Improved Kernels for Edge Modification Problems

Yixin Cao
Department of Computing, Hong Kong Polytechnic University, Hong Kong, China

Yuping Ke
Department of Computing, Hong Kong Polytechnic University, Hong Kong, China

Abstract

In an edge modification problem, we are asked to modify at most k edges of a given graph to make the graph satisfy a certain property. Depending on the operations allowed, we have the completion problems and the edge deletion problems. A great amount of efforts have been devoted to understanding the kernelization complexity of these problems. We revisit several well-studied edge modification problems, and develop improved kernels for them:

- a 2k-vertex kernel for the cluster edge deletion problem,
- a $3k^2$-vertex kernel for the trivially perfect completion problem,
- a $5k^{1.5}$-vertex kernel for the split completion problem and the split edge deletion problem, and
- a $5k^{1.5}$-vertex kernel for the pseudo-split completion problem and the pseudo-split edge deletion problem.

Moreover, our kernels for split completion and pseudo-split completion have only $O(k^{2.5})$ edges. Our results also include a 2k-vertex kernel for the strong triadic closure problem, which is related to cluster edge deletion.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Kernelization, edge modification, cluster, trivially perfect graphs, split graphs

Digital Object Identifier 10.4230/LIPIcs.IPEC.2021.13

Related Version *Full Version*: `https://arxiv.org/abs/2104.14510`

Funding Supported by RGC grants 15201317 and 15226116, and NSFC grant 61972330.

1 Introduction

In an edge modification problem, we are asked to modify at most k edges of a given graph G to make the graph satisfy a certain property. In particular, we have edge deletion problems and completion problems when the allowed operations are edge deletions and, respectively, edge additions. There is also a more general version that allows both operations. The present paper will be focused on a single type of modifications. For most graph properties, these edge modification problems are known to be NP-complete [21, 18, 15]. A graph G having a certain property is equivalent to that G belongs to some specific graph class. Cai [2] observed that if the desired graph class can be characterized by a finite number of forbidden induced subgraphs, then these problems are fixed-parameter tractable.

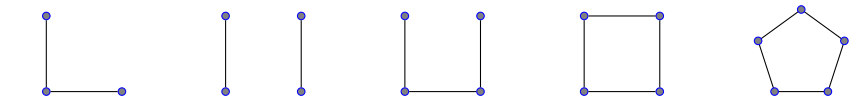
One is then naturally interested in the kernelization complexity of edge modification problems toward these *easy* graph classes. Given an instance (G, k), a *k*ernelization algorithm produces in polynomial time an equivalent instance (G', k') – (G, k) is a yes-instance if and only if (G', k') is a yes-instance – such that $k' \leqslant k$ and the size of G' is bounded by a computable function of k. The output instance (G', k') is a *polynomial kernel* if the size of G' is bounded from above by a polynomial function of k'. Although progress has been made in this regard, we get stuck for several important graph classes. We have evidence that some of them do not have polynomial kernels, under certain complexity assumptions, and it is believed that those that do have are exceptions [16]. This makes a sharp contrast with the vertex deletion problems (deleting vertices instead of edges), for which a polynomial kernel

is guaranteed when the number of forbidden induced subgraphs is finite [9]. We refer the reader to the recent survey of Crespelle et al. [4], particularly its Section 2.1 and Table 1, for the most relevant results.

We revisit several well-studied edge modification problems, and develop improved kernels for them. Our results are summarized in Table 1. All the target graph classes can be defined by a small number of forbidden induced subgraphs (listed in Figure 1). It is worth mentioning that the edge deletion problem to a graph class is polynomially equivalent to the completion problem to its complement graph class (consisting of the complements of all graphs in the original graph class). Moreover, some graph classes, e.g., split graphs ($\{2K_2, C_4, C_5\}$-free), are self-complementary, and thus the edge deletion problem and the completion problem toward such a class are equivalent.

Table 1 Main results of this paper, shown as the number of vertices in the kernels.

problem	previous result	our result
cluster edge deletion	$4k$ [11]	$2k$
trivially perfect completion	$O(k^7)$ [7]	$3k^2$
split completion (edge deletion)	$O(k^2)$ [10]	$5k^{1.5}$
pseudo-split completion (edge deletion)	-	$5k^{1.5}$
strong triadic closure	$4k$ [11]	$2k$



(a) P_3. **(b)** $2K_2$. **(c)** P_4. **(d)** C_4. **(e)** C_5.

Figure 1 Forbidden induced graphs. Note that $2K_2$ and C_4 are complements of each other, while the complements of P_4 and C_5 are themselves.

A cluster graph is a disjoint union of cliques. Since cluster graphs are precisely P_3-free graphs, edge modification problems to cluster graphs are the simplest of all nontrivial edge modification problems. Note that edge modification problems toward P_2-free graphs, i.e., edgeless graphs, are trivial. Also trivial is the cluster completion problem: the minimum solution is to add edges to make every component of the input graph complete. Both cluster edge editing and cluster edge deletion are NP-complete and have received wide attentions. After a sequence of results, Cao and Chen [3] devised a $2k$-vertex kernel for the cluster edge editing problem. Their algorithm actually implies a $2k$-vertex kernel for the cluster edge deletion problem. We record this simple result here for future reference. Less trivially, we show that the same algorithm produces a kernel of the same size for the strong triadic closure problem, which, though originally not posed as an edge modification problem, is closely related to cluster edge deletion [14]. As the original results [3], both algorithms work for the weighted versions of the problems as well.

