On the Complexity of Establishing Hereditary Graph Properties via Vertex Splitting

Alexander Firbas
TU Wien, Austria

Manuel Sorge
TU Wien, Austria

Abstract

Vertex splitting is a graph operation that replaces a vertex v with two nonadjacent new vertices u, w and makes each neighbor of v adjacent with one or both of u or w. Vertex splitting has been used in contexts from circuit design to statistical analysis. In this work, we generalize from specific vertex-splitting problems and systematically explore the computational complexity of achieving a given graph property Π by a limited number of vertex splits, formalized as the problem Π Vertex Splitting (Π-VS). We focus on hereditary graph properties and contribute four groups of results: First, we classify the classical complexity of Π-VS for graph properties characterized by forbidden subgraphs of order at most 3. Second, we provide a framework that allows one to show NP-completeness whenever one can construct a combination of a forbidden subgraph and prescribed vertex splits that satisfy certain conditions. Using this framework we show NP-completeness when Π is characterized by sufficiently well-connected forbidden subgraphs. In particular, we show that F-Free-VS is NP-complete for each biconnected graph F. Third, we study infinite families of forbidden subgraphs, obtaining NP-completeness for Bipartite-VS and Perfect-VS, contrasting the known result that Π-VS is in P if Π is the set of all cycles. Finally, we contribute to the study of the parameterized complexity of Π-VS with respect to the number of allowed splits. We show para-NP-hardness for K_3-Free-VS and derive an XP-algorithm when each vertex is only allowed to be split at most once, showing that the ability to split a vertex more than once is a key driver of the problems' complexity.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases NP-completeness, polynomial-time solvability, graph theory, graph transformation, graph modification

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.30

Related Version *Full Version*: https://arxiv.org/abs/2401.16296 [15]

Funding Alexander Firbas acknowledges support from the Vienna Science and Technology Fund (WWTF) [10.47379/ICT22029]. Manuel Sorge acknowledges partial support from the Alexander von Humboldt Foundation.

1 Introduction

Vertex splitting is the graph operation in which we take a vertex v, remove it from the graph, add two *descendants* v_1, v_2 of v, and make each former neighbor of v adjacent with v_1, v_2, or both. Vertex splitting has been used in circuit design [28, 31], the visualization of nonplanar graphs in a planar way [2, 4, 11, 12, 24, 30], improving force-based graph layouts [10], in graph clustering with overlaps [1, 3, 5, 14], in statistics [9, 20] (see [14]), in subgraph counting [18, 32], and variants of vertex splitting in which we may make the copies adjacent play roles in graph theory [25, 29], in particular in Fleischner's Splitting Lemma [16] and in Tutte's theorem relating wheels and general three-connected graphs [33]. Such a variant of vertex splitting can also be thought of as an inverse operation of vertex contraction, which is an underlying operation of the graph parameters twinwidth (see, e.g., [6]) and fusion-width [7, 17].

In some of the above applications, we are given a graph and want to establish a graph property by splitting the least number of times: In circuit design, we aim to bound the longest path length [28, 31], when visualizing non-planar graphs we aim to establish planarity [2, 11, 12, 30] or pathwidth one [4], and in statistics and when clustering with overlaps we want to obtain a cluster graph (a disjoint union of cliques) [1, 3, 9, 14, 20].

This motivates generalizing these problems by letting Π be any graph property (a family of graphs) and studying the problem Π Vertex Splitting (Π-VS): Given a graph G and an integer k, is it possible to apply at most k vertex split operations to G to obtain a graph in Π? The above-mentioned graph properties are closed under taking induced subgraphs and thus we mainly focus on this case. For graph operations different from vertex splitting the complexity of establishing graph properties Π is well studied, such as for deleting vertices (e.g., [26, 27]), adding or deleting edges (see the recent survey [8]), or edge contractions (e.g., [19, 21–23]). In this work, we aim to start this direction for vertex splitting, that is, how can we characterize for which graph properties Π-VS is tractable? Our main focus here is the classical complexity, that is, NP-hardness vs. polynomial-time solvability, but we also touch on the parameterized complexity with respect to the number of allowed splits.

Our results are as follows. Each graph property Π that is closed under taking induced subgraphs is characterized by a family $\mathcal{F}$ of *forbidden induced subgraphs*. We also write Π as $\text{Free}_{\prec}(\mathcal{F})$. It is thus natural to begin by considering small forbidden subgraphs. We classify for each family $\mathcal{F}$ that contains graphs of order at most 3 whether $\text{Free}_{\prec}(\mathcal{F})$-VS is polynomial-time solvable or NP-complete. Indeed, it is NP-complete precisely if $\mathcal{F}$ contains only the path P_3 on three vertices or a triangle K_3:

▶ **Theorem 1.1 (★).** *Let $\mathcal{F}$ be a set of graphs containing graphs of at most three vertices each. Then, $\text{Free}_{\prec}(\mathcal{F})$-VS is NP-complete if $\mathcal{F} = \{P_3\}$ or $\mathcal{F} = \{K_3\}$ and is in P otherwise.*

The polynomial-time results use a plethora of different approaches and also extend to Threshold-VS and Split-VS. The NP-hardness for $\Pi = \text{Free}_{\prec}(\{K_3\})$ can be shown using a reduction from the Vertex Cover problem. In this reduction, we replace each edge of a graph by a K_3. It is then not hard to show a correspondence between splitting a set of at most k vertices to destroy all induced K_3 and a vertex cover of size at most k for the original graph. Together with our results below, we also obtain NP-completeness for each connected forbidden subgraph F with four vertices except for P_4s and claws $K_{1,3}$, for which the complexity remains open.

Second, the hardness construction for K_3-free graphs indicates that high connectivity in forbidden subgraphs makes Π-VS hard and thus we explored this direction further. As the naïve approach breaks down in the general setting, we reduce from a special variant of Vertex Cover and develop a framework for showing NP-hardness of Π-VS whenever one can use forbidden induced subgraphs to construct certain splitting configurations (Section 3). That is, a graph H together with a recipe specifying distinguished vertices that will be connected to the outside of H and how to split them. Essentially, if one can provide a splitting configuration that avoids introducing new forbidden subgraphs and that decreases the connectivity to the outside well enough, then we can use such a configuration to give a hardness construction. We then provide ways to obtain such splitting configurations, allowing us to show the following hardness results, where we write $\text{Free}_{\subseteq}(\mathcal{F})$ to exclude the graphs in $\mathcal{F}$ as subgraphs, rather than induced subgraphs:

▶ **Theorem 1.2 (★).** *Let $\mathcal{F}$ be a family of graphs.*

1. *If $\mathcal{F}$ consists of a single biconnected graph, then $\text{Free}_{\prec}(\mathcal{F})$-VS and $\text{Free}_{\subseteq}(\mathcal{F})$-VS are NP-complete.*

2. *If all graphs in $\mathcal{F}$ are triconnected and the family has bounded diameter, then* Free$_\prec(\mathcal{F})$-*VS and* Free$_\subseteq(\mathcal{F})$-*VS are* NP*-hard.*
3. *If all graphs in $\mathcal{F}$ are 4-connected, then* Free$_\prec(\mathcal{F})$-*VS and* Free$_\subseteq(\mathcal{F})$-*VS are* NP*-hard.*

NP*-completeness in item 2 and 3 holds when* Free$_{\prec/\subseteq}(\mathcal{F})$ *is decidable in polynomial time.*

Third, the above results do not cover the case where $\mathcal{F}$ is the family of all cycles, and this must be so because Forest-VS is polynomial-time solvable [4, 13]. However, we show that if we forbid only cycles of at most a certain length, or all cycles of odd length, then Π-VS becomes NP-complete again. This hardness extends also to perfect graphs:

▶ **Theorem 1.3 (★).** *Bipartite-VS and Perfect-VS are* NP*-complete.*

The hardness construction for Bipartite Vertex Splitting is similar to the one for Free$_\prec(\{K_3\})$-VS mentioned above. The nontrivial part of the proof is, given a vertex cover, how to split vertices such that the resulting graph is two-colorable. By carefully checking all the possible configurations of vertices in the vertex cover and splitting them in the right way, we obtain subconfigurations that are two-colorable and whose colorings can be combined into a two-coloring for the whole graph. The reduction for Perfect Vertex Splitting uses five-vertex cycles instead of K_3's and the correctness additionally uses a degree-based argument to show that the graph after splitting is also odd-anti-hole-free and thereby perfect.

