Two Edge Modification Problems Without Polynomial Kernels

Stefan Kratsch and Magnus Wahlström

Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

Abstract. Given a graph G and an integer k, the Π Edge Completion/Editing/Deletion problem asks whether it is possible to add, edit, or delete at most k edges in G such that one obtains a graph that fulfills the property Π. Edge modification problems have received considerable interest from a parameterized point of view. When parameterized by k, many of these problems turned out to be fixed-parameter tractable and some are known to admit polynomial kernelizations, i.e., efficient preprocessing with a size guarantee that is polynomial in k. This paper answers an open problem posed by Cai (IWPEC 2006), namely, whether the Π Edge Deletion problem, parameterized by the number of deletions, admits a polynomial kernelization when Π can be characterized by a finite set of forbidden induced subgraphs. We answer this question negatively based on recent work by Bodlaender et al. (ICALP 2008) which provided a framework for proving polynomial lower bounds for kernelizability. We present a graph H on seven vertices such that H-free Edge Deletion and H-free Edge Editing do not admit polynomial kernelizations, unless NP $\subseteq$ coNP/poly. The application of the framework is not immediate and requires a lower bound for a Not-1-in-3 SAT problem that may be of independent interest.

1 Introduction

In recent years the kernelizability of edge modification problems has received considerable attention [5, 9, 13–15]. For a graph property Π the Π Edge Completion/Editing/Deletion problem is defined as

Input: A graph $G = (V, E)$ and an integer k.
Parameter: k.
Task: Decide whether adding and/or deleting at most k edges in G yields a graph with property Π.

Edge modification problems have a number of applications, including machine learning, numerical algebra, and molecular biology [6, 17–19]. In typical applications the input graphs arise from experiments and edge modification serves to correct the (hopefully) few errors. For Similarity Clustering, for example, the vertices represent entities that are linked by an edge if their similarity exceeds a certain threshold. Given a perfect similarity measure one would obtain a cluster graph, i.e., a disjoint union of cliques. In practice though, the obtained

Property/Class	Modification	Kernel size	Characterization
Chain	Deletion	$O(k^2)$ [13]	$(2K_2, K_3, C_5)$-free
Split	Deletion	$O(k^4)$ [13]	$(2K_2, C_4, C_5)$-free
Threshold	Deletion	$O(k^3)$ [13]	$(2K_2, C_4, P_4)$-free
Co-Trivially Perfect	Deletion	$O(k^3)$ [13]	$(2K_2, P_4)$-free
Triangle-Free	Deletion	$6k$ [5]	K_3-free
Cluster	Editing	$4k$ [14]	P_3-free
Grid	Editing	$O(k^4)$ [9]	None finite
Bi-connectivity	Completion	$O(k^2)$ [15]	None finite
Bridge-connectivity	Completion	$O(k^2)$ [15]	None finite

Table 1. Kernelization bounds for some edge modification problems.

graph will at best be close to a cluster graph, i.e., within few edge modifications. Clearly a large number of modifications would yield a clustering of the entities that deviates strongly from the similarity measure, allowing the conclusion that the measure is probably faulty. Thus *fpt-algorithms* that solve the problem efficiently when the number of modifications is not too large are a good way of solving this and similar problems (see Section 2 for formal definitions).

Another parameterized way of dealing with edge modification problems lies in kernelization. The notion of *kernelization* or *reduction to a problem kernel* is an important contribution of parameterized complexity, in formalizing preprocessing to allow a rigorous study. A kernelization is a polynomial-time computable mapping that transforms a given instance, say (x, k), of a parameterized problem into an equivalent instance (x', k') with size of x' and value of k' bounded by a computable function in k. It is known that a parameterized problem is fixed-parameter tractable if and only if it is decidable and kernelizable (cf. [11]). However this fact does not imply the existence of a polynomial kernelization.

Related Work: There exists rich literature on the complexity of edge modification problems; recent surveys were given by Natanzon et al. [17] and Burzyn et al. [6]. Cai [7] showed that a very general version of this problem, also allowing vertex deletions, is FPT when the desired property Π can be characterized by a finite set of forbidden induced subgraphs. A finite set of graphs $\mathcal{H}$ is a *finite forbidden subgraph characterization* of a property Π if for any graph G, G has property Π if and only if G does not contain a graph from $\mathcal{H}$ as an induced subgraph, i.e., G is $\mathcal{H}$*-free*.