The second problem is the trivially perfect completion problem. Drange and Pilipczuk [7] presented an $O(k^7)$-vertex kernel for this problem.[1] We propose a very simple kernelization algorithm, which has only two simple reduction rules, and the resulting kernel contains at

[1] Guo [12] has claimed an $O(k^3)$-vertex kernel for this problem, with details deferred to a full version that has never appeared. The algorithm of Drange and Pilipczuk [7] works for the more general trivially perfect editing problem. Independent to our work, Dumas et al. [8] improved it to $O(k^3)$, which also implies an $O(k^3)$-vertex kernel for the trivially perfect completion problem.

most $2k^2 + 2k$ vertices. The forbidden induced subgraphs of trivially perfect graphs are P_4 and C_4. Note that adding the edge to connect the two ends of a P_4 merely turns it into a C_4. Thus, in each P_4 or C_4, there are two missing edges such that every solution needs to contain at least one of them. Note that each vertex of the P_4 or C_4 is an end of one of the two missing edges. Our first rule is the most routine for this kind of problems, namely, adding a missing edge if it is one of the two possible missing edges in $k + 1$ or more P_4's and C_4's. Our second rule removes all vertices that are not contained in any P_4 or C_4 of G. Now the analysis is similar as Buss and Goldsmith's kernelization algorithm for the vertex cover problem [1]. Since every solution contains at least one of the pair of potential missing edges (for some P_4 or C_4), and since each potential edge is in at most k pairs, there cannot be more than $k^2 + k$ potential edges in a yes-instance. On the other hand, every vertex is in a P_4 or C_4, hence an end of some potential edge. We are thus safe to return a trivial no-instance when $|V(G)| > 2k^2 + 2k$. Toward this result we also obtain some nontrivial observations on minimal solutions of the problem with respect to modules of the input graph.

A graph is a split graph if its vertex set can be partitioned into a clique and an independent set. Split graphs are $\{2K_2, C_4, C_5\}$-free graphs. The split completion problem, which is equivalent to split edge deletion, is NP-complete [17], while somewhat surprisingly, the split edge editing problem can be solved in polynomial time [13]. Guo [12] presented an $O(k^4)$-vertex kernel for the split completion problem, which was improved to $O(k^2)$ by Ghosh et al. [10]. For the convenience of presentation, we work on the edge deletion problem. We consider the partition of the vertex set after applying an optimal solution. We observe that for most of the vertices we know to which side they have to belong. It is nevertheless not safe to directly delete these "decided" vertices. We thus work on the annotated version, where we mark certain vertices that have to be in the independent set. Guo [12] has proved that it is safe to remove a vertex that is not contained in any $2K_2$, C_4, or C_5. We show that a similar rule can be applied to annotated instances, and after its application, there can be at most $O(k^{1.5})$ vertices in a yes-instance. Finally, a simple step that removes the marks concludes the algorithm. Our kernel for split completion has only $O(k^{2.5})$ edges.[2] With minor tweaks, our algorithm produces a kernel of the same size for the pseudo-split ($\{2K_2, C_4\}$-free graphs) edge deletion problem. A pseudo-split graph is either a split graph or a split graph plus a C_5 such that every vertex on the C_5 is adjacent to every vertex in the clique part of the split graph and is nonadjacent to any vertex in the independent part of the split graph. The first difficulty toward this adaptation is that it is not always safe to remove vertices not contained in any $2K_2$ or C_4. We get over this obstacle by observing that we can remove vertices not contained in any $2K_2$, C_4, or C_5. As we recycle the reduction rules for split edge deletion, only the arguments for their safeness need to be slightly revised. Drange et al. [6] have used a similar approach to develop a subexponential-time algorithm for the pseudo-split completion problem.

2 Preliminaries

All graphs discussed in this paper are undirected and simple. The vertex set and edge set of a graph G are denoted by, respectively, $V(G)$ and $E(G)$. For a subset $U \subseteq V(G)$, denote by $G[U]$ the subgraph of G induced by U, and by $G - U$ the subgraph $G[V(G) \setminus U]$, which is further shortened to $G - v$ when $U = \{v\}$. The *neighborhood* of a vertex v in G, denoted

[2] Independent to our work, Bathie, Bousquet, and Pierron (arXiv:2105.09566) use a similar technique to obtain a linear-vertex kernel for these problems.

by $N_G(v)$, comprises vertices adjacent to v, i.e., $N_G(v) = \{u \mid uv \in E(G)\}$, and the *closed neighborhood* of v is $N_G[v] = N_G(v) \cup \{v\}$. The *closed neighborhood* and the *neighborhood* of a set $U \subseteq V(G)$ of vertices are defined as $N_G[U] = \bigcup_{v \in U} N_G[v]$ and $N_G(U) = N_G[U] \setminus U$, respectively. We may omit the subscript when there is no ambiguity on the graph under discussion. Two vertices u and v are true twins in G if $N[u] = N[v]$; note that true twins are necessarily adjacent. A *clique* is a set of pairwise adjacent vertices, and an *independent set* is a set of pairwise nonadjacent vertices. A graph G is *complete* if $V(G)$ is a clique. A vertex v is *simplicial* if $N[v]$ is a clique, and a vertex v is *universal* if $N[v] = V(G)$. An induced path and an induced cycle on ℓ vertices are denoted by P_ℓ and C_ℓ respectively.

For any two subsets $X, Y \subseteq V(G)$, we use $E(X, Y)$ to denote the set of edges of which one endpoint is in X and the other in Y. Note that we do not require X and Y to be disjoint. Thus, $E(X, X) = E(G[X])$, i.e., all the edges with both endpoints in X, and $E(X, V(G))$ consists of all the edges with at least one endpoint in X.

Let F be a fixed graph. We say that a graph G is F-*free* if G does not contain F as an induced subgraph. For a set $\mathcal{F}$ of graphs, a graph G is $\mathcal{F}$-*free* if G is F-free for every $F \in \mathcal{F}$. If every $F \in \mathcal{F}$ is minimal, i.e., not containing any $F' \in \mathcal{F}$ as a proper induced subgraph, then the set $\mathcal{F}$ of graphs are the (minimal) *forbidden induced subgraphs* of this class. See Figure 1 for the forbidden induced subgraphs considered in the present paper. For a set E' of non-edges, we denote by $G + E'$ the graph with vertex set $V(G)$ and edge set $E(G) \cup E'$; for a set $E' \subseteq E(G)$, we denote by $G - E'$ the graph with vertex set $V(G)$ and edge set $E(G) \setminus E'$. The problems to be studied are formally defined as follows, where $\mathcal{G}$ is a graph class.

$\mathcal{G}$ completion

Input: A graph G and a nonnegative integer k.
Output: Is there a set E_+ of at most k edges such that $G + E_+$ is in $\mathcal{G}$?

$\mathcal{G}$ edge deletion

Input: A graph G and a nonnegative integer k.
Output: Is there a set E_- of at most k edges such that $G - E_-$ is in $\mathcal{G}$?

Since it is always clear from the context what problem we are talking about, when we mention an instance (G, k), we do not always explicitly specify the problem. We use $\mathrm{opt}(G)$ to denote the size of optimal solutions of G for the optimization version of a certain problem. Thus, (G, k) is a yes-instance if and only if $\mathrm{opt}(G) \leqslant k$.

