(Sub)linear Kernels for Edge Modification Problems Towards Structured Graph Classes

Gabriel Bathie ✉
École Normale Supérieure de Lyon, France

Nicolas Bousquet ✉
LIRIS, CNRS, Université Claude Bernard Lyon 1, Université de Lyon, France

Théo Pierron ✉
LIRIS, CNRS, Université Claude Bernard Lyon 1, Université de Lyon, France

Abstract

In a (parameterized) graph edge modification problem, we are given a graph G, an integer k and a (usually well-structured) class of graphs $\mathcal{G}$, and ask whether it is possible to transform G into a graph $G' \in \mathcal{G}$ by adding and/or removing at most k edges. Parameterized graph edge modification problems received considerable attention in the last decades.

In this paper, we focus on finding small kernels for edge modification problems. One of the most studied problems is the Cluster Editing problem, in which the goal is to partition the vertex set into a disjoint union of cliques. Even if this problem admits a $2k$ kernel [7], this kernel does not reduce the size of most instances. Therefore, we explore the question of whether linear kernels are a theoretical limit in edge modification problems, in particular when the target graphs are very structured (such as a partition into cliques for instance). We prove, as far as we know, the first sublinear kernel for an edge modification problem. Namely, we show that Clique + Independent Set Deletion, which is a restriction of Cluster Deletion, admits a kernel of size $O(k/\log k)$.

We also obtain small kernels for several other edge modification problems. We prove that Split Addition (and the equivalent Split Deletion) admits a linear kernel, improving the existing quadratic kernel of Ghosh et al. [19]. We complement this result by proving that Trivially Perfect Addition admits a quadratic kernel (improving the cubic kernel of Guo [21]), and finally prove that its triangle-free version (Starforest Deletion) admits a linear kernel, which is optimal under ETH.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases kernelization, graph editing, split graphs, (sub)linear kernels

Digital Object Identifier 10.4230/LIPIcs.IPEC.2021.8

Related Version *Full Version*: https://arxiv.org/abs/2105.09566

Funding This work was supported by ANR project GrR (ANR-18-CE40-0032).

1 Introduction

A central problem in the context of data transmission, collection or storage, is to recover the original information when the data has been altered. Although it is not possible to know what the original data was in the general setting, it may be possible when we have some knowledge of the structure of the original data. When we know that the alteration is limited, it is reasonable to assume that the original data is an element that has the desired structure and is the closest to the altered data.

When the data that we are reconstructing is a graph, the problem becomes the following: given a graph G (the altered data) and a class of graphs $\mathcal{G}$ (the structure of the data), find the graph in $\mathcal{G}$ that is the "closest" to G (candidate for the original data). There are multiple ways to define the distance between two graphs, but the most widely used is the minimum number of vertex modifications or edge modifications needed to turn one into the other. This type of problem, called graph modification problems, received considerable attention, for instance in computational biology [2], machine learning [1], and image processing [28].

In this work, we focus on edge modification problems, i.e. the distance is the minimum number of edge modifications (see Section 2 for formal definitions of these problems). A line of work, initiated by Yannakakis [29], showed that deciding whether a graph G is at distance at most k from $\mathcal{G}$ is NP-complete for most classes of graphs, even for very restricted classes such as bipartite graphs. See [5, 25, 26] for an overview of the different results.

Therefore, in the last decades, edge modification problems received considerable attention from the point of view of parameterized complexity, which studies the resources required to solve NP-complete problems in a fine-grained way. See for example [3, 4, 6, 12, 15, 19, 27], and see [10] for a recent survey on the topic. In this paper, we will consider problems parameterized by the size k of the solution (that is, the set of edges to add or remove).

In this work, we focus on graph classes that can be characterized by a finite number of forbidden induced subgraphs. In his seminal paper, Cai [6] showed that, for every class of graphs $\mathcal{G}$ that can be characterized by a finite number of forbidden induced subgraphs, the $\mathcal{G}$-edge modifications problems are FPT parameterized by k. In other words, there exists a constant c, a function f (that only depend on $\mathcal{G}$), and an algorithm running in time $f(k) \cdot n^c$ that either finds a solution of size at most k, or returns that there is no such solution. Therefore, most of the subsequent efforts focused on determining for which $\mathcal{G}$ these problems admit a polynomial kernel. Intuitively, a kernel is a polynomial-time preprocessing algorithm that extracts the "hard" part of an instance (G, k): it solves easy parts of the instance and returns an equivalent instance (G', k'), whose size is bounded by $f(k)$, for some function f. A kernel is a polynomial kernel if f is a polynomial. The interested reader is referred to [18] for more details.

One of the most studied edge modification problem is Cluster Editing, in which the goal is to partition the graph into a disjoint union of cliques. This problem is known to admit a kernel with at most $2k$ vertices. While this result seems impressive at first glance (for many parameterized problems, a linear kernel is asymptotically optimal), we can remark that here, we are comparing the number of vertices of the kernel with the number of edges in the solution. It turns out that for most graphs in practice, the number of edges that have to be modified to obtain a cluster graph is larger than the number of vertices. For example, it is the case for most of the public instances of the PACE challenge 2021 on cluster editing[1].

This raises the question of whether linear kernels are optimal, in particular for Cluster Editing. We partially answer this question by giving a sublinear kernel for the closely related Clique + IS Deletion problem. It provides a "proof of concept" that linear kernels are not always optimal. As far as we know, it is the first example of a sublinear kernel for a graph edge modification problem. We complete this result with linear or quadratic kernels for several other edge modification problems.

Due to space constraints, all the proofs that are not in this extended abstract can be found in the appendix.

