Feedback Vertex Set and Even Cycle Transversal for H-Free Graphs: Finding Large Block Graphs

Giacomo Paesani
Department of Computer Science, Durham University, UK

Daniël Paulusma
Department of Computer Science, Durham University, UK

Paweł Rzążewski
Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
Faculty of Mathematics, Informatics, and Mechanics, University of Warsaw, Poland

Abstract

We prove new complexity results for Feedback Vertex Set and Even Cycle Transversal on H-free graphs, that is, graphs that do not contain some fixed graph H as an induced subgraph. In particular, we prove that both problems are polynomial-time solvable for sP_3-free graphs for every integer $s \geq 1$; here, the graph sP_3 denotes the disjoint union of s paths on three vertices. Our results show that both problems exhibit the same behaviour on H-free graphs (subject to some open cases). This is in part explained by a new general algorithm we design for finding in a graph G a largest induced subgraph whose blocks belong to some finite class $\mathcal{C}$ of graphs. We also compare our results with the state-of-the-art results for the Odd Cycle Transversal problem, which is known to behave differently on H-free graphs.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Feedback vertex set, even cycle transversal, odd cactus, forest, block

Digital Object Identifier 10.4230/LIPIcs.MFCS.2021.82

Related Version *Full Version*: https://arxiv.org/abs/2105.02736

Funding *Daniël Paulusma*: Supported by the Leverhulme Trust (RPG-2016-258).
Paweł Rzążewski: Supported by Polish National Science Centre grant no. 2018/31/D/ST6/00062.

Acknowledgements The first author thanks Carl Feghali for an inspiring initial discussion.

1 Introduction

For a set of graphs $\mathcal{F}$, an $\mathcal{F}$*-transversal* of a graph G is a set of vertices that intersects the vertex set of every (not necessarily induced) subgraph of G that is isomorphic to some graph of $\mathcal{F}$. The problem Min $\mathcal{F}$-Transversal (also called $\mathcal{F}$-Deletion) is to find an $\mathcal{F}$-transversal of minimum size (or size at most k, in the decision variant). Graph transversals form a central topic in Discrete Mathematics and Theoretical Computer Science, both from a structural and an algorithmic point of view.

If $\mathcal{F}$ is the set of all cycles, the set of all even cycles or odd cycles, then we obtain the problems Feedback Vertex Set, Even Cycle Transversal and Odd Cycle Transversal, respectively. All three problems are NP-complete; hence, they have been studied for special graph classes, in particular *hereditary* graph classes, that is, classes closed under vertex deletion. Such classes can be characterized by a (unique) set $\mathcal{H}$ of minimal forbidden induced subgraphs. Then, in order to initiate a systematic study, it is standard to first consider the case where $\mathcal{H}$ has size 1, say $\mathcal{H} = \{H\}$ for some graph H.

46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).
Editors: Filippo Bonchi and Simon J. Puglisi; Article No. 82; pp. 82:1–82:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We aim to extend known complexity results for Feedback Vertex Set for H-free graphs and to perform a new, similar study for Even Cycle Transversal (for which, so far, mainly parameterized complexity results exist [2, 3, 11, 12]). To describe the known and new results we need some terminology. The cycle and path on r vertices are denoted C_r and P_r, respectively. The *disjoint union* of two vertex-disjoint graphs G_1 and G_2 is the graph $G_1 + G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$. We write sG for the disjoint union of s copies of G. For a set $S \subseteq V$, let $G[S]$ be the subgraph of G induced by S. We write $H \subseteq_i G$ (or $G \supseteq_i H$) if H is an induced subgraph of G.

1.1 Known Results

By Poljak's construction [14], for every integer $g \geq 3$, Feedback Vertex Set is NP-complete for graphs of girth at least g (the *girth* of a graph is the length of its shortest cycle). The same holds for Odd Cycle Transversal [7]. It is also known that Feedback Vertex Set [13] and Odd Cycle Transversal [7] are NP-complete for line graphs and thus for claw-free graphs (the claw is the 4-vertex star). Hence, both problems are NP-complete for the class of H-free graphs whenever H has a cycle or claw. A graph with no cycles and no claws is a forest of maximum degree at most 2. Thus, it remains to consider the case where H is a *linear forest*, that is, a collection of disjoint paths. Both problems are polynomial-time solvable on permutation graphs [5] and thus on P_4-free graphs [5], on sP_2-free graphs for every $s \geq 1$ [7] and on $(sP_1 + P_3)$-free graphs for every $s \geq 0$ [9]. Additionally, Feedback Vertex Set is polynomial-time solvable on P_5-free graphs [1], and Odd Cycle Transversal is NP-complete for $(P_2 + P_5, P_6)$-free graphs [9]. A similar NP-hardness result for Feedback Vertex Set or Even Cycle Transversal is unlikely: for every linear forest H, both problems are quasipolynomial-time solvable on H-free graphs [9] (see Section 4 for details).

1.2 Our Results

We first note that Min $\mathcal{F}$-Transversal is polynomially equivalent to Max Induced $\mathcal{F}$-Subgraph, the problem of finding a maximum-size induced subgraph of the input graph G that does not belong to $\mathcal{F}$ (where we assume that G has at least one such subgraph). We say that Max Induced $\mathcal{F}$-Subgraph is the *complementary* problem of Min $\mathcal{F}$-Transversal, and vice versa. For example, setting $\mathcal{F} = \{P_2\}$ yields the well-known complementary problems Min Vertex Cover and Max Independent Set.

Using the complementary perspective, we now argue that Feedback Vertex Set and Even Cycle Transversal are closely related, in contrast to Odd Cycle Transversal. A graph G is *biconnected* if it has at least two vertices, is connected, and $G - u$ is connected for every $u \in V(G)$. A *block* of a graph G is an inclusion-wise maximal biconnected subgraph of G. We now let $\mathcal{C}$ be a set of biconnected graphs. A graph G is a *$\mathcal{C}$-block graph* if every block of G is isomorphic to some graph in $\mathcal{C}$. If $\mathcal{C} = \{P_2\}$, then $\mathcal{C}$-block graphs are precisely forests, and if $\mathcal{C} = \{P_2, C_3, C_5, C_7, \ldots\}$, then $\mathcal{C}$-block graphs are called *odd cacti*. It is well known that a graph is an odd cactus if and only if it does not contain any even cycle as a subgraph. Hence, the complementary problems of Even Cycle Transversal and Feedback Vertex Set are somewhat similar: in particular, both forests and odd cacti have bounded treewidth and their blocks have a very simple structure. This is in stark contrast to Odd Cycle Transversal, whose complementary problem is to find a large induced bipartite subgraph, which might be arbitrarily complicated.

The commonality of complementary problems of Even Cycle Transversal and Feedback Vertex Set leads to the following optimization problem, where $\mathcal{C}$ is some fixed class of biconnected graphs, that is, $\mathcal{C}$ is not part of the input but specified in advance. Note that we consider the more general setting in which every vertex v of G is equipped with a weight $\mathfrak{w}(v) > 0$, and we must find a solution with maximum total weight, respectively.