For each problem, we apply a sequence of reduction rules. Each rule transforms an instance (G, k) to a new instance (G', k'). We say that a rule is *safe* if (G, k) is a yes-instance if and only if (G', k') is a yes-instance. Since all of our reduction rules are very simple and most of them are obviously doable in polynomial time, we omit the details of their implementation and analyze their running time only when it is nontrivial.

3 Cluster edge deletion and strong triadic closure

A graph is a cluster graph if every component of this graph is a complete subgraph. It is well known that a graph is a cluster graph if and only if it is P_3-free. Our first problem is the cluster edge deletion problem. For a vertex set $U \subseteq V(G)$, we write $d(U) = |E(U, V(G) \setminus U)|$, i.e., the number of edges between U and $V(G) \setminus U$; we write $d(v)$ instead of $d(\{v\})$ for a singleton set.

▶ **Rule 3.1.** *If there is a simplicial vertex* v *such that* $d(N[v]) \leqslant d(v)$*, then remove* $N[v]$ *and decrease* k *by* $d(N[v])$*.*

Safeness of Rule 3.1. We show that $\mathrm{opt}(G) = \mathrm{opt}(G - N[v]) + d(N[v])$. Let E_- be an optimal solution to the graph G. We have nothing to show if $N[v]$ is a separate component of $G - E_-$. In the rest of the proof, $N[v]$ is not a component of $G - E_-$. Let $G[X]$ denote the component of $G - E_-$ that contains v. Since X is a clique and $N[v] \neq X$, we have $X \subset N[v]$. In other words, neither X nor $N[v] \setminus X$ is empty. Since any induced subgraph of $G - E_-$ is a cluster graph, the subset of edges in E_- with both endpoints in $V(G) \setminus N[v]$ is a solution to $G - N[v]$. Noting that this solution is disjoint from $E(X, V(G) \setminus X)$, we have

$$\begin{aligned}\mathrm{opt}(G) \geqslant & |E_- \cap E(G - N[v])| + d(X)\\ \geqslant & \mathrm{opt}(G - N[v]) + |X| \cdot |N[v] \setminus X|\\ \geqslant & \mathrm{opt}(G - N[v]) + |X| + |N[v] \setminus X| - 1 \qquad (1)\\ = & \mathrm{opt}(G - N[v]) + d(v),\end{aligned}$$

where the third inequality holds because both $|X|$ and $|N[v] \setminus X|$ are positive integers. For any solution E'_- of $G - N[v]$, the set $E'_- \cup E(N[v], V(G) \setminus N[v])$ is a solution of G. Thus,

$$\mathrm{opt}(G) \leqslant \mathrm{opt}(G - N[v]) + d(N[v]) \leqslant \mathrm{opt}(G - N[v]) + d(v). \qquad (2)$$

Therefore, all the inequalities in (1) and (2) are tight. In other words, if we remove all the edges between $N[v]$ and $V(G) \setminus N[v]$, and then delete an optimal solution to $G - N[v]$, then we have an optimal solution to the graph G. ◀

A trivial but crucial fact is that a solution E_- has at most $2|E_-|$ endpoints. If a vertex v is not an end of any edge in E_-, then v has to be simplicial.

▶ **Theorem 1.** *There is a* $2k$*-vertex kernel for the cluster edge deletion problem.*

Proof. Let G be a graph to which Rule 3.1 is not applicable. We show that if (G, k) is a yes-instance, then $|V(G)| \leqslant 2k$. Let E_- be an optimal solution to G, and let $\{v_1, v_2, \ldots, v_r\}$ be the vertices that are not incident to any edge in E_-; they have to be simplicial. For $i = 1, \ldots, r$, the set $N[v_i]$ forms a component of $G - E_-$. Note that for distinct $i, j \in \{1, \ldots, r\}$, the sets $N[v_i]$ and $N[v_j]$ are either the same (when v_i and v_j are true twins) or mutually disjoint: if $N[v_i] \neq N[v_j]$ and there exists $x \in N[v_i] \cap N[v_j]$, then one of xv_i and xv_j needs to be in E_-. We divide the cost of each edge $uv \in E_-$ and assign them to u and v equally. For $i = 1, \ldots, r$, the total cost attributed to all the vertices in $N[v_i]$ is $d(N[v_i])/2$. Each of the vertices not in $\bigcup_{i=1}^{r} N[v_i]$ is an end of at least one edge in E_- and therefore bears cost at least $1/2$. Summing them up, we get a lower bound for the total cost:

$$\begin{aligned}|E_-| \geqslant & \frac{1}{2}\sum_{i=1}^{r} d(N[v_i]) + \frac{1}{2}|V(G) \setminus \bigcup_{i=1}^{r} N[v_i]|\\ \geqslant & \frac{1}{2}\sum_{i=1}^{r} |N[v_i]| + \frac{1}{2}|V(G) \setminus \bigcup_{i=1}^{r} N[v_i]|\\ \geqslant & \frac{1}{2}|V(G)|.\end{aligned}$$

The second inequality holds because Rule 3.1 does not apply to v_i for $i = 1, \ldots, r$. Thus, $|V(G)|/2 \leqslant |E_-| \leqslant k$ for a yes-instance, and we can return a trivial no-instance if $|V(G)| > 2k$. This concludes the proof. ◀

Let us mention that the condition of Rule 3.1 can be weakened to $d(N[v]) < 2d(v) - 1$. We do not prove the stronger statement because it does not improve the analysis of the kernel size, but let us briefly explain why it is true. The bound $\mathrm{opt}(G) \geqslant \mathrm{opt}(G - N[v]) + 2d(v) - 1$ holds unless $|X| = 1$ or $|N[v] \setminus X| = 1$; see the third inequality of (1). In the first case, v itself makes a trivial component, and all the vertices in $N(v)$ are in the same component; this can only happen when there exists another vertex u with $N(v) \subseteq N(u)$. In the second case, a vertex $u \in N(v)$ is incident to all the edges between $N(v)$ and $V(G) \setminus N[v]$. If $d(N[v]) < 2d(v) - 1$, then $\mathrm{opt}(G) \geqslant \mathrm{opt}(G - N[v]) + 2d(v) - 1$ holds in both cases.