Our results. In this work, our goal is to understand when it is possible to obtain small, and in particular linear or sublinear kernels for edge modification problems. We focus in particular on graph classes where the vertex set can be partitioned into highly structured classes such as cliques or independent sets. A typical example of such graph class is the class of split graphs, i.e. graphs that can be partitioned into a clique and an independent set.

[1] See `https://pacechallenge.org/2021/` for more information.

Most of our results are based on a high-level technique, that we call Label-And-Reduce, which helps to design efficient kernelization algorithms for edge modification problems. The key idea is to use the strong structure of each of the graphs in $\mathcal{G}$ to find a highly structured partition $X_1, \dots, X_\ell$ of the graphs of $\mathcal{G}$ (e.g. a partition in cliques or independent sets, complete bipartition between subsets...). We then define rules that label vertices x as belonging to X_i, in such a way that if there is a solution, there is one for which $x \in X_i$. We finally show that 1) when no rule can be applied, the number of unlabeled vertices is $O(\text{poly}(k))$ when (G, k) is a positive instance and 2) the number of labeled vertices in each class X_i can be reduced to $O(\text{poly}(k))$.

Sublinear kernel for Clique+Independent Set deletion. The problems of graph edge modification towards cluster graphs have received considerable attention in the last two decades in parameterized complexity, see e.g. [4, 7, 9, 17, 27].

The deletion version, Cluster Deletion admits a cubic kernel [20]. We focus on a restricted version of this problem, where all clusters but at most one have size 1. It corresponds to graphs that are the disjoint union of a clique and an independent set. In what follows, we will refer to this class as the class of *clique + IS graphs*, and to the corresponding problem as Clique + IS Deletion. Since clique + IS graphs are $(P_3, 2K_2)$-free graphs, Clique + IS Deletion is FPT by [6].

While Clique + IS Addition is trivial, both Clique + IS Deletion and Clique + IS Edition are NP-complete (reduction from the Clique problem), and both can be solved in subexponential time ($O^*(1.64^{\sqrt{k \ln k}})$ and $O^*(2^{\sqrt{k \ln k}})$ respectively[2]) [12]. Both problems also admit a simple $2k$-kernel, based on twin reduction rules [10].

Our result, proved in Section 3, is the following.

▶ **Theorem 1.1.** *Clique + IS Deletion admits a kernel of size $2k/\log k + 1$.*

Our algorithm uses the structure of clique+IS graphs to remove vertices with small degree and to reduce the instance when the minimum degree of the input is large. Theorem 1.1 is, as far as we know, the first sublinear kernel for edge modification problems. We conjecture that the size of this kernel is not optimal, and ask the following:

▶ **Open Problem 1.** Is there an $O(k^{1-\varepsilon})$ kernel for Clique + IS Deletion for some $\varepsilon > 0$?

Moreover, it is plausible that other edge modification problems towards highly structured classes also admit a sublinear kernel. A natural candidate is the closely related Cluster Editing problem, which already admits a $2k$ kernel [7, 9].

▶ **Open Problem 2.** Does Cluster Editing (resp. Cluster Deletion) admit a sublinear kernel?

Linear kernel for Split addition. Split graphs are graphs whose vertex set can be partitioned into a clique K and an independent set I (with no constraint on the set of edges between K and I). Since split graphs are auto-complementary, the Split Addition and Split Deletion problems are equivalent. Natanzon et al. showed that these two problems are NP-complete [26]. Since split graphs are $(2K_2, P_4, C_5)$-free graphs, the latter problems are

[2] Recall that O^* denotes the complexity up to polynomial factors.

FPT [6]. Ghosh et al. [19] later showed that these problems can be solved in subexponential $O^*(2^{O(\sqrt{k}\log k)})$ time and that they admit a quadratic kernel. Cygan et al. [11] improved the complexity to $O^*(2^{\sqrt{k}})$. Hammer and Simeone [22] showed that, rather surprisingly, the related Split Edition problem can be decided in polynomial time.

We improve upon the result of Ghosh et al. [19] by showing that the Split Addition (and therefore Split Deletion) problem admits a linear kernel in Section 4.

▶ **Theorem 1.2.** *Split Addition and Split Deletion admit a kernel with at most $11k + 6\sqrt{2k} + 4$ vertices.*

This result is the main technical contribution of the paper. From a very high-level perspective, our algorithm works as follows. Let (G, k) be a positive instance. If the clique of the solution is large enough, the neighborhood of many vertices of that clique has not been modified and we show that we can detect some of them and label them as clique vertices. Since the number of unlabeled clique vertices of the solution is bounded by a linear function, we can prove via a tricky and short argument that the number of unlabeled vertices of the independent set can be bounded. While the reduction rules are not very complicated, showing that the answer is negative when the number of unlabeled vertices is too large is the core of the proof. We finally show that we reduce the number of labeled vertices to $O(k)$ vertices.

Quadratic kernel for trivially perfect graphs. A trivially perfect graph is a graph such that for any pair of adjacent vertices u, v, $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$. The class of trivially perfect graphs can equivalently be characterized as the class of (P_4, C_4)-free graphs. Drange et al. [13, 14] showed that, under the Exponential Time Hypothesis (ETH), Trivially Perfect Deletion and Trivially Perfect Edition cannot be solved in subexponential time. Liu et al. [24] gave an FPT algorithm for Trivially Perfect Deletion running in time $O^*(2.42^k)$. On the other hand, the Trivially Perfect Addition problem does not admit such lower bounds. Drange et al. [13] designed a subexponential $O(2^{\sqrt{k \log k}})$ algorithm for the problem and Bliznets et al. [3] showed that assuming ETH, this cannot be improved beyond $O(2^{k^{1/4}})$. In 2018, Drange and Pilipczuk [14] showed that the three problems admit a polynomial kernel of size $O(k^7)$, recently improved by Dumas et al. [16] into $O(k^3)$.