Max $\mathcal{C}$-Block Graph	
Instance:	a graph $G = (V, E)$ with a vertex weight function $\mathfrak{w} : V \to \mathbb{Q}^+$.
Objective:	find a maximum-weight set $X \subseteq V$ such that $G[X]$ is a $\mathcal{C}$-block graph.

We observe that Max $\mathcal{C}$-Block Graph is well-defined for every set $\mathcal{C}$, including $\mathcal{C} = \emptyset$, as every independent set in a graph forms a solution. A restriction of the Max $\mathcal{C}$-Block Graph problem was introduced and studied from a parameterized complexity perspective by Bonnet et al. [4] as Bounded $\mathcal{C}$-Block Vertex Deletion (so from the complementary perspective) where each block must in addition have bounded size.

In Section 2 we slightly extend a previously known result, concerning the so-called *blob graphs* [10]. This extended version of the result forms a key ingredient for the proof of our main result, shown in Section 3, which is the following theorem for sP_3-free graphs (these are the graphs that become a disjoint union of cliques after removing the vertices of any induced $(s-1)P_3$ and their neighbours).

▶ **Theorem 1.** *For every integer $s \geq 1$ and every* finite *class $\mathcal{C}$ of biconnected graphs,* Max *$\mathcal{C}$-*Block Graph *can be solved in polynomial time for sP_3-free graphs.*

Theorem 1 implies the corresponding result for Feedback Vertex Set, as it is equivalent to Max $\{P_2\}$-Block Graph. The condition for $\mathcal{C}$ to be finite is critical for our proof technique. Nevertheless, we still have the corresponding result for Even Cycle Transversal as well: for sP_3-free graphs, the cases $\mathcal{C} = \{P_2, C_3, C_5, C_7, \ldots\}$ and $\mathcal{C} = \{P_2, C_3, C_5, \ldots, C_{4s-3}\}$ are equivalent. Note that we cannot make such an argument for Odd Cycle Transversal, as arbitrarily large bicliques are $2P_3$-free.

▶ **Corollary 2.** *For every integer $s \geq 1$,* Feedback Vertex Set *and* Even Cycle Transversal *can be solved in polynomial time for sP_3-free graphs.*

Corollary 2 extends the aforementioned results for Feedback Vertex Set on sP_2-free graphs and $(sP_1 + P_3)$-free graphs. In Section 4 we observe that as a direct consequence of a more general result [1], Even Cycle Transversal is polynomial-time solvable for P_5-free graphs. There we also prove that Even Cycle Transversal is NP-complete for graphs of large girth and for line graphs, and consequently, for H-free graphs where H contains a cycle or a claw. Hence, Feedback Vertex Set and Even Cycle Transversal behave similarly on H-free graphs, subject to a number of open cases, which we listed in Table 1.

2 Blob Graph of Graphs With No Large Linear Forest

Let $G = (V, E)$ be a graph. A *(connected) component* is a maximal connected subgraph of G. The *neighbourhood* of a vertex $u \in V$ is the set $N_G(u) = \{v \mid uv \in E\}$. For $U \subseteq V$, we let $N_G(U) = \bigcup_{u \in U} N(u) \setminus U$. Two sets $X_1, X_2 \subseteq V(G)$ are *adjacent* if $X_1 \cap X_2 \neq \emptyset$ or there exists an edge with one endpoint in X_1 and the other in X_2. The *blob graph* G° of G is defined as follows.

$$V(G^\circ) \coloneqq \{X \subseteq V(G) \mid G[X] \text{ is connected}\} \text{ and } E(G^\circ) \coloneqq \{X_1X_2 \mid X_1 \text{ and } X_2 \text{ are adjacent}\}.$$

Gartland et al. [10] showed that for every graph G, the length of a longest induced path in G° is equal to the length of a longest induced path in G. We slightly generalize this result.

Table 1 The complexity of FEEDBACK VERTEX SET (FVS), EVEN CYCLE TRANSVERSAL (ECT) and ODD CYCLE TRANSVERSAL (OCT) on H-free graphs for a linear forest H. All three problems are NP-complete for H-free graphs when H is not a linear forest (see also Section 4). The two blue cases (one for FVS, one for ECT) are the *algorithmic* contributions of this paper. We write $H \subseteq_i H'$ if H is an induced subgraph of H'. See Section 1.1 for references to the known results in the table.

	polynomial-time	unresolved	NP-complete
FVS	$H \subseteq_i P_5$ or sP_3 for $s \geq 1$	$H \supseteq_i P_1 + P_4$	none
ECT	$H \subseteq_i P_5$ or sP_3 for $s \geq 1$	$H \supseteq_i P_1 + P_4$	none
OCT	$H = P_4$ or $H \subseteq_i sP_1 + P_3$ or sP_2 for $s \geq 1$	$H = sP_1 + P_5$ for $s \geq 0$ or $H = sP_1 + tP_2 + uP_3 + vP_4$ for $s, t, u \geq 0$, $v \geq 1$ with $\min\{s, t, u\} \geq 1$ if $v = 1$, or $H = sP_1 + tP_2 + uP_3$ for $s, t \geq 0$, $u \geq 1$ with $u \geq 2$ if $t = 0$	$H \supseteq_i P_6$ or P_2+P_5

▶ **Theorem 3.** *For every linear forest H, a graph G contains H as an induced subgraph if and only if G° contains H as an induced subgraph.*

Proof. As G is an induced subgraph of G°, the ($\Rightarrow$) implication is immediate. We prove the ($\Leftarrow$) implication by induction on the number k of connected components of H. If $k = 1$, then the claim follows directly from the aforementioned result of Gartland et al. [10]. So assume that $k \geq 2$ and the statement holds for all linear forests H with fewer than k connected components. Let P' be one of the connected components of H, and define $H' := H - P'$.

Suppose that G° contains an induced subgraph isomorphic to H. Let $\mathcal{X}$ be the set of vertices of G°, such that $G^\circ[\mathcal{X}]$ is isomorphic to H. Furthermore, let $\mathcal{Y} \subseteq \mathcal{X}$ be the set of vertices that induce in $G^\circ[\mathcal{X}]$ the component P' of H, that is, $G^\circ[\mathcal{Y}]$ is isomorphic to P'.

Let $Y \subseteq V(G)$ be the union of sets in $\mathcal{Y}$. Note that $G^\circ[\mathcal{Y}]$ is an induced subgraph of $(G[Y])^\circ$. Thus, by the inductive assumption, $G[Y]$ contains an induced copy of P'.

