Incompressibility of H-Free Edge Modification Problems: Towards a Dichotomy

Dániel Marx
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
marx@cispa.saarland

R. B. Sandeep
Department of Computer Science and Engineering, Indian Institute of Technology Dharwad, India
sandeeprb@iitdh.ac.in

Abstract

Given a graph G and an integer k, the H-free Edge Editing problem is to find whether there exist at most k pairs of vertices in G such that changing the adjacency of the pairs in G results in a graph without any induced copy of H. The existence of polynomial kernels for H-free Edge Editing (that is, whether it is possible to reduce the size of the instance to $k^{O(1)}$ in polynomial time) received significant attention in the parameterized complexity literature. Nontrivial polynomial kernels are known to exist for some graphs H with at most 4 vertices (e.g., path on 3 or 4 vertices, diamond, paw), but starting from 5 vertices, polynomial kernels are known only if H is either complete or empty. This suggests the conjecture that there is no other H with at least 5 vertices were H-free Edge Editing admits a polynomial kernel. Towards this goal, we obtain a set $\mathcal{H}$ of nine 5-vertex graphs such that if for every $H \in \mathcal{H}$, H-free Edge Editing is incompressible and the complexity assumption $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$ holds, then H-free Edge Editing is incompressible for every graph H with at least five vertices that is neither complete nor empty. That is, proving incompressibility for these nine graphs would give a complete classification of the kernelization complexity of H-free Edge Editing for every H with at least 5 vertices.

We obtain similar result also for H-free Edge Deletion. Here the picture is more complicated due to the existence of another infinite family of graphs H where the problem is trivial (graphs with exactly one edge). We obtain a larger set $\mathcal{H}$ of nineteen graphs whose incompressibility would give a complete classification of the kernelization complexity of H-free Edge Deletion for every graph H with at least 5 vertices. Analogous results follow also for the H-free Edge Completion problem by simple complementation.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases incompressibility, edge modification problems, H-free graphs

Digital Object Identifier 10.4230/LIPIcs.ESA.2020.72

Related Version A full version of the paper is available at `https://arxiv.org/abs/2004.11761`.

Funding Research supported by the European Research Council (ERC) grant SYSTEMATIC-GRAPH: "Systematic mapping of the complexity landscape of hard algorithmic graph problems", reference 725978.
R. B. Sandeep: Partially supported by SERB Grant SRG/2019/002276: "Complexity Dichotomies for Graph Modification Problems".

1 Introduction

In a typical graph modification problem, the input is a graph G and an integer k, and the task is to make at most k allowed editing operations on G to make it belong to a certain graph class or satisfy a certain property. For example, Vertex Cover (remove k vertices to make the graph edgeless), Feedback Vertex Set (remove k vertices to make the graph acyclic), Odd Cycle Transversal (remove k edges/vertices to make the graph bipartite),

and Minimum Fill-in (add k edges to make the graph chordal) are particularly well-studied members of this problem family. Most natural graph modification problems are known to be NP-hard, in fact, there are general hardness results proving hardness for many problems [19, 21, 22]. On the other hand, most of these problems are fixed-parameter tractable (FPT) parameterized by k: it can be solved in time $f(k)n^{O(1)}$, where f is a computable function depending only on k [3, 8, 17, 18]. Looking at the parameterized complexity literature, one can observe that, even though there are certain recurring approaches and techniques, these FPT results are highly problem specific, and often rely on a very detailed understanding of the graph classes at hand.

A class of problems that can be treated somewhat more uniformly is H-free Edge Editing. This is a separate problem for every fixed graph H: given a graph G and an integer k, the task is to find whether there exist at most k pairs of vertices in G such that changing the adjacency of the pairs in G results in a graph without any induced copy of H. Aravind et al. [2] proved that H-free Edge Editing is NP-hard for every graph H with at least 3 vertices. However, a simple application of the technique of bounded-depth search trees shows that H-free Edge Editing is FPT parameterized by k for every fixed H [3].

Graph modification problems were explored also from the viewpoint of polynomial kernelization: is there a polynomial-time preprocessing algorithm that does not necessarily solve the problem, but at least reduces the size of the an instance to be bounded by a polynomial of k? The existence of a polynomial kernelization immediately implies that the problem is FPT (after the preprocessing, one can solve the reduced instance by brute force or any exact method). Therefore, one can view polynomial kernelization as a special type of FPT result that tries to formalize the question whether the problem can be efficiently preprocessed in a way that helps exhaustive search methods. There is a wide literature on algorithms for kernelization (see, e.g., [14]). Conversely, incompressibility results can show, typically under the complexity assumption $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, that a parameterized problem has no polynomial kernelization.

Most of the highly nontrivial FPT algorithms for graph modification problems do not give kernelization results and, in many cases, it required significant amount of additional work to obtain kernelization algorithms. In particular, the FPT algorithm for H-free Edge Editing based on the technique of bounded-depth search trees does not give polynomial kernels. For the specific case when $H = K_r$ is a complete graph, it is easy to see that there is a solution using only deletions. Now the problem essentially becomes a Hitting Set problem with sets of bounded size: we have to select at least one edge from the edge set of each copy of K_r. Therefore, known kernelization results for Hitting Set can be used to show that K_r-free Edge Editing has a polynomial kernel for every fixed r. A similar argument works if H is an empty graph on r vertices.

Besides cliques and empty graphs, it is known for certain graphs H of at most 4 vertices (diamond [5, 9], path [6, 15, 16], paw [7, 13], and their complements) that H-free Edge Editing has a polynomial kernel, but these algorithms use very specific arguments exploiting the structure of H-free graphs. As there is a very deep known structure theory of claw-free (i.e, $K_{1,3}$-free) graphs, it might be possible to obtain a polynomial kernel for Claw-free Edge Editing, but this is currently a major open question [4, 10, 12]. However, besides cliques and empty graphs, no H with at least 5 vertices is known where H-free Edge Editing has a polynomial kernel and there is no obvious candidate H for which one would expect a kernel. This suggests the following conjecture:

▶ **Conjecture 1.** *If H is a graph with at least 5 vertices, then H-free Edge Editing has a polynomial kernel if and only if H is a complete or empty graph.*

We are not able to resolve this conjecture, but make substantial progress towards it by showing that only a finite number of key cases needs to be understood. Our main result for H-free Edge Editing is the following.

▶ **Theorem 1.** *There exists a set $\mathcal{H}^E$ of nine graphs, each with five vertices such that if H-free Edge Editing is incompressible for every $H \in \mathcal{H}^E$, then for a graph H with at least five vertices H-free Edge Editing is incompressible if and only if H is neither complete nor empty, where the incompressibility assumes* NP $\not\subseteq$ coNP/poly.

The set $\mathcal{H}^E$ of nine graphs are shown in Figure 1a. Note that a simple reduction by complementation shows that H-free Edge Editing and $\overline{H}$-free Edge Editing have the same complexity. Therefore, for each of these nine graphs, we could put either it or its complement into the set $\mathcal{H}^E$. As it will be apparent later, we made significant efforts to reduce the size of $\mathcal{H}^E$ as much as possible. However, the known techniques for proving incompressibility do not seem to work for these graphs. Let us observe that most of these graphs are very close to the known cases that admit a polynomial kernel: for example, they can be seen as a path, paw, or diamond with an extra isolated vertex or with an extra degree-1 vertex attached. Thus resolving the kernelization complexity of H-free Edge Editing for any of these remaining graphs seems to be a particularly good research question: either one needs to extend in a nontrivial way the known kernelization results, or a significant new ideas are needed for proving hardness.

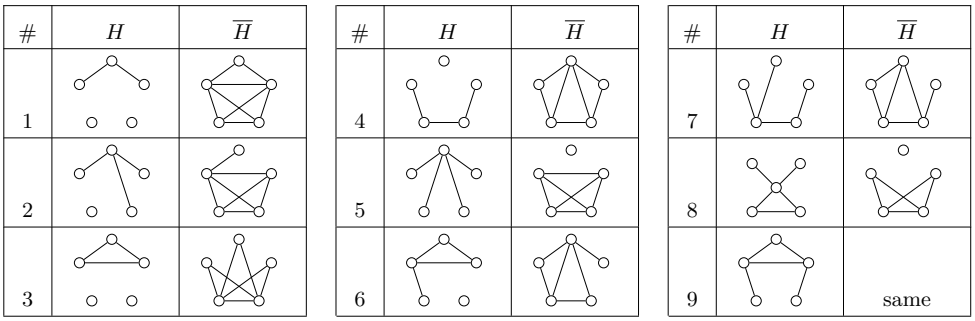
The reader might not be convinced of the validity of Conjecture 1 and may wonder about the value of Theorem 1 when the conjecture is false. However, we can argue that Theorem 1 is meaningful even in this case. It shows that if there is any H violating Conjecture 1, then one of the 9 graphs in $\mathcal{H}^E$ also violates it. That is, if we believe that there are kernelization results violating the conjecture, then we should focus on the 9 graphs in $\mathcal{H}^E$, as these are the easiest cases where we may have a kernelization result. In other words, Theorem 1 precisely shows the frontier where new algorithmic results are most likely to exist.

H-free Edge Deletion is the variant of H-free Edge Editing where only edge removal is allowed. For the same fixed graph H, it seems that H-free Edge Deletion should be a simpler problem than H-free Edge Editing, but we want to emphasize that H-free Edge Deletion is *not* a special case of H-free Edge Editing. There is no known general reduction from the former to the latter, although the technique of completion enforcers (see Section 5 and [4]) can be used for many specific graphs H. There is a known case where H-free Edge Deletion seems to be strictly easier: if H has at most one edge, then there is only one way of destroying a copy of an induced H by edge removal, making the problem polynomial-time solvable. Aravind et al. [1] showed that having at most one edge is the only condition that makes H-free Edge Deletion polynomial-time solvable: if H has at least two edges, then the problem is NP-hard. Therefore, the counterpart of Conjecture 1 for H-free Edge Deletion should take this case also into account.

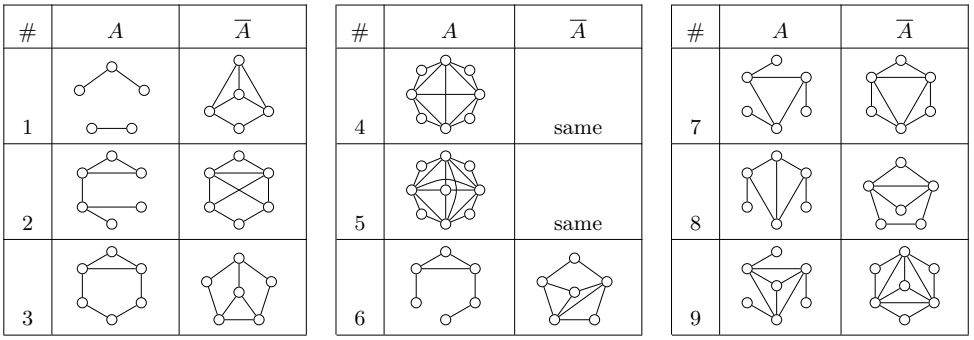
▶ **Conjecture 2.** *If H is a graph with at least 5 vertices, then H-free Edge Deletion has a polynomial kernel if and only if H is a complete graph or has at most one edge.*

Working toward this conjecture, we show that only a finite number of cases needs to be shown incompressible.

