Preprocessing for Treewidth: A Combinatorial Analysis through Kernelization*

Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch

Utrecht University, The Netherlands, \{hansb,bart,kratsch\}@cs.uu.nl

Abstract. Using the framework of kernelization we study whether efficient preprocessing schemes for the Treewidth problem can give provable bounds on the size of the processed instances. Assuming the AND-distillation conjecture to hold, the standard parameterization of Treewidth does not have a kernel of polynomial size and thus instances \((G, k)\) of the decision problem of Treewidth cannot be efficiently reduced to equivalent instances of size polynomial in \(k\). In this paper, we consider different parameterizations of Treewidth. We show that Treewidth has a kernel with \(O(\ell^3)\) vertices, where \(\ell\) denotes the size of a vertex cover, and a kernel with \(O(\ell^4)\) vertices, where \(\ell\) denotes the size of a feedback vertex set. This implies that given an instance \((G, k)\) of Treewidth we can efficiently reduce its size to \(O((\ell^*)^4)\) vertices, where \(\ell^*\) is the size of a minimum feedback vertex set in \(G\). In contrast, we show that Treewidth parameterized by the vertex-deletion distance to a co-cluster graph and Weighted Treewidth parameterized by the size of a vertex cover do not have polynomial kernels unless \(NP \subseteq coNP/poly\). Treewidth parameterized by the target value plus the deletion distance to a cluster graph has no polynomial kernel unless the AND-distillation conjecture does not hold.

1 Introduction

Treewidth is a well-studied graph parameter, with many theoretical and practical applications. A related parameter is Weighted Treewidth, where vertices have weights, and the width of a tree decomposition is the maximum over all bags of the sum of the weights of the vertices in a bag minus one. In this work we study the decision problems related to these width parameters, which given a graph \(G\) and integer \(k\) ask whether the (weighted) treewidth of \(G\) is at most \(k\). For precise definitions, see Section 2.

Preprocessing heuristics for Treewidth and Weighted Treewidth have been studied in a practical setting [8,9,18]. The experimental results reported in these papers show that there are preprocessing heuristics that give significant reductions in size for many practical instances, making it more feasible to compute, exactly or approximately, the treewidth of those graphs. However, these

* This work was supported by the Netherlands Organization for Scientific Research (N.W.O.), project “KERNELS: Combinatorial Analysis of Data Reduction”.
heuristics do not give any guarantees on the effectiveness of the preprocessing: there is no provable bound on the size of the processed instances. The purpose of this work is to give a theoretical analysis of the potential of preprocessing for Treewidth, studying whether there are efficient preprocessing procedures whose effectiveness can be proven, and what the resulting size bounds look like. Such investigations are made possible using the concept of kernelization, which is a relatively young subfield of algorithm design and analysis. A kernelization algorithm (or kernel) is a polynomial-time algorithm which given an instance \((x, k) \in \Sigma^* \times \mathbb{N}\) of some parameterized problem, computes an equivalent instance \((x', k')\) whose size is bounded by a function \(f(k)\) depending only on the chosen parameter, i.e., \(|x'|, k' \leq f(k)\). The function \(f\) is the size of the kernel, and polynomial kernels \((f \in k^{O(1)})\) are of particular interest.

From a theoretical point of view, the fact that Treewidth belongs to FPT (see for instance [3, 21]), implies that there is a kernel for the problem. However, the size of such a kernel depends on the function of the parameter in the running time of the FPT algorithm; with the current state of FPT algorithms for Treewidth this size would be exponential in \(k^3\) (where \(k\) is the target treewidth). Bodlaender et al. [6] have shown that Treewidth with standard parameterization (i.e., parameterized by \(k\)) has no polynomial kernel unless all coNP-complete problems have distillation algorithms; hence it is unlikely that there is a polynomial-time algorithm that reduces the size of an instance \((G, k)\) of Treewidth to a polynomial in the desired treewidth \(k\). We therefore turn to other parameters (e.g., the vertex cover number of the input graph), and determine whether we can efficiently shrink an input of Treewidth to a size which is polynomial in such a parameter. We consider different structural parameters of the input graph: these parameters measure the number of vertex deletions needed to transform the input into a graph of some very simple graph class. All parameterized problems we consider fit the following template, where \(F\) is a class of graphs:

**Treewidth parameterized by a modulator to \(F\)**

**Instance:** A graph \(G = (V, E)\), a positive integer \(k\), and a set \(S \subseteq V\) such that \(G - S \in F\).

**Parameter:** \(\ell := |S|\).

**Question:** \(\text{tw}(G) \leq k?\)

The set \(S\) is a modulator to the class \(F\).

*Our work.* In this paper, we add positive theoretical results to the positive experimental work. Our theoretical results can possibly be of practical value, but an experimental evaluation has not yet been undertaken. We first take as parameter the size of a vertex cover of \(G\), resulting in the problem Treewidth parameterized by a vertex cover (which fits into the given template when using \(F\) as the class of edgeless graphs). We prove that this problem admits a polynomial kernel with \(O(\ell^3)\) vertices. Since we can first compute a 2-approximation for the minimum vertex cover and then feed this to our kernelization algorithm, this implies that an instance \((G, k)\) of Treewidth on a graph with a minimum vertex cover
cover of size $\ell^*$ can be shrunk in polynomial-time into an instance with $O((\ell^*)^3)$ vertices, even if we are not given a minimum vertex cover in the input.

We then turn to the parameter “feedback vertex number”, which is easily seen to be at most the value of the vertex cover in the input. We extend our positive results by showing that $\text{Treewidth parameterized by a feedback vertex set}$ (which fits the template when $F$ is the class of forests) admits a kernel with $O(\ell^4)$ vertices. By using a polynomial-time 2-approximation algorithm for $\text{Feedback Vertex Set}$ [2], we can again drop the assumption that such a set is supplied in the input.

After these two examples it becomes an interesting question whether there is a parameter even smaller than the feedback vertex number in which the size of an instance can be bounded efficiently. Since $\text{Treewidth}$ is trivially solvable on chordal graphs, and the deletion distance to a chordal graph is at most the feedback vertex number, one might hope that $\text{Treewidth parameterized by a modulator to chordal graphs}$ admits a polynomial kernel. This appears to be very unlikely. Assuming the AND-distillation conjecture [6] we prove the stronger statement that even when using the compound parameter “target treewidth plus the size of a given modulator to cluster graphs”, $\text{Treewidth}$ does not admit a polynomial kernel - recall that a cluster graph is a disjoint union of cliques. We also prove that $\text{Treewidth parameterized by a modulator to co-cluster graphs}$ does not admit a polynomial kernel unless $\text{NP} \subseteq \text{coNP/poly}$, and use this result to show that, under the same assumption, the $\text{Weighted Treewidth}$ problem does not even admit a polynomial kernel when parameterized by the size of a vertex cover. It is interesting to note the difference between $\text{Treewidth}$ and $\text{Weighted Treewidth}$ when parameterized by vertex cover.

Organization of the paper. After this introduction, we give preliminary definitions and results in Section 2. In Section 3 we show that $\text{Treewidth parameterized by a vertex cover}$ has a kernel with $O(\ell^3)$ vertices. To do so, we introduce a number of ‘safe’ reduction rules, that are variants of rules from existing treewidth algorithms and preprocessing methods, including rules that remove simplicial vertices. In Section 4 we turn to $\text{Treewidth parameterized by a feedback vertex set}$ (of size $\ell$) and show a kernel with $O(\ell^4)$ vertices. A key role, in addition to variants of the rules for vertex cover, will be played by $\text{almost simplicial vertices}$ and we will give a set of safe reduction rules that remove all those vertices. In Section 5 we present our lower bound results for $\text{Treewidth}$ parameterized by distance from cluster respectively co-cluster graphs as well as for $\text{Weighted Treewidth}$ parameterized by a vertex cover; we build upon the recent framework of Bodlaender et al. [6] as well as the notion of cross-composition [7]. Some final remarks are made in Section 6.

2 Preliminaries

In this work all graphs are finite, simple, and undirected. The open neighborhood of a vertex $v \in V$ in a graph $G$ is denoted by $N_G(v)$, and its closed neighborhood
is $N_G[v]$. If $S \subseteq V$ is a vertex set then $G - S$ denotes the graph obtained from $G$ by deleting all vertices of $S$ and their incident edges. A vertex $v$ is simplicial in a graph $G$ if $N_G(v)$ is a clique. A vertex $v \in V$ is almost simplicial in a graph $G$ if $v$ has a neighbor $w$ such that $N_G(v) - \{w\}$ is a clique. In such a case, we call $w$ the special neighbor of $v$.

A tree decomposition of a graph $G = (V, E)$ is a pair $\{(X_i \mid i \in I), T = (I, F)\}$ with $\{X_i \mid i \in I\}$ a family of subsets of $V$, and $T$ a tree on edge set $F$, such that

- $\bigcup_{i \in I} X_i = V$.
- For all $\{v, w\} \in E$, there is an $i \in I$ with $v, w \in X_i$.
- For all $v \in V$, the set $I_v = \{i \in I \mid v \in X_i\}$ induces a subtree of $T$.

The sets $X_i$ are called the bags of the tree decomposition. The width of a tree decomposition $\{(X_i \mid i \in I), T = (I, F)\}$ is $\max_{i \in I} |X_i| - 1$, and the treewidth of $G$ is the minimum width of a tree decomposition of $G$.

Suppose we have a graph $G = (V, E)$ with a weight function $w : V \to \mathbb{N}$. The weighted width of a tree decomposition $\{(X_i \mid i \in I), T = (I, F)\}$ of $G$ equals $\max_{i \in I} \sum_{v \in X_i} w(v) - 1$, and the weighted treewidth of $G$ is the minimum weighted width of a tree decomposition of $G$.

A graph $H = (W, F)$ is a minor of a graph $G = (V, E)$, if $H$ can be obtained from $G$ by a series of zero or more vertex deletions, edge deletions, and/or edge contractions. An edge contraction is the operation where two adjacent vertices $v, w$ are replaced by one vertex with neighborhood $(N(v) \cup N(w)) \setminus \{v, w\}$.

The following proposition is well known.

**Proposition 1.** Let $H$ be a minor of $G$. Then the treewidth of $H$ is at most the treewidth of $G$.

The kernelization algorithms we present consist of a number of reduction rules. In each case, the input to the rule is a graph $G = (V, E)$, an integer $k$, and a deletion set $S \subseteq V$ such that $G - S$ is a member of the relevant graph class $\mathcal{F}$, and the output is an instance $(G', (V', E'), k', S')$. A rule is said to be safe if for all inputs $(G, k, S)$ which satisfy $G - S \in \mathcal{F}$ we have $\text{tw}(G) \leq k$ if $G - S \in \mathcal{F}$ we have $\text{tw}(G') \leq k'$ and $G' - S' \in \mathcal{F}$. We will sometimes say that the algorithm answers yes or no; this should be interpreted as outputting a constant-size yes or no instance of the problem at hand, i.e., a clique on three vertices with $k = 2$, respectively the same clique with $k = 1$.

### 3 Kernelization with respect to vertex cover number: eliminating simplicial vertices

In this section we show our kernelization for **Treewidth parameterized by a vertex cover** (i.e., parameterized by a modulator to an independent set). The kernelization focuses mostly on simplicial vertices; removing them (and possibly updating a bound for the treewidth) is a well known and often used preprocessing rule for **Treewidth**; see the discussion in [9]. Another rule, first used in the
linear time algorithm for bounded treewidth in \cite{3} adds edges between vertices with many common neighbors. The rule was also used in lower bound heuristics for treewidth, see \cite{16,17}, see also \cite{10,4}.

**Rule 1 (Low degree simplicial vertex)** If \( v \) is a simplicial vertex of degree at most \( k \) then remove \( v \).

**Rule 2 (High degree simplicial vertex)** If \( v \) is a simplicial vertex of degree greater than \( k \) then answer no.

Standard theory on treewidth shows that Rules 1 and 2 are safe. It is well known \cite{3} that if non-adjacent vertices \( v, w \) have at least \( k+1 \) common neighbors, then adding the edge \{\( v, w \)\} does not affect whether the treewidth of the graph is at most \( k \). We use this rule in a restricted setting, to ensure that \( S \) remains a vertex cover of the graph.

**Rule 3 (Common neighbors improvement)** Suppose that \{\( v, w \)\} \( \not\in \) \( E \) and that \( v \in S \) or \( w \in S \). If \( v \) and \( w \) have at least \( k+1 \) common neighbors, then add the edge \{\( v, w \)\}.

Yet another simple rule is the following, using that \( S \) is a vertex cover of \( G \).

**Rule 4 (Trivial decision)** If \( k \geq |S| \), then answer yes.

Safeness can be argued as follows. The treewidth of \( G \) is at most \( |S| \): for each \( v \in V - S \), take a bag with vertex set \( S \cup \{v\} \), and connect these bags in any way. This gives a tree decomposition of \( G \) of width at most \( |S| \).

It is not hard to argue the, possibly surprising, fact that the exhaustive application of Rules 1-4 (i.e., until we answer no or yes or no application of one of these rules is possible) already gives a polynomial kernel for **Treewidth parameterized by a vertex cover**. It is clear that this reduction can be performed in polynomial time (it is easy to do it in time \( O(|V| \cdot |E|) \)).

**Theorem 1.** **Treewidth parameterized by a vertex cover** has a kernel with \( O(\ell^3) \) vertices.

**Proof.** Let \((G, k, S)\) be an instance of **Treewidth parameterized by a vertex cover**. Let \((G', k', S')\) be the instance obtained from exhaustive application of Rules 1-4. By safety of the reduction rules \((G', k', S')\) is yes if and only if \((G, k, S)\) is yes.

