Independent Feedback Vertex Set for P_5-free Graphs[*†]

Marthe Bonamy[1], Konrad K. Dabrowski[2], Carl Feghali[3], Matthew Johnson[4], and Daniël Paulusma[5]

1 CNRS, LaBRI, Université de Bordeaux, France,
marthe.bonamy@u-bordeaux.fr
2 Durham University, Durham, UK,
konrad.dabrowski@durham.ac.uk
3 IRIF & Université Paris Diderot, France,
feghali@irif.fr
4 Durham University, Durham, UK,
matthew.johnson2@durham.ac.uk
5 Durham University, Durham, UK,
daniel.paulusma@durham.ac.uk

Abstract

The NP-complete problem Feedback Vertex Set is to decide if 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 if 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 in P for P_5-free graphs. We prove analogous results for the Independent Odd Cycle Transversal problem, which asks if a graph has an independent odd cycle transversal of size at most k for a given integer $k \geq 0$.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases feedback vertex set, odd cycle transversal, independent set, H-free graph

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2017.16

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 paper received support from EPSRC (EP/K025090/1), London Mathematical Society (41536), the Leverhulme Trust (RPG-2016-258) and Fondation Sciences Mathématiques de Paris.
[†] A full version of the paper is available at https://arxiv.org/abs/1707.09402.

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 if 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 was one of the original problems proven to be NP-complete by Karp. The proof of this implies NP-completeness of the undirected version even for graphs of maximum degree 4.

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.*

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 [6] for a survey on Independent Dominating Set, and [10] for Independent Odd Cycle Transversal, also known as Stable Bipartization. We survey results on Independent Feedback Vertex Set below.

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 recognising these 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 [15] or diameter 3 [3]. 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 [14]). 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; see [1, 11] for FPT algorithms for Independent Feedback Vertex Set when parameterized by k. 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. Tamura et al. [14] 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. [4], who proved that Near-Bipartiteness is in P for P_4-free graphs.

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 3-Colouring, which we observe is equivalent to the problem of deciding if a graph has an *independent odd cycle transversal*, that is, a set of vertices whose deletion makes the graph bipartite. 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 [14] for P_4-free graphs in a systematic way.

In Section 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 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 [14] for P_4-free graphs, the first open case to consider is when $H = P_5$.

The class of P_5-free graphs is a well-studied graph class. For instance, Hoàng et al. [8] proved that for every integer k, k-Colouring is polynomial-time solvable for P_5-free graphs, whereas Golovach and Heggernes [7] 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. [9] solved a long-standing open problem by giving a polynomial-time algorithm for Independent Set restricted to P_5-free graphs.

Our main result is that Independent Feedback Vertex Set is polynomial-time solvable for P_5-free graphs and this is proved in Sections 3 and 4. In Section 3 we give a polynomial-time algorithm for Near-Bipartiteness on P_5-free graphs. Then in Section 4 we show how to *extend* this algorithm to solve Independent Feedback Vertex Set in polynomial time for P_5-free graphs. Our results for Independent Feedback Vertex Set also hold for Independent Odd Cycle Transversal (see the arXiv version of our paper).

2 Hardness When H Contains a Cycle or Claw

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 with vertices a, b, c, d and edges ab, ac, ad. It is well known and easy to see that every line graph is claw-free. We omit the proof of Theorem 1.

▶ **Theorem 1.** Near-Bipartiteness *is* NP*-complete for line graphs of planar subcubic bipartite graphs.*

Feedback Vertex Set is also NP-complete for line graphs of planar cubic bipartite graphs [12]. Theorem 1 has the following immediate consequence (take $k = n$).

▶ **Corollary 2.** Independent Feedback Vertex Set *is* NP*-complete for line graphs of planar subcubic bipartite graphs.*

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. Proposition 3 follows from known results; we omit the proof.

▶ **Proposition 3.** *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.*

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 2 and Proposition 3 to obtain the following result.

▶ **Corollary 4.** *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 in P for P_5-free graphs, i.e. we give a polynomial-time algorithm for testing if 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 (apart from the step where we encode an instance of Trouble-Free Colouring as an instance of 2-Satisfiability) and then do the additional work described in Section 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 if 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.

List Semi-Acyclic 3-Colouring
Instance: a graph G and a function $L : V(G) \to \{S \mid S \subseteq \{1, 2, 3\}\}$.
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 recognise near-bipartite P_5-free graphs in polynomial time, we will show the stronger statement 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 5** ([2]). *Every connected P_5-free graph admits a dominating set that induces either a clique or a P_3.*

Lemma 5 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. [13] 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 [5], 2-List Colouring can be translated to an instance of 2-Satisfiability, which is 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 5 on 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, 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. We omit the proof of the next theorem.

