Hardness of Approximation for H-Free Edge Modification Problems

Ivan Bliznets[*1], **Marek Cygan**[†2], **Paweł Komosa**[†3], **and Michał Pilipczuk**[‡4]

1 **St. Petersburg Department of Steklov Institute of Mathematics, Russia**
iabliznets@gmail.com
2 **Institute of Informatics, University of Warsaw, Poland**
cygan@mimuw.edu.pl
3 **Institute of Informatics, University of Warsaw, Poland**
p.komosa@mimuw.edu.pl
4 **Institute of Informatics, University of Warsaw, Poland**
michal.pilipczuk@mimuw.edu.pl

Abstract

The H-free Edge Deletion problem asks, for a given graph G and integer k, whether it is possible to delete at most k edges from G to make it *H-free*, that is, not containing H as an induced subgraph. The H-free Edge Completion problem is defined similarly, but we add edges instead of deleting them. The study of these two problem families has recently been the subject of intensive studies from the point of view of parameterized complexity and kernelization. In particular, it was shown that the problems do not admit polynomial kernels (under plausible complexity assumptions) for almost all graphs H, with several important exceptions occurring when the class of H-free graphs exhibits some structural properties.

In this work we complement the parameterized study of edge modification problems to H-free graphs by considering their approximability. We prove that whenever H is 3-connected and has at least two non-edges, then both H-free Edge Deletion and H-free Edge Completion are very hard to approximate: they do not admit poly(OPT)-approximation in polynomial time, unless P = NP, or even in time subexponential in OPT, unless the Exponential Time Hypothesis fails. The assumption of the existence of two non-edges appears to be important: we show that whenever H is a complete graph without one edge, then H-free Edge Deletion is tightly connected to the Min Horn Deletion problem, whose approximability is still open. Finally, in an attempt to extend our hardness results beyond 3-connected graphs, we consider the cases of H being a path or a cycle, and we achieve an almost complete dichotomy there.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases hardness of approximation, parameterized complexity, kernelization, edge modification problems

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2016.3

[*] The research of I. Bliznets is supported by the Government of the Russian Federation (grant 14.Z50.31.0030), by the Grant of the President of the Russian Federation (MK-6550.2015.1) as well as by Warsaw Center of Mathematics and Computer Science.

[†] The work of M. Cygan and P. Kamosa is the part of a project TOTAL that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 677651).

[‡] The research of M. Pilipczuk is supported by Polish National Science Centre grant UMO-2013/11/D/ST6/03073. M. Pilipczuk is also supported by the Foundation for Polish Science (FNP) via the START stipend programme.

1 Introduction

We consider the following general setting of *graph modification problems*: given a graph G, one would like to modify G as little as possible in order to make it satisfy some fixed property of global nature. Motivated by applications in de-noising data derived from imprecise experimental measurements, graph modification problems occupy a prominent role in the field of parameterized complexity and kernelization. This is because the allowed number of modifications usually can be assumed to be small compared to the total instance size, which exactly fits the motivation of considering it as the parameter of the instance.

Moving to the formal setting, consider some hereditary class of graphs Π, that is, a class closed under taking induced subgraphs. For such a class Π, we can define several problems depending on the set of allowed modifications. In each case the input consists of a graph G and integer k, and the question is whether one can apply at most k modification to G so that it falls into class Π. In this paper we will consider *deletion* and *completion* problems, where we are allowed only to delete edges, respectively only to add edges. However, other studied variants include *vertex deletion* problems (the allowed modification is removal of a vertex) and *editing* problems (both edge deletions and completions are allowed). Moreover, we restrict ourselves to classes Π characterized by one forbidden induced subgraph H. In other words, Π is the class of H-free graphs, that is, graphs that do not contain H as an induced subgraph (H is assumed to be constant).

The study of the parameterized complexity of H-free Edge Deletion and H-free Edge Completion focused on two aspects: designing fixed-parameter algorithms and kernelization procedures. The classic observation of Cai [3] shows that H-free Edge Deletion (Completion) can be both solved in time $c^k \cdot n^{\mathcal{O}(1)}$ for some constant c depending only on H, using a straightforward branching strategy. However, for several completion problems related to chordal graphs and their subclasses, like (proper) interval graphs or trivially perfect graphs, one can design *subexponential parameterized algorithms*, typically with the running time of $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$. The study of this surprising subexponential phenomenon, and of its limits, has recently been the subject of intensive studies; we refer to the introductory section of [2] for more details. However, for the vast majority of graphs H, the running time of the form $c^k \cdot n^{\mathcal{O}(1)}$ is essentially the best one can hope for H-free Edge Deletion (Completion). Indeed, Aravind et al. [1] proved that, whenever H has at least two edges, then H-free Edge Deletion is NP-hard and has no $2^{o(k)} \cdot n^{\mathcal{O}(1)}$-time algorithm unless the Exponential Time Hypothesis fails, and the same result holds for H-free Edge Completion whenever H has at least two non-edges. The remaining cases are easily seen to be polynomial-time solvable, so this establishes a full dichotomy.

Another interesting aspect of graph modification problems is their kernelization complexity. Recall that a *polynomial kernel* for a parameterized problem is a polynomial-time algorithm that, given an instance of the problem with parameter k, reduces it to another instance of the same problem that has size bounded polynomially in k. While every H-Free Vertex Deletion problem admits a simple polynomial kernel by a reduction to the d-Hitting Set problem (for $d = |V(H)|$), the situation for edge deletion and edge completion problems is much more complex. This is because the removal/addition of some edge may create new induced copies of H that were originally not present, and hence the obstacles can "propagate" in the graph. In fact, a line of work [4, 5, 9, 12] showed that, unless NP $\subseteq$ coNP/poly, polynomial kernels for the H-free Edge Deletion (Completion) problems exist only for very simple graphs H, for which the class of H-free graphs exhibits some structural property. This line culminated in the work of Cai and Cai [4, 5], who attempted to obtain a complete

dichotomy. While this goal was not fully achieved and there are some cases missing, the obtained complexity picture explains the general situation very well. For example, Cai and Cai [4, 5] showed that polynomial kernels do not exist (under NP $\not\subseteq$ coNP/poly) for the H-free Edge Deletion (Completion) problems whenever H is 3-connected and has at least 2 non-edges. Nontrivial positive cases include e.g. H being a path on 4 vertices [9] (that is, Cograph Edge Deletion (Completion)), and H being a K_4 minus one edge [5] (that is, Diamond-free Edge Deletion). One of the most prominent open cases left is the kernelization complexity of Claw-Free Edge Deletion [4, 6].

Our motivation and results

The starting point of our work is the realization that the propagational character of H-free Edge Deletion (Completion), which is the basic explanation of its apparent kernelization hardness, also makes the greedy approach to approximation incorrect. One cannot greedily remove all the edges of any copy of H in the graph, because removing an edge does not necessarily always help: it may create new copies of H in the instance. Hence, the approximation complexity of H-free Edge Deletion (Completion) is actually also highly unclear. On the other hand, the links between approximation and kernelization are well-known in parameterized complexity: it is often the case that a polynomial kernel for a problem can be turned into a poly(OPT)-approximation algorithm (i.e. an algorithm that returns a solution of cost bounded by some polynomial function of the optimum), by just taking greedily the kernel and reverting the reduction rules. While this intuitive link is far from being formal, and actually there are examples of problems behaving differently [8], it is definitely the case that the combinatorial insight given by kernelization algorithms may be very useful in the approximation setting.