In Table 1 we give an overview of kernelization results for a number of edge modification problems, along with their $\mathcal{H}$-free characterizations, where applicable. Six of the properties listed in the table can be characterized by such a finite set of forbidden induced subgraphs, leading to the question whether a finite characterization implies the existence of a polynomial kernelization. It is easy to see that the answer is yes if only vertex deletions are allowed since this translates directly to an instance of d-Hitting Set, where the constant d is the largest number of vertices among forbidden induced subgraphs (see [1] for a d-Hitting Set kernelization). Cai posed the question whether the same is

true for the Π Edge Deletion problem, i.e., achieving property Π by at most k edge deletions when $\Pi = \mathcal{H}$-free for some finite set $\mathcal{H}$ of graphs (see [2]). To our best knowledge the kernelizability of the $\mathcal{H}$-free Edge Editing problem is also open. We point out that $\mathcal{H}$-free Edge Completion is equivalent to $\bar{\mathcal{H}}$-free Edge Deletion, where $\bar{\mathcal{H}}$ contains the complements of the graphs in $\mathcal{H}$.

Our Work: The contribution of this paper is to present a small graph H of seven vertices such that H-free Edge Deletion and H-free Edge Editing do not admit polynomial kernelizations unless the polynomial hierarchy collapses (more specifically, unless NP $\subseteq$ coNP/poly, which is known to imply PH$= \Sigma_3^p$ [20]). This result builds upon recent seminal work by Bodlaender et al. [3] showing that a polynomial kernelization for any so-called compositional problem whose unparameterized version is NP-complete would imply such a collapse.

The application of [3] on the H-free modification problems is not immediate. We first define a problem Not-1-in-3 SAT and prove that it does not admit a polynomial kernelization, then reduce this problem to H-free Edge Deletion and H-free Edge Editing, respectively, by polynomial-time reductions with polynomial bounds on the new parameter values. Such reductions were proposed by Bodlaender et al. [4] as a natural way to extend kernelization results to further problems.

Structure of the paper: In Section 2 we introduce some notation as well as giving the necessary definitions from parameterized complexity and reviewing briefly the framework for lower bounds for kernelization. In Section 3 we prove that Not-1-in-3 SAT does not admit a polynomial kernelization, unless NP $\subseteq$ coNP/poly, and extend this result to the version without repeated variables. The latter problem is then reduced to H-free Edge Deletion and H-free Edge Editing, proving the claimed lower bounds, in Section 4. We conclude in Section 5.

2 Preliminaries

2.1 Notation

Graphs: We consider undirected, simple graphs $G = (V, E)$. An *induced subgraph* of G is a graph $G' = (V', E')$ with $V' \subseteq V$ and with $E' = \{\{u, v\} \in E \mid u, v \in V'\}$. We denote this subgraph by $G[V']$. A graph G is *H-free* if it does not contain H as an induced subgraph (i.e., a subgraph G' which is isomorphic to H). For a set $\mathcal{H}$ of graphs, G is *$\mathcal{H}$-free* if it is H-free for every graph $H \in \mathcal{H}$. We denote by K_i, C_i, and P_i the clique, cycle, and path of i vertices respectively. The graph $2K_2$ is the disjoint union of two K_2.

Not-1-in-3 SAT: We define Not-1-in-3 SAT as a satisfiability problem asking for a feasible assignment with at most k ones to a conjunction of not-1-in-3-constraints $R(x, y, z)$, and constraints $(x \neq 0)$. Inputs to the problem consist of such a conjunction F and an integer k. An assignment ϕ to the variables of F is feasible if $\phi(x) = 1$ for every variable x with a constraint $(x \neq 0)$ and

$$(\phi(x), \phi(y), \phi(z)) \in \{(0,0,0), (0,1,1), (1,0,1), (1,1,0), (1,1,1)\}$$

for every not-1-in-3-constraint $R(x, y, z)$. The *weight* of an assignment is the number of ones it assigns.

Parameterized Complexity: A *parameterized problem* $\mathcal{Q}$ over a finite alphabet Σ is a subset of $\Sigma^* \times \mathbb{N}$. The second component is called the *parameter*. The problem $\mathcal{Q}$ is *fixed-parameter tractable* if there is an algorithm that decides whether $(x, k) \in \mathcal{Q}$ in time $f(k) \cdot |x|^c$, where f is a computable function and c is a constant independent of k. Such an algorithm is called an *fpt-algorithm* for $\mathcal{Q}$.

