



## Independent Feedback Vertex Set for $P_5$ -Free Graphs

Marthe Bonamy<sup>1</sup> · Konrad K. Dabrowski<sup>2</sup> · Carl Feghali<sup>3</sup> ·  
Matthew Johnson<sup>2</sup> · Daniël Paulusma<sup>2</sup>

Received: 29 September 2017 / Accepted: 19 June 2018 / Published online: 26 June 2018  
© The Author(s) 2018

### Abstract

The NP-complete problem FEEDBACK VERTEX SET is that of deciding whether or not it is possible, for a given integer  $k \geq 0$ , to delete at most  $k$  vertices from a given graph so that what remains is a forest. The variant in which the deleted vertices must form an independent set is called INDEPENDENT FEEDBACK VERTEX SET and is also NP-complete. In fact, even deciding if an independent feedback vertex set exists is NP-complete and this problem is closely related to the 3-COLOURING problem, or equivalently, to the problem of deciding whether or not a graph has an independent odd cycle transversal, that is, an independent set of vertices whose deletion makes the graph bipartite. We initiate a systematic study of the complexity of INDEPENDENT FEEDBACK VERTEX SET for  $H$ -free graphs. We prove that it is NP-complete if  $H$  contains a claw or cycle. Tamura, Ito and Zhou proved that it is polynomial-time solvable for  $P_4$ -free graphs. We show that it remains polynomial-time solvable for  $P_5$ -free graphs. We prove analogous results for the INDEPENDENT ODD CYCLE TRANSVERSAL problem, which asks whether or not a graph has an independent odd cycle transversal of size at most  $k$  for a given integer  $k \geq 0$ . Finally, in line with our underlying research aim, we compare the complexity of INDEPENDENT FEEDBACK VERTEX SET for  $H$ -free graphs with the complexity of 3-COLOURING, INDEPENDENT ODD CYCLE TRANSVERSAL and other related problems.

**Keywords** Independent feedback vertex set · Near-bipartiteness · Independent odd cycle transversal · 3-Colouring

---

This paper received support from EPSRC (EP/K025090/1), London Mathematical Society (41536), the Leverhulme Trust (RPG-2016-258) and Fondation Sciences Mathématiques de Paris. An extended abstract of this paper appeared in the proceedings of ISAAC 2017 [4].

Extended author information available on the last page of the article

## 1 Introduction

Many computational problems in the theory and application of graphs can be formulated as modification problems: from a graph  $G$ , some other graph  $H$  with a desired property must be obtained using certain permitted operations. The number of graph operations used (or some other measure of cost) must be minimised. The computational complexity of a graph modification problem depends on the desired property, the operations allowed and the possible inputs; that is, we can prescribe the class of graphs to which  $G$  must belong. This leads to a rich variety of different problems, which makes graph modification a central area of research in algorithmic graph theory.

A set  $S$  of vertices in a graph  $G$  is a *feedback vertex set* of  $G$  if removing the vertices of  $S$  results in an acyclic graph, that is, the graph  $G - S$  is a forest. The FEEDBACK VERTEX SET problem asks whether or not a graph has a feedback vertex set of size at most  $k$  for some integer  $k \geq 0$  and is a well-known example of a graph modification problem: the desired property is that the obtained graph is acyclic and the permitted operation is vertex deletion. The directed variant of FEEDBACK VERTEX SET was one of the original problems proven to be NP-complete by Karp [26]. The proof of this implies NP-completeness of the undirected version even for graphs of maximum degree 4 (see [17]). We refer to the survey of Festa et al. [16] for further details of this classic problem.

In this paper, we consider the problem where we require the feedback vertex set to be an independent set. We call such a set an *independent feedback vertex set*. We have the following decision problem.

INDEPENDENT FEEDBACK VERTEX SET

*Instance:* a graph  $G$  and an integer  $k \geq 0$ .

*Question:* does  $G$  have an independent feedback vertex set of size at most  $k$ ?

Many other graph problems have variants with an additional constraint that a set of vertices must be independent. For example, see [18] for a survey on INDEPENDENT DOMINATING SET, and [33] for INDEPENDENT ODD CYCLE TRANSVERSAL, also known as STABLE BIPARTIZATION. An *independent odd cycle transversal* of a graph  $G$  is an independent set  $S$  such that  $G - S$  is bipartite, and the latter problem is that of deciding whether or not a graph has such a set of size at most  $k$  for some given integer  $k$ .

We survey known results on INDEPENDENT FEEDBACK VERTEX SET below.

### 1.1 Related Work

Not every graph admits an independent feedback vertex set (consider complete graphs on at least four vertices). Graphs that do admit an independent feedback vertex set are said to be *near-bipartite*, and we can ask about the decision problem of recognising such graphs.

NEAR- BIPARTITENESS

*Instance:* a graph  $G$ .

*Question:* is  $G$  near-bipartite (that is, does  $G$  have an independent feedback vertex set)?

NEAR- BIPARTITENESS is NP-complete even for graphs of maximum degree 4 [44] and for graphs of diameter 3 [6]. Hence, by setting  $k = n$ , we find that INDEPENDENT FEEDBACK VERTEX SET is NP-complete for these two graph classes. The INDEPENDENT FEEDBACK VERTEX SET problem is even NP-complete for planar bipartite graphs of maximum degree 4 (see [43]). As bipartite graphs are near-bipartite, this result shows that there are classes of graphs where INDEPENDENT FEEDBACK VERTEX SET is harder than NEAR- BIPARTITENESS. To obtain tractability results for INDEPENDENT FEEDBACK VERTEX SET, we need to make some further assumptions.

One way is to consider the problem from a parameterized point of view. Taking  $k$  as the parameter, Misra et al. [35] proved that INDEPENDENT FEEDBACK VERTEX SET is fixed-parameter tractable by giving a cubic kernel. This is in line with the fixed-parameter tractability of the general FEEDBACK VERTEX SET problem (see [27] for the fastest known FPT algorithm). Later, Agrawal et al. [1] gave a faster FPT algorithm for INDEPENDENT FEEDBACK VERTEX SET and also obtained an upper bound on the number of minimal independent feedback vertex sets of a graph.

Another way to obtain tractability results is to restrict the input to special graph classes in order to determine graph properties that make the problem polynomial-time solvable. We already mentioned some classes for which INDEPENDENT FEEDBACK VERTEX SET is NP-complete. In a companion paper [6], we show that the problem is polynomial-time solvable for graphs of diameter 2, and as stated above, the problem is NP-complete on graphs of diameter 3. Tamura et al. [43] showed that INDEPENDENT FEEDBACK VERTEX SET is polynomial-time solvable for chordal graphs, graphs of bounded treewidth and for cographs. The latter graphs are also known as  $P_4$ -free graphs ( $P_r$  denotes the path on  $r$  vertices and a graph is  $H$ -free if it has no induced subgraph isomorphic to  $H$ ), and this strengthened a result of Brandstädt et al. [8], who proved that NEAR- BIPARTITENESS is polynomial-time solvable for  $P_4$ -free graphs.

## 1.2 Our Contribution

The INDEPENDENT FEEDBACK VERTEX SET problem is equivalent to asking for a (proper) 3-colouring of a graph, such that one colour class has at most  $k$  vertices and the union of the other two induces a forest. We wish to compare the behaviour of INDEPENDENT FEEDBACK VERTEX SET with that of the 3- COLOURING problem. It is well known that the latter problem is also NP-complete [31] in general and polynomial-time solvable on many graph classes (see, for instance, the surveys [20,40]). We also observe that 3- COLOURING is equivalent to asking whether or not a graph has an independent odd cycle transversal (of any size). However, so far very few graph classes are known for which INDEPENDENT FEEDBACK VERTEX SET is tractable and our goal is to find more of them. For this purpose, we consider  $H$ -free graphs and extend the result [43] for  $P_4$ -free graphs in a systematic way.

In Sect. 2, we consider the cases where  $H$  contains a cycle or a claw. We first prove that NEAR- BIPARTITENESS, and thus INDEPENDENT FEEDBACK VERTEX SET, is NP-complete on line graphs, which form a subclass of the class of claw-free graphs. We then prove that INDEPENDENT FEEDBACK VERTEX SET is NP-complete for graphs of arbitrarily large girth. Together, these results imply that INDEPENDENT FEEDBACK



Fig. 1 The paths on four and five vertices

VERTEX SET is NP-complete for  $H$ -free graphs if  $H$  contains a cycle or claw. Hence, only the cases where  $H$  is a *linear forest*, that is, a disjoint union of paths, remain open. In particular, the case where  $H$  is a single path has not yet been resolved. Due to the result of [43] for  $P_4$ -free graphs, the first open case to consider is when  $H = P_5$  (see also Fig. 1).

The class of  $P_5$ -free graphs is a well-studied graph class. For instance, Hoàng et al. [24] proved that for every integer  $k$ ,  $k$ - COLOURING is polynomial-time solvable for  $P_5$ -free graphs, whereas Golovach and Heggeres [19] showed that CHOOSABILITY is fixed-parameter tractable for  $P_5$ -free graphs when parameterized by the size of the lists of admissible colours. Lokshantov et al. [30] solved a long-standing open problem by giving a polynomial-time algorithm for INDEPENDENT SET restricted to  $P_5$ -free graphs (recently, their result was extended to  $P_6$ -free graphs by Grzesik et al. [22]).

Our main result is that INDEPENDENT FEEDBACK VERTEX SET is polynomial-time solvable for  $P_5$ -free graphs. This is proved in Sects. 3 and 4: in Sect. 3 we give a polynomial-time algorithm for NEAR- BIPARTITENESS on  $P_5$ -free graphs, and in Sect. 4 we show how to *extend* this algorithm to solve INDEPENDENT FEEDBACK VERTEX SET in polynomial time for  $P_5$ -free graphs.

In Sect. 5 we consider the related problem INDEPENDENT ODD CYCLE TRANSVERSAL. We prove that our results for INDEPENDENT FEEDBACK VERTEX SET also hold for INDEPENDENT ODD CYCLE TRANSVERSAL.

In Sect. 6, we compare the complexities of INDEPENDENT FEEDBACK VERTEX SET and INDEPENDENT ODD CYCLE TRANSVERSAL for  $H$ -free graphs with the complexity of 3- COLOURING and several other related problems, such as FEEDBACK VERTEX SET, VERTEX COVER, INDEPENDENT VERTEX COVER and DOMINATING INDUCED MATCHING. We also survey some related open problems.

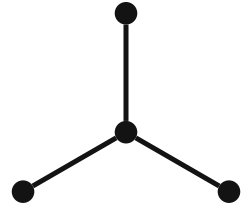
## 2 Hardness When $H$ Contains a Cycle or Claw

Before stating the results in this section, we first introduce some necessary terminology. The *line graph*  $L(G)$  of a graph  $G = (V, E)$  has the edge set  $E$  of  $G$  as its vertex set, and two vertices  $e_1$  and  $e_2$  of  $L(G)$  are adjacent if and only if  $e_1$  and  $e_2$  share a common end-vertex in  $G$ . The *claw* is the graph shown in Fig. 2. It is well known and easy to see that every line graph is claw-free. A graph is (*sub*)*cubic* if every vertex has (at most) degree 3.

We first prove that NEAR- BIPARTITENESS is NP-complete for line graphs. It was already known that FEEDBACK VERTEX SET is NP-complete for line graphs of planar cubic bipartite graphs [37].

**Theorem 1** NEAR- BIPARTITENESS is NP-complete for line graphs of planar subcubic bipartite graphs.

Fig. 2 The claw



**Proof** As the problem is readily seen to be in NP, it suffices to prove NP-hardness. We reduce from the HAMILTON CYCLE THROUGH SPECIFIED EDGE problem. Given a graph  $G$  and an edge  $e$  of  $G$ , this problem asks whether  $G$  has a Hamilton cycle through  $e$ . Labarre [29] observed that HAMILTON CYCLE THROUGH SPECIFIED EDGE is NP-complete for planar cubic bipartite graphs by noting that it follows easily from the analogous result of Akiyama et al. [2] for HAMILTON CYCLE. Let  $(G, e)$  be an instance of HAMILTON CYCLE THROUGH SPECIFIED EDGE. By the aforementioned result, we may assume that  $G$  is a planar cubic bipartite graph. Let  $u_1$  and  $u_2$  be the two end-vertices of  $e$ . Delete the edge  $u_1u_2$  and add two new vertices  $v_1$  and  $v_2$ , and two new edges  $e_1 = u_1v_1$  and  $e_2 = v_2u_2$ . Let  $G'$  be the resulting graph. We note that both  $v_1$  and  $v_2$  have degree 1 in  $G'$ , whereas every other vertex of  $G'$  has degree 3. Hence  $G'$  is subcubic. Since  $G$  is planar and bipartite, it follows that  $G'$  is planar and bipartite. Moreover, we make the following observation.

