b_i, c_i in each nonempty H_i. This takes $O(n^3)$ time in total.

In order to show that Phase 3 runs in polynomial time, we need to show that each F_i has bounded size. This is true by the following claim proven below.

Claim 1. Every F_i is the disjoint union of at most 396 edges.

We prove this claim as follows. In order to derive a contradiction, suppose there is an F_i, say F_1, that is the disjoint union of edges $y_1z_1, \ldots, y_qz_q$ for some $q \geq 397$. By definition, each vertex of F_1 has a list of admissible colors $\{2, 3, 4\}$. This means that every vertex in F_1 must be adjacent to a vertex that received color 1 already. Note that such a vertex must be from the dominating set D that we precolored in Phase 1.

By Lemma 3, the size of a maximum independent set in $G[D]$ is at most $2 \cdot 4 + 4 = 12$. Since pairs of vertices with the same color are not adjacent, this implies that there are at most 12 vertices that received color 1 in Phase 1. Because $q \geq 397$, we then find that there exists a vertex $u \in D$ with color 1 that is adjacent to at least 34 vertices from the set $\{y_1, \ldots, y_q\}$. We assume without loss of generality that u is adjacent to $y_1, \ldots, y_{34}$. Because G is K_3-free, u is adjacent to no vertex of $\{z_1, \ldots, z_{34}\}$. Since there exist at most eleven other vertices with color 1, this means that there exists a vertex $v \in D$ with color 1 that is adjacent to at least four vertices in $\{z_1, \ldots, z_{34}\}$. We assume without loss of generality that v is adjacent to z_1, z_2, z_3, z_4. Because G is K_3-free, v is adjacent to no vertex of $\{y_1, y_2, y_3, y_4\}$. Since u and v both received color 1, they are not adjacent. However, then the paths y_1uy_2 and z_3vz_4 form an induced $2P_3$ in G. This is not possible, because G is $2P_3$-free. Hence, we have proven Claim 1.

We are left to determine the number of suitable list-assignments of G. This number equals the number of different precolorings of the vertices in $D \cup \{a_1, b_1, c_1\} \cup \{a_2, b_2, c_2\} \cup \{a_3, b_3, c_3\} \cup \{a_4, b_4, c_4\} \cup V(F_1) \cup V(F_2) \cup V(F_3) \cup V(F_4)$, which is at most $4^{15} \cdot (3^3)^4 \cdot (3^{792})^4$. This completes the proof of Lemma 5. □

Due to Lemma 4, our algorithm is left with the following task.

Check for each suitable list-assignment L' whether G has a coloring that respects L'.

Due to Lemma 5, our algorithm runs in polynomial time if it performs the above task in polynomial time for every suitable list-assignment. In Section 3.4 we consider a single suitable list-assignment L' of G and show that this is indeed the case.

3.4 Reducing the lists of size three in a suitable list-assignment

Let L' be a suitable list-assignment created from a $(K_3, 2P_3)$-free graph G with minimum degree at least 4. Recall that, due to the preprocessing, $V(G) = A \cup B_1 \cup B_2 \cup B_3 \cup B_4$ with the following four properties; also recall that $D \subseteq A$ is the dominating set that got precolored in Phase 1.

P1. $|L'(u)| \leq 2$ for every $u \in A$;

P2. $L'(v) = \{1,2,3,4\}\backslash\{i\}$ for every $v \in B_i$ and for every $1 \leq i \leq 4$;

P3. B_i is an independent set for every $1 \leq i \leq 4$;

P4. Every vertex of B_i is adjacent to at least one vertex of D that has color i.

Our algorithm now starts a branching procedure in order to reduce the number of admissible colors in the list of every vertex in each B_i by at least one, thereby enabling the use of Observation 1. This is described below.

Phase 4. The branching.
Our algorithm first considers B_1, then B_2, then B_3, and finally B_4 by applying the following procedure for each B_i. Recall that we use the notation $B_i(X) = N_G(X) \cap B_i$ and $B_i(\overline{X}) = B_i \backslash N_G(X)$.

(i) Determine a clique X in $G[V\backslash D]$ with $X = \{x\}$ or $X = \{x_1, x_2\}$ that pseudo-dominates B_i.