A *kernelization* of $\mathcal{Q}$ is a polynomial-time computable mapping $K : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N} : (x, k) \mapsto (x', k')$ such that

$$\forall (x, k) \in \Sigma^* \times \mathbb{N} : ((x, k) \in \mathcal{Q} \Leftrightarrow (x', k') \in \mathcal{Q}) \text{ and } x', k' \leq h(k),$$

for some computable function $h : \mathbb{N} \to \mathbb{N}$. The kernelization K is *polynomial* if h is a polynomial. We say that $\mathcal{Q}$ *admits a (polynomial) kernelization* if there exists a (polynomial) kernelization of $\mathcal{Q}$. We refer the reader to [10] for an introduction to parameterized complexity.

2.2 Polynomial lower bounds for kernelization

Bodlaender et al. [3] provided the first polynomial lower bounds for the kernelizability of some parameterized problems. They introduced the notions of *or-* respectively *and-compositionality* for parameterized problems and showed that such an algorithm together with a polynomial kernelization yields an *or-* respectively *and-distillation algorithm* for the unparameterized version of the problem. Based on the hypothesis that NP-complete problems do not admit either variant of distillation algorithm, this proves non-existence of polynomial kernelizations for some problems. A related article by Fortnow and Santhanam [12] showed that an or-distillation algorithm for any NP-complete problem would imply NP $\subseteq$ coNP/poly, causing a collapse of the polynomial hierarchy [20]. In this paper we refer only to the results for or-compositional problems and hence omit the prefix for readability.

Let $\mathcal{Q}$ be a parameterized problem. A *composition algorithm* for $\mathcal{Q}$ is an algorithm that on input $(x_1, k), \ldots, (x_t, k) \subseteq \Sigma^* \times \mathbb{N}$ uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$ and outputs (y, k') such that

1. k' is bounded by a polynomial in k and
2. $(y, k') \in \mathcal{Q}$ if and only if $(x_i, k) \in \mathcal{Q}$ for at least one $i \in \{1, \ldots, t\}$.

The *unparameterized version* or *derived classical problem* $\tilde{\mathcal{Q}}$ of $\mathcal{Q}$ is defined by $\tilde{\mathcal{Q}} = \{x\#1^k \mid (x, k) \in \mathcal{Q}\}$, where $\# \notin \Sigma$ is the blank letter and 1 is any letter from Σ.

Theorem 1 ([3, 12]). *Let L be a compositional parameterized problem whose unparameterized version $\tilde{L}$ is NP-complete. The problem L does not admit a polynomial kernelization unless NP $\subseteq$ coNP/poly.*

In [4] Bodlaender et al. proposed the use of polynomial-time reductions, where the new parameter value is polynomially bounded in the old one, as a tool to extend the lower bounds to further problems.

A *polynomial time and parameter transformation* from $\mathcal{Q}$ to $\mathcal{Q}'$ is a polynomial-time mapping $H : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N} : (x, k) \mapsto (x', k')$ such that

$$\forall (x, k) \in \Sigma^* \times \mathbb{N} : ((x, k) \in \mathcal{Q} \Leftrightarrow (x', k') \in \mathcal{Q}') \text{ and } k' \leq p(k),$$

for some polynomial p. We use the name *polynomial parameter transformation*.

Theorem 2 ([4]). *Let $\mathcal{P}$ and $\mathcal{Q}$ be parameterized problems, and let $\tilde{\mathcal{P}}$ and $\tilde{\mathcal{Q}}$ be their derived classical problems. Suppose that $\tilde{\mathcal{P}}$ is NP-complete and $\tilde{\mathcal{Q}} \in$NP. If there is a polynomial parameter transformation from $\mathcal{P}$ to $\mathcal{Q}$ and $\mathcal{Q}$ admits a polynomial kernelization, then $\mathcal{P}$ also admits a polynomial kernelization.*

3 Hardness of Not-One-In-Three SAT

In this section, we show that the Not-1-in-3 SAT problem does not admit a polynomial kernelization, unless NP $\subseteq$ coNP/poly. A polynomial parameter transformation to an H-free Edge Deletion problem and an H-free Edge Editing problem in Section 4 will then yield the desired lower bounds.

To apply Theorem 1 we need to show that Not-1-in-3 SAT is compositional and that the unparameterized version is NP-hard. For showing this, two basic observations are useful. Firstly, we have the ability to force $x = 0$ for a variable x of any instance of Not-1-in-3 SAT, by the next lemma. Secondly, using such a variable, we are able to force $x = y$ by a not-1-in-3-constraint $R(x, y, 0)$ where 0 is a variable that has been forced to be false.