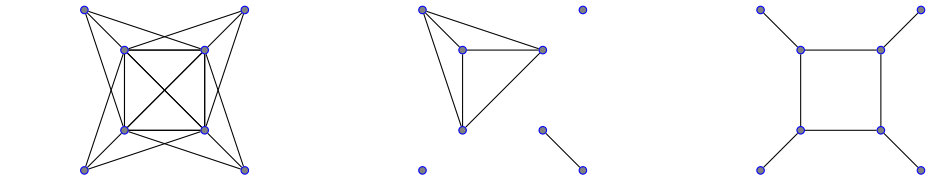


Figure 2 The example given by Konstantinidis et al. [14]: (a) the input graph; (b) a maximum cluster subgraph with seven edges; and (c) a maximum strong triadic closure with eight edges.

In the original definition, which was motivated by applications in social networks, the *strong triadic closure* problem asks for a partition of the edge set of the input graph into strong edges and weak ones, such that for every two vertices that are linked to a common neighbor with strong edges are adjacent. The objective is to maximize the number of strong edges. For our purpose, it is more convenient to define the problem as follows.

Strong triadic closure

Input:	A graph G and a nonnegative integer k.
Output:	Is there a set E_- of at most k edges such that the missing edge of every P_3 of $G - E_-$ is in $E(G)$?

Thus, we call the set of weak edges as the solution to the strong triadic closure problem. For any set $E_- \subseteq E(G)$, if $G - E_-$ is a cluster graph, then E_- is also a solution to the strong triadic closure problem: setting all edges in E_- weak, and all other edges strong is a feasible partition of $E(G)$. As illustration in Figure 2, however, a strong triadic closure of a graph can have fewer weak edges than an optimal solution to the cluster edge deletion problem on the same graph. Surprisingly, Rule 3.1 works for the strong triadic closure problem without change.

▶ **Lemma 2.** *Rule 3.1 is safe for the strong triadic closure problem.*

Proof. We show that $\mathrm{opt}(G) = \mathrm{opt}(G - N[v]) + d(N[v])$. Let E_- be an optimal solution to the graph G. We have nothing to show if $N[v]$ is a separate component of $G - E_-$. In the rest of the proof, $N[v]$ is not a component of $G - E_-$. Let X denote the set of vertices with $N[X] = N[v]$, and $Y \subseteq N[v]$ the endpoints of these edges in $E(N[v], V(G) \setminus N[v]) \setminus E_-$ (i.e., edges between $N[v]$ and $V(G) \setminus N[v]$ that are not in E_-). Note that $X \neq \emptyset$ because $v \in X$, and $Y \neq \emptyset$ because $E(N[v], V(G) \setminus N[v]) \not\subseteq E_-$ (otherwise $N[v]$ is a component of $G - E_-$ by the minimality of E_-).

By definition, the subset of edges in E_- with both endpoints in $G - N[v]$ is a solution to $G - N[v]$. By the selection of X and Y, every vertex in $N[v] \setminus (X \cup Y)$ is incident to at least one edge in $E_- \cap E(N[v], V(G) \setminus N[v])$. For every $x \in X$ and every $y \in Y$, there exists

$z \in V(G) \setminus N[v]$ that is adjacent to y but not x; hence, xyz is a P_3. As a result, all the edges between X and Y have to be in E_-. Thus,

$$\begin{aligned}\mathrm{opt}(G) =& |E_- \cap E(G - N[v])| + |E_- \cap E(N[v], V(G) \setminus N[v])| + |E_- \cap E(N[v])| \\ \geqslant& \mathrm{opt}(G - N[v]) + |N[v] \setminus (X \cup Y)| + |X| \cdot |Y| \\ \geqslant& \mathrm{opt}(G - N[v]) + |N[v]| - |X| - |Y| + |X| + |Y| - 1 \\ \geqslant& \mathrm{opt}(G - N[v]) + |N[v]| - 1 \\ =& \mathrm{opt}(G - N[v]) + d(v),\end{aligned} \tag{3}$$

where $|X| \cdot |Y| \geqslant |X| + |Y| - 1$ because both $|X|$ and $|Y|$ are positive integers. For any solution E'_- of $G - N[v]$, the set $E'_- \cup E(N[v], V(G) \setminus N[v])$ is a solution of G. Thus,

$$\mathrm{opt}(G) \leqslant \mathrm{opt}(G - N[v]) + d(N[v]) \leqslant \mathrm{opt}(G - N[v]) + d(v). \tag{4}$$

Therefore, all the inequalities in (3) and (4) are tight. In other words, if we remove all the edges between $N[v]$ and $V(G) \setminus N[v]$, and then delete an optimal solution to $G - N[v]$, then we have an optimal solution to the graph G. ◀

The proof of the following theorem is a word-for-word copy of that for Theorem 1, hence omitted.

▶ **Theorem 3.** *There is a* $2k$*-vertex kernel for the strong triadic closure problem.*

We should remark that our kernelization algorithms for the cluster edge deletion problem and the strong triadic closure problem work for the weighted versions as well; see [3].

4 Trivially perfect completion

In this section we study the trivially perfect completion problem. Trivially perfect graphs are $\{P_4, C_4\}$-free graphs. If there is a pair of adjacent vertices u, v such that neither $N[u] \setminus N[v]$ nor $N[v] \setminus N[u]$ is empty, then they are contained in a P_4 or C_4. Trivially perfect graphs have many nice characterizations. Here are two of them.

▶ **Theorem 4** ([19, 20])**.** *The following are equivalent for a graph* H*.*

- **i)** H *is a trivially perfect graph.*
- **ii)** *Every connected induced subgraph of* H *contains a universal vertex.*
- **iii)** *For every pair of adjacent vertices* u *and* v*, one of* $N[u]$ *and* $N[v]$ *is a subset of the other.*

If a vertex v is not contained in any P_4 or C_4, then for every neighbor u of v, one of $N[u]$ and $N[v]$ is a subset of the other.

▶ **Lemma 5** ($\star$[3])**.** *If a vertex* v *is not contained in any* P_4 *or* C_4*, then* $\mathrm{opt}(G - v) = \mathrm{opt}(G)$*.*

As a simple result of Lemma 5, we have the following reduction rule (which was mentioned by Guo [12], without a proof). In particular, all universal vertices of every component of G can be removed.