**Claim 1** *The graph  $G$  has a Hamilton cycle through  $e$  if and only if  $G'$  has a Hamilton path from  $v_1$  to  $v_2$ .*

Let  $n$  be the number of vertices in  $G$ . Then the number of vertices in  $G'$  is  $n + 2$ , meaning that a Hamilton path in  $G'$  has  $n + 1$  edges. Moreover, as  $G$  is cubic,  $G$  has  $\frac{3}{2}n$  edges, so  $G'$  has  $\frac{3}{2}n + 1$  edges, implying that  $L(G')$  has  $\frac{3}{2}n + 1$  vertices. Furthermore,  $e_1$  and  $e_2$  have degree 2 in  $L(G')$  and all other vertices in  $L(G')$  have degree 4, so  $L(G')$  has maximum degree at most 4.

**Claim 2** *The graph  $G'$  has a Hamilton path from  $v_1$  to  $v_2$  if and only if  $L(G')$  is near-bipartite.*

To prove Claim 2, first suppose that  $G'$  has a Hamilton path  $P$  from  $v_1$  to  $v_2$ . Then, as every vertex in  $G'$  apart from  $v_1$  and  $v_2$  has degree 3, it follows that  $G' - E(P)$  consists of two isolated vertices ( $v_1$  and  $v_2$ ) and a set  $S$  of isolated edges. Thus  $S$  is an independent set in  $L(G')$ , and so  $L(G')$  is near-bipartite.

Now suppose that  $L(G')$  is near-bipartite. Then  $L(G')$  contains a set  $S$  of vertices, such that  $F = L(G') - S$  is a forest. As  $L(G')$  is a line graph,  $L(G')$  is claw-free. This means that  $F$  is the disjoint union of one or more paths. Suppose that  $F$  contains more than one path. Then, as  $e_1$  and  $e_2$  are the only two vertices in  $L(G')$  that are of degree 2 and all other vertices of  $L(G')$  have degree 4, at least one path of  $F$  has an end-vertex of degree 4 in  $L(G')$ . Let  $f$  be this vertex. As  $f$  is the end-vertex of a path in  $F$ , we find that  $f$  has three neighbours  $f_1, f_2, f_3$  in  $S$ . As  $S$  is an independent set,  $\{f, f_1, f_2, f_3\}$  induces a claw in  $L(G')$ . This contradiction tells us that  $F$  consists of exactly one path  $P$ , and by the same reasoning,  $e_1$  and  $e_2$  must be the end-vertices of  $P$ .

Hence every vertex of  $P$  apart from its end-vertices has two neighbours in  $S$  and every vertex in  $S$  has four neighbours on  $P$ . Moreover,  $e_1$  and  $e_2$  have exactly one neighbour in  $S$ . This means that  $1 + 1 + 2(|V(P)| - 2) = 4|S|$ , so  $|S| = \frac{1}{2}(|V(P)| - 1)$ . Hence we find that

$$\begin{aligned} \frac{3}{2}(|V(P)| - 1) + 1 &= |V(P)| + \frac{1}{2}(|V(P)| - 1) \\ &= |V(P)| + |S| \\ &= |L(G')| \\ &= \frac{3}{2}n + 1, \end{aligned}$$

so  $|V(P)| = n + 1$ . Hence, as  $G'$  has  $n + 2$  vertices, the pre-image of  $P$  in  $G'$  is a Hamilton path of  $G'$  with end-vertices  $v_1$  and  $v_2$ .

By combining Claims 1 and 2 we have completed our hardness reduction and the theorem is proved. □

Theorem 1 has the following immediate consequence (take  $k = n$ ).

**Corollary 1** INDEPENDENT FEEDBACK VERTEX SET is NP-complete for line graphs of planar subcubic bipartite graphs.

We will now prove that INDEPENDENT FEEDBACK VERTEX SET is NP-complete for graphs with no small cycles even if their maximum degree is small. The length of a cycle  $C$  is the number of edges of  $C$ . The girth  $g(G)$  of a graph  $G$  is the length of a shortest cycle of  $G$ ; if  $G$  has no cycles then  $g(G) = \infty$ . The subdivision of an edge  $e = uv$  in a graph deletes  $e$  and adds a new vertex  $w$  and edges  $uw$  and  $wv$ . We first need the following observation, which is well known. For completeness we give a short proof.

**Lemma 1** (see e.g. [35]) Let  $uv$  be an edge in a graph  $G$ . Let  $G'$  be the graph obtained from  $G$  after subdividing  $uv$ . Then  $G$  has a feedback vertex set of size at most  $k$  if and only if  $G'$  does.

**Proof** Let  $w$  denote the new vertex obtained from subdividing  $uv$ . Any feedback vertex set  $S$  of  $G$  is a feedback vertex set of  $G'$ . Suppose  $S'$  is a feedback vertex set of  $G'$ . If  $w \notin S'$ , then  $S'$  is a feedback vertex set of  $G$ . Suppose  $w \in S'$ . If at least one of  $u$  and  $v$  are in  $S'$  as well, then  $S' \setminus \{w\}$  is a feedback vertex set of  $G$ . If neither  $u$  nor  $v$  belong to  $S'$ , then  $(S' \setminus \{w\}) \cup \{u\}$  is a feedback vertex set of  $G$  with the same size as  $S'$ . □

Lemma 1 implies that FEEDBACK VERTEX SET is NP-complete for graphs of girth at least  $g$  for every constant  $g \geq 3$ . We also use this lemma to prove our next result.

**Proposition 1** For every constant  $g \geq 3$ , INDEPENDENT FEEDBACK VERTEX SET is NP-complete for graphs of maximum degree at most 4 and girth at least  $g$ .

**Proof** For a graph  $G$ , let  $G_s$  be the graph obtained from  $G$  after subdividing every edge of  $G$ ; we say that  $G_s$  is a subdivided copy of  $G$ . Let  $\mathcal{G}_s$  be the graph class obtained from

a graph class  $\mathcal{G}$  after replacing each  $G \in \mathcal{G}$  by its subdivided copy  $G_s$ . It follows from Lemma 1 that if FEEDBACK VERTEX SET is NP-complete for some graph class  $\mathcal{G}$ , then it is also NP-complete for  $\mathcal{G}_s$ . By starting from the fact that FEEDBACK VERTEX SET is NP-complete for line graphs of planar cubic bipartite graphs [37] and applying this observation a sufficient number of times, we find that for any constant  $g \geq 3$ , FEEDBACK VERTEX SET is NP-complete for graphs of maximum degree at most 4 and girth at least  $g$ . Moreover, any non-independent feedback vertex set  $S$  of a subdivided copy  $G_s$  of a graph  $G$  contains two adjacent vertices, one of which has degree 2 in  $G_s$ . Hence, we can remove such a degree 2 vertex from  $S$  to obtain a smaller feedback vertex set of  $G_s$ . Thus all minimum feedback vertex sets of  $G_s$  are independent. This observation, which can also be found in [35], together with NP-hardness for FEEDBACK VERTEX SET for graphs of maximum degree at most 4 and with arbitrarily large girth proves the proposition.  $\square$

Recall that every line graph is claw-free. We also observe that for a graph  $H$  with a cycle  $C$ , the class of graphs of girth at least  $|C| + 1$  is a subclass of the class of  $H$ -free graphs. Hence, we can combine Corollary 1 and Proposition 1 to obtain the following result.

**Corollary 2** *Let  $H$  be a graph that contains a claw or a cycle. Then INDEPENDENT FEEDBACK VERTEX SET is NP-complete for  $H$ -free graphs of maximum degree at most 4.*

### 3 Near-Bipartiteness of $P_5$ -Free Graphs

In this section, we show that NEAR-BIPARTITENESS is polynomial-time solvable for  $P_5$ -free graphs, that is, we give a polynomial-time algorithm for testing whether or not a  $P_5$ -free graph has an independent feedback vertex set. To obtain a *minimum* feedback vertex set we need to first run this algorithm and then do the additional work described in Sect. 4.

Our algorithm in this section solves a slightly more general problem, which is a special variant of LIST 3-COLOURING. In the LIST 3-COLOURING problem each vertex  $v$  is assigned a subset  $L(v)$  of colours from  $\{1, 2, 3\}$  and we must verify whether or not a 3-colouring exists in which each vertex  $v$  is coloured with a colour from  $L(v)$ . We say that a 3-colouring of a graph  $G$  is *semi-acyclic* if the vertices coloured 2 or 3 induce a forest, and we note that  $G$  has such a colouring if and only if  $G$  is near-bipartite. This leads to the following variant of LIST 3-COLOURING.

LIST SEMI-ACYCLIC 3-COLOURING

*Instance:* a graph  $G$  and a function  $L : V(G) \rightarrow \{S \mid S \subseteq \{1, 2, 3\}\}$ .

*Question:* does  $G$  have a semi-acyclic 3-colouring  $c$  such that  $c(v) \in L(v)$  for all  $v \in V(G)$ ?

A graph  $G$  is near-bipartite if and only if  $(G, L)$ , with  $L(v) = \{1, 2, 3\}$  for all  $v \in V(G)$ , is a *yes*-instance of LIST SEMI-ACYCLIC 3-COLOURING. To show that near-bipartite  $P_5$ -free graphs can be recognised in polynomial time, we will prove the



stronger fact that LIST SEMI-ACYCLIC 3-COLOURING is polynomial-time solvable for  $P_5$ -free graphs.

A set of vertices in a graph  $G$  is *dominating* if every vertex of  $G$  is either in the set or has at least one neighbour in it. We will use a lemma of Bacsó and Tuza.

**Lemma 2** [3] *Every connected  $P_5$ -free graph admits a dominating set that induces either a clique or a  $P_3$ .*

Lemma 2 implies that every connected 3-colourable  $P_5$ -free graph has a dominating set of size at most 3 (since it has no clique on more than three vertices). This was used by Randerath et al. [41] to show that 3-COLOURING is polynomial-time solvable on  $P_5$ -free graphs. Their algorithm tries all possible 3-colourings of a dominating set of size at most 3. It then adjusts the lists of the other vertices (which were originally set to  $\{1, 2, 3\}$ ) to lists of size at most 2. As shown by Edwards [14], 2-LIST COLOURING can be translated to an instance of 2-SATISFIABILITY, which is well known and readily seen to be solvable in linear time. Hence this approach results in a polynomial (even constant) number of instances of the 2-SATISFIABILITY problem.

Our goal is also to apply Lemma 2 to a connected  $P_5$ -free graph  $G$  and to reduce an instance  $(G, L)$  of LIST SEMI-ACYCLIC 3-COLOURING to a polynomial number of instances of 2-SATISFIABILITY. However, in our case this is less straightforward than in the case of 3-COLOURING restricted to  $P_5$ -free graphs: the restriction of LIST SEMI-ACYCLIC 3-COLOURING to lists of size 2 turns out to be NP-complete for general graphs even if every list consists of either colours 1 and 3 or only colour 2.

**Theorem 2** LIST SEMI-ACYCLIC 3-COLOURING is NP-complete even if  $L(v) \in \{\{1, 3\}, \{2\}\}$  for every vertex  $v$  in the input graph.

**Proof** The problem is clearly in NP so we need only show that it is NP-hard. We do this by reduction from SATISFIABILITY.