In the specific case of Trivially Perfect Addition, a cubic kernel was already provided by Guo [21]. We improve this result in Section 5 by showing the following.

▶ **Theorem 1.3.** *Trivially Perfect Addition admits a kernel with $2k^2 + 2k$ vertices.*

Our kernel is based on the claim of Guo [21], which states that the instance can be reduced to vertices that belong to at least one obstruction (that is, an induced P_4 or C_4). Using this claim, Guo proved the existence of a cubic kernel. By counting obstructions more precisely, we actually show very simply that the size of the kernel can be reduced to $O(k^2)$.

Linear kernelization of starforests. We finally focus on triangle-free trivially perfect graphs, also known as star forests. A *star* is a tree with at most one internal vertex. A star with n vertices is called an *n-star*. Note that the single vertex graph and K_2 are stars. The class of *starforests* graphs is the class of graphs that are a disjoint union of stars, that is every connected component is a star. Alternatively, it may be defined as the class of K_3, C_4, P_4-free graphs.

One can remark that removing an edge from a starforest yields another starforest, hence it is never interesting to add edges to obtain a star forest. Therefore, Starforest Addition is trivial, and Starforest Edition is equivalent to Starforest Deletion. Drange et al. showed in [15] that Starforest Deletion is NP-complete and cannot be solved in subexponential time (that is in time $O(2^{o(k)}\operatorname{poly}(n))$), assuming the ETH [23].

In Section 6, we prove the following result.

▶ **Theorem 1.4.** *Starforest Deletion admits a kernel with at most $4k + 2$ vertices.*

We also show that, under ETH, Starforest Deletion does not admit a sublinear kernel. To the best of our knowledge, this work is the first formally published work on kernelization of Starforest Deletion.

Note. Cao and Yuping [8] obtained independently at the same time results that are very similar to ours: they designed the same quadratic kernel for Trivially Perfect Addition and obtained a similar kernel for Split Addition. However, they were only able to prove an $O(k^{1.5})$ upper bound for the latter, whereas we prove a tighter $O(k)$ bound.

2 Preliminaries

Elementary definitions. In this work, all the graphs are undirected and simple (i.e. with no parallel edges or self-loops). When G is a graph, $V(G)$ denotes the set of vertices of G, and $E(G)$ denotes its set of edges. Throughout the paper, we use n (resp. m) to denote the size of $V(G)$ (resp. $E(G)$). If $uv \in E(G)$, we say that u and v are *adjacent*. Given a vertex $u \in V(G)$, $N(u) = \{v \text{ such that } uv \text{ is an edge}\}$ is the *open neighborhood* of u, and $N[u] = N(u) \cup \{u\}$ is the *closed neighborhood* of u. The *degree* of u in G, denoted $d(u)$, is the size of $N(u)$. We use $\delta(G)$ to denote the minimum degree of G. The *complement graph* $\bar{G}$ of G is the graph with vertex set $V(G)$ and edge set $\{uv : u \neq v \text{ and } uv \notin E(G)\}$. A *dominating set* of G is a set D of vertices of G such that every vertex of G is either in D or adjacent to a vertex of D. An *independent set* of G is a set I of vertices of G that are pairwise not adjacent.

Kernelization algorithms. A *kernelization algorithm* (in short, a kernel) is a polynomial-time algorithm that takes as input an instance (G, k) of a parameterized problem Π and outputs an instance (G', k') that is positive if and only if (G, k) is positive, the size of G' is at most $f(k')$ for some computable function f. When f is a polynomial, we say that the algorithm is a *polynomial kernel*. When dealing with graph problems, the size of the instance is often measured in terms of the number of vertices of G'. Most kernelization algorithms (including those presented in this report) consist of the iterative application of *reduction rules*. A reduction rule is a polynomial-time algorithm that input an instance (G, k) and outputs another instance (G', k'). We say that a reduction rule R is *safe* when (G', k') is positive if and only if (G, k) is.

Graph edge modification problems. Let $\mathcal{G}$ be a class of graphs. In a (parameterized) $\mathcal{G}$-graph edge modification problem, we are given a graph G, an integer k, and ask whether it is possible to transform G into a graph $G' \in \mathcal{G}$ by modifying (*adding*, *removing*, or doing both, which is called *editing*) at most k edges.

Given a set of edges F, we use the notation $G + F$, $G - F$, and $G\Delta F$ to denote the graphs with vertex set $V(G)$ and respective set of edges $E(G) \cup F$, $E(G) \setminus F$ and $E(G)\Delta F$.

Formally, we will consider the following problems:

▶ **Problem 1** ($\mathcal{G}$-ADDITION (resp. DELETION, resp. EDITION)).
Input: A graph G, an integer $k \in \mathbb{N}$.
Output: "YES" if there exists a set F of at most k edges of G such that $G+F$ (resp. $G-F$, resp. $G\Delta F$) is in $\mathcal{G}$, "NO" otherwise.

3 Sublinear kernel for the Clique + Independent set deletion problem

The goal of this section is to prove Theorem 1.1. To obtain the announced kernel, we apply the LABEL-AND-REDUCE technique. For this problem, the labeling rules aim to identify vertices that will be in the independent set of a solution if it exists. We can then delete all the edges incident to these vertices, decrease the parameter accordingly and remove these vertices from the graph.