Finally, we contribute to the parameterized complexity of Π-VS with respect to the number k of allowed vertex splits. Previously it was known that Π-VS is fixed-parameter tractable when Π is closed under taking minors [30], when $\Pi = \text{Free}_\prec(\{P_3\})$ [13, 14], and when Π consists of graphs of pathwidth one or when Π is MSO_2-definable and of bounded treewidth [4]. In contrast, we observe that Free$_\prec(\{K_3\})$-VS is NP-hard even for $k = 2$:

▶ **Theorem 1.4 (★).** Free$_\prec(\{K_3\})$*-VS is* NP*-complete for two splits.*

The idea behind this result is to reduce from 3-Coloring on K_3-free graphs: Barring the technical details, we add a universal vertex u to a graph G, and gadgets to ensure that the vertex u must be split. Then, the constraint that two adjacent vertices v, w in G need to be colored differently translates to the constraint that v and w need to be adjacent to different descendants of u. The crux herein is that one can split a vertex multiple times: In contrast, if we instead can split each vertex at most once, resulting in the problem Shallow Triangle-Free Vertex Splitting, then we obtain an XP algorithm:

▶ **Theorem 1.5 (★).** *Shallow Triangle-Free Vertex Splitting, parameterized by the number k of splits, admits an $O(\sqrt{2}^{k^2} \cdot n^{k+3})$-time* XP *algorithm.*

The basic idea is that we can guess which vertices are split and how they are split with respect to each other. Formulating the condition that the guess was correct can then be done using a 2-SAT formula.

Due to space constraints, we only provide details for the dichotomy result for small forbidden subgraphs (Theorem 1.1) and the framework for proving Theorem 1.2. All remaining results are marked with ★ and are proved in the full version of this paper [15].

1.1 Preliminaries

General (Graph) Notation. For a function $f\colon A \to B$, we let $\text{Domain}(f) := A$ and $\text{Range}(f) := \{b \mid \exists a \in A\colon f(a) = b\}$. For a set X, we let $\mathcal{P}(X)$ be its power set. Unless stated otherwise, all graphs are undirected and without parallel edges or self-loops. Let G

be a graph with vertex set $V(G)$ and edge set $E(G)$. We denote the neighborhood of a vertex $v \in V(G)$ by $N_G(v)$. The graph induced by a vertex set $V' \subseteq V(G)$ is written as $G[V']$. For $u, v \in V(G)$, we write uv as a shorthand for $\{u, v\}$, $G - v$ for $G[V(G) \setminus \{v\}]$, $\deg_G(v)$ for $|N_G(v)|$, $d_G(u, v)$ for the length of a shortest path from u to v if there is one and ∞ otherwise, and $\operatorname{diam}(G)$ for the diameter of G, that is, $\max_{u,v \in V(G)} d_G(u, v)$. We denote the complement of G by $\overline{G}$. The graph K_n is the complete graph on n vertices and C_n the cycle graph of n vertices. If a graph G is isomorphic to a graph H, we write $G \simeq H$. The *circumference* of a graph G is the length of a longest cycle of G if G it is not acyclic and zero otherwise. A *k-subdivision* of a graph G is a graph obtained by replacing each of G's edges uv with a path $u, p_1^{uv}, p_2^{uv}, \ldots, p_k^{uv}, v$, where $p_1^{uv}, p_2^{uv}, \ldots, p_k^{uv}$ are new vertices. We mark directed graphs $\vec{G}$ with an arrow. For an arc $uv \in E(\vec{G})$, u is the source vertex and v is the target vertex. All directed graphs are oriented, that is, for each $uv \in E(\vec{G})$, we have $vu \notin E(\vec{G})$. We denote the in-neighborhood by $N^-_{\vec{G}}(\cdot)$ and the out-neighborhood by $N^+_{\vec{G}}(\cdot)$. A directed graph $\vec{G}$ is an *orientation* of G if the underlying undirected graph of $\vec{G}$ is G.

Vertex Splitting. Let G be a graph, $v \in V(G)$, and V_1, V_2 subsets of $N_G(v)$ such that $V_1 \cup V_2 = N_G(v)$. Furthermore, let v_1 and v_2 denote two fresh vertices, that is, $\{v_1, v_2\} \cap V(G) = \varnothing$. Consider the graph G' that is obtained from G by deleting v, and adding v_1 and v_2 such that $N_{G'}(v_1) = V_1$ and $N_{G'}(v_2) = V_2$. Then, we say G' was obtained from G by *splitting* v (via a *vertex split*). If $V_1 \cap V_2 = \varnothing$, we speak of a *disjoint* vertex split, and if either $V_1 = \varnothing$ or $V_2 = \varnothing$, we say the split is *trivial*. Furthermore, we say v was *split into* v_1 and v_2, and call these vertices the *descendants* of v. Conversely, v is called the *ancestor* of v_1 and v_2. Finally, consider an edge v_1w (resp. v_2w) of G'. We say that the edge vw of G was *assigned* to v_1 (resp. v_2) in the split, and call v_1w (resp. v_2w) a *descendant edge* of vw.

A *splitting sequence* of k splits is a sequence of graphs $G_0, G_1, \ldots, G_k$, such that G_{i+1} is obtainable from G_i via a vertex split for $i \in \{0, \ldots, k-1\}$. The notion of descendant vertices (resp. ancestor vertices) is extended in a transitive and reflexive way (that is, a vertex is its own ancester and descendant) to splitting sequences.

Later, the following shorthand notation will be useful: Let H be a graph, $v \in V(H)$, $X_1, X_2 \subseteq N_H(v)$ with $X_1 \cup X_2 = N_H(v)$, and v_1, v_2 two distinct vertices. Further, let H' be the graph obtained by splitting v into v_1 and v_2 while setting $N_{H'}(v_1) = X_1$, $N_{H'}(v_2) = X_2$. Then, we identify H' with the shorthand $\operatorname{Split}(H, v, X_1, X_2, v_1, v_2)$.

Embeddings and Hereditary Graph Properties. For graphs G and H, we write $\operatorname{Emb}_\prec(G, H)$ (resp. $\operatorname{Emb}_\subseteq(G, H)$) to denote the set of all *induced embeddings* of G in H (resp. *subgraph embeddings*), that is, the set of all injective $f\colon V(G) \to V(H)$ where $\forall uv \in V(G)^2\colon uv \in E(G) \iff f(u)f(v) \in E(H)$ (resp. $\forall uv \in V(G)^2\colon uv \in E(G) \implies f(u)f(v) \in E(H)$). In case $\operatorname{Emb}_\prec(G, H) \neq \varnothing$ (resp. $\operatorname{Emb}_\subseteq(G, H) \neq \varnothing$), we write $G \prec H$ (resp. $G \subseteq H$) and say G is an *induced subgraph* (resp. a *subgraph*) of H.