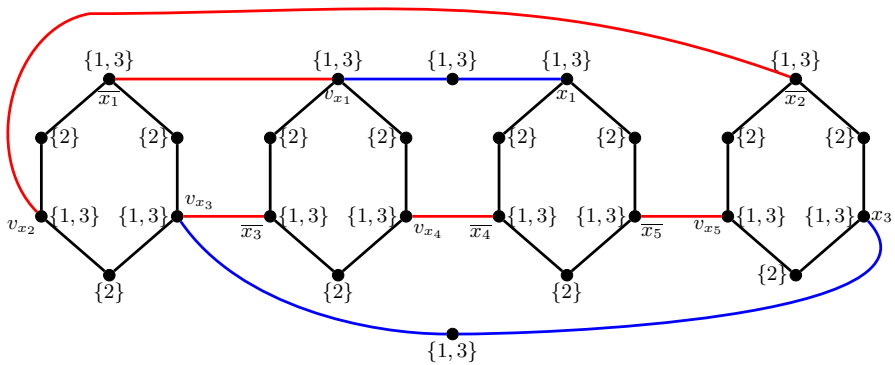
Let  $\phi$  be an instance of SATISFIABILITY. Note that we can assume that, for each variable  $x$ ,  $\phi$  contains both the literals  $x$  and  $\bar{x}$ , and that each clause contains more than one literal (otherwise in polynomial-time we can obtain another smaller instance whose satisfiability is the same as that of  $\phi$ ). We create an instance  $(G, L)$  of LIST SEMI-ACYCLIC 3-COLOURING as follows (see also Fig. 3):

- For each clause  $C$  of  $\phi$  create a  $(2|C|)$ -cycle and assign lists  $\{1, 3\}$  and  $\{2\}$  alternately to vertices around the cycle. Let the literals of the clause be represented by distinct vertices with lists  $\{1, 3\}$ .
- For each variable  $x$ , choose a clause containing the positive literal  $x$  and let  $v_x$  be the vertex representing  $x$  in the corresponding cycle. For every other occurrence (if there are any) of the positive literal  $x$ , let the corresponding vertex be adjacent to a new *middle* vertex that is also joined by an edge to  $v_x$ . Assign the list  $\{1, 3\}$  to the middle vertex. For every occurrence of the negative literal  $\bar{x}$ , add an edge so that the corresponding vertex is adjacent to  $v_x$ .

We claim that  $(G, L)$  is a yes-instance of LIST SEMI-ACYCLIC 3-COLOURING if and only if  $\phi$  is a yes-instance of SATISFIABILITY.

First suppose that  $G$  has a semi-acyclic 3-colouring that respects  $L$ , and let us show that a satisfying assignment for  $\phi$  can be found. For each variable  $x$ , if  $v_x$  is





**Fig. 3** The instance  $(G, L)$  of LIST SEMI-ACYCLIC 3-COLOURING formed from the following instance of SATISFIABILITY:  $(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (\bar{x}_2 \vee x_3 \vee x_5)$ . The list for each vertex is displayed and literals label the vertices that represent them except that, for each variable  $x$ , one vertex is labelled  $v_x$

coloured 1, then let  $x$  be true; if it is coloured 3, let  $x$  be false. Note that every other vertex corresponding to the positive literal  $x$  must be coloured the same as  $x$ , and every vertex corresponding to an instance of  $\bar{x}$  is coloured differently, so each literal is coloured 1 if and only if it is true. Thus every clause contains a true literal, otherwise in the corresponding cycle every vertex would be coloured 2 or 3 and the colouring would not be semi-acyclic.

Now suppose that  $\phi$  has a satisfying assignment. If a literal is true in this assignment, colour the corresponding vertex 1, otherwise colour it 3. Colour each middle vertex with the colour not used on its neighbours (which must be coloured alike). Clearly this is a 3-colouring, and each cycle corresponding to a clause contains a vertex coloured 1 as it contains a true literal. No other cycle in the graph is coloured with only 2 and 3, as the only edges that do not belong to the cycles representing the clauses each join a vertex coloured 1 to a vertex coloured 3. Thus the colouring is semi-acyclic.  $\square$

By Theorem 2, to prove that LIST SEMI-ACYCLIC 3-COLOURING is polynomial-time solvable on  $P_5$ -free graphs, we need to refine our analysis and exploit  $P_5$ -freeness beyond the use of Lemma 2. We adapt the approach used by Hoàng et al. [24] to show that  $k$ -COLOURING is polynomial-time solvable on  $P_5$ -free graphs for all  $k \geq 3$  (extending the analogous result of Randerath et al. [41] for 3-COLOURING). Let us first outline the proof of [24].

The approach of Hoàng et al. [24] to solve  $k$ -COLOURING for  $P_5$ -free graphs for any integer  $k$  uses Lemma 2 as a starting point, just as the approach of Randerath et al. [41] does for the  $k = 3$  case. Lemma 2 implies that every  $k$ -colourable  $P_5$ -free graph  $G$  has a dominating set  $D$  of size at most  $k$  (as the clique number of a  $k$ -colourable graph is at most  $k$ ). Let the vertex set of  $D$  be  $\{v_1, \dots, v_{|D|}\}$ . Then decompose the set of vertices not in  $D$  into  $|D|$  “layers” so that the vertices in a layer  $i$  are adjacent to  $v_i$  (and possibly to  $v_j$  for  $j > i$ ) but not to any  $v_h$  with  $h < i$ . Using the  $P_5$ -freeness of  $G$  to analyse the adjacencies between different layers, it is possible to branch in such a way that a polynomial number of instances of  $(k - 1)$ -COLOURING are obtained.

Hence, by repetition, a polynomial number of instances of 2- COLOURING are reached, which can each be solved in polynomial time due to the result of [14].

The algorithm of [24] works by considering the more general LIST  $k$ - COLOURING problem, where each vertex  $v$  is assigned a list  $L(v) \subseteq \{1, \dots, k\}$  of permitted colours and the question is whether there is a colouring in which each vertex is assigned a colour from its list. The algorithm immediately removes any vertices whose lists have size 1 at any point (and then adjusts the lists of admissible colours of all neighbours of such vertices). We will follow the approach of [24]. In our case  $k = 3$ , but we cannot remove any vertices whose lists contain a singleton colour if this colour is 2 or 3. To overcome this extra complication we carefully analyse the 4-vertex cycles in the graph after observing that these cycles are the only obstacles that may prevent a 3-colouring of a  $P_5$ -free graph from being semi-acyclic.

For a subset  $S \subseteq V(G)$  of the vertex set of a graph  $G$ , we let  $G[S]$  denote the subgraph of  $G$  induced by  $S$ .

**Theorem 3** LIST SEMI- ACYCLIC 3- COLOURING is solvable on  $P_5$ -free graphs in  $O(n^{16})$  time.

*Proof* Consider an input  $(G, L)$  for the problem such that  $G$  is  $P_5$ -free. Since the problem can be solved component-wise, we may assume that  $G$  is connected. If  $G$  contains a  $K_4$ , then it is not 3-colourable and the input is a no-instance. As we can test whether or not  $G$  contains a  $K_4$  in  $O(n^4)$  time, we now assume that  $G$  is  $K_4$ -free. We may also assume that  $G$  contains at least three vertices, otherwise the problem can be trivially solved.

For  $i \in \{1, 2, 3\}$  let  $G_{-i} = G[\{v \in V(G) \mid i \notin L(v)\}]$ . We apply the following propagation rules exhaustively, and, later in the proof, every time we branch on possibilities, we assume that these rules are again applied exhaustively immediately afterwards.

**Rule 1.** If  $u, v \in V(G)$  are adjacent and  $|L(u)| = 1$ , set  $L(v) := L(v) \setminus L(u)$ .

**Rule 2.** If  $L(v) = \emptyset$  for some  $v \in V(G)$ , return no.

**Rule 3.** If  $G_{-i}$  is not bipartite for some  $i \in \{1, 2, 3\}$ , return no.

**Rule 4.** If  $G_{-1}$  contains an induced  $C_4$ , return no.

**Rule 5.** If  $G$  contains an induced  $C_4$ , and exactly one vertex  $v$  of this cycle has a list containing the colour 1, set  $L(v) = \{1\}$ .

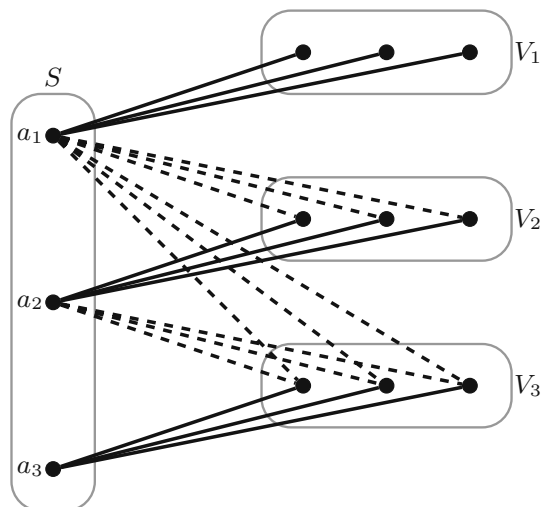
We must show that these rules are safe. That is, that when they modify the instance they do not affect whether or not it is a yes-instance or a no-instance, and when they return the answer no, this is correct and no semi-acyclic colouring that respects the lists can exist. This is trivial for Rules 1 and 2. We may apply Rule 3 since in any 3-colouring of  $G$  every pair of colour classes must induce a bipartite graph. We may apply Rules 4 and 5 since in every solution, every induced  $C_4$  must contain at least one vertex coloured with colour 1. In fact, if there is a 3-colouring of  $G$  with an induced cycle made of vertices coloured only 2 and 3, then this induced cycle must be an even cycle. Since  $G$  is  $P_5$ -free, such an induced cycle must in fact be isomorphic to  $C_4$ . Hence the problem, when restricted to  $P_5$ -free graphs, is equivalent to testing whether  $G$  has a 3-colouring respecting the lists such that every induced  $C_4$  contains at least one vertex coloured with colour 1.

By Lemma 2,  $G$  has a dominating set  $S$  that either is a clique or induces a  $P_3$ . If  $S$  is a clique, then it has at most three vertices, as  $G$  is  $K_4$ -free, so we can find such a set in  $O(n^4)$  time. Thus, adding vertices arbitrarily if necessary, we may assume  $S = \{a_1, a_2, a_3\}$ . We consider all possible combinations of colours that can be assigned to the vertices in  $S$ , that is, we branch into at most  $3^3$  cases, in which  $a_1, a_2$  and  $a_3$  have each received a colour, or equivalently, have had their list of permissible colours reduced to size exactly 1. In each case we proceed as follows.

Assume that  $L(a_1) = \{c_1\}$ ,  $L(a_2) = \{c_2\}$  and  $L(a_3) = \{c_3\}$  and again apply the propagation rules above. Partition the vertices of  $V \setminus S$  into three parts  $V_1, V_2, V_3$ : let  $V_1$  be the set of neighbours of  $a_1$  in  $V \setminus S$ , let  $V_2$  be the set of neighbours of  $a_2$  in  $V \setminus S$  that are not adjacent to  $a_1$ , and let  $V_3 = V(G) \setminus (S \cup V_1 \cup V_2)$  (see also Fig. 4). Each vertex in  $V_3$  is non-adjacent to  $a_1$  and  $a_2$ , so it is adjacent to  $a_3$ , as  $S$  is dominating. For  $i \in \{1, 2, 3\}$ , if  $v \in V_i$ , then  $L(v) \subseteq \{1, 2, 3\} \setminus \{c_i\}$  by Rule 1, so each vertex has at most two colours in its list. For  $i \in \{1, 2, 3\}$  let  $V'_i$  be the subset of vertices  $v$  in  $V_i$  with  $L(v) = \{1, 2, 3\} \setminus \{c_i\}$ . Recall that for  $i \in \{1, 2, 3\}$ , we defined  $G_{-i} = G[\{v \in V(G) \mid i \notin L(v)\}]$ . Since for every  $i \in \{1, 2, 3\}$ , every vertex of  $V_i$  belongs to  $G_{-c_i}$ , it follows that  $V_1, V_2$  and  $V_3$  each induce a bipartite graph in  $G$  by Rule 3. Therefore, we may partition each  $V'_i$  into two (possibly empty) independent sets  $V''_i$  and  $V'''_i$ .

Our strategy is to reduce the instance  $(G, L)$  to a polynomial number of instances  $(G, L')$ , in which there are no edges between any two distinct sets  $V'_i$  and  $V'_j$  (defined with respect to  $L'$ ). We will do this by branching on possible partial colourings in such a way that afterwards there are no edges between  $V''_i$  and  $V''_j$ , no edges between  $V''_i$  and  $V'''_j$  and no edges between  $V'''_i$  and  $V'''_j$  for every pair  $i, j \in \{1, 2, 3\}$  with  $i \neq j$ . As the branching procedure is similar for each of these possible combinations, we pick an arbitrary pair, namely  $V''_1$  and  $V''_2$ . As we shall see, we do not remove any edges between  $V''_1$  and  $V''_2$ . Instead, we decrease the lists of some of their

**Fig. 4** The sets  $S, V_1, V_2$  and  $V_3$ . Dashed lines indicate edges that are not present. Edges that are not shown may or may not be present. In particular, vertices in  $V_1$  can be adjacent to  $v_2$  or  $v_3$  and vertices in  $V_2$  can be adjacent to  $v_3$



vertices to size 1, so that these vertices will leave  $V'_1 \cup V'_2$  by definition of  $V'_1$  and  $V'_2$  (and therefore leave  $V''_1$  and  $V''_2$  by definition of  $V''_1$  and  $V''_2$ ).