We assume that k is smaller than m since otherwise, the instance is trivial: we obtain an independent set (which is a clique+IS graph) by deleting all the edges in G.

▶ **Rule 3.1** (Low degree reduction rule 1). *If $v \in V(G)$ has degree $d(v) < \sqrt{2(m-k)} - 1$, delete v from G and decrease the parameter by $d(v)$.*

This rule can be implemented to run in linear time. It is moreover safe. Indeed, since we consider the deletion problem, any vertex v deleted by the rule has degree smaller than $\sqrt{2(m-k)} - 1$ in $G-F$, hence cannot be in the clique of an optimal solution according to the following lemma.

▶ **Lemma 3.2.** *Let (G,k) is a positive instance of CLIQUE + IS DELETION. If F is a solution of (G,k), then the clique in $G-F$ has size at least $\sqrt{2(m-k)}$.*

Proof. Since F contains at most k edges, the graph $G-F$ has at least $m-k$ edges. Moreover, $G-F$ is a clique+IS graph, therefore all its edges are the edges of a unique clique. Therefore, the size c of the clique of $G-F$ satisfies $\binom{c}{2} \geqslant m-k$, hence $c \geqslant \sqrt{2(m-k)}$. ◀

One can prove that this first rule can be extended to obtain a linear kernel: when this rule cannot be applied, assuming that $m \geq 2k$ implies that $n = O(\sqrt{m})$. When $m = O(k^2)$, we are done, and when $m = \Omega(k^2)$, the minimum degree of the graph is $\Omega(k)$, therefore at most $O(1)$ vertices can be removed, and in that case, the existence of a solution can be tested in polynomial time.

To further reduce the size of the kernel to $O(k/\log k)$, we use the two following rules to take care of very sparse or very dense instances.

▶ **Rule 3.3** (Low degree reduction rule 2). *Let v be a vertex of degree at most $2\log k - 1$. If there is no solution F of (G,k) such that v is in the clique of $G-F$, remove v from G and decrease k by $d(v)$.*

This rule is trivially safe. Moreover, it can be performed in polynomial time. Indeed, since we consider an edge-deletion problem, if v lies in the clique K of $G-F$, then every vertex of K is adjacent to v in G, i.e. $K \subseteq N[v]$. Since the degree of v is at most $2\log k - 1$, there are at most k^2 subsets in $N[v]$. We can therefore try all of them and decide in polynomial time whether there exists a solution F of (G,k) such that v is in the clique in $G-F$.

▶ **Rule 3.4** (High degree). *If G has minimum degree $\delta(G) \geq k/(2\log k)$, solve the instance and output a trivial equivalent instance.*

Again, this rule is clearly safe. The not-so-easy part is to show that CLIQUE + IS DELETION can be decided in polynomial time when $\delta(G) \geq k/(2\log k)$.

▶ **Lemma 3.5.** *Rule 3.4 can be applied in polynomial time.*

To finish the proof of Theorem 1.1, it remains to bound the size of the reduced graph. This is the goal of the following lemma.

▶ **Lemma 3.6.** *If (G, k) is a positive instance and none of the rules can be applied, then* $|V(G)| \leq 2 \cdot \frac{k}{\log k} + 1$.

Concluding remarks. Finally, one can easily show that Rule 3.1 can be adapted for the Clique + IS Edition problem by modifying the constant. On the other hand, it seems that Rules 3.3 and 3.4 do not readily generalize to Clique + IS Edition, therefore we were not able to obtain an $O(k/\log k)$ kernel for this problem. However, it is an easy exercise to show that we can weaken them to obtain a kernel with at most k/c vertices for any possible constant $c > 1$, at the cost of a running time in $O(n^c)$.

4 Linear kernel for addition towards split graphs

The goal of this section is to prove Theorem 1.2. Since the class of split graphs is closed under complementation, it is sufficient to prove that Theorem 1.2 holds for Split Addition.

We use the structure of the input graph to detect and label vertices that will be in the clique or the independent set part of a split decomposition of a well-chosen solution. More precisely, we show that, if the instance (G, k) is positive, the labeling constructed by our algorithm satisfies that there exists a solution F of (G, k) and a split decomposition (K^*, I^*) of $G + F$ such that all the vertices labeled as "clique" (resp. "independent set") are in K^* (resp. I^*). We then prove that if (G, k) is a positive instance, then the number of unlabeled vertices at the end of the algorithm is $O(k)$. Moreover, we show that we can reduce the number of labeled vertices to $O(k)$. Combining the above yields a linear kernel.

We present our reduction rules in Section 4.1 and prove them in subsequent sections.

4.1 Labeling and reduction rules

Our algorithm keeps track of a partition (K, I, D) of $V(G)$, which corresponds to the labels of the vertices of G. The set K (resp. I) stands for the vertices already labeled "clique" (resp. "independent set") while D (for "don't know") contains the vertices that are not yet labeled. Initially, no vertex is labeled, hence $K = \emptyset, I = \emptyset$ and $D = V(G)$.

We will apply the following reduction rules, whose correction is postponed to Section 4.2.

▶ **Rule 4.1** (I-rules)**.** *Move $v \in D$ to I whenever at least one of the following holds:*
(a) *v has all of its neighbors in K,*
(b) *v is non-adjacent to at least $k + 1$ vertices of K.*
Notice that this rule applies to isolated vertices since whenever v is isolated, $N(v) = \emptyset \subseteq K$.

▶ **Rule 4.2** (K-rules)**.** *Move $v \in D$ to K whenever at least one of the following holds:*
(a) *v has a neighbor in I,*
(b) *$N(v)$ contains at least $k + 1$ non-edges,*
(c) *v dominates $K \cup D$.*

The following reduction rule simply ensures that K is a clique and I an independent set.