▶ **Rule 4.1.** *If there is a vertex* v *that is not contained in any* P_4 *or* C_4*, then remove* v*.*

[3] Proofs of propositions marked with $\star$ are deferred to the full version.

For each induced 4-path or 4-cycle $v_1v_2v_3v_4$, we call the missing edges $\{v_1, v_3\}$ and $\{v_2, v_4\}$ the *candidate edges* for this path or cycle. Clearly, any solution of a graph G contains at least one candidate edge of every P_4 or C_4; note that a P_4 has another missing edge, the addition of which merely turns the P_4 into a C_4.

▶ **Rule 4.2.** *If uv is a candidate edge of $k+1$ or more P_4's and C_4's in G, then add the edge uv and decrease k by one.*

Safeness of Rule 4.2. Since each P_4 or C_4 of G has precisely two candidate edges, if a solution E_+ of G does not contain uv, then E_+ must contain the other candidate edge of each of the $k+1$ P_4's and C_4's, hence $|E_+| > k$. ◀

We are thus ready for the main result of this section.

▶ **Theorem 6.** *There is a $(2k^2+2k)$-vertex kernel for the trivially perfect completion problem.*

Proof. After applying Rule 4.2 and then Rule 4.1 exhaustively, we return (G, k) if $|V(G)| \leqslant 2k^2 + 2k$, or a trivial no-instance otherwise. We consider all the candidate edges of G. We say that two candidate edges are associated if they belong to the same P_4 or C_4; i.e., their endpoints are disjoint and together induce a P_4 or C_4. Since Rule 4.2 is not applicable, each candidate edge is associated with at most k candidate edges. On the other hand, of any two associated edges, one has to be in any solution of G. Thus, if (G, k) is a yes-instance, there can be at most $k^2 + k$ candidate edges. Since Rule 4.1 is not applicable, every vertex is in some P_4 or C_4, and hence is an end of a candidate edge. Thus, $|V(G)| \leqslant 2k^2 + 2k$ if (G, k) is a yes-instance. ◀

The analysis of the kernel in Theorem 6 is essentially the same as Buss and Goldsmith's kernelization algorithm for the vertex cover problem [1]. In a sense, we are looking for a vertex cover of an auxiliary graph in which each vertex corresponds to a candidate edge of G, and two vertices are adjacent if their corresponding edges are associated. We note that the same approach implies a simple $O(k^2)$-vertex kernel for the threshold completion problem, matching the result of Drange et al. [5]. The forbidden induced subgraphs of threshold graphs are $2K_2$, P_4, and C_4. The observation on the missing edges of a P_4 or C_4 is the same as above, while the four missing edges of a $2K_2$ can be organized as two pairs such that each solution has to contain at least one from each pair. However, we are not able to employ the $2k$-vertex kernels for vertex cover to directly derive a linear-vertex kernel for either of the two problems.

Before closing this section, let us mention some observations that might be of independent interest. A set M of vertices is a module if $N(M) = N(v) \setminus M$ for every $v \in M$.

▶ **Lemma 7** ($\star$). *A module M of a graph G remains a module in any minimal trivially perfect completion $\widehat{G}$ of G.*

5 Split edge deletion and split completion

A graph is a *split graph* if its vertex set can be partitioned into a clique and an independent set. We use $C \uplus I$, where C being a clique and I an independent set, to denote a *split partition* of a split graph. Note that a split graph may have more than one split partition; e.g., a complete graph on n vertices has $n+1$ different split partitions. The forbidden induced subgraphs of split graphs are $2K_2$, C_4, and C_5. From both the definition and the forbidden induced subgraphs we can see that the complement of a split graph is also a split graph. Thus, the split completion problem is polynomially equivalent to the split edge deletion problem. For the convenience of presentation, we work on the edge deletion problem.

Note that (G, k) is a yes-instance if and only if there exists a partition $C \uplus I$ of $V(G)$ such that C is a clique and $|E(I, I)| \leqslant k$; this is a split partition of $G - E(I, I)$. We call such a partition a *valid partition* of the instance (G, k). The problem is thus equivalent to finding a valid partition. We notice that some vertices can be easily decided to which side of a valid partition they should belong. For example, unless the instance is trivial, a simplicial vertex always belong to the independent set in any valid partition. Even after we know the destinations of these vertices, however, we cannot safely delete them. This brings us to the *annotated version* of the problem, where we mark certain vertices that can only be put into the independent set in a valid partition. We use (G, I_0, k) to denote such an annotated instance, where I_0 denotes the set of marked vertices. The original instance can be viewed as $(G, \emptyset, k)$, and a valid partition of an annotated instance (G, I_0, k) needs to satisfy the additional requirement that $I_0 \subseteq I$.

We can easily retrieve back an unannotated instance from an annotated instance. It suffices to add a small number of new vertices and make each of them adjacent to all other vertices but I_0.

▶ **Rule 5.1.** *Let (G, I_0, k) be an annotated instance. Add a clique of $\sqrt{2k} + 1$ new vertices, and make each of them adjacent to all the vertices in $V(G) \setminus I_0$. Return the result as an unannotated instance.*

Safeness of Rule 5.1. Let K denote the clique of new vertices, and let (G', k) be the resulting instance. For any valid partition $C \uplus I$ of (G, I_0, k), the partition $(C \cup K) \uplus I$ is a valid partition of (G', k) because $C \subseteq V(G) \setminus I \subseteq N(x)$ for every $x \in K$. For a valid partition $C \uplus I$ of (G', k), if any vertex in I_0 is in C, then we must have $K \subseteq I$. Since K is a clique of order $\sqrt{2k} + 1$, we have $|E(I, I)| > k$, which contradicts the validity of the partition. ◀

The aforementioned observation on simplicial vertices is formalized by the following rule.