The reduction rules guarantee that \( S' \subseteq S \) is a vertex cover in \( G' \), with \( |S'| \leq \ell \). Each vertex \( v \in V' - S' \) has at least one pair of distinct neighbors in \( S' \) that are not adjacent, otherwise \( v \) is simplicial and would have been handled by Rule 1 or Rule 2. Assign \( v \) to this pair. If we assign \( v \) to the pair \{\( w, x \)\}, then \( v \) is a common neighbor of \( w \) and \( x \). Hence a pair cannot have more than \( k \) vertices assigned to it, otherwise Rule 3 applies. As there are at most \( \ell \cdot (\ell - 1)/2 \) pairs of non-adjacent neighbors in \( S' \), we have \( |V' - S'| \leq k \cdot \ell \cdot (\ell - 1)/2 \leq \ell^2 \cdot (\ell - 1)/2 \in O(\ell^3). \) \( \square \)

\(^1\) In any triangulation, either \{\( v, w \)\} is an edge, or all common neighbors of \( v \) and \( w \) form a clique. In the latter case, this clique plus \( v \) is a clique with at least \( k + 2 \) vertices, implying a tree decomposition of width at least \( k + 1 \).
By combining Theorem 1 with a polynomial-time 2-approximation algorithm for vertex cover, we obtain the following corollary.

**Corollary 1.** There is a polynomial-time algorithm that given an instance \((G = (V, E), k)\) of TIGHTWIDTH computes an equivalent instance \((G' = (V', E'), k)\) such that \(V' \subseteq V\) and \(|V'| \in O((\ell^*)^3)\), where \(\ell^*\) is the size of a minimum vertex cover of \(G\).

## 4 A polynomial kernel for Treewidth parameterized by Feedback Vertex Set

In this section, we give the proof that TIGHTWIDTH \(\text{P}arameterized\ by\ \text{A\ Feed-}

back\ Vertex\ Set\) (of size \(\ell\)) has a kernel with \(O(\ell^4)\) vertices. The kernelization algorithm is again given by a set of safe reduction rules, that are applied while possible: two simple rules, three rules that remove all almost simplicial vertices (Section 4.1), and four rules that reduce the graph when there is a clique-seeing path (defined in Section 4.2). In Section 4.3 we show that graphs to which no rule applies have \(O(\ell^4)\) vertices, and thus arrive at our kernel bound.

The first new rule generalizes Rule 3; it was used in experiments [16] and its correctness is proven in [4]. The rule can be implemented in polynomial time by using a maximum flow algorithm to find the disjoint paths.

**Rule 5 (Disjoint paths improvement)** Suppose \(\{v, w\} \notin E\) and that \(v \in S\) or \(w \in S\). If there are at least \(k + 1\) internally vertex-disjoint paths between \(v\) and \(w\), then add the edge \(\{v, w\}\).

The second new rule is straightforward, and is correct because reasoning similar to that of Rule 4 shows that each graph with a feedback vertex set of size \(\ell\) has treewidth at most \(\ell + 1\).

**Rule 6 (Trivial decision)** If \(k \geq |S| + 1\), then answer yes.

### 4.1 Almost simplicial vertices

In [9], the notion of *almost simplicial vertex* was introduced, and a reduction rule was given that removed almost simplicial vertices whose degree was at most a known lower bound for the treewidth of the input graph. In this section, we give a set of rules that also remove almost simplicial vertices of higher degree. Rule 7 is a reformulation the *Low Degree Almost Simplicial Vertex Rule* from [9]. Rule 8 gives a simple way to deal with almost simplicial vertices of degree larger than \(k + 1\). Its correctness is obvious: \(v\) with its neighbors except its special neighbor forms a clique with at least \(k + 2\) vertices, so the treewidth is larger than \(k\).

**Rule 7 (Low Degree Almost Simplicial Vertex)** Let \(v\) be an almost simplicial vertex with special neighbor \(w\). If the degree of \(v\) is at most \(k\), then contract the edge \(\{v, w\}\) into \(w\) obtaining \(G'\). If \(v \in S\), then let \(S' := S \setminus \{v\} \cup \{w\}\), else let \(S' := S\).
Lemma 1. Rule 7 is safe.

Proof. It is clear that $S'$ is a feedback vertex set of $G'$.

Let $G'$ be the graph resulting after the operation. If the treewidth of $G$ is at most $k$, then the treewidth of $G'$ is at most $k$ as the treewidth cannot increase by contraction (Proposition 1).

Suppose the treewidth of $G'$ is at most $k$. Take a tree decomposition of $G'$ of width at most $k$. It is well-known that because $N_G(v)$ is a clique in $G'$, there must be a bag that contains all vertices of $N_G(v)$, say $N_G(v) \subseteq X_i$. Add a new bag with vertex set $N_G[v]$ and make it adjacent in the tree decomposition to node $i$; we obtain a tree decomposition of $G$ of width at most $k$. \hfill \Box

Rule 8 (High Degree Almost Simplicial Vertex) Let $v$ be an almost simplicial vertex. If the degree of $v$ is at least $k+2$, then answer no.

We introduce a new, more complex rule that deals with almost simplicial vertices of degree exactly $k+1$. The correctness proof can be found in the appendix.

Rule 9 (Degree $k+1$ Almost Simplicial Vertex) Let $v$ be an almost simplicial vertex with special neighbor $w$, and let the degree of $v$ be exactly $k+1$.

- If for each vertex $x \in N_G(v) - \{w\}$, there is an edge $\{x, w\} \in E$ or a path in $G$ from $x$ to $w$ that avoids $N_G[v] - \{x, w\}$, answer no.
- Otherwise, contract the edge $\{v, w\}$ to a new vertex $x$, obtaining $G'$. If $v \in S$ or $w \in S$, then let $S' := S \setminus \{v, w\} \cup \{x\}$, else let $S' := S$.

Note that the rules for almost simplicial vertices (Rules 7 and 8) can be easily seen to subsume the rules for simplicial vertices (Rules 1 and 2) used in the previous section (the first case of Rule 9 covers simplicial vertices of degree $k+1$).

The following proposition follows from counting arguments similar to those in the proof of Theorem 1 by observing that after exhaustive application of the rules no leaf in the forest $G - S$ is almost simplicial, and hence every such leaf must have a pair of neighbors in $S$ which are non-adjacent.

Proposition 2. Suppose an instance $(G, k)$ of Treewidth is given together with a feedback vertex set $S$ of size $\ell$. If we exhaustively apply Rules 4 to 9 then we obtain in polynomial time an equivalent instance $(G', k)$ with a feedback vertex set $S'$ of size at most $\ell$, such that the forest $G' - S'$ has $O(\ell^3)$ leaves.

Preprocessing heuristics. At this point, we want to make a sidestep. In [9], the Almost Simplicial Vertex Rule was given as a preprocessing heuristic for the optimization version of Treewidth. There, a variable low was invariantly kept as lower bound on the treewidth of the original input graph; the rule could be applied to almost simplicial vertices of degree $d$ whenever $d$ was at most this value low. Thus, in the work of [9], almost simplicial vertices of degree at least low +1 could not be dealt with. Using a variant of Rule 9 we can now extend the Almost Simplicial Vertex Rule and preprocess a graph such that we
remove all almost simplicial vertices, as follows. Suppose \( v \) is an almost simplicial vertex of degree \( d \) with special neighbor \( w \). If \( d - 1 > \text{LOW} \), then \( \text{LOW} \) is set to \( d - 1 \). Then, if \( d - 1 = \text{LOW} \), we check if for each vertex \( x \in N_G(v) - \{w\} \), there is an edge \( \{x, w\} \in E \) or a path from \( x \) to \( w \) that does not use any vertex in \( N_G(v) - \{x, w\} \). If so, \( \text{LOW} \) is increased by one. Now, we contract the edge \( \{x, w\} \). This rule is safe in the sense that the treewidth of the original graph \( G \) equals the maximum of \( \text{LOW} \) and the treewidth of the reduced graph.

An experimental evaluation of this ‘extended simplicial vertex rule’, similar as the work in [9] has not yet been undertaken.

### 4.2 Clique-seeing paths

We call a path \((v_0, v_1, \ldots, v_r, v_{r+1})\) a **clique-seeing path** that sees clique \( X \), if

- \( X = \bigcup_{i=1}^{r-1} N_G(v_i) \setminus \{v_0, v_1, \ldots, v_{r+1}\} \) is a clique.
- For each each \( i, 1 \leq i \leq r \), \( N(v_i) \subseteq \{v_i-1, v_i+1\} \cup X \).

An example is given in Fig. 1. Note that \( v_0 \) and \( v_{r+1} \) play a special role. Each \( v_i, 1 \leq i \leq r \), has exactly two neighbors outside \( X \), namely the previous and next vertex on the path \((v_i-1, v_i, v_i+1)\). In our analysis it is sufficient when we look at clique-seeing paths with all vertices on the path in the forest \( V - S \), and all vertices in the seen clique in the feedback vertex set \( S \), but the rules are also safe in other cases.

We present four reduction rules. The first rule deals with clique-seeing paths that see a clique of size at most \( k - 2 \). The second and third consider the case that \( \{v_1, \ldots, v_r\} \cup X \) separate \( v_0 \) and \( v_{r+1} \) in the graph. The fourth decides no if the path has at least \( 6k + 6 \) inner vertices and no other rule applies. All proofs are deferred to the appendix.

**Rule 10** Suppose we have a clique-seeing path \((v_0, v_1, \ldots, v_r, v_{r+1})\), that sees a clique \( X \) with \( |X| \leq k - 2 \). If

\[
N(v_r) \cap X \subseteq \bigcup_{1 \leq i \leq r-1} N(v_i) \cap X
\]

then contract the edge \( \{v_r, v_{r+1}\} \) into the vertex \( v_{r+1} \), obtaining \( G' \). If \( v_r \in S \), then let \( S' := S \setminus \{v_r\} \cup \{v_{r+1}\} \), else let \( S' := S \).
The next rule is based upon the notion of minimal almost clique separators from \[8\]. A set of vertices \(Q\) separates vertices \(v\) and \(w\) if each path from \(v\) to \(w\) uses at least one vertex in \(Q\). A set of vertices is a separator if there exist vertices \(v\) and \(w\) such that \(Q\) separates \(v\) from \(w\). \(Q\) is a minimal separator if there is a pair of vertices \(v\) and \(w\) that are minimally separated by \(Q\). \(Q\) minimally separates \(v\) and \(w\) if it separates \(v\) and \(w\) but there is no proper subset of \(Q\) that separates \(v\) and \(w\). A set of vertices \(Q\) is a minimal almost clique separator, if it is a minimal separator and there is a vertex \(v\in Q\) such that \(Q−\{v\}\) is a clique.

Bodlaender and Koster \[8\] have shown that the treewidth is not changed when edges are added to make a minimal almost clique separator into a clique. We use a version of this rule that ensures that \(S\) still is a feedback vertex set. Safeness of the following rule thus follows directly from the analysis in \[8\]. The discussion in \[8\] also shows that it can be tested in polynomial time.

**Rule 11** Let \(Q\) be a minimal almost clique separator in \(G\), and suppose there is at most one vertex \(v\in Q\) with \(v\not\in S\). Then, add an edge between each pair of non-adjacent vertices in \(Q\).

Exhaustive application of the following rule will provide us with a useful connectivity property that allows a rejection of instances with clique-seeing paths which remain long after application of Rules 5 to 12.

**Rule 12** Suppose \((v_0,v_1,v_2,v_3,v_4)\) is a clique-seeing path in \(V\setminus S\) that sees clique \(X\subseteq S\). Suppose \(\{v_1,v_2,v_3\}\cup X\) separates \(v_0\) from \(v_4\) and suppose that Rule 11 cannot be applied. Compute the treewidth of \(G[\{v_1,v_2,v_3\}\cup X]\). If it is larger than \(k\), then answer no, otherwise remove \(v_2\) from \(G\).

Polynomial-time computability of the treewidth of \(G[\{v_1,v_2,v_3\}\cup X]\) follows from a result in \[12\]. A detailed counting argument shows safeness of our last rule.

**Rule 13** Suppose we have a clique-seeing path \((v_0,...,v_{r+1})\) with \(r\geq 6k+6\), and suppose Rules 5–12 are not applicable. Then answer no.

### 4.3 The kernelization

We show that exhaustive application of Rules 5 through 13 gives the following kernelization result.

**Theorem 2.** Treewidth parameterized by a feedback vertex set has a kernel with \(O(\ell^4)\) vertices.

**Proof.** Let \((G,k,S)\) be an instance of Treewidth parameterized by a feedback vertex set and let \((G',k,S')\) be obtained from exhaustive application of Rules 5 through 13. Rules 12 and 13 are only tested for paths in \(G−S\); as that is a forest, we need to test at most \(O(n^2)\) paths, and thus the test has to be done a polynomial number of times. It was showed that all rules are safe and
that they can be performed exhaustively in polynomial time, so the instances are equivalent, $S'$ is a feedback vertex set of $G'$, and $|S'| \leq |S|$.

Let us analyze the size of $G'$. The forest $G' - S'$ has $O(\ell^3)$ leaves (Proposition 2), and hence $O(\ell^3)$ vertices of degree at least three. There are $O(\ell^3)$ paths in the forest connecting leaves and vertices of degree at least three. Each path of length at least $6k + 8$, i.e., with at least $6k + 6$ internal vertices is not clique-seeing by Rule 13. Thus, for the analysis, we split the paths into parts of size $6k + 8$ which are therefore not clique-seeing. At most $6k + 7$ vertices per path will not belong to such a part, but these are at most $O(\ell^3 \cdot k)$ in total. We assign each part to a pair of non-adjacent vertices in $S$ which are adjacent to internal vertices of the part. We can assign at most $k$ parts to a pair \{$u,v$\}: indeed, since the parts are disjoint they would otherwise give rise to more than $k$ disjoint $u - v$ paths, contradicting $(G', k, S')$ being reduced under Rule 5. Thus, we have $O(k(6k + 8)\ell^2) + O(\ell^3 \cdot k) = O(\ell^4)$ vertices in the forest $G' - S'$, and thus $O(\ell^4)$ vertices in $G'$.

Using a 2-approximation algorithm for feedback vertex set we obtain a corollary similar to the one given in the previous section.