For a set of graphs $\mathcal{F}$, we write $\operatorname{Free}_\prec(\mathcal{F})$ (resp. $\operatorname{Free}_\subseteq(\mathcal{F})$) to denote the set of graphs where $G \in \operatorname{Free}_\prec(\mathcal{F})$ (resp. $G \in \operatorname{Free}_\subseteq(\mathcal{F})$) iff $\operatorname{Emb}_\prec(F, G) = \varnothing$ (resp. $\operatorname{Emb}_\subseteq(F, G) = \varnothing$) for all $F \in \mathcal{F}$. Set $\mathcal{F}$ is the set of *forbidden induced subgraphs* (resp. *forbidden subgraphs*) that *characterize* the *hereditary (graph) property* $\operatorname{Free}_\prec(\mathcal{F})$ (resp. $\operatorname{Free}_\subseteq(\mathcal{F})$).

2 Properties Characterized by Small Forbidden Induced Subgraphs

We now give an outline of the characterization of Π-VS for Π characterized by families $\mathcal{F}$ of forbidden induced subgraphs with at most three vertices. The full version of this section is given in the full version of this paper [15]. First, we can make several simple observations: If

one of K_0, K_1, K_2, or $\overline{K_2}$ is forbidden and it is present in the input graph, then there is no way to destroy these forbidden subgraphs with vertex splitting and hence we can immediately return a failure symbol. This gives a trivial algorithm if $K_0 \in \mathcal{F}$ or $K_1 \in \mathcal{F}$. Moreover, if $\overline{K_2} \in \mathcal{F}$, then the input graph is a clique or we can return failure. Since splitting introduces a $\overline{K_2}$, instance (G, k) is positive if and only if $(G, 0)$ is positive, which we can check in polynomial time. Similarly, if $K_2 \in \mathcal{F}$, then the input graph is an independent set or we can return failure. Through splitting, we can only introduce more independent vertices and thus (G, k) is positive if and only if $(G, 0)$ is positive.

It follows that we can focus on families $\mathcal{F}$ that contain subgraphs with exactly 3 vertices, that is, $\mathcal{F} \subseteq \{P_3, \overline{P_3}, K_3, \overline{K_3}\}$. If $\mathcal{F}$ contains $\overline{P_3}$ or $\overline{K_3}$ but neither P_3 nor K_3, then we have a similar observation as above: $\overline{P_3}$ and $\overline{K_3}$ cannot be destroyed by vertex splits and thus (G, k) is positive if and only if $(G, 0)$ is, which is checkable in polynomial time.

It thus remains to classify families $\mathcal{F} \subseteq \{P_3, \overline{P_3}, K_3, \overline{K_3}\}$ that contain P_3 or K_3. If $\mathcal{F} = \{P_3\}$ then $\text{Free}_{\prec}(\mathcal{F})$-VS is NP-complete by a result of Firbas et al. [14, Theorem 4.4]. If $\mathcal{F} = \{K_3\}$ then NP-completeness follows from Theorem 1.2 or Theorem 1.4, which we prove below. However, if we combine P_3 and K_3 or if we add $\overline{P_3}$ and/or $\overline{K_3}$ then the problems once again become polynomial-time solvable for subtle and different reasons:

For $\mathcal{F} = \{P_3, K_3\}$ all solution graphs are the union of an independent set and a matching and there is essentially only one minimal sequence of vertex splits. In the case where $\{K_3, \overline{K_3}\} \subseteq \mathcal{F}$ we can apply Ramsey-type arguments to show that an algorithm only needs to check for a constant number of different yes-instances. If $\overline{P_3} \in \mathcal{F}$ we can observe that destroying any P_3 or K_3 necessarily introduces a $\overline{P_3}$, which cannot be removed afterwards.

This takes care of all cases for $\mathcal{F}$ except $\mathcal{F} = \{P_3, \overline{K_3}\}$. For this case we can observe that the graphs resulting from a splitting solution are *cluster graphs*, disjoint unions of cliques, with at most two clusters (cliques). As $\overline{K_3}$ cannot be destroyed by vertex splitting, the input graph may only contain P_3s. Furthermore, P_3s can only be destroyed by splitting their midpoints. It is thus intuitive that the input graph of a yes-instance must consist of two cliques that may overlap and, furthermore, the overlap must not exceed the number k of allowed splits. This is indeed what we can show and, moreover, such graphs can be recognized in polynomial time. This finishes the outline of our characterization and we obtain:

▶ **Theorem 1.1 (★).** *Let $\mathcal{F}$ be a set of graphs containing graphs of at most three vertices each. Then,* $\text{Free}_{\prec}(\mathcal{F})$*-VS is* NP*-complete if $\mathcal{F} = \{P_3\}$ or $\mathcal{F} = \{K_3\}$ and is in* P *otherwise.*

Our polynomial-time results for split- and threshold graphs (★) use the observation that destroying some of their forbidden subgraphs by splitting, namely P_4, C_4, or C_5, necessarily creates another forbidden subgraph $\overline{C_4}$, reducing the problem to checking whether the input graph has the respective property. This seems to be a general principle worthy of further exploration.

3 A General Framework to Show NP-hardness

In this section, we introduce a reduction framework and employ it to show NP-hardness of Π-VS characterized by a type of well-connected forbidden subgraphs. For each fixed $\ell \in \mathbb{N}$, consider the 2ℓ-Subdivided Cubic Vertex Cover problem: Given a tuple (G^*, k), where G^* is a 2ℓ-subdivision of a cubic graph G and $k \in \mathbb{N}$, is there a vertex cover C of G^* with $|C| \leq k$? The NP-hardness of this problem for each $\ell \in \mathbb{N}$ follows from a result by Uehara [34] and "folklore" techniques. Nevertheless, we provide a formal proof in the the full version of this paper [15].

Informally, our reduction works as follows: For a given set of forbidden subgraphs $\mathcal{F}$, to show the NP-hardness of either Free$_{\prec}(\mathcal{F})$-VS or Free$_{\subseteq}(\mathcal{F})$-VS, we reduce from 2ℓ-Subdivided Cubic Vertex Cover to the chosen problem. Here, ℓ will depend on the choice of $\mathcal{F}$.

Consider an instance (G, k) of the selected vertex cover problem. To build an instance of the vertex-splitting problem in question, we select some $H \in \mathcal{F}$ and designate two *"endpoint"* vertices of H. Then we replace each of G's edges with a copy of H (we call this copy an *edge gadget*) and keep k the same.

It is straightforward to see that, if one can split the constructed graph at most k times while destroying all forbidden graphs $\mathcal{F}$, one can find a corresponding vertex cover of G of size at most k: Analogous to how a vertex cover needs to "hit" each edge of G, the splits performed in the constructed graph need to destroy all the inserted forbidden copies of H.

The converse direction, that is, to show how a vertex cover of G can be used to destroy all forbidden subgraphs in the construction, is substantially more involved. Essentially, for each vertex in the vertex cover, we split in a particular way the corresponding vertex in the construction where edge gadgets meet (which we call an *attachment point*). This way, we can easily destroy all "original" embeddings of H. The hard part is to ensure that apart from these embeddings of H, by performing the construction and then the splits, no new embeddings of forbidden subgraphs of $\mathcal{F}$ are introduced.

To make the above outline precise, we first introduce the concept of a *splitting configuration*. Intuitively, a splitting configuration consists of a graph H, a selection of two of its vertices a and b, which we call H's *a-end* and *b-end*, and an encoding of a specific strategy of how to split a and b in H.