(ii) If $X = \{x\}$, then do as follows for every pair $p, q \in \{1,2,3,4\}\backslash\{i\}$ with $p \neq q$:
1. Set $L'(u) := \{p, q\}$ for every $u \in B_i(\{x\})$.
2. Remove all vertices in $B_i(\overline{\{x\}})$ with at most two neighbors in $V\backslash D$.
3. Precolor all remaining vertices in $B_i(\overline{\{x\}})$ respecting L'.
4. If $i \leq 3$, then start Phase 4 with B_{i+1}; otherwise apply Observation 1.

If the above branching does not lead to a coloring of G respecting L', then the algorithm does as follows. If $|B_i(\{x\})| \leq 2$ or $i \notin L'(x)$ it outputs `No`. Otherwise, the algorithm precolors x by i, removes x from G and repeats Phase 4 with set B_i.

(iii) If $X = \{x_1, x_2\}$, then do as follows for all 4-tuples (p, q, r, s) with $p, q, r, s \in \{1,2,3,4\}\backslash\{i\}$, $p \neq q$ and $r \neq s$:
1. Set $L'(u) := \{p, q\}$ for every $u \in B_i(\{x_1\})$.
2. Set $L'(v) := \{r, s\}$ for every $v \in B_i(\{x_2\})$.
3. Remove all vertices in $B_i(\overline{X})$ with at most two neighbors in $V\backslash D$.
4. Precolor all remaining vertices in $B_i(\overline{X})$ respecting L'.
5. If $i \leq 3$, then start Phase 4 with B_{i+1}; otherwise apply Observation 1.

If the above branching does not lead to a coloring of G respecting L', then the algorithm does as follows. If there is no $x \in X$ with $|B_i(x)| \geq 3$ and $i \in L'(x)$ then it outputs `No`. Otherwise, at least one of the vertices x_1, x_2 has three neighbors in B_i and color i in its list. If $|B_i(x_2)| \geq 3$ and $i \in L'(x_2)$ then the algorithm does as follows for every $p \in L'(x_1)\backslash\{i\}$:

6. Set $L'(x_1) = p$.
7. Set $L'(u) := L'(u) \backslash \{p\}$ for every $u \in N_G(x_1)$.
8. Set $L'(x_2) := \{i\}$.
9. Remove x_1, x_2 from G, set $B_i := B_i \backslash B_i(\{x_1\})$, and repeat Phase 4 with B_i.

If the above branching does not lead to a coloring of G respecting L', then the algorithm outputs `No` unless $|B(x_1)| \geq 3$ and $i \in L'(x_1)$. In that case, the algorithm does as follows for every $r \in L'(x_2) \backslash \{i\}$:

10. Set $L'(x_2) = r$.
11. Set $L'(v) := L'(v) \backslash \{r\}$ for every $v \in N_G(x_2)$.
12. Set $L'(x_1) := \{i\}$.
13. Remove x_1, x_2 from G, set $B_i := B_i \backslash B_i(\{x_2\})$, and repeat Phase 4 with B_i.

(iv) If all calls to Observation 1 in steps (ii) and (iii) yield no coloring, then the algorithm outputs `No`. Otherwise, if there is a call to Observation 1 that yields a coloring c, then the algorithm extends c to a coloring of G that respects L' by coloring the vertices it has removed from G in the reverse order of their removal, in such a way that L' is respected.

We prove the correctness of our branching algorithm in Lemma 6 and perform a running time analysis in Lemma 7.

Lemma 6. *Let L' be a suitable list-assignment of a $(K_3, 2P_3)$-free graph G with minimum degree at least four. Then G has a coloring respecting L' if and only if a coloring is produced in Phase 4.*

Proof. Let L' be a suitable list-assignment of a $(K_3, 2P_3)$-free graph G with minimum degree at least four. Let $V(G)$ be partitioned into the sets A, B_1, B_2, B_3, B_4 that satisfy properties P1–P4, where $D \subseteq A$ denotes the dominating set that got precolored in Phase 1. Below we show that G has a coloring respecting L' if and only if our algorithm produces a coloring in Phase 4.