Suppose that  $G[V''_1 \cup V''_2]$  contains an induced  $2P_2$  (see Fig. 5) with edges  $uu'$  and  $vv'$  for  $u, v \in V''_1$  and  $u', v' \in V''_2$ . Then  $G[\{u', u, a_1, v, v'\}]$  is a  $P_5$ , a contradiction. It follows that  $G[V''_1 \cup V''_2]$  is a  $2P_2$ -free bipartite graph, that is, the edges between  $V''_1$  and  $V''_2$  form a chain graph, which means that the vertices of  $V''_1$  can be linearly ordered by inclusion of neighbourhood in  $V''_2$ . In other words, we fix an ordering  $V''_1 = \{u_1, \dots, u_k\}$  such that  $N_{V''_2}(u_1) \supseteq \dots \supseteq N_{V''_2}(u_k)$ .

We choose an arbitrary colour  $c' \in \{1, 2, 3\} \setminus \{c_1, c_2\}$ . Note that if  $c_1 \neq c_2$  then this choice is unique and otherwise there are two choices (as we will show, it suffices to branch on only one choice). Also note that every vertex in  $V''_1$  and  $V''_2$  has colour  $c'$  in its list.

We now branch over  $k + 1$  possibilities, namely the possibilities that vertex  $u_i$  is the first vertex coloured with colour  $c'$  (so vertices  $u_1, \dots, u_{i-1}$ , if they exist, do not get colour  $c'$ ) and the remaining possibility that no vertex of  $V''_1$  is coloured with colour  $c'$ . To be more precise, for branch  $i = 1$  we set  $L(u_1) = \{c'\}$ , for each branch  $2 \leq i \leq k$  we remove colour  $c'$  from each of  $L(u_1), \dots, L(u_{i-1})$  and set  $L(u_i) = \{c'\}$  and for branch  $i = k + 1$  we remove colour  $c'$  from each of  $L(u_1), \dots, L(u_k)$ . If  $i = k + 1$ , all vertices of  $V''_1$  will have a unique colour in their list and thus leave  $V''_1$  and thus  $V''_1$  by definition of  $V''_1$ . Hence,  $V''_1$  becomes empty and thus, as required, we no longer have edges between  $V''_1$  and  $V''_2$ . Otherwise, if  $i \leq k$ , then all of  $u_1, \dots, u_i$  will have a list containing exactly one colour, so they will leave  $V''_1$  and therefore  $V''_1$ . By Rule 1 all neighbours of  $u_i$  in  $V''_2$  will have  $c'$  removed from their lists, so they will leave  $V''_2$  and therefore  $V''_2$ . By the ordering of neighbourhoods of vertices in  $V''_1$ , this means that no vertex remaining in  $V''_1$  has a neighbour remaining in  $V''_2$ , so if  $i \leq k$ , then it is also the case that we no longer have edges between  $V''_1$  and  $V''_2$ .

Note that removing all the edges between distinct sets  $V'_i$  and  $V'_j$  in the above way involves branching into  $O(n^{12})$  cases. We consider each case separately, and for each case we proceed as below.

By the above branching we may assume that there are no edges between any two distinct sets  $V'_i$  and  $V'_j$ . We say that an induced  $C_4$  is *tricky* if there exists a (proper) colouring of it (not necessarily extendable to all of  $G$ ) using only the colours 2 and 3 such that every vertex receives a colour from its list. We say that a vertex in an induced  $C_4$  is *good* for this induced  $C_4$  if its list contains the colour 1. By definition of tricky, every good vertex for a tricky induced  $C_4$  must belong to  $V'_1 \cup V'_2 \cup V'_3$ . By Rules 4 and 5, every tricky induced  $C_4$  must contain at least two good vertices. If an induced  $C_4$  contains two good vertices that are adjacent, then they must belong to the same set  $V'_i$  (since there are no edges between any two distinct sets  $V'_i$  and  $V'_j$ ), so they must have the same list. This means that in every colouring of this induced  $C_4$  that respects the lists, one of the good vertices in this induced  $C_4$  will be coloured with colour 1, contradicting the definition of tricky. We conclude that every tricky induced  $C_4$  must contain exactly two good vertices, which must be non-adjacent.

Fig. 5 The graph  $2P_2$



Suppose  $G$  contains a tricky induced  $C_4$  on vertices  $v_1, v_2, v_3, v_4$ , in that order, such that  $v_1$  and  $v_3$  are good. Since the  $C_4$  is tricky, we must either have:

- $2 \in L(v_1), 3 \in L(v_2), 2 \in L(v_3)$  and  $3 \in L(v_4)$  or
- $3 \in L(v_1), 2 \in L(v_2), 3 \in L(v_3)$  and  $2 \in L(v_4)$ .

Since  $v_2$  and  $v_4$  are not good, and there are no edges between distinct sets of the form  $V'_i$ , the above implies that one of the following must hold:

- $L(v_1) = \{1, 2\}, L(v_2) = \{3\}, L(v_3) = \{1, 2\}$  and  $L(v_4) = \{3\}$  or
- $L(v_1) = \{1, 3\}, L(v_2) = \{2\}, L(v_3) = \{1, 3\}$  and  $L(v_4) = \{2\}$ .

We say that an induced  $C_4$  is *strongly tricky* if its vertices have lists of this form (see also Fig. 6). Note that, by the above arguments, we may assume that all tricky induced  $C_4$ s in the instances we consider are in fact strongly tricky.

For  $S \subsetneq \{1, 2, 3\}$ , let  $L_S$  denote the set of vertices  $v$  with  $L(v) = S$  (to simplify notation, we will write  $L_i$  instead of  $L_{\{i\}}$  and  $L_{i,j}$  instead of  $L_{\{i,j\}}$  wherever possible). Note that for distinct sets  $S, T \subseteq \{1, 2, 3\}$  with  $|S| = |T| = 2$ , no vertex in  $L_S$  can have a neighbour in  $L_T$ , because such vertices would be in different sets  $V'_i$ , and therefore cannot be adjacent by our branching. By Rule 1, if  $S \subsetneq T \subsetneq \{1, 2, 3\}$  with  $|S| = 1$  and  $|T| = 2$ , then no vertex in  $L_S$  can have a neighbour in  $L_T$ . From the above two arguments it follows that if a vertex is in  $L_{1,2}, L_{2,3}$  or  $L_{1,3}$ , then all its neighbours outside this set must be in  $L_3, L_1$  or  $L_2$ , respectively (see also Fig. 7).

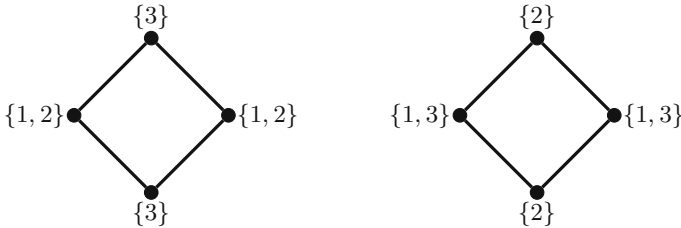
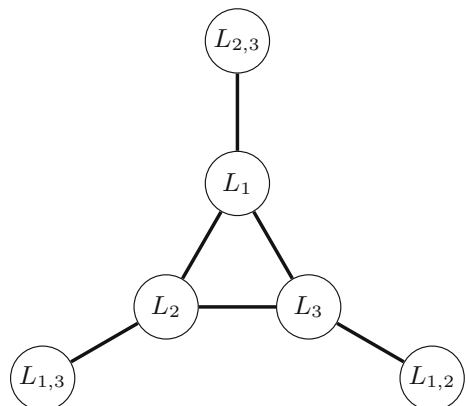


Fig. 6 Strongly tricky  $C_4$ s

Fig. 7 The possible adjacencies of vertices in the sets  $L_S$  for  $S \subseteq \{1, 2, 3\}$ . An edge is shown between two sets if and only if it is possible for a vertex in one of the sets to be adjacent to a vertex in the other. Every vertex in the graph has a list of size either 1 or 2



Recall that every tricky induced  $C_4$  is strongly tricky, and is therefore entirely contained in either  $G[L_2 \cup L_{1,3}]$  or  $G[L_3 \cup L_{1,2}]$ . By Rule 3,  $G_{-1}$  and therefore  $G[L_{2,3}]$  is bipartite. Hence we can colour the vertices of  $L_{2,3}$  with colours from their lists such that no vertex in  $L_{2,3}$  is adjacent to a vertex of the same colour in  $G$  and no induced  $C_4$ s are coloured with colours alternating between 2 and 3 (indeed, recall that induced  $C_4$ s cannot exist in  $G[L_{2,3}]$  by Rule 4). It therefore remains to check whether the vertices of  $G[L_2 \cup L_{1,3}]$  (and  $G[L_3 \cup L_{1,2}]$ ) can be coloured with colours from their lists so that no pair of adjacent vertices in  $L_{1,3}$  (resp.  $L_{1,2}$ ) receive the same colour and every strongly tricky  $C_4$  has at least one vertex coloured 1. By symmetry, it is sufficient to show how to solve the  $G[L_2 \cup L_{1,3}]$  case. Hence we have reduced the original instance  $(G, L)$  to a polynomial number of instances of a new problem, which we define below after first defining the instances.

**Definition 1** A graph  $G = (V, E)$  is *troublesome* if every vertex  $v$  in  $G$  has list either  $L(v) = \{2\}$  or  $L(v) = \{1, 3\}$ , such that  $L_2$  is an independent set and  $L_{1,3}$  induces a bipartite graph.

In particular, for each of our created instances the set  $L_2$  is independent due to Rule 1 and  $L_{1,3}$  induces a bipartite graph by Rule 3. Note that by definition of troublesome, all tricky induced  $C_4$ s in a troublesome graph are strongly tricky.

**Definition 2** Let  $G$  be a troublesome graph. A 3-colouring of the graph  $G$  is *trouble-free* if each vertex receives a colour from its list, no two adjacent vertices of  $G$  are coloured alike and at least one vertex of every strongly tricky induced  $C_4$  of  $G$  receives colour 1.

This leads to the following problem.

**TROUBLE-FREE COLOURING**

*Instance:* a troublesome  $P_5$ -free graph  $G$

*Question:* does  $G$  have a trouble-free colouring?

We can encode an instance of TROUBLE-FREE COLOURING as an instance of 2-SATISFIABILITY as follows. For each vertex  $u \in L_{1,3}$ , we create two variables  $u_1$  and  $u_3$ . If we assign  $u_1$  or  $u_3$  to be true, this means that  $u$  will be assigned colour 1 or 3, respectively. Hence, we need to add the clauses  $(u_1 \vee u_3)$  and  $(\bar{u}_1 \vee \bar{u}_3)$ . To ensure that two adjacent vertices are not coloured alike, for each pair of adjacent vertices  $u, v \in L_{1,3}$  we add the clauses  $(\bar{u}_1 \vee \bar{v}_1)$  and  $(\bar{u}_3 \vee \bar{v}_3)$ . For each strongly tricky  $C_4$  with good vertices  $u$  and  $v$ , we add the clause  $(u_1 \vee v_1)$  to ensure that at least one of them will be assigned colour 1. Let  $\mathcal{I}$  be the resulting instance of 2-SATISFIABILITY. We claim that  $G$  has a trouble-free colouring if and only if  $\mathcal{I}$  is satisfiable.

First suppose that  $G$  has a trouble-free colouring. Then setting  $x_c$  to be true if the vertex  $x$  receives colour  $c$  gives a satisfying assignment for  $\mathcal{I}$ . Now suppose that  $\mathcal{I}$  has a satisfying assignment. Then we colour the vertex  $v \in L_{1,3}$  with colour  $c$  if  $v_c$  is set to true in this assignment. For  $v \in L_2$  we colour  $v$  with colour 2. No two adjacent vertices of  $L_{1,3}$  are assigned the same colour, because  $\mathcal{I}$  is satisfied. No vertex of  $L_2$  is assigned the same colour as one of its neighbours, since  $L_2$  is an independent set

and every vertex of  $L_{1,3}$  is assigned colour 1 or 3. Therefore the obtained colouring is a 3-colouring of  $G$ . Since  $\mathcal{I}$  is satisfied, every strongly tricky  $C_4$  contains at least one vertex coloured 1. Hence  $G$  has a trouble-free colouring.