**Corollary 2.** There is a polynomial-time algorithm that given an instance $(G = (V,E), k)$ of Treewidth computes an equivalent instance $(G' = (V',E'), k)$ such that $V' \subseteq V$ and $|V'| \in O((\ell^*)^4)$, where $\ell^*$ is the size of a minimum feedback vertex set of $G$.

5 Kernelization lower bounds

In this section we present our lower bound results for Treewidth and Weighted Treewidth; due to space restrictions the proofs are deferred to Section 13 of the appendix. We show the following results.

**Theorem 3.** Unless the AND-distillation conjecture fails and all coNP-complete problems have OR-distillation algorithms, Treewidth parameterized by target treewidth plus the size of a given modulator to cluster graphs does not admit a polynomial kernel.

For this result we use that Treewidth on co-bipartite graphs is NP-complete [1, Theorem 3.3]. The key idea is that by identifying one bipartition from each of $t$ co-bipartite graphs into one clique one obtains a graph that has treewidth at most some integer $k$ if and only if all $t$ graphs have treewidth at most $k$; this constitutes an AND-composition [6].

**Theorem 4.** Treewidth parameterized by a modulator to co-cluster graphs does not admit a polynomial kernelization unless $NP \subseteq coNP/poly$ (which would imply a collapse of the polynomial hierarchy to its third level).

The proof goes by giving a cross-composition from Treewidth to the target problem, i.e., a reduction of some $t$ instances of Treewidth into one instance.
of the parameterized problem with a small parameter value, which is yes iff one of the $t$ instances is yes (for details on cross-composition see [7]). We use that the join of $t$ graphs each on $n$ vertices has a treewidth equaling $(t - 1) \cdot n$ plus the minimum treewidth of any of the graphs (a corollary of work by Bodlaender and Möhring [11]). To turn the join of the graphs into an almost co-cluster, their edges are replaced by a set of $m$ cliques (each connected to the endpoints of one edge per graph), which are then added to the modulator.

**Theorem 5.** Weighted Treewidth parameterized by a vertex cover does not admit a polynomial kernelization unless $NP \subseteq coNP/poly$.

This final result is obtained by an extension of the proof for Theorem 4. In addition to that construction, we also replace the join edges by a set of $2n^2 \log t$ vertices of high weight. The effect on the treewidth is the same, but adding the vertices to the modulator we obtain an independent set instead of a co-cluster.

### 6 Conclusions

We considered different parameterizations for the Treewidth problem and obtained both positive and negative results for the existence of polynomial kernels. Our first positive result, a cubic kernel for Treewidth parameterized by a vertex cover, is interesting as its algorithm largely consists of elements of existing preprocessing heuristics for Treewidth, and thus also sheds some light on the experimentally observed success of these heuristics. Our second positive result, a polynomial kernel for Treewidth parameterized by a feedback vertex set, is not only interesting from a theoretical point of view, but we expect that some of the reduction rules are also of practical value. In that regard it would be interesting to carry out an algorithmic engineering study, and implement (part of) the algorithms. E.g., does the Degree $k+1$ Almost Simplicial Vertex Rule (Rule 9) give significant reductions of instance sizes for practical instances? This could be compared to the experiments reported in [9,18].

Our lower bounds, for distance from cluster respectively co-cluster graphs, rule out polynomial kernels for Treewidth for a number of possibly interesting parameters like distance from cographs or from chordal, interval, or split graphs. We recall also that Treewidth is NP-hard on bipartite graphs (easily seen by subdividing all edges), implying that parameterization by distance $\ell$ from bipartite or perfect graphs does not even permit $O(n^{f(\ell)})$ time algorithms unless $P = NP$; hence also no polynomial kernels.

Apart from improving the kernel sizes, e.g., for parameterization by a feedback vertex set, or giving polynomial lower bounds (e.g., $\Omega(\ell^2)$), it seems interesting whether parameter-wise one can obtain stronger results. For example, is there a polynomial kernel with respect to the size of a modulator to outerplanar graphs or even to planar graphs? For the lower bounds, further research may consider whether they can be extended to distance from a single clique (which is both cluster and co-cluster); complementary to a vertex cover. If true,
then the method needs to generalize proofs using ANDs and ORs of sets of input instances (cf. \cite{4}). If instead there is a polynomial kernel, then the reduction rules have to handle large cliques, the main building block of our lower bounds.

References

1. S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a $k$-tree. SIAM Journal on Algebraic and Discrete Methods, 8:277–284, 1987.
2. A. Becker and D. Geiger. Optimization of Pearl’s method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. Artificial Intelligence, 83:167–188, 1996.
3. H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. SIAM Journal on Computing, 25:1305–1317, 1996.
4. H. L. Bodlaender. Necessary edges in $k$-chordalizations of graphs. Journal of Combinatorial Optimization, 7:283–290, 2003.
5. H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In IWPEC 2009, volume 5917 of LNCS, pages 17–37, 2009.
6. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. Journal of Computer and System Sciences, 75:423–434, 2009.
7. H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Cross-composition: A new technique for kernelization lower bounds. In STACS 2011, pages 165–176, 2011.
8. H. L. Bodlaender and A. M. C. A. Koster. Safe separators for treewidth. Discrete Mathematics, 306:337–350, 2006.
9. H. L. Bodlaender, A. M. C. A. Koster, and F. v. d. Eijkhof. Pre-processing rules for triangulation of probabilistic networks. Computational Intelligence, 21(3):286–305, 2005.
10. H. L. Bodlaender, A. M. C. A. Koster, and T. Wolle. Contraction and treewidth lower bounds. Journal of Graph Algorithms and Applications, 10:5–49, 2006.
11. H. L. Bodlaender and R. H. Möhring. The pathwidth and treewidth of cographs. SIAM Journal on Discrete Mathematics, 6:181–188, 1993.
12. H. L. Bodlaender and U. Rotics. Computing the treewidth and the minimum fill-in with the modular decomposition. Algorithmica, 36:375–408, 2003.
13. V. Bouchitté and I. Todinca. Treewidth and minimum fill-in: Grouping the minimal separators. SIAM Journal on Computing, 31:212–232, 2001.
14. V. Bouchitté and I. Todinca. Listing all potential maximal cliques of a graph. Theoretical Computer Science, 276:17–32, 2002.
15. A. Brandstädt, V. Le, and J. P. Spinrad. Graph Classes, A Survey. SIAM Monographs on Discrete Mathematics and Applications. SIAM, 1999.
16. F. Clautiaux, J. Carlier, A. Moukrim, and S. Négre. New lower and upper bounds for graph treewidth. In WEA 2003, volume 2647 of LNCS, pages 70–80, 2003.
17. F. Clautiaux, A. Moukrim, S. Négre, and J. Carlier. Heuristic and meta-heuristic methods for computing graph treewidth. RAIRO Operations Research, 38:13–26, 2004.
18. F. v. d. Eijkhof, H. L. Bodlaender, and A. M. C. A. Koster. Safe reduction rules for weighted treewidth. Algorithmica, 47:138–158, 2007.
19. L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. Journal of Computer and System Sciences, 77(1):91–106, 2011.
20. J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38:31–45, 2007.

21. J. Lagergren and S. Arnborg. Finding minimal forbidden minors using a finite congruence. In *ICALP 1991*, volume 510 of *LNCS*, pages 532–543, 1991.

22. K. G. Olesen and A. L. Madsen. Maximal prime subgraph decomposition of Bayesian networks. *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, 32:21–31, 2002.
Fig. 2. An example of an application of the Degree $k+1$ Almost Simplicial Vertex Rule, second case ($k = 2$).

A Omitted proofs and reduction rules for Treewidth parameterized by Feedback Vertex Set

In this section we present the omitted pieces for the polynomial kernel for TREEWIDTH PARAMETERIZED BY A FEEDBACK VERTEX SET. We show safeness of Rules 9 through 13 and discuss a polynomial time implementation of Rule 12.

A.1 Correctness of Rule 9

Let us recall Rule 9; together with Rules 7 and 8 it removes all simplicial vertices from the graph.

Rule 9 (Degree $k+1$ Almost Simplicial Vertex) Let $v$ be an almost simplicial vertex with special neighbor $w$, and let the degree of $v$ be exactly $k + 1$.

- If for each vertex $x \in N_G(v) - \{w\}$, there is an edge $\{x, w\}$ in $E$ or a path in $G$ from $x$ to $w$ that avoids $N_G[v] - \{x, w\}$, answer no.
- Otherwise, contract the edge $\{v, w\}$ to a new vertex $x$, obtaining $G'$. If $v \in S$ or $w \in S$, then let $S' := S \setminus \{v, w\} \cup \{x\}$, else let $S' := S$.

An example of an application of the second case of the Degree $k+1$ Almost Simplicial Vertex Rule is given in Fig. 2. It is easy to see that $S'$ is a feedback vertex set for $G'$. Safeness is harder to argue, and needs a few intermediate lemmas. The first one shows that deciding no in the first case is indeed correct.

Lemma 2. Let $v, w$ as in Rule 9. Suppose that for each vertex $x \in N_G(v) - \{w\}$, there is an edge $\{x, w\} \in E$ or a path in $G$ from $x$ to $w$ that does not use any vertex in $N_G[v] - \{x, w\}$. Then the treewidth of $G$ is at least $k + 1$.

Proof. We are building a minor of $G$ in the following way. Consider the connected components of $G[V - N_G[v]]$. Each connected component that has no vertex incident to $w$ is removed. Each connected component with a vertex incident to $w$ is contracted to $w$. We claim that this gives a clique on the vertex set $N_G[v]$, i.e., $G$ has a clique with $k + 2$ vertices as a minor, and hence has treewidth at least $k + 1$. (We use Proposition 1.)
Note that the only edges that are missing in $G[N_G[v]]$ are edges between vertices $x \in N_G(v) - \{w\}$ and $w$. If $\{x, w\} \notin E$, then there is a path from $x$ to $w$ in $G$ that avoids $N_G[v] - \{x, w\}$. Hence, this path must belong to a connected component of $G[V - N_G[v]]$ that is connected to $w$, and thus, the edge $\{x, w\}$ is present in the constructed minor. This shows the claim, and thus the lemma. \hfill \Box

For showing the safeness of the second step, we need a well known lemma on treewidth and clique separators.

**Lemma 3 (See e.g., [22]).** Let $S \subseteq V$ form a clique in $G = (V, E)$. Suppose $V_1, \ldots, V_r$ are the vertex sets of the connected components of $G[V - S]$. Then the treewidth of $G$ equals the maximum treewidth of $G'[S \cup V_i]$ for $i \in \{1, \ldots, r\}$.

Now, we show that the operation in the second case of Rule 9 is safe.

**Lemma 4.** Let $v, w$ as in Rule 9. Suppose that there is a vertex $z \in N_G(v)$ with $z \neq w$ and $\{z, w\} \notin E$, such that no path in $G$ from $z$ to $w$ avoids $N_G[v] - \{z, w\}$. Then the treewidth of $G$ is at most $k$, if and only if the treewidth of the graph $G' = (V', E')$, obtained by contracting $\{v, w\}$ is at most $k$.

**Proof.** We recall the notation of Rule 9: $x$ denotes the vertex obtained by contracting $\{v, w\}$ and $G'$ denotes the obtained graph. If the treewidth of $G$ is at most $k$ then the treewidth of its minor $G'$ is at most $k$ (Proposition 1). It suffices to show the converse.

Suppose the treewidth of $G'$ is at most $k$. Write $Z = N_G(v) - \{w\}$; the set $Z$ is a clique. Suppose $V_1, \ldots, V_r$ are the connected components of $G[V - Z]$. There is one such connected component, say $V_1$, that contains $v$ and $w$. The connected components of $G'[V' - Z]$ are $V'_1, V'_2, \ldots, V'_r$; with $V'_1 = V_1 - \{v, w\} \cup \{x\}$, i.e., $V'_1$ is obtained from $V_1$ by the contraction of edge $\{v, w\}$.

The treewidth of $G'[Z \cup V'_1]$ is at most $k$ and the same is true for $G'[Z \cup V_i]$ for each $i \in \{2, \ldots, r\}$ as the treewidth never increases by taking subgraphs. Further, the treewidth of $G'[Z - \{z\} \cup V'_1]$ is at most $k$ as well.

We claim that the treewidth of $G[Z \cup V_1]$ is at most $k$. Take a tree decomposition of $G'[Z - \{z\} \cup V'_1]$. Replace in each bag an occurrence of $x$ by an occurrence of $w$. As $N_G(v) - \{w\} \cup \{z\}$ is a clique in $G'$, we now have a bag containing $N_G(v) - \{z\}$, say $X_i$. We now add two new bags to the tree decomposition. The first, say $j_1$, contains $N_G[v] - \{z\}$, i.e., $v$ and all its neighbors except $z$. The second, say $j_2$, contains $N_G[v] - \{w\}$, i.e., $v$ and all its neighbors except $w$. Make $j_1$ incident to $i$ and $j_2$ incident to $j_1$ in the tree decomposition. All bags have size at most $k + 1$ (recall that the degree of $v$ is $k + 1$). One can verify that we obtained a tree decomposition of $G[Z \cup V_1]$. E.g., all vertices in $G[Z \cup V_1]$ that are incident to $z$ belong to $N_G[v] - \{w\}$, and they indeed share bag $j_2$ with $z$.

So the treewidth of $G[Z \cup V_1]$ is at most $k$. (See Fig. 3 for an example.)

Lemma 3 applied to $G[Z \cup V_1]$ as well as $G'[Z \cup V_2] = G[Z \cup V_2]$ to $G'[Z \cup V_r] = G[Z \cup V_r]$ now implies that the treewidth of $G$ is at most $k$. \hfill \Box

Safeness of Rule 9 now follows from Lemmas 2 and 4.

**Lemma 5.** Rule 9 is safe.
A.2 Correctness of Rule 10

**Rule 10** Suppose we have a clique-seeing path \((v_0, v_1, \ldots, v_r, v_{r+1})\), that sees a clique \(X\). If \(|X| \leq k - 2\) and

\[
N(v_r) \cap X \subseteq \bigcup_{1 \leq i \leq r-1} N(v_i) \cap X
\]

then contract the edge \(\{v_r, v_{r+1}\}\) into the vertex \(v_{r+1}\), obtaining \(G'\). If \(v_r \in S\), then let \(S' := S \setminus \{v_r\} \cup \{v_{r+1}\}\), else let \(S' := S\).