Suppose G has a coloring c respecting L'. Because G is $(K_3, 2P_3)$-free, we can use Lemma 1. By this lemma, we are guaranteed to find a set X as described in Phase 4 (i). Note that X has indeed size at most two, because G is K_3-free. Suppose Phase 4 is performed on the set B_i. We must show that our algorithm branches in every possible way; in that case it will find a coloring respecting L', because G has at least one such coloring, namely c.

First consider the case $X = \{x\}$ for some $x \in V \backslash D$. Suppose the branching in operations 1–4 of step (ii) does not lead to a coloring of G respecting L'. Then we find that all three colors from $\{1, 2, 3, 4\} \backslash \{i\}$ must occur on $B_i(\{x\})$, and that consequently x must receive color i. Hence, we branch in every possible way.

Now consider the case $X = \{x_1, x_2\}$ for two vertices $x_1, x_2 \in V \backslash D$ with $x_1 \neq x_2$. Suppose the branching in operations 1–5 of step (iii) does not lead to a coloring of G respecting L'. Then we find that all three colors from $\{1, 2, 3, 4\} \backslash \{i\}$

must occur on $B_i(\{x_1\})$ or $B_i(\{x_2\})$. In the first case x_1 must have color i. In the second case x_2 must have color i. Because x_1 and x_2 are adjacent they cannot receive both color i. The algorithm first explores the case in which x_2 gets color i; this is done in operations 6–9 of step (iii). If this branching does not lead to a coloring of G respecting L', then it explores the case in which x_1 gets color i; this is done in operations 10–13 of step (iii). Hence, we branch in every possible way.

Now suppose our algorithm produces a coloring c. Then this coloring will respect L', because during Phase 4 the algorithm only removed colors from the lists assigned to the vertices by L'. However, the algorithm may have removed vertices from G, and such vertices did not get a color. In that case c is a coloring of a subgraph of G, and we must show how to extend c to a coloring of G. Let S be the set of vertices that have been removed. In step (iv) the algorithm considers the vertices in S in the reverse order of their removal. Let $u \in S$. At the moment u got removed, u had a list of three admissible colors and at most two neighbors in $V \setminus D$. Hence, there is always a color available to color u while respecting its list. This completes the proof of Lemma 6. □

Lemma 7. *Phase 4 runs in polynomial time.*

Proof. Let L' be a suitable list-assignment of a $(K_3, 2P_3)$-free graph G on n vertices with minimum degree at least four. Let $V(G)$ be partitioned into the sets A, B_1, B_2, B_3, B_4 that satisfy properties P1–P4, where $D \subseteq A$ denotes the dominating set that got precolored in Phase 1.

The algorithm performs step (i) in $O(n^3)$ time, because it has to find a set X of size at most two and then check whether X pseudo-dominates a set B_i. We observe that all operations in steps (ii) and (iii) can be done in polynomial time and that a call to Observation 1 takes polynomial time as well. Furthermore, if the algorithm finds a coloring then extending it to a coloring of the whole graph in step (iv) can be performed in polynomial time. Hence, we are left to show that the branching in steps (ii) and (iii) can be done in polynomial time, i.e., that the total number of calls to Observation 1 is bounded by a polynomial in n.

Suppose the algorithm is in Phase 4 and considers B_1. First assume that the set X determined in step (i) has size 1, say $X = \{x\}$ for some $x \in V \setminus D$; note that $x \in A \cup B_2 \cup B_3 \cup B_4$ by definition of X. Then our algorithm starts to branch as prescribed in step (ii). Let B_1^* denote the set of vertices in $B_1(\overline{\{x\}})$ that have at least three neighbors in $V \setminus D$. Note that these vertices get precolored by operation 3 of step (ii). We prove the following claim, which shows that B_1^* has bounded size.

Claim 1. B_1^* contains at most 48 vertices.