Lemma 1. *Let (F, k) be an instance of Not-1-in-3 SAT containing a variable x. Adding only not-1-in-3-constraints, we can create an instance (F', k) such that there is a feasible assignment for F' with at most k ones if and only if there is a feasible assignment ϕ for F with at most k ones such that $\phi(x) = 0$.*

Proof. For each $1 \leq i \leq k$, add a not-1-in-3-constraint $R(x, y_{2i-1}, y_{2i})$, where y_i are new variables. Now for any feasible assignment ϕ for F', if $\phi(x) = 1$, then k further variables must be assigned 1, implying that ϕ assigns more than k ones. Thus feasible assignments ϕ for F' of weight at most k have $\phi(x) = 0$, and can be restriced to feasible assignments for F.

Feasible assignments ϕ for F with $\phi(x) = 0$ can be extended to feasible assignments for F', setting $y_i = 0$ for all added variables, at no extra cost. □

Lemma 2. *Not-1-in-3 SAT is NP-complete and compositional.*

Proof. The NP-completeness follows from Khanna et al. [16]. This extends also to the variant where k is encoded in unary since values of k greater than the number of variables are meaningless (i.e., can be replaced by the number of variables). We will now show that Not-1-in-3 SAT is compositional.

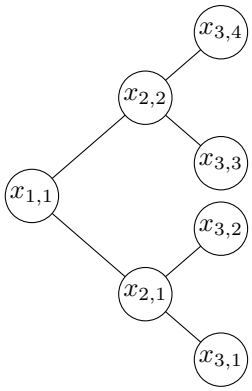

Fig. 1. First three levels of a composition tree. Every parent-node is in a not-1-in-3-constraint with its two sons, e.g. $R(x_{1,1}, x_{2,1}, x_{2,2})$. If $x_{1,1}$ is true, then on every level i some variable $x_{i,j}$ must be true.

Let (F_1, k), ..., (F_t, k) be instances of Not-1-in-3 SAT; we will create a new instance (F', k') with $k' = O(k)$ that is a yes-instance if and only if (F_i, k) is a yes-instance for some $1 \leq i \leq t$. If $t \geq 3^k$, then we have time to solve every instance exactly by branching, and output some dummy yes- or no-instance. We assume for the rest of the proof that $t < 3^k$. We also assume that t is a power of two (or else duplicate some instance (F_i, k)): say $t = 2^l$.

Start building F' by creating variable-disjoint copies of all formulas F_i. For every formula F_i, additionally create an activity variable s_i, and for every variable x in F_i such that F_i contains a constraint $x \neq 0$, replace this constraint by $x = s_i$ (i.e., a not-1-in-3-constraint $R(x, s_i, 0)$ where 0 is a variable forced to be false as in Lemma 1). The variable s_i now decides whether F_i is active: if $s_i = 0$ then every variable of F_i can be set to zero, and if $s_i = 1$, then all constraints of the formula F_i are active. We will complete the construction by creating a formula that forces some s_i to be true, in a form of *composition tree*.

Figure 1 illustrates the construction used for this part. We use variables $x_{i,j}$ where i is the *level* of the variable, and $1 \leq j \leq 2^{i-1}$. The first-level variable $x_{1,1}$ is forced to be true by a constraint $(x_{1,1} \neq 0)$, and for every internal variable in the composition tree, we use a not-1-in-3-constraint $R(x_{i,j}, x_{i+1,2j-1}, x_{i+1,2j})$. Finally, for every variable $x_{l+1,j}$ on the last level, we add a constraint such that $s_j = x_{l+1,j}$. Since a not-1-in-3-constraint $R(x, y, z)$ implies $x \to (y \vee z)$, we see inductively that at least one variable per level must be true, and furthermore for every variable $x_{l+1,j}$ on the last level, there exists an assignment where $x_{l+1,j} = 1$ and exactly one variable per level is true. Thus, assuming a minimal assignment to the composed instance F', we can treat the composition tree as adding weight exactly $l + 1$ and encoding a disjunction over the activity variables s_j. We obtain the composed instance $(F', k + l + 2)$.

Now if F' has an assignment ϕ' of weight at most $k + l + 2$ then $\phi'(s_j) = 1$ for some $j \in \{1, \ldots, t\}$. Thus the constraints of F_j are active and ϕ' can be

restricted to a feasible assignment for F_j (excluding s_j) of weight at most k. If some input formula F_j has an assignment of weight at most k then this can be extended to an assignment for F' of weight at most $k + l + 2$ by assigning 1 to each $x_{i,l}$ on the path from s_j to $x_{1,1}$ in the composition tree and 0 to all other variables of F'. Finally, the parameter $k' = k + l + 2$ is $O(k)$ since $t < 3^k$, and the work performed is polynomial in the length of all input instances. □

Now by Theorem 1 we obtain the following result.