To show correctness of Rule 10 we use the following two lemmas on treewidth.

**Lemma 6 (Folklore).** Let \(G = (V, E)\) be a graph, and let \((v_1, v_2, \ldots, v_r)\) be a path in \(G\). Let \((\{X_i \mid i \in I\}, T = (I, F))\) be a tree decomposition of \(G\). Suppose \(i_1, i_2, i_3 \in I\) and \(i_2\) is on the path in \(T\) from \(i_1\) to \(i_3\). Suppose \(v_1 \in X_{i_1}\) and \(v_r \in X_{i_3}\). Then \(\{v_1, v_2, \ldots, v_r\} \cap X_{i_2} \neq \emptyset\).

**Lemma 7.** Let \(G = (V, E)\) be a graph, and let \(W_1, \ldots, W_r\) be sets of vertices, such that

- For each \(i\), \(W_i\) induces a connected subgraph of \(G\).
- For all \(i, j\), \(W_i \cap W_j \neq \emptyset\) or \(W_i\) contains a vertex that is adjacent to a vertex in \(W_j\).

Let \((\{X_i \mid i \in I\}, T = (I, F))\) be a tree decomposition of \(G\). Then there exists a bag that contains at least a vertex of each \(W_i\).

Here is a sketch of the proof of Lemma 7. The lemma follows from the Helly property for subtrees of a tree [15, Definition 1.3.4] after observing that for each \(i\), the bags containing a vertex in \(W_i\) form a subtree of \(T\), and for each \(i, j\), there is a bag containing a vertex of \(W_i\) and a vertex of \(W_j\).

**Lemma 8.** Rule 10 is safe.

**Proof.** It is trivial that \(S'\) is a feedback vertex set for \(G'\) and that \(|S'| \leq |S|\). Let \(v_{r+1}\) be the name of the vertex resulting from the contraction. As \(G'\) is a
minor of \( G \), if the treewidth of \( G \) is at most \( k \), then the treewidth of \( G' \) is at most \( k \) (Proposition 1).

Now, suppose that the treewidth of \( G' \) is at most \( k \). The remainder of this proof will show that the treewidth of \( G \) is at most \( k \). Consider a tree decomposition \( (\{X_i \mid i \in I\}, T = (I, F)) \) of \( G' \) of width at most \( k \). Write \( Y = \bigcup_{1 \leq i \leq r-1} N(v_i) \cap X \). Using Lemma 7 (with one set for \( \{v_1, \ldots, v_{r-1}\} \), and one set for each vertex in \( Y \)) it follows that there is a bag \( i_1 \in I \) such that:

- there is a \( j \), \( 1 \leq j \leq r - 1 \) with \( v_j \in X_{i_1} \)
- \( Y \subseteq X_{i_1} \)

As \( \{v_{r+1}\} \cup (N_G(v_{r+1}) \cap X) \) is a clique in \( G' \), there is a bag \( i_2 \) such that \( X_{i_2} \) contains \( \{v_{r+1}\} \cup (N_G(v_{r+1}) \cap X) \). From the definition of tree decomposition, and as \( \{v_{r-1}, v_{r+1}\} \) is an edge in \( G' \), it follows that there must be a bag \( i_3 \) with \( \{v_{r-1}, v_{r+1}\} \in X_{i_3} \). At least one of the following cases must hold:

1. Node \( i_3 \) is on the path in \( T \) from \( i_1 \) to \( i_2 \).
2. Node \( i_1 \) is on the path in \( T \) from \( i_2 \) to \( i_3 \).
3. Node \( i_2 \) is on the path in \( T \) from \( i_1 \) to \( i_3 \).
4. There is a node \( i_4 \) such that \( i_1, i_2, \) and \( i_3 \) are in different subtrees of \( T - \{i_4\} \).

(The order of the cases is chosen in this way to make the proof easier to follow.)

**Case 1: \( i_3 \) is on the path in \( T \) from \( i_1 \) to \( i_2 \)** Note that \( N(v_r) \cap X \subseteq Y \cap (N_G(v_{r+1}) \cap X) \subseteq X_{i_1} \cap X_{i_2} \). So, \( N(v_r) \cap X \subseteq X_{i_3} \).

Now add a new bag \( i' \) to the tree decomposition of \( G' \). Make \( i' \) adjacent to \( i_3 \), and set \( X_{i'} = \{v_{r-1}, v_r, v_{r+1}\} \cup (N_G(v_r) \cap X) \). This is a tree decomposition of \( G \). The new bag has size at most \( 3 + X \leq k + 1 \) so the width is at most \( k \), which concludes this case.

This first case was convenient since it guaranteed the existence of a bag containing \( N(v_r) \), making it possible to construct a tree decomposition for the original graph \( G \) by just appending another bag there containing \( N(v_r) \cup \{v_r\} \).

In the remaining cases we will have to restructure the tree decomposition more severely. The main idea behind each of the remaining cases will be to find a suffix \( (v_{j'}, v_{j'+1}, \ldots, v_{r-1}) \) of the clique-seeing path such that there exists a special bag containing \( v_{j'}, v_{r+1} \), and all the neighbors that the vertices of \( \{v_{j'}, \ldots, v_{r-1}\} \) have in the set \( X \). We then build a new tree decomposition by taking the vertices of the path suffix out of the existing bags, making a path decomposition for this suffix to which we add all the path’s neighbors in \( X \), and attaching this path decomposition to the special bag. So let us now go into details.

**Case 2: \( i_1 \) is on the path in \( T \) from \( i_2 \) to \( i_3 \)** From the definition of tree decomposition, it follows that \( v_{r+1} \in X_{i_1} \).

We now modify the tree decomposition as follows:

- For \( j', j < j' \leq r - 1 \), remove \( v_{j'} \) from all (old) bags that it appears in.
- Add new bags \( i'_j, i'_{j+1}, \ldots, i'_{r-1} \).
Case 3: \(i_2\) is on the path in \(T\) from \(i_1\) to \(i_3\) Consider the path from \(v_j\) to \(v_{r-1}\). As \(v_j \in X_{i_1}\) and \(v_{r-1} \in X_{i_3}\), there must be a \(j'\), \(j \leq j' \leq r-1\) with \(v_{j'} \in X_{i_2}\). Suppose \(j'\) is the largest value \(\leq r-1\) with \(v_{j'} \in X_{i_2}\), i.e., for \(j'', j'' < j'\) \(\leq r-1\), \(v_{j''} \notin X_{i_2}\).

Consider a value \(j''\) with \(j' < j'' \leq r-1\); we will show that any neighbors of \(j''\) in the set \(X\) must be contained in \(X_{i_2}\). Therefore consider a vertex \(w \in N(v_{j''}) \cap X\). There must be some bag \(i_0\) with \(v_{j''} \in X_{i_0}\) and \(w \in X_{i_0}\). There are two cases:

1. \(i_0\) belongs to the same subtree of \(T - \{i_2\}\) as \(i_3\). In this case, we have that \(w \in Y \subseteq X_{i_1}\), and hence, by the properties of tree decompositions, \(w \in X_{i_2}\).
2. \(i_0\) belongs to a different subtree of \(T - \{i_2\}\) as \(i_3\). Now, \(v_{j''} \in X_{i_0}\) and \(v_{r-1} \in X_{i_3}\). From Lemma 6, it follows that there must be a \(j''\) with \(j'' \leq j'' \leq r-1\) with \(v_{j''} \in X_{i_2}\). This contradicts the choice of \(j'\).

As the second case gives a contradiction, the first case must always hold, and thus we have that

\[
\bigcup_{j' < j'' \leq r-1} N(v_{j''}) \cap X \subseteq X_{i_2}.
\]

Write \(Z = \bigcup_{j' < j'' \leq r} N(v_{j''}) \cap X\), and observe carefully that in the definition of \(Z\) the union also includes \(v_r\), which was excluded in the earlier formula. As \(N(v_r) \subseteq N_{G'}(v_{r+1})\) (as we obtained \(v_{r+1}\) from a contraction that involved \(v_r\)), we have that \(Z \subseteq X_{i_2}\).

We now modify the tree decomposition, more or less similarly as in the previous case:

- For \(j'', j' < j'' \leq r-1\), remove \(v_{j''}\) from all (old) bags that it appears in.
- Add new bags \(i'_j, i'_{j'+1}, \ldots, i'_{r-1}\).
- Make bag \(i'_{j'}\) adjacent in \(T\) to \(i_2\).
- For \(j'', j' \leq j'' < r-1\), make bag \(i'_{j''}\) adjacent to bag \(i'_{j''+1}\).
- For \(j'', j' \leq j'' \leq r-1\), set \(X_{i,j''} = Z \cup \{v_{j''}, v_{j''+1}, v_{r+1}\}\).

Similarly as in the previous case, we find that this is a tree decomposition of width at most \(k\) of \(G\).

Case 4: There is a node \(i_4\) such that \(i_1, i_2\) and \(i_3\) belong to different subtrees of \(T - \{i_4\}\) The analysis is more or less similar as in Case 3. We have that \(v_{r+1} \in X_{i_4}\), and \(N(v_r) \cap X \subseteq X_{i_1} \cap X_{i_2}\), hence \(N(v_r) \cap X \subseteq X_{i_4}\).
We have \( v_j \in X_{i_1} \) and \( v_{r-1} \in X_{i_4} \), so there is a \( j', j \leq j' \leq r - 1 \) with \( v_{j'} \in X_{i_4} \). Assume \( j' \) is the maximum value with \( j \leq j' \leq r - 1 \) and \( v_{j'} \in X_{i_4} \).

Consider a \( j'' \) with \( j'' < j' \leq r - 1 \). Since \( v_{r-1} \in X_{i_4} \), all bags containing \( v_{j''} \) must belong to the same subtree of \( T \setminus \{i_4\} \) as \( i_4 \), as otherwise \( i_4 \) must contain a vertex from \( \{v_{j'+1}, v_{j'+2}, \ldots, v_{r-1}\} \). Thus, all neighbors of \( v_{j''} \) in \( X \) belong to some bag in the same subtree of \( T \setminus \{i_4\} \) as \( i_2 \). As \( N(v_{j''}) \cap X \subseteq Y \subseteq X_{i_4} \), we have that \( N(v_{j''}) \cap X \subseteq X_{i_4} \).

We can now use the same construction as in Case 3, except that we attach the new part of the tree decomposition to \( i_4 \).

So, in all cases, we obtain a tree decomposition of \( G \) of width at most \( k \), and thus can conclude that the rule is safe.

\[\square\]

### A.3 Correctness of Rule 12

We recall Rule 12.

#### Rule 12

Suppose \((v_0, v_1, v_2, v_3, v_4)\) is a clique-seen path in \( V \setminus S \) that sees clique \( X \subseteq S \). Suppose \( \{v_1, v_2, v_3\} \cup X \) separates \( v_0 \) from \( v_4 \) and suppose that Rule 11 cannot be applied. Compute the treewidth of \( G[\{v_1, v_2, v_3\} \cup X] \). If it is larger than \( k \), then answer no, otherwise remove \( v_2 \) from \( G \).

#### Lemma 9

Rule 12 is safe.

**Proof.** Let \((v_0, v_1, v_2, v_3, v_4)\) be a clique-seen path in \( V \setminus S \) which sees clique \( X \) such that \( \{v_1, v_2, v_3\} \cup X \) separates \( v_0 \) from \( v_4 \). Before we prove the correctness of the reduction rule, we first establish a claim about the structure of the path if Rule 11 cannot be applied.

**Claim.** The set \( \{v_1\} \cup N(v_1) \cap X \) is a clique separator which separates \( v_0 \) from \( v_2 \).

**Proof.** Since \( X \) is a clique it is easy to see that such a set is a clique. Now let us verify that it separates \( v_0 \) from \( v_2 \).

First assume for a contradiction that \( v_1 \cup X \) does not separate \( v_0 \) from \( v_2 \), and consider a simple path \( P \) from \( v_0 \) to \( v_2 \) avoiding this set. From the fact that \((v_0, v_1, \ldots, v_4)\) is a clique-seen path which sees clique \( X \) it is not hard to see that path \( P \) must use vertex \( v_4 \); but this contradicts the assumption that \( \{v_1, v_2, v_3\} \cup X \) separates \( v_0 \) from \( v_4 \). So we know that \( v_1 \cup X \) separates \( v_0 \) from \( v_2 \).

Since \( X \) is a clique, the set \( \{v_1\} \cup X \) is almost a clique, and since we just established it is a \( v_0 - v_2 \) separator it follows that it is an almost clique separator. Therefore the set \( \{v_1\} \cup X \) contains a minimal almost clique separator \( Z \) which separates \( v_0 \) from \( v_2 \), and \( Z \) must contain \( v_1 \) because of the edges \( \{v_0, v_1\} \), and \( \{v_1, v_2\} \). If \( Z \) is a clique, then \( Z \subseteq \{v_1\} \cup N(v_1) \cap X \) which proves that the superset \( \{v_1\} \cup N(v_1) \cap X \) is also a \( v_0 - v_2 \) separator (and hence a clique separator) and we are done. So assume for contradiction that \( Z \) is not a clique.

Since the clique \( X \) is a subset of the feedback vertex set \( S \), we know that the minimal almost clique separator \( Z \) contains at most 1 vertex which is not from \( S \), and therefore Rule 11 is applicable; a contradiction. This proves the claim.

\[\square\]
It follows easily from the proof of the claim (by symmetry) that \( \{v_3\} \cup N(v_3) \cap X \) is a clique separator which separates \( v_2 \) from \( v_4 \).

**Claim.** For each vertex \( u \notin \{v_1, v_2, v_3\} \cup X \) there is a clique separator \( Z \subseteq \{v_1, v_3\} \cup X \) which separates \( u \) from \( v_2 \).