Therefore, we propose to study the approximability of H-free Edge Deletion (Completion) as well, alongside with the best possible running times of fixed-parameter algorithms and the existence of polynomial kernels. This work is the first step in this direction.

We prove that the H-free Edge Deletion (Completion) problems are very hard to approximate for a vast majority of graphs H, which mirrors the kernelization hardness results of Cai and Cai [4, 5]. The following theorem explains our main result formally.

▶ **Theorem 1.** *Let H be a 3-connected graph with at least two non-edges. Then, unless* P = NP*, neither H-free Edge Deletion nor H-free Edge Completion admits a* poly(OPT)*-approximation algorithm running in polynomial time. Moreover, unless the Exponential Time Hypothesis fails, neither of these problems admits even a* poly(OPT)*-approximation algorithm running in time $2^{o(\mathsf{OPT})} \cdot n^{\mathcal{O}(1)}$.*

Theorem 1 makes two structural assumptions about graph H: that it is 3-connected, and has at least two non-edges. The first one is a crucial technical ingredient in the reductions, because it enables us to argue that for any vertex cut of size 2, every copy of H in the graph is completely contained on one side of the cut. Relaxing this assumption is a major issue addressed by Cai and Cai [4, 5] in their work. In an attempt to lift this assumption in our setting as well, we try to resolve the case of H being a path or a cycle first; this reflects the development of the story of kernelization hardness for the considered problems [5, 4, 9, 12]. The following theorem summarizes our results in this direction.

▶ **Theorem 2.** *Let H be a cycle on at least 4 vertices or a path on at least 5 vertices. Then, unless* P = NP*, neither H-free Edge Deletion nor H-free Edge Completion admits a* poly(OPT)*-approximation algorithm running in polynomial time. Moreover, unless the*

Exponential Time Hypothesis fails, neither of these problems admits even a poly(OPT)*-approximation algorithm running in time* $2^{o(\mathsf{OPT})} \cdot n^{\mathcal{O}(1)}$.

Together with some easy cases and known positive results [14], this gives an almost complete dichotomy for paths and cycles. The only missing case is Cograph Edge Deletion (for $H = P_4$), for which we expect a positive answer due to the existence of a polynomial kernel [9]. However, our preliminary attempt at lifting the kernel of Guillemot et al. [9] showed that the approach does not directly work for approximation, and new insight seems to be necessary.

Finally, somewhat surprisingly we show that the assumption that H has at least two non-edges appears to be important. Suppose $H = K_n \setminus e$ is a complete graph on $n \geq 5$ vertices with one edge removed. While H-free Edge Completion is trivially polynomial-time solvable, due to each obstacle having only one way to be destroyed, the complexity of H-free Edge Deletion turns out to be much more interesting. Namely, we show that it is tightly connected to the complexity of Min Horn Deletion, which apparently is one of the remaining open cases in the classification of the approximation complexity of CSP problems of Khanna et al. [11]. Hence, the following theorem shows that the case of H being a complete graph without an edge may be an interesting outlier in the whole complexity picture.

▶ **Theorem 3.** *For any* $n \geq 5$*, the* $K_n \setminus e$*-free Edge Deletion problem is* Min Horn Deletion*-complete with respect to A-reductions.*

The exact meaning of Min Horn Deletion-completeness, A-reductions and other definitions related to the hardness of approximation for CSP problems are explained in Section 4. A direct consequence of Theorem 3 and the work of Khanna et al. [11] is that $K_n \setminus e$-free Edge Deletion does not admit a $2^{\mathcal{O}(\log^{1-\epsilon}|E|)}$-approximation algorithm working in polynomial time, for any $\epsilon > 0$, where $|E|$ is the number of edges in a given graph. Moreover, Theorem 3 implies that $K_n \setminus e$-free Edge Deletion is poly-APX-hard if and only if each Min Horn Deletion-complete problem is poly-APX-hard, the latter being an intriguing open problem left by Khanna et al. [11] in their study of approximability of CSPs.

While there is no direct connection between the existence of a poly(OPT) approximation and poly-APX-hardness, we still believe that our reduction corroborates the hardness of resolving approximation question of $K_n \setminus e$-free Edge Deletion in terms of optimum value. Intuitively, showing poly-APX-hardness should be easier than refuting poly(OPT) approximation. Below we state formally what our reduction actually implies.

▶ **Corollary 4.** *Let* $n \geq 5$*. Then it is* NP*-hard to approximate the* $K_n \setminus e$*-free Edge Deletion problem within factor* $2^{\mathcal{O}(\log^{1-\epsilon}|E|)}$ *for any* $\epsilon > 0$*, where* $|E|$ *is the number of edges in a given graph.*

▶ **Corollary 5.** *Let* $n \geq 5$*. Then the* $K_n \setminus e$*-free Edge Deletion problem admits an* n^{δ}*-approximation for all* $\delta > 0$*, if and only if each* Min Horn Deletion*-complete problem admits an* n^{δ_1}*-approximation for all* $\delta_1 > 0$*.*

Our techniques

To prove our main result, Theorem 1, we employ the following strategy. We first consider the *sandwich problem* defined as follows: in Sandwich H-Free Edge Deletion we are given a graph G together with a subset D of *undeletable edges*, and the question is whether there exists a subset $F \subseteq E(G) \setminus D$ of deletable edges for which $G - F$ is H-free. Note

that the sandwich problem differs from the standard H-free Edge Deletion problem in two aspects: first, some edges are forbidden to be deleted, and, second, it is a decision problem about the existence of *any* solution—we do not impose any constraint on its size. For completion, the sandwich problem is defined similarly: we have *non-fillable non-edges*, i.e., non-edges that are forbidden to be added in the solution.

The crux of the approach is to prove that Sandwich H-Free Edge Deletion is actually NP-hard under the given assumptions on H. The next step is to reduce from the sandwich problem to the standard optimization variant. This is done by adding gadgets that emulate undeletable edges by introducing a large approximation gap, as follows. For each undeletable edge e, attach a large number of copies of H to e, so that each copy becomes an induced H-subgraph if e gets deleted. Then any solution that deletes the undeletable edge e must have a very large cost, due to all the disjoint copies of H that appear after the removal of e. The assumption that H is 3-connected is very useful for showing that the constructions do not introduce any additional, unwanted copies of H in the graph.

The approach for completion problems is similar. To prove Theorem 2 that concerns paths and cycles, we give problem-specific constructions using the same approach. Some of them are based on previous ETH-hardness proofs for the problems, given by Drange et al. [7].

As far as Theorem 3 is concerned, we employ a similar reduction strategy, but instead of starting from 3SAT, we start from a carefully selected MinOnes($\mathcal{F}$) problem: the problem of optimizing the number of ones in a satisfying assignment to a boolean formula that uses only constraints from some fixed family $\mathcal{F}$. In particular, the constraint family $\mathcal{F}$ needs to be rich enough to be Min Horn Deletion-hard, while at the same time it needs to restrictive enough so that it can be expressed in the language of $K_n \setminus e$-free Edge Deletion.