▶ **Definition 3.1.** *Let H be a graph, $a, b \in V(H)$ distinct vertices, $A_1, A_2 \subseteq N_H(a)$, and $B_1, B_2 \subseteq N_H(b)$, such that $A_1 \cup A_2 = N_H(a)$, $B_1 \cup B_2 = N_H(b)$, and A_1, A_2, B_1, B_2 are non-empty. Then, $(H, a, A_1, A_2, b, B_1, B_2)$ is called a* splitting configuration. *If $A_1 \cap A_2 = B_1 \cap B_2 = \varnothing$, we speak of a* disjoint splitting configuration. *Furthermore, we say the splitting configuration is* based on $\mathcal{F}$ *if $H \in \mathcal{F}$.*

Going forward, we aim to show how, depending on $\mathcal{F}$, one can find a suitable ℓ and a splitting configuration based on $\mathcal{F}$ such that the described reduction is guaranteed to be correct. We now make precise how to perform the construction. To specify which way (a-end, b-end) or (b-end, a-end) to insert the edge gadgets, we arbitrarily orient the graph G from the Vertex Cover instance to obtain a directed graph $\vec{G}$ as the "skeleton" graph. Furthermore, we also need a splitting configuration C (that also encodes the gadget-graph H and its a/b-ends to use). To simplify the correctness proof later on, we explain a more general construction than used in the reduction. That is, in addition, the construction takes a subset S of $\vec{G}$'s vertices as input; this set S specifies that the corresponding attachment points should be split in the construction according to the splitting configuration. When computing the reduction, we simply set $S = \varnothing$. See Figure 1 for a concrete example.

Towards defining Constr$(\vec{G}, C, S)$. Below, whenever we encounter a graph G' that is a copy of a graph G, we use $v^{G'}$ to denote the vertex that corresponds to $v \in V(G)$ in G'. We also do likewise for sets of vertices. Let $\vec{G}$ be a directed, oriented graph without loops, C a splitting configuration with $C = (H, a, A_1, A_2, b, B_1, B_2)$, and $S \subseteq V(G)$. We aim to define the graph Constr$(\vec{G}, C, S)$ and the map $\chi_{\text{Constr}(\vec{G},C,S)}$ which we will use to refer to particular subsets of vertices in the construction in the correctness proofs later on. For this we first define how to obtain the edge gadget graphs (H_e for each $e \in E(\vec{G})$), a map α that specifies

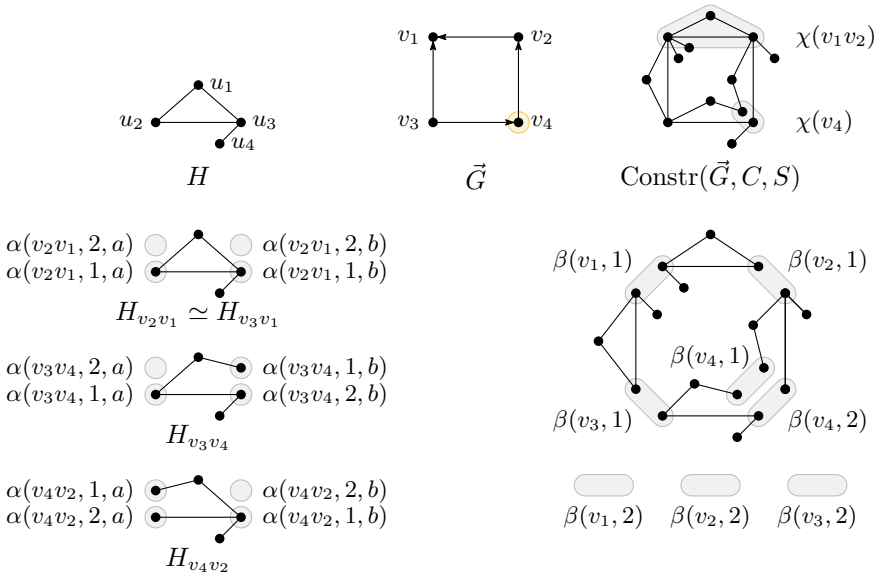

Figure 1 Example of Definition 3.2. The construction $\mathrm{Constr}(\vec{G}, C, S)$ is carried out for the "skeleton" graph $\vec{G}$, the splitting configuration C given by $(H, u_2, \{u_1\}, \{u_3\}, u_3, \{u_1\}, \{u_2, u_4\})$, and the set of vertices $S = \{v_4\}$ marked in yellow. The edge gadget graph is H; its "a-end" is u_2 and its "b-end" is u_3. The subscript of χ, $\mathrm{Constr}(\vec{G}, C, S)$, is dropped for brevity.

the attachment points inside the edge gadgets, and a map β that specifies which attachment points stemming from distinct edge gadgets should be merged to form the final attachment points where edge gadgets meet.

Towards defining H_e, with each arc $e = v_av_b \in E(\vec{G})$, we associate a fresh copy of H and call it H'_e. The vertices $a^{H'_e}, b^{H'_e}$ and the sets of vertices $A_1^{H'_e}, A_2^{H'_e}, B_1^{H'_e}, B_2^{H'_e}$ denote the corresponding vertex (resp. set of vertices) of H in its copy, H'_e. We obtain H_e by splitting a subset of $\{a^{H'_e}, b^{H'_e}\}$ in H'_e. Whether we split zero, one, or two vertices is dictated by S ($a^{H'_e}$ is split iff $v_a \in S$, $b^{H'_e}$ is split iff $v_b \in S$); the precise manner vertices are split is dictated by the splitting configuration C. More specifically, the neighborhoods of the descendant vertices of $a^{H'_e}$ (resp. $b^{H'_e}$) are given by $A_1^{H'_e}, A_2^{H'_e}$ (resp. $B_1^{H'_e}, B_2^{H'_e}$). With this, we can specify formally how H_e is obtained from each $e = v_av_b$ of $E(\vec{G})$:

$$H_e := \begin{cases} H'_e & \text{if } v_a \notin S, v_b \notin S, \\ \mathrm{Split}(H'_e, a^{H'_e}, A_1^{H'_e}, A_2^{H'_e}, a_1^{H_e}, a_2^{H_e}) & \text{if } v_a \in S, v_b \notin S, \\ \mathrm{Split}(H'_e, b^{H'_e}, B_1^{H'_e}, B_2^{H'_e}, b_1^{H_e}, b_2^{H_e}) & \text{if } v_a \notin S, v_b \in S, \text{ and} \\ \mathrm{Split}\big(\mathrm{Split}(H'_e, a^{H'_e}, A_1^{H'_e}, A_2^{H'_e}, a_1^{H_e}, a_2^{H_e}), & \\ \qquad b^{H'_e}, B_1^*, B_2^*, b_1^{H_e}, b_2^{H_e}\big) & \text{otherwise,} \end{cases}$$

where B_1^* (resp. B_2^*) denote the descendant vertices of $B_1^{H'_e}$ (resp. $B_2^{H'_e}$) with respect to the split described by $\mathrm{Split}(H'_e, a^{H'_e}, A_1^{H'_e}, A_2^{H'_e}, a_1^{H_e}, a_2^{H_e})$.[1] The set $\{H_e \mid e \in E(\vec{G})\}$ of edge gadgets provides the basic building blocks of $\mathrm{Constr}(\vec{G}, C, S)$. Note that the vertex sets of all H_e with $e \in E(\vec{G})$ are disjoint; to construct the final graph $\mathrm{Constr}(\vec{G}, C, S)$, we join the edge gadgets according to the structure of $\vec{G}$.

[1] This additional care is required to cover the case when a and b are neighbors in H.