▶ **Theorem 6.** List Semi-Acyclic 3-Colouring *is* NP*-complete even if $L(v) \in \{\{1, 3\}, \{2\}\}$ for every vertex v in the input graph.*

By Theorem 6, to prove that List Semi-Acyclic 3-Colouring is in P on P_5-free graphs, we need to refine our analysis and exploit P_5-freeness beyond the use of Lemma 5. We adapt the approach used by Hoàng et al. [8] to show that k-Colouring is in P on P_5-free graphs for all $k \geq 4$ (extending the analogous result of Randerath et al. [13] for 3-Colouring). Let us outline the proof of [8]. Lemma 5 implies that every k-colourable P_5-free graph G has a dominating set D of size at most k (as the clique number is at most k). Fix an ordering $D = \{v_1, \ldots, v_{|D|}\}$. 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 3-Colouring are reached, which can be solved in polynomial time due to the result of [13].

The algorithm of [8] works by considering the more general List k-Colouring problem, where each vertex v is assigned a list $L(v) \subseteq \{1, \ldots, k\}$ 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 [8], but 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 a graph G, we let $G[S]$ denote the subgraph of G induced by S.

▶ **Theorem 7.** 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 an induced K_4 then it is not 3-colourable and the input is a `no`-instance. As we can test whether G contains an induced 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 a cycle made of vertices coloured only 2 and 3, then this cycle must be an even cycle. Since G is P_5-free, such a 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 5, G has a dominating set S that either is a clique or induces a P_3. If it 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 $V_3 = V(G) \setminus (S \cup V_1 \cup V_2)$. 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 $G_i = G[\{v \in V(G) \mid i \notin L(v)\}]$. As for every $i \in \{1, 2, 3\}$, every vertex of V_i belongs to G_{c_i}, 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 vertices to size 1, so that these vertices will leave $V_1' \cup V_2'$ by definition of V_1', V_2' (and thus leave V_1'' and V_2'' by definition of V_1'', V_2'').

Suppose that $G[V_1'' \cup V_2'']$ contains an induced $2P_2$ 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 $V_1'' = \{u_1, \ldots, u_k\}$ such that $N_{V_2''}(u_1) \supseteq \cdots \supseteq N_{V_2''}(u_k)$.

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 $u_1, \ldots, u_{i-1}$, 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 $L(u_1), \ldots, L(u_{i-1})$ and set $L(u_i) = \{c'\}$ and for branch $i = k + 1$ we remove colour c' from each of $L(u_1), \ldots, L(u_k)$. 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 we no longer have edges between V_1'' and V_2''. Otherwise, if $i \leq k$, then all of $u_1, \ldots, u_i$ 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.

Thus 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 C_4 if its list contains the colour 1. By definition of tricky, every good vertex for a tricky C_4 must belong to $V'_1 \cup V'_2 \cup V'_3$. By Rules 4 and 5, every tricky C_4 must contain at least two good vertices. If a 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 C_4 that respects the lists, one of the good vertices in this C_4 will be coloured with colour 1, contradicting the definition of tricky. We conclude that every tricky C_4 must contain exactly two good vertices, which must be non-adjacent.

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. Note that, by the above arguments, we may assume that all tricky induced C_4s 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.

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_4s are coloured with colours alternating between 2 and 3 (indeed, recall that induced C_4s 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 $L_{1,3}$ (respectively $L_{1,2}$) 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 8.** 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_4s in a troublesome graph are strongly tricky.

▶ **Definition 9.** 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?

It is easy to verify that TROUBLE-FREE COLOURING can be encoded as an instance of 2-SATISFIABILITY. 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. We omit the runtime analysis. ◀

We obtain the following corollary.

▶ **Corollary 10.** 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 7. ◀

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 Section 3. We omit the proof of Lemma 11, which uses the proof of Theorem 7. As such, we heavily use Definitions 8 and 9. 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.

▶ **Lemma 11.** *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.*

By Lemma 11, it suffices to prove the following lemma (in its proof we again use the terminology introduced in the proof of Theorem 7).

▶ **Lemma 12.** *Let G be a troublesome P_5-free graph on n vertices. Determining $t(G)$ can be done in $O(n^4)$ 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 call a colouring of H *feasible* if the following two conditions are met:

1. no red edge is monochromatic, that is, the two end-vertices of every red edge must be coloured 1&3 respectively or 3&1 respectively;
2. 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 $R_1, \ldots, R_p$ be the components of $G[L_{1,3}]$, or equivalently, of the graph obtained from H after removing all blue edges. We call these components *red.* 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 five rules on H exhaustively (we omit proofs of correctness for these rules)

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 uv' where $u \in X_i \cup Y_i$, $v \in X_j$ and $v' \in Y_j$ ($j \neq i$), then assign colour 1 to 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 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 `no`.