Theorem 3. *Not-1-in-3 SAT does not admit a polynomial kernelization unless NP ⊆ coNP/poly.*

To simplify the reduction to H-free Edge Deletion in the following section, we consider a variant of Not-1-in-3 SAT that does not allow repeated variables in the not-1-in-3-constraints, e.g. $R(x, x, y)$. The following lemma implicitly extends our lower bound to that variant.

Lemma 3. *Not-1-in-3 SAT reduces to Not-1-in-3 SAT without repeated variables by a polynomial parameter transformation.*

Proof. Observe that both the construction of Lemma 1 to force a variable to take value 0, and the usage of a not-1-in-3-constraint $R(x, y, 0)$ to force $x = y$ work without repeating variables. Therefore, from an instance (F, k) of Not-1-in-3 SAT, we can create an instance (F', k') of Not-1-in-3 SAT without repeated variables where $k' = 2k$ in the following manner:

1. Place constraints in F' forcing $z = 0$ for some unique variable z according to Lemma 1.
2. For every variable x in F, create two variables x, x' in F'. Force $x = x'$ using a not-1-in-3-constraint $R(x, x', z)$.
3. For every not-1-in-3-constraint $R(x, x, y)$ in F, with $x \neq y$, place $R(x, x', y)$ in F'. Copy not-1-in-3-constraints $R(x, y, z)$ for distinct variables x, y, z and constraints $x \neq 0$ into F', and ignore any not-1-in-3-constraints $R(x, x, x)$.

The formula F' has a solution with at most $2k$ true variables iff F has a solution with at most k true variables. □

4 Lower bounds for two H-free modification problems

Throughout this section let $H = (V_H, E_H)$ with vertex set $V_H = \{a, b, c, d, e, f, g\}$ and edge set $E_H = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{a, f\}, \{a, g\}, \{b, c\}, \{d, e\}\}$, as depicted in Figure 2. We will show that Not-1-in-3 SAT without repeated variables reduces to H-free Edge Deletion by a polynomial parameter transformation, thereby proving that the latter problem does not admit a polynomial kernelization unless NP ⊆ coNP/poly, according to Theorem 2. Consecutively we use a slightly more involved reduction from Not-1-in-3 SAT without repeated variables to H-free Edge Editing.

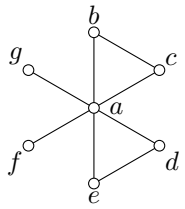

Fig. 2. The graph H.

Lemma 4. *Not-1-in-3 SAT without repeated variables reduces to H-free Edge Deletion by a polynomial parameter transformation.*

Proof. Let (F, k) be an instance of Not-1-in-3 SAT without repeated variables. We will construct an equivalent instance (G, k') of H-free Edge Deletion.

Let α be the number of variables x of F that have a constraint $(x \neq 0)$. Clearly, if $k < \alpha$ then (F, k) is a no-instance and we map it to a dummy no-instance of H-free Edge Deletion, e.g. $(H, 0)$. Henceforth we assume that $k \geq \alpha$ and we let $k' = k - \alpha$. Starting from the empty graph we use the following two steps to construct G:

1. For each variable x of F we add two vertices p_x and q_x to G. We add the edge $\{p_x, q_x\}$ to G if there is no constraint $(x \neq 0)$ in the formula F.
2. For each not-1-in-3-constraint $R(x, y, z)$ add $k' + 1$ new vertices $r_1, \ldots, r_{k'+1}$ to G. Connect each r_i to p_x, q_x, p_y, q_y, p_z, and q_z. Recall that x, y, and z are different by our restriction on the source problem.

See Figure 3 for an example of a formula F and the resulting graph. It is easy to see that this construction can be accomplished in polynomial time and that $k' = k - \alpha \in O(k)$.

We denote by X the set of vertices created in Step 1 and by C the set of vertices created in Step 2. Thus $G = (X \cup C, E)$. We make two simple observations:

(1) Each vertex of X has at most one neighbor in X (e.g. p_x may only be adjacent to q_x).
(2) C is an independent set in G.

We will now show that (F, k) is a yes-instance of Not-1-in-3 SAT without repeated variables if and only if (G, k') is a yes-instance of H-free Edge Deletion.