▶ **Rule 4.3** (Reduction rules)**.** *Apply one of the following rules as long as possible:*
(a) *if there is a non-edge e between vertices of K, then add e to $E(G)$ and decrease k by 1.*
(b) *if $k < 0$ then return a trivially negative instance.*
(c) *if there is an edge between vertices of I, then return a trivially negative instance.*

We apply these rules exhaustively, and stop when none can be applied. At each step, we remove a vertex from D or we add an edge to G. Then the algorithm stops after at most n^2 steps. Moreover, one can easily apply the rules in polynomial time. When none of the previous rules can be applied, we apply the following reduction rule.

▶ **Rule 4.4** (Unlabeling algorithm).

(a) *If K contains at least $k+1$ vertices, replace K by a set $K' = \{v'_1, \dots, v'_k\}$ of k vertices and denote by G' the resulting graph. Moreover, for each vertex $v \in D$, if v is non-adjacent to t vertices of K in G, then connect v to $v'_{t+1}, \dots, v'_k$ and do not connect it to $v'_1, \dots, v'_t$.*

(b) *Replace I by an independent set I' of size $\sqrt{2k}$ connected to K' in a complete bipartite manner, and not connected to D.*

With this rule, we can bound the number of vertices of the resulting graph by $|D|$ plus at most k vertices (for K'), plus at most $\sqrt{2k}$ vertices (for I'). Therefore, Theorem 1.2 boils down to the following lemma.

▶ **Lemma 4.5.** *If (G, k) is a positive instance, then $|D| \leq 10k + 5\sqrt{2k} + 4$.*

While it is not very difficult to prove that the reduction rules are safe, the main technical contribution of this section consists in proving Lemma 4.5. The proof is split into two parts. First, we prove that the number of vertices of D in the clique K^* of the solution is linear in k. We prove it by showing that, if too many vertices of K^* are in D, the neighborhood of many of them is not modified. And amongst them, one must be complete to $K \cup D$, a contradiction with Rule 4.2.

Arguing that the number of vertices of D in the independent set I^* of the solution is $O(k)$ is more involved. First note that if a vertex has an independent set of size larger than $O(\sqrt{k})$ in its neighborhood, it is added to K by Rule 4.2-b. Since D only contains $O(k)$ vertices in the clique, the number of vertices of D in the independent set is at most $O(k^{3/2})$. To obtain a better upper bound on the size of D, we carefully distinguish the size of the neighborhood of the vertices of $D \cap K^*$ in I^*. Very roughly, we prove that the number of vertices in $D \cap K^*$ with many neighbors in I^* is bounded by a sublinear function which permits to improve the size of the kernel. The proof of Lemma 4.5 is postponed to Section 4.3.

Lemma 4.5 together with Rule 4.4 ensure that the following reduction rule is correct, which completes the proof of Theorem 1.2:

▶ **Rule 4.6** (Final Rule). *If none of the previous rules can be applied, and the size of the instance is at least $11k + 6\sqrt{2k} + 5$, return a trivially negative instance.*

4.2 Correctness of the reduction rules

To analyze our algorithm, we study the evolution of the instance (G, k) with the partition $P = (K, I, D)$ after the application of each rule. We will refer to the tuple (G, k, P) as a *generalized instance* of Split Addition.

The following definition formalizes when a labeling of G is compatible with a solution F.

▶ **Definition 4.7.** *Let H be a graph, let F be a set of edges such that $H + F$ is a split graph, and let $P = (K, I, D)$ be a partition of $V(H)$. We say that P* is compatible *with F, and denote it $P \vDash F$, if there exists a split decomposition (K^*, I^*) of $H + F$ such that $K \subseteq K^*$ and $I \subseteq I^*$. In that case, we say that the decomposition (K^*, I^*)* witnesses *the fact $P \vDash F$.*

A generalized instance represents a graph along with a partial labeling of the vertices. Such an instance is positive when there exists a solution that is compatible with the labeling. This leads to the following definition.

▶ **Definition 4.8** (Positive generalized instance). *A generalized instance (G, k, P) is positive if there exists a solution F of (G, k) such that $P \vDash F$.*

This allows us to extend safeness properties to reduction rules operating on generalized instances. We now show that the labeling and reduction rules preserve the existence of a solution.

▶ **Lemma 4.9.** *Rules 4.1 to 4.3 are safe.*

Note that the initial labeling $P = (\emptyset, \emptyset, V(G))$ is compatible with every solution of (G, k) (if any). Therefore, by applying transitively Lemma 4.9, we get that the labeling and reduction process is safe.

We finally show that Rule 4.4 is safe.

▶ **Lemma 4.10.** *Rule 4.4 is safe.*

4.3 Structure of positive instances

This section is devoted to the proof of Lemma 4.5, restated below.

▶ **Lemma 4.5.** *If (G, k) is a positive instance, then $|D| \leq 10k + 5\sqrt{2k} + 4$.*

In what follows, we assume that the input is a positive instance and the labeling/reduction process stopped and returned a generalized instance (G, k, P). In particular, Rules 4.1 to 4.3 cannot be applied. By Lemma 4.9, we get that there exists a solution F of (G, k) such that $|F| \leq k$ and $P \vDash F$. Unrolling the definition, this means that there exists a split decomposition (K^*, I^*) of $G + F$ such that $K \subseteq K^*$ and $I \subseteq I^*$. Let $K^D = D \cap K^*$ be the set of unlabeled vertices that belong to the clique, and let $I^D = D \cap I^*$ be the set of the unlabeled vertices that belong to the independent set. For every $v \in D$, let $I_v = N(v) \cap I^D$. We give an upper bound on the cardinality of D by giving separate upper bounds on the respective cardinalities of K^D and I^D.