Let $X \subseteq V(G)$ be the union of sets in $\mathcal{X} \setminus \mathcal{Y}$. Since the copy of H in G° is induced, we know that in G° there are no edges between $\mathcal{X} \setminus \mathcal{Y}$ and $\mathcal{Y}$. This is equivalent to saying that $X \cap N[Y] = \emptyset$. So we conclude that $G^\circ[\mathcal{X} \setminus \mathcal{Y}]$ is an induced subgraph of $(G - N[Y])^\circ$. Since $G^\circ[\mathcal{X} \setminus \mathcal{Y}]$, and thus $(G - N[Y])^\circ$, contains an induced copy of H', by the inductive assumption we know that $G - N[Y]$ contains an induced copy of H'. Combining this subgraph with the induced copy of P' in $G[Y]$, we obtain an induced copy of H in G. ◀

3 The Proof of Theorem 1

We start with analyzing the structure of sP_3-free $\mathcal{C}$-block graphs in Section 3.1, where $\mathcal{C}$ is any finite class of biconnected graphs. Then, in Section 3.2, we present our algorithm for MAX $\mathcal{C}$-BLOCK GRAPH on sP_3-free graphs.

3.1 Blocks and Terminals in sP_3-free Graphs

From now on, let $\mathcal{C}$ be a finite class of biconnected graphs. For some fixed positive integer s, let $G = (V, E)$ be an sP_3-free graph with n vertices and vertex weights $\mathfrak{w}\colon V \to \mathbb{Q}^+$. Let $X \subseteq V$ such that $F = G[X]$ is a $\mathcal{C}$-block graph. A component of F is *trivial* if it is a single vertex or a single block, otherwise it is *non-trivial.* Let F' be the graph obtained from F by removing all trivial components. Note that F' and F are sP_3-free, as G is sP_3-free.

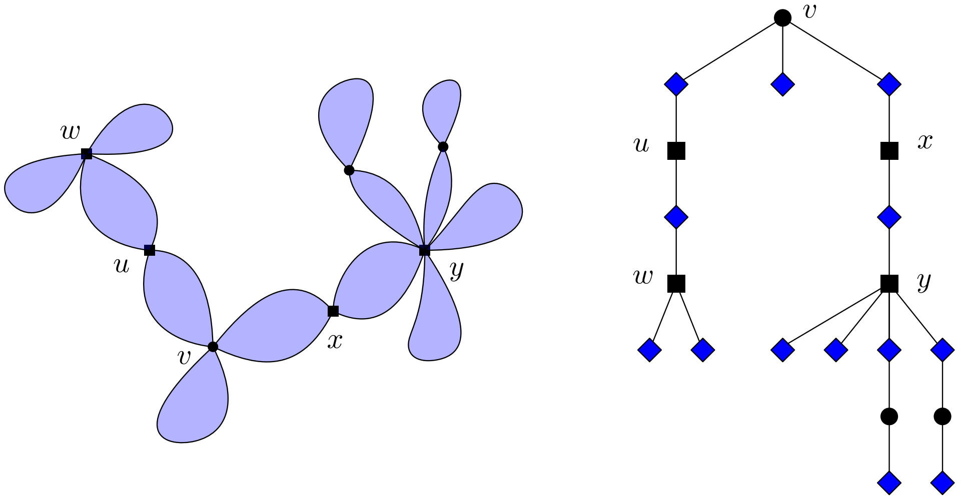

Figure 1 Left: a graph F'. Blue shapes are blocks, squares are terminals, and dots are non-terminal cutvertices. Right: $\mathsf{BCF}(F')$, rooted in the cutvertex v. Blue diamonds are blocks; w is a terminal of type 1, u and x are terminals of type 2, and y is a terminal of both types. The remaining cutvertices are not terminals. *We also use this example with this particular* $\mathsf{BCF}(F')$ *in later figures.*

We denote the set of cutvertices of F' and the set of blocks of F' by $\mathsf{Cutvertices}(F')$ and $\mathsf{Blocks}(F')$, respectively. The *block-cut forest* $\mathsf{BCF}(F')$ of F' has vertex set $\mathsf{Cutvertices}(F') \cup \mathsf{Blocks}(F')$ and an edge set that consists of all edges xb such that $x \in \mathsf{Cutvertices}(F')$ and $b \in \mathsf{Blocks}(F')$, and x belongs to b. By definition, each component of F' has a cutvertex; we pick an arbitrary one as root for the corresponding tree in $\mathsf{BCF}(F')$ to get a parent-child relation. Each leaf of $\mathsf{BCF}(F')$ belongs to $\mathsf{Blocks}(F')$, and we call such blocks *leaf blocks*.

A cutvertex x of F' is a *terminal of type 1* if x has at least two children in $\mathsf{BCF}(F')$ that are leaves, whereas x is a *terminal of type 2* if there exists a leaf block, whose great-grandparent in $\mathsf{BCF}(F')$ is x. In the latter case, there is a three-edge downward path from x to a leaf in $\mathsf{BCF}(F')$; see also Fig. 1. Let d be the maximum number of vertices of a graph in $\mathcal{C}$.

▶ **Lemma 4.** *At most $d \cdot (s-1)$ vertices of F' are terminals of type 1.*

Proof. For contradiction, suppose that there are at least $d \cdot (s-1)+1$ terminals of type 1. We observe that F' is d-colourable. Indeed, each block has at most d vertices, so d colours are sufficient to colour each block. Furthermore, we can permute the colours in each block, so that the colourings agree on cutvertices.

This implies that there is an independent set X of size at least s, whose every element is a terminal of type 1. For each such terminal v, let its *private* P_3 be a 3-vertex path with v as the central vertex and each endpoint belonging to a different leaf block that is a child of v in $\mathsf{BCF}(F')$. Note that each private P_3 is induced. Furthermore, the private P_3's of vertices in X are pairwise non-adjacent: this follows from the definition of terminals of type 1 and the fact that X is independent. Thus we have found an induced sP_3 in F, a contradiction. ◀

▶ **Lemma 5.** *At most $(d+1) \cdot (s-1)$ vertices of F' are terminals of type 2.*

Proof. For contradiction, suppose that there are at least $(d+1) \cdot (s-1)+1$ terminals of type 2. Observe that F' has a proper $(d+1)$-colouring f, satisfying the following two properties:

1. the vertices in each block receive pairwise distinct colours and
2. if b is a block, then any vertex of b receives a colour which is different than the colour of the cutvertex which is the great-grandparent of b in $\mathsf{BCF}(F')$ (if such a cutvertex exists).

It is easy to find such a colouring of each tree in $\mathsf{BCF}(F')$ by choosing an arbitrary colour for the root and proceeding in a top-down fashion. Suppose we want to colour the block b and its parent in $\mathsf{BCF}(F')$ is the cutvertex v. Recall that b has at most d vertices and exactly one of them is already coloured. Furthermore, we want to avoid the colour of the grandparent of v (if such a vertex exists), so we have sufficiently many free colours to colour each vertex of $b \setminus \{v\}$ with a different one.

Now, by our assumption, there is a set X of at least s terminals of type 2 that received the same colour in f. For each $v \in X$, we define its *private* P_3 as follows. Recall that by the definition of a terminal of type 2, there is a leaf block b, whose great-grandparent in $\mathsf{BCF}(F')$ is v. The private P_3 of v is given by the first three vertices on a shortest path P from v to b. Note that in the extreme case it might happen that both b and its grandparent in $\mathsf{BCF}(F')$ are edges, but P always has at least three vertices.