Suppose that (F, k) is a yes-instance of Not-1-in-3 SAT without repeated variables and let ϕ be a feasible assignment for F with at most k ones. We define a subset $D \subseteq E$ as those edges $\{p_x, q_x\}$ of G with $\phi(x) = 1$. Observe that $\{p_x, q_x\} \in E$ if and only if F contains no constraint $(x \neq 0)$. Therefore $|D| \leq k - \alpha = k'$ since at most k variables x have $\phi(x) = 1$ but α of those variables have a constraint $(x \neq 0)$.

We assume for contradiction that $G - D$ is not H-free. Let $a, \ldots, g$ be vertices of G that induce a subgraph H. For simplicity let the adjacencies be the same as in Figure 2.

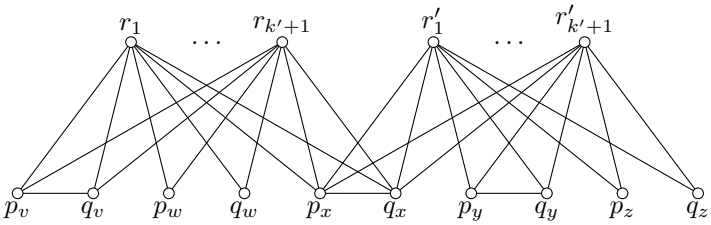

Fig. 3. The resulting graph for $F = R(v, w, x) \wedge R(x, y, z) \wedge (w \neq 0) \wedge (z \neq 0)$.

It is easy to see that $a \in C$: Otherwise, if $a \in X$, then by Observation (1) at most one of its neighbors can be in X. This would imply that $b, c \in C$ or $d, e \in C$ contradicting Observation (2), namely, that C is an independent set.

Since $a \in C$ the other vertices $b, \ldots, g$ must be in X by Observation (2). Recall that in $G[X]$, i.e., among vertices of X, there are only edges of the form $\{p_x, q_x\}$ where x is a variable of F. Since $\{b, c\}$ and $\{d, e\}$ are edges of $G[X]$ there must be variables x and y of F such that w.l.o.g. $b = p_x$, $c = q_x$, $d = p_y$, and $e = q_y$. By Step 2 of the construction there must be a third variable z of F such that w.l.o.g. $f = p_z$ and $g = q_z$.

From $\{p_x, q_x\}, \{p_y, q_y\} \in E \setminus D$ (i.e., the edge set of $G - D$) it follows that $\phi(x) = \phi(y) = 0$. Since $\{p_z, q_z\} \notin E \setminus D$ it follows that $\{p_z, q_z\} \in D$ or $\{p_z, q_z\} \notin E$. In the first case we have that $\phi(z) = 1$. In the latter case there must be a constraint $(z \neq 0)$ in F, which also implies that $\phi(z) = 1$ since ϕ is feasible for F. However, $\phi(x) = \phi(y) = 0$ and $\phi(z) = 1$ contradict the feasibility of ϕ since $R(x, y, z)$ is a not-1-in-3-constraint of F. Hence $G - D$ is an H-free graph, implying that (G, k') is a yes-instance of H-free Edge Deletion, since $|D| \leq k'$.

Now, suppose that (G, k') is a yes-instance of H-free Edge Deletion. We will construct a feasible assignment ϕ with at most k ones for F. Let $D \subseteq E$ with $|D| \leq k'$ such that $G - D$ is H-free. We define ϕ by

$$\phi(x) = \begin{cases} 1 & \text{if } \{p_x, q_x\} \in D, \\ 1 & \text{if } (x \neq 0) \text{ is a constraint of } F, \\ 0 & \text{otherwise.} \end{cases}$$

Thus the number of ones of ϕ is at most the cardinality of D plus the number of variables x of F with an $(x \neq 0)$-constraint, i.e., $|D| + \alpha \leq k$.

We assume for contradiction that ϕ is not feasible for F. This implies that there is a not-1-in-3-constraint, say $R(x, y, z)$, in F that is not satisfied by ϕ. The reason is that, by definition, ϕ satisfies all $(x \neq 0)$-constraints.

Not satisfying $R(x, y, z)$ implies that ϕ sets exactly one of the three variables to 1. Recall that x, y, and z must be pairwise different by problem definition. W.l.o.g. we assume that $\phi(x) = 1$ and $\phi(y) = \phi(z) = 0$. This immediately implies that there are no constraits $(y \neq 0)$ or $(z \neq 0)$ in F and that $\{p_y, q_y\}$ and $\{p_z, q_z\}$ are not contained in D, by definition of ϕ. Hence $\{p_y, q_y\}$ and $\{p_z, q_z\}$ are contained in the edge set $E \setminus D$ of $G - D$.