Our constructions are inspired by the rich toolbox of hardness proofs for kernelization and fixed-parameter algorithms for edge modification problems [1, 4, 5, 7, 12, 9]. In particular, the idea of considering sandwich problems can be traced back to the work of Cai and Cai [4, 5], who use the term *quarantine* for the optimization variants of sandwich edge modification problems, with undeletable edges and non-fillable non-edges. Quarantined problems serve a technical, auxiliary role in the work of Cai and Cai [4, 5]: one first proves hardness of the quarantined problem, and then lifts the quarantine by attaching gadgets, similarly as we do.

However, we would like to point out the new challenges that appear in the approximation setting. Most importantly, the vast majority of previous reductions heavily use budget constraints (i.e. the fact that the solution is stipulated to be of size at most k) to argue the correctness; this includes the general results of Cai and Cai [4, 5]. In our setting, we cannot use arguments about the tightness of the budget, because we need to introduce a large approximation gap at the end of the construction. The usage of the sandwich problems *without* any budget constraints is precisely the way we overcome this difficulty. Thus, most of the old reductions do not work directly in our setting, but of course some technical constructions and ideas can be salvaged.

Outline

In Section 2 we introduce terminology and recall the most important facts from the previous works. Section 3 is devoted to the proof of our main result, Theorem 1. However, as the proof for H-free Edge Completion is similar to the proof for H-free Edge Deletion, the details are postponed to the full version of the paper (arXiv:1606.02688). In Section 4 we discuss the proof of Theorem 3. The proof of Theorem 2 is also omitted, and included in the full version of the paper. Concluding remarks and prospects on future work are in Section 5.

2 Preliminaries

2.1 Basic graph definitions

We use standard graph notation. For a graph G by $V(G)$ and $E(G)$ we denote the set of vertices and edges of G, respectively. Throughout the paper we consider simple graphs only, i.e., there are no self-loops nor parallel edges. We use K_n to denote the complete graph on n vertices. By P_ℓ (C_ℓ) we denote the path (cycle) with exactly ℓ vertices. By $\overline{G}$ we denote the *complement* of G, i.e., a graph on the same vertex set, where two distinct vertices are adjacent if and only if they were not adjacent in G. We say that a graph G is H-free, if G does not contain H as an induced subgraph.

We define a graph G to be 3-*vertex-connected* if G has at least 3 vertices, and removing any set of at most two vertices causes G to stay connected. For brevity, we call such graphs 3-*connected*.

2.2 Problems and approximation algorithms

In the decision version the H-free Edge Deletion (Completion) problem, for a given graph G and an integer k, one is to decide whether it is possible to delete (add) at most k edges from (to) G to make it H-free. In particular, we consider the $\overline{P}_5$-Free Deletion (Completion) problem, and call it House-Free Deletion (Completion). However, in the optimization variant of H-free Edge Deletion (Completion) the value of k is not given and the goal is to find a minimum size solution. It will be clear from the context whether we refer to a decision or optimization variant.

In the Sandwich H-Free Edge Deletion (Completion) problem we are given a graph G together with a subset D of *undeletable edges* (*non-fillable non-edges*). The question is whether there exists a subset $F \subseteq E(G) \setminus D$ ($F \subseteq \overline{E(G)} \setminus D$) of deletable (fillable) edges for which $G - F$ ($G + F$) is H-free. Note that it is a decision problem, where we ask about existence of any solution, i.e., we do not impose any constraint on the solution size.

Let f be a fixed non-decreasing function on positive integers. An $f(OPT)$-*factor approximation algorithm* for a minimization problem X is an algorithm that finds a solution of size at most $f(OPT) \cdot OPT$, where OPT is the size of an optimal solution for a given instance of X.

2.3 Satisfiability and Exponential Time Hypothesis

We employ the standard notation related to satisfiability problems. A 3CNF formula is a conjunction of clauses, where a clause is a disjunction of at most three literals. The 3SAT problem asks, for a given formula φ, whether there is a satisfying assignment to φ.

The Exponential Time Hypothesis (ETH), introduced by Impagliazzo, Paturi and Zane [10] is now an established tool used for proving conditional lower bounds in the parameterized complexity area (see [13] for a survey on ETH-based lower bounds).

▶ **Hypothesis 6** (Exponential Time Hypothesis (ETH) [10]). *There is no $2^{o(n)}$ time algorithm for* 3SAT*, where n is the number of variables of the input formula.*

The main consequence of the Sparsification Lemma of [10] is the following theorem: there is no subexponential algorithm for 3SAT even in terms of the number of clauses of the formula.

▶ **Theorem 7** ([10]). *Unless ETH fails, there is no $2^{o(n+m)}$ time algorithm for* 3SAT*, where n, m are the number of variables, and clauses, respectively.*

3 Hardness for 3-connected H

In this section we present the proof of Theorem 1 for H-free Edge Deletion.

3.1 Deletion problems

We start with proving hardness of the sandwich problem.

▶ **Lemma 8.** *Let H be a 3-connected graph with at least 2 non-edges. There is a polynomial-time reduction, which given an instance of* 3SAT *with n variables and m clauses, creates an equivalent instance of* Sandwich H-free Edge Deletion *with $\mathcal{O}(n+m)$ edges. Consequently,* Sandwich H-free Edge Deletion *is* NP*-hard for such graphs H.*

Proof. Let φ be the given formula in 3CNF, and let `vars` and `cls` be the sets of variables and clauses of φ. By standard modifications of the formula, we may assume that each clause contains exactly three literals of pairwise different variables. We construct an instance G of Sandwich H-free Edge Deletion as follows. The graph G is created from three types of gadgets: a clause gadget, a variable gadget, and a connector gadget. They are depicted in Figure 1, where presented edges are deletable, and all others are undeletable.

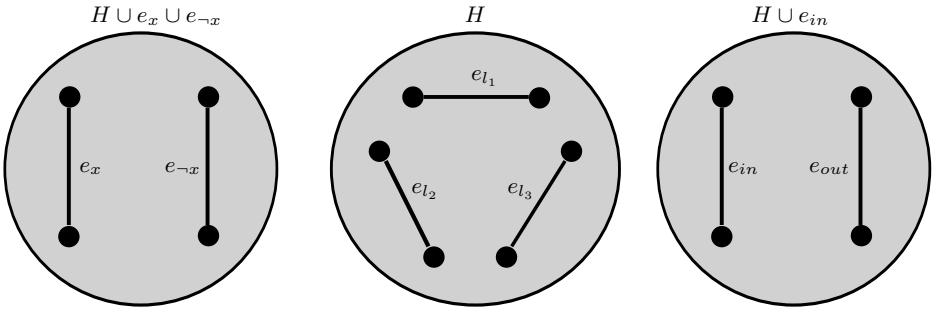
We first explain constructions of the gadgets, and then discuss connections between them. For each variable $x \in$ `vars`, we create a variable gadget G^x, which is the graph H with two added edges e_x and $e_{\neg x}$ in place of any two non-edges of H. In the graph H_x, all edges are marked as undeletable except e_x and $e_{\neg x}$. Intuitively, deletion of the edge e_x or $e_{\neg x}$ mimics an assignment of the corresponding literal to true. The variable gadget forbids simultaneous assignments of both literals to true. If we delete both edges e_x and $e_{\neg x}$, we get an induced subgraph H in which we cannot delete any edge.