**Proof.** We first identify a number of vertex sets.

- **A:** the vertices in the same connected component as \( v_0 \) in the graph \( G - (\{v_1\} \cup N(v_1) \cap X) \).
- **B:** the vertices in the same connected component as \( v_4 \) in the graph \( G - (\{v_3\} \cup N(v_3) \cap X) \).
- **C:** all vertices in another connected component as \( v_2 \) in the graph \( G - X \).

We will prove that each vertex \( u \notin \{v_1, v_2, v_3\} \cup X \) is contained in at least one of \( A, B, C \). We will do this by showing that all such vertices \( u \) which are not contained in \( C \), must be contained in \( A \) or \( B \). So assume there is some vertex \( u \) which is not in set \( C \) because it is in the same connected component as \( v_2 \) in \( G - X \). A simple path from \( u \) to \( v_2 \) in \( G - X \) will traverse either \( v_0 \) or \( v_4 \), and there cannot be two simple paths in \( G - X \), one which traverses \( v_0 \) but not \( v_4 \) to get to \( v_2 \) and the other which traverses \( v_4 \) but not \( v_0 \) to get to \( v_2 \), since that would violate the assumption that \( \{v_1, v_2, v_3\} \cup X \) separates \( v_0 \) from \( v_4 \).

Assume for now that a simple path from \( u \) to \( v_2 \) in \( G - X \) traverses \( v_0 \). Then \( u \) is in the same connected component as \( v_0 \) in the graph \( G - (\{v_1, v_2, v_3\} \cup X) \). But this trivially implies that \( u \) is in the same connected component as \( v_0 \) in the graph \( G - (\{v_1\} \cup N(v_1) \cap X) \), and hence \( u \) is contained in the set \( A \).

Similarly we find that if a path from \( u \) to \( v_2 \) in \( G - X \) traverses \( v_4 \), then \( u \) is contained in the set \( B \). This shows that all vertices \( u \notin \{v_1, v_2, v_3\} \cup X \) are contained in at least one of the sets \( A, B, C \).

Now we are in position to prove the claim. It is easy to see that the sets \( \{v_1\} \cup N(v_1) \cap X \), \( \{v_3\} \cup N(v_3) \cap X \), and \( X \) are all cliques, and subsets of \( \{v_1, v_3\} \cup X \). By the previously proved claim, the set \( \{v_1\} \cup N(v_1) \cap X \) separates \( v_0 \) from \( v_2 \), and must therefore separate all vertices in \( A \) from \( v_2 \). Similarly the set \( \{v_3\} \cup N(v_3) \cap X \) separates all vertices in \( B \) from \( v_2 \). And finally the set \( X \) separates all vertices of \( C \) from \( v_2 \), which completes the proof of the claim.

**Claim.** The treewidth of \( G \) is the maximum of the treewidth of \( G - \{v_2\} \) and the treewidth of \( G[\{v_1, v_2, v_3\} \cup X] \).

**Proof.** This follows from applying Lemma \( \star \) repeatedly, using the established claim that all vertices \( u \notin \{v_1, v_2, v_3\} \cup X \) are separated from \( v_2 \) by clique separators that are subsets of \( \{v_1, v_3\} \cup X \).

Clearly, if the treewidth of \( G[\{v_1, v_2, v_3\} \cup X] \) greater than \( k \) it is correct to answer **no**. Otherwise, if its treewidth is at most \( k \), it follows from the claim that the treewidth of \( G \) is at most \( k \) if and only if the treewidth of \( G - \{v_2\} \) is at most \( k \).
We now show that given a path \((v_0, v_1, \ldots, v_r, v_{r+1})\) we can apply Rule 12 in polynomial time, when applicable. Since it is easy to verify whether the neighbors seen by the path form a clique, and whether \(v_0\) and \(v_{r+1}\) are separated, the crucial part is to compute the treewidth of \(G\); this is covered by the following lemma which uses modular decomposition. We briefly recall the relevant notions. If \(G = (V, E)\) is a graph, then a module in \(G\) is a set of vertices \(X \subseteq V\) such that each vertex in \(V \setminus X\) is either adjacent to all vertices of \(X\), or to no vertices of \(X\). A graph is prime if all its modules are empty sets, singleton sets or the set \(V\).

**Lemma 10.** The treewidth of \(G\{v_1, v_2, v_3\} \cup X\) can be computed in polynomial time.

**Proof.** Consider the partition of \(G\{v_1, v_2, v_3\} \cup X\) into sets \(\{v_1\}, \{v_2\}, \{v_3\}\) and the maximal subsets of \(X\) which have the same neighbors among \(v_1, v_2, v_3\). As the clique \(X\) splits into at most 8 subsets, this is a partition of \(G\{v_1, v_2, v_3\} \cup X\) into at most 11 sets, and for every such set \(Q\) all its subsets form a module in the graph \(G\{v_1, v_2, v_3\} \cup X\). This implies that any prime subgraph of \(G\{v_1, v_2, v_3\} \cup X\) contains at most 11 vertices, since two vertices of the same set of the partition form a non-trivial module. From the work in [12], it follows that once we have the modular decomposition of \(G\{v_1, v_2, v_3\} \cup X\), we can compute its treewidth in constant time. Since the modular decomposition can be computed in linear time, this concludes the proof. 

Thus Rule 12 is correct and uses polynomial time. From exhaustive application of the rule, we obtain a connectivity condition on the endpoints of clique-seeing paths which is captured by the following proposition.

**Proposition 3.** If Rule 11 and Rule 12 cannot be applied, and there is a clique-seeing path \((v_0, v_1, \ldots, v_r, v_{r+1})\) with \(r \geq 3\) which sees a clique \(X \subseteq S\) then \(v_0\) and \(v_{r+1}\) are connected via a path that avoids \(v_1, \ldots, v_r\) and \(X\).

**Proof.** Assume \((v_0, v_1, \ldots, v_r, v_{r+1})\) with \(r \geq 3\) sees the clique \(X \subseteq S\), but the set \(\{v_1, \ldots, v_r\} \cup X\) separates \(v_0\) from \(v_{r+1}\). From the structure of the clique-seeing paths it follows that for every \(v_i \in \{v_1, \ldots, v_r\}\) the set \(\{v_i\} \cup X\) separates \(v_0\) from \(v_{r+1}\). So in particular, if we look at the clique-seeing subpath \((v_0, v_1, v_2, v_3, v_4)\) we know that \(\{v_1, v_2, v_3\} \cup X\) separates \(v_0\) from \(v_4\); but then Rule 12 is applicable, a contradiction.

**A.4 Correctness of Rule 13**

**Rule 13** Suppose we have a clique-seeing path \((v_0, \ldots, v_{r+1})\) with \(r \geq 6k + 6\), and suppose Rules 11, 12 are not applicable. Then answer no.

Safeness of this rule is established by the following two lemmas.

**Lemma 11.** Let \((v_0, \ldots, v_{r+1})\) be a clique-seeing path, seeing clique \(X\). Suppose that \(v_0\) and \(v_{r+1}\) belong to the same connected component of \(G - \{v_1, \ldots, v_r\} - X\). Let \(Z\) be the vertex set of the connected component of \(G - \{v_1, \ldots, v_r\} - X\) that contains \(v_0\). If one of the following cases holds, then \(G\) has treewidth at least \(k+1\).
1. $|X| \geq k + 1$.
2. $|X| = k$ and each vertex in $X$ has a neighbor in $Z$.
3. $|X| = k - 1$ and each vertex in $X$ has a neighbor in $Z$, and there is an $\ell$, $1 \leq \ell < r$, such that each vertex in $X$ is incident to at least one vertex in $\{v_1, \ldots, v_\ell\}$, and each vertex in $X$ is incident to at least one vertex in $\{v_{\ell+1}, \ldots, v_r\}$.

**Proof.** In each case $G$ contains a clique of size $k + 2$ as a minor, and hence has treewidth at least $k + 1$, by Proposition 1. In the first case, contract the path to a single vertex; it then forms a clique with the vertices in $X$. In the second case, contract the path to a single vertex, contract $Z$ to a single vertex; these form a clique with the vertices in $X$. In the last case, take $X$, and contract each of the following to a single vertex: all vertices in $Z$, $\{v_1, \ldots, v_\ell\}$, and $\{v_{\ell+1}, \ldots, v_r\}$.

**Lemma 12.** Suppose that none of the Rules $5$–$12$ can be applied to $G$. Suppose that $(v_0, v_1, \ldots, v_{r+1})$ is a clique-seeing path, seeing clique $X$.

1. If $|X| \leq k - 2$ then $r \leq |X| \leq k - 2$.
2. If $|X| = k - 1$ and $r \geq 3k + 3$ then the treewidth of $G$ is greater than $k$.
3. If $|X| = k$ and $r \geq 6k + 6$ then the treewidth of $G$ is greater than $k$.
4. If $|X| \geq k + 1$ then the treewidth of $G$ is greater than $k$.

**Proof.** 1. Suppose $|X| \leq k - 2$. In particular, Rule 10 cannot be applied. Thus for each $i$, $1 < i \leq r$:

$$N(v_i) \cap X \nsubseteq \bigcup_{j=1}^{i-1} N(v_j) \cap X.$$ 

Hence, via induction it follows that $|\bigcup_{j=1}^{i} N(v_j) \cap X| \geq i$, and therefore that $r \leq |\bigcup_{j=1}^{r} N(v_j) \cap X| \leq |X|$.

2. Suppose $|X| = k - 1$ and $r \geq 3k + 3$. Let

$$X_1 = \bigcup_{j=1}^{k+1} N(v_j) \cap X, \quad X_2 = \bigcup_{j=k+2}^{2k+2} N(v_j) \cap X, \quad X_3 = \bigcup_{j=2k+3}^{3k+3} N(v_j) \cap X.$$ 

If $X_1 \neq X$, then $(v_0, v_1, \ldots, v_{k+1}, v_{k+2})$ is a clique-seeing path with $k + 1$ internal vertices and seeing clique $X_1$, with $|X_1| < k - 1$, contradicting Case 1. Similarly, we get a contradiction when $X_2 \neq X$ or $X_3 \neq X$.

Therefore, assume $X_1 = X_2 = X_3 = X$. Consider the path $(v_0, v_1, \ldots, v_{2k+3})$ seeing clique $X$. By Proposition 2 there is a path from $v_0$ to $v_{2k+3}$ that avoids $\{v_1, \ldots, v_{2k+2}\} \cup X$. Hence, the component $Z$ containing $v_0$ in $G - (\{v_1, \ldots, v_{2k+2}\} \cup X)$ also contains $v_{2k+3}$, and hence $\{v_{2k+3}, \ldots, v_{3k+3}\} \subseteq Z$.

Thus by $X_1 = X_2 = X_3 = X$ each vertex of $X$ has a neighbor in each of $\{v_1, \ldots, v_{k+1}\}$, $\{v_{k+2}, \ldots, v_{2k+2}\}$, and $\{v_{2k+3}, \ldots, v_{3k+3}\} \subseteq Z$. Therefore, the conditions of Lemma 11 are fulfilled for $(v_0, \ldots, v_{2k+3})$ and $\ell = k + 1$. It follows that $G$ has treewidth at least $k + 1$.
3. Suppose $|X| = k$ and $r \geq 6k + 6$. Let

$$X_1 = \bigcup_{j=1}^{3k+3} N(v_j) \cap X \quad \text{and} \quad X_2 = \bigcup_{j=3k+4}^{6k+6} N(v_j) \cap X.$$ 

If $X_1 \neq X$, then considering the clique-seeing path $(v_0, v_1, \ldots, v_{3k+3}, v_{3k+4})$ we get a contradiction to Case 1 if $|X_1| \leq k - 2$ or the claim follows from Case 2 if $|X_1| = k - 1$. This can be argued similarly when $X_2 \neq X$.

Therefore, $X_1 = X_2 = X$. Now consider the path $(v_0, \ldots, v_{3k+4})$. By Proposition 2 there is a path from $v_0$ to $v_{3k+4}$ in $G - ((v_1, \ldots, v_{3k+3}) \cup X)$. It follows that the connected component $Z$ of $v_0$ in that graph also contains the vertices $v_{3k+4}, \ldots, v_{6k+6}$. Thus the path $(v_0, \ldots, v_{3k+4})$ fulfills the conditions of the second case of Lemma 11 and we conclude that the treewidth of $G$ is at least $k + 1$.

4. This follows directly from the first case of Lemma 11. \qed

Correctness of Rule 13 follows immediately from the lemma: If a clique-seeing path has at least $6k + 6$ internal vertices, then (assuming that none of the Rules 5–12 can be applied) it sees a clique of size at least $k - 1$ (by Case 1). Cases 2–4 then show that answering no is correct.

B Kernelization lower bounds

Kernelization lower bounds conditioned on $NP \not\subseteq \text{coNP}/\text{poly}$. All kernelization lower bounds in this paper are conditional. Two have the form: Assuming that $NP \not\subseteq \text{coNP}/\text{poly}$ then some parameterization of $\text{TREewidth}$ does not admit a polynomial kernel. For the kernelization lower bounds with respect to this condition, we use the notion of a cross-composition \cite{Bodlaender} which is an extension of the known lower bound techniques based on work of Bodlaender et al. \cite{Bodlaender00} as well as of Fortnow and Santhanam \cite{FortnowSanthanam}. We first give two necessary definitions:

**Definition 1** \cite{Bodlaender}. An equivalence relation $R$ on $\Sigma^*$ is called a polynomial equivalence relation if the following two conditions hold:

1. There is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether $x$ and $y$ belong to the same equivalence class in $(|x| + |y|)^{O(1)}$ time.
2. For any finite set $S \subseteq \Sigma^*$ the equivalence relation $R$ partitions the elements of $S$ into at most $(\max_{x \in S} |x|)^{O(1)}$ classes.