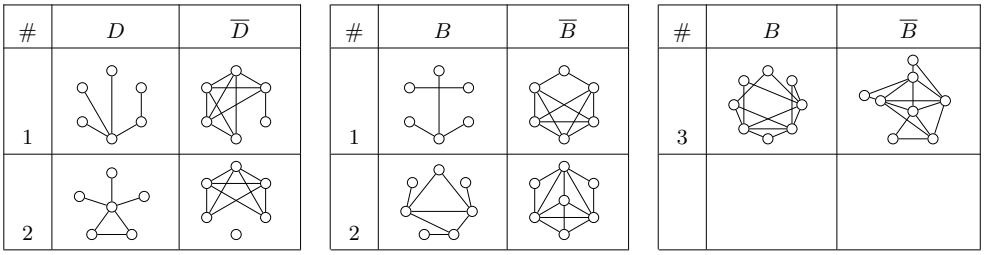
▶ **Theorem 2.** *There exists a set $\mathcal{H}^D$ of nineteen graphs, each with either five or six vertices such that if H-free Edge Deletion is incompressible for every $H \in \mathcal{H}^D$ then for a graph H with at least five vertices, H-free Edge Deletion is incompressible if and only if H is a graph with at least two edges but not complete, where the incompressibility assumes* NP $\not\subseteq$ coNP/poly.



(a) The set $\mathcal{H}$ of graphs.

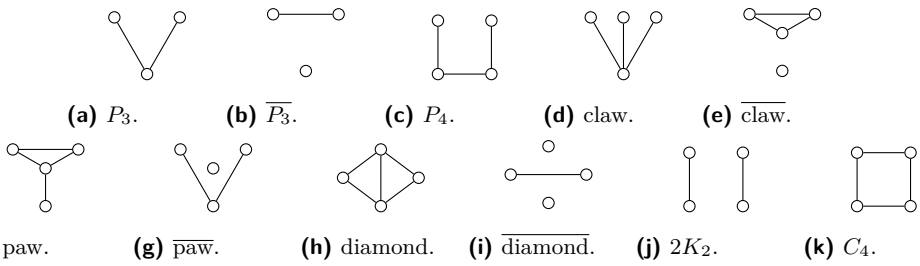


(b) The set $\mathcal{A}$ of graphs.



(c) The sets $\mathcal{D}$ and $\mathcal{B}$ of graphs.

Figure 1



(a) P_3. **(b)** $\overline{P_3}$. **(c)** P_4. **(d)** claw. **(e)** $\overline{\text{claw}}$.

(f) paw. **(g)** $\overline{\text{paw}}$. **(h)** diamond. **(i)** $\overline{\text{diamond}}$. **(j)** $2K_2$. **(k)** C_4.

Figure 2 All non-empty and non-complete graphs with at most four vertices.

The set $\mathcal{H}^D$ contains the graphs in set $\mathcal{H}^E$, as well their complements. This seems reasonable and hard to avoid: if we do not have an incompressibility result for H-free Edge Editing for some $H \in \mathcal{H}^E$, then it is unlikely that we can find such a result for H-free Edge Deletion (even though, as discussed above, there is no formal justification for this). Together with these 17 graphs (note that H_9 is the same as its complement), we need to include into

$\mathcal{H}^D$ the two graphs D_1 and D_2 shown in Figure 1c. In the case of editing, we can prove incompressibility for these two graphs by a reduction from H-free Edge Editing where H is the graph with 5 vertices and one edge. However, H-free Edge Deletion for this H is polynomial-time solvable.

Finally, let us consider the H-free Edge Completion problem, where we have to make G induced H-free by adding at most k edges. As H-free Edge Completion is essentially the same problem as $\overline{H}$-free Edge Deletion, we can obtain a counterpart of Theorem 2 by simple complementation:

▶ **Theorem 3.** *There exists a set $\mathcal{H}^C$ of nineteen graphs, each with either five or six vertices such that if H-free Edge Completion is incompressible for every $H \in \mathcal{H}^C$ then for a graph H with at least five vertices, H-free Edge Completion is incompressible if and only if H is a graph with at least two nonedges but not empty, where the incompressibility assumes* $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$.

Our techniques. We crucially use two earlier results. First, Cai and Cai [4] proved that H-free Edge Editing is incompressible (assuming $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$) when H or $\overline{H}$ is a cycle or a path of length at least 4, or 3-connected but not complete. While these result handle many graphs and prove to be very useful for our proofs, they do not come close to a complete classification. Second, we use a key tool in the polynomial-time dichotomy result of Aravind et al. [1]: if V_ℓ is the set of lowest degree vertices of H, then $(H - V_\ell)$-free Edge Editing can be reduced to H-free Edge Editing. The same statement holds for the set V_h of highest degree vertices.

Our proofs of Theorems 1–3 introduce new incompressibility results and new reductions, which we put together to obtain an almost complete classification by a graph-theoretic analysis. Additionally, to make the arguments simpler, we handle small graphs by an exhaustive computer search. In the following, we highlight some of the main ideas that appear in the paper.

- **Analysis of graphs.** Our goal is to prove Theorem 1 by induction on the size of H. First we handle the case when H is regular: we show that this typically implies that either H or $\overline{H}$ is 3-connected, and the result of Cai and Cai [4] can be used. If H is not regular, then the graphs $H - V_\ell$ and $H - V_h$ are nonempty and have stricly fewer vertices than H. If one of them, say $H - V_\ell$, has at least 5 vertices and is neither complete nor empty, then the induction hypothesis gives an incompressibility result for $(H - V_\ell)$-free Edge Editing, which gives an incompressibility result for H-free Edge Editing by the reduction of Aravind et al. [1]. Therefore, we only need to handle those graphs H where it is true for both $H - V_\ell$ and $H - V_h$ that they are either small, complete, or empty. But we can obtain a good structural understanding of H in each of these cases, which allows us to show that either H or $\overline{H}$ is 3-connected, or H has some very well defined structure. With these arguments, we can reduce the problem to the incompressibility of H-free Edge Editing for a few dozen specific graphs H and for a few well-structured infinite families (such as $K_{2,t}$).
  For H-free Edge Deletion, we have the additional complication that one or both of $H - V_\ell$ and $H - V_h$ can be near-empty (i.e., has exactly one edge), which is not an incompressible case for this problem. We need additional case analysis to cover such graphs, but the spirit of the proof remains the same.
- **Computer search.** Our analysis of graphs becomes considerably simpler if we assume that H is not too small. In this case, we can assume that at least one of $H - V_\ell$ and $H - V_h$ is a complete or empty graph of certain minimum size, which is a very helpful

starting point for proving the 3-connectivity of H or $\bar{H}$, respectively. Therefore, we handle every graph with at most 9 vertices using an exhaustive computer search and assume in the proof that H has at least 10 vertices. The list provided by McKay [20] shows that there are 288266 different graphs with at most 9 vertices, which is feasible for a computer search. In principle, it would be possible to extend our case analysis to avoid this computer search, but it would significantly complicate the proof and is not clear what additional insight it would give.

- **Reductions.** We investigate different reductions that allow us to reduce H'-free Edge Editing to H-free Edge Editing when H' is an induced subgraph of H satisfying certain conditions. With extensive use of such reductions, we can reduce the remaining cases of H-free Edge Editing that needs to be handled to a smaller finite set.
- **Incompressibility results.** We carefully revisit the proof of Cai and Cai [4] showing the incompressibility of H-free Edge Editing when H is 3-connected, and observe that, with additional ideas, it can be made to work also for certain 2-connected graphs that are not 3-connected (the set $\mathcal{A}$ of graphs shown in Figure 1b and the set $\mathcal{B}$ of graphs shown in Figure 1c). This allows us to handle every graph, except those finite sets that are mentioned in Theorems 1–3. A key step in many of these incompressibility results is to establish first incompressibility for the Restricted H-free Edge Deletion problem, which is the generalization of H-free Edge Deletion where some of the edges of G are marked as forbidden in the input, and the solution is not allowed to delete forbidden edges. Then we use deletion and completion enforcer gadgets specific to H to reduce Restricted H-free Edge Deletion to H-free Edge Editing.

The paper is organized as follows. Preliminaries are in Section 2. Section 3 presents the churning procedure, our main technical tool in the analysis of graphs, and shows that it reduces the problem to a finite number of graphs, plus a few well-defined infinite families. Section 4 presents reductions (old and new) that allow us to further reduce the number of graphs we need to handle. Finally, in Setion 5, we give new incompressibility results, showing that only the cases stated in Theorems 1–3 need to be proved incompressible to complete the exploration of the complexity landscape of the problems. All proofs have been moved to a full version of the paper due to space constraints.

2 Preliminaries

Graph-theoretic notation and terminology. For a graph G, $V(G)$ and $E(G)$ denote the set of vertices and the set of edges of G respectively. For a set $V' \subseteq V(G)$, $G - V'$ denotes the graph obtained by removing all vertices in V' and their incident edges from G. For a set F of pairs of vertices and a graph G, $G \triangle F$ denotes the graph G' such that $V(G') = V(G)$ and $E(G') = \{(u,v) \mid ((u,v) \in E(G) \text{ and } (u,v) \notin F) \text{ or } (u,v \in V(G),\ (u,v) \notin E(G), \text{ and } (u,v) \in F)\}$. Whenever we say that a set of (non)edges F is a solution of an instance (G,k) of a problem, we refers to a subset of F containing all (non)edges where both the end vertices are in $V(G)$. A graph is *empty* if it does not have any edges. A graph is *near-empty* if it has exactly one edge. A graph is *complete* if it has no nonedges. A component of a graph is a *largest component* if it has maximum number of vertices among all components of the graph. Similarly, a component of a graph is a *smallest component* if it has minimum number of vertices among all components of the graph. For a graph H which is not complete, the vertex connectivity of H is the minimum integer c such that there exists a set $S \subseteq V(H)$ such that $|S| = c$ and $H - S$ is disconnected. For a graph H with vertex connectivity 1, a vertex v in H is known as a cut vertex if $H - v$ is disconnected. A graph is k-connected, if

the vertex connectivity of it is at least k. An induced subgraph H' of H is known as a 2-connected component if H' is a maximal 2-connected induced subgraph of H. The adjectives "largest" and "smallest" can be applied to 2-connected components as done for components. A *twin-star* graph T_{ℓ_1,ℓ_2} for $\ell_1, \ell_2 \geq 0$ is defined as the tree with two adjacent vertices u and v such that $|N(u) \setminus \{v\}| = \ell_1, |N(v) \setminus \{u\}| = \ell_2$, and every vertex in $N(u) \cup N(v) \setminus \{u, v\}$ has degree 1. A graph G is H-free if G does not contain any induced copy of H. For two graphs G_1 and G_2, the disjoint union of G_1 and G_2 denoted by $G_1 \cup G_2$ (or $G_2 \cup G_1$) is the graph G such that $V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$. For two graphs G_1 and G_2, the join of G_1 and G_2 denoted by $G_1 \boxtimes G_2$ (or $G_2 \boxtimes G_1$), is the graph G such that $V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2) \cup \{(x, y) \mid x \in V(G_1), y \in V(G_2)\}$. A complete graph, a cycle, and a path with t vertices are denoted by K_t, C_t, and P_t respectively. By $K_t - e$, we denote the graph obtained by deleting an edge from a complete graph on t vertices. We call a graph *non-regular* if it is not regular. A *modular decomposition* $\mathcal{M}$ of a graph G is a partitioning of its vertices into maximal sets, known as *modules*, such that for every set $M \in \mathcal{M}$, every vertex in M has the same neighborhood outside M. Let $\mathcal{M}' \subseteq \mathcal{M}$. Let $V' = \bigcup_{M \in \mathcal{M}'} M$. Then we say that $\mathcal{M}'$ corresponds to V'. For a set $\mathcal{S}$ of graphs, by $\overline{\mathcal{S}}$ we denote the set of complements of graphs in $\mathcal{S}$. Figure 2 shows all graphs with at most four vertices which are neither empty nor complete.