For this purpose, we designate two numbered *attachment points* for the a-end, and two numbered attachment points for the b-end of each H_e, where an attachment point is a possibly empty subset of H_e's vertices. We denote the attachment points with the map $\alpha(\cdot,\cdot,\cdot)$, defined as follows: For a given $e \in E(\vec{G})$, $x \in \{a,b\}$, $i \in \{1,2\}$ we set

$$\alpha(e,i,x) \coloneqq \begin{cases} \{x_i^{H_e}\} & \text{if } v_x \in S, \\ \{x^{H_e}\} & \text{if } v_x \notin S \wedge i = 1, \text{ and} \\ \varnothing & \text{if } v_x \notin S \wedge i = 2. \end{cases}$$

To join the edge gadgets, we define two equivalence classes for each $v \in V(\vec{G})$, stemming from the circumstance that we have two attachment points per edge gadget end. The set of equivalence classes is given by $\bigcup_{v \in V(\vec{G})} \{\beta(v,1), \beta(v,2)\}$, where for each $v \in V(\vec{G})$ and $i \in \{1,2\}$, we define

$$\beta(v,i) \coloneqq \left\{ \left(\bigcup_{u \in N^-_{\vec{G}}(v)} \alpha(uv,i,b) \right) \cup \left(\bigcup_{u \in N^+_{\vec{G}}(v)} \alpha(vu,i,a) \right) \right\}.$$

See Figure 1 for a concrete example of $\alpha(\cdot,\cdot,\cdot)$ and $\beta(\cdot,\cdot)$.

▶ **Definition 3.2.** *The graph* $\mathrm{Constr}(\vec{G}, C, S)$ *is built by composing all* $(H_e)_{e \in E(\vec{G})}$ *into a single graph and merging all equivalent vertices into one representative vertex each.*

Later on, we will need to refer to specific vertex-subsets of the construction: For $G^* = \mathrm{Constr}(\vec{G}, C, S)$ and a given edge $e \in E(\vec{G})$, we write $\chi(e)_{G^*}$ for the set of vertices in G^* that stem from e's edge gadget (including the descendants of the gadgets "attachment"-vertices which are in general not unique to e); For a given vertex $v \in V(\vec{G})$, we write $\chi(v)_{G^*}$ to refer to either the set of the single "attachment"-vertex in G^* corresponding to v if $v \notin S$, and the two descendants of said vertex otherwise. See Figure 1 for a concrete example of $\chi(\cdot)$. A formal definition of χ is provided in the full version of this paper [15].

Abstracting from a single instantiation of our construction, we also introduce notation to capture the class of all possible constructions based on a given splitting configuration and an undirected graph together with all of its vertex covers.

▶ **Definition 3.3.** *Let* G *be a simple graph and* C *a splitting configuration. Then, we write* $\mathrm{AllConstr}(G, C)$ *to describe the set of all graphs* $\mathrm{Constr}(\vec{G}, C, S)$*, where* $\vec{G}$ *is an orientation of* G *and* $S \subseteq V(G)$ *is a vertex cover of* G*.*

3.1 Proving the Correctness of the Reduction

In this subsection, we define the property of *admissibility* for a splitting configuration C and show that, when using an admissible splitting configuration for the construction, the reduction outlined above is correct. The next subsection then deals with finding admissible splitting configurations for various classes of hereditary properties.

The backward direction of the correctness proof, that is, extracting a vertex cover from a splitting sequence that destroys all forbidden subgraphs, is straightforward and works independently of the choice of C and ℓ (★). However, the forward direction, where we use a vertex cover to find a splitting sequence that destroys all forbidden subgraphs in the construction, is more difficult. Here, the choice of C and ℓ will matter. We are given a vertex cover of the "skeleton graph" $\vec{G}$ and split all of the attachment points in the construction according to a corresponding splitting configuration. In the final graph of the splitting

sequence, the whole construction needs to be free of embeddings of forbidden (induced) subgraphs. This can be rephrased as two separate properties that a splitting configuration must guarantee when applying our construction to any conceivable instance of 2ℓ-Subdivided Cubic Vertex Cover and splitting it according to a vertex cover:

- There are no embeddings of forbidden (induced) subgraphs reaching from one edge gadget to a neighboring edge gadget.
- There are no embeddings of forbidden (induced) subgraphs contained entirely within any individual edge gadget.

In Definition 3.4, we formalize both these requirements. Note that the requirement on $\mathcal{F}$ to be of bounded diameter will serve to guarantee that a suitable L can be found.

▶ **Definition 3.4.** *Let $\mathcal{F}$ be a family of graphs of bounded diameter with $H \in \mathcal{F}$ and $C = (H, a, A_1, A_2, b, B_1, B_2)$ a splitting configuration. Furthermore, let $L \coloneqq 2 \cdot \max_{F \in \mathcal{F}} \operatorname{diam}(F)$. Then, C is called* separating *for $\mathcal{F}$ if for all graphs G that are an L-subdivision of some cubic graph, we have*

$$\forall G^* \in \operatorname{AllConstr}(G, C) \colon \forall F \in \mathcal{F} \colon \forall \pi \in \operatorname{Emb}_{\subseteq}(F, G^*) \colon \exists e \in E(G) \colon \operatorname{Range}(\pi) \subseteq \chi_{G^*}(e).$$

Furthermore, if $\operatorname{AllConstr}(K_2, C) \subseteq \operatorname{Free}_{\subseteq}(\mathcal{F})$, we say that C is intra-edge embedding-free *for $\mathcal{F}$. Finally, the splitting configuration C is called* admissible *for $\mathcal{F}$ if it is both separating for $\mathcal{F}$ as well as intra-edge embedding-free for $\mathcal{F}$.*

If such an admissible splitting configuration is known to exist, the converse direction of the correctness proof is straightforward (★). The following NP-hardness result follows directly by combining both directions:

▶ **Lemma 3.5 (★).** *Let $\mathcal{F}$ be a family of graphs of bounded diameter and let C be a splitting configuration admissible for $\mathcal{F}$. Then, $\operatorname{Free}_{\prec}(\mathcal{F})$-VS and $\operatorname{Free}_{\subseteq}(\mathcal{F})$-VS are NP-hard.*

3.2 Biconnected Forbidden Subgraphs and Beyond

We just established a method for obtaining NP-hardness for vertex-splitting problems, provided an appropriate admissible splitting configuration exists. This subsection addresses how to find such splitting configurations for the case of biconnected, triconnected, and 4-connected forbidden (induced) subgraphs.

Towards this goal, we define a last piece of notation: the *width* of a splitting configuration, denoted by $\operatorname{wdt}(\cdot)$, represents the minimum distance between the two descendants of a split endpoint, a and b, respectively, after H has been split according to the splitting configuration.

First, we deal with biconnectedness. To that end we show that, given a splitting configuration of a certain width that is not separating (for some family of graphs $\mathcal{F}$ of bounded diameter and circumference), we can derive a new splitting configuration of increased width (Lemma 3.6). Since we cannot apply this process ad infinitum (when restricted to $\mathcal{F}$ of bounded circumference), we will arrive at a separating splitting configuration (★).

▶ **Lemma 3.6.** *Let $\mathcal{F}$ be a family of biconnected graphs of bounded diameter and let $C^0 = (H^0, a^0, A_1^0, A_2^0, b^0, B_1^0, B_2^0)$ be a disjoint splitting configuration of finite width with $H^0 \in \mathcal{F}$ that is not separating for $\mathcal{F}$. Then, there exists a disjoint splitting configuration $C^1 = (H^1, a^1, A_1^1, A_2^1, b^1, B_1^1, B_2^1)$ with $H^1 \in \mathcal{F}$ of finite width satisfying $\operatorname{wdt}(C^1) > \operatorname{wdt}(C^0)$.*

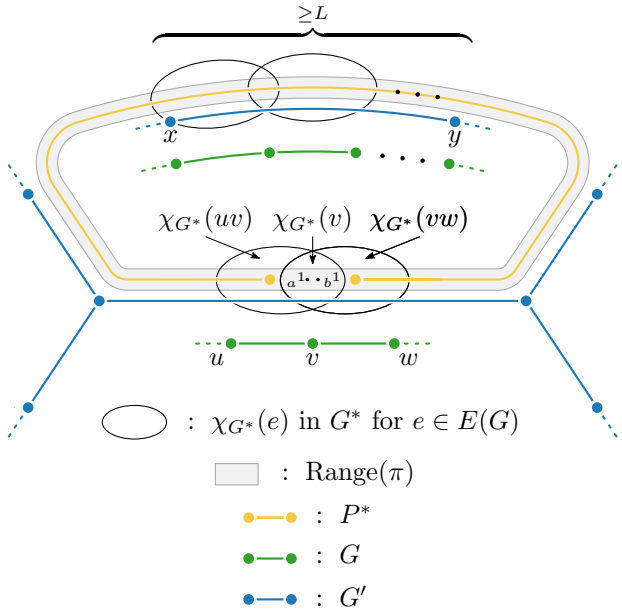

Figure 2 Illustration accompanying Lemma 3.6. The black ovals denote the edge gadgets in G^*, G is displayed in green, and the underlying graph G' is rendered in blue. The gray area shows the range of a hypothetical embedding π of F in G^*, that has to "go around" in the construction, since it cannot span across the intersection of edge gadgets $\chi_{G^*}(v)$. Additionally, in yellow, the path P^* traversing the embedding is shown.