**Definition 2** \cite{Bodlaender}. Let $L \subseteq \Sigma^*$ be a set and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that $L$ cross-composes into $Q$ if there is a polynomial equivalence relation $R$ and an algorithm which, given $t$ strings $x_1, x_2, \ldots, x_t$ belonging to the same equivalence class of $R$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that:

1. $(x^*, k^*) \in Q \iff x_i \in L$ for some $1 \leq i \leq t$, \[ \text{Correctness of Rule 13 follows immediately from the lemma: If a clique-seeing path has at least } 6k + 6 \text{ internal vertices, then (assuming that none of the Rules 5–12 can be applied) it sees a clique of size at least } k - 1 \text{ (by Case 1). Cases 2–4 then show that answering no is correct.} \]

**B Kernelization lower bounds**

Kernelization lower bounds conditioned on $NP \not\subseteq \text{coNP}/\text{poly}$. All kernelization lower bounds in this paper are conditional. Two have the form: Assuming that $NP \not\subseteq \text{coNP}/\text{poly}$ then some parameterization of $\text{TREewidth}$ does not admit a polynomial kernel. For the kernelization lower bounds with respect to this condition, we use the notion of a cross-composition \cite{Bodlaender} which is an extension of the known lower bound techniques based on work of Bodlaender et al. \cite{Bodlaender00} as well as of Fortnow and Santhanam \cite{FortnowSanthanam}. We first give two necessary definitions:

**Definition 1** \cite{Bodlaender}. An equivalence relation $R$ on $\Sigma^*$ is called a polynomial equivalence relation if the following two conditions hold:

1. There is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether $x$ and $y$ belong to the same equivalence class in $(|x| + |y|)^{O(1)}$ time.
2. For any finite set $S \subseteq \Sigma^*$ the equivalence relation $R$ partitions the elements of $S$ into at most $(\max_{x \in S} |x|)^{O(1)}$ classes.

**Definition 2** \cite{Bodlaender}. Let $L \subseteq \Sigma^*$ be a set and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that $L$ cross-composes into $Q$ if there is a polynomial equivalence relation $R$ and an algorithm which, given $t$ strings $x_1, x_2, \ldots, x_t$ belonging to the same equivalence class of $R$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that:

1. $(x^*, k^*) \in Q \iff x_i \in L$ for some $1 \leq i \leq t$, \[ \text{Correctness of Rule 13 follows immediately from the lemma: If a clique-seeing path has at least } 6k + 6 \text{ internal vertices, then (assuming that none of the Rules 5–12 can be applied) it sees a clique of size at least } k - 1 \text{ (by Case 1). Cases 2–4 then show that answering no is correct.} \]
2. \( k^* \) is bounded by a polynomial in \( \max_{i=1}^t |x_i| + \log t \).

It is known that a cross-composition of any NP-hard set into a parameterized problem \( Q \) rules out polynomial kernels for \( Q \) unless \( NP \subseteq \text{coNP}/\text{poly} \).

**Theorem 6** ([7], Corollary 10). If some set \( L \) is NP-hard under Karp reductions and \( L \) cross-composes into the parameterized problem \( Q \) then there is no polynomial kernel for \( Q \) unless \( NP \subseteq \text{coNP}/\text{poly} \).

**Kernelization lower bounds conditioned on the AND-distillation conjecture**

We also give one kernelization lower bound conditioned on the so-called “AND-distillation conjecture”. It is an open problem whether a failure of the AND-distillation conjecture would imply a collapse in classic complexity, but within the parameterized complexity community this conjecture is widely believed to hold. To give kernelization lower bounds conditioned on the AND-distillation conjecture, we need the following definitions.

**Definition 3** ([5,6]). The unparameterized version of a parameterized problem \( Q \subseteq \Sigma^* \times \mathbb{N} \) is the language \( \tilde{Q} = \{ x#1^k \mid (x,k) \in Q \} \), where \( # \) denotes a new character that we add to the alphabet and \( 1 \) is an arbitrary letter in \( \Sigma \).

**Definition 4** ([5,6]). An AND-composition for a parameterized problem \( Q \subseteq \Sigma^* \times \mathbb{N} \) is an algorithm that gets as input a sequence \( ((x_1,k), \ldots, (x_t,k)) \), with each \( (x_i,k) \in \Sigma^* \times \mathbb{N} \), and outputs a pair \( (x',k') \), such that
- the algorithm uses time polynomial in \( \sum_{i=1}^t |x_i| + k \),
- \( k' \) is bounded by a polynomial in \( k \),
- \( (x',k') \in Q \) if and only if \( (x_i,k) \in Q \) for all \( i \) with \( 1 \leq i \leq t \).

If a parameterized problem \( Q \) admits an AND-composition algorithm then it is AND-compositional.

**Theorem 7** ([5]). If \( \tilde{Q} \) is an AND-compositional parameterized problem and \( \tilde{Q} \) is NP-complete, then \( Q \) does not admit a polynomial kernel unless the AND-distillation conjecture fails.

It is implicit in the paper by Bodlaender et al. [6] that if the AND-distillation conjecture fails, then all coNP-complete problems have OR-distillation algorithms.

Finally, we need the following notion to transfer lower bounds from one problem to another.

**Definition 5** ([5]). Let \( P \) and \( Q \) be parameterized problems. We say that \( P \) is polynomial parameter reducible to \( Q \), written \( P \leq_{\text{pp}} Q \), if there exists a polynomial time computable function \( f : \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \times \mathbb{N} \), and a polynomial \( p : \mathbb{N} \rightarrow \mathbb{N} \) such that for all \( (x,k) \in \Sigma^* \times \mathbb{N} \) it holds that (a) \( (x,k) \in P \) if and only if \( (x',k') = f(x,k) \in Q \) and (b) \( k' \leq p(k) \). The function \( f \) is called a polynomial parameter transformation.

It is not hard to see that if \( \tilde{P} \) and \( \tilde{Q} \) are NP-complete, then a polynomial parameter transformation from \( P \) to \( Q \) together with a polynomial kernel for \( Q \) yields a polynomial kernel for \( P \). The contrapositive of this statement can be used to obtain kernelization lower bounds.
B.1 A kernelization lower bound for Treewidth parameterized by distance from cluster graphs

It is interesting to try to draw a borderline for what structural graph parameters there exist polynomial kernelizations. In earlier sections of our paper we showed that the structural parameterization “deletion distance to a forest” (feedback vertex set size) admits a polynomial kernel. In attempts to generalize such a result, one can consider parameters that express the number of vertices to obtain different simple graph classes. In this section, we consider the parameterization “deletion distance to a cluster graph”, and recall that a cluster graph is a vertex-disjoint union of cliques. We will show that this parameterization does not admit a polynomial kernel unless the AND-distillation conjecture fails. This negative result implies directly also the unlikeliness of polynomial kernels for Treewidth with some related parameterizations, e.g., by the number of vertices that must be deleted to obtain a chordal graph. To obtain the result, we show that the restricted problem where the deletion set induces a clique in the graph is AND-compositional and NP-complete.

Therefore we start our investigation with the following problem:

\textbf{Treewidth parameterized by a clique modulator to cluster graphs (TWCMC)}

\textbf{Instance:} A graph $G = (V,E)$, a set $S \subseteq V$ such that $G[S]$ is a clique and $G - S$ is a cluster graph.

\textbf{Parameter:} $\ell := |S|$.

\textbf{Question:} $\text{tw}(G) \leq |S| - 1$?

It is easy to see that TWCMC is in FPT since the problem parameter is directly related to the treewidth that is asked for and $k$-Treewidth is in FPT \cite{DBLP:journals/jcss/ArnborgBGL98}. It is known \cite{DBLP:journals/siamcomp/ArnborgBGL98} that the $k$-Treewidth problem does not admit a polynomial kernel unless the AND-distillation conjecture fails. We will show that TWCMC does not admit a polynomial kernel unless the AND-distillation conjecture fails. Since a polynomial kernel for $k$-Treewidth would imply a polynomial kernel for TWCMC (because of the relationship between the parameter of TWCMC and the target treewidth), our new theorem strengthens the existing result.

The framework for proving conditional kernelization lower bounds through AND-compositions requires us to establish that the unparameterized version of the problem is NP-complete. We prove this by a reduction from the treewidth problem restricted to co-bipartite graphs, which is defined as follows:

\textbf{Treewidth on co-bipartite graphs with given bipartition}

\textbf{Instance:} A co-bipartite graph $G = (V,E)$ with bipartition $V = X \cup Y$ such that $G[X]$ and $G[Y]$ are complete graphs, integer $k$.

\textbf{Question:} $\text{tw}(G) \leq k$?

Arnborg et al. \cite{DBLP:journals/siamcomp/ArnborgBGL98} proved that this problem is NP-complete. The fact that the bipartition is given is not essential: a bipartition of a co-bipartite graph can be found in polynomial time by using the standard bipartition algorithm on the complement of the graph.
Theorem 8 (1, Theorem 3.3). Treewidth on co-bipartite graphs with given bipartition is NP-complete.

With this information we can prove that the unparameterized variant of TWCMC is NP-complete.

Lemma 13. The unparameterized variant of Treewidth parameterized by a clique modulator to cluster graphs is NP-complete.

Proof. Membership in NP is trivial by giving a tree decomposition. To prove NP-hardness we give a reduction from Treewidth on co-bipartite graphs with given bipartition (TWC). Let \((G, X, Y, k)\) be an instance of TWC, and assume without loss of generality that \(|X| \geq |Y|\). If \(k < |X| - 1\) then the answer to the instance must be no, since \(X\) induces a clique in \(G\) and the treewidth of a graph is at least the size of the largest clique minus one. If \(k = |X| - 1\) then if we set \(S := X\) we find that \(G - S\) is a cluster graph (in fact a single clique) and \(G[S]\) is a clique; hence by definition the instance \((G, X, Y, k)\) of TWC is then equivalent to the instance \((G, S)\) of TWCMC. If \(k > |X| - 1\) then consider the co-bipartite graph \(G'\) with bipartition \(X', Y'\) obtained from \(G\) by adding a single vertex \(v\) to the partite set \(X\) and making it adjacent to all other vertices in \(X\). By definition the new vertex \(v\) is simplicial, and by [1] Proposition 1 we know that \(\text{tw}(G') = \max(\deg_{G'}(v), \text{tw}(G' - \{v\})) = \text{tw}(G)\). Hence if \(k > |X| - 1\) then \(\deg_{G'}(v) = |X| \leq k\) which implies \(\text{tw}(G') \leq k \Leftrightarrow \text{tw}(G) \leq k\). By repeatedly adding simplicial vertices to the partition \(X\) we obtain an equivalent instance of TWC for which \(k = |X| - 1\), and by the argument above we then find an equivalent instance of TWCMC. Since this construction takes polynomial time this proves that TWCMC is NP-complete.

To give a kernelization lower bound for TWCMC it now remains to prove that the problem is AND-compositional.

Lemma 14. Treewidth parameterized by a clique modulator to cluster graphs is AND-compositional.

Proof. Given \(t\) instances \((G_1, S_1), (G_2, S_2), \ldots, (G_t, S_t)\) of TWCMC with the same parameter value \(|S_i|\) (so \(|S_1| = |S_2| = \ldots = |S_t| = \ell\)) we show how to construct in polynomial time an instance \((G', S')\) of TWCMC which is yes if and only if all of the input instances are yes, with \(|S'| = \ell|\).

The graph \(G'\) is built through the following procedure. For each \(i \in [t]\) label the vertices of \(S_i\) arbitrarily with unique labels in the range 1 to \(\ell\). Initialize \(G'\) as the disjoint union of the input graphs \(G_i\) for \(i \in [t]\). As the next step we identify the separate sets \(S_i\) for \(i \in [t]\) into one set \(S'\). For each \(j \in [t]\), there is exactly one vertex in each \(S_i\) which is labeled \(j\). Identify all these vertices labeled \(j\) into one vertex, and place it in the set \(S'\). Output the instance \((G', S')\).

Since \(S'\) contains one vertex for each \(j \in [t]\) it follows that \(|S'| = \ell|\), so the parameter of the output instance equals the parameter value of the input instances. Since each \(S_i\) for \(i \in [t]\) induces a clique in the corresponding graph \(G_i\) by the definition of TWCMC, it follows that \(S'\) induces a clique in \(G'\). Because we
constructed $G'$ by identifying the clique modulators of the disjoint union of the input instances, the output graph $G'$ encodes all the input graphs simultaneously because $G'[V(G_i - S_i) \cup S']$ is isomorphic to $G_i$ for all $i \in [t]$.

It is easy to see that the instance $(G', S')$ can be constructed in polynomial time. We have already seen that the output parameter is appropriately bounded; it remains to prove that the output instance is equivalent to the AND of the input parameters. Let us consider the treewidth of the graph $(G', S')$. The vertex set $S'$ induces a clique in $G'$, and in fact it is a clique separator for the graph. Let $C_1, \ldots, C_m$ be the vertex sets of the connected components of $G' - S'$. By Lemma 3 we know that $\text{tw}(G') = \max_{1 \leq i \leq m} \text{tw}(G'[S' \cup C_i])$. With this information we prove that the output instance expresses the AND of the input instances. To simplify the proof we actually prove the complementary statement: $\text{tw}(G') > k \iff \exists i \in [t] : \text{tw}(G_i) > k$.

$(\Rightarrow)$ Assume that $\text{tw}(G') > k$. By the above observation about the treewidth of $G'$ this implies that there is a connected component $C_j$ of $G' - S'$ such that $\text{tw}(G'[S' \cup C_j]) > k$. But by construction of the graph $G'$, for each connected component $C_j$ of $G' - S'$ the vertices of $C_j$ must belong to exactly one input graph $G_i$. But since $G'[V(G_i - S_i) \cup S']$ is isomorphic to $G_i$ by our earlier observation, and since $G'[S' \cup C_j]$ is a subgraph of $G'[V(G_i - S_i) \cup S']$, this proves that a subgraph of $G_i$ has treewidth more than $k$. Then $G_i$ has treewidth more than $k$, which proves the claim in this direction.