Since $\phi(x) = 1$ we conclude that $\{p_x, q_x\} \in D$ or that $(x \neq 0)$ is a constraint of F. In the latter case, by Step 1 in the construction of G, the edge $\{p_x, q_x\}$ does not occur in G. Thus, in both cases, $\{p_x, q_x\}$ is not an edge of $G - D$, i.e., $\{p_x, q_x\} \notin E \setminus D$.

We will now consider the vertices that were added for $R(x, y, z)$ in Step 2 of the construction, say $r_1, \ldots, r_{k'+1}$. Since $|D| \leq k'$ at least one r_i is adjacent to all six vertices p_x, q_x, p_y, q_y, p_z, and q_z. Together with r_i those vertices induce a subgraph H. This contradicts the choice of D, implying that ϕ is feasible for F.

Thus (F, k) is a yes-instance of Not-1-in-3 SAT without repeated variables if and only if (G, k') is a yes-instance of H-free Edge Deletion. □

By Theorem 2 we obtain the desired result.

Theorem 4. *For the graph $H = (V_H, E_H)$ with $V_H = \{a, b, c, d, e, f, g\}$ and with $E_H = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{a, f\}, \{a, g\}, \{b, c\}, \{d, e\}\})$ the H-free Edge Deletion problem does not admit a polynomial kernelization unless NP $\subseteq$ coNP/poly.*

The following corollary proves that the H-free Edge Editing problem does not admit a polynomial kernelization either.

Corollary 1. *For the same graph H as in Theorem 4, the H-free Edge Editing problem does not admit a polynomial kernelization unless NP $\subseteq$ coNP/poly.*

Proof. We extend the construction used in Lemma 4 to obtain a polynomial parameter transformation from Not-1-in-3 SAT without repeated variables to H-free Edge Editing. Let (F, k) be an instance of Not-1-in-3 SAT without repeated variables. We create an instance (G, k') according to the construction in Lemma 4, except for using $3k' + 1$ vertices r_i per not-1-in-3-constraint in Step 2.

Essentially we only need one additional gadget to ensure that no edges are added between vertices of X in G (recall that X contains vertices p_x and q_x for every variable of F). Adding this gadget for every non-edge of $G[X]$ will yield a graph G' and we will show that (G', k') and (F, k) are equivalent.

Firstly, let us recall that the only edges in $G[X]$ are of the form $\{p_x, q_x\}$ for some variable x of F that has no constraint $(x \neq 0)$. For every other pair of vertices, say (p_x, p_y) with $x \neq y$, we add $k' + 1$ vertices s_i, t_i, u_i, v_i, and w_i, and for each i connect s_i to p_x, p_y, t_i, u_i, v_i, and w_i, and connect t_i to u_i. That is, create constructions which will induce $k' + 1$ subgraphs H if the edge $\{p_x, p_y\}$ is added. Note that these subgraphs have only one pair of vertices in common. The same construction applies also for pairs (p_x, q_y) and (q_x, q_y) as well as for pairs (p_x, q_x) where x does have a constraint $(x \neq 0)$ in F. Let G' denote the graph that is obtained by adding these vertices and edges to G.

Now, for proving equivalence of (G', k') and (F, k), let us assume that (G', k') is a yes-instance of H-free Edge Editing and let D be a set of edges, with $|D| \leq k'$ such that $G' \triangle D := (V(G'), E(G') \triangle D)$ is H-free.

Firstly, we claim that D adds no edge between vertices of X. Otherwise, if for example $\{p_x, p_y\} \in D$ then, referring to the new gadget, there would

be $k' + 1$ induced subgraphs H that share only p_x and p_y. Since removal of these $k' + 1$ subgraphs would demand at least $k' + 1$ modifications, this would imply that $G' \triangle D$ is not H-free. Thus D restricted to $G'[X]$ contains only deletions; let D' be these deletions. Clearly $|D'| \leq k'$.

Let us consider the r-vertices that are added in Step 2. Clearly, since $|D| \leq k'$, for every not-1-in-3-constraint of F at least $k' + 1$ of its corresponding vertices r_i are not incident with edges of D (recall that we started with $3k'+1$ vertices r_i per constraint). We select a set R of vertices, picking, for every not-1-in-3-constraint of F, $k' + 1$ of its corresponding r-vertices which are not incident with edges of D. Let G'' be the graph induced by $R \cup X$.

Now let (G_0, k') be the H-free Edge Deletion-instance created from (F, k) in Lemma 4. It is easy to see that G_0 and G'' are isomorphic, and that the only modifications made to G'' by D are the deletions D'. Since $G' \triangle D$ is H-free, so is $G'' \triangle D' = (G' \triangle D)[R \cup X]$. Thus D' constitutes a solution to the H-free Edge Deletion instance (G'', k), which by Lemma 4 maps to a solution to (F, k).