Clearly, each private P_3 is an induced path. We claim that the private P_3's associated with the vertices of X are non-adjacent. For contradiction, suppose otherwise. Let v_1, v_2 be distinct vertices of X, and let v_i, x_i, y_i be the consecutive vertices of the private P_3 associated with v_i. Let b_i be the block containing v_i and x_i.

First, observe that the sets $\{v_1, x_1, y_1\}$ and $\{v_2, x_2, y_2\}$ are disjoint. Indeed, we know that $v_1 \neq v_2$ and because $\mathsf{BCF}(F')$ is a rooted tree, we have that $\{x_1, y_1\} \cap \{x_2, y_2\} = \emptyset$. Furthermore, recall that $f(v_1) = f(v_2)$ and by the definition of f, we have that the colours of x_i and of y_i are different from the colour of v_i.

So now suppose that there is an edge with one endpoint in $\{v_1, x_1, y_1\}$ and the other in $\{v_2, x_2, y_2\}$. Clearly this edge cannot join v_1 and v_2, as the colouring f is proper. Furthermore, there is no edge between $\{x_1, y_1\}$ and $\{x_2, y_2\}$, as v_1 and v_2 are cutvertices of a rooted tree. Suppose that v_2 is adjacent to x_1 (the case that v_1 is adjacent to x_2 is symmetric). As each vertex of b_1 gets a different colour in f, we observe that v_2 cannot belong to b_1. Thus x_1 is a cutvertex. However, by the second property of f, we obtain that the colour of v_2 must be different than the colour of v_1, a contradiction.

So finally suppose that v_2 is adjacent to y_1. Note that then y_1 cannot belong to a leaf block, meaning that y_1 belongs to b_1. Similarly to the previous paragraph, the definition of f implies that the colour of v_2 must be different than the colour of v_1, a contradiction.

Thus we have found an induced sP_3 in F', a contradiction. ◀

Lemmas 4 and 5 imply the following.

▶ **Lemma 6.** *The number of terminals of F' is at most $(2d + 1) \cdot (s - 1)$.*

If v is a terminal of type 2, then by definition there is a cutvertex w that belongs to both a block containing v as well as to some leaf block. We call such w a *witness* of v. We now partition the set of blocks of F' into the following subsets; see also Fig. 2:

- $\mathcal{B}_{l_1}$ is the set of leaf blocks containing a terminal of type 1,
- $\mathcal{B}_{l_2}$ is the set of leaf blocks containing a witness w that is not a terminal of type 1,
- $\mathcal{B}_{l_3}$ is the set of remaining leaf blocks, that is, the ones with a cutvertex that is neither a terminal nor a witness,
- $\mathcal{B}_w$ is the set of blocks with precisely two cutvertices, one of which is a terminal of type 2 and the other one the non-terminal witness of that type-2 terminal, and
- $\mathcal{B}_{in}$ is the set of all remaining blocks.

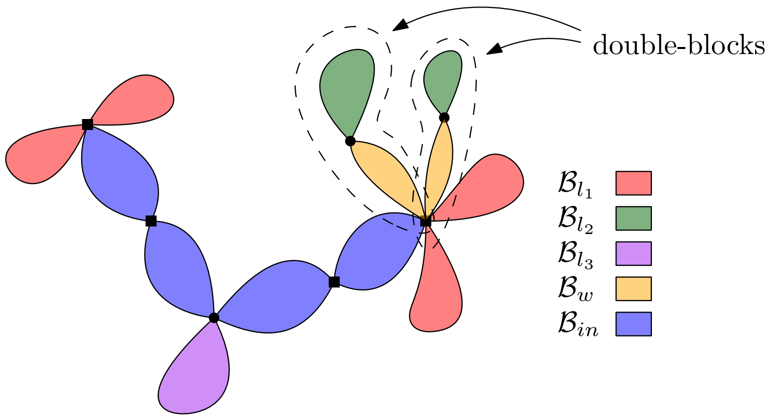

Figure 2 The classification of blocks of the example of Figure 1.

Note that blocks in $\mathcal{B}_{l_2}$ and $\mathcal{B}_w$ come in pairs, that is, for each block B in one of these sets, there is exactly one block B' in the other set, such that B and B' share a vertex (it is the witness w, using the notation introduced above). We will consider these two blocks as one object. Formally, a *double-block* is a graph consisting of two blocks sharing a cutvertex. Let $\mathcal{B}_d$ be the family of double-blocks of F' obtained from blocks in $\mathcal{B}_{l_2}$ and $\mathcal{B}_w$ in the way described above, i.e., $\mathcal{B}_d$ consists of graphs $G[V(B) \cup V(B')]$, where $B \in \mathcal{B}_{l_2}, B' \in \mathcal{B}_w$ and $V(B) \cap V(B') \neq \emptyset$. Note that each double-block in $\mathcal{B}_d$ has fewer than $2d$ vertices and contains exactly one terminal of type 2.

A *backbone* of a component Z of F' is a minimum tree T_Z contained in Z that connects all terminals of F' that belong to Z; observe that all leaves of T_Z are terminals. The *skeleton* S of F' is the graph obtained from F' by removing all vertices from the blocks in $\mathcal{B}_{l_1}$ except terminals of type 1 and all vertices from the double-blocks in $\mathcal{B}_d$ except terminals of type 2. Note that every backbone is a subgraph of S.

3.2 The Algorithm

Outline. Our polynomial-time algorithm consists of the following two phases:

1. *Branching Phase*, which consists of the following three steps:
   1. guessing the terminals of F';
   2. guessing the backbones of the components of F'; and
   3. guessing the skeleton of F', and
2. *Completion Phase*, where we extend the partial solutions obtained in the Branching Phase to complete ones by finding non-skeleton vertices of F' and trivial components of F; we do this by:
   1. reducing the problem to Max Weight Independent Set for sP_3-free graphs using the blob graph construction in Section 2, and
   2. solving this problem using the polynomial-time algorithm of Brandstädt and Mosca [6].

We now describe our algorithm, prove its correctness and perform a running time analysis.