Each clause $c = \ell_1 \vee \ell_2 \vee \ell_3 \in$ `cls` has the corresponding clause gadget H^c, which is a copy of the graph H. As H^c is 3-connected, it has at least 3 edges. We pick arbitrarily three edges of H^c and label them by $e_{\ell_1}, e_{\ell_2}, e_{\ell_3}$. We mark all others edges as undeletable. In order to make the clause gadget H-free, we have to delete at least one edge from $e_{\ell_1}, e_{\ell_2}, e_{\ell_3}$ (note that some of the three distinguished edges might potentially share an endpoint). Intuitively, deletion of the edge labeled by e_ℓ corresponds to assigning value true to literal ℓ.

The third type of gadgets is the connector gadget. The connector gadget C is a copy of the graph H, with one added edge in place of any non-edge of H. We label this edge as e_{in}. In C, there also exists another edge that does not share any of its endpoints with e_{in}. To see this, for the sake of contradiction suppose that every edge of C is incident to one of the endpoints of e_{in}. If C has at least two vertices other than these endpoints, then the endpoints of e_{in} form a vertex cut of size 2 separating them, a contradiction with 3-connectedness of H. Otherwise C has only one vertex other than the endpoints of e_{in}, so H has at most 3 vertices; again, a contradiction with the 3-connectedness of H, as we assume H to have at least 2 non-edges. We select any edge in H that does not share endpoints with e_{in}, and we label it as e_{out}. Edges e_{in} and e_{out} are made deletable, and all other edges of C are made undeletable. Note that deletion of the edge e_{in} creates an induced subgraph H, and then we have to delete e_{out} in order to destroy this subgraph.

Knowing the structure of all gadgets, we can proceed with the main construction of our reduction.

Given a formula φ, for each clause $c \in$ `cls` and variable $x \in$ `vars`, we create the clause gadget H^c and the variable gadget G^x, respectively. Moreover, for each literal ℓ belonging to the clause $c \in$ `cls`, we create a chain $C^{\ell,c}_1, C^{\ell,c}_2, \ldots, C^{\ell,c}_{p+2}$ consisting of $p+2$ copies of the connector gadget, where $p = |V(H)|$. This chain is constructed in the following way: the



(a) Variable gadget G^x **(b)** Clause gadget H^c **(c)** Connector gadget C

Figure 1 Gadgets for Sandwich H-free Edge Deletion.

edge e_{out} of $C_i^{\ell,c}$ is identified with the edge e_{in} of $C_{i+1}^{\ell,c}$, for $i = 1, \ldots, p+1$. We also identify the edge e_{out} in the subgraph $C_{p+2}^{\ell,c}$ with the edge e_ℓ in the variable gadget of the variable of ℓ. Moreover, the edge e_{in} in the subgraph $C_1^{\ell,c}$ is identified with the edge e_ℓ from the clause gadget H^c. We use those chains to not allow the copy of H to be shared by any two gadgets, and we will prove it in the claim below.

Clearly, the constructed graph G has at most $\mathcal{O}(n+m)$ edges.

▶ **Claim 9.** *If G is a YES instance, then φ is satisfiable.*

Proof. Take any solution to the instance G. Note that in each clause gadget we must delete at least one edge. We set the literals corresponding to the deleted edges to true, thus satisfying every clause. We prove now that for each variable x we have not set both literals x and $\neg x$ to true, so that we can find a true/false assignment to the variables that sets the literals accordingly. Deletion of an edge in the clause gadget propagates deletions up to the variable gadget via the chain of connector gadgets. This happens because the deletion of e_{in} in $C_1^{\ell,c}$ forces us to delete the e_{out} in $C_1^{\ell,c}$, which is e_{in} in $C_2^{\ell,c}$, so we are forced to delete e_{out} in $C_2^{\ell,c}$, and so on. Following the chain of connector gadgets, it is easy to see that the edge e_ℓ must be deleted in the corresponding variable gadget. As the solution to the instance G cannot delete both edges e_x and $e_{\neg x}$ in any variable gadget at the same time, we obtain that there are no variables with both of its literals set to true. ◀

▶ **Claim 10.** *If φ is satisfiable, then G is a YES instance.*

Proof. Consider a true/false assignment that satisfies the formula φ and delete all edges in all clause gadgets that correspond to literals taking value true. Propagate deletions to all the connector and variable gadgets, as in the proof of Claim 9. It remains to prove that the obtained graph is indeed an H-free graph. By counting the number of edges in each gadgets, it follows that after the deletions, all gadgets become not isomorphic to H: in every variable gadget, we deleted exactly one edge, in every clause gadget, we deleted at least one edge, and in each connector gadget we deleted zero or two edges. So if the obtained graph contains an induced subgraph of H, then H is distributed across several gadgets. However, this is also not possible for the following reason.

For the sake of contradiction, suppose after the deletions there is an induced copy H' of the graph H. Since H' is connected and is distributed among more than one gadget, there have to be two different gadgets G_1, G_2 that share a vertex, for which H' contains both some vertex $u \in V(G_1) \setminus V(G_2)$, and some vertex $v \in V(G_2) \setminus V(G_1)$. Since H' is 3-connected, there are 3 internally vertex-disjoint paths in H' that lead from u to v. But

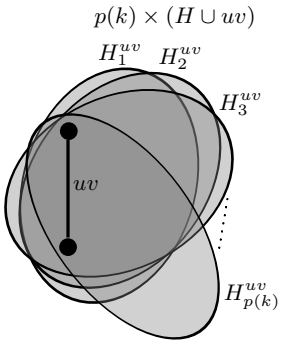


Figure 2 Gadgets H_i^{uv} for H-free Edge Deletion.

every two gadgets share at most two common vertices, so at least one of these paths, say P, avoids $V(G_1) \cap V(G_2)$. Since the path P avoids $V(G_1) \cap V(G_2)$, from the construction of G it easily follows that such path P contains at least one vertex of some variable gadget and at least one vertex of some clause gadget. However, the distance between e_{in} and e_{out} in each connector gadget is at least 1, so the distance between any variable gadget and any clause gadget is at least $|V(H)|$. But the path P is entirely contained in H', thus its length is at most $|V(H)| - 1$, a contradiction. ◀

Claims 9 and 10 ensure that the output instance G is equivalent to the input instance φ of 3SAT, so we are done. ◀

Now, we show how to reduce Sandwich H-free Edge Deletion to the optimization variant of H-free Edge Deletion. Note that we only require H to have at least one non-edge; this is because we will reuse this lemma in the next section.