So, by branching, we have reduced the original instance  $(G, L)$  of LIST SEMI-ACYCLIC 3-COLOURING to a polynomial number of instances of 2-SATISFIABILITY. If we find that one of the instances of the latter problem is a *yes*-instance, then we obtain a corresponding *yes*-instance of TROUBLE-FREE COLOURING. We therefore solve TROUBLE-FREE COLOURING on  $G[L_2 \cup L_{1,3}]$  and (after swapping colours 2 and 3) on  $G[L_3 \cup L_{1,2}]$ . If one of these two instances of TROUBLE-FREE COLOURING is a *no*-instance, then we return *no* for this branch and try the next one. If both of these are *yes*-instances, then we return *yes* and obtain a semi-acyclic 3-colouring by combining the colourings on  $G[L_1 \cup L_{2,3}]$ ,  $G[L_2 \cup L_{1,3}]$  and (after swapping colours 2 and 3 back)  $G[L_3 \cup L_{1,2}]$ . If every branch returns *no* then the original graph has no semi-acyclic 3-colouring. This completes the proof of the correctness of the algorithm and it remains to analyse its runtime.

Let  $n$  be the number of vertices in  $G$ . Recall that we can check if  $G$  is  $K_4$ -free in  $O(n^4)$  time and if it is then we can find a dominating set of size at most 3 in  $O(n^4)$  time. Rule 1 can be applied in  $O(n^2)$  time. Rule 2 can be applied in  $O(n)$  time. Rule 3 can be applied in  $O(n^2)$  time. Rules 4 and 5 can be applied in  $O(n^4)$  time. We first branch up to  $3^3$  times and then sub-branch  $O(n^{12})$  times and in each case we apply the rules. We find all (strongly) tricky  $C_4$ s and the two corresponding good vertices in  $O(n^4)$  time. It is readily seen that every created instance of 2-SATISFIABILITY is solvable in  $O(n^2)$  time (see also Edwards [14]). This leads to a total runtime of  $O(n^4) + O(1) \times O(n^{12}) \times (O(n^4) + O(n^4) + O(n^2)) = O(n^{16})$ .  $\square$

As mentioned, Theorem 3 has the following consequence.

**Corollary 3** NEAR-BIPARTITENESS can be solved in  $O(n^{16})$  time for  $P_5$ -free graphs.

**Proof** Let  $G$  be a graph. Set  $L(v) = \{1, 2, 3\}$  for all  $v \in V(G)$ . Then  $G$  is near-bipartite if and only if  $(G, L)$  is a *yes*-instance of LIST SEMI-ACYCLIC 3-COLOURING. In particular, the vertices coloured 1 by a semi-acyclic colouring of  $G$  form an independent feedback vertex set of  $G$ . The corollary follows by Theorem 3.  $\square$

## 4 Independent Feedback Vertex Sets of $P_5$ -Free Graphs

In this section we prove that INDEPENDENT FEEDBACK VERTEX SET is polynomial-time solvable for  $P_5$ -free graphs by extending the algorithm from Sect. 3: the first part of our proof uses the proof of Theorem 3, as we will explain in the proof of Lemma 3. As such, we heavily use Definitions 1 and 2. Let  $G = (V, E)$  be a troublesome  $P_5$ -free graph. For a trouble-free colouring  $c$  of  $G$ , let  $t_c(G) = |\{u \in V \mid c(u) = 1\}|$  denote the number of vertices of  $G$  coloured 1 by  $c$ . Let  $t(G)$  be the minimum value  $t_c(G)$  over all trouble-free colourings  $c$  of  $G$ , and set  $t_c(G) = \infty$  if no such colouring exists.

**Lemma 3** Let  $G$  be a near-bipartite  $P_5$ -free graph. In  $O(n^{16})$  time it is possible to reduce the problem of finding the smallest independent feedback vertex set of  $G$  to finding the value  $t(G')$  of  $O(n^{12})$  instances of TROUBLE-FREE COLOURING, all on induced subgraphs of  $G$ .



**Proof** Let  $G$  be a near-bipartite  $P_5$ -free graph, that is, we assume that  $G$  has an independent feedback vertex set. We may assume that  $G$  is connected, otherwise, we solve the problem component-wise. We set  $L(v) = \{1, 2, 3\}$  for all  $v \in V(G)$  and run the algorithm of Theorem 3. As can be seen from the proof of Theorem 3, this algorithm branches up to  $3^3$  times and then sub-branches  $O(n^{12})$  times. Each branch gives us, after some preprocessing in  $O(n^4)$  time, either a no answer, in which case we discard this branch, or two vertex-disjoint instances of TROUBLE-FREE COLOURING (one on  $G[L_2 \cup L_{1,3}]$  and one (after swapping colours 2 and 3) on  $G[L_3 \cup L_{1,2}]$ ) and we will denote these two instances by  $G'$  and  $G''$ , respectively. Such instances consist of a troublesome graph  $G'$  or  $G''$ , which is an induced subgraph of  $G$  and whose vertices have lists of admissible colours determined by the branching.

As explained in the proof of Theorem 3, in any branch that we did not discard,  $G[L_1 \cup L_{2,3}]$  will have a semi-acyclic 3-colouring that respects the lists and  $L_1$  will be the set of vertices that are coloured 1 in any such colouring. Therefore, given trouble-free colourings  $c$  and  $c'$  of  $G'$  and  $G''$ , respectively, we can obtain an independent feedback vertex set  $S(c, c', G', G'')$  by taking the union of the set  $L_1$  of  $G[L_1 \cup L_{2,3}]$  and the sets of vertices in  $G'$  and  $G''$  that  $c$  or  $c'$  colour with colour 1.

Now let  $c^*$  and  $c^{**}$  be such that  $t(G') = t_{c^*}(G')$  and  $t(G'') = t_{c^{**}}(G'')$ . If we know  $t(G')$  and  $t(G'')$ , we can compute the size  $s(G', G'') = |S(c^*, c^{**}, G', G'')|$  in  $O(1)$  time (and, if we know  $c^*$  and  $c^{**}$ , we can compute the corresponding independent feedback vertex set  $S(c^*, c^{**}, G', G'')$  in  $O(n)$  time). Let  $\hat{s}$  be the minimum  $s(G', G'')$  over all branches of our procedure. As our procedure had  $O(n^{12})$  branches, given the values of  $t(G')$  and  $t(G'')$  for every branch, we can compute  $\hat{s}$  in  $O(n^{12})$  time. As we branched in every possible way,  $\hat{s}$  is the size of a minimum independent feedback vertex set of  $G$ . □

We still need a polynomial-time algorithm that computes  $t(G)$  for a given troublesome  $P_5$ -free graph. We present such an algorithm in the following lemma (in the proof of this lemma we again use Definitions 1 and 2).

**Lemma 4** *Let  $G$  be a troublesome  $P_5$ -free graph on  $n$  vertices. Determining  $t(G)$  can be done in  $O(n^3)$  time.*

**Proof** Let  $G = (V, E)$  be a troublesome  $P_5$ -free graph. Note that in  $G$ , an induced  $C_4$  on vertices  $v_1, v_2, v_3, v_4$ , in that order, is strongly tricky if  $v_1, v_3 \in L_{1,3}$  and  $v_2, v_4 \in L_2$ .

We construct an auxiliary graph  $H$  as follows. We let  $V(H) = L_{1,3}$ . Every edge of  $G[L_{1,3}]$  belongs to  $H$ . We say that such edges are *red*. For non-adjacent vertices  $v_1, v_3 \in L_{1,3}$ , if there is a strongly tricky induced  $C_4$  on vertices  $v_1, v_2, v_3, v_4$  with  $v_2, v_4 \in L_2$ , we add the edge  $v_1v_3$  to  $H$ . We say that such edges are *blue*. Note that  $H$  is a supergraph of  $G[L_{1,3}]$  and that there exists at most one edge, which is either blue or red, between any two vertices of  $H$ .

We say that a colouring of  $H$  *feasible* if the following two conditions are met:

- (i) no red edge is monochromatic, that is, the two end-vertices of every red edge must be coloured, respectively, 1&3 or 3&1;

- (ii) the two end-vertices of every blue edge must be coloured, respectively, 1&3, 3&1 or 1&1 (the only forbidden combination is 3&3, as in this case we obtain a strongly tricky induced  $C_4$  in  $G$  with colours 2 and 3).

We note that there is a one-to-one correspondence between the set of trouble-free colourings of  $G$  and the set of feasible colourings of  $H$ . Hence, we need to find a feasible colouring of  $H$  that minimises the number of vertices coloured 1.

Let  $R_1, \dots, R_p$  be the components of  $G[L_{1,3}]$ , or equivalently, of the graph obtained from  $H$  after removing all blue edges. We say that these are *red* components. As  $G[L_{1,3}]$  is bipartite and  $P_5$ -free, all red components of  $H$  are bipartite and  $P_5$ -free. We denote the bipartition classes of each  $R_i$  by  $X_i$  and  $Y_i$ , arbitrarily (note that these classes are unique, up to swapping their order). We apply the following rules on  $H$  exhaustively, making sure to only apply a rule if all previous rules have been applied exhaustively.

- Rule 1.** If there is a blue edge in  $H$  between two vertices  $u, v \in X_i$  or two vertices  $u, v \in Y_i$ , then assign colour 1 to  $u$  and  $v$ .
- Rule 2.** If there is a blue edge  $e$  in  $H$  between a vertex  $u \in X_i$  and a vertex  $v \in Y_i$ , then delete  $e$  from  $H$ .
- Rule 3.** If there are blue edges  $uv$  and  $u'v'$  such that either  $u, u' \in X_i$  or  $u, u' \in Y_i$  (where  $u = u'$  is possible), while  $v \in X_j$  and  $v' \in Y_j$  for some  $j \neq i$ , then assign colour 1 to  $u$  and  $u'$ .
- Rule 4.** If an uncoloured vertex  $u$  is adjacent to a vertex with colour 3 via a blue edge, then assign colour 1 to  $u$ .
- Rule 5.** If an uncoloured vertex  $u$  is adjacent to a coloured vertex  $v$  via a red edge, then assign colour 1 to  $u$  if  $v$  has colour 3 and assign colour 3 to  $u$  otherwise.
- Rule 6.** If there is a red edge with end-vertices both coloured 1 or both coloured 3, or a blue edge with end-vertices both coloured 3, then return  $\text{no}$ .
- Rule 7.** Remove all vertices that have received colour 1 or colour 3, keeping track of the number of vertices coloured 1.

Since each  $R_i$  is connected and bipartite, in every feasible colouring of  $H$ , for all  $i$  either all vertices in the set  $X_i$  must be coloured 1 and all vertices in the set  $Y_i$  must be coloured 3, or vice versa. Therefore we may safely apply Rules 1 and 2. Suppose that there exist vertices  $u$  and  $u'$  (we allow the case where  $u = u'$ ) in some  $X_i$  or in some  $Y_i$ , such that  $u$  is incident with a blue edge  $uv$  and  $u'$  is incident with a blue edge  $u'v'$  for two vertices  $v$  and  $v'$  that belong to different partition classes of the same red component  $R_j$  for some  $j \neq i$ . Then, as either  $v$  or  $v'$  must get colour 3 in every feasible colouring of  $H$  and  $u, u'$  must be coloured alike, we find that  $u$  and  $u'$  must receive colour 1. Hence Rule 3 is also safe to apply. Rules 4–6 are also safe; this follows immediately from the definition of a feasible colouring. If a vertex  $v$  is assigned colour 3, then by Rule 4 all its neighbours along blue edges get colour 1, so Property (ii) of a feasible colouring is satisfied for all blue edges with end-vertex  $v$ . If a vertex  $v$  is assigned a colour, then by Rule 5 all its neighbours along red edges get a different colour, so Property (i) of a feasible colouring is satisfied. We conclude that Rule 7 is safe.