For $t \geq 3$, let J_t be the graph obtained from $K_2 \boxtimes tK_1$ and C_4 by identifying an edge of C_4 with the edge between the highest degree vertices in $K_2 \boxtimes tK_1$. Let Q_t be the graph graph obtained from $K_{2,t}$, for some $t \geq 3$, by adding a path of length three between the highest degree vertices in $K_{2,t}$. Let $\mathcal{H}$, $\mathcal{A}$, $\mathcal{D}$, $\mathcal{B}$, $\mathcal{S}$ denote the graphs (H, A, D, B, S respectively) shown in Figures 1a, 1b, 1c, and 3a. Let $\mathcal{F}$ be the union of graphs in the classes of graphs shown in column $\mathcal{F}$ of Figure 3b. The graphs in $\mathcal{S}$ and $\mathcal{F}$ are handled in Section 4 and the graphs in $\mathcal{A}$ and $\mathcal{B}$ are handled in Section 5. For all these classes of graphs, we use subscripts to identify each graph/graph class. For example H_1 is $P_3 \cup 2K_1$ and $\mathcal{F}_1$ is the class of graphs $K_{1,t}$. Let $\mathcal{W}$ be the set $\mathcal{H} \cup \overline{\mathcal{H}} \cup \mathcal{A} \cup \overline{\mathcal{A}} \cup \mathcal{D} \cup \overline{\mathcal{D}} \cup \mathcal{B} \cup \overline{\mathcal{B}} \cup \mathcal{S} \cup \overline{\mathcal{S}} \cup \mathcal{F} \cup \overline{\mathcal{F}}$. We observe that $\overline{\mathcal{W}} = \mathcal{W}$.

Parameterized problems and transformations. A parameterized problem is a classical problem with an additional integer input known as the parameter. A parameterized problem admits a polynomial kernel if there is a polynomial-time algorithm which takes as input an instance (I, k) of the problem and outputs an instance (I', k') of the same problem, where $|I'|, k' \leq p(k)$, where $p(k)$ is a polynomial in k, such that (I, k) is a yes-instance if and only if (I', k') is a yes-instance. A parameterized problem is incompressible if it does not admit a polynomial kernel. A *Polynomial Parameter Transformation* (PPT) from one parameterized problem Q to another parameterized problem Q' is a polynomial-time algorithm which takes as input an instance (I, k) of Q and produces an instance (I', k') of Q' such that (I, k) is a yes-instance of Q if and only if (I', k') is a yes-instance of Q', and $k' \leq p(k)$, for some polynomial $p(.)$. It is known that if there is a PPT from Q to Q', then if Q is incompressible, then so is Q'. We refer to the book [11] for various concepts in parameterized algorithms and complexity.

The parameterized problems we deal with in this paper are listed below.

H-free Edge Editing: Given a graph G and an integer k, do there exist at most k edges such that editing (adding or deleting) them in G results in an H-free graph?
Parameter: k

H-free Edge Deletion: Given a graph G and an integer k, do there exist at most k edges such that, deleting them from G results in an H-free graph? **Parameter**: k

H-free Edge Completion: Given a graph G and an integer k, do there exist at most k edges such that, adding them in G results in an H-free graph? **Parameter**: k

Basic results. Proposition 4 follows from the observations that (G, k) is a yes-instance of H-free Edge Editing(Deletion) if and only if $(\overline{G}, k)$ is a yes-instance of $\overline{H}$-free Edge Editing(Completion). It enables us to focus only on H-free Edge Editing and H-free Edge Deletion.

▶ **Proposition 4** (folklore). *Let H be any graph. Then H-free Edge Deletion is incompressible if and only if $\overline{H}$-free Edge Completion is incompressible. Similarly, H-free Edge Editing is incompressible if and only if $\overline{H}$-free Edge Editing is incompressible.*

For graphs H and H', by "*H simulates H'*" and by "*H' is simulated by H*", we mean that, there is a PPT from H'-free Edge Editing to H-free Edge Editing, there is a PPT from H'-free Edge Deletion to H-free Edge Deletion, and there is a PPT from H'-free Edge Completion to H-free Edge Completion. We observe that this is transitive, i.e., if H simulates H' and H' simulates H'', then H simulates H''. A set of graphs $\mathcal{H}$ is called a *base* for a set $\mathcal{G}$ of graphs if for every graph $H \in \mathcal{G}$ there is a graph $H' \in \mathcal{H}$ such that H simulates H'. The objective of the rest of the paper is to find, for each of the problems, a base $\mathcal{H} \cup \mathcal{X}$ for all graphs with at least five vertices, except the trivial cases, such that the following conditions are satisfied: (i) $\mathcal{H}$ is finite and the incompressibility is not known for any graph in it; (ii) for every graph in $\mathcal{X}$, the problem is known to be incompressible.

Proposition 4 implies Corollary 5 and Proposition 6 can be deduced directly from the definitions.

▶ **Corollary 5.** *Let H and H' be graphs such that H simulates H'. Then $\overline{H}$ simulates $\overline{H'}$.*

▶ **Proposition 6.** *Let $\mathcal{H}$ be a base for a set $\mathcal{G}$ of graphs. Assume that for every graph $H' \in \mathcal{H}$, H'-free Edge Editing (Deletion) is incompressible. Then for every graph $H \in \mathcal{G}$, H-free Edge Editing (Deletion) is incompressible.*

Intuitively, if H' is an induced subgraph of H, then H-free Edge Editing (Deletion) seems harder than H'-free Edge Editing (Deletion). However, there is no general argument why this should be true: there does not seem to be a completely general reduction that would reduce H'-free Edge Editing (Deletion) to H-free Edge Editing (Deletion). There is, however, a fairly natural idea for trying to do such a reduction: we extend the graph by attaching copies of $H - H'$ at every place where a copy of H' can potentially appear. The following construction is essentially the same as the main construction used in [2].

▶ **Construction 1** (see [2]). *Let (G', k, H, V') be an input to the construction, where G' and H are graphs, k is a positive integer and V' is a subset of vertices of H. We construct a graph G from G' as follows. For every injective function $f : V' \longrightarrow V(G')$, do the following:*

- *Introduce $k+1$ sets of vertices $V_1, V_2, \ldots, V_{k+1}$, each of size $|V(H) \setminus V'|$, and $k+1$ bijective functions $g_i : V(H) \longrightarrow (f(V') \cup V_i)$, for $1 \leq i \leq k+1$, such that $g_i(v') = f(v')$ for every $v' \in V'$;*

- *For each set V_i, introduce an edge set $E_i = \{(u,v) \mid u \in (f(V') \cup V_i), v \in V_i, (g_i^{-1}(u), g_i^{-1}(v)) \in E(H)\}$.*

This completes the construction. Let the constructed graph be G.

For convenience, we call every set V_i of vertices introduced in the construction as a *satellite* and the vertices in it as *satellite vertices*. This reduction works correctly in one direction: it ensures that the operations that make the new graph G H-free should ensure that the copy of G' inside G is H'-free.

▶ **Proposition 7** (see Lemma 2.6 in [2]). *Let G be obtained by Construction 1 on the input (G', k, H, V'), where G' and H are graphs, k is a positive integer and $V' \subseteq V(H)$. Then, if (G, k) is a yes-instance of H-free Edge Editing (Deletion), then (G', k) is a yes-instance of H'-free Edge Editing (Deletion), where H' is $H[V']$.*

However, the other direction of the correctness of the reduction does not hold in general (this is easy to see for example for $H = K_{1,2}$ and $H' = K_2$). As we shall see, there are particular cases where we can prove the converse of Proposition 7, for example, when $H - H'$ consists of exactly the highest- or lowest-degree vertices. Application of such arguments will be our main tool in reducing the complexity of H-free Edge Editing (Deletion) to simpler cases. Propositions 8 to 11 summarize the major results on the incompressibility of H-free edge modification problems known so far.

▶ **Proposition 8** ([4]). *Assuming* NP $\not\subseteq$ coNP/poly, *H-free Edge Editing, H-free Edge Deletion, and H-free Edge Completion are incompressible if H is either of the following graphs.*

- **(i)** *C_ℓ for any $\ell \geq 4$;*
- **(ii)** *P_ℓ for any $\ell \geq 5$;*
- **(iii)** *$2K_2$.*

We observe that Proposition 8(iii) follows from Proposition 8(i) and Proposition 4.

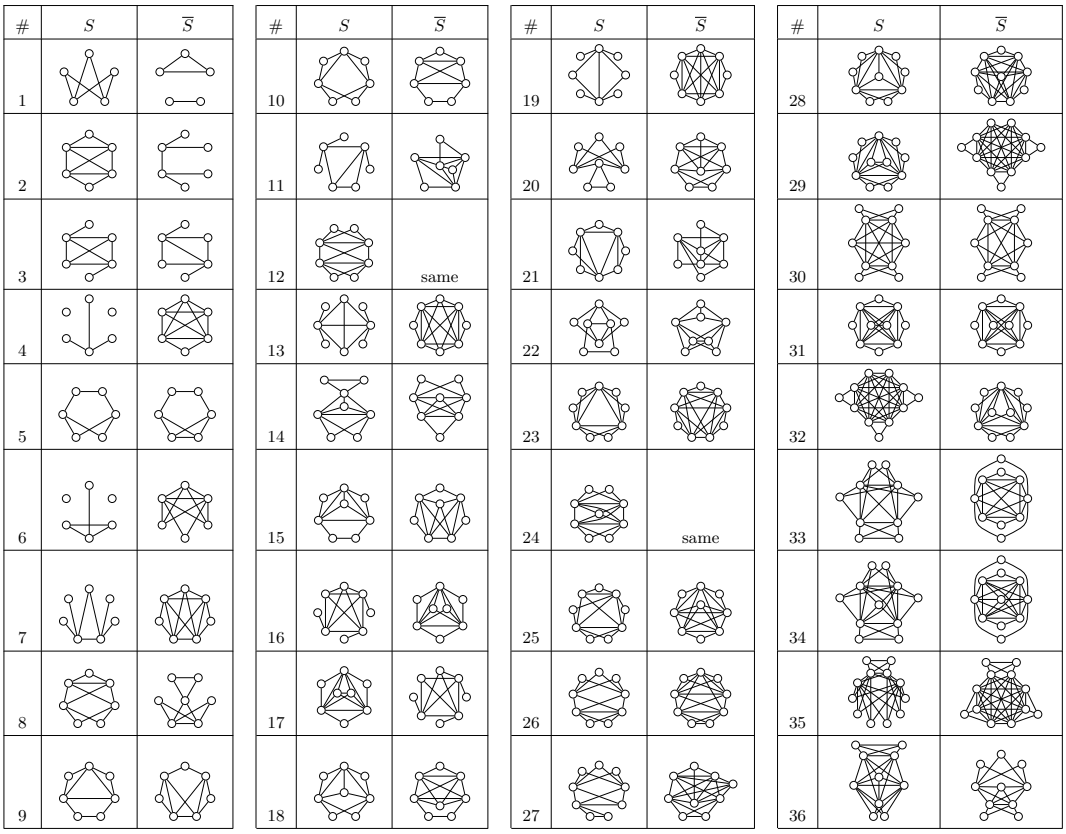
▶ **Proposition 9** ([4]). *Assuming* NP $\not\subseteq$ coNP/poly, *for 3-connected graphs H, H-free Edge Editing and H-free Edge Deletion are incompressible if H is not complete and H-free Edge Completion is incompressible if H has at least two nonedges.*

▶ **Proposition 10** ([4], folklore). *If H is a complete or empty graph, then H-free Edge Editing admits polynomial kernelization. If H is complete or has at most one edge then H-free Edge Deletion admits polynomial kernelization. If H is an empty graph or has at most one nonedge then H-free Edge Completion admits polynomial kernelization.*

▶ **Proposition 11.** *H-free Edge Editing, H-free Edge Deletion, and H-free Edge Completion admit polynomial kernels when H is a P_3 [6, 15], P_4 [16], paw [7, 13], or a diamond [5, 9].*

3 Churning

In this section, we introduce and analyze the churning procedure. The main result of the section is that assuming incompressibility for the class $\mathcal{W}$ of graphs defined in the previous section explains incompressibility for every graph with at least five vertices, except the trivial cases. Recall that $\mathcal{W}$ is not finite, as it contains the infinite families shown in Figure 3b. In Sections 4 and 5, we will further reduce $\mathcal{W}$ to a finite set.