We prove this claim by contradiction. Suppose $|B_1^*| \geq 49$. By property P4, every vertex of B_1^* is adjacent to a vertex in D precolored 1. By Lemma 3, the size of a maximum independent set in $G[D]$ is at most $2 \cdot 4 + 4 = 12$. Since vertices with the same color are not adjacent, this implies that there are at most 12 vertices

that received color 1 in Phase 1. Because $|B_1^*| \geq 49$, we then find that there exists a vertex $u \in D$ with color 1 that is adjacent to at least five vertices from B_1^*. Let $y_1, \ldots, y_5$ denote these five vertices. By definition of B_1^*, every y_i has at least three neighbors a_i, b_i, c_i that are not in D. We recall that $B_1^* \subseteq B_1(\overline{\{x\}})$. Then, by definition of $B_1(\overline{\{x\}})$, we obtain that none of $\{a_i, b_i, c_i\}$ is adjacent to y_j whenever $j \neq i$.

We claim that the 9-vertex subgraph of G induced by $a_1, a_2, a_3, b_1, b_2, b_3$, c_1, c_2, c_3 contains an induced path P on three vertices. This can be seen as follows. Because G is K_3-free, the sets $\{a_1, a_2, a_3\}$, $\{b_1, b_2, b_3\}$ and $\{c_1, c_2, c_3\}$ are independent. Then there must exist a vertex in $\{a_1, b_1\}$ that is adjacent to a vertex in $\{a_2, b_2\}$. Otherwise, the paths $a_1y_1b_1$ and $a_2y_2b_2$ form an induced $2P_3$ in G, which is not possible because G is $2P_3$-free. We assume without loss of generality that $a_1a_2 \in E$. By applying the same arguments on the sets $\{b_1, c_1\}$ and $\{b_2, c_2\}$, we may also assume without loss of generality that $b_1b_2 \in E$. We now consider the pairs $\{a_1, b_1\}$ and $\{a_3, b_3\}$. By the previous arguments we find that there exists an edge between a vertex from $\{a_1, b_1\}$ and a vertex from $\{a_3, b_3\}$. Suppose without loss of generality that $a_1a_3 \in E$. Then $a_2a_1a_3$ is the desired path P; it is induced because G is K_3-free. Due to the K_3-freeness of G, no vertex of P is adjacent to u. Thus P and the path y_4uy_5 form an induced $2P_3$ of G. This is not possible, because G is $2P_3$-free. Hence, we have proven Claim 1.

Note that we have 3 possible choices for reducing the lists of the vertices in $B_i(\{x\})$ in operation 1 of step (ii). Then, due to Claim 1, we only have to consider $3 \cdot 3^{48}$ possible branches in step (ii). Every time we perform step (ii), we reduce the list of every vertex in B_1 by at least one color, thus afterwards the only vertices with a list of three admissible colors are the vertices in $B_2 \cup B_3 \cup B_4$.

Now assume that the set X determined in step (i) has size 2, say $X = \{x_1, x_2\}$ for some $x_1, x_2 \in A \cup B_2 \cup B_3 \cup B_4$. Then our algorithm starts to branch as prescribed in (iii). By using exactly the same arguments as in the proof of Claim 1, we find that the set $B_1^*(X)$ of vertices with at least three neighbors in $V \setminus D$ has size at most 48. Then we find that the total number of branches after performing operations 1–5 is $3 \cdot 3 \cdot 3^{48}$. Contrary to step (ii), we might also need to branch by performing 6–9 and 10–13 in every possible way. However, the following claim shows that we only need to branch for at most one set X of size 2 according to operations 6–9 *and* according to operations 10–13.

Claim 2. If the algorithm performs both operations 6–9 and operations 10–13 for $X = \{x_1, x_2\}$ when processing B_1, then it will not perform both operations 6–9 and operations 10–13 for any later set $X' = \{x_1', x_2'\}$ when it processes B_1 further.

We prove this claim as follows. Suppose the algorithm must perform both operations 6–9 and operations 10–13 for $X = \{x_1, x_2\}$. We assume that X is the first set for which the algorithm does this. In order to derive a contradiction suppose that, when it is processing B_1 further, the algorithm finds a new set X' containing two vertices x_1', x_2' for which it must also perform both operations 6–9 and operations 10–13.

Let $A_1 = B_1(\{x_1\})$ and $A_2 = B_1(\{x_2\})$ be the set of neighbors of x_1, x_2, respectively, in B_1. We assume without loss of generality that X' was computed in the branch that assigns color 1 to x_1. Let A'_1, A'_2 be the set of neighbors of x'_1, x'_2, respectively, in B_1. We will prove the following six properties.