By Rules 1 and 2, if two vertices are in the same red component  $R_i$ , we may assume that they are not connected by a blue edge. Hence, we may assume from now on that

red components contain no blue edges in  $H$ . By Rule 3, we may also assume that if  $i \neq j$  then either no vertex in  $X_j$  is joined by a blue edge to a vertex of  $X_i$  or no vertex in  $Y_j$  is joined by a blue edge to a vertex of  $X_i$  (and similarly with  $X_i$  replaced by  $Y_i$ ).

From  $H$  we construct another auxiliary graph  $H^*$  as follows. First, we replace each red component  $R_i$  on more than two vertices by an edge  $x_i y_i$ , which we say is a *red* edge. Hence, the set of red components of  $H$  is reduced to a set of *red* components in  $H^*$  in such a way that each red component of  $H^*$  is either an edge or a single vertex. Next, for  $i \neq j$  we add an edge, which we say is a *blue* edge, between two vertices  $x_i$  and  $x_j$  if and only if there is a blue edge between a vertex in  $X_i$  and a vertex in  $X_j$ . Similarly, for  $i \neq j$  we add a blue edge, between two vertices  $y_i$  and  $x_j$  (resp.  $y_j$ ) if and only if there is a blue edge between a vertex in  $Y_i$  and a vertex in  $X_j$  (resp.  $Y_j$ ).

Recall that, by Rules 1 and 2, no two vertices in the same component  $R_i$  are connected by a blue edge in  $H$ . So every feasible colouring of  $H$  corresponds to a feasible colouring of  $H^*$  and vice versa. To keep track of the number of vertices coloured 1, we introduce a weight function  $w : V(H^*) \rightarrow \mathbb{Z}_+$  by setting  $w(x_i) = |X_i|$  and  $w(y_i) = |Y_i|$ . Our new goal is to find a feasible colouring  $c$  of  $H^*$  that minimises the sum of the weights of the vertices coloured 1, which we denote by  $w(c)$ .

Since in the graph  $H$ , for  $i \neq j$  either no vertex in  $X_j$  is joined by a blue edge to a vertex of  $X_i$  or no vertex in  $Y_j$  is joined by a blue edge to a vertex of  $X_i$  (and similarly with  $X_i$  replaced by  $Y_i$ ), we find that  $H^*$  contains no triangle consisting of one red edge and two blue edges. As red edges induce a disjoint union of isolated edges, this means that the only triangles in  $H^*$  consist of only blue edges. Let  $B_1, \dots, B_q$  be the components of the graph obtained from  $H^*$  after removing all red edges. We say that these are *blue* components (this includes the case where they are singletons).

We will now show that all blue components of  $H^*$  are complete.

**Claim 1** *Each  $B_i$  is a complete graph.*

We prove Claim 1 as follows. For contradiction, suppose there is a blue component  $B_i$  that is not a complete graph. Then  $B_i$  contains three vertices  $u, v, w$  such that  $uv$  and  $vw$  are blue edges and  $uw$  is not a blue edge. As  $uv$  and  $vw$  are blue edges,  $v$  is not in the same red component of  $H^*$  as  $u$  or  $w$ . As no triangle in  $H^*$  can have two blue edges and one red edge,  $u$  and  $w$  are not adjacent in  $H^*$ , meaning that  $u, v, w$  in fact belong to three different red components in  $H$ . Let  $u'v'$  and  $v''w'$  be blue edges of  $H$  corresponding to the edges  $uv$  and  $vw$ , respectively. As  $uw$  is not a blue edge in  $H^*$ , we find that  $u'w'$  is not a blue edge in  $H'$ . We distinguish between two cases and show that neither of them is possible.

**Case 1**  $v' = v''$ .

As  $u'v'$  is a blue edge in  $H$ , we find that in  $G$ , the vertices  $u'$  and  $v'$  must have at least two common neighbours in  $L_2$ . For the same reason, in  $G$ , the vertices  $v'$  and  $w'$  must have at least two common neighbours in  $L_2$ . Since  $u'w'$  is not a blue edge in  $H$ , we find that in  $G$ , the vertices  $u'$  and  $w'$  have at most one common neighbour in  $L_2$ . Therefore  $G$  contains two vertices  $p, q \in L_2$  such that  $p$  is adjacent to  $u'$  and  $v'$  but non-adjacent to  $w'$  and  $q$  is adjacent to  $v'$  and  $w'$  but non-adjacent to  $u'$ . As  $L_2$  is an independent set in  $G$ ,  $p$  is non-adjacent to  $q$ . Now  $G[\{u', p, v', q, w'\}]$  is a  $P_5$ , which is a contradiction.

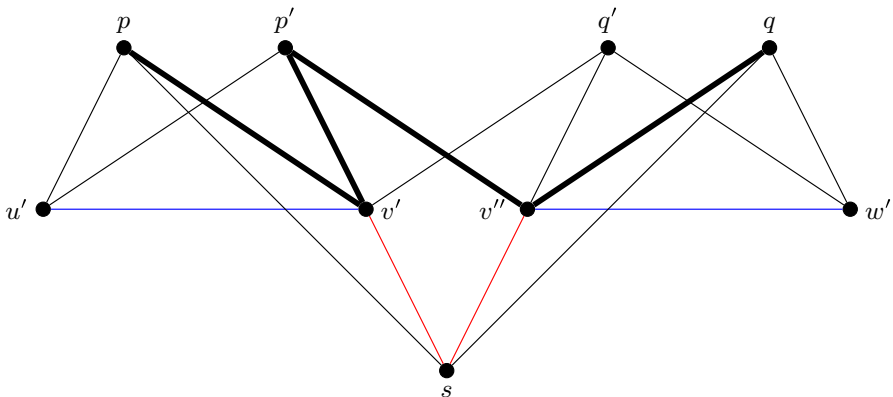
**Case 2**  $v' \neq v''$ .

Let  $R_i$  be the red component of  $H$  containing  $v'$  and  $v''$ . Then, due to the way the red edges of  $H^*$  are constructed, either  $v'$  and  $v''$  both belong to  $X_i$ , or they both belong to  $Y_i$ . As  $R_i$  is bipartite, connected and  $P_5$ -free,  $R_i$  must contain a vertex  $s$  that is adjacent to both  $v'$  and  $v''$ . Just as in Case 1, in  $G$  the vertices  $u'$  and  $v'$  have at least two common neighbours  $p, p' \in L_2$ , and  $v'$  and  $w'$  also have at least two common neighbours  $q, q' \in L_2$ . As  $L_2$  is independent in  $G$ , it follows that  $\{p, p', q, q'\}$  is also an independent set (which may have size smaller than 4).

Now  $p$  must be adjacent to at least one vertex in  $\{s, v''\}$ , as otherwise  $G[\{u', p, v', s, v''\}]$  would be a  $P_5$ . Similarly,  $p'$  must be adjacent to at least one vertex in  $\{s, v''\}$ . If  $p$  and  $p'$  are both adjacent to  $s$ , then there is a blue edge between  $u'$  and  $s$  in  $H$ . This is not possible, as then we would have applied Rule 3. If  $p$  and  $p'$  are both adjacent to  $v''$  then there is a blue edge between  $u'$  and  $v''$  in  $H$ , so Case 1 applies and we are done. We may therefore assume that  $p$  is adjacent to  $s$  but non-adjacent to  $v''$ , and that  $p'$  is adjacent to  $v''$  but non-adjacent to  $s$ . Similarly, we may assume that  $q$  is adjacent to  $s$  but non-adjacent to  $v'$ , and that  $q'$  is adjacent to  $v'$ , but non-adjacent to  $s$ . As  $p, p', q$  have different neighbourhoods in  $\{s, v''\}$ , we find that  $p, p', q$  are pairwise distinct. Recalling that  $\{p, p', q\} \subseteq L_2$  is an independent set, it follows that  $G[\{p, v', p', v'', q\}]$  is a  $P_5$  (see also Fig. 8). This contradiction completes Case 2. Hence we have proven Claim 1.

By Claim 1,  $H^*$  is the disjoint union of several blue complete graphs with red edges between them. Recall that we allow the case where these blue complete graphs contain only one vertex. On  $H^*$  we apply the following rule exhaustively in combination with Rules 4–7. While doing this we keep track of the weights of the vertices coloured 1.

**Rule 8.** If there exist (red) edges  $u_1v_1$  and  $u_2v_2$  for  $u_1, u_2 \in B_i$  and  $v_1, v_2 \in B_j$  ( $i \neq j$ ), then assign colour 1 to every vertex in  $(B_i \cup B_j) \setminus \{u_1, u_2, v_1, v_2\}$ .



**Fig. 8** An illustration of Case 2 of Claim 1 with the forbidden  $P_5$  shown in bold. The red edges edges  $sv'$  and  $sv''$  are present in both  $G$  and  $H$ . The blue edges  $u'v'$  and  $v''w'$  are present in  $H$ , but not  $G$ . The vertices  $p, p', q, q'$  are present in  $G$ , but not  $H$ , and the same holds for all of their incident edges. Note that the vertices  $p', q'$  are not necessarily distinct. The edges  $pw', p'w', qu'$  and  $q'u'$  may or may not be present in  $G$  (Color figure online)

Since Rules 4 and 5 can be safely applied on  $H$ , they can be safely applied on  $H^*$ . It follows that Rules 6 and 7 can also be safely applied on  $H^*$ . We may also safely apply Rule 8: the red edges  $u_1v_1$  and  $u_2v_2$  force  $u_i$  and  $v_i$  to have different colours for  $i \in \{1, 2\}$ , whereas the blue components forbid  $u_1, u_2$  both being coloured 3 and  $v_1, v_2$  both being coloured 3. Hence, exactly one of  $u_1, u_2$  and exactly one of  $v_1, v_2$  must be coloured 3. Because at most one vertex in any blue component may be coloured 3, this implies that all vertices in  $(B_i \cup B_j) \setminus \{u_1, u_2, v_1, v_2\}$  must be coloured 1.

As every vertex is incident with at most one red edge in  $H^*$ , we obtain a resulting graph that is an induced subgraph of  $H^*$  with the following property: if there exist (red) edges  $u_1v_1$  and  $u_2v_2$  for  $u_1, u_2 \in B_i$  and  $v_1, v_2 \in B_j$ , then  $\{u_1, u_2, v_1, v_2\}$  induces a connected component of  $H^*$ . We can colour such a 4-vertex component in exactly two ways and we remember the colouring with minimum weight (either  $w(u_1) + w(v_2)$  or  $w(u_2) + w(v_1)$  depending on whether  $u_1$  gets colour 1 or 3, respectively). Hence, from now on we may assume that the resulting graph, which we again denote by  $H^*$ , does not have such components. That is, there is at most one red edge between any two blue components of  $H^*$ . As we can colour  $H^*$  component-wise, we may assume without loss of generality that  $H^*$  is connected.

For each  $B_i$  we define the subset  $B'_i$  to consist of those vertices of  $B_i$  not incident with a red edge, and we let  $B''_i = B_i \setminus B'_i$ . We note the following. If we colour every vertex of some  $B''_i$  with colour 1, then every neighbour of every vertex of  $B''_i$  in any other blue component  $B_j$  must be coloured 3 by Rule 5 (recall that vertices in different blue components are connected to each other only via red edges). As soon as one vertex  $u$  in some blue component  $B_j$  has colour 3, all other vertices in  $B_j - u$  must get colour 1 by Rule 4. In this way we can use Rules 4 and 5 exhaustively to propagate the colouring to other vertices of  $H^*$  where we have no choice over what colour to use.

Recall that no vertex of  $H^*$  is incident with more than one red edge. This is a crucial fact: it implies that propagation to other blue components of  $H^*$  happens only via red edges  $vw$  between two blue components, one end-vertex of which, say  $v$ , is first coloured 1, which implies that the other end-vertex  $w$  of such an edge must get colour 3; this in turn implies that all other vertices in the blue component containing  $w$  must get colour 1 and so on. Hence, as  $H^*$  was assumed to be connected, colouring every vertex of a set  $B''_i$  with colour 1 propagates to all vertices of  $H^*$  except for the vertices of  $B'_i$ . Note that we may still colour (at most) one vertex of  $B'_i$  with colour 3.

Due to the above, we now do as follows for each  $i \in \{1, \dots, q\}$  in turn: We colour every vertex of  $B''_i$  with colour 1 and propagate to all vertices of  $H^*$  except for the vertices of  $B'_i$ . If we obtain a monochromatic red edge or a blue edge whose end-vertices are coloured 3, we discard this option (by Rule 6). Otherwise, we assign colour 3 to a vertex  $u \in B'_i$  with maximum weight  $w(u)$  over all vertices in  $B'_i$  (if  $B'_i \neq \emptyset$ ). We store the resulting colouring  $c_i$  that corresponds to this option.