(a) The set $\mathcal{S}$ of graphs.

#	$\mathcal{F}$	$\overline{\mathcal{F}}$	Comment
1	$K_{2,t}$	$K_t \cup K_2$	$4 \leq t$
2	$K_{1,t}$	$K_t \cup K_1$	$5 \leq t$
3	$K_2 \boxtimes tK_1$	$K_t \cup 2K_1$	$4 \leq t$
4	$T_{t,1}$	$\overline{T_{t,1}}$	$4 \leq t$
5	$\overline{(K_t - e) \cup 2K_1}$	$(K_t - e) \cup 2K_1$	$4 \leq t$
6	$\overline{(K_t - e) \cup K_2}$	$(K_t - e) \cup K_2$	$4 \leq t$
7	$K_{1,t} \cup K_2$	$\overline{K_{1,t} \cup K_2}$	$4 \leq t$
8	$\overline{(K_t - e) \cup K_1}$	$(K_t - e) \cup K_1$	$6 \leq t$
9	J_t	$\overline{J_t}$	$3 \leq t$
10	Q_t	$\overline{Q_t}$	$3 \leq t$

(b) The set $\mathcal{F}$ of infinite sets of graphs.

Figure 3

▶ **Lemma 12.** *If H-free Edge Editing is incompressible for every $H \in \mathcal{W}$, then H-free Edge Editing is incompressible for every H having at least five vertices but is neither complete nor empty, where the incompressibility assumes* NP $\not\subseteq$ coNP/poly.

▶ **Lemma 13.** *If H-free Edge Deletion is incompressible for every $H \in \mathcal{W}$, then H-free Edge Deletion is incompressible for every H having at least five vertices and at least two edges but not complete, where the incompressibility assumes* NP $\not\subseteq$ coNP/poly.

Corollary 14 follows from Lemma 13, Proposition 4 and from the fact that $\overline{\mathcal{W}} = \mathcal{W}$.

▶ **Corollary 14.** *If H-free Edge Completion is incompressible for every $H \in \mathcal{W}$, then H-free Edge Completion is incompressible for every H having at least five vertices and at least two nonedges but not empty, where the incompressibility assumes* NP $\not\subseteq$ coNP/poly.

By $\mathcal{X}_E$ we denote the set of all graphs (and their complements) listed in Proposition 8, Proposition 9, and Theorem 18 for which the incompressibility is known (assuming $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$) for H-free Edge Editing. By $\mathcal{Y}_E$, we denote the set of all graphs (and their complements) listed in Proposition 10 and 11 for which there exist polynomial kernels for H-free Edge Editing; additionally, we include into $\mathcal{Y}_E$ the claw and its complement (as we do not want to conjecture the incompressibility for these cases). Similarly, we define the set $\mathcal{X}_D$ of "hard" and the set $\mathcal{Y}_D$ of "nonhard" cases for H-free Edge Deletion. More formally,

$\mathcal{X}_D = \{C_\ell, \overline{C_\ell}$ for all $\ell \geq 4$,
$P_\ell, \overline{P_\ell}$ for all $\ell \geq 5$,
H such that H is regular but is neither complete nor empty,
H such that either H is 3-connected but not complete
or $\overline{H}$ is 3-connected with at least two nonedges$\}$

$\mathcal{X}_E = \mathcal{X}_D \cup \{H$ such that H has at most one edge and at least five vertices$\}$

$\mathcal{Y}_E = \{K_t, \overline{K_t}$ for all $t \geq 1$,
$P_3, \overline{P_3}, P_4$,
diamond, $\overline{\text{diamond}}$, paw, $\overline{\text{paw}}$, claw, $\overline{\text{claw}}\}$

$\mathcal{Y}_D = \mathcal{Y}_E \cup \{H$ such that H has at most one edge and at least five vertices$\}$

Additionally we define $\mathcal{Y}' = \{P_3, \overline{P_3}, P_4, \text{claw}, \overline{\text{claw}}, \text{paw}, \overline{\text{paw}}, \text{diamond}, \overline{\text{diamond}}\}$. We observe that $\mathcal{Y}' \subseteq \mathcal{Y}_E \cap \mathcal{Y}_D$ and the set of graphs with at most four vertices is a subset of $\mathcal{X}_E \cup \mathcal{Y}_E$ and $\mathcal{X}_D \cup \mathcal{Y}_D$. Further, we observe that near-empty graphs with at least five vertices are in $\mathcal{Y}_D$ but their complements are 3-connected and are in $\mathcal{X}_D$. We also note that both these graphs and their complements are in $\mathcal{X}_E$.

The main technical result of the section is the following lemma. It states that if a graph is not in the set $\mathcal{Y}_D$ of "easy" graphs, then it simulates a "hard" graph in $\mathcal{X}_\mathcal{D}$ or $\mathcal{W}$ (and there is a similar result for $\mathcal{X}_E$ and $\mathcal{Y}_E$).

▶ **Lemma 15.** *If $H \notin \mathcal{Y}_D$, then H simulates a graph in $\mathcal{X}_D \cup \mathcal{W}$. If $H \notin \mathcal{Y}_E$, then H simulates a graph in $\mathcal{X}_E \cup \mathcal{W}$.*

In the rest of the paper, integer ℓ and set V_ℓ denote the lowest degree and the set of lowest degree vertices in H respectively; integer h and set V_h denote the highest degree and the set of highest degree vertices in H respectively; and set V_m denotes the set $V(H) \setminus (V_\ell \cup V_h)$. By h^* we denote the degree of vertices of V_h in $\overline{H}$, i.e., $h^* = |V(H)| - h - 1$.

Now we introduce a procedure (Churn) which is similar to the one used to obtain dichotomy results on the polynomial-time solvable and NP-hard cases of these problems (see Section 5 in [2]). The basic observation is that H can simulate the graphs $H - V_\ell$ and $H - V_h$. This follows from proving that Construction 1 gives a PPT in these cases.

▶ **Proposition 16** (Corollary 2.9 in [2]). *Let H' be $H - V_\ell$ or $H - V_h$. Then H simulates H'.*

To deal with both H-free Edge Editing and H-free Edge Deletion in a uniform way, we define $\mathcal{X} = \mathcal{X}_E$ and $\mathcal{Y} = \mathcal{Y}_D$. We observe that $\overline{\mathcal{X} \cup \mathcal{Y}} = \mathcal{X} \cup \mathcal{Y}$ and $\mathcal{X} \cup \mathcal{Y} = \mathcal{X}_E \cup \mathcal{Y}_E = \mathcal{X}_D \cup \mathcal{Y}_D$.

Churn(H):
Step 1: If H is regular, then return H.
Step 2: If $H - V_\ell \notin \mathcal{Y}$, then return Churn($H - V_\ell$).
Step 3: If $H - V_h \notin \mathcal{Y}$, then return Churn($H - V_h$).
Step 4: Return H.

Proposition 16 implies Corollary 17.

▶ **Corollary 17.** *Let H' be the output of Churn(H). Then H simulates H'.*

We prove Lemma 15 by analyzing Churn() and showing that the graph returned by it always satisfies the requirements of the lemma. The procedure first handles the case when H is regular. In Section 3.1, we show that if H is regular, then it is safe to return H, as it is already in $\mathcal{X}_D \subseteq \mathcal{X}_E$. If H is not regular, then $H - V_\ell$ and $H - V_h$ are both defined. If one of these two graphs is not in $\mathcal{Y}$, then Proposition 16 allows us to proceed by recursion on that graph. Step 4 is reached when both $H - V_\ell$ and $H - V_h$ are in $\mathcal{Y}$. However, at this point the conditions on $H - V_\ell$ and $H - V_h$ give us important structural information about the graph H, which can be exploited to show that it is in $\mathcal{X}_D \cup \mathcal{W}$. Recall that $\mathcal{Y}$ is the union of complete, empty, near-empty, and the finite graphs in $\mathcal{Y}'$. This means we can split the problem into $4 \cdot 4$ different cases, with very strict structural restrictions on H in each case. These cases are analysed in a sequence of lemmas/corollaries (Lemma 19 to Lemma 34 in Sections 3.2–3.5).

3.1 Regular graphs

In this section, we handle the case when H is regular.

▶ **Theorem 18.** *Let H be a regular graph. Then H-free Edge Deletion, H-free Edge Completion, and H-free Edge Editing are incompressible if and only if H is neither complete nor empty, where the incompressibility assumes* NP $\not\subseteq$ coNP/poly.

3.2 Small graphs

If both $H - V_\ell$ and $H - V_h$ are in the in the finite set $\mathcal{Y}'$ of graphs, then H has bounded size. An exhaustive computer search showed the correctness of the procedure in this case.

▶ **Lemma 19.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that both $H - V_\ell$ and $H - V_h$ are in $\mathcal{Y}'$. Then $H \in \mathcal{W}$.*

3.3 Cliques and empty graphs

In this section, we consider the cases when both $H - V_\ell$ and $H - V_h$ are cliques or empty graphs. In this case, the structure of H is very limited. In principle, we need to consider four cases separately depending on the type of $H - V_\ell$ and $H - V_h$. However, a simple complementation argument shows that the case when both of them are cliques is equivalent to the case when both of them are empty.

▶ **Lemma 20.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that both $H - V_\ell$ and $H - V_h$ are complete graphs. Then $H \in \mathcal{W}$.*

Corollaries in this section and in Sections 3.4 and 3.5 use the facts that various sets we consider are self-complementary, i.e., $\mathcal{X} \cup \mathcal{Y} = \overline{\mathcal{X} \cup \mathcal{Y}}, \mathcal{W} = \overline{\mathcal{W}}, \mathcal{Y}' = \overline{\mathcal{Y}'}$.

▶ **Corollary 21.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that both $H - V_\ell$ and $H - V_h$ are empty graphs. Then $H \in \mathcal{W}$.*

▶ **Lemma 22.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that $H - V_\ell$ is a complete graph and $H - V_h$ is an empty graph. Then $H \in \mathcal{W}$.*

Our last case is when $H - V_\ell$ is empty and $H - V_h$ is complete. Let us observe that this case *does not* follow from Lemma 22 by complementation. If $\overline{V_\ell}$ and $\overline{V_h}$ are the lowest- and highest-degree vertices in $\overline{H}$, then $\overline{V_\ell} = V_h$, $\overline{V_h} = V_\ell$ and hence $\overline{H} - \overline{V_\ell}$ is empty and $\overline{H} - \overline{V_h}$ is a clique, that is, we have the same condition as for H. Fortunately, this last case is very simple to handle.

▶ **Lemma 23.** *There exists no graph $H \notin \mathcal{X} \cup \mathcal{Y}$ such that $H - V_\ell$ is an empty graph and $H - V_h$ is a complete graph.*

3.4 Cliques/empty graphs plus small graphs

Next we consider the cases when one of $H - V_\ell$ or $H - V_h$ is a clique or an empty graph, while the other is a graph from the finite set $\mathcal{Y}'$. Assuming that H is not too small, this means that H is essentially a clique or an empty graph, and intuitively it should follow that H or $\overline{H}$ is 3-connected, respectively. However, this requires a detailed proof considering several cases.