Before diving into the details of the proof, let us make two observations on the structure of D, that follow from the fact that the labeling rules cannot be applied.

▶ **Observation 4.11.** *For every vertex $v \in K^D$, $|I_v| \leq \sqrt{2k} + 1$.*

▶ **Observation 4.12.** *Every vertex $v \in I^D$ has a neighbor in K^D.*

We first prove that $|K^D| = O(k)$.

▶ **Lemma 4.13.** *We have $|K^D| \leq 4k$.*

Proof. Let us prove this statement by contradiction: we prove that if $|K^D| \geq 4k + 1$, then there is a vertex in D that dominates $D \cup K$, which contradicts the fact that Rule 4.2-c cannot be applied.

By assumption, K^* is a clique in $G + F$. Since F contains at most k edges, there are at most $2k$ vertices of K^D that are adjacent to edges of F. Since $|K^D| \geq 4k + 1$, there are at least $2k + 1$ vertices in K^D that dominate $K^* = K^D \cup K$. Let X denote the set of such vertices. We will now show that there is a vertex in X that also dominates I^D, that is, a vertex of K^D that dominates $K^D \cup I^D \cup K = D \cup K$. To prove the existence of this vertex, we will prove that for any vertex u in X such that $I_u \neq I^D$, there exists a vertex $v \in X$ such that $|I_v| > |I_u|$. By applying this property repeatedly, we eventually find a vertex v such that $I_v = I^D$.

Let u be a vertex of X such that $I_u \neq I^D$. Since $K^D \subseteq N[u]$ and Rule 4.2-b cannot be applied, there are at most k non-edges between I_u and K^D. Hence, these non-edges are adjacent to at most k vertices of X (X being a clique, every non-edge is incident to at most one vertex of X), and then at least $k+1$ vertices of X dominate I_u. Let X' be the subset of vertices of X that dominate $K^* \cup I_u$. Let w be a vertex of $I^D \setminus I_u$. As noted in Observation 4.12, w is adjacent to some vertex $v \in K^D$. Assume that w is anticomplete to X', so that $v \notin X'$. Since $v \in K^*$, every vertex of X' is adjacent to v. Therefore v contains at least $k+1$ non-edges in its neighborhood, namely the edges between w and X', a contradiction.

Therefore, $v \in X'$ and the conclusion follows since I_v contains I_u and w. ◀

By bounding locally the size of the neighborhood of each vertex in K^D using Observation 4.11, Lemma 4.13 directly provides an $O(k^{\frac{3}{2}})$ kernel. However, as we will show, this is not tight. Using a more global counting argument, we can show that $|I^D| = O(k)$.

▶ **Lemma 4.14.** *We have* $|I^D| \leq 6k + 5\sqrt{2k} + 4$.

Proof. First, notice that Observation 4.12 implies that $I^D \subseteq \bigcup_{v \in K^D} N(v)$. Therefore, if $|K^D| \leq \sqrt{8k}$, Observation 4.11 implies the following upper bound on the cardinality of I^D:

$$|I^D| \leq |K^D| \cdot (\sqrt{2k} + 1) \leq 4k + 2\sqrt{2k} \leq 6k + 5\sqrt{2k} + 4.$$

In what follows, we assume that $|K^D| > \sqrt{8k}$. We partition I^D into two sets: I^+, the set of vertices that have degree at least $|K^D|/4$, i.e. vertices that are adjacent to at least $|K^D|/4$ vertices of K^D, and $I^- = I^D \setminus I^+$. We bound their sizes independently.

First, by counting the number n_e of edges between K^D and I^+ from the point of view of K^D, we get $n_e \leq |K^D| \cdot (\sqrt{2k} + 1)$. From the point of view of I^+, we get $n_e \geq |K^D| \cdot |I^+|/4$. By combining the two inequalities, we get $|I^+| \leq 4(\sqrt{2k} + 1)$.

It remains to show that $|I^-| \leqslant 6k + \sqrt{2k}$. To this end, we consider two types of vertices in K^D: those that are adjacent to more than $\sqrt{2k}$ edges of F in the solution, and the others. We then bound the number of vertices in I^- adjacent to (at least) a vertex of each type.

Since we add at most k edges to G, there are at most $\sqrt{2k}$ vertices in K^D incident to more than $\sqrt{2k}$ edges of F. By Observation 4.11, these vertices of K^D have at most $\sqrt{2k}(\sqrt{2k} + 1) \leq 2k + \sqrt{2k}$ neighbors in I^D (and therefore in I^-).

To conclude the proof, it is thus sufficient to show that there are at most $4k$ vertices in I^- that are adjacent to vertices of K^D of the second type.