▶ **Rule 5.2.** *Let v be a simplicial vertex in $V(G) \setminus I_0$. If $|E(G - (N[v] \setminus I_0))| \leqslant k$, then return a trivial yes-instance. Otherwise, add v to I_0.*

Safeness of Rule 5.2. In the first case, $(N[v] \setminus I_0) \uplus (V(G) \setminus N[v] \cup I_0)$ is a valid partition. Otherwise, we show by contradiction that $v \in I$ in any valid partition $C \uplus I$ of (G, I_0, k). Since C is a clique, if $v \in C$, then $C \subseteq N[v] \setminus I_0$. Thus, $E(G - (N[v] \setminus I_0)) \subseteq E(I, I)$, but then $|E(I, I)| > k$, contradicting the validity of the partition. ◀

We construct a modulator M as follows. We greedily find a maximal packing of vertex-disjoint $2K_2$'s, C_4's, and C_5's. Let M be the set of vertices in all subgraphs we found. We can terminate the algorithm by returning a trivial no-instance if we have found more than k vertex-disjoint forbidden induced subgraphs from G. Henceforth, we may assume that $|M| \leqslant 5k$, and we fix a split partition $C_M \uplus I_M$ of $G - M$. The following simple observation enables us to know the destinations of more vertices.

▶ **Lemma 8** ($\star$). *For any valid partition $C \uplus I$ of (G, k), if one exists, (i) $|I_M \cap C| \leqslant 1$; and (ii) $|C_M \cap I| \leqslant \sqrt{2k}$.*

We say that a vertex is a *c-vertex*, respectively, an *i-vertex*, if it is in C, respectively, in I, for any valid partition $C \uplus I$ of (G, I_0, k). By Lemma 8(i), C contains at most one vertex in I_M. Thus, every vertex that has more than $k + 1$ neighbors in I_M is a c-vertex, while the following are i-vertices:

- every vertex with more than $\sqrt{2k}$ non-neighbors in C_M; and
- every vertex nonadjacent to a c-vertex.

We can indeed delete all the c-vertices, as long as we keep their non-neighbors marked. Note that after obtaining the initial split partition $C_M \uplus I_M$ of $G - M$, we do not need to maintain the invariant that M is a modulator, though we do maintain that C_M is a clique and that I_M is an independent set throughout. During our algorithm, we maintain M, C_M, I_M, and I_0 as a partition of $V(G)$. Therefore, whenever we mark a vertex, we remove it from the set that originally contains it, and move it to I_0.

▶ **Rule 5.3.** *Let* (G, I_0, k) *be an annotated instance.*

i) *Mark every vertex that has more than* $\sqrt{2k}$ *non-neighbors in* C_M*.*

ii) *If a vertex* v *has more than* $k+1$ *neighbors in* $I_M \cup I_0$*, then mark every vertex in* $V(G) \setminus N[v]$ *and delete* v*.*

Safeness of Rule 5.3. Let I_0' denote the set of marked vertices after the reduction. It is trivial that if the resulting instance of i) is a yes-instance, then the original is also a yes-instance. For ii), any valid partition $C' \uplus I'$ of $(G - v, I_0', k)$ can be extended to a valid partition $(C' \cup \{v\}) \uplus I'$ of (G, I_0, k) because $C' \subseteq V(G) \setminus I_0' \subseteq N[v]$.

For the other direction, let $C \uplus I$ be any valid partition of (G, I_0, k). i) Since C is a clique, $C_M \setminus N(v) \subseteq I$ for every $v \in C$. By Lemma 8(ii), if $|C_M \setminus N(v)| > \sqrt{2k}$ for some vertex v, then v has to be in I. Thus, $C \uplus I$ is also a valid partition of the new instance (G, I_0', k). ii) By Lemma 8(i), $|I_M \setminus I| \leqslant 1$. As $I_0 \subseteq I$ and $|N(v) \cap (I_M \cup I_0)| > k+1$, there are at least $k+1$ edges between v and I. Since $|E(I, I)| \leqslant k$, we must have $v \in C$. Moreover, since C is a clique, $C \subseteq N[v]$, and every vertex nonadjacent to v has to be in I. This justifies the marking of $V(G) \setminus N[v]$. Clearly, $(C \setminus \{v\}) \uplus I$ is a valid partition of $(G - v, I_0', k)$. ◀

The next rule is straightforward: since I_0 has to be in the independent set, every solution contains all the edges in $E(I_0, I_0)$.

▶ **Rule 5.4.** *Let* (G, I_0, k) *be an annotated instance. Remove all the edges in* $E(I_0, I_0)$*, and decrease* k *accordingly.*

Safeness of Rule 5.4. By the definition of the annotated instance, any solution E_- of (G, I_0, k) contains all the edges in $E(I_0, I_0)$. Moreover, $E_- \setminus E(I_0, I_0)$ is a solution to $G - E(I_0, I_0)$, and its size is at most $k - |E(I_0, I_0)|$. On the other hand, if $(G - E(I_0, I_0), k - |E(I_0, I_0)|)$ is a yes-instance, then any solution of this instance, together with $E(I_0, I_0)$, makes a solution of (G, I_0, k) of size at most k. ◀

Once there are no edges among vertices in I_0, we can replace I_0 with another independent set as long as we keep track of the number of edges between every vertex $v \in V(G) \setminus I_0$ and I_0. The following rule reduces the cardinality of I_0. Note that if Rule 5.3 is not applicable, then $p \leqslant k$.

▶ **Rule 5.5.** *Let* (G, I_0, k) *be an annotated instance where* I_0 *is an independent set. Introduce* p *new vertices* $v_1, v_2, \ldots, v_p$*, where* $p = \max_{v \in V(G)} |N(v) \cap I_0|$*. For each vertex* $x \in N(I_0)$*, make* x *adjacent to* $v_1, \ldots, v_{|N(x) \cap I_0|}$*. Remove all vertices in* I_0*, and mark the set of new vertices.*

Instead of proving the safeness of Rule 5.5, we prove a stronger statement.

▶ **Lemma 9.** *Let* (G, I_0, k) *and* (G', I_0', k) *be two annotated instances where* $G - I_0 = G' - I_0'$ *and both* I_0 *and* I_0' *are independent sets. If* $|N_G(x) \cap I_0| = |N_{G'}(x) \cap I_0'|$ *for every* $x \in V(G) \setminus I_0$*, then* (G, I_0, k) *is a yes-instance if and only if* (G', I_0', k) *is a yes-instance.*

Proof. We show that $C \uplus I$ is a valid partition of (G, I_0, k) if and only if $C \uplus ((I \setminus I_0) \cup I_0')$ is a valid partition of (G', I_0', k). Note that

$$|E(I \setminus I_0, I_0)| = \sum_{x \in I \setminus I_0} |N_G(x) \cap I_0| = \sum_{x \in I \setminus I_0'} |N_{G'}(x) \cap I_0'| = |E(I \setminus I_0', I_0')|.$$

Since $G - I_0 = G' - I_0'$, and since there is no edge in $G[I_0]$ or $G'[I_0']$, the claim follows. ◀

Let us recall an important observation of Guo [12].