After doing the above for all  $q$  options, it remains to consider the cases where every  $B''_i$  contains (exactly) one vertex coloured 3. Before we can use another propagation argument that tells us which vertices get colour 3, we first perform the following steps, only applying a step when the previous ones have been applied exhaustively. These steps follow immediately from the assumption that every  $B''_i$  contains a vertex coloured 3.

- (i) Colour all vertices of every  $B'_i$  with colour 1 (doing this does not cause any propagation).
- (ii) If some  $B''_i$  consists of a single vertex, then colour this vertex with colour 3, and afterwards propagate by using Rule 5 exhaustively.
- (iii) Remove coloured vertices using Rule 7.

If due to (ii) we obtain a monochromatic red edge or a blue edge whose end-vertices are coloured 3, we discard this option (using Rule 6). Otherwise, we may assume from now on that  $B'_i = \emptyset$ , so  $B''_i = B_i$  due to (i) and that  $|B_i| \geq 2$  due to (ii). Note that doing (iii) does not disconnect the graph: the vertices in the vertices in  $B'_i$  that are coloured in (i) only have neighbours in the clique  $B_i$  (and these are via blue edges) and if a vertex of  $v \in B''_i$  is coloured with colour 3 in (ii), then its only neighbour  $w$  (via a red edge) is in a set  $B''_j$  and since (i) has been applied exhaustively, the only other neighbours of  $w$  are in  $B''_j$  (via blue edges), so the propagation stops there and the graph does not become disconnected.

By our procedure, every vertex of every blue component  $B_i$  is incident with a red edge, so the total number of outgoing red edges for each  $B_i$  is equal to  $|B_i| \geq 2$ , and all outgoing red edges go to  $|B_i|$  different blue components. Hence the graph  $H'$  obtained from  $H^*$  by contracting each blue component to a single vertex has minimum degree at least 2. As  $H'$  has minimum degree at least 2, we find that  $H'$  contains an edge that is not a bridge (a bridge in a connected graph is an edge whose removal disconnects the graph). Let  $uv$  be the corresponding red edge in  $H^*$ , say  $u$  belongs to  $B_i$  and  $v$  belongs to  $B_j$ .

We have two options to colour  $u$  and  $v$ , namely by 1, 3 or 3, 1. We try them both. Suppose we first give colour 1 to  $u$ . Then we propagate in the same way as before. Because  $uv$  is not a bridge in  $H'$ , eventually we propagate back to  $B_i$  by giving colour 3 to an uncoloured vertex of  $B_i$ . When that happens we have “identified” the colour-3 vertex of  $B_i$  and then need to colour all other vertices of  $B_i$  with colour 1. This means that we can in fact propagate to all blue components of  $H^*$ , just as before. If at some point we obtain a monochromatic red edge or a blue edge with end-vertices coloured 3, then we discard this option (by Rule 6). Next, we give colour 1 to  $v$  and proceed similarly.

At the end we have at most  $q + 2$  different feasible colourings of  $H^*$ . We pick the one with minimum weight and translate the colouring to a feasible colouring of  $H$ . Finally, we translate the feasible colouring of  $H$  to a trouble-free colouring of the original graph  $G$ .

It remains to analyse the runtime. Let  $n$  be the number of vertices in  $G$ . Given two non-adjacent vertices in  $L_{1,3}$ , we can test whether they have two common neighbours in  $L_2$  in  $O(n)$  time. Therefore we can construct  $H$  in  $O(n^3)$  time.

Applying Rules 1 and 2 takes  $O(n^2)$  time. Applying Rule 3 takes  $O(n^3)$  time. Rules 1–3 only need to be applied exhaustively once, just after  $H$  is first constructed. Rules 4 and 5 can be applied exhaustively in  $O(n^3)$  time. Rule 6 can be applied in  $O(n^2)$  time. Rule 7 can be applied in  $O(n)$  time.

Constructing  $H^*$  takes  $O(n^2)$  time. By Claim 1, in  $H^*$  every blue component is a clique, so Rule 4 can be applied exhaustively on  $H^*$  in  $O(n^2)$  time. By construction, every red component of  $H^*$  contains at most one edge, so applying Rules 5 and 8 on  $H^*$

can be done in  $O(n^2)$  time. Therefore, Rules 4–8 can be applied to  $H^*$  in  $O(n^2)$  time. It follows that each option of colouring the vertices of some  $B_i''$  with colour 1 and then doing the propagation and colouring the vertices of  $B_i'$  takes  $O(n^2)$  time. Since there are  $q \leq n$  blue components, the total time for this is  $O(n^3)$ . Then afterwards we consider the situation where each blue component of  $H^*$  has exactly one vertex coloured 3.

We construct  $H'$  in  $O(n^2)$  time and also identify a non-bridge of  $H'$  in  $O(n^2)$  time. Colouring the corresponding red edges in both ways and doing the propagation takes  $O(n^2)$  time again. Then, if there is at least one possibility for which we did not return a no-answer, then we have obtained  $O(n)$  different feasible colourings of  $H^*$ . Finding the colouring with minimum weight and translating this colouring into a feasible colouring of  $H$  and then into a trouble-free colouring of the original graph  $G$  also takes  $O(n^2)$  time. □

We are now ready to state and prove the main result of our paper.

**Theorem 4** *The size of a minimum independent feedback vertex set of a  $P_5$ -free graph on  $n$  vertices can be computed in  $O(n^{16})$  time.*

**Proof** Let  $G$  be a  $P_5$ -free graph on  $n$  vertices. As we can check in  $O(n^{16})$  time whether or not  $G$  is near-bipartite, we may assume without loss of generality that  $G$  is near-bipartite. Then, by Lemma 3, in  $O(n^{16})$  time we can reduce the problem finding the value  $t(G')$  of  $O(n^{12})$  instances of TROUBLE-FREE COLOURING, all on induced subgraphs of  $G$  (which have at most  $n$  vertices). By Lemma 4 we can compute  $t(G')$  in  $O(n^3)$  time for each such instance. This gives a total runtime of  $O(n^{16})$ . The result follows. □

**Remark 1** From our proof, we can find in polynomial time not just the size of a minimum independent feedback vertex set, but also the set itself. The corresponding algorithm can also be adapted to find in polynomial time a maximum independent feedback vertex of a  $P_5$ -free graph, or an independent feedback vertex set of arbitrary fixed size (if one exists).

## 5 Independent Odd Cycle Transversal

Recall that an (independent) set  $S \subseteq V$  of a graph  $G$  is an (independent) odd cycle transversal if  $G - S$  is bipartite. We also recall that a graph  $G$  has an independent odd cycle transversal if and only if  $G$  is 3-colourable. This means that if 3-COLOURING is NP-complete for a graph class  $\mathcal{G}$ , then so is INDEPENDENT ODD CYCLE TRANSVERSAL. Hence, as 3-COLOURING is NP-complete for graphs of girth at least  $g$  for any constant  $g \geq 3$  [15] (see also [28,32]) and for line graphs [25], we find the following result.

**Proposition 2** INDEPENDENT ODD CYCLE TRANSVERSAL is NP-complete for

- graphs of girth at least  $g$  for any constant  $g \geq 3$ ;
- for line graphs.



As shown by Chiarelli et al. [13], ODD CYCLE TRANSVERSAL is also NP-complete for graphs of girth at least  $g$  for any constant  $g \geq 3$  and for line graphs. Hence, both problems are NP-complete for  $H$ -free graphs if  $H$  contains a cycle or a claw.

Our algorithm for INDEPENDENT FEEDBACK VERTEX SET restricted to  $P_5$ -free graphs can be adapted in the following way to solve INDEPENDENT ODD CYCLE TRANSVERSAL for  $P_5$ -free graphs. We follow the proof of Theorem 3 but remove Rules 4 and 5 used in that proof. For each branch, we still obtain the situation displayed in Fig. 7, and it remains to colour the vertices in  $L_{1,3}$  and  $L_{1,2}$  greedily and component-wise, such that the number of vertices with colour 1 is minimized.

**Theorem 5** *The size of a minimum independent odd cycle transversal of a  $P_5$ -free graph on  $n$  vertices can be computed in  $O(n^{16})$  time.*

## 6 Conclusions

Our main result is that INDEPENDENT FEEDBACK VERTEX SET is polynomial-time solvable for  $P_5$ -free graphs. As explained in Sect. 5, our algorithm can be readily adapted to also solve INDEPENDENT ODD CYCLE TRANSVERSAL for  $P_5$ -free graphs in polynomial time. We also proved that INDEPENDENT FEEDBACK VERTEX SET is NP-complete for  $H$ -free graphs if  $H$  contains a cycle or a claw. As discussed, the same hardness results were known for FEEDBACK VERTEX SET and 3-COLOURING, and the hardness results for 3-COLOURING immediately transfer across to INDEPENDENT ODD CYCLE TRANSVERSAL.

Another problem that is closely related to 3-COLOURING is INDEPENDENT VERTEX COVER, which is the independent problem variant of VERTEX COVER. The latter problem is that of testing whether or not a given graph  $G$  has a set  $S$  of size at most  $k$  for some given integer  $k$ , such that the vertices of  $G - S$  form an independent set. Similarly, the INDEPENDENT VERTEX COVER problem requires  $S$  to be an independent set and is equivalent to asking whether or not a graph has a 2-colouring such that one colour class has size at most  $k$ . This problem is clearly solvable in polynomial time. In contrast, VERTEX COVER is NP-complete for graphs of girth at least  $g$  for any constant  $g \geq 3$  [38], but VERTEX COVER stays polynomial-time solvable for claw-free graphs [34,42].

Apart from INDEPENDENT VERTEX COVER, the complexities of the other problems that we discussed are not settled for  $H$ -free graphs when  $H$  is a linear forest (disjoint union of one or more paths), or even when  $H$  is a path. Randerath and Schiermeyer [39] proved that 3-COLOURING is polynomial-time solvable for  $P_r$ -free graphs for  $r = 6$ , and more recently, Bonomo et al. [7] proved this for  $r = 7$ . The complexity of 3-COLOURING for  $P_r$ -free graphs is not known for  $r \geq 8$  (we refer to [20] for further details on  $k$ -COLOURING for  $P_r$ -free graphs).

The problems FEEDBACK VERTEX SET and ODD CYCLE TRANSVERSAL are polynomial-time solvable for the class of permutation graphs [9], which contains the class of  $P_4$ -free graphs [9], but their complexity is not known for  $P_r$ -free graphs when  $r \geq 5$ . This is in contrast to INDEPENDENT FEEDBACK VERTEX SET and INDEPENDENT ODD CYCLE TRANSVERSAL due to our result on  $P_5$ -free graphs. For these

**Table 1** The complexity of 3-COLOURING and eight related problems for  $P_r$ -free graphs, where  $r \geq 1$  is a fixed integer (the columns  $r \leq 4$  and  $r \geq 8$  represent multiple cases)

|                                   | $r \leq 4$ | $r = 5$ | $r = 6$ | $r = 7$ | $r = 8$ | $r \geq 9$ |
|-----------------------------------|------------|---------|---------|---------|---------|------------|
| 3-COLOURING                       | P          | P [41]  | P [39]  | P [7]   | ?       | ?          |
| VERTEX COVER                      | P          | P [30]  | P [22]  | ?       | ?       | ?          |
| INDEPENDENT VERTEX COVER          | P          | P       | P       | P       | P       | P          |
| FEEDBACK VERTEX SET               | P [9]      | ?       | ?       | ?       | ?       | ?          |
| INDEPENDENT FEEDBACK VERTEX SET   | P [43]     | P       | ?       | ?       | ?       | ?          |
| NEAR-BIPARTITENESS                | P [8]      | P       | ?       | ?       | ?       | ?          |
| ODD CYCLE TRANSVERSAL             | P [9]      | ?       | ?       | ?       | ?       | ?          |
| INDEPENDENT ODD CYCLE TRANSVERSAL | P          | P       | ?       | ?       | ?       | ?          |
| DOMINATING INDUCED MATCHING       | P          | P       | P       | P [10]  | P [11]  | ?          |

Here, P stands for being polynomial-time solvable, whereas open cases are denoted by ?

two problems we do not know their complexity for  $r \geq 6$ . As mentioned, Lokshantov et al. [30] proved that VERTEX COVER (or equivalently, INDEPENDENT SET) is polynomial-time solvable for  $P_5$ -free graphs, and this result was recently extended to  $P_6$ -free graphs by Grzesik et al. [22]. The computational complexity of VERTEX COVER for  $P_r$ -free graphs is not known for  $r \geq 7$ .