▶ **Lemma 24.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that $H - V_\ell \in \mathcal{Y}'$ and $H - V_h$ is a complete graph. Then $H \in \mathcal{W}$.*

▶ **Corollary 25.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that $H - V_\ell$ is an empty graph and $H - V_h \in \mathcal{Y}'$. Then $H \in \mathcal{W}$.*

▶ **Lemma 26.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that $H - V_\ell \in \mathcal{Y}'$ and $H - V_h$ is an empty graph. Then $H \in \mathcal{W}$.*

▶ **Corollary 27.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that $H - V_\ell$ is a complete graph and $H - V_h \in \mathcal{Y}'$. Then $H \in \mathcal{W}$.*

3.5 Near-empty graphs

Finally, we consider the cases when one of $H - V_\ell$ or $H - H_h$ is near empty. These cases are similar to the corresponding ones for empty graphs, but more technical and a higher number of corner cases need to be handled. Let us remark that this part of the proof is needed only for the H-free Edge Deletion problem: near-empty graphs are not in $\mathcal{Y}_E$, hence if our goal is to prove Theorem 1 for H-free Edge Editing, then the churning procedure can recurse on such graphs.

▶ **Lemma 28.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that $H - V_\ell$ is a complete graph and $H - V_h$ is a near-empty graph. Then $H \in \mathcal{W}$.*

▶ **Lemma 29.** *There exists no $H \notin \mathcal{X} \cup \mathcal{Y}$ such that $H - V_\ell$ is a near-empty graph and $H - V_h$ is a complete graph.*

▶ **Lemma 30.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ such that $H - V_\ell$ is an empty graph and $H - V_h$ is a near-empty graph. Then $H \in \mathcal{W}$.*

▶ **Lemma 31.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ such that $H - V_\ell$ is a near-empty graph and $H - V_h$ is an empty graph. Then $H \in \mathcal{W}$.*

▶ **Lemma 32.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that both $H - V_\ell$ and $H - V_h$ are near-empty graphs. Then $H \in \mathcal{W}$.*

▶ **Lemma 33.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that $H - V_\ell$ is a near-empty graph and $H - V_h \in \mathcal{Y}'$. Then $H \in \mathcal{W}$.*

▶ **Lemma 34.** *Let $H \notin \mathcal{X} \cup \mathcal{Y}$ be such that $H - V_\ell \in \mathcal{Y}'$ and $H - V_h$ is a near-empty graph. Then $H \in \mathcal{W}$.*

4 Reductions

Recall that we defined $\mathcal{W} = \mathcal{H} \cup \overline{\mathcal{H}} \cup \mathcal{A} \cup \overline{\mathcal{A}} \cup \mathcal{D} \cup \overline{\mathcal{D}} \cup \mathcal{B} \cup \overline{\mathcal{B}} \cup \mathcal{S} \cup \overline{\mathcal{S}} \cup \mathcal{F} \cup \overline{\mathcal{F}}$ and Section 3 reduced our main questions to assuming incompressibility for the set $\mathcal{W}$. In this section, we further refine the result and show that incompressibility needs to be assumed only for the finite set $\mathcal{W}' = \mathcal{H} \cup \overline{\mathcal{H}} \cup \mathcal{A} \cup \overline{\mathcal{A}} \cup \mathcal{B} \cup \mathcal{D}$. That is, we recall and introduce some further simple reductions and use them to prove that every graph in $\mathcal{W} \setminus \mathcal{W}'$ simulates a graph in $\mathcal{W}' \cup \mathcal{X}_D$. To begin with, we observe that deleting the lowest degree vertices in the graphs in $\overline{\mathcal{B}} \cup \overline{\mathcal{D}}$ results in 3-connected graphs which are not complete. Then by Proposition 16, we have:

▶ **Proposition 35.** *If $H \in \overline{\mathcal{B}} \cup \overline{\mathcal{D}}$, then H-free Edge Editing and H-free Edge Deletion are incompressible, assuming* NP $\not\subseteq$ coNP/poly.

The following reductions are based on Construction 1 and a few other similar constructions.

4.1 Reductions based on Construction 1

The following lemma can be proved using a straight-forward application of Construction 1.

▶ **Lemma 36.** *Let H be $J \cup K_t$, for some graph J and integer $t \geq 1$, where the K_t is induced by V_ℓ. Let V' be $V(H) \setminus \{v\}$, where v is any vertex in the K_t. Let H' be $H[V']$. Then H simulates H'. In particular, H simulates $J \cup K_1$.*

▶ **Corollary 37.**
- **(i)** *Let H be $K_t \cup K_2$, for $t \geq 4$ ($\in \overline{\mathcal{F}_1}$). Then H simulates $K_t \cup K_1$ ($\in \{\overline{H_5}\} \cup \overline{\mathcal{F}_2}$).*
- **(ii)** *Let H be $(K_t - e) \cup K_2$, for $t \geq 4$ ($\in \overline{\mathcal{F}_6}$). Then H simulates $(K_t - e) \cup K_1$ ($\in \{\overline{H_8}, \overline{D_2}\} \cup \overline{\mathcal{F}_8}$).*

Next we consider the removal of a path of degree-2 vertices. We can prove the correctness of the reduction only under a certain uniqueness condition on the path.

▶ **Lemma 38.** *Let H be a graph with minimum degree two and let $p \geq 2$ be an integer such that there is a unique induced path P of length p with the property that all the internal vertices of the path are having degree exactly two in H. Let H' be obtained from H by removing all internal vertices of P. Then H simulates H'.*

▶ **Corollary 39.** *Let H be J_t, for some $t \geq 3$ ($\in \mathcal{F}_9$). Then H simulates $K_2 \boxtimes tK_1$ ($\in \{\overline{H_3}\} \cup \mathcal{F}_3$).*

▶ **Corollary 40.** *Let H be Q_t, for some $t \geq 3$ ($\in \mathcal{F}_{10}$). Then H simulates $K_{2,t}$ ($\in \{S_1\} \cup \mathcal{F}_1$).*

▶ **Corollary 41.**
- **(i)** *S_5 simulates C_4.*
- **(ii)** *S_9 simulates $\overline{H_4}$.*
- **(iii)** *Let H be S_{15}. Then H simulates $\overline{H_9 \cup K_1}$. Further H simulates H_9 (Proposition 16).*
- **(iv)** *S_{22} simulates $\overline{diamond \cup K_2}$ ($\in \overline{\mathcal{F}_6}$).*

Lemma 42 essentially says the following: If H has vertex connectivity 1 and has a unique smallest 2-connected component which is a "leaf" in the tree formed by the 2-connected components, then H simulates a graph obtained by removing all vertices in the 2-connected component except the cut vertex.

▶ **Lemma 42.** *Let H be a graph with vertex connectivity 1 and be not a complete graph. Let $\mathcal{C}$ be the set of all 2-connected components of H having exactly one cut vertex of H. Assume that there exists a unique smallest (among $\mathcal{C}$) 2-connected component J in $\mathcal{C}$. Let v be the cut vertex of H in J. Let H' be $H - \{J \setminus \{v\}\}$. Then H' is simulated by H.*

▶ **Corollary 43.**

(i) *S_{20} simulates $K_{2,4}$ ($\in \mathcal{F}_1$).*

(ii) *S_{27} simulates $\overline{(K_5 - e) \cup K_2}$ ($\in \mathcal{F}_6$).*

▶ **Lemma 44.** *Let H be S_{35}. Then H simulates a 3-connected graph, which is not complete ($\in \mathcal{X}_D$).*

4.2 Reductions based on Construction 2

The following is a simplified version of Construction 1.

▶ **Construction 2.** *Let (G', k, ℓ) be an input to the construction, where G' is a graph and k and ℓ are positive integers. For every set S of ℓ vertices in G' introduce a clique C of $k+1$ vertices and make all the vertices in C adjacent to all the vertices in S.*

As before, we call every clique C introduced during the construction as a *satellite* and the vertices in it as *satellite vertices*. Lemma 45 can be proved using a straight-forward application of Construction 2. It says that if H satisfies some properties, then H simulates H' where H' is obtained by removing one vertex from each module of H contained within V_ℓ.

▶ **Lemma 45.** *Let H be a non-regular graph such that the following conditions hold true:*

(i) *$1 \leq \ell \leq 2, |V(H)| \geq 5$;*

(ii) *V_ℓ is an independent set, $V_h \cup V_m$ induces a connected graph, and every vertex in V_h is adjacent to at least one vertex in V_ℓ;*

(iii) *Every vertex in V_m has at least $\ell + 1$ neighbors outside V_m or there exists no pair u, v of adjacent vertices in V_m such that $N(u) \setminus \{v\} = N(v) \setminus \{u\}$.*

Consider a modular decomposition $\mathcal{M}$ of H. Let $\mathcal{M}' \subseteq \mathcal{M}$ corresponds to V_ℓ. Let H' be the graph obtained from H by removing one vertex from each module in $\mathcal{M}'$. Then H simulates H'.

The following corollary lists many graphs that can be handled by Lemma 45.

▶ **Corollary 46.**

(i) *S_7 simulates H_9.*

(ii) *S_8 simulates $\overline{A_1}$.*

(iii) *S_{10} simulates C_4.*

(iv) *S_{11} simulates $\overline{H_7}$.*

(v) *S_{12} simulates S_2.*

(vi) *S_{13} simulates S_3.*

(vii) *$\overline{S_{14}}$ simulates $\overline{B_1}$.*

(viii) *S_{18} simulates $\overline{A_1}$.*

(ix) *S_{19} simulates $\overline{S_3}$.*

(x) *S_{21} simulates $\overline{H_4}$.*

(xi) *S_{23} simulates $\overline{A_7}$.*

(xii) *S_{24} simulates $\overline{S_7}$.*

(xiii) *S_{25} simulates $\overline{H_1}$.*

(xiv) *S_{26} simulates S_8.*

(xv) *S_{28} simulates $\overline{A_9}$.*

(xvi) *S_{29} simulates S_{17}.*

(xvii) *S_{30} simulates $\overline{S_{19}}$.*

(xviii) *$\overline{S_{32}}$ simulates $\overline{S_{16}}$.*

(xix) *$\overline{S_{33}}$ simulates $\overline{K_{1,4} \cup K_2} \in \overline{\mathcal{F}_7}$.*

(xx) *$\overline{S_{34}}$ simulates $\overline{K_{1,5} \cup K_2} \in \overline{\mathcal{F}_7}$.*

(xxi) *$\overline{S_{36}}$ simulates $\overline{S_{14}}$.*

The following three corollaries are obtained by application of Lemma 45: they show that in certain families of graphs, every member simulates the simplest member. Corollary 47 deals with star graphs ($K_{1,t}$). For every graph H in this class, V_ℓ is a single module of the graph and H simulates a graph H', where H' is obtained by removing one vertex from V_ℓ. Corollary 48 handles $K_2 \boxtimes sK_1$, where V_ℓ forms a single module of the graph. As in the previous case, H' is obtained by removing one vertex from V_ℓ. Corollary 49 deals with the set of twin-star graphs (T_{t_1,t_2}). For every graph H in this class, there are two modules of H in V_ℓ : t_1 vertices adjacent to one vertex in $H - V_\ell$ and t_2 vertices adjacent to the other vertex in $H - V_\ell$. Then H simulates a graph H', where H' is obtained by removing one vertex each from the two modules.