Branching Phase. This phase of our algorithm consists of a series of guesses, where we find certain vertices and substructures in G. The total number of vertices to be guessed will be $\mathcal{O}(s^2d^2)$. Since we guess them exhaustively, this results in a recursion tree with $\mathcal{O}(n^{\mathcal{O}(s^2d^2)})$ leaves. As both s and d are constants, this bound is polynomial in n. We will ensure that

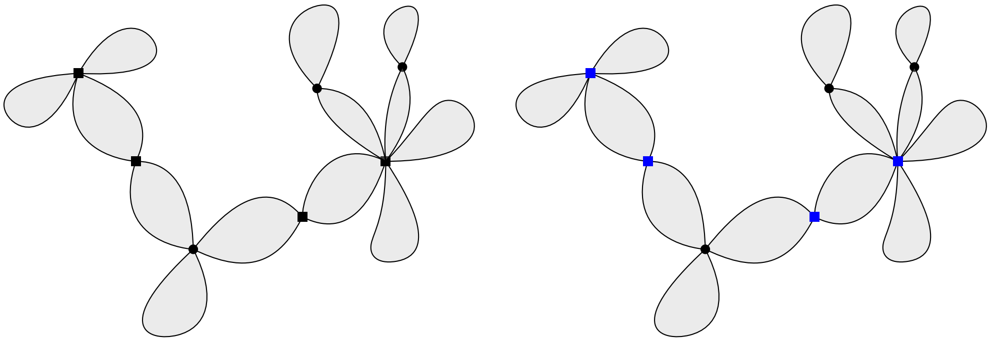

Figure 3 Step 1 of the Branching Phase. Left: the graph F'. Right: the terminals of F'.

the optimum solution $F = G[X]$ will be found in the call corresponding to at least one of the leaves of the recursion tree. Based on the properties of F, we will expect the guessed vertices to satisfy certain conditions. If, at some point, the guessed vertices do not satisfy these conditions, we just terminate the current call, as it will not lead us to find F. This will be applied implicitly throughout the execution of the algorithm.

The branching phase is illustrated in Figures 3–5. We use the convention that gray/black elements are still unknown and blue elements are the ones that we have already guessed.

Step 1. Guessing the terminals of $\mathbf{F'}$. We guess the set $C \subseteq V$ of terminals of F'. By Lemma 6, the total number of terminals is bounded by $(2d+1)\cdot(s-1) \leq 3ds$. Furthermore, for each terminal, we guess its type (1, 2, or both). This results in $3^{|C|} \leq 3^{3ds}$ possibilities. We also guess the partition of C, corresponding to the connected components of F. This results in at most $|C|^{|C|} \leq (3ds)^{3ds}$ additional branches. In total, we have $\mathcal{O}(n^{\mathcal{O}(ds)})$ branches.

Step 2. Guessing the backbone of each component of $\mathbf{F'}$. Let Z be a component of F'. Let $C_Z \subseteq C$ be the subset of terminals that are in Z. Let T_Z be the backbone of Z. Let T'_Z be the tree obtained from T_Z by contracting every path in T_Z whose internal vertices are all non-terminals and of degree 2 to an edge. Note that every non-terminal vertex of T'_Z has degree at least 3. Since T'_Z has at most $|C_Z|$ vertices of degree at most 2, by the handshaking lemma we observe that the total number of vertices of T'_Z is at most $2|C_Z|$. Recall that every edge of T'_Z corresponds to an induced path in T_Z. Since F' is sP_3-free and thus P_{4s-1}-free, we conclude that T_Z has at most $2|C_Z|\cdot(4s-2) \leq 8s\cdot|C_Z|$ vertices.

Let T be the forest whose components are the guessed backbones of the components of F'. Note that the total number of vertices of T is at most $\sum_Z 8s\cdot|C_Z| = 8s\cdot|C| \leq 24ds^2$. Thus we may guess the whole forest T, which results in $\mathcal{O}(n^{\mathcal{O}(ds^2)})$ branches.

Step 3. Guessing the skeleton of $\mathbf{F'}$. Let T be the forest guessed in the previous step; recall that T has at most $24ds^2$ vertices. We guess the partition of $E(T)$ corresponding to *blocks* of F'; note that a vertex v may be in several blocks: this happens precisely if v is a cutvertex in F'. This results in at most $|E(T)|^{\mathcal{O}(|E(T)|)} \leq |V(T)|^{\mathcal{O}(|V(T)|)} \leq (ds)^{\mathcal{O}(ds^2)}$ branches.

We now discuss some properties of the (double-)blocks. We use the names of vertices as in the definitions introduced in Section 3.1, recall also Fig. 2. The crucial observation is that now there is a branch, where:

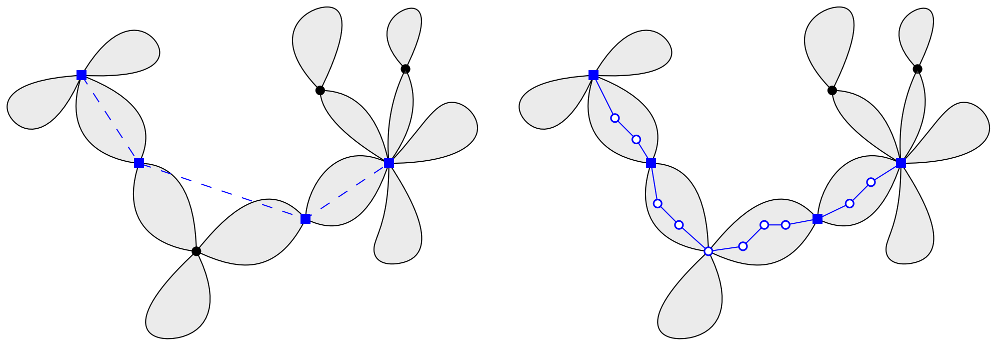

Figure 4 Step 2 of the Branching Phase. Left: the tree T'_Z. Right: the tree T_Z.

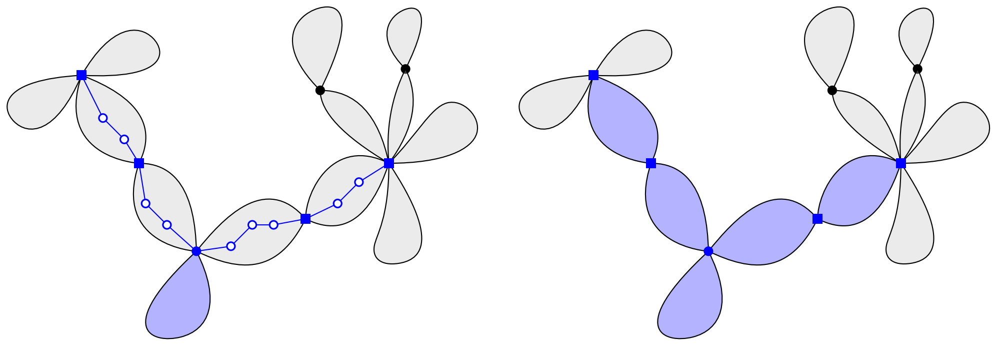

Figure 5 Step 3 of the Branching Phase. Left: our knowledge about F' after guessing the blocks in $\mathcal{B}_{l_3}$. Right: our knowledge about F' after guessing the blocks in $\mathcal{B}_{in}$.