Let v be a vertex of K^D of the second type. We write $K_v = N(v) \cap K^D$ and $I_v^- = N(v) \cap I^-$. Observe that, by definition, $|K_v| \geqslant |K^D| - \sqrt{2k} \geq |K^D|/2$. Let $\bar{d}$ be the average degree in K_v of vertices in I_v^-. Since Rule 4.2-b cannot be applied, there are at least $|K_v| \cdot |I_v^-| - k$ edges between K_v and I_v^-, hence $\bar{d} \geqslant |K_v| - k/|I_v^-| \geqslant |K^D|/2 - k/|I_v^-|$. However, by definition of I^-, each vertex has degree at most $|K^D|/4$ in K_v hence $\bar{d} \leqslant |K^D|/4$. Combining the above yields $|I_v^-| \leqslant 4k/|K^D|$. Since there are at most $|K^D|$ vertices of the second type, the union of their neighborhoods has size at most $|K^D| \cdot 4k/|K^D| = 4k$, which is the sought result. ◀

5 Quadratic kernel for addition towards trivially perfect graphs

The goal of this section is to prove Theorem 1.3. Recall that trivially perfect graphs are (C_4, P_4)-free graphs. In what follows, we refer to induced P_4 or C_4 of a graph as its *obstructions.* We say that a pair (u, v) of vertices is a *diagonal* if $uv \notin E$ and there exists two vertices a, b such that $uavb$ is a P_4 or a C_4. Given a diagonal (u, v), the number of obstructions containing (u, v) is the number of distinct pairs a, b such that $uavb$ is a P_4 or a C_4. Note that every obstruction contains exactly two diagonals and any solution must contain at least one of the two diagonals of each obstruction.

We first present a reduction rule that should be applied exhaustively, and then two reduction rules that should be applied once.

▶ **Rule 5.1.** *Let u, v be two non-adjacent vertices. If the number of obstructions containing u, v is at least $k + 1$, then add uv to E and decrease k by 1.*

▶ **Lemma 5.2.** *Rule 5.1 is safe.*

Moreover, Rule 5.1 can easily be applied in polynomial time.

The *modulator* $X(G)$ of G is the subset of vertices of G that are in at least one obstruction. Guo [21, Theorem 4] stated that (G, k) is a positive instance if and only if $(G[X(G)], k)$ is. Therefore, the following reduction rule is safe:

▶ **Rule 5.3.** *If $X(G) \neq V(G)$, remove all vertices of G that are not in $X(G)$.*

When the first two rules cannot be applied, we perform the following rule which detects trivially negative instances.

▶ **Rule 5.4.** *If $|V(G)| > 2k^2 + 2k$, output a trivially negative instance.*

To complete our proof, we simply have to prove that after applying the first two rules exhaustively, the size of a positive instance is quadratic. The next lemma ensures that Rule 5.4 is safe, which concludes the proof of Theorem 1.3.

▶ **Lemma 5.5.** *If (G, k) is a positive instance and every diagonal belongs to at most k obstructions, then $|X(G)| \leq 2k^2 + 2k$.*

6 Linear kernel for Starforest deletion

The goal of this section is to prove Theorem 1.4. Stars can be divided into two sets: centers and leaves. Let us define the notion of *center set* of a star forest.

▶ **Definition 6.1** (Center set). *Let $\mathbb{S}$ be a star-forest. A set $C^* \subseteq V(\mathbb{S})$ is a* center set *of $\mathbb{S}$ if C^* is a dominating set of $\mathbb{S}$ such that every star S of $\mathbb{S}$ contains exactly one vertex c of C^*. This vertex is called the center of S.*

Note that a center set is not necessarily unique since, in 2-stars, both vertices can be selected as a center. Given a star forest $\mathbb{S}$ with a set of centers C^*, the *leaves* of $\mathbb{S}$ are the vertices outside of C^*. By definition, every leaf has degree 1 and its unique neighbor is in C^*.

In what follows, we show how to use the structure of the input graph to identify and label vertices that are centers of an optimal solution, which leads to a Label-And-Reduce kernelization algorithm.

Let (G, k) be an instance of Starforest Deletion. Our first reduction rule, which is indeed safe, removes trivial connected components.

▶ **Rule 6.2** (Clean-up rule). *Remove from G any connected component with 1 or 2 vertices.*

Assume now that Rule 6.2 cannot be applied anymore.

▶ **Rule 6.3** (Center labeling rule). *Let C be the set of vertices of G that are adjacent to a vertex of degree 1 in G.*

(a) *For every $v \notin C$, if v is adjacent to a vertex u of C, delete all the other edges between v and C, and decrease the parameter accordingly.*

(b) *For every $u, v \in C$, if u and v are adjacent then remove uv from G and decrease k by 1.*

The fact that Rule 6.3 is safe is a consequence of the following lemma:

▶ **Lemma 6.4.** *Let C be the set of vertices of G that are adjacent to a vertex of degree 1. If (G, k) is a positive instance, then there exists a solution F of (G, k) and a center-set C^* of $G - F$ such that $C \subseteq C^*$.*

Lemma 6.4 ensures that Rule 6.3 is safe. Indeed, if there exists a solution, then there is also a solution where C is in the center-set. Hence we can safely remove all the edges between the vertices of C since each star only contains one vertex of the center-set of $G - F$. Moreover, if an edge between v and a vertex w of C is kept in $G - F$, then we can choose to keep any other edge between v and C instead of vw, since all the vertices of C are centers of their stars.

When neither Rule 6.2 nor Rule 6.3 can be applied, we apply the following rule:

▶ **Rule 6.5** (Center reduction rule). *Merge all the vertices of C, and remove all but $k + 2$ vertices of degree 1.*

▶ **Lemma 6.6.** *Rule 6.5 is safe.*

When Rules 6.2 to 6.5 cannot be applied, we apply the following rule once.

▶ **Rule 6.7** (Kernel size rule). *If $|V(G)| > 4k + 3$, return a trivial negative instance (e.g. $(P_4, 0)$). Otherwise, return (G, k).*

Rule 6.7 ensures that the returned kernel has at most $4k + 3$ vertices. In the remainder of this section, we study the structure of positive instances of Starforest Deletion to prove that Rule 6.7 is safe.

In the two following lemmas, we assume that none of Rules 6.2 to 6.5 can be applied. The following lemma uses the sparsity of starforests (they have many vertices of degree 1) to get information on the structure of positive instances.