$(\Leftarrow)$ Assume that $\text{tw}(G_i) > k$ for some index $i^* \in [t]$. Since the set $S_{i^*}$ is a clique separator for graph $G_{i^*}$, by Lemma 3 there is a connected component $C_j$ of $G_{i^*} - S_{i^*}$ such that $\text{tw}(G_{i^*}[C_j \cup S_{i^*}]) > k$. But by construction of $G'$ the vertex set of $C_j$ also exists in graph $G'$, and in fact $G_{i^*}[C_j \cup S_{i^*}]$ is isomorphic to $G'[C_j \cup S']$. But this proves that a subgraph of $G'$ has treewidth more than $k$, which implies $G'$ has treewidth more than $k$.

The derived relationship between the input and output instances show that the output instance is no if and only if one of the input instances is no; hence the output is yes if and only if all of the inputs are yes. Since we have verified that the algorithm satisfies all conditions of an AND-composition algorithm stated in Definition 3 this proves the lemma.

Since treewidth parameterized by a clique modulator to cluster graphs is AND-compositional (Lemma 14) and its unparameterized variant is NP-complete (Lemma 13) we obtain the following result by applying Theorem 7.

**Theorem 9.** Unless the AND-distillation conjecture fails and all coNP-complete problems have OR-distillation algorithms, Treewidth parameterized by a clique modulator to cluster graphs does not admit a polynomial kernel.

Recall that the original aim of this section was to study treewidth with a structural parameterization, which measures how many vertex deletions the graph is away from a cluster graph. Since we have identified an NP-complete special case of this problem which does not have a polynomial kernel (unless the AND-distillation conjecture fails), under the same assumption this also rules
out polynomial kernels for the more general problem by applying standard lower bound tools; in particular it follows from contraposition on Theorem 8 of [5]. For completeness we state the general problem and the kernelization bound.

**Treewidth parameterized by target treewidth plus the size of a given modulator to cluster graphs**

**Instance:** A graph $G = (V, E)$, an integer $k$, and a set $S \subseteq V$ of size $\ell$ such that $G - S$ is a cluster graph.

**Parameter:** $\ell + k$.

**Question:** $\text{tw}(G) \leq k$?

We are now in position to prove the main result of this section.

**Theorem 10.** Treewidth parameterized by target treewidth plus the size of a given modulator to cluster graphs does not admit a polynomial kernel unless the AND-distillation conjecture fails and all coNP-complete problems have OR-distillation algorithms.

**Proof.** The main part of the proof consists of a polynomial parameter transformation from Treewidth parameterized by a clique modulator to cluster graphs (TWCMC) to Treewidth parameterized by target treewidth plus the size of a given modulator to cluster graphs (TWTWMC). So following Definition [5] consider the trivial polynomial parameter transformation which maps an instance $(G, S)$ of TWCMC which asks whether the treewidth of $G$ is at most $|S| - 1$, to the output instance $(G', S', k')$ of TWTWMC with $G' := G$, $S' := S$ and $k' := |S| - 1$. Since the parameter of the input instance is $|S|$, it is easy to see that the parameter $|S'| + k'$ of the output instance is bounded polynomially by the parameter of the input instance. The equivalence of the instances $(G, S)$ and $(G', S', k')$ follows directly from the problem definitions. The existence of a polynomial parameter transformation from TWCMC to TWTWMC proves, together with the fact that the unparameterized versions of these problems are easily seen to be NP-complete, that TWTWMC does not admit a polynomial kernel under the same conditions as TWCMC by applying a standard theorem from the theory of kernelization lower bounds [5, Theorem 8]. Together with Theorem 9, this concludes the proof. □

To justify our kernel lower-bound, we give a brief argument to show fixed-parameter tractability of the problem, even when only parameterized by the deletion distance to a cluster graph (i.e. parameterized by $\ell$ instead of $\ell + k$).

**Proposition 4.** Treewidth parameterized by a modulator to cluster graphs is fixed-parameter tractable.

**Proof.** Consider vertices $v, w$ in $G = (V, E)$, with $G[V - S]$ a cluster graph. We claim that the number of minimal $v, w$-separators is bounded by $4^\ell$.

Suppose that $Q$ is a minimal $v, w$-separator. Partition the vertices from $S$ into four sets: $A$ are all vertices in $S$ that belong to the same connected component
as \( v \) in \( G[V - Q] \). \( B \) are all vertices in \( S \) that belong to the same connected component as \( w \) in \( G[V - Q] \). \( C = S \cap Q \), and \( D = S - (A \cup B \cup C) \). Now, the choice of \( v, w \), and the partition of \( S \) into \( A, B, C, \) and \( D \) fixes precisely which vertices in \( V - S \) belong to \( Q \). As we have \( 4^\ell \) partitions of \( S \), this gives a bound of at most \( 4^\ell \) minimal \( v,w \)-separators, and thus we have at most \( 4^\ell \cdot n^2 \) minimal separators in \( G \).

Bouchitté and Todinca [13,14] have shown that the treewidth can be computed in time, polynomial in the number of minimal separators, i.e., polynomial in \( 4^\ell \cdot n^2 \).

\[ \square \]

### B.2 A kernelization lower bound for Treewidth parameterized by distance from co-cluster graphs

In this section we show that Treewidth parameterized by a modulator to co-cluster graphs (TWMCC) does not admit a polynomial kernel, unless \( \text{NP} \subseteq \text{coNP}/\text{poly} \). The result nicely complements the result of the previous section, and its proof is a step-up for a similar proof for Weighted Treewidth parameterized by a vertex cover in the next section. Formally, we define the problem as follows:

**Treewidth parameterized by a modulator to co-clusters**

**Instance:** A graph \( G = (V,E) \), an integer \( k \), and a set \( S \subseteq V \) such that \( G - S \) is a co-cluster graph.

**Parameter:** \( \ell := |S| \).

**Question:** \( \text{tw}(G) \leq k? \)

We emphasize that giving the modulator as a part of the input can only make the problem easier, suitable for proving a lower bound. Furthermore, there is a straightforward 3-approximation for the minimum modulator to a co-cluster since co-clusters are exactly the \( (K_2 \cup K_1) \)-free graphs (via 3-Hitting Set). Fixed-parameter tractability is proven in Proposition 5.

Our lower bound proof uses the following lemma due to Bodlaender and Möhring [11]. By \( G_1 \otimes G_2 \) we denote the join of two graphs, obtained by taking their disjoint union and adding all edges between the two graphs.

**Lemma 15** ([11]). For any two graphs \( G_1 \) and \( G_2 \) it holds that

\[
\text{tw}(G_1 \otimes G_2) = \min(\text{tw}(G_1) + |V(G_2)|, \text{tw}(G_2) + |V(G_1)|).
\]

From Lemma 15, we directly obtain the following corollary.

**Corollary 3.** For any \( t \) graphs \( G_1, \ldots, G_t \) each on \( n \) vertices it holds that

\[
\text{tw}(G_1 \otimes \ldots \otimes G_t) = (t - 1) \cdot n + \min_{i \in [t]} \text{tw}(G_i).
\]

Additionally, we also use the following lemma (based on [3]); the result was already used for the correctness of Rule 3.
Lemma 16. Let $G$ be a graph, let $k$ be an integer, and let $u, v$ be two vertices of $G$ that have at least $k + 1$ shared neighbors. Then in any tree decomposition of $G$ of width at most $k$ there must be a bag containing both $u$ and $v$.

The same is true for tree decompositions of weighted width at most $k$, if $u$ and $v$ have shared neighbors with a total weight of at least $k + 1$.

For the kernelization lower bound for TWMCC we give a cross-composition from the unparameterized Treewidth problem.

Theorem 11. Treewidth parameterized by a modulator to co-cluster graphs does not admit a polynomial kernelization unless $NP \subseteq \text{coNP}/\text{poly}$ (which would imply a collapse of the polynomial hierarchy its third level).

Proof. We give a cross-composition from Treewidth to TWMCC. We define a simple polynomial equivalence relation $R$ such that two instances $x = (G, k)$ and $x' = (G', k')$ of Treewidth are equivalent if the graphs $G$ and $G'$ have the same numbers $n$ and $m$ of vertices and edges, respectively, and if $k = k' \leq n$.

For technical reasons, all instances where $k$ exceeds the number of vertices also form one equivalence class, but we will tacitly ignore these trivial instances in the following (it is easy to see that the presence of such an instance makes a cross-composition trivial). Clearly, equivalence under $R$ can be checked in polynomial time and a set of instances each of size at most $N$ is partitioned into at most $N^3 + 1$ equivalence classes.

Let $x_1, \ldots, x_t$ be $t$ instances of Treewidth which are equivalent under $R$, each instance $x_i$ asking for a tree decomposition of width at most $k$ for some graph $G_i$. Let $n$ and $m$ denote the number of vertices and edges in graphs $G_i$, W.l.o.g. we assume that $t \geq n$, or otherwise make sufficiently many copies of one of the instances. This will at most increase the input size by a factor of $n$.

We construct a graph $G'$ as follows; together with some integer $k'$ and a modulator to a co-cluster this will constitute the cross-composed instance:

1. Add $t$ independent sets $\mathcal{I}_1, \ldots, \mathcal{I}_t$ each on $n$ vertices to $G'$. These represent the vertex sets of the graphs $G_i$; let each vertex of $\mathcal{I}_j$ correspond directly to a unique vertex of $G_j$. Add all edges between different independent sets, i.e., at this point we have $G' = \mathcal{I}_1 \otimes \ldots \otimes \mathcal{I}_t$.
2. Add $m$ cliques $\mathcal{K}_1, \ldots, \mathcal{K}_m$ each on $n + 1$ vertices to $G'$.
3. For each edge of $G_i$ connect the corresponding vertices in $\mathcal{I}_i$ to all vertices of one of the $m$ cliques; use a different clique for each edge of $G_i$ and repeat this for all graphs $G_i$. Finally each clique $\mathcal{K}_i$ will have a pair of neighbors in each independent set $\mathcal{I}_j$ corresponding to an edge of $G_j$.

We define the cross-composed instance $x'$ as $(G', k', \bigcup \mathcal{K}_i)$, where the target treewidth is $k' = (t - 1) \cdot n + k$. We observe that $G' - \bigcup \mathcal{K}_i = \mathcal{I}_1 \otimes \ldots \otimes \mathcal{I}_t$ is a co-cluster graph, and that the size of the modulator $\bigcup \mathcal{K}_i$, i.e., the parameter value of $x'$, is polynomial in $\max_i |x_i|$, fulfilling the definition of a cross-composition. Finally, it is easy to see that the construction can be performed in time polynomial in $\sum_i |x_i| + k$. 
For correctness of the cross-composition it suffices to show that
\[ \text{tw}(G') \leq k' = (t - 1) \cdot n + k \iff \exists i \in [t] : \text{tw}(G_i) \leq k. \]

(\(\Leftarrow\)) Let \(i \in [t]\) such that \(\text{tw}(G_i) \leq k\) and let \(\mathcal{T}\) be a tree decomposition of \(G_i\) of width at most \(k\); we construct a tree decomposition \(\mathcal{T}'\) for \(G'\) of width at most \((t - 1) \cdot n + k\).

Let \(\mathcal{I}_i := \bigcup_{j \neq i} \mathcal{I}_j\), i.e., the set containing the \((t - 1) \cdot n\) vertices of all independent sets except \(\mathcal{I}_i\). We add the set \(\mathcal{I}_i\) to all bags of \(\mathcal{T}\) obtaining \(\mathcal{T}'\). Now, let \(\{p, q\}\) be any edge of \(G_i\). There must be a bag of \(\mathcal{T}\) that contains \(p\) and \(q\), and hence there is a bag of \(\mathcal{T}'\) that contains \(\mathcal{I}_i \cup \{p, q\}\). Thus for each clique \(\mathcal{K}_j\) of \(G'\) with neighbors \(p, q \in \mathcal{I}_i\) there is a bag of \(\mathcal{T}'\) that contains all neighbors of \(\mathcal{K}_j\), since all other neighbors are in \(\mathcal{I}_i\). Hence to that bag we may append a new bag of size \((n + 1) + 2t\) containing all vertices of the clique as well as its \(2t\) neighbors in \(\mathcal{I}_1 \cup \ldots \cup \mathcal{I}_t\). This way we obtain a tree decomposition for \(G'\) of width
\[ \max\{(n + 1) + 2t - 1, (t - 1) \cdot n + k\}. \]

Since \(t \geq n\) (and ignoring the case that \(n < 4\) in which the brute force solution of all instances \(x_i\) suffices) this tree decomposition for \(G'\) has width \((t - 1) \cdot n + k\).

(\(\Rightarrow\)) Let \(\text{tw}(G') \leq k'\) and let \(\mathcal{T}\) be a tree decomposition of \(G'\) of width at most \(\text{tw}(G') \leq k' = (t - 1) \cdot n + k\). We will first show how to get a tree decomposition for \(G'' := G_1 \otimes \ldots \otimes G_t\) of at most the same width. Second, we will apply Corollary \[3\] to get that \(\exists i \in [t] : \text{tw}(G_i) \leq k\).

We remark that \(G'\) and \(G''\) are very similar in that they contain the join of the \(t\) \(n\)-vertex independent sets as (not necessarily induced) subgraphs. It suffices, therefore, to ensure that the edges inside the copies of graphs \(G_i\) in \(G''\) are properly represented in \(\mathcal{T}\).

Let \(\{p, q\}\) be an edge of any graph \(G_i\) and consider the corresponding vertices in \(\mathcal{I}_i\) in \(G'\). The vertices \(p\) and \(q\) have \((t - 1) \cdot n\) shared neighbors in the other independent sets (on account of the join edges). Furthermore, both vertices are adjacent to one of the \(n + 1\)-vertex cliques, say \(\mathcal{K}_j\), by construction of \(G'\). Hence, they have at least \(t \cdot n + 1 \geq \text{tw}(G') + 1\) shared neighbors; it can be easily seen that the treewidth of \(G'\) is at most \(t \cdot n - 1\): Indeed, making one bag of size \(t \cdot n\) containing the vertices of the independent sets \(\mathcal{I}_1, \ldots, \mathcal{I}_t\) and adding an adjacent bag for each clique \(\mathcal{K}_j\), containing the vertices of the clique and its 2\(t\) neighbors, suffices to get a tree decomposition of width \(t \cdot n - 1\) (we note that the size of the bags for the cliques is \(n + 1 + 2t \leq t \cdot n\) when \(t \geq n \geq 4\)). By Lemma \[10\] this implies that there must be a bag of \(\mathcal{T}\) that contains both \(p\) and \(q\).