▶ **Corollary 47** (see Lemma 6.4 in [2] for a partial result)**.** *Let H be $K_{1,t}$, for any $t \geq 5$ ($\in \mathcal{F}_2$). Let H' be $K_{1,t-1}$. Then H simulates H'. Furthermore, H simulates H_5 ($K_{1,4}$).*

▶ **Corollary 48** (see Lemma 4.5 in [1] for a partial result)**.** *Let H be $K_2 \boxtimes sK_1$, for any $s \geq 4$ ($\in \mathcal{F}_3$) and let H' be $K_2 \boxtimes (s-1)K_1$. Then H simulates H'. Furthermore, H simulates $\overline{H_3}$ ($K_2 \boxtimes 3K_1$).*

▶ **Corollary 49** (see Lemma 6.6 in [2] for a partial result)**.** *Let H be a twin-star graph T_{t_1,t_2}, such that $t_1, t_2 \geq 1$. Let H' be T_{t_1-1,t_2-1}. Then H simulates H'. In particular, if H is $T_{t,1}$, for some $t \geq 4$ ($\in \mathcal{F}_4$), then H simulates $K_{1,t}$ ($\in \{H_5\} \cup \mathcal{F}_2$).*

▶ **Lemma 50.** *$K_{2,3}$ ($= S_1$) simulates C_4.*

4.3 Reductions based on Construction 3

Now we give another construction that will be used in a few reductions.

▶ **Construction 3.** *Let (G', k, t) be an input to the construction, where G' is a graph and k and t are positive integers. For every set S of t vertices in G' introduce an independent set I_S of $k+2$ vertices such that every vertex in I_S is adjacent to every vertex in G' except those in S. Let $\bigcup_{S \subseteq V(G'), |S|=t} I_S = I$. Let the resultant graph be G.*

▶ **Lemma 51.** *Let H be a graph such that V_h forms a clique and for every pair of vertices $u, v \in V_h$, $H - u$ is isomorphic to $H - v$. Further assume that there exists no independent set S of size $s \geq 2$ where each vertex in S has degree at least $h - s + 1$ in H. Then H simulates $H - u$, where u is any vertex in V_h.*

Additional results used by Corollary 52 are shown in parenthesis.

▶ **Corollary 52.**

(i) *S_2 simulates $\overline{H_7}$.*
(ii) *S_3 simulates H_6.*
(iii) *S_{16} simulates S_3 (Proposition 16).*
(iv) *S_{17} simulates $\overline{H_1}$ (Proposition 16).*
(v) *S_{31} simulates S_3 (Proposition 16)*

▶ **Lemma 53.** *Let H be $(K_t - e) \cup K_1$ for $t \geq 6$ ($\overline{\mathcal{F}_8}$). Let H' be $K_{t-2} \cup K_1$ ($\in \{\overline{H_5}\} \cup \overline{\mathcal{F}_2}$). Then H simulates H'.*

4.4 Other reductions

To resolve graphs in $\mathcal{F}_7$ ($= K_{1,t} \cup K_2$), we resort to a known reduction. There is a PPT in [2] from H'-free Edge Editing to H-free Edge Editing, where H' is a largest component in H. It is a composition of two reductions: one from H'-free Edge Editing to H''-free Edge Editing and another from H''-free Edge Editing to H-free Edge Editing, where H'' is the union of all components in H isomorphic to H'. The first reduction uses a simple construction (take a disjoint union of the input graph and join of $k+1$ copies of H') and the second reduction uses Construction 1.

▶ **Proposition 54** (see Lemma 3.5 in [2]). *Let H' be a largest component of H. Then H simulates H'.*

▶ **Corollary 55.** *Let H be $K_{1,t} \cup K_2$, for $t \geq 4$ ($\in \mathcal{F}_7$). Then H simulates $K_{1,t}$ ($\in \{H_5\} \cup \mathcal{F}_2$).*

The following statement consider reduction that involve the removal of independent vertices.

▶ **Lemma 56.** *Let H be $J \cup tK_1$, for any $t \geq 2$ such that J has no component which is a clique. Let H' be $J \cup (t-1)K_1$. Then H simulates H'. In particular, H simulates $J \cup K_1$.*

▶ **Corollary 57.**

(i) *S_4 simulates H_2.*
(ii) *S_6 simulates H_6.*
(iii) *Let H be $(K_t - e) \cup 2K_1$, for $t \geq 4$ ($\in \overline{\mathcal{F}_5}$). Then H simulates $(K_t - e) \cup K_1$ ($\in \{\overline{H_8}, \overline{D_2}\} \cup \overline{\mathcal{F}_8}$).*

Summary of results in this section handling graphs in $\mathcal{S}$ and $\mathcal{F}$ are given in Figure 4a and 4b respectively. Lemma 58 follows from Corollary 5, Proposition 35, the transitivity of PPTs, and other results in this section (see Figures 4a and 4b) for details.

▶ **Lemma 58.** *Let $H \in \mathcal{W} \setminus \mathcal{W}'$. Then H simulates a graph in $\mathcal{W}' \cup \mathcal{X}_D$.*

5 Incompressibility results for the graphs in $\mathcal{A}$ and $\mathcal{B}$

In this section, we prove that for every graph $H \in \mathcal{A} \cup \overline{\mathcal{A}}$, all three problems H-free Edge Editing, H-free Edge Deletion, and H-free Edge Completion are incompressible, assuming NP $\not\subseteq$ coNP/poly. With the same assumption, we prove that H-free Edge Deletion is incompressible for every graph $H \in \mathcal{B}$; Proposition 4 then implies incompressibility of H-free Edge Completion for every $H \in \overline{\mathcal{B}}$.

▶ **Theorem 59.** *Assuming* NP $\not\subseteq$ coNP/poly*:*

(i) *Let $H \in \mathcal{A}$. Then H-free Edge Editing is incompressible.*
(ii) *Let $H \in \mathcal{A} \cup \overline{\mathcal{A}} \cup \mathcal{B}$. Then H-free Edge Deletion is incompressible.*
(iii) *Let $H \in \mathcal{A} \cup \overline{\mathcal{A}} \cup \overline{\mathcal{B}}$. Then H-free Edge Completion is incompressible.*

We apply the technique used by Cai and Cai [4] by which they obtained a complete dichotomy on the incompressibility of H-free edge modification problems on 3-connected graphs H. We will give a self-contained summary of their proof technique, with only a few references to proofs of formal statements. The reader is referred to [4] for a more detailed exposition of terminology and concepts discussed in this section.

The first step in the proof is to establish incompressibility for the *restricted* versions of H-free Edge Deletion and H-free Edge Completion, where only allowed edges can be deleted/added. Then deletion and completion *enforcer gadgets* can be used to reduce

H	Simulates	By
S_1	C_4	Lemma 50
S_2	$\overline{H_7}$	Corollary 52
S_3	H_6	Corollary 52
S_4	H_2	Corollary 57
S_5	C_4	Corollary 41
S_6	H_6	Corollary 57
S_7	H_9	Corollary 46
S_8	$\overline{A_1}$	Corollary 46
S_9	$\overline{H_4}$	Corollary 41
S_{10}	C_4	Corollary 46
S_{11}	$\overline{H_7}$	Corollary 46
S_{12}	S_2	Corollary 46

H	Simulates	By
S_{13}	S_3	Corollary 46
S_{14}	B_1	Corollary 46
S_{15}	H_9	Corollary 41
S_{16}	S_3	Corollary 52
S_{17}	$\overline{H_1}$	Corollary 52
S_{18}	$\overline{A_1}$	Corollary 46
S_{19}	$\overline{S_3}$	Corollary 46
S_{20}	a graph in $\mathcal{F}_1$	Corollary 43
S_{21}	$\overline{H_4}$	Corollary 46
S_{22}	a graph in $\overline{\mathcal{F}_6}$	Corollary 41
S_{23}	$\overline{A_7}$	Corollary 46
S_{24}	$\overline{S_7}$	Corollary 46

H	Simulates	By
S_{25}	$\overline{H_1}$	Corollary 46
S_{26}	S_8	Corollary 46
S_{27}	a graph in $\mathcal{F}_6$	Corollary 43
S_{28}	$\overline{A_9}$	Corollary 46
S_{29}	S_{17}	Corollary 46
S_{30}	$\overline{S_{19}}$	Corollary 46
S_{31}	S_3	Corollary 52
S_{32}	S_{16}	Corollary 46
S_{33}	a graph in $\mathcal{F}_7$	Corollary 46
S_{34}	a graph in $\mathcal{F}_7$	Corollary 46
S_{35}	a graph in $\mathcal{X}_D$	Lemma 44
S_{36}	S_{14}	Corollary 46

(a) Summary of results in Section 4 handling graphs in $\mathcal{S}$.

$H \in$	Simulates a graph in	By
$\mathcal{F}_1$	$\{H_5\} \cup \mathcal{F}_2$	Corollary 37
$\mathcal{F}_2$	$\{H_5\}$	Corollary 47
$\mathcal{F}_3$	$\{\overline{H_3}\}$	Corollary 48
$\mathcal{F}_4$	$\{H_5\} \cup \mathcal{F}_2$	Corollary 49
$\mathcal{F}_5$	$\{H_8, D_2\} \cup \mathcal{F}_8$	Corollary 57

$H \in$	Simulates a graph in	By
$\mathcal{F}_6$	$\{H_8, D_2\} \cup \mathcal{F}_8$	Corollary 37
$\mathcal{F}_7$	$\{H_5\} \cup \mathcal{F}_2$	Corollary 55
$\mathcal{F}_8$	$\{H_5\} \cup \mathcal{F}_2$	Lemma 53
$\mathcal{F}_9$	$\{\overline{H_3}\} \cup \mathcal{F}_3$	Corollary 39
$\mathcal{F}_{10}$	$\{S_1\} \cup \mathcal{F}_1$	Corollary 40

(b) Summary of results in Section 4 handling graphs in $\mathcal{F}$.

Figure 4

the restricted problems to the original versions. Cai and Cai [4] presented constructions that were proved to work correctly when H is 3-connected. We show, by careful inspection, that the same technique works for certain graphs H that are not 3-connected. For certain graphs H, we can prove incompressibility of the restricted problem, but enforcer gadgets of the required form provably do not exist. In these cases, we use ad hoc ideas to reduce the restricted version to the original one. In yet further cases, we need even trickier reductions, where we reduce H'-free Edge Deletion to H-free Edge Deletion for some $H' \neq H$.

5.1 Incompressibility results for the restricted problems

A graph is called *edge-restricted* if a subset of its edges are marked as *forbidden*. All edges other than forbidden are *allowed*. A graph is called *nonedge-restricted* if a subset of its nonedges are marked as *forbidden*. All nonedges other than forbidden are *allowed*.

Restricted H-free Edge Deletion: Given a graph G, an integer k, and a set R of edges of G, do there exist at most k edges disjoint from R such that deleting them from G results in an H-free graph? **Parameter**: k

Restricted H-free Edge Completion: Given a graph G, an integer k, and a set R of nonedges of G, do there exist at most k nonedges disjoint from R such that adding them in G results in an H-free graph? **Parameter**: k

Propagational formula satisfiability. A ternary Boolean function $f(x, y, z)$ (where x, y, and z are either Boolean variables or constants 0 or 1) is *propagational* if $f(1, 0, 0) = 0, f(0, 0, 0) = f(1, 0, 1) = f(1, 1, 0) = f(1, 1, 1) = 1$. This has the meaning: if x is true then either y is true or z is true.

Propagational-f Satisfiability: Given a conjunctive formula φ of a propagational ternary function f with distinct variables in each clause of φ, find whether there exists a satisfying truth assignment with weight at most k. The parameter we consider is k.