▶ **Lemma 10** ([12]). *If a vertex v is not contained in any $2K_2$, C_4, or C_5, then* $\mathrm{opt}(G - v) = \mathrm{opt}(G)$.

Both Guo [12] and Ghosh et al. [10] used a rule derived from this observation to delete vertices, and this is their only rule that removes vertices from the graph. We may show that the same rule indeed works for our annotated instances, for which however we have to go through the original argument of [12]. We note that if a vertex v in I_0 is adjacent to two vertices u and w with $uw \notin E(G)$, then any solution has to contain at least one of edges uv and vw (u and w cannot be both in the clique). We say that an induced P_3 is I_0-*centered* if the degree-two vertex of this P_3 is from I_0. In a sense, I_0-centered P_3's are "minimal forbidden structures" for our annotated instances. Accordingly, a C_4 or C_5 involving a vertex from I_0 is no longer minimal. In summary, the "minimal forbidden structures" are C_4's and C_5's in $G - I_0$, all $2K_2$'s, and I_0-centered P_3's. Note that a "minimal forbidden structure" intersecting I_0 has to be a $2K_2$ or an I_0-centered P_3, and this gives another explanation of the correctness of Lemma 9, which exchanges these two kinds of "minimal forbidden structures" with each other. The following rule can be viewed as the annotated version of the rule of Guo [12], and its safeness can be argued using Lemma 10.

▶ **Rule 5.6** ($\star$). *Let (G, I_0, k) be an annotated instance where I_0 is an independent set, and let v be a vertex in C_M. If v is not contained in any $2K_2$ or any I_0-centered P_3, and every C_4 and C_5 that contains v intersects I_0, then remove v from G.*

We call an annotated instance *reduced* if none of Rules 5.2–5.6 is applicable to this instance. The following lemma bounds the cardinalities of C_M and I_M in a reduced instance.

▶ **Lemma 11.** *If a reduced instance (G, I_0, k) is a yes-instance, then $|C_M| \leqslant 3k\sqrt{2k}$ and $|I_M| \leqslant k + 1$.*

Proof. Let E_- be any solution to (G, I_0, k) with at most k edges. Since Rule 5.6 is not applicable, every vertex in C_M is contained in some $2K_2$ or I_0-centered P_3, or some C_4 or C_5 in $G - I_0$. Any of these structures contains an edge in E_-. Therefore, to bound $|C_M|$, it suffices to count how many vertices in C_M can form a $2K_2$ or I_0-centered P_3, or a C_4 or C_5 in $G - I_0$ with an edge $xy \in E_-$.

- If a vertex $v \in C_M$ is in a $2K_2$ with edge xy, then either $v \in \{x, y\}$ or v is adjacent to neither x nor y. In the first case, no other vertex in C_M can occur in any $2K_2$ with xy. Since $xy \in E(G)$, at least one of them is not in I_0 (Rule 5.4). This vertex has at most $\sqrt{2k}$ non-neighbors in C_M. Therefore, the total number of vertices in C_M that can occur in any $2K_2$ with xy is at most $\sqrt{2k}$.
- If xy is an edge in any I_0-centered P_3, then precisely one of them is in I_0. Assume without loss of generality $x \in I_0$. If a vertex $v \in C_M$ is in an I_0-centered P_3 with the edge xy, then either $v = y$, or v is not adjacent to y. Since $y \notin I_0$, it has at most $\sqrt{2k}$ non-neighbors in C_M. Thus, the total number of vertices in C_M that can occur in any I_0-centered P_3 containing xy is at most $\sqrt{2k} + 1$.

- If a vertex $v \in C_M$ is in a C_4 or C_5 that contains xy, then v is adjacent to at most one of x and y. Since this C_4 or C_5 is in $G - I_0$, each of x and y has at most $\sqrt{2k}$ non-neighbors in C_M. Thus, the total number of vertices in C_M that can occur in such a C_4 or C_5 is at most $2\sqrt{2k}$.

Noting that an edge cannot satisfy the conditions of both the second ($|\{x, y\} \cap I_0| = 1$) and third ($|\{x, y\} \cap I_0| = 0$) categories, we can conclude $|C_M| \leqslant k(\sqrt{2k} + 2\sqrt{2k}) = 3k\sqrt{2k}$.

Since Rule 5.2 is not applicable, no vertex in I_M is simplicial. Suppose that $C \uplus I$ is a valid partition of G. Since C is a clique, for each vertex $v \in I_M \cap I$, at least one neighbor of v is in I. Therefore, each vertex $v \in I_M \cap I$ is incident to an edge in the solution $E(I, I)$. Noting that I_M is an independent set, we have $k \geqslant |I_M \cap I| \geqslant |I_M| - 1$, where the second inequality follows from Lemma 8(i). Thus, $|I_M| \leqslant k + 1$, and this concludes this proof. ◀

Note that the application of Rule 5.1 is different from the other ones. The application of one of Rules 5.2–5.6 may trigger the applicable of another. After the application of Rule 5.1, the instance is no longer annotated, and we will not go back to check the other rules. We summarize the algorithm in Algorithm 1.

Algorithm 1 A summary of our kernelization algorithm for split edge deletion.

INPUT: an instance (G, k) of the split edge deletion problem.
OUTPUT: an equivalent instance (G', k') with $|V(G')| = O(k'^{1.5})$.

1. $I_0 \leftarrow \emptyset$;
1. $M \leftarrow$ a maximal packing of vertex-disjoint $2K_2$'s, C_4's, and C_5's;
2. **if** $|M| > 5k$ **then return** a trivial no-instance;
3. **if** $k < 0$ **then return** a trivial no-instance;
4. **for each** simplicial vertex $v \in V(G) \setminus I_0$ **do** (Rule 5.2)
   if $|E(G - (N[v] \setminus I_0))| \leqslant k$ **then return** a trivial yes-instance;
   else $I_0 \leftarrow I_0 \cup \{v\}$;
5. remove c-vertices and mark i-vertices (Rule 5.3);
6. **if** $E(I_0, I_0) \neq \emptyset$ **then**
   remove edges in $E(I_0, I_0)$ and decrease k (Rule 5.4);
7. merge I_0 into $\leqslant k$ vertices (Rule 5.5);
8. remove vertices in C_M not contained in certain structures (Rule 5.6);
9. **if** any of Rules 5.2–5.4 and 5.6 made a change **then goto** 3;
10. **if** $|C_M| + |I_M| > 3k\sqrt{2k} + k + 1$ **then return** a trivial no-instance;
11. add $\sqrt{2k} + 1$ new vertices and remove all marks (Rule 5.1);
12. **return** (G, k).