The previous argumentation holds for all edges of any graph \(G_i\). Therefore, by discarding the vertices of the cliques \(\mathcal{K}_1, \ldots, \mathcal{K}_m\) we obtain a tree decomposition \(\mathcal{T}'\) for \(G''\), since the edges of graphs \(G_i\) are the only ones that are present in \(G''\) but not in \(G'\). Clearly, \(\mathcal{T}'\) has width at most \(\text{tw}(G')\).

By Corollary \[3\] this implies that
\[ (t - 1) \cdot n + k \geq \text{tw}(G') \geq \text{tw}(G'') = (t - 1) \cdot n + \min_i \text{tw}(G_i). \]
Let $G$ be a graph which has deletion distance $\ell$ to a co-cluster graph. Since co-cluster graphs are exactly the $P_3$-free graphs, we can find a minimum-size set of vertices $X$ such that $G - X$ is a co-cluster graph by applying a bounded-depth branching algorithm, finding a $P_3$ in the graph and branching on which of the three vertices to put into the set. Such a procedure can easily be implemented to run in $O(3^\ell n^2)$ time.

Having found the set $S$, it is easy to identify the (complements of) clusters in the graph $G - X$; these are cliques in the complement of $G - X$. Let $C_1, \ldots, C_r$ be the co-clusters of $G - X$. Observe that in any chordal supergraph $G'$ of $G$, there can be at most one co-cluster $C_i$ which does not induce a clique in $G'$. To see this, suppose that there are two co-clusters $C_i$ and $C_j$ which do not induce a clique in $G'$; then there is a non-edge $\{u_1, v_1\} \in G'[C_i]$ and a non-edge $\{u_2, v_2\} \in G'[C_j]$. But then $(u_1, u_2, v_1, v_2)$ must be a chordless cycle of length 4 in $G'$ contradicting the assumption that $G'$ is chordal; the existence of the edges $\{u_1, u_2\}, \{u_2, v_1\}, \{v_1, v_2\}, \{v_2, u_1\}$ follows from the fact that the two vertices of each pair belong to different co-clusters. Hence in any chordal supergraph of $G$ we know there is at most one co-cluster which does not induce a clique. It follows from this observation that the treewidth of $G$ is the minimum of the treewidth of a graph in $G_1, \ldots, G_r$, where the graph $G_i$ is obtained from $G$ by completing all clusters except $C_i$ into a clique. Hence it suffices to show how to compute the treewidth of such a graph $G_i$ in FPT time. Observe that a graph $G_i$ has a restricted structure: after completing all the co-clusters except $G_i$ into a clique, we actually have that $G_i[\bigcup_{j \neq i} C_j]$ is a clique. Hence a graph $G_i$ composes into the co-cluster $C_i$ (which is an independent set), the deletion set $X$, and a clique $\bigcup_{j \neq i} C_j$.

We will define a partition of the vertex set of $G_i$ into sets of vertices which have similar neighborhoods. Take a singleton set $\{v\}$ for each vertex $v \in X$. Partition the vertices of $C_i$ into at most $2^\ell$ sets, putting vertices of $C_i$ which have the same set of neighbors in $X$ into the same set. In the same way, partition the vertices of $\bigcup_{j \neq i} C_j$ into at most $2^\ell$ sets depending on their neighborhood in the set $X$. It is not hard to verify that we end up with a partition into at most $2 \cdot 2^\ell + \ell$ sets. For every set $Q$ in the partition, for all subsets $Q' \subset Q$ the vertices of $Q'$ have the same neighbors outside $Q$. This follows from the fact that $C_i$ is an independent set, $\bigcup_{j \neq i} C_j$ is a clique and all vertices of $C_i$ are adjacent to all vertices of $\bigcup_{j \neq i} C_j$. This partition shows that if we consider a modular decomposition of $G_i$, any prime subgraph of $G_i$ has at most $2 \cdot 2^\ell + \ell$ vertices, since two vertices from the same set of our partition form a non-trivial module. By a result of Bodlaender and Rotics [12, Corollary 13] the treewidth...
problem can be solved in \( f(n,m) + n + m \) time on graph \( G_i \), where \( f(n,m) \) is the time to compute the Weighted Tree Width of a prime subgraph of \( G_i \) on \( n \) vertices and \( m \) edges. Since we have shown that the number of vertices in such a prime subgraph is bounded by a function in \( \ell \), it follows that \( f(n,m) \) can be bounded by a function depending only on \( \ell \) (any brute-force algorithm to solve Weighted Tree Width will suffice), implying that the treewidth of \( G_i \) can be computed in \( f(\ell)n^{O(1)} \) time. Since we can compute the treewidth of \( G \) by computing the treewidth of all graphs \( G_i \) for \( 1 \leq i \leq r \), and \( r \) is bounded by the number of vertices in \( G \), this implies that Treewidth parameterized by deletion distance to a co-cluster graph is fixed-parameter tractable. □

### B.3 A kernelization lower bound for Weighted Treewidth parameterized by a vertex cover

The Weighted Treewidth problem asks for a given vertex weighted graph \( G \) and integer \( k \), if the weighted treewidth of \( G \) is at most \( k \). Preprocessing heuristics for the problem were considered in [18]. We consider a parameterized version of this problem, where we choose the size of a vertex cover as the parameter:

**Weighted Treewidth parameterized by a vertex cover**

**Instance:** A graph \( G = (V,E) \), a weight function \( w : V \to \mathbb{N} \), an integer \( k \), and a vertex cover \( S \subseteq V \) of \( G \).

**Parameter:** \( \ell := |S| \).

**Question:** Does \( G \) have a tree decomposition of weighted width at most \( k \) with respect to \( w \)?

We show that **Weighted Treewidth parameterized by a vertex cover** (WTWVC) does not admit a polynomial kernelization unless the polynomial hierarchy collapses. To this end, we will extend the construction which we used for the cross-composition of Treewidth into Treewidth parameterized by a modulator from co-clusters.

**Theorem 12.** Weighted Treewidth parameterized by a vertex cover does not admit a polynomial kernelization unless \( NP \subseteq \text{coNP}/\text{poly} \).

**Proof.** We prove the theorem by giving a cross-composition from the classic (unweighted) Treewidth problem to **Weighted Treewidth parameterized by a vertex cover** (WTWVC). We reuse the polynomial equivalence relation from Theorem 11. Accordingly, let \( x_1, \ldots, x_t \) be \( t \) instances of Treewidth, each instance \( x_i \) asking whether some graph \( G_i \) on \( n \) vertices and \( m \) edges has a tree decomposition of (unweighted) width at most \( k \). We will create a cross-composed instance \( x' \) of WTWVC which is **yes** if and only if one of the instances \( x_i \) is **yes**.

For a start we repeat the construction from the proof of Theorem 11. We obtain a graph \( G' \) consisting of \( I_1 \otimes \ldots \otimes I_t \), i.e., a join of \( t \) independent sets each on \( n \) vertices, as well as \( m \) cliques \( K_1, \ldots, K_m \) each on \( n + 1 \) vertices. We have that \( \text{tw}(G') \leq k' = (t-1) \cdot n + k \) if and only if at least one of the graphs \( G_i \) has treewidth at most \( k \).
We will now construct a graph $G''$ from $G'$ by removing the edges between the independent sets and replacing them by new vertices of high weight. The new vertices together with the vertices of the cliques $K_1, \ldots, K_m$ will constitute a vertex cover of $G''$:

- Remove all edges between the vertices of $I_1, \ldots, I_t$; their union now induces an independent set.
- Add new vertices $v_{p,i,j}$ and $v'_{p,i,j}$ for all $p \in [\log t]$ and $i, j \in [n]$ and let their weight be $k' + 1 - t$. Connect each $v_{p,i,j}$ as follows:
  - Make an edge between $v_{p,i,j}$ and the $i$th vertex of all independent sets $I_r$ for which the $p$th bit of $r$ is 0.
  - Make an edge between $v_{p,i,j}$ and the $j$th vertex of all independent sets $I_r$ for which the $p$th bit of $r$ is 1.

Do the same for the $v'_{p,i,j}$ vertices.

There are $2\log t \cdot n^2$ such vertices and their weights can be encoded in a number of bits that is polynomial in $n + \log t$.

Let us first observe that the set of all vertices $v_{p,i,j}$ and $v'_{p,i,j}$ together with the $m \cdot (n + 1)$ vertices of the $m$ cliques $K_1, \ldots, K_m$ constitute a vertex cover of $G''$ of size $2\log t \cdot n^2 + m \cdot (n + 1)$, i.e., polynomial in $n + \log t$; we denote this vertex cover by $S$. As the composed instance we return $(G'', w, k', S)$ where $w$ assigns weight one to all vertices except $v_{p,i,j}$ and $v'_{p,i,j}$ which are assigned weight $k' + 1 - t$.

For correctness of the cross-composition it suffices to show that $G''$ has treewidth at most $k'$ if and only if at least one graph $G_i$ has treewidth at most $k$:

($\leftarrow\rightarrow$) : Assume that some graph $G_i$ has treewidth at most $k$ and let $T$ be a matching tree decomposition. We know from the proof of Theorem 11 how to construct a tree decomposition $T'$ of $G'$ by adding the set $\tilde{I}_i = \bigcup_{j \neq i} I_j$ to each bag of $T$ and adding additional bags for the cliques $K_1, \ldots, K_m$. It suffices to check that we can extend this tree decomposition to include also the $v_{i,j}$ and $v'_{i,j}$ vertices.

Consider some vertex $v_{p,a,b}$ and let $u$ be its neighbor in $I_j$ (which corresponds to the graph $G_j$ chosen above). There is a bag of $T$ that contains $u$. Hence, there is a bag of $T'$ that contains $\tilde{I}_i \cup \{u\}$ and, therefore, all neighbors of $v_{p,a,b}$. Adjacent to this bag we may add a new bag containing $v_{p,a,b}$ and its $t$ neighbors in the independent sets; its total weight is $k' + 1 - t + t = k' + 1$. Thus, applying this procedure for all vertices $v_{i,j}$ and $v'_{i,j}$, does not increase the treewidth above $k'$. We obtain a tree decomposition for $G''$ of width at most $k'$, as claimed.

($\Rightarrow$) : Now, assume that the treewidth of $G''$ is at most $k''$ and let $T$ be a tree decomposition of $G''$ of width at most $k'$. Our strategy is to construct a tree decomposition for $G'$ of at most the same width. From the proof of Theorem 11 it will then follow that at least one graph $G_i$ has treewidth at most $k$.

Note that the difference between $G'$ and $G''$ lies in the presence of the join edges between the independent sets in $G'$ versus the additional vertices $v_{i,j}$ in $G''$.

We will show that for any two vertices $i, j$ from different independent sets $I_i$ and $I_s$, $r \neq s$, there is a bag of $T$ that contains both $i$ and $j$ (here we identify the
vertices directly with their number in the independent set). Since \( r \neq s \) there must be at least one position \( p \in \lceil \log t \rceil \) where the binary expansions of \( r \) and \( s \) differ. Thus either \( v_{p,i,j} \) or \( v'_{p,i,j} \) (depending on the \( p \)th positions of \( r \) and \( s \)) must be adjacent to both \( i \) and \( j \). W.l.o.g. \( v_{p,i,j} \) (hence also \( v'_{p,i,j} \)) is adjacent to \( i \) and \( j \) in \( G'' \). Thus \( i \) and \( j \) have shared neighbors with a total weight exceeding the width of \( T \):

\[
2 \cdot (k' + 1 - t) = k' + (k' + 2 - 2t) = k' + ((t - 1) \cdot n + k - 2(t - 1)) > k',
\]

when \( n \geq 3 \). Hence, in \( T \) there must be a bag that contains both \( i \) and \( j \), by Lemma \[16\]. Thus for any two vertices from different independent sets there must be a bag of \( T \) that contains both. Since edges between such pairs of vertices are the only edges that we find in \( G' \) but not in \( G'' \), removing all vertices \( v_{p,i,j} \) and \( v'_{p,i,j} \) yields a tree decomposition for \( G' \) of width at most \( k' \). Thus, following the proof of Theorem \[11\] at least one graph \( G_i \) has treewidth at most \( k \).

Hence, \( x' \) is yes if and only if at least one instance \( x_i \) is yes.

To justify the kernel lower-bound, we provide a short argument showing that Weighted Treewidth parameterized by a vertex cover belongs to the class FPT.

**Proposition 6.** Weighted Treewidth parameterized by a vertex cover is fixed-parameter tractable.

*Proof.* This follows from the work of Bodlaender and Rotics \[12\] by considerations regarding modular decomposition, similar as in the proof of Proposition \[5\].

Given a graph \( G = (V,E) \) and weight function \( w \), we can obtain an equivalent unweighted graph \( G' \) such that the weighted treewidth of \( (G,w) \) is equal to the (unweighted) treewidth of \( G' \), by replacing each vertex \( v \in G \) by a clique of size \( w(v) \). If we start with a weighted graph \( G = (V,E) \) which has a vertex cover \( X \) of size \( \ell \), and let \( G' \) be the resulting unweighted graph, then the maximum number of vertices in a prime subgraph of \( G' \) is bounded by a function of \( \ell \) alone: if we replaced a vertex \( v \in V \) by a clique of \( w(v) \) vertices, then no prime subgraph can contain two or more vertices from such a clique since they would form a module. Similarly, no prime subgraph can contain two vertices from \( G - X \) which have the same set of neighbors in \( X \). Hence the maximum number of vertices in a prime subgraph of \( G' \) is bounded by \( O(2^\ell + \ell) \) which implies by a result of Bodlaender and Rotics \[12\] that the treewidth of \( G' \) (and therefore the weighted treewidth of \( G \)) can be computed in \( f(\ell)n^{O(1)} \) time. \( \square \)