▶ **Proposition 60** (Theorem 3.4 in [4]). *For any propagational ternary Boolean function f,* Propagational-f Satisfiability *on 3-regular conjunctive formulas (every variable appears exactly three times) admits no polynomial kernel, assuming* NP $\not\subseteq$ coNP/poly.

Satisfaction-testing components. For H-free Edge Deletion, a *satisfaction-testing component* $S_D(x, y, z)$ is a constant-size edge-restricted H-free graph with exactly three allowed edges $\{x, y, z\}$ such that there is a propagational Boolean function $f(x, y, z)$ such that $f(x, y, z) = 1$ if and only if the graph obtained from $S_D(x, y, z)$ by deleting edges in $\{x, y, z\}$ with value 1 is H-free. For H-free Edge Completion, a *satisfaction-testing component* $S_C(x, y, z)$ is a constant-size nonedge-restricted H-free graph with exactly three allowed nonedges $\{x, y, z\}$ such that there is a propagational Boolean function $f(x, y, z)$ such that $f(x, y, z) = 1$ if and only if the graph obtained from $S_C(x, y, z)$ by adding edges in $\{x, y, z\}$ with value 1 is H-free.

There is an easy construction (Lemma 4.3 in [4]) showing that $S_D(x, y, z)$ exists for every connected graph H with at least four vertices but not complete and $S_C\{x, y, z\}$ exists for every connected graph with at least four vertices and at least two nonedges. The construction for this is as follows. $S_D\{x, y, z\}$: Let x be a nonedge, and y and z be two edges in H. Then $H + x$ is a $S_D(x, y, z)$ where x, y, z are the only allowed edges. $S_C\{x, y, z\}$: Let x be an edge, and y and z be two nonedges in H. Then $H - x$ is a $S_C(x, y, z)$ where x, y, z are the only allowed nonedges.

Truth-setting components. For H-free Edge Deletion, a *truth-setting component* $(T_D(u))$ is a constant-sized, edge-restricted H-free graph such that it contains at least three allowed edges x, y, z without a common vertex and admits exactly two deletion sets $\emptyset$ and the set of all allowed edges. For H-free Edge Completion, a *truth-setting component* $(T_C(u))$ is a constant-sized, nonedge-restricted H-free graph such that it contains at least three allowed nonedges x, y, z without a common vertex and admits exactly two completion sets $\emptyset$ and the set of all allowed nonedges.

There is a construction given in [4] for $T_D(u)$ and $T_C(u)$ when H is 3-connected but not complete. The constructions are given below.

Construction of $T_D(u)$: Let e', e be a nonedge and an edge sharing no common vertex in H. Let the *basic unit* $U = H + e'$ and set all edges except e and e' in U as forbidden. Let p be the number of vertices in H. Take p copies $U_1, U_2, \ldots, U_p$ of U. Identify the edge e of U_i with the edge e' of U_{i+1} to form a chain of U's. This is a *basic chain* $B(u)$. Let us call the unidentified edge e' of U_1 as the *left-most allowed edge* of $B(u)$ and unidentified edge of U_p as the *right-most allowed edge* of $B(u)$. Take three basic chains B_0, B_1, and B_2. Attach them in a cyclic fashion: Identify the right-most allowed edge of B_i with the left-most allowed edge of B_{i+1}, where indices are taken mod 3. This is the claimed truth-setting component $T_D(u)$. Let us call the allowed edges thus identified as *variable edges*. We note that there are exactly three variable edges in $T_D(u)$.

It is easy to see that, for every H, there are only two possible deletion sets in $T_D(u)$: the empty set and the set of all allowed edges. To see this, observe that if we remove any of the allowed edges, then it creates a copy of H in one of the units, forcing us to remove the next allowed edge as well. However, it is not clear if these two deletion sets really make the graph H free. As Cai and Cai [4] show, this construction for $T_D(u)$ works correctly for 3-connected graphs H: Since the "cycle" of basic units is long enough, every subgraph having vertices from different basic units and having at most $|V(H)|$ vertices has vertex connectivity at most 2. In general, the construction may not give correct truth-setting components for 2-connected graphs H. But, as we shall see later, by carefully choosing e and e' in these constructions, we can obtain truth-setting components for many 2-connected graphs H.

Construction of $T_C(u)$: Let e', e be a nonedge and an edge sharing no common vertex in H. Let the *basic unit* $U = H - e$ and set all nonedges except e and e' in U as forbidden. Let p be the number of vertices in H. Take p copies $U_1, U_2, \ldots, U_p$ of U. Identify the nonedge e of U_i with the nonedge e' of U_{i+1} to form a chain of U's. This is a *basic chain* $B(u)$. Let us call the unidentified nonedge e' of U_1 as the *left-most allowed nonedge* of $B(u)$ and unidentified nonedge of U_p as the *right-most allowed nonedge* of $B(u)$. Take three basic chains B_0, B_1, and B_2. Attach them in a cyclic fashion: Identify the right-most allowed nonedge of B_i with the left-most allowed nonedge of B_{i+1}, where indices are taken mod 3. This is the claimed truth-setting component $T_C(u)$. Let us call the allowed nonedges thus identified as *variable nonedges*. We note that there are exactly three variable nonedges in $T_C(u)$. Similarly to $T_D(u)$, we can argue that for any H, there are only two potential completion sets (the empty set and the set of all allowed nonedges), and for 3-connected H, these two sets are indeed completion sets.

The following is the construction used in the reduction from Propagational-f Satisfiability to Restricted H-free Edge Deletion (Completion).

▶ **Construction 4.** *Let (φ, k, H) be an input to the construction, where φ is a 3-regular conjunctive formula on a propagational ternary Boolean function f, and k is a positive integer. The construction gives a graph G_φ, an integer k', and a set of restricted (non)edges in G_φ.*

- *For every clause in φ, introduce a satisfaction-testing component $S_D(x, y, z)$ ($S_C(x, y, z)$) for H-free Edge Deletion (Completion).*
- *If $c \in \{x, y, z\}$ is 1, then the corresponding allowed (non)edge is deleted(added) and if $c = 0$ then the corresponding allowed (non)edge is set as forbidden.*
- *For every variable u in f, introduce a truth-setting component $T_D(u)$ ($T_C(u)$) for H-free Edge Deletion (Completion)*
- *For every variable u, identify each of the variable (non)edges in $T_D(u)$ ($T_C(u)$) with an allowed (non)edge in a satisfaction-testing component corresponds to a different clause in which u appears – since φ is 3-regular, u appears in exactly three clauses.*

Let the graph obtained be G_φ and let $k' = 3|V(H)|k$. For the deletion problem the set R of forbidden edges is all the edges in G_φ except the allowed edges in the units. For the completion problem, the set R of forbidden nonedges contains every nonedge of G_φ except the allowed nonedges in the units.

Let H be a graph and (φ, k) be an instance of a Propagational-f Satisfiability problem. Let (G_φ, k', R) be the output of the Construction 4 applied on (φ, k, H). The construction works correctly in one direction: If (G_φ, k', R) is a yes-instance of Restricted H-free Edge Deletion (Completion), then (φ, k) is a yes-instance of Propagational-f Satisfiability. To see this, let F be a solution of (G_φ, k', R). By the definition of $T_D(u)$ ($T_C(u)$), if an allowed (non)edge is in F then so is every allowed (non)edge in it. Therefore,

since $|F| \leq k' = 3k|V(H)|$ and every truth-setting component has exactly $3|V(H)|$ many allowed (non)edges, only at most k (non)edges of satisfaction-testing components are in F. By the definition of $S_D(x,y,z)$ ($S_C(x,y,z)$), if $x \in F$ then either y or z is in F, otherwise there is an induced H in $G_\varphi + F$. Therefore, setting the variables to 1 corresponding to the (non)edges, which are part of F, in satisfaction-testing components, we obtain that (φ, k) is a yes-instance of Propagational-f Satisfiability. Thus we have the following Proposition.

▶ **Proposition 61** (see Lemma 5.1 in [4]). *Let (φ, k) be an instance of* Propagational-f Satisfiability. *For a graph H, let (G_φ, k', R) be obtained by applying Construction 4 on (φ, k, H). Then, if (G_φ, k', R) is a yes-instance of* Restricted H-free Edge Deletion *(Completion) then (φ, k) is a yes-instance of* Propagational-f Satisfiability.

We remark that the proof of Proposition 61 works even if we use a gadget for the truth-setting component which satisfies only a weak property: it has at most two deletion (completion) sets, the $\emptyset$ and the set of all allowed (non)edges. As we have seen, the construction of $T_D(u)$ and $T_C(u)$ discussed above satisfies this weak property. To prove the other direction, one needs to show that there is no induced H in the "vicinity" of a satisfaction-testing component after deleting (adding) the (non)edges corresponding to the variables being set to 1 in φ. This can be done very easily for 3-connected graphs H. Proving this direction for 2-connected graphs H (if provable) requires careful structural analysis of the constructed graph G_φ. In Figure 5, we give various gadgets required for the proofs of this section. We use *unit* as a general term to refer to a satisfaction-testing component or a basic unit.

▶ **Lemma 62.** *Let $H \in \{\overline{A_1}, \overline{A_2}, A_3, \overline{A_3}, A_4, A_5, \overline{A_7}, \overline{A_9}\}$. Then* Restricted H-free Edge Deletion *is incompressible, assuming* NP $\not\subseteq$ coNP/poly.

The following corollary follows from the fact that there is no subgraph isomorphic to a C_4 where all edges are allowed in the graph G_φ constructed in the proof of Lemma 62 for Restricted H-free Edge Deletion, when H is a $\overline{A_1}$. We will be using this result later to handle $\overline{A_7}$.

▶ **Corollary 63.** *Let H be $\overline{A_1}$. Then, assuming* NP $\not\subseteq$ coNP/poly, Restricted H-free Edge Deletion *is incompressible even if input graphs does not contain a subgraph (not necessarily induced) isomorphic to H where all the side-edges of the diamond (a side-edge of a diamond is an edge incident to a degree-2 vertex in the diamond) in the subgraph are allowed.*

▶ **Lemma 64.** *Let $H \in \{\overline{A_2}, \overline{A_7}, \overline{A_8}, \overline{A_9}, \overline{B_1}, \overline{B_2}, \overline{B_3}\}$. Then* Restricted H-free Edge Completion *is incompressible, assuming* NP $\not\subseteq$ coNP/poly.

5.2 Using enforcers to reduce to the unrestricted problems

If we want to reduce Restricted H-free Edge Deletion to H-free Edge Deletion, then there is a fairly natural idea to try: for each restricted edge $e' = x'y'$, we introduce a copy of H on set U of new vertices and identify $x'y'$ with xy, where $x, y \in U$ are nonadjacent vertices. Now U induces a copy of H plus an extra edge, but as soon as e' is deleted, it becomes a copy of H, effectively preventing the deletion of e'.