▶ **Lemma 11.** *Let H be a 3-connected graph with at least one non-edge, and $p(\cdot)$ be a polynomial with $p(\ell) \geq \ell$, for all positive ℓ. Then there is a polynomial-time reduction which, given an instance G of* Sandwich H-free Edge Deletion*, creates an instance (G', k) of H-free Edge Deletion, such that:*

- *k is the number of deletable edges in G;*
- *G' has $\mathcal{O}(p(k) \cdot |E(G)| \cdot |E(H)|)$ edges;*
- *If G is a YES instance, then (G', k) is a YES instance;*
- *If G is a NO instance, then $(G', p(k))$ is a NO instance.*

Proof. We create G' in the following way. For each undeletable edge uv, we add $p(k)$ copies H_i^{uv} of the graph H, $i = 1, \ldots, p(k)$. In each copy, we choose any non-edge $u_i v_i$ and identify the vertex u_i with u, and v_i with v. The construction is presented in Figure 2.

Note that if we delete the edge uv in G', we also must delete at least one edge in every H_i^{uv}. Hence, at least $p(k)+1$ edges will be deleted in such a situation. With this observation in mind, we proceed to the proof of the correctness.

▶ **Claim 12.** *If G is a YES instance, then (G', k) is a YES instance.*

Proof. Let F be a subset deletable edges, such that $G - F$ is H-free. Obviously $|F| \leq k$, because there are k deletable edges in G in total. We will prove that $G' - F$ is also H-free, which implies that (G', k) is a YES instance.

Let us assume otherwise, that there is an induced copy H' of H in G'. Since $G - F$ is H-free, we have that H' has to contain at least one vertex of $V(G') \setminus V(G)$. Say that H'

contains some vertex x of $V(H_i^{uv}) \setminus V(G)$, for some undeletable edge uv and some index i. The edge uv is undeletable in G, so it is not included in F. Consequently, the subgraph of G' induced by $V(H_i^{uv})$ contains one more edge than H, so it is not isomorphic to H. We conclude that H' must contain some vertex y that lies outside of $V(H_i^{uv})$. Since H is 3-connected, there are 3 internally vertex-disjoint paths between x and y in H. However, in G, the set $V(H_i^{uv}) \cap V(G) = \{u, v\}$ is a vertex cut of size 2 that separates x and y. This is a contradiction, so $G' - F$ is indeed H-free. ◀

▶ **Claim 13.** *If G is a NO instance, then $(G', p(k))$ is a NO instance.*

Proof. For the sake of contradiction, suppose there is a set F' of at most $p(k)$ edges of G', such that $G' - F'$ is H-free. Note that, F' has to contain at least one undeletable edge uv, as otherwise $F' \cap E(G)$ would be a solution to G. But then F' has to contain at least $p(k)$ more edges inside gadgets H_i^{uv}, for $i = 1, 2, \ldots, p(k)$, which is a contradiction with $|F'| \leq p(k)$. ◀

Claims 12 and 13 ensure the correctness of the reduction, and hence we are done. ◀

By composing the reductions of Lemmas 8 and 11, we can deduce the part of Theorem 1 concerning deletion problems. Indeed, suppose H-free Edge Deletion admitted a polynomial-time $q(\mathsf{OPT})$-factor approximation algorithm, for some polynomial q. Take any instance of 3SAT, and apply first the reduction of Lemma 8, and then the reduction of Lemma 11 for polynomial $p(\ell) = q(\ell) \cdot \ell + 1$. Finally, observe that the application of the hypothetical approximation algorithm for H-free Edge Deletion to the resulting instance would resolve whether the optimum value is at most k or at least $p(k)$, which, by Lemma 11, resolves whether the input instance of 3SAT is satisfiable. The subexponential hardness of approximation under ETH follows from the same reasoning and the observation that the value of k in the output instance is bounded linearly in the size of the input formula.

4 Connections with Min Horn Deletion

In this section we prove Theorem 3. First, we need to introduce some definitions and notation regarding Min Horn Deletion hardness and completeness.

Khanna et al. [11] attempted to establish a full classification of approximability of boolean constraint satisfaction problems. In particular, many problems have been classified as APX-complete or poly-APX-complete. Even though some cases remained unresolved, Khanna et al. [11] grouped them into classes, such that all problems from the same class are equivalent (with respect to appropriately defined reductions) to a particular representative problem. One such representative problem is Min Horn Deletion, defined as follows: Given is a boolean formula φ in CNF that contains only unary clauses, and clauses with three literals out of which exactly one is negative. The problem asks for minimizing the number of ones in a satisfying assignment for φ.

We are not going to operate on instances of Min Horn Deletion directly, so the definition above is given only in order to complete the picture for the reader. Instead, we will rely on the approximation hardness results exhibited by Khanna et al. [11], which relate the approximability of various boolean CSPs to Min Horn Deletion. In particular, it is known that Min Horn Deletion does not admit a $2^{\mathcal{O}(\log^{1-\epsilon} n_{\text{vars}})}$ approximation algorithm, unless P = NP, where n_{vars} is the number of variables in the instance. On the other hand, it is an open problem whether any Min Horn Deletion-complete problem (under A-reductions, defined below) is actually poly-APX-complete.

▶ **Definition 14** (A-reducibility, Definition 2.6 of [11]). A combinatorial optimization problem is said to be an NPO problem if instances and solutions can be recognized in polynomial time, solutions are polynomially-bounded in the input size, and the objective function can be computed in polynomial time from an instance and a solution.

An NPO problem P is said to be *A-reducible* to an NPO problem Q, denoted $P \leq_A Q$, if there are two polynomial-time computable functions F and G and a constant α, such that:

1. For any instance $\mathcal{I}$ of P, $F(\mathcal{I})$ is an instance of Q.
2. For any instance $\mathcal{I}$ of P and any feasible solution $\mathcal{S}'$ for $F(\mathcal{I})$, $G(\mathcal{I}, \mathcal{S}')$ is a feasible solution for $\mathcal{I}$.
3. For any instance $\mathcal{I}$ of P and any $r \geq 1$, if $\mathcal{S}'$ is an r-approximate solution for $F(\mathcal{I})$, then $G(\mathcal{I}, \mathcal{S}')$ is an (αr)-approximate solution for $\mathcal{I}$.

Intuitively, A-reductions preserve approximability problems up to a constant factor (or higher). As a source of Min Horn Deletion-hardness we will use the MinOnes($\mathcal{F}$) problem, defined below, for a particular choice of the family of constraints $\mathcal{F}$.

In the MinOnes($\mathcal{F}$) problem, we are given a ground set of boolean variables X together with a set of boolean constraints. Each constraint f is taken from a specified family $\mathcal{F}$, and f is applied to some tuple of variables from X. The goal of the problem is to find an assignment satisfying all the constraints, while minimizing the number of variables set to one. Note that the family $\mathcal{F}$ is considered a part of the problem definition, not part of the input. In order to use known results for the MinOnes($\mathcal{F}$) problem we need to define some properties of boolean constraints.