▶ **Theorem 12.** *There is an $O(k^{1.5})$-vertex kernel for the split edge deletion problem.*

Proof. We use the algorithm described in Algorithm 1. The first two steps build the modulator, and their correctness follows from that any solution contains at least one edge of each forbidden induced subgraph of G. Step 3 is obviously correct. Steps 4–8 follow from the safeness of the rules; so is step 11. The correctness of step 10 is ensured by Lemma 11.

The cardinality of M is at most $5k$, and it never increases during the algorithm. After step 7, $|I_0| \leqslant k$. We have bounded the cardinalities of C_M and I_M in Lemma 11. Step 11 increases $|C_M|$ by $\sqrt{2k} + 1$. Putting them together, we have

$$|V(G)| \leqslant 5k + k + (3k\sqrt{2k} + \sqrt{2k} + 1) + k + 1 = O(k^{1.5}).$$

It is easy to verify that each reduction rule can be checked and applied in polynomial time. To see that the algorithm runs in polynomial time, note that if any of Rules 5.2–5.4 and 5.6 made a change to the instance, then either k decreases by one (Rule 5.4), or the cardinality of $V(G) \setminus I_0$ decreases by one (the other three rules). ◀

Since the class of split graphs is self-complementary, our algorithm also implies a kernel for the split completion problem. This kernel actually has fewer edges than the one for split edge deletion.

▶ **Theorem 13.** *There is a kernel of* $O(k^{1.5})$ *vertices and* $O(k^{2.5})$ *edges for the split completion problem.*

Proof. Let (G, k) be the input instance of the split completion problem. We can either take the complement of the input graph and consider it as an instance of the split edge deletion problem, or run the "complemented versions" of the rules. In the final result, we have an independent set of at most $O(k\sqrt{k})$ vertices, and at most $O(k)$ other vertices. The claim then follows. ◀

6 Pseudo-split edge deletion and pseudo-split completion

The algorithm in Alg. 1 also works for the pseudo-split ($\{2K_2, C_4\}$-free) edge deletion problem.

▶ **Theorem 14.** *There is an* $O(k^{1.5})$*-vertex kernel for the pseudo-split edge deletion problem. There is a kernel of* $O(k^{1.5})$ *vertices and* $O(k^{2.5})$ *edges for the pseudo-split completion problem.*

References

1. Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM Journal on Computing*, 22(3):560–572, 1993. doi:10.1137/0222038.
2. Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.
3. Yixin Cao and Jianer Chen. Cluster editing: Kernelization based on edge cuts. *Algorithmica*, 64(1):152–169, 2012. doi:10.1007/s00453-011-9595-1.
4. Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification. arXiv:2001.06867, 2020.
5. Pål Grønås Drange, Markus Sortland Dregi, Daniel Lokshtanov, and Blair D. Sullivan. On the threshold of intractability. In Nikhil Bansal and Irene Finocchi, editors, *Proceedings of the 23rd*, volume 9294 of *LNCS*, pages 411–423. Springer, 2015. doi:10.1007/978-3-662-48350-3_35.
6. 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. doi:10.1145/2799640.
7. Pål Grønås Drange and Michal Pilipczuk. A polynomial kernel for trivially perfect editing. *Algorithmica*, 80(12):3481–3524, 2018. doi:10.1007/s00453-017-0401-6.
8. Maël Dumas, Anthony Perez, and Ioan Todinca. A Cubic Vertex-Kernel for Trivially Perfect Editing. In *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 45:1–45:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.MFCS.2021.45.
9. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.
10. Esha Ghosh, Sudeshna Kolay, Mrinal Kumar, Pranabendu Misra, Fahad Panolan, Ashutosh Rai, and M. S. Ramanujan. Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, 71(4):989–1006, 2015. doi:10.1007/s00453-013-9837-5.
11. Niels Grüttemeier and Christian Komusiewicz. On the relation of strong triadic closure and cluster deletion. *Algorithmica*, 82(4):853–880, 2020. doi:10.1007/s00453-019-00617-1.

12 Jiong Guo. Problem kernels for NP-complete edge deletion problems: Split and related graphs. In Takeshi Tokuyama, editor, *Proceedings of the 18th*, volume 4835 of *LNCS*, pages 915–926. Springer, 2007. doi:10.1007/978-3-540-77120-3_79.

13 Peter L. Hammer and Bruno Simeone. The splittance of a graph. *Combinatorica*, 1(3):275–284, 1981. doi:10.1007/BF02579333.

14 Athanasios L. Konstantinidis, Stavros D. Nikolopoulos, and Charis Papadopoulos. Strong triadic closure in cographs and graphs of low maximum degree. *Theoretical Computer Science*, 740:76–84, 2018. doi:10.1016/j.tcs.2018.05.012.

15 Federico Mancini. *Graph Modification Problems Related to Graph Classes*. PhD thesis, University of Bergen, Bergen, Norway, 2008.

16 Dániel Marx and R. B. Sandeep. Incompressibility of H-free edge modification problems: Towards a dichotomy. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *Proceedings of the 28th*, volume 173 of *LIPIcs*, pages 72:1–72:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.72.

17 Assaf Natanzon, Ron Shamir, and Roded Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109–128, 2001. doi:10.1016/S0166-218X(00)00391-7.

18 Roded Sharan. *Graph Modification Problems and their Applications to Genomic Research*. PhD thesis, Tel-Aviv University, Tel Aviv, Israel, 2002.

19 E. S. Wolk. The comparability graph of a tree. *Proceedings of the American Mathematical Society*, 13:789–795, 1962. doi:10.1090/S0002-9939-1962-0172273-0.

20 Jing-Ho Yan, Jer-Jeong Chen, and Gerard Jennhwa Chang. Quasi-threshold graphs. *Discrete Applied Mathematics*, 69(3):247–255, 1996. doi:10.1016/0166-218X(96)00094-7.

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