- For each block in $\mathcal{B}_{l_1}$, we have guessed its cutvertex and no other vertices.
- For each block in $\mathcal{B}_{l_2}$, we have not guessed any vertices.
- For each block in $\mathcal{B}_{l_3}$, we have guessed its cutvertex v connecting it to the rest of F' and no other vertices; note that v is not a terminal. Moreover, for each such v there is at most one block in $\mathcal{B}_{l_3}$.
- For each block in $\mathcal{B}_w$, we have guessed its cutvertex v that does not belong to a block in $\mathcal{B}_{l_2}$ and we guessed no other vertices. Thus, for each double-block in $\mathcal{B}_d$, we have guessed its cutvertex connecting it to the rest of F' and no other vertices.
- For each block in $\mathcal{B}_{in}$, we have guessed at least two vertices.

Now we proceed to the final guessing step, see Fig. 5. First, we guess all blocks in $\mathcal{B}_{l_3}$. Note that we can do it, as (i) we know their cutvertices, (ii) the number of these cutvertices is at most $|V(T)| \leq 24ds^2$, (iii) each cutvertex is contained in at most one block from $\mathcal{B}_{l_3}$, and (iv) each block has at most d vertices. This results in at most $n^{\mathcal{O}(|V(T)|\cdot d)} = n^{\mathcal{O}(d^2s^2)}$ branches.

Next, we guess all blocks in $\mathcal{B}_{in}$. Again, we can do it as (i) we know at least two vertices of such a block, (ii) the number of these blocks is at most $|E(T)| \leq 24ds^2$, and (iii) each block has at most d vertices. This results in at most $n^{\mathcal{O}(|V(T)|\cdot d)} = n^{\mathcal{O}(d^2s^2)}$ further branches.

The following claim summarizes the outcome of the guessing phase of the algorithm.

▷ Claim A. In time $\mathcal{O}(n^{\mathcal{O}(s^2d^2)})$ we can enumerate a collection $\mathcal{S}$ of $\mathcal{O}(n^{\mathcal{O}(s^2d^2)})$ triples (S, C_1, C_2), where $S \subseteq V$ and $C_1, C_2 \subseteq S$ such that $\mathcal{S}$ has the following property. Let $X \subseteq V$, such that $F = G[X]$ is a $\mathcal{C}$-block graph. Let $X' \subseteq X$ be the vertex set of the graph F' obtained from F by removing all trivial components. Then there is at least one triple $(S, C_1, C_2) \in \mathcal{S}$, where

a) C_1 is the set of terminals of type 1 in F',
b) C_2 is the set of terminals of type 2 in F',
c) $G[S]$ is the skeleton of F'.

Completion Phase. Let $\mathcal{S}$ be the the collection from Claim A and let $(S, C_1, C_2) \in \mathcal{S}$ be a triple that satisfies the properties listed in the statement of Claim A for an optimum solution $F = G[X]$. Let $\mathcal{X} := \mathcal{X}_0 \cup \mathcal{X}_1 \cup \mathcal{X}_2$ be the family of subsets of V with:

$$\mathcal{X}_0 := \{\{v\} \mid v \in V\},$$
$$\mathcal{X}_1 := \{B \subseteq V \mid G[B] \in \mathcal{C}\}, \text{and}$$
$$\mathcal{X}_2 := \{B \subseteq V \mid B \text{ is a double-block whose blocks are in } \mathcal{C}\},$$

Let $G^{\mathcal{C}}$ be the graph whose vertex set is $\mathcal{X}$, and edges join sets that are adjacent in G. Furthermore, we define a weight function $\mathfrak{w}^{\mathcal{C}} \colon \mathcal{X} \to \mathbb{Q}^+$ as

$$\mathfrak{w}^{\mathcal{C}}(A) = \sum_{v \in A} \mathfrak{w}(v).$$

Note that in order to complete S to the optimum solution $F = G[X]$, we need to determine:

- all blocks in $\mathcal{B}_{l_1}$,
- all double-blocks in $\mathcal{B}_d$,
- all trivial components of F.

Note that the vertex sets of all these subgraphs are in the family $\mathcal{X}$ and they form an independent set in $G^{\mathcal{C}}$. Furthermore, since X is of maximum weight, the total weight of selected subsets must be maximized. Thus the idea behind the last step is to reduce the problem to solving Max Weight Independent Set in an appropriately defined subgraph of $G^{\mathcal{C}}$ and weights $\mathfrak{w}^{\mathcal{C}}$.

To ensure that the selected subsets are consistent with our guess $(S, C_1, C_2) \in \mathcal{S}$, we will remove certain vertices from $G^{\mathcal{C}}$. In particular, let $\mathcal{X}'$ consist of the sets $A \in \mathcal{X}$, such that:

1. $A \in \mathcal{X}_0 \cup \mathcal{X}_1$ and A is non-adjacent to S; these are the candidates for trivial components of F,
2. $A \in \mathcal{X}_1$ and A intersects S in exactly one vertex, which is in C_1; these are the candidates for blocks in $\mathcal{B}_{l_1}$,
3. $A \in \mathcal{X}_2$ and A intersects S in exactly one vertex, which is in C_2 and is not the cutvertex of $G[A]$; these are the candidates for double-blocks in $\mathcal{B}_d$.

Now let $\mathcal{I} \subseteq \mathcal{X}'$ be an independent set of $G^{\mathcal{C}}$, and let $S' = \bigcup_{A \in \mathcal{I}} A$. It is straightforward to verify that if $(S, C_1, C_2) \in \mathcal{S}$ satisfies the properties listed in Claim A, then $G[S \cup S']$ is a $\mathcal{C}$-block graph. Thus, in one of the branches, we will find the optimum solution $F = G[X]$.

Now let us argue that the last step can be performed in polynomial time. First, observe that $|\mathcal{X}| \leq n + n^d + n^{2d} = n^{\mathcal{O}(d)}$ and the family $\mathcal{X}$ can be exhaustively enumerated in time $n^{\mathcal{O}(d)}$. Next, $\mathcal{X}'$ can be computed in time polynomial in $|\mathcal{X}|$, and thus in n. This implies that the graph $G^{\mathcal{C}}[\mathcal{X}']$ can be computed in time polynomial in n. We observe that $G^{\mathcal{C}}$, and thus $G^{\mathcal{C}}[\mathcal{X}']$, is an induced subgraph of the blob graph G°, introduced in Section 2. Hence, by Theorem 3, we conclude that $G^{\mathcal{C}}[\mathcal{X}']$ is sP_3-free.

The final ingredient is the polynomial-time algorithm for Max Weight Independent Set in sP_3-free graphs by Brandstädt and Mosca [6]. Its running time on an n'-vertex graph is $n'^{\mathcal{O}(s)}$. Since the number of vertices of $G^{\mathcal{C}}[\mathcal{X}']$ is $n^{\mathcal{O}(d)}$, we conclude that a maximum-weight independent set in $G^{\mathcal{C}}[\mathcal{X}']$ can be found in time $n^{\mathcal{O}(sd)}$.