Proof. As C^0 is not separating for $\mathcal{F}$, there is a graph G that is an L-subdivision of some cubic graph G', $G^* \in \text{AllConstr}(G, C^0)$, $F \in \mathcal{F}$ with $\pi \in \text{Emb}_{\subseteq}(F, G^*)$, as well as

$$L := 2 \cdot \max_{F \in \mathcal{F}} \text{diam}(F),$$

$$\text{Range}(\pi) \cap (\chi_{G^*}(uv) \setminus \chi_{G^*}(v)) \neq \varnothing, \text{ and}$$

$$\text{Range}(\pi) \cap (\chi_{G^*}(vw) \setminus \chi_{G^*}(v)) \neq \varnothing.$$

In other words, G^* is a graph constructed according to Definition 3.2 using a highly subdivided cubic graph (G) as a basis, where its edges were replaced by some forbidden graph $H \in \mathcal{F}$, and was split at the "attachment points" of edge gadgets according to some vertex cover of G' and the splitting configuration C^0. For this graph, we are provided a witness certifying that the splitting configuration C^0 is not separating with respect to $\mathcal{F}$ in the form of an embedding π of $F \in \mathcal{F}$ into G^*, where the embedding of F is not constrained to a single edge gadget, but rather uses vertices of at least two neighboring edge gadgets (of edges $uv, vw \in E(G')$), $\chi_{G^*}(uv)$ and $\chi_{G^*}(vw)$, such that the embedding is not entirely contained in the shared intersection $\chi_{G^*}(v)$. Notice that $\pi(\cdot)^{-1}$ refers to vertices of F, whereas $\pi(\cdot)$ refers to vertices of G^*. See Figure 2 for an illustration.

We now show that $\pi^{-1}(\chi_{G^*}(v) \cap \text{Range}(\pi))$ is a vertex separator of F, that is, if these vertices are deleted from F, the resulting graph is disconnected. We show this basically by observing that F can be embedded into G^* in a particular way (as witnessed by π), and since G^* has certain structural features, these carry over to F, leading to a contradiction.

Suppose that $\pi^{-1}(\chi_{G^*}(v) \cap \text{Range}(\pi))$ is not a vertex separator of F. Then, all neighbors of $\pi^{-1}(\chi_{G^*}(v) \cap \text{Range}(\pi))$ in $V(F) \setminus \pi^{-1}(\chi_{G^*}(v) \cap \text{Range}(\pi))$ are pairwise connected via some path in F not using any of $\pi^{-1}(\chi_{G^*}(v) \cap \text{Range}(\pi))$ each. Select any one of these paths and call

it P. Without loss of generality, P starts with a vertex of $\pi^{-1}((\chi_{G^*}(uv) \setminus \chi_{G^*}(v)) \cap \mathrm{Range}(\pi))$ and ends in a vertex of $\pi^{-1}((\chi_{G^*}(vw) \setminus \chi_{G^*}(v)) \cap \mathrm{Range}(\pi))$. Due to the existence of π, we know that $P^* \coloneqq \pi(P)$ gives an isomorphic path in G^*. Since P does not use vertices of $\pi^{-1}(\chi_{G^*}(v) \cap \mathrm{Range}(\pi))$, P^* does not use vertices of $\chi_{G^*}(v)$.

By construction of G^*, all paths connecting the first and last vertex of P^* in G^* that are constrained to the union of the vertex sets of both edge gadgets, that is to $\chi_{G^*}(uv) \cup \chi_{G^*}(vw)$, must traverse the intersection of both edge gadgets, that is, $\chi_{G^*}(uv) \cap \chi_{G^*}(vw) = \chi_{G^*}(v)$. But P^* does not intersect with $\chi_{G^*}(v)$, hence it is not one of these paths. Therefore, P^* must traverse G^* using edge gadgets the "other way around", that is, not use the direct connection.

Observe that P^* induces a path corresponding to the edge gadgets it traverses in G, which in turn induces a path of length at least three in the underlying cubic graph G'. At least one of these edges in G', say xy, must be fully traversed by P^* in the corresponding part of G^*. Thus, there are $x', y' \in V(P^*)$ where $x' \in \chi_{G^*}(x) \cap V(P^*)$ and $y' \in \chi_{G^*}(y) \cap V(P^*)$. The distance between x' and y' in G^* is at least $L = 2 \cdot \max_{F \in \mathcal{F}} \mathrm{diam}(F)$, the number of times xy is subdivided in G. But then $\pi^{-1}(x')$ and $\pi^{-1}(y')$, vertices of F, have distance of at least L in F as well, a contradiction to the choice of L. Thus, $\pi^{-1}(\chi_{G^*}(v) \cap \mathrm{Range}(\pi))$ is a vertex separator of F. Furthermore, since $|\chi_{G^*}(v)| \leq 2$ and F is biconnected, the vertex separator contains exactly two vertices. We shall denote its two elements by a^1 and b^1.

We continue exploiting the structure of F to obtain a splitting configuration satisfying the conditions of this lemma. Let D be any connected component of $F \setminus \{a^1, b^1\}$. Suppose there is only one edge of the form dv with $d \in V(D)$ and $v \in \{a^1, b^1\}$ in $E(F)$. Then, F could not be biconnected, for the removal of a single vertex (either a^1 or b^1) would suffice to render F disconnected. Thus, there is a path P^1 from a^1 to b^1 in F with $P^1 \subseteq V(D) \cup \{a^1, b^1\}$. Since G^* was constructed with respect to the splitting configuration C^0, we notice that $|P^1| \geq \mathrm{wdt}(C^0)$.

Let X be the vertex set of some distinct connected component of $F \setminus \{a^1, b^1\}$, and let $Y \coloneqq V(F) \setminus (\{a^1, b^1\} \cup X)$. We notice that $a^1 b^1 \notin E(F)$, since $\pi(a^1)$ and $\pi(b^1)$ are descendants of the same split in the construction of G^*. Furthermore, X and Y form a partition of $V(F) \setminus \{a^1, b^1\}$. Thus, we may define a new disjoint splitting configuration C^1 as follows:

$$
\begin{aligned}
&C^1 \coloneqq (F, a^1, A_1^1, A_2^1, b^1, B_1^1, B_2^1), \text{ where}\\
&A_1^1 \coloneqq N_F(a^1) \cap X,\\
&A_2^1 \coloneqq N_F(a^1) \cap Y,\\
&B_1^1 \coloneqq N_F(b^1) \cap X, \text{ and}\\
&B_2^1 \coloneqq N_F(b^1) \cap Y.
\end{aligned}
$$

Remember that by definition, the width of C^1 is $\min\{d_{F_1}(a_1^1, a_2^1), d_{F_2}(b_1^1, b_2^1)\}$, where $F_1 \coloneqq \mathrm{Split}(F, a^1, A_1^1, A_2^1, a_1^1, a_2^1)$ and $F_2 \coloneqq \mathrm{Split}(F, b^1, B_1^1, B_2^1, b_1^1, b_2^1)$, such that a_1^1, a_2^1, b_1^1, and b_2^1 are fresh vertices. Consider F_1: By the argument above, we deduce that there is a shortest path through the descendant vertices of X from a_1^1 to b^1 in F_1. Furthermore, since $F \setminus \{a^1, b^1\}$ is comprised of at least two connected components, there also exists a shortest path through one of them (using descendant vertices of Y) from b^1 to a_2^1. Each of the considered shortest paths must have length at least $\mathrm{wdt}(C^0)$, as G^* was constructed with respect to the splitting configuration C^0. Also, note that all paths connecting a_1^1 and a_2^1 in F must traverse b^1. Thus, combining these paths yields that $d_{F_1}(a_1^1, a_2^1) \geq 2\,\mathrm{wdt}(C^0)$. See Figure 3 for an illustration.