- A boolean constraint f is called *weakly positive* if it can be expressed using a CNF formula that has at most one negated variable in each clause.
- A boolean constraint f is 0-*valid* if the all-zeroes assignment satisfies it.
- A boolean constraint f is IHS-B^+ if it can be expressed using a CNF formula in which the clauses are all of one of the following types: $x_1 \vee \cdots \vee x_k$ for some positive integer $k \leq B$, or $\neg x_1 \vee x_2$, or $\neg x_1$. IHS-B^- constraints are defined analogously, with every literal being replaced by its complement.

The definition can be naturally extended to families of constraints, e.g., a family of constraints is weakly positive if all its constraints are weakly positive. We say that a family of constraints is IHS-B if it is either IHS-B^+ or IHS-B^- (or both). The following result was proved by Khanna et al. [11].

▶ **Theorem 15** (Lemmas 8.7 and 8.14 from [11]). *If a family of constraints $\mathcal{F}$ is weakly positive, but it is neither* 0*-valid nor IHS-B for any constant B, then the problem* MinOnes($\mathcal{F}$) *is* Min Horn Deletion*-complete under A-reductions; that is, there is an A-reduction from* Min Horn Deletion *to* MinOnes($\mathcal{F}$) *and an A-reduction from* MinOnes($\mathcal{F}$) *to* Min Horn Deletion*. Consequently, it is NP-hard to approximate* MinOnes($\mathcal{F}$) *within factor* $2^{\mathcal{O}(\log^{1-\epsilon} n_{\text{vars}})}$ *for any $\epsilon > 0$, where n_{vars} is the number of variables in the given instance.*

Our strategy for the proof of Theorem 3 is as follows. In Section 4.1 we show a reduction from MinOnes($\mathcal{F}$) to a properly defined quarantined version of $K_n \setminus e$-free Edge Deletion. Next, in Section 4.2 we show a reduction which removes the quarantine. Finally, in Section 4.3 we conclude the proof of Theorem 3 and show the completeness with respect to A-reductions.

Note that having Theorem 3, we can immediately infer Corollaries 4,5 using Theorem 15 and the definition of an A-reduction.

4.1 From MinOnes (F) to Quarantined H-free Edge Deletion

In the Quarantined H-free Edge Deletion problem we are given a graph G, some edges of which are marked as undeletable. Quarantined H-free Edge Deletion is an optimization problem, where the goal is to obtain an H-free graph by removing the minimum number of deletable edges.

Next, we define the family of constraints that will be used in the MinOnes($\mathcal{F}$) problem.

▶ **Definition 16.** We define the following constraints:
- a constraint $f_1(x_1, x_2, x_3)$, which is equal to zero if and only if exactly one of the variables x_1, x_2, x_3 is set to 1;
- a constraint $f_2(x) = x$.

The family of constraints $\mathcal{F}'$ is defined as $\mathcal{F}' = \{f_1, f_2\}$.

A direct check, presented below, verifies that $\mathcal{F}'$ has the properties needed to claim, using Theorem 15, that MinOnes($\mathcal{F}'$) is Min Horn Deletion-hard.

▶ **Lemma 17.** *The family of constraints $\mathcal{F}' = \{f_1, f_2\}$ is weakly positive, and at the same time it is neither 0-valid, nor IHS-B for any B.*

Proof. Note that f_1 is weakly positive since $f_1(x_1, x_2, x_3) = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$. Constraint f_2 is clearly weakly positive by definition. As f_2 is not 0-valid, we have that $\mathcal{F}'$ is not 0-valid either.

We prove now that f_1 is not IHS-B for any B. First, observe that any CNF formula expressing f_1 cannot contain a clause with only positive literals, as such a clause would not be satisfied by the assignment $x_1 = x_2 = x_3 = 0$, which in turn satisfies f_1. Similarly, no clause can have only negative literals. Due to the definition of IHS-B, the only remaining case is a 2-clause with one positive and one negative literal. Without loss of generality, consider a clause $x_1 \vee \neg x_2$. Observe, that it is not satisfied by the assignment $x_1 = 0$, $x_2 = x_3 = 1$, which however satisfies f_1. Therefore f_1, and consequently $\mathcal{F}'$, is not IHS-B for any B. ◀

Consequently, Theorem 15 and Lemma 17 together imply that MinOnes($\mathcal{F}'$) is Min Horn Deletion-hard under A-reductions. We now give our main reduction, from MinOnes($\mathcal{F}'$) to Quarantined $K_n \setminus e$-free Edge Deletion.

▶ **Lemma 18.** *Let $n \geq 5$. There is a polynomial-time computable transformation T which, given an instance $\mathcal{I}$ of the* MinOnes($\mathcal{F}'$) *problem, outputs an instance $T(\mathcal{I})$ of the* Quarantined $K_n \setminus e$-free Edge Deletion *problem, such that:*
- *if $\mathcal{I}$ admits a satisfying assignment with k ones, then there is a solution of cost $\Delta \cdot k$ for the instance $T(\mathcal{I})$,*
- *if $T(\mathcal{I})$ admits a solution of cost k', then there is a satisfying assignment with $\lfloor k'/\Delta \rfloor$ ones for the instance $\mathcal{I}$,*

where $\Delta = 9n_{\text{vars}}^2 + 2$ and n_{vars} is the number of variables in $\mathcal{I}$.

Proof. First, we show how to transform an instance $\mathcal{I}$ (with a formula φ) of MinOnes($\mathcal{F}'$) into an instance $T(\mathcal{I})$ (with a graph G) of Quarantined $K_n \setminus e$-free Edge Deletion. Given an instance $\mathcal{I}$, for any constraint $f_1(x, y, z)$ we create a separate clique K_n, which will be called the *constraint clique*. We arbitrarily choose three edges in the clique and label them x, y, z. Mark all edges as undeletable except edges labelled by x, y, z. Moreover, for each variable x we additionally create a clique K_n (called further the *variable clique*), and mark all edges in the clique as undeletable except two edges, which we label by x_{in}, x_{out}. The

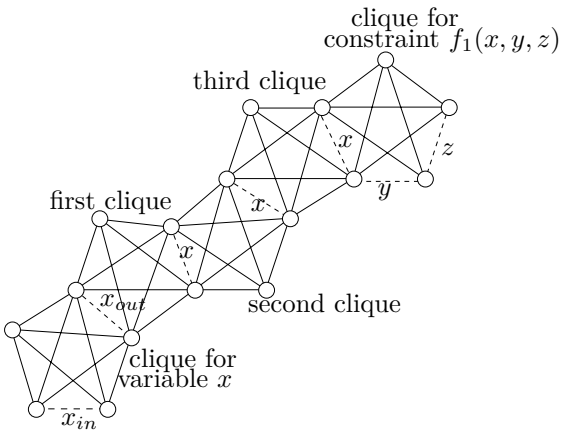


Figure 3 Gadgets for Quarantined $K_n \setminus e$-Free Edge Deletion. Deletable edges are shown by dashed lines.

edges x_{in}, x_{out} are selected arbitrarily, however we require that they do not share common endpoints.

Now we connect the variable cliques with the constraint cliques. For each variable x and a constraint f_1 of the instance $\mathcal{I}$ which contains x among its arguments, we add three cliques, as shown in Figure 3, such that the following properties are satisfied:

- The first added clique shares with the variable clique of x only the edge x_{out}.
- The second added clique shares one deletable edge with the first clique and a different deletable edge with the third clique. Label both these deletable edges by x.
- The third added clique shares with the clique corresponding to the constraint only the edge labelled (in the constraint clique) by x.