Rule 7. Remove all vertices that have received colour 1 or colour 3, keeping track of the number of vertices coloured 1.

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 no vertex in $V(H) \setminus V(R_j)$ is joined via blue edges to both a vertex in X_j and a vertex in Y_j.

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_iy_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. We add blue edges between vertices y_i and y_j, and between vertices x_i and y_j in the same way.

Recall that, by Rules 1 and 2, no two vertices in any R_i are connected by a blue edge. 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 $w : V(H^*) \to \mathbb{Z}_+$ 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 for each i no vertex in $V(H) \setminus V(R_i)$ is joined via blue edges to both a vertex in X_i and a vertex in 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 $B_1, \ldots, B_q$ be the components of the graph obtained from H^* after removing all red edges. We call these components *blue* (even in the case where they are singletons). We need the following claim (we omit its proof).

▶ **Claim 13.** *Each B_i is a complete graph.*

By Claim 13, 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 $(B_i \cup B_j) \setminus \{u_1, u_2, v_1, v_2\}$.

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 Rule 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 $i \in \{1, \ldots, q\}$ 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. We omit the analysis of the runtime. ◀

▶ **Theorem 14.** *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 if G is near-bipartite, we may assume without loss of generality that G is near-bipartite. Then, by Lemma 11, 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 12 we can compute $t(G')$ in $O(n^4)$ time for each such instance. This gives a total runtime of $O(n^{16})$. ◀

▶ **Remark 15.** *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). Our algorithm can also be adapted to solve* Independent Odd Cycle Transversal *in $O(n^{16})$ time.*

References

1 Akanksha Agrawal, Sushmita Gupta, Saket Saurabh, and Roohani Sharma. Improved algorithms and combinatorial bounds for independent feedback vertex set. *Proc. IPEC 2016, LIPIcs*, 63:2:1–2:14, 2017.

2 Gábor Bacsó and Zsolt Tuza. Dominating cliques in P_5-free graphs. *Periodica Mathematica Hungarica*, 21(4):303–308, 1990.

3 Marthe Bonamy, Konrad K. Dabrowski, Carl Feghali, Matthew Johnson, and Daniël Paulusma. Recognizing graphs close to bipartite graphs. *Proc. MFCS 2017, LIPIcs*, 83:70:1–70:14, 2017.

4 Andreas Brandstädt, Synara Brito, Sulamita Klein, Loana Tito Nogueira, and Fábio Protti. Cycle transversals in perfect graphs and cographs. *Theoretical Computer Science*, 469:15–23, 2013.

5 Keith Edwards. The complexity of colouring problems on dense graphs. *Theoretical Computer Science*, 43:337–343, 1986.

6 Wayne Goddard and Michael A. Henning. Independent domination in graphs: A survey and recent results. *Discrete Mathematics*, 313(7):839–854, 2013.

7 Petr A. Golovach and Pinar Heggernes. Choosability of P_5-free graphs. *Proc. MFCS 2009, LNCS*, 5734:382–391, 2009.

8 Chính T. Hoàng, Marcin Kamiński, Vadim V. Lozin, Joe Sawada, and Xiao Shu. Deciding k-colorability of P_5-free graphs in polynomial time. *Algorithmica*, 57(1):74–81, 2010.

9 Daniel Lokshantov, Martin Vatshelle, and Yngve Villanger. Independent set in P_5-free graphs in polynomial time. *Proc. SODA 2014*, pages 570–581, 2014.

10 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30:1–30:35, 2013.

11 Neeldhara Misra, Geevarghese Philip, Venkatesh Raman, and Saket Saurabh. On parameterized independent feedback vertex set. *Theoretical Computer Science*, 461:65–75, 2012.

12 Andrea Munaro. On line graphs of subcubic triangle-free graphs. *Discrete Mathematics*, 340(6):1210–1226, 2017.

13 Bert Randerath, Ingo Schiermeyer, and Meike Tewes. Three-colourability and forbidden subgraphs. II: polynomial algorithms. *Discrete Mathematics*, 251(1–3):137–153, 2002.

14 Yuma Tamura, Takehiro Ito, and Xiao Zhou. Algorithms for the independent feedback vertex set problem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E98-A(6):1179–1188, 2015.

15 Aifeng Yang and Jinjiang Yuan. Partition the vertices of a graph into one independent set and one acyclic set. *Discrete Mathematics*, 306(12):1207–1216, 2006.