We proceed symmetrically for F_2. Hence, we obtain that $\mathrm{wdt}(C^1) > \mathrm{wdt}(C^0)$, and thus C^1 is a splitting configuration satisfying the required conditions. ◀

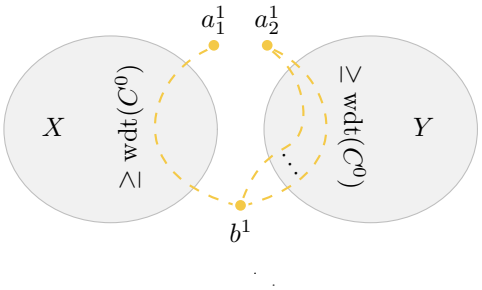

Figure 3 The derived splitting configuration C^1 has width at least $2\,\mathrm{wdt}(C^0)$.

It remains to ensure that the separating splitting configuration is additionally intra-edge embedding-free and therefore admissible. This property is implied when restricting $\mathcal{F}$ such that when any $F \in \mathcal{F}$ is destroyed by one or two non-trivial disjoint splits, the resulting graph is free of forbidden (induced) subgraphs. For example, each finite set of cycles satisfies this condition, or each set $\{F\}$ where F is biconnected (★). Finally, we can apply Lemma 3.5 to obtain NP-hardness of the corresponding vertex-splitting problems (★). In total, this then concludes the proof of the first part of Theorem 1.2, i.e., $\mathrm{Free}_{\prec/\subseteq}(\{F\})$-VS is NP-complete when F is biconnected. For the other parts, as we progress onward from biconnected graphs to higher degrees of connectedness, we can use similar techniques to show NP-hardness, but the restrictions imposed on the forbidden subgraphs relax. For the 4-connected case, no further restrictions are required.

4 Conclusion

In summary, for large families of graph classes Π, it is the case that Π-VS is NP-hard and, so far, nontrivial polynomial-time solvable cases are sporadic, such as Forest-VS and $\mathrm{Free}_{\prec}(\overline{K_3}, P_3)$-VS. Hence, the line of separation between tractability and intractability is much more jagged than in the case of Π Vertex Deletion, where a classical result by Lewis and Yannakakis shows the problem is NP-hard for hereditary Π if and only if Π is nontrivial, that is, Π and $\overline{\Pi}$ are infinite [27]. In contrast, the "complexity boundary" of Π-VS seems much more reminiscent of the classical Π Edge Deletion problem, for which no such characterization is known, despite extensive study since the late seventies.

Since for well-connected forbidden subgraphs our results imply hardness, a natural direction to further trace the line of separation between tractability and intractability would be to study more fragile forbidden subgraphs, that is, for instance, determining the complexity of $\mathrm{Free}_{\prec}(\{P_4\})$-VS and $\mathrm{Free}_{\prec}(\{K_{1,3}\})$-VS and seeing if patterns emerge in this regime. For the former, we can show a relation to a cograph-covering problem which we tend to believe is NP-hard. The latter we consider fully open. On the other hand, our results pave the way for studying broader notions of tractability instead of polynomial-time solvability such as approximating the optimal number of splits needed and further studying the parameterized complexity with respect to the number of splits.

In terms of approximation, our reduction for $\mathrm{Free}_{\prec}(\{K_3\})$-VS implies that minimizing the number of splits cannot be polynomial-time approximated to within an arbitrary fixed approximation factor, that is, there is no PTAS. However, constant-factor approximations may still exist and it would be interesting to see whether Π-VS is constant-factor approximable in polynomial-time if Π is characterized by a finite number of forbidden induced subgraphs or some large subfamily of such Π.

The parameterized complexity of Π-VS with respect to the number of splits also offers interesting contrasts and invites further investigation: $\text{Free}_{\prec}(\{P_3\})$-VS is fixed-parameter tractable [14] but $\text{Free}_{\prec}(\{K_3\})$-VS is para-NP-hard (Theorem 1.4). This raises the question to classify for which hereditary classes Π problem Π-VS is fixed-parameter tractable (analogous to the vertex-deletion version [26]). On the intractability side, it seems worthwhile to explore generalizations of the hardness construction for $\text{Free}_{\prec}(\{K_3\})$-VS and constant number of splits per vertex (Theorem 1.4): The crucial property that we have exploited in the construction is that all constraints imposed by K_3s can only be solved by mapping edges between copies of the single vertex that we can feasibly split. If we use larger graphs instead of K_3, it is not obvious how to maintain this property. To obtain NP-hardness for a constant number of splits per vertex is it possible to replace K_3 by K_4, by K_ℓ for any fixed $\ell \geq 3$, or even a fixed graph of a more general graph class?

For tractability, it is tempting to exploit the connection to Hitting Set to try and obtain fixed-parameter tractability for Π characterized by a finite number of forbidden induced subgraphs. However, one has to work around two problems: First, the vertices to split are not necessarily a minimal hitting set (consider $\Pi = K_3$-free and the wheel graph of six vertices: the center vertex hits all forbidden triangles, yet at least two splits are needed to solve the instance). One thus has to efficiently find the additional split vertices that are not contained in an underlying minimal hitting set. Second, even after determining the vertices to split, one has to tackle interesting, often coloring-related problems such as in the case of $\text{Free}_{\prec}(\{K_3\})$-VS.

Finally, it would be interesting to carry out a complexity classification program for Π-VS when Π is characterized by forbidden minors instead. An interesting starting point might be the contrast between the polynomial-time solvability of Forest-VS, that is, vertex splitting to K_3-minor free graphs, and NP-hardness of Planar-VS, that is, K_5 and $K_{3,3}$-minor free graphs.

References

1 Faisal N. Abu-Khzam, Judith Egan, Serge Gaspers, Alexis Shaw, and Peter Shaw. Cluster editing with vertex splitting. In Jon Lee, Giovanni Rinaldi, and Ali Ridha Mahjoub, editors, *Proceedings of the 5th International Symposium of Combinatorial Optimization (ISCO 2018)*, volume 10856 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2018. doi:10.1007/978-3-319-96151-4_1.

2 Abu Reyan Ahmed, Stephen G. Kobourov, and Myroslav Kryven. An FPT algorithm for bipartite vertex splitting. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Proceedings of the 30th International Symposium Graph Drawing and Network Visualization (GD 2022)*, volume 13764 of *Lecture Notes in Computer Science*, pages 261–268. Springer, 2022. doi:10.1007/978-3-031-22203-0_19.

3 Emmanuel Arrighi, Matthias Bentert, Pål Grønås Drange, Blair D. Sullivan, and Petra Wolf. Cluster editing with overlapping communities. In Neeldhara Misra and Magnus Wahlström, editors, *Proceedings of the 18th International Symposium on Parameterized and Exact Computation (IPEC 2023)*, volume 285 of *LIPIcs*, pages 2:1–2:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.IPEC.2023.2.