Summing up, in the guessing phase, in time $n^{\mathcal{O}(s^2d^2)}$ we enumerate the family $\mathcal{S}$ of size $n^{\mathcal{O}(s^2d^2)}$. Then, for each member (S, C_1, C_2) of $\mathcal{S}$, we try to extend the partial solution to a complete one. This takes time $n^{\mathcal{O}(sd)}$ per element of $\mathcal{S}$. Among all found solutions, we return the one with maximum weight. The total running time of the algorithm is $n^{\mathcal{O}(s^2d^2)}$, which is polynomial in n, since s and d are constants. This completes the proof of Theorem 1.

4 More Results for Even Cycle Transversal on H-Free Graphs

In this section we prove that subject to a number of unsolved cases, the complexity of Even Cycle Transversal for H-free graphs coincides with the one for Feedback Vertex Set.

$\mathsf{CMSO_2}$ and Even Cycle Transversal. *Monadic Second-Order Logic* ($\mathsf{MSO_2}$) over graphs consists of formulas with vertex variables, edge variables, vertex set variables, and edge set variables, quantifiers, and standard logic operators. We also have a predicate $\mathsf{inc}(v, e)$, indicating that the vertex v belongs to the edge e. *Counting Monadic Second-Order Logic* ($\mathsf{CMSO_2}$) is an extension of $\mathsf{MSO_2}$ which allows atomic formulas of the form $|X| \equiv p \bmod q$, where X is a set variable and $0 \leq p < q$ are integers.

Abrishami et al. [1, Theorems 5.3 and 7.3] proved that for any fixed $\mathsf{CMSO_2}$ formula Φ and any constant t, the following problem is polynomial-time solvable: given a P_5-free graph G with weight function $\mathfrak{w} : V(G) \to \mathbb{Q}^+$, find a maximum-weight set $X \subseteq V(G)$, such that $G[X]$ is of treewidth at most t and satisfies Φ. This immediately yields a polynomial-time algorithm for Feedback Vertex Set in P_5-free graphs: just take $t = 1$ and a trivial formula Φ that is satisfied for all graphs (see also [1]).

A similar argument can also be applied for Even Cycle Transversal. First, note that every odd cactus has treewidth at most 2. Hence, it remains to show an appropriate $\mathsf{CMSO_2}$ formula Φ that enforces $G[X]$ to be an odd cactus. We will again look from the complementary perspective: we need to say that $G[X]$ has no even cycle. For this, it is enough to say that there is no set E' of edges in $G[X]$, such that: (i) each vertex of X is incident to 0 or 2 edges from E', (ii) the edges from E' induce a connected subgraph of $G[X]$, and (iii) the number of edges in E' is even. Properties (i) and (ii) are easily expressible in $\mathsf{MSO_2}$, see [8, Section 7.4], and property (iii) is expressed by the formula $|E'| \equiv 0 \pmod 2$, which is allowed in $\mathsf{CMSO_2}$. This immediately yields the following corollary.

▶ **Corollary 7.** Even Cycle Transversal *is polynomial-time solvable for P_5-free graphs.*

Finally, the problem of finding a maximum-weight subset that induces a constant-treewidth graph satisfying some fixed $\mathsf{CMSO_2}$ formula can be solved in *quasipolynomial time* for P_r-free graphs for any fixed r [10]. This implies a quasipolynomial-time algorithm for Feedback Vertex Set and Even Cycle Transversal for H-free graphs if H is a linear forest.

Hardness Results. An *odd cycle factor* of a graph G is a set of odd cycles such that every vertex of G belongs to exactly one of them. The Odd Cycle Factor problem, which asks if a graph has an odd cycle factor, is known to be NP-complete [15]. The *line graph* $L(G)$ of a graph $G = (V, E)$ has vertex set E and an edge between two distinct vertices e and f if and only if e and f share an end-vertex in G.

The proof of our next result for line graphs is somewhat similar to a proof for Odd Cycle Transversal of [7] but uses some different arguments as well.

▶ **Theorem 8.** Even Cycle Transversal *is* NP*-complete for line graphs.*

Proof. Let $G = (V, E)$ be an instance of Odd Cycle Factor with n vertices and m edges. We claim that G has an odd cycle factor if and only if its line graph $L := L(G)$ has an even cycle transversal of size at most $m - n$, see Fig. 6.

First suppose G has an odd cycle factor. Then there is $E' \subseteq E$, such that $|E'| = n$ and $L[E']$ is a disjoint union of odd cycles. Hence, $S := E \setminus E'$ is an even cycle transversal of L of size $|E| - n = m - n$. Now suppose L has an even cycle transversal S with $|S| \leq m - n$. Let $E' := E \setminus S$, As $|E| = m$, we have $|E'| \geq n$.

We prove the following claim.

▷ Claim B. Every component of $L[E']$ is either an odd cycle or the line graph of a tree.

Proof. Let D be a component of $L[E']$. If D has no cycle, then D is a path, as L is a line graph and thus is claw-free. Hence, D is the line graph of a path, and thus a tree.

So suppose D has a cycle C. Then C is odd and induced, as $L[E']$ is an odd cactus. If D has no vertices except for the ones of C, then D is an odd cycle and we are done. Suppose otherwise.

First, assume that C has at least five vertices. Since D has vertices outside C, there is a vertex of C with a neighbour outside C. Hence, D contains either an even cycle or an induced claw, both of which are not possible. So now suppose that C has at most four vertices. Then C is a triangle, as D has no even cycles. Since D is an induced subgraph of L, there exists a subgraph T of G such that $D = L(T)$. As D is a connected graph with at least four vertices, containing a triangle, T is a connected graph with at least four vertices.

We aim to show that T is a tree. For contradiction, suppose that T contains a cycle C_T. Then C_T must be a triangle, as otherwise D would contain an even cycle or an odd cycle with at least five vertices. Let a, b, c be the vertices of C_T. As T is connected and has at least four vertices, at least one of $\{a, b, c\}$, say a, must have a neighbour $d \notin \{b, c\}$. However, the edges $ad - ab - bc - ac$ form a C_4 in D, a contradiction with D being an odd cactus. So we conclude that T contains no cycles and thus T is a tree. ◁

Each component of $L[E']$ that is an odd cycle corresponds to an odd cycle in G. By Claim B, each component D of $L[E']$ that is not an odd cycle is the line graph of some subtree T of G. So, if D has r vertices, then T has $r + 1$ vertices. Furthermore, the vertex sets of G corresponding to distinct components of $L[E']$ are pairwise disjoint. Suppose that $L[E']$ has $p \geq 0$ components that are not odd cycles. Let Q be the set of vertices incident to at least one edge of E'. Then $n = |V(G)| \geq |Q| = |E'| + p \geq n + p$. Hence, $p = 0$ and $|Q| = n$. So, the components of $L[E']$ correspond to an odd cycle factor of G. This completes the proof. ◀

We make a straightforward observation similar to an observation for Feedback Vertex Set [7, 14], except that we must subdivide edges of a graph an even number of times.