(i) $A'_1 \subseteq B_1 \backslash A_2$ and $A'_2 \subseteq B_1 \backslash A_2$;

(ii) $|A_1| \geq 3$ and $|A_2| \geq 3$;

(iii) $|A'_1| \geq 3$ and $|A'_2| \geq 3$;

(iv) $x_1x'_1 \notin E(G)$ and $x_1x'_2 \notin E(G)$;

(v) $A_1 \cap A_2 = \emptyset$;

(vi) $A'_1 \cap A'_2 = \emptyset$;

(vii) $A'_1 \cup A'_2 \cup A_1 \cup A_2$ is an independent set.

Property (i) follows from our assumption that we consider the branch in which x_1 gets color 1. In that branch, B_1 gets reduced to $B_1 \backslash A_2$ in operation 13 of step (ii). Property (ii) holds, because the algorithm must perform both operations 6–9 and 10–13 for $\{x_1, x_2\}$. Property (iii) holds, because the algorithm must perform both operations 6–9 and 10–13 for $\{x'_1, x'_2\}$. By the same argument, we find that 1 must be a color in the list of x'_1 and x'_2. Consequently, x'_1 and x'_2 cannot be adjacent to x_1 which already received color 1. This shows that property (iv) is valid. Properties (v) and (vi) follow from the K_3-freeness of G and the fact that $x_1x_2 \in E(G)$ and $x'_1x'_2 \in E(G)$, respectively. Finally, property (vii) follows directly from P3 that states that the set B_1 at the start of Phase 4 is an independent set.

Because G is K_3-free and $x'_1x'_2$ is an edge, at least one of the two vertices in $\{x'_1, x'_2\}$, say x'_1, is not adjacent to x_2. We claim that x'_1 is adjacent to at least $|A_2| - 1$ vertices of A_2. In order to see this, suppose x'_1 is adjacent to at most $|A_2| - 2$ vertices in A_2. By (ii) we have $|A_2| \geq 3$. Hence, we find two different vertices $a, a^* \in A_2$ with $ax'_1 \notin E(G)$ and $a^*x'_1 \notin E(G)$. By (iii) we have $|A'_1| \geq 3 \geq 2$. Hence, there exist two different vertices $b, b^* \in A'_1$. By (i) we have $A'_1 \subseteq B_1 \backslash A_2$. Then, x_2 is neither adjacent to b nor to b^*. By (vii) we find that $\{a, a^*, b, b^*\}$ is an independent set. However, then bx'_1b^* and ax_2a^* form an induced $2P_3$ in G. This is not possible, because G is $2P_3$-free. We conclude that x'_1 is adjacent to at least $|A_2| - 1$ vertices of A_2. Then, because $|A_2| \geq 3 \geq 2$ by (ii), there exist two vertices $c, c^* \in A_2$ with $cx'_1 \in E$ and $c^*x'_1 \in E$.

We now show that $|A_1 \cap A'_2| \geq 2$. To see this, suppose that $|A_1 \cap A'_2| \leq 1$. Then, because $|A_1| \geq 3$ due to (ii) and $|A'_2| \geq 3$ due to (iii), there exist two vertices $s_1, s_2 \in A_1 \backslash A'_2$ and two vertices $t_1, t_2 \in A'_2 \backslash A_1$. By (iv), we have that x_1 and x'_2 are not adjacent. By (vii), we find that $\{s_1, s_2, t_1, t_2\}$ is an independent set. Consequently, G contains two paths, namely $s_1x_1s_2$ and $t_1x'_2t_2$, that form an induced $2P_3$. This is not possible, because G is $2P_3$-free. Hence, we find that A_1 and A'_2 have at least two common vertices. Denote these vertices by d, d^*.

By (v) we have $A_1 \cap A_2 = \emptyset$, so x_1 has no neighbor in A_2. By (vi) we have $A'_1 \cap A'_2 = \emptyset$, so x'_1 has no neighbor in A'_2. We conclude that $\{c, c^*, d, d^*\}$ is a set

of 4 vertices. By (vii), this set is independent. By (iv) we have that $x_1x'_1$ is not an edge. However, then cx'_1c^* and dx_1d^* form an induced $2P_3$ in G. This is not possible, because G is $2P_3$-free. Hence, we have completed the proof of Claim 2.