Let us now assume that (F, k) is a yes-instance and let ϕ be a feasible assignment of weight at most k. Let D be obtained as in Lemma 4, i.e., D will only delete some $\{p_x, q_x\}$-edges from G'. We show that $G' \triangle D$ is H-free.

Consider the vertex which would form the vertex a of H, i.e., the vertex of degree six in H. It cannot be a p-vertex, since the neighborhood of a p-vertex contains only edges incident on the corresponding q-vertex (all r- and s-vertices are independent of one another, and a p-vertex has only such neighbors, in addition to its q-vertex). By symmetry, this excludes q-vertices as well. The r-vertices have only six neighbors, so any induced H would correspond to a contradicted constraint R which by assumption does not exist (same argument as in Lemma 4). Furthermore, in the solution constructed, the neighborhood of any s-vertex contains only one edge, as D only deletes edges. The remaining types of vertices, i.e., the t, u, v, and w vertices of the gadget, have degree less than six. □

5 Conclusion

We have presented a small graph H such that the H-free Edge Deletion problem and the H-free Edge Editing problem do not admit polynomial kernelizations under a reasonable complexity hypothesis. This answers an open question by Cai, namely, whether the $\mathcal{H}$-free Edge Deletion problem admits a polynomial kernelization for every finite set $\mathcal{H}$ of graphs.

It is an interesting open problem to further characterize kernelizability of $\mathcal{H}$-free modification problems. Considering the structure of the known positive examples (see introduction), one might first ask whether the $\mathcal{H}$-free Edge Deletion problem always admits a polynomial kernelization when $\mathcal{H}$ consists only of paths, cycles, and cliques (or sums thereof, such as $2K_2$). One particular interesting open case here is Cograph Deletion, i.e., P_4-free Edge Deletion.

References

1. Faisal N. Abu-Khzam. Kernelization algorithms for d-hitting set problems. In Dehne et al. [8], pages 434–445.
2. Hans L. Bodlaender, Leizhen Cai, Jianer Chen, Michael R. Fellows, Jan Arne Telle, and Dániel Marx. Open problems in parameterized and exact computation – IWPEC 2006, 2006.
3. Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels (extended abstract). In *ICALP (1)*, volume 5125 of *LNCS*, pages 563–574. Springer, 2008.
4. Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Analysis of data reduction: Transformations give evidence for non-existence of polynomial kernels. Technical Report UU-CS-2008-030, Department of Information and Computing Sciences, Utrecht University, 2008.
5. Daniel Brügmann, Christian Komusiewicz, and Hannes Moser. On generating triangle-free graphs. *Electronic Notes in Discrete Mathematics*, 32:51–58, 2009.
6. Pablo Burzyn, Flavia Bonomo, and Guillermo Durán. NP-completeness results for edge modification problems. *Discrete Applied Mathematics*, 154(13):1824–1844, 2006.
7. Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.
8. Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh, editors. *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, volume 4619 of *Lecture Notes in Computer Science*. Springer, 2007.
9. Josep Díaz and Dimitrios M. Thilikos. Fast fpt-algorithms for cleaning grids. In Bruno Durand and Wolfgang Thomas, editors, *STACS*, volume 3884 of *Lecture Notes in Computer Science*, pages 361–371. Springer, 2006.
10. Rod G. Downey and M. R. Fellows. *Parameterized Complexity (Monographs in Computer Science)*. Springer, November 1998.
11. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer, March 2006.
12. Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *STOC*, pages 133–142. ACM, 2008.
13. Jiong Guo. Problem kernels for NP-complete edge deletion problems: Split and related graphs. In Takeshi Tokuyama, editor, *ISAAC*, volume 4835 of *Lecture Notes in Computer Science*, pages 915–926. Springer, 2007.
14. Jiong Guo. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, 410(8-10):718–726, 2009.
15. Jiong Guo and Johannes Uhlmann. Kernelization and complexity results for connectivity augmentation problems. In Dehne et al. [8], pages 483–494.
16. Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing.*, 30(6):1863–1920, 2000.
17. Assaf Natanzon, Ron Shamir, and Roded Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109–128, 2001.
18. Donald J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. *Graph Theory and Computing*, pages 183–217, 1972.

19. Roded Sharan, Adi Maron-Katz, and Ron Shamir. CLICK and EXPANDER: a system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–1799, 2003.
20. Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theor. Comput. Sci.*, 26:287–300, 1983.