▶ **Theorem 9.** *For every $p \geq 3$,* Even Cycle Transversal *is* NP*-complete for graphs of girth at least p.*

Proof. We reduce from Even Cycle Transversal for general graphs by noting the following. Namely, the size of a minimum even cycle transversal in G is equal to the size of a minimum even cycle transversal in the graph G' obtained from G by subdividing every edge $2p$ times, and the girth of G' is at least p. ◀

The next theorem is analogous to the one for Feedback Vertex Set; see also Table 1.

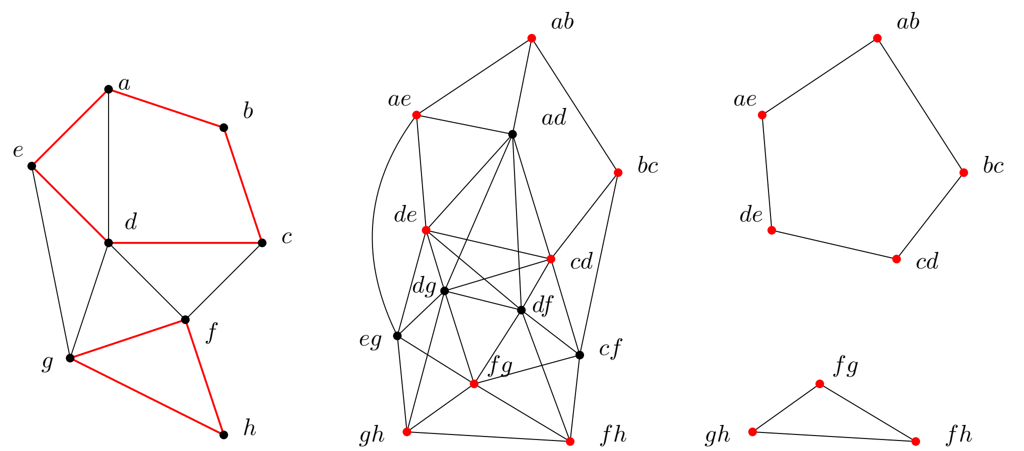

Figure 6 Left: a graph G with an odd cycle factor. Middle: the graph $L = L(G)$ and the set E' (red). Black vertices form an even cycle factor. Right: the odd cactus $L[E']$.

▶ **Theorem 10.** *Let H be a graph. Then* Even Cycle Transversal *for H-free graphs is polynomial-time solvable if $H \subseteq_i sP_3$ for some $s \geq 1$ or $H \subseteq_i P_5$, and it is* NP*-complete if H is not a linear forest.*

Proof. If $H \subseteq_i sP_3$, use Corollary 2, and if $H \subseteq_i P_5$, use Corollary 7. If H is not a linear forest, then it has a cycle or a claw. If H has a cycle, then we apply Theorem 9 for $p = |V(H)| + 1$. Otherwise, H has an induced claw and we apply Theorem 8. ◀

5 Conclusions

We prove that for a large family of graphs $\mathcal{F}$, the Min $\mathcal{F}$-Transversal problem is polynomial-time solvable on sP_3-free graphs (for every $s \geq 1$). The two best-known problems in this framework are Feedback Vertex Set and Even Cycle Transversal. Our result for Feedback Vertex Set generalizes two known results from the literature [7, 9]. We also prove that in contrast to the situation for Odd Cycle Transversal, all other known complexity results for Feedback Vertex Set on H-free graphs hold for Even Cycle Transversal as well. Hence, so far both problems behave the same on special graph classes, and it would be interesting to prove polynomial equivalency of the two problems more generally. Table 1 still shows some missing cases for each of the three problems.

In particular, we highlight a borderline case:

Is each of the three problems is polynomial-time solvable for $(P_1 + P_4)$-free graphs?

The main obstacle is that we know no polynomial-time algorithm for finding a maximum induced disjoint union of stars in a $(P_1 + P_4)$-free graph; note that such a subgraph could be a potential optimal solution for each of the three problems.

We also recall that Feedback Vertex Set and Even Cycle Transversal can be solved in quasipolynomial time for P_r-free graphs [10] for every $r \geq 1$, whereas Odd Cycle Transversal is NP-complete even for P_6-free graphs [9]. An affirmative answer to the above question for Feedback Vertex Set and Even Cycle Transversal would be a first step in proving that these two problems are polynomial-time solvable on P_6-free graphs. If that turns out to be the case, then we will have further evidence that these two problems, restricted to H-free graphs, differ in their complexity from Odd Cycle Transversal.

References

1 Tara Abrishami, Maria Chudnovsky, Marcin Pilipczuk, Paweł Rzążewski, and Paul Seymour. Induced subgraphs of bounded treewidth and the container method. *Proc. SODA 2021*, pages 1948–1964, 2021.

2 Yuuki Aoike, Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, Yusuke Kobayashi, Kazuhiro Kurita, and Yota Otachi. An improved deterministic parameterized algorithm for cactus vertex deletion. *CoRR*, abs/2012.04910, 2020. `arXiv:2012.04910`.

3 Benjamin Bergougnoux, Édouard Bonnet, Nick Brettell, and O-Joung Kwon. Close relatives of feedback vertex set without single-exponential algorithms parameterized by treewidth. *Proc. IPEC 2020, LIPIcs*, 180(3):1–17, 2020.

4 Édouard Bonnet, Nick Brettell, O-Joung Kwon, and Dániel Marx. Parameterized vertex deletion problems for hereditary graph classes with a block property. *Proc. WG2016, LNCS*, 9941:233–244, 2016.

5 Andreas Brandstädt and Dieter Kratsch. On the restriction of some NP-complete graph problems to permutation graphs. *Proc. FCT 1985, LNCS*, 199:53–62, 1985.

6 Andreas Brandstädt and Raffaele Mosca. Maximum weight independent set for l-claw-free graphs in polynomial time. *Discrete Applied Mathematics*, 237:57–64, 2018.

7 Nina Chiarelli, Tatiana R. Hartinger, Matthew Johnson, Martin Milanič, and Daniël Paulusma. Minimum connected transversals in graphs: New hardness results and tractable cases using the price of connectivity. *Theoretical Computer Science*, 705:75–83, 2018.

8 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

9 Konrad K. Dabrowski, Carl Feghali, Matthew Johnson, Giacomo Paesani, Daniël Paulusma, and Paweł Rzążewski. On cycle transversals and their connected variants in the absence of a small linear forest. *Algorithmica*, 82(10):2841–2866, 2020.

10 Peter Gartland, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Paweł Rzążewski. Finding large induced sparse subgraphs in $C_{>t}$ -free graphs in quasipolynomial time. *Proc. STOC 2021, ACM*, pages 330–341, 2021.

11 Sudeshna Kolay, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Quick but odd growth of cacti. *Algorithmica*, 79:271–290, 2017.

12 Pranabendu Misra, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Parameterized algorithms for even cycle transversal. *Proc. WG 2012*, 7551:172–183, 2012.

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

14 Svatopluk Poljak. A note on stable sets and colorings of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 15:307–309, 1974.

15 O. Vornberger. Komplexeität von Wegeproblemen in Graphen. *Reihe Theoretische Informatik*, 5, 1979.