There are two problems with this approach. First, the solution could delete other edges from the new copy of H, and then it is not necessarily true that the removal of e' automatically creates an induced copy of H. However, this problem is easy to avoid by repeating this

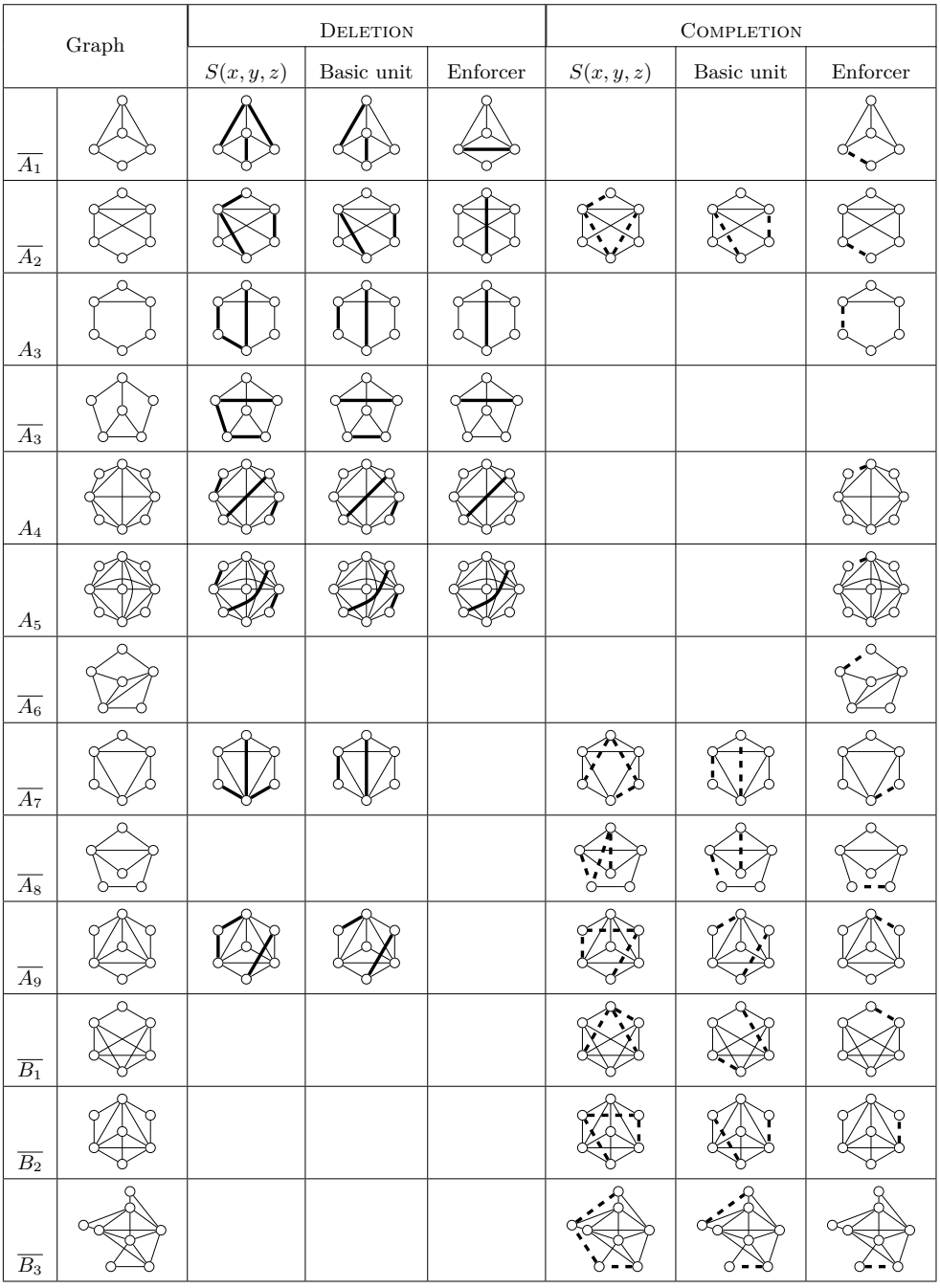


Figure 5 Various gadgets used in the proofs of this section. In a satisfaction-testing component $S_D(x,y,z)$ ($S_C(x,y,z)$), x is the darkened (non)edge added(deleted) in H to obtain the gadget, and y and z are the other two darkened (non)edges. Both the allowed (non)edges in basic units are darkened. The distinguished edge in a deletion enforcer and the distinguished nonedge in a completion enforcer are darkened.

gadget construction $k+1$ times: a solution of size at most k cannot interfere with all $k+1$ gadgets. The second problem is more serious: it is possible that attaching the new vertices creates a copy of H, even when e is not deleted. For certain graphs H, with a careful choice of x and y we can ensure that this does not happen: no induced copy of H can go through the separator x, y.

An *H-free deletion enforcer* (X, e) consists of an H-free graph X and a distinguished edge e in X such that (a) $X - e$ contains an induced H, and (b) for any graph G vertex disjoint with X, and any edge e' of G, all induced copies of H in the graph obtained by attaching X to G through identifying e with e' reside entirely inside G. Similarly, an *H-free completion enforcer* (X, e) consists of an H-free graph X and a distinguished nonedge e such that (a) $X + e$ contains an induced H, and (b) for any graph G vertex disjoint with X, and any nonedge e' in G, all induced copies of H in the graph obtained by attaching X to G through identifying e with e' reside entirely inside G. It can be shown that if we can come up with enforcer gadgets satisfying these conditions, then the ideas sketched above can be made to work, and we obtain a reduction from the restricted problem to the unrestricted version.

▶ **Proposition 65** (See Lemma 6.5 in [4]). *For a graph H:*

(i) *If* Restricted H-free Edge Deletion *is incompressible and there exists an H-free deletion enforcer, then H-*free Edge Deletion *is incompressible.*

(ii) *If* Restricted H-free Edge Completion *is incompressible and there exists an H-free completion enforcer, then H-*free Edge Completion *is incompressible.*

(iii) *If H-*free Edge Deletion *is incompressible and there exists an H-free completion enforcer, then H-*free Edge Editing *is incompressible.*

In the rest of the section, we establish the existence of enforcer gadgets for certain graphs H.

▶ **Lemma 66.** *Let $H \in \{\overline{A_1}, \overline{A_2}, A_3, \overline{A_3}, A_4, A_5\}$. Then the gadget X with a distinguished edge e shown in the corresponding cell in the column "Enforcer" (under* Deletion*) in Figure 5 is an H-free deletion enforcer.*

▶ **Lemma 67.** *Let $H \in \{\overline{A_1}, \overline{A_2}, A_3, A_4, A_5, \overline{A_6}, \overline{A_7}, \overline{A_8}, \overline{A_9}, \overline{B_1}, \overline{B_2}, \overline{B_3}\}$. Then the gadget X with a distinguished nonedge e shown in the corresponding cell in the column "Enforcer" (under* Completion*) in Figure 5 is an H-free completion enforcer.*

▶ **Lemma 68.** *Let $H \in \{\overline{A_1}, \overline{A_2}, A_3, \overline{A_3}, A_4, A_5\}$. Then H-*free Edge Deletion *and H-*free Edge Editing *are incompressible, assuming* NP $\not\subseteq$ coNP/poly.

Similarly, we can prove Lemma 69. The cases of H being A_4 or A_5 follows from the fact that H and $\overline{H}$ are isomorphic (see Proposition 4).

▶ **Lemma 69.** *Let $H \in \{\overline{A_2}, A_4, A_5, \overline{A_7}, \overline{A_8}, \overline{A_9}, \overline{B_1}, \overline{B_2}, \overline{B_3}\}$. Then H-*free Edge Completion *is incompressible, assuming* NP $\not\subseteq$ coNP/poly.

5.3 Further tricky reductions

There are graphs for which we can show that no completion/deletion enforcers, as defined in the previous section, exist (this can be checked by going through every pair x, y of (non)adjacent vertices). For some of these graphs, we can find a different way of enforcing that certain edges are forbidden; typically, we introduce some vertices that are used globally by every enforcer gadget. Furthermore, there are graphs H, where we were unable to obtain a reduction from Restricted H-free Edge Deletion (Completion), but could choose an induced subgraphs $H' \subseteq H$ and obtain a reduction from Restricted H'-free Edge Deletion (Completion), whose incompressibility was established earlier.

▶ **Lemma 70.** *Assuming* NP $\not\subseteq$ coNP/poly, H-free Edge Editing *and* H-free Edge Deletion *are incompressible, when* $H \in \{\overline{A_6}, \overline{A_7}, \overline{A_8}, \overline{A_9}\}$ *and* H-free Edge Completion *is incompressible when* $H \in \{\overline{A_1}, \overline{A_6}\}$.

Now, Theorem 59(i) follows from Lemma 68, Lemma 70, and Proposition 4. Similarly, Theorem 59(ii) follows from Lemma 68, 70, 69, and Proposition 4. Theorem 59(iii) follows from Theorem 59(ii) and Proposition 4. Theorem 1 follows from Lemma 12, 58, Theorem 59(i), and Proposition 6. Similarly, Theorem 2 follows from Lemma 13, 58, Theorem 59(ii), and Proposition 6.

References

1. N. R. Aravind, R. B. Sandeep, and Naveen Sivadasan. Parameterized lower bounds and dichotomy results for the NP-completeness of H-free edge modification problems. In *Proc. LATIN 2016*, pages 82–95, 2016. doi:10.1007/978-3-662-49529-2_7.
2. N. R. Aravind, R. B. Sandeep, and Naveen Sivadasan. Dichotomy results on the hardness of H-free edge modification problems. *SIAM J. Discrete Math.*, 31(1):542–561, 2017. doi:10.1137/16M1055797.
3. Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.
4. Leizhen Cai and Yufei Cai. Incompressibility of H-free edge modification problems. *Algorithmica*, 71(3):731–757, 2015. doi:10.1007/s00453-014-9937-x.
5. Yufei Cai. Polynomial kernelisation of H-free edge modification problems. Mphil thesis, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong SAR, China, 2012.
6. Yixin Cao and Jianer Chen. Cluster editing: Kernelization based on edge cuts. *Algorithmica*, 64(1):152–169, 2012. doi:10.1007/s00453-011-9595-1.
7. Yixin Cao, Yuping Ke, and Hanchun Yuan. Polynomial kernels for paw-free edge modification problems. In *Proc. TAMC 2020*, pages –, 2020.
8. Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016. doi:10.1007/s00453-015-0014-x.
9. Yixin Cao, Ashutosh Rai, R. B. Sandeep, and Junjie Ye. A polynomial kernel for diamond-free editing. In *Proc. ESA 2018*, pages 10:1–10:13, 2018. doi:10.4230/LIPIcs.ESA.2018.10.
10. Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification. *arXiv*, 2020. arXiv:2001.06867.
11. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
12. Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, Erik Jan van Leeuwen, and Marcin Wrochna. Polynomial kernelization for removing induced claws and diamonds. *Theory Comput. Syst.*, 60(4):615–636, 2017. doi:10.1007/s00224-016-9689-x.
13. Eduard Eiben, William Lochet, and Saket Saurabh. A polynomial kernel for paw-free editing. *arXiv*, 2019. arXiv:1911.03683.
14. Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.
15. Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Graph-modeled data clustering: Fixed-parameter algorithms for clique generation. In *Proc CIAC 2003*, pages 108–119, 2003. doi:10.1007/3-540-44849-7_17.
16. Sylvain Guillemot, Frédéric Havet, Christophe Paul, and Anthony Perez. On the (non-)existence of polynomial kernels for P_ℓ-free edge modification problems. *Algorithmica*, 65(4):900–926, 2013. doi:10.1007/s00453-012-9619-5.

17 Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006. doi:10.1016/j.jcss.2006.02.001.

18 Ken-ichi Kawarabayashi and Bruce A. Reed. Computing crossing number in linear time. In *Proc. STOC 2007*, pages 382–390. ACM, 2007. doi:10.1145/1250790.1250848.

19 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-Complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

20 Brendan McKay. *Graphs*, (accessed June 11, 2017). https://users.cecs.anu.edu.au/ bdm/data/-graphs.html.

21 Mihalis Yannakakis. Edge-deletion problems. *SIAM J. Comput.*, 10(2):297–309, 1981. doi:10.1137/0210021.

22 Mihalis Yannakakis. Node-deletion problems on bipartite graphs. *SIAM J. Comput.*, 10(2):310–327, 1981. doi:10.1137/0210022.