4 Jakob Baumann, Matthias Pfretzschner, and Ignaz Rutter. Parameterized complexity of vertex splitting to pathwidth at most 1. In *Proceedings of the 49th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2023)*, volume 14093 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 2023. doi:10.1007/978-3-031-43380-1_3.

5 Matthias Bentert, Alex Crane, Pål Grønås Drange, Felix Reidl, and Blair D. Sullivan. Correlation clustering with vertex splitting. In Hans L. Bodlaender, editor, *Proceedings of the 19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024)*, volume 294 of *LIPIcs*, pages 8:1–8:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.SWAT.2024.8.

6 Édouard Bonnet, Eun Jung Kim, Stéphan Thomassé, and Rémi Watrigant. Twin-width I: tractable FO model checking. *Journal of the ACM*, 69(1):3:1–3:46, 2022. doi:10.1145/3486655.

7 Vera Chekan and Stefan Kratsch. Tight algorithmic applications of clique-width generalizations. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *Proceedings of the 48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, volume 272 of *LIPIcs*, pages 35:1–35:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.MFCS.2023.35.

8 Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification. *Computer Science Review*, 48:100556, 2023. doi:10.1016/j.cosrev.2023.100556.

9 A. Davoodi, R. Javadi, and B. Omoomi. Edge clique covering sum of graphs. *Acta Mathematica Hungarica*, 149(1):82–91, 2016. doi:10.1007/s10474-016-0586-1.

10 Peter Eades and Candido Ferreira Xavier de Mendonça Neto. Vertex splitting and tension-free layout. In *Proceedings of the International Symposium on Graph Drawing (GD 1995)*, volume 1027 of *Lecture Notes in Computer Science*, pages 202–211. Springer, 1995. doi:10.1007/BFb0021804.

11 David Eppstein, Philipp Kindermann, Stephen Kobourov, Giuseppe Liotta, Anna Lubiw, Aude Maignan, Debajyoti Mondal, Hamideh Vosoughpour, Sue Whitesides, and Stephen Wismath. On the planar split thickness of graphs. *Algorithmica*, 80:977–994, 2018. doi:10.1007/s00453-017-0328-y.

12 Luérbio Faria, Celina M. H. de Figueiredo, and Candido Ferreira Xavier de Mendonça Neto. SPLITTING NUMBER is NP-complete. *Discrete Applied Mathematics*, 108(1-2):65–83, 2001. doi:10.1016/S0166-218X(00)00220-1.

13 Alexander Firbas. Establishing hereditary graph properties via vertex splitting. Master's thesis, TU Wien, 2023. doi:10.34726/hss.2023.103864.

14 Alexander Firbas, Alexander Dobler, Fabian Holzer, Jakob Schafellner, Manuel Sorge, Anaïs Villedieu, and Monika Wißmann. The complexity of cluster vertex splitting and company. In Henning Fernau, Serge Gaspers, and Ralf Klasing, editors, *Proceedings of the 49th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2024)*, volume 14519 of *Lecture Notes in Computer Science*, pages 226–239. Springer, 2024. doi:10.1007/978-3-031-52113-3_16.

15 Alexander Firbas and Manuel Sorge. On the complexity of establishing hereditary graph properties via vertex splitting, 2024. arXiv:2401.16296, doi:10.48550/arXiv.2401.16296.

16 Herbert Fleischner. *Eulerian graphs and related topics*. North-Holland, 1990.

17 Martin Fürer. A natural generalization of bounded tree-width and bounded clique-width. In Alberto Pardo and Alfredo Viola, editors, *Proceedings of the 11th Latin American Symposium on Theoretical Informatics (LATIN 2014)*, volume 8392 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2014. doi:10.1007/978-3-642-54423-1_7.

18 Leslie Ann Goldberg and Marc Roth. Parameterised and fine-grained subgraph counting, modulo 2. *Algorithmica*, 86(4):944–1005, 2024. doi:10.1007/S00453-023-01178-0.

19 Petr A. Golovach, Pim van 't Hof, and Daniël Paulusma. Obtaining planarity by contracting few edges. *Theoretical Computer Science*, 476:38–46, 2013. doi:10.1016/j.tcs.2012.12.041.

20 Jens Gramm, Jiong Guo, Falk Hüffner, Rolf Niedermeier, Hans-Peter Piepho, and Ramona Schmid. Algorithms for compact letter displays: Comparison and evaluation. *Computational Statistics & Data Analysis*, 52(2):725–736, 2007. doi:10.1016/j.csda.2006.09.035.

21 Sylvain Guillemot and Dániel Marx. A faster FPT algorithm for bipartite contraction. *Information Processing Letters*, 113(22-24):906–912, 2013. doi:10.1016/j.ipl.2013.09.004.

22 Chengwei Guo and Leizhen Cai. Obtaining split graphs by edge contraction. *Theoretical Computer Science*, 607:60–67, 2015. doi:10.1016/j.tcs.2015.01.056.

23 Pinar Heggernes, Pim van 't Hof, Daniel Lokshtanov, and Christophe Paul. Obtaining a bipartite graph by contracting few edges. *SIAM Journal on Discrete Mathematics*, 27(4):2143–2156, 2013. doi:10.1137/130907392.

24 Nathalie y Henr, Anastasia Bezerianos, and Jean-Daniel Fekete. Improving the readability of clustered social networks using node duplication. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1317–1324, 2008. doi:10.1109/TVCG.2008.141.

25 Anthony J. W. Hilton and C. Zhao. Vertex-splitting and chromatic index critical graphs. *Discrete Applied Mathematics*, 76(1-3):205–211, 1997. doi:10.1016/S0166-218X(96)00125-4.

26 Subhash Khot and Venkatesh Raman. Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science*, 289(2):997–1008, 2002. doi:10.1016/S0304-3975(01)00414-5.

27 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

28 Matthias Mayer and Fikret Erçal. Genetic algorithms for vertex splitting in DAGs. In *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA 1993)*, page 646. Morgan Kaufmann, 1993. URL: https://scholarsmine.mst.edu/comsci_techreports/25/.

29 George B. Mertzios and Derek G. Corneil. Vertex splitting and the recognition of trapezoid graphs. *Discrete Applied Mathematics*, 159(11):1131–1147, 2011. doi:10.1016/j.dam.2011.03.023.

30 Martin Nöllenburg, Manuel Sorge, Soeren Terziadis, Anaïs Villedieu, Hsiang-Yun Wu, and Jules Wulms. Planarizing graphs and their drawings by vertex splitting. In *Proceedings of the 30th International Symposium on Graph Drawing and Network Visualization (GD 2022)*, pages 232–246, Cham, 2023. Springer International Publishing. doi:10.1007/978-3-031-22203-0_17.

31 Doowon Paik, Sudhakar M. Reddy, and Sartaj Sahni. Vertex splitting in dags and applications to partial scan designs and lossy circuits. *International Journal of Foundations of Computer Science*, 9(4):377–398, 1998. doi:10.1142/S0129054198000301.

32 Norbert Peyerimhoff, Marc Roth, Johannes Schmitt, Jakob Stix, Alina Vdovina, and Philip Wellnitz. Parameterized counting and Cayley graph expanders. *SIAM Journal on Discrete Mathematics*, 37(2):405–486, 2023. doi:10.1137/22M1479804.

33 W. T. Tutte. *Connectivity in graphs*. University of Toronto Press, 1966.

34 Ryuhei Uehara. NP-complete problems on a 3-connected cubic planar graph and their applications. *Tokyo Woman's Christian University, Tokyo, Japan, Tech. Rep. TWCU-M-0004*, 1996.