▶ **Lemma 6.8.** *If (G, k) is a positive instance of Starforest Deletion with m edges, then G contains at least $m - 3k$ vertices of degree 1.*

In the last step of Rule 6.3, we remove all but $k + 2$ vertices of degree 1. In the following lemma, we apply Lemma 6.8 to show that the number of remaining vertices must be small.

▶ **Lemma 6.9.** *If (G, k) is a positive instance where no rule can be applied, then $|V(G)| \leq 4k + 3$.*

By applying the contrapositive of Lemma 6.9, we get that Rule 6.7 is safe.

Improving the multiplicative constant in the linear bound. In the proof of Lemma 6.9, we use a simple argument based on the minimum degree to show that the $3k$ remaining edges span at most $3k + 1$ vertices. The worst case is when every vertex has degree 2, that is, when every connected component is a cycle. In a cycle, an optimal solution can easily be found in polynomial time, and therefore we can remove cycles. We can also show that long induced paths can be reduced. Combining these results gives a smaller kernel, at the cost of an increased running time and a slightly more involved analysis.

However, these improvements do not yield a sublinear kernel. It turns out that, under the Exponential Time Hypothesis, Starforest Deletion does not have a sublinear kernel. Indeed, Drange et al. [15] proved that, under ETH, Starforest Deletion does not admit a subexponential FPT algorithm, i.e. an algorithm running in time $O^*(2^{o(k)})$. Moreover, there is an $O^*(2^n)$ algorithm for Starforest Deletion: for each subset S of vertices, test whether there exists a solution in which S is the center set. Therefore, a kernel with $o(k)$ vertices would imply an $O^*(2^{o(k)})$ algorithm; a contradiction.

References

1 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1):89–113, 2004.

2 Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of computational biology*, 6(3-4):281–297, 1999.

3 Ivan Bliznets, Marek Cygan, Pawel Komosa, Lukáš Mach, and Michał Pilipczuk. Lower bounds for the parameterized complexity of minimum fill-in and other completion problems. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1132–1151. SIAM, 2016.

4 Sebastian Böcker. A golden ratio parameterized algorithm for cluster editing. *Journal of Discrete Algorithms*, 16:79–89, 2012.

5 Pablo Burzyn, Flavia Bonomo, and Guillermo Durán. Np-completeness results for edge modification problems. *Discrete Applied Mathematics*, 154(13):1824–1844, 2006.

6 Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.

7 Yixin Cao and Jianer Chen. Cluster editing: Kernelization based on edge cuts. *Algorithmica*, 64(1):152–169, 2012.

8 Yixin Cao and Yuping Ke. Improved kernels for edge modification problems. *arXiv preprint arXiv:2104.14510*, 2021.

9 Jianer Chen and Jie Meng. A 2k kernel for the cluster editing problem. *Journal of Computer and System Sciences*, 78(1):211–220, 2012.

10 Christophe Crespelle, Pål Grønås Drange, Fedor V Fomin, and Petr A Golovach. A survey of parameterized algorithms and the complexity of edge modification. *arXiv preprint arXiv:2001.06867*, 2020.

11 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.

12 Peter Damaschke and Olof Mogren. Editing the simplest graphs. In *International Workshop on Algorithms and Computation*, pages 249–260. Springer, 2014.

13 Pål Grønås Drange, Fedor V Fomin, Michał Pilipczuk, and Yngve Villanger. Exploring the subexponential complexity of completion problems. *ACM Transactions on Computation Theory (TOCT)*, 7(4):1–38, 2015.

14 Pål Grønås Drange and Michał Pilipczuk. A polynomial kernel for trivially perfect editing. *Algorithmica*, 80(12):3481–3524, 2018.

15 Pål Grønås Drange, Felix Reidl, Fernando Sánchez Villaamil, and Somnath Sikdar. Fast biclustering by dual parameterization. *arXiv preprint arXiv:1507.08158*, 2015.

16 Mael Dumas, Anthony Perez, and Ioan Todinca. A cubic kernel for trivially perfect edition. *private communication.*

17 Fedor V Fomin, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Yngve Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *Journal of Computer and System Sciences*, 80(7):1430–1447, 2014.

18 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

19 Esha Ghosh, Sudeshna Kolay, Mrinal Kumar, Pranabendu Misra, Fahad Panolan, Ashutosh Rai, and MS Ramanujan. Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, 71(4):989–1006, 2015.

20 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory of Computing Systems*, 38(4):373–392, 2005.

21 Jiong Guo. Problem kernels for np-complete edge deletion problems: Split and related graphs. In *International Symposium on Algorithms and Computation*, pages 915–926. Springer, 2007.

22 Peter L Hammer and Bruno Simeone. The splittance of a graph. *Combinatorica*, 1(3):275–284, 1981.

23 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

24 Yunlong Liu, Jianxin Wang, Jie You, Jianer Chen, and Yixin Cao. Edge deletion problems: Branching facilitated by modular decomposition. *Theoretical Computer Science*, 573:63–70, 2015.

25 Federico Mancini. Graph modification problems related to graph classes. *PhD degree dissertation, University of Bergen Norway*, 2, 2008.

26 Assaf Natanzon, Ron Shamir, and Roded Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109–128, 2001.

27 René van Bevern, Vincent Froese, and Christian Komusiewicz. Parameterizing edge modification problems above lower bounds. *Theory of Computing Systems*, 62(3):739–770, 2018.

28 Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11):1101–1113, 1993.

29 Mihalis Yannakakis. Node-and edge-deletion np-complete problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 253–264, 1978.