Indeed, due to Claim 2, our algorithm will never perform both operations 6-9 and operations 10-13 in step (ii) for more than one set X of size 2. Note that each time our algorithm performs step (ii) or (iii) the size of G is decreased by at least one vertex, because we remove the vertices in X from G. We conclude that our algorithm creates at most $2 \cdot (6 \cdot 6 \cdot 3^{48} + 3)|B_1| = O(|B_1|)$ list-assignments of G that each assign only lists of size 3 to vertices in $B_2 \cup B_3 \cup B_4$ and that each can be considered as separate inputs for the algorithm when it starts to run Phase 4 for B_2.

For B_2, and also for B_3 and B_4, we follow the same reasoning. This means that the total number of calls to Observation 1 is $O(|B_1||B_2||B_3||B_4|) = O(n^4)$, which is polynomial, as desired. This completes the proof of Lemma 7. □

By Lemmas 4–7 we immediately obtain the following result.

Theorem 3. *The* 4-Coloring *problem can be solved in polynomial time for the class of $(K_3, 2P_3)$-free graphs.*

Remark. A reader might have noticed that we are quite generous with respect to the running time of our algorithm. Indeed, there are several ways to make our algorithm faster, e.g., by showing that the size of the dominating sets constructed by the algorithm can be reduced, or by replacing Phase 3 by some extra branching steps in Phase 4, and so on. We decided not to introduce these extra technicalities for the following two reasons. Firstly, our main motivation is to show polynomial-time solvability. Secondly, we believe that the extra adjustments that are necessary to decrease the running time distract from the underlying key ideas behind our algorithm.

4 Determining the chromatic number

We present a polynomial-time algorithm that solves the Vertex Coloring problem for $(K_3, 2P_3)$-free graphs. We need the following theorem before we are able to present our main result, Theorem 5.

Theorem 4. *Every $(K_3, 2P_3)$-free graph can be colored with at most 5 colors.*

Proof. Let G be a $(K_3, 2P_3)$-free graph. We may assume that G is connected and that $|V(G)| \geq 3$. This implies that G contains an induced P_3, say on vertex set $P = \{v_1, v_2, v_3\}$, where v_2 is the vertex with degree 2 in the induced P_3. By the assumptions, every component of the subgraph of G induced by $V(G) \setminus (P \cup N(P))$ is either a vertex or an edge; we denote the set of components isomorphic to K_1 by I and the set of components isomorphic to K_2 by M.

Consider the following assignment of colors from $\{1, 2, 3, 4, 5\}$ to the vertices of G: assign color 1 to v_1, v_3 and all the (other) neighbors of v_2, assign color 2

to v_2 and all the (other) neighbors of v_1, and assign color 3 to all the uncolored neighbors of v_3. Because G is K_3-free, this is a 3-coloring of the subgraph of G induced by $P \cup N(P)$. This 3-coloring can be extended to a 5-coloring of G by assigning color 4 to all the vertices of I and colors 4 and 5 to all the pairs of adjacent vertices of the edges of M. □

Theorem 5. *The* Vertex Coloring *problem can be solved in polynomial time for the class of* $(K_3, 2P_3)$*-free graphs.*

Proof. Let G be a $(K_3, 2P_3)$-free graph. Our algorithm checks whether G is k-colorable for increasing value of k from 1 up to 4. If in the end no coloring of G has been found then $\chi_G = 5$ is outputted. The correctness of this algorithm immediately follows from Theorem 4. Below we show that it runs in polynomial time.

We first note that a graph can be colored with at most one color if and only if it consists of isolated vertices only. Secondly, a graph can be colored with at most two colors if and only if it is bipartite. By Theorem 1 and 3, we can test in polynomial time whether G is 3-colorable or 4-colorable, respectively. Hence, we conclude that our algorithm runs in polynomial time. □

5 Future research

One can explore various directions to extend the polynomial-time results in this paper, and determining the complexity of the following problems is still open.

1. 4-Coloring for (K_3, sP_3)-free graphs for any fixed $s \geq 3$;
2. 3-Coloring for (K_3, P_7)-free graphs.