All the other edges of the introduced cliques, not mentioned above, are marked as undeletable. Note that each of the introduced cliques shares two edges with two different cliques. We may perform this construction so that these two edges never share endpoints (as depicted Figure 3), and hence we will assume this property.

Denote by $\delta(x)$ the number of occurrences of the variable x in all f_1-type constraints. Note that, by removing superfluous copies of the same constraint, we can assume that all f_1-type constraints are pairwise different, so in particular there is at most n^3_{vars} of them. As each variable can occur in one constraint at most three times, for any variable x we have $\delta(x) \leq 3n^2_{\text{vars}}$.

Next, for each variable x we add $3\cdot(3n^2_{\text{vars}}-\delta(x))$ or $3\cdot(3n^2_{\text{vars}}-\delta(x))+1$ cliques that share the deletable edge x_{in} from the variable clique of x, and are otherwise disjoint. Moreover, in each such clique we make one more edge deletable; we label it by x. We add $3\cdot(3n^2_{\text{vars}}-\delta(x))$ cliques if the formula does contain the clause $f_2(x) = x$, and $3\cdot(3n^2_{\text{vars}}-\delta(x))+1$ cliques otherwise.

Finally, if there is a clause $f_2(x) = x$ in the instance $\mathcal{I}$, then we delete the edge labelled by x_{in} in the corresponding variable clique.

Observe that in the constructed instance of Quarantined $K_n \setminus e$-free Edge Deletion, among all the $9n^2_{\text{vars}}+2$ edges labelled by x, x_{in}, x_{out}, where x is any variable, we have to delete either none, or all of them. This is because the deletion of any of them forces the deletion of all the others due to the appearance of induced copies of $K_n \setminus e$ in the graph. Moreover, if the edge x_{in} is not present due to the existence of constraint $f_2(x) = x$ in $\mathcal{I}$, then all of them have to be deleted.

▶ **Claim 19.** *If there is a satisfying assignment with k ones for the instance $\mathcal{I}$, then it is possible to delete $(9n_{\text{vars}}^2 + 2) \cdot k$ edges in $T(\mathcal{I})$ in order to make it a $K_n \setminus e$-free graph.*

Proof. It is enough to delete all edges labelled by x, x_{in}, x_{out} for all variables x that are set to 1 in the satisfying assignment; the number of such edges is exactly $(9n_{\text{vars}}^2 + 2) \cdot k$. Let us prove the statement. Suppose the obtained graph is not $K_n \setminus e$-free. Let H' be an induced subgraph isomorphic to $K_n \setminus e$. Note that for $n \geq 5$ the graph $K_n \setminus e$ is 3-connected. Moreover, even after deletion of two arbitrary vertices in $K_n \setminus e$, there are no two vertices at distance larger than two. Consequently, a direct check shows that the assumed H' subgraph must stay completely in one of the cliques corresponding to a constraint or to a variable, or in one of the cliques connecting a variable clique with a constraint clique. Obviously, H' cannot be contained in a variable clique or a connection clique, as in such cliques either all edges are present, or two edges are missing. This means that H' must stay in a constraint clique, so exactly one of the edges of this constraint clique is deleted. However, this is equivalent with the corresponding constraint being not satisfied under the considered assignment; this is a contradiction. ◀

▶ **Claim 20.** *If $T(\mathcal{I})$ admits a solution of cost k', then there is a satisfying assignment for the instance $\mathcal{I}$ with $\lfloor k'/(9n_{\text{vars}}^2 + 2) \rfloor$ ones.*

Proof. Take any solution for the output instance $T(\mathcal{I})$. As mentioned earlier, in any solution for $T(\mathcal{I})$, for any variable x either all edges labeled by x, x_{in}, x_{out} are deleted or none of them is deleted. The number of such edges for one variable x is equal to $9n_{\text{vars}}^2 + 2$. We set a variable to 1 if and only if the corresponding edges are deleted in the considered solution for $T(\mathcal{I})$. All clauses of the form $f_2(x)$ will be satisfied, since in the construction of $T(\mathcal{I})$ we delete x_{in} if the clause $f_2(x) = x$ is present in $\mathcal{I}$. All f_1-type constraints will be satisfied as well, as otherwise in the clique corresponding to an unsatisfied constraint only one edge would be deleted and, hence, the graph would not be $K_n \setminus e$-free. ◀

The correctness of the transformation follows from Claims 19 and 20; hence the proof of Lemma 18 is complete. ◀

4.2 Lifting the quarantine

In the following lemma we show how to reduce an instance of the quarantined problem to its regular version, using the same approach as in the proof of Lemma 11.

▶ **Lemma 21.** *Let $n \geq 5$. There is a polynomial-time reduction which, given an instance G of* Quarantined $K_n \setminus e$-free Edge Deletion *with m edges, outputs an instance G' of* $K_n \setminus e$-free Edge Deletion *such that:*

- *G' has $\mathcal{O}(m^3)$ vertices and edges.*
- *If there is a solution of size k for the instance G, then there is a solution of size k for the instance G'.*
- *If there is a solution of size $k \leq m^2$ for the instance G', then there is a solution of size k for the instance G.*

Proof. We apply the reduction described in the proof of Lemma 11 for $p(m) = m^2$ and $H = K_n \setminus e$. Now we verify that G' has the claimed properties. The bound on the size of G' follows directly from the size bound given by Lemma 11.

Suppose first that G has some solution of size k. In the proof of Lemma 11 we argued that the same solution also works for the instance G' (see the proof of Claim 12). Hence, G' also has a solution of size k.

Suppose now that G' has a solution F of some size $k \leq m^2$. In the proof of Claim 13 we argued that F does not delete any of the undeletable edges of G, because this would require deleting at least m^2 more edges in the attached gadgets. Hence, $F \cap E(G)$ is a set of size at most k, whose deletion turns G into an H-free graph, due to being an induced subgraph of $G' - F$. Hence, G has some solution of size at most k. ◀

The composition of the reductions of Lemmas 18 and 21 gives an A-reduction (for $\alpha = 1$) from a Min Horn Deletion-hard problem MinOnes($\mathcal{F}$), yielding the hardness part of Theorem 3. Indeed, given an instance $\mathcal{I}$ of MinOnes($\mathcal{F}$) we can transform it into an instance G of Quarantined $K_n \setminus e$-free Edge Deletion using Lemma 18, which in turn we can further transform into an instance G' of $K_n \setminus e$-free Edge Deletion using Lemma 21. Given any feasible solution F' for G' we check whether $|F'| \leq |E(G)|^2$. If this is the case, we translate back the solution F' into a solution F for G (using Lemma 21) and then into a solution for the initial instance $\mathcal{I}$ (using Lemma 18). On the other hand, if $|F'| > |E(G)|^2$, then we may take a trivial solution being an assignment setting all the variables to one. This is an r-approximation where $r > |E(G)|$, as $|E(G)| > n_{vars}$ for the initial instance $\mathcal{I}$. The assignment will satisfy all the contraints and will be at least an r-approximation as we need to assign at least one variable to one, otherwise we may output all zeroes assignment.