We refer to Table 1 for a summary of the above problems. In this table we also added the DOMINATING INDUCED MATCHING problem, which is also known as the EFFICIENT EDGE DOMINATION problem. This problem is that of deciding whether or not a graph  $G$  has an independent set  $S$  such that  $G - S$  is an *induced matching*, that is, the disjoint union of a set of isolated edges. Cardoso et al. [12] proved that  $G - S$  is in fact a maximum induced matching. We note that every graph  $G$  whose vertex set allows a partition into an independent set and an induced matching is 3-colourable. Grinstead et al. [21] proved that DOMINATING INDUCED MATCHING is NP-complete. Later, the problem was shown to be NP-complete or polynomial-time solvable for various graph classes. In particular, Brandstädt and Mosca [10] proved that DOMINATING INDUCED MATCHING for  $P_r$ -free graphs is polynomial-time solvable if  $r = 7$ . Later they extended their result to  $r = 8$  [11]. The complexity status of DOMINATING INDUCED MATCHING is unknown for  $r \geq 9$ . Hertz et al. [23] conjectured that the problem is polynomial-time solvable for  $H$ -free graphs whenever  $H$  is a forest, each connected component is a subdivided claw, a path or an isolated vertex.

Completing Table 1 is a highly non-trivial task. In particular, we note that no NP-hardness results are known for any of the problems in Table 1 when restricted to  $P_r$ -free graphs. As such, it would be interesting to know whether the problem of determining whether or not a  $P_r$ -free graph has an independent feedback vertex set (or equivalently, whether or not a  $P_r$ -free graph is near-bipartite) is polynomially equivalent to the 3-COLOURING problem restricted to  $P_{f(r)}$ -free graphs for some function  $f$ .

To solve INDEPENDENT FEEDBACK VERTEX on  $P_r$ -free graphs for  $r \in \{6, 7, 8\}$ , one could try to exploit the techniques used to solve 3-COLOURING for  $P_r$ -free graphs, just as we did for the  $r = 5$  case in this paper. However, this seems difficult due to additional complications and a different approach may be required.

Finally, we point out that the connected problem variants CONNECTED FEEDBACK VERTEX SET, CONNECTED ODD CYCLE TRANSVERSAL, and CONNECTED VERTEX COVER, which each require the desired set  $S$  of size at most  $k$  to induce a connected graph, are also known to be NP-complete for line graphs and graphs of arbitrarily large girth. This was shown by Chiarelli et al. [13] for CONNECTED FEEDBACK VERTEX SET and CONNECTED ODD CYCLE TRANSVERSAL, whereas Munaro [36] proved that CONNECTED VERTEX COVER is NP-hard for line graphs (of planar cubic bipartite graphs) and for graphs of arbitrarily large girth. Moreover, for these three problems the complexity has not yet been settled for  $H$ -free graphs when  $H$  is a linear forest (see [13] for some partial results in this direction).

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

1. Agrawal, A., Gupta, S., Saurabh, S., Sharma, R.: Improved algorithms and combinatorial bounds for independent feedback vertex set. In: Proceedings of IPEC 2016, LIPIcs, vol. 63, pp. 2:1–2:14 (2017)
2. Akiyama, T., Nishizeki, T., Saito, N.: NP-Completeness of the Hamiltonian cycle problem for bipartite graphs. *J. Inf. Process.* **3**(2), 73–76 (1980)
3. Bacsó, G., Tuza, Zs: Dominating cliques in  $P_5$ -free graphs. *Period. Math. Hung.* **21**(4), 303–308 (1990)
4. Bonamy, M., Dabrowski, K.K., Feghali, C., Johnson, M., Paulusma, D.: Independent feedback vertex set for  $P_5$ -free graphs. In: Proceedings of ISAAC 2017, LIPIcs, vol. 92, pp. 16:1–16:12 (2017)
5. Bonamy, M., Dabrowski, K.K., Feghali, C., Johnson, M., Paulusma, D.: Recognizing graphs close to bipartite graphs. In: Proceedings of MFCS 2017, LIPIcs, vol. 83, pp. 70:1–70:14 (2017)
6. Bonamy, M., Dabrowski, K.K., Feghali, C., Johnson, M., Paulusma, D.: Independent feedback vertex sets for graphs of bounded diameter. *Inf. Process. Lett.* **131**, 26–32 (2018)
7. Bonomo, F., Chudnovsky, M., Maceli, P., Schaudt, O., Stein, M., Zhong, M.: Three-coloring and list three-coloring of graphs without induced paths on seven vertices. *Combinatorica* (**in press**)
8. Brandstädt, A., Brito, S., Klein, S., Nogueira, L.T., Protti, F.: Cycle transversals in perfect graphs and cographs. *Theor. Comput. Sci.* **469**, 15–23 (2013)
9. Brandstädt, A., Kratsch, D.: On the restriction of some NP-complete graph problems to permutation graphs. In: Proceedings of FCT 1985, LNCS, vol. 199, pp. 53–62 (1985)
10. Brandstädt, A., Mosca, R.: Dominating induced matchings for  $P_7$ -free graphs in linear time. *Algorithmica* **68**(4), 998–1018 (2014)
11. Brandstädt, A., Mosca, R.: Finding dominating induced matchings in  $P_8$ -free graphs in polynomial time. *Algorithmica* **77**(4), 1283–1302 (2017)
12. Cardoso, D.M., Cerdeira, J.O., Delorme, C., Silva, P.C.: Efficient edge domination in regular graphs. *Discrete Appl. Math.* **156**(15), 3060–3065 (2008)
13. Chiarelli, N., Hartinger, T.R., Johnson, M., Milanič, M., Paulusma, D.: Minimum connected transversals in graphs: new hardness results and tractable cases using the price of connectivity. *Theor. Comput. Sci.* **705**, 75–83 (2018)
14. Edwards, K.: The complexity of colouring problems on dense graphs. *Theor. Comput. Sci.* **43**, 337–343 (1986)
15. Emden-Weinert, T., Hougardy, S., Kreuter, B.: Uniquely colourable graphs and the hardness of colouring graphs of large girth. *Comb. Probab. Comput.* **7**(04), 375–386 (1998)
16. Festa, P., Pardalos, P.M., Resende, M.G.C.: Feedback set problems. In: Floudas, C.A., Pardalos, P.M. (eds.) *Encyclopedia of Optimization*, 2nd edn, pp. 1005–1016. Springer, Berlin (2009)
17. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York (1979)
18. Goddard, W., Henning, M.A.: Independent domination in graphs: a survey and recent results. *Discrete Math.* **313**(7), 839–854 (2013)
19. Golovach, P.A., Heggernes, P.: Choosability of  $P_5$ -free graphs. In: Proceedings of MFCS 2009, LNCS, vol. 5734, pp. 382–391 (2009)
20. Golovach, P.A., Johnson, M., Paulusma, D., Song, J.: A survey on the computational complexity of colouring graphs with forbidden subgraphs. *J. Graph Theory* **84**(4), 331–363 (2017)
21. Grinstead, D.L., Slater, P.J., Sherwani, N.A., Holmes, N.D.: Efficient edge domination problems in graphs. *Inf. Process. Lett.* **48**(5), 221–228 (1993)
22. Grzesik, A., Klímová, T., Pilipczuk, M., Pilipczuk, M.: Polynomial-time algorithm for maximum weight independent set on  $P_6$ -free graphs. [arXiv:1707.05491](https://arxiv.org/abs/1707.05491) (2017)
23. Hertz, A., Lozin, V.V., Ries, B., Zamaraev, V., de Werra, D.: Dominating induced matchings in graphs containing no long claw. *J. Graph Theory* **88**(1), 18–39 (2018)
24. Hoàng, C.T., Kamiński, M., Lozin, V.V., Sawada, J., Shu, X.: Deciding  $k$ -colorability of  $P_5$ -free graphs in polynomial time. *Algorithmica* **57**(1), 74–81 (2010)

25. Holyer, I.: The NP-completeness of edge-coloring. *SIAM J. Comput.* **10**(4), 718–720 (1981)
26. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) *Complexity of Computer Computations*, pp. 85–103. Plenum Press, New York (1972)
27. Kociumaka, T., Pilipczuk, M.: Faster deterministic feedback vertex set. *Inf. Process. Lett.* **114**(10), 556–560 (2014)
28. Král', D., Kratochvíl, J., Tuza, Zs., Woeginger, J.G.: Complexity of coloring graphs without forbidden induced subgraphs. In: *Proceedings of WG 2001, LNCS*, vol. 2204, pp. 254–262 (2001)
29. Labarre, A.: Comment on “complexity of finding 2 vertex-disjoint  $(|V|/2)$ -cycles in cubic graphs?”. <http://csttheory.stackexchange.com/questions/6107/complexity-of-finding-2-vertex-disjoint-v-2-cycles-in-cubic-graphs> (2011). Accessed 24 June 2018
30. Lokshantov, D., Vatschelle, M., Villanger, Y.: Independent set in  $P_5$ -free graphs in polynomial time. In: *Proceedings of SODA*, pp. 570–581 (2014)
31. Lovász, L.: Coverings and coloring of hypergraphs. In: *Proceedings of the 4th Southeastern Conference on Combinatorics, Graph Theory, and Computing*. *Congressus Numerantium*, vol. VIII, pp. 3–12 (1973).
32. Lozin, V.V., Kamiński, M.: Coloring edges and vertices of graphs without short or long cycles. *Contrib. Discrete Math.* **2**(1), 61–66 (2007)
33. Marx, D., O'Sullivan, B., Razgon, I.: Finding small separators in linear time via treewidth reduction. *ACM Trans. Algorithms* **9**(4), 30:1–30:35 (2013)
34. Minty, G.J.: On maximal independent sets of vertices in claw-free graphs. *J. Comb. Theory Ser. B* **28**(3), 284–304 (1980)
35. Misra, N., Philip, G., Raman, V., Saurabh, S.: On parameterized independent feedback vertex set. *Theor. Comput. Sci.* **461**, 65–75 (2012)
36. Munaro, A.: Boundary classes for graph problems involving non-local properties. *Theor. Comput. Sci.* **692**, 46–71 (2017)
37. Munaro, A.: On line graphs of subcubic triangle-free graphs. *Discrete Math.* **340**(6), 1210–1226 (2017)
38. Poljak, S.: A note on stable sets and colorings of graphs. *Comment. Math. Univ. Carol.* **15**, 307–309 (1974)
39. Randerath, B., Schiermeyer, I.: 3-Colorability  $\in P$  for  $P_6$ -free graphs. *Discrete Appl. Math.* **136**(2–3), 299–313 (2004)
40. Randerath, B., Schiermeyer, I.: Vertex colouring and forbidden subgraphs: a survey. *Graphs Comb.* **20**(1), 1–40 (2004)
41. Randerath, B., Schiermeyer, I., Tewes, M.: Three-colourability and forbidden subgraphs. II: polynomial algorithms. *Discrete Math.* **251**(1–3), 137–153 (2002)
42. Sbihi, N.: Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. *Discrete Math.* **29**(1), 53–76 (1980)
43. Tamura, Y., Ito, T., Zhou, X.: Algorithms for the independent feedback vertex set problem. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **E98-A**(6), 1179–1188 (2015)
44. Yang, A., Yuan, J.: Partition the vertices of a graph into one independent set and one acyclic set. *Discrete Math.* **306**(12), 1207–1216 (2006)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Affiliations

Marthe Bonamy<sup>1</sup> · Konrad K. Dabrowski<sup>2</sup>  · Carl Feghali<sup>3</sup>  ·  
Matthew Johnson<sup>2</sup>  · Daniël Paulusma<sup>2</sup> 

✉ Matthew Johnson  
matthew.johnson2@durham.ac.uk

Marthe Bonamy  
marthe.bonamy@u-bordeaux.fr

Konrad K. Dabrowski  
konrad.dabrowski@durham.ac.uk

Carl Feghali  
feghali@irif.fr

Daniël Paulusma  
daniel.paulusma@durham.ac.uk

- 1 LaBRI, CNRS, Bordeaux, France
- 2 Department of Computer Science, Durham University, Durham, UK
- 3 IRIF, Université Paris Diderot, Paris, France