With respect to problem 1, we note that we can solve the 3-Coloring problem for sP_3-free graphs for any fixed $s \geq 1$ [5]. Furthermore, Dabrowski et al. [8] combined results from Balas and Yu [1], Brandt [3], and Tsukiyama et al. [24] to show that Vertex Coloring is polynomial-time solvable for the class of (K_3, sK_2)-free graphs for any fixed $s \geq 1$.

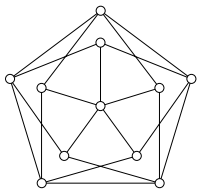

Fig. 2. the Grötzsch graph.

Another interesting research direction is to characterize the class of $(K_3, 2P_3)$-free graphs with chromatic number 4 or 5, respectively. Such a characterization

could lead to a certifying algorithm, just as Bruce, Hoàng, and Sawada [6] successfully show for 3-Colorability of P_5-free graphs. An infinite class of 4-chromatic $(K_3, 2P_3)$-free graphs can for example be obtained from the Grötzsch graph (see Figure 2) by replacing the unique vertex of degree 5 by a set of vertices, all adjacent to its five neighbors. We have no examples of $(K_3, 2P_3)$-free graphs with chromatic number 5. We note that even when such graphs do no exist at all our polynomial-time algorithm in Section 3 for solving 4-Coloring is still useful, because it produces a 4-coloring of a 4-chromatic $(K_3, 2P_3)$-free graph. Furthermore, we expect that the techniques applied to this algorithm are useful for solving problem 1.

Acknowledgments. We thank the anonymous referees for their 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. S. Brandt, Triangle-free graphs and forbidden subgraphs, Discrete Appl. Math. 120, 25–33 (2002).
4. H.J. Broersma, F.V. Fomin, P.A. Golovach and D. Paulusma, Three complexity results on coloring P_k-free graphs, Proceedings of the 20th International Workshop on Combinatorial Algorithms (IWOCA 2009), Lecture Notes in Computer Science 5874, 95–104 (2009).
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 the 20th International Symposium on Algorithms and Computation (ISAAC 2009), Lecture Notes in Computer Science 5878, 594-604 (2009).
7. B. Courcelle, J.A. Makowsky and U. Rotics, Linear time solvable optimization problems on graphs of bounded clique-width, Theory Comput. Systems 33, 125–150 (2000).
8. K. Dabrowski, V. Lozin, R. Raman and B. Ries, Colouring vertices of triangle-free graphs, Proceedings of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2010), Lecture Notes in Computer Science 6410,184–195 (2010).
9. K. Edwards, The complexity of coloring problems on dense graphs, Theoret. Comput. Science 43, 337–343 (1986).
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, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1, 169–197 (1981).
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. 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).
14. M. Kamiński and V.V. Lozin, Vertex 3-colorability of Claw-free Graphs. Algorithmic Operations Research 21, (2007).
15. M. Kochol, V.V. Lozin and B. Randerath, The 3-Colorability Problem on Graphs with Maximum Degree Four, SIAM J. Comput. 32, 1128–1139 (2003).
16. D. Král', J. Kratochvíl, Zs. Tuza, and G.J. Woeginger, Complexity of coloring graphs without forbidden induced subgraphs, Proceedings of the 27th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2001), Lecture Notes in Computer Science 2204, 254–262 (2001).
17. J. Kratochvíl, Precoloring extension with fixed color bound, Acta Math. Univ. Comen. 62, 139–153 (1993).
18. V.B. Le, B. Randerath and I. Schiermeyer, On the complexity of 4-coloring graphs without long induced paths, Theor. Comp. Science 389, 330–335 (2007).
19. V. Lozin and R. Mosca, Independent sets in extensions of $2K_2$-free graphs, Discrete Appl. Math. 146, 74–80 (2005).
20. V. Lozin and J. Volz, The clique-width of bipartite graphs in monogenic classes, Inter- national Journal of Foundations of Computer Science 19, 477494 (2008).
21. F. Maffray and M. Preissmann, On the NP-completeness of the k-colorability problem for triangle-free graphs, Discrete Math. 162, 313–317 (1996).
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 and Combin. 20, 1–40 (2004).
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 constraints - 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 Cybern. 15, 107–117 (2001).