4.3 Completeness

To finish the proof of Theorem 3 it remains to show a reduction in the other direction: from $K_n \setminus e$-free Edge Deletion to Min Horn Deletion. We achieve this goal by presenting an A-reduction from the $K_n \setminus e$-free Edge Deletion problem to another variant of MinOnes($\mathcal{F}$), which is Min Horn Deletion-complete.

▶ **Definition 22.** Let $n \geq 5$, and let $t = n(n-1)/2$. We define family of constraints $\mathcal{F}''_n = \{f_n, g_n\}$ as follows:

- $f_n(x_1, x_2, \dots x_t) = 0$ if and only if exactly one of the variables takes value 1;
- $g_n(x_1, x_2, \dots x_{t-1}) = 0$ if and only if all the variables take value 0.

The proof of the following lemma is a technical check that is essentially the same as the proof of Lemma 17. Hence, we leave it to the reader.

▶ **Lemma 23.** *For each $n \geq 5$, the set of constraints $\mathcal{F}''_n = \{f_n, g_n\}$ is weakly positive, and at the same time it is neither 0-valid, nor IHS-B for any B.*

Therefore, by Theorem 15 we know that MinOnes($\mathcal{F}''_n$) is Min Horn Deletion-complete and it suffices to present an A-reduction from $K_n \setminus e$-free Edge Deletion to MinOnes($\mathcal{F}''_n$).

▶ **Lemma 24.** *There is a polynomial-time algorithm, which given an instance G of $K_n \setminus e$-free Edge Deletion produces an instance $\mathcal{I}$ of MinOnes($\mathcal{F}''_n$), such that it is possible to remove exactly k edges in G to make it $K_n \setminus e$-free if and only if one can find a satisfying assignment for $\mathcal{I}$ that sets exactly k variables to 1.*

Proof. Consider an instance G of the $K_n \setminus e$-free Edge Deletion problem. We enumerate all the edges in the graph G as $e_1, e_2, \dots, e_m$, and to each edge e_i we assign a fresh boolean variable x_i. For any induced subgraph H isomorphic to $K_n \setminus e$ we list all its edges $e_{i_1}, e_{i_2}, \dots, e_{i_{t-1}}$ and create a corresponding constraint $g(x_{i_1}, x_{i_2}, \dots, x_{i_{t-1}})$. For any induced clique K containing n vertices and edges $e_{i_1}, e_{i_2}, \dots, e_{i_t}$, we create a constraint

$f(x_{i_1}, x_{i_2}, \ldots, x_{i_t})$. The output instance $\mathcal{I}$ of MinOnes($\mathcal{F}''_n$) is obtained by taking x_i to be the variable set, and putting all the constraints constructed above.

Note that if we delete some edges in the graph G, then an induced copy of the graph $K_n \setminus e$ can be obtained only on vertices that originally were inducing $K_n \setminus e$ or K_n. The constraints in the constructed instance guarantee that in each induced $K_n \setminus e$ subgraph at least one edge from the subgraph must be deleted, and in each induced subgraph K_n either at least two edges should be deleted, or none of the edges should be deleted. So, for any $S \subseteq \{1, 2, \ldots, |E(G)|\}$, the graph $G - F$, where $F = \{e_i : i \in S\}$, is $K_n \setminus e$-free if and only if the assignment $\{x_i = 1 \text{ iff } i \in S\}$ satisfies $\mathcal{I}$. This equivalence of solution sets immediately proves the lemma. ◀

As discussed earlier, Lemma 24 gives an A-reduction from $K_n \setminus e$-free Edge Deletion to MinOnes($\mathcal{F}''_n$), which is Min Horn Deletion-complete, thereby proving that $K_n \setminus e$-free Edge Deletion is A-reducible to Min Horn Deletion. This concludes the proof of Theorem 3.

5 Conclusions

In this work we initiated the study of approximability of edge modification problems related to the classes of H-free graphs. Mirroring known kernelization hardness results, we have shown that the problems are hard to approximate whenever H is a 3-connected graph with at least two non-edges, or it is a long enough path or cycle. It therefore seems that the approximation complexity of H-free Edge Deletion (Completion) somewhat matches the kernelization complexity in the cases considered so far, so it is tempting to formulate a conjecture that for every graph H, the H-free Edge Deletion (Completion) problem admits a polynomial kernel if and only if it admits a poly(OPT)-approximation algorithm. Since neither for kernelization nor for approximability the classification is close to being complete, this conjecture should be regarded as a very distant goal. However, one very concrete open question that arises is whether Cograph Edge Deletion (equivalent to $H = P_4$) admits a poly(OPT)-approximation. Here, we expect the answer to be positive, due to the existence of the polynomial kernel of Guillemot et al. [9]. The same question can be asked about the diamond graph, that is, a K_4 minus an edge; a polynomial kernel for Diamond-Free Edge Deletion was given by Cai [5]. Also, further investigation of the links between the case of a complete graph without one edge and the Min Horn Deletion problem, seems like an interesting direction.

References

1. N. R. Aravind, R. B. Sandeep, and Naveen Sivadasan. Parameterized lower bound and NP-completeness of some h-free edge deletion problems. In *COCOA 2015*, volume 9486 of *LNCS*, pages 424–438. Springer, 2015.
2. Ivan Bliznets, Marek Cygan, Paweł Komosa, Lukáš Mach, and Michał Pilipczuk. Lower bounds for the parameterized complexity of Minimum Fill-in and other completion problems. In *SODA 2016*, pages 1132–1151. SIAM, 2016.
3. Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996.
4. Leizhen Cai and Yufei Cai. Incompressibility of H-free edge modification problems. *Algorithmica*, 71(3):731–757, 2015.
5. Yufei Cai. Polynomial kernelisation of H-free edge modification problems. Master's thesis, The Chinese University of Hong Kong, 2012. Available at author's website.

6 Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, Erik Jan van Leeuwen, and Marcin Wrochna. Polynomial kernelization for removing induced claws and diamonds. *CoRR*, abs/1503.00704, 2015. To appear in the proceedings of WG 2015.

7 Pål Grønås Drange, Fedor V. Fomin, Michał Pilipczuk, and Yngve Villanger. Exploring the subexponential complexity of completion problems. *TOCT*, 7(4):14, 2015.

8 Archontia C. Giannopoulou, Daniel Lokshtanov, Saket Saurabh, and Ondrej Suchý. Tree deletion set has a polynomial kernel (but no $OPT^{O(1)}$ approximation). In *FSTTCS 2014*, volume 29 of *LIPIcs*, pages 85–96. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014.

9 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.

10 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

11 Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001. doi:10.1137/S0097539799349948.

12 Stefan Kratsch and Magnus Wahlström. Two edge modification problems without polynomial kernels. *Discrete Optimization*, 10(3):193–199, 2013.

13 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

14 Assaf Natanzon. Complexity and approximation of some graph modification problems. Master's thesis, Department of Computer Science, Tel Aviv University, 1999.