On Sparsification for Computing Treewidth[*]

Bart M.P. Jansen

University of Bergen, Norway
bart.jansen@ii.uib.no

Abstract. We investigate whether an n-vertex instance (G, k) of Treewidth, asking whether the graph G has treewidth at most k, can efficiently be made sparse without changing its answer. By giving a special form of or-cross-composition, we prove that this is unlikely: if there is an $\epsilon > 0$ and a polynomial-time algorithm that reduces n-vertex Treewidth instances to equivalent instances, of an arbitrary problem, with $\mathcal{O}(n^{2-\epsilon})$ bits, then NP $\subseteq$ coNP/poly and the polynomial hierarchy collapses to its third level.

Our sparsification lower bound has implications for structural parameterizations of Treewidth: parameterizations by measures that do not exceed the vertex count, cannot have kernels with $\mathcal{O}(k^{2-\epsilon})$ bits for any $\epsilon > 0$, unless NP $\subseteq$ coNP/poly. Motivated by the question of determining the optimal kernel size for Treewidth parameterized by vertex cover, we improve the $\mathcal{O}(k^3)$-vertex kernel from Bodlaender et al. (STACS 2011) to a kernel with $\mathcal{O}(k^2)$ vertices. Our improved kernel is based on a novel form of *treewidth-invariant set*. We use the q-expansion lemma of Fomin et al. (STACS 2011) to find such sets efficiently in graphs whose vertex count is superquadratic in their vertex cover number.

1 Introduction

The task of preprocessing inputs to computational problems to make them less dense, called *sparsification*, has been studied intensively due to its theoretical and practical importance. Sparsification, and more generally, preprocessing, is a vital step in speeding up resource-demanding computations in practical settings. In the context of theoretical analysis, the *sparsification lemma* due to Impagliazzo et al. [21] has proven to be an important asset for studying subexponential-time algorithms. The work of Dell and van Melkebeek [15] on sparsification for Satisfiability has led to important advances in the area of kernelization lower bounds. They proved that for all $\epsilon > 0$ and $q \geq 3$, assuming NP $\nsubseteq$ coNP/poly, there is no polynomial-time algorithm that maps an instance of q-CNF-SAT on n variables to an equivalent instance on $\mathcal{O}(n^{q-\epsilon})$ bits — not even if it is an instance of a *different* problem.

This paper deals with sparsification for the task of building minimum-width tree decompositions of graphs, or, in the setting of decision problems, of determining whether the treewidth of a graph G is bounded by a given integer k.

[*] This work was supported by ERC Starting Grant 306992 "Parameterized Approximation".

Preprocessing procedures for Treewidth have been studied in applied [10,11,26] and theoretical settings [3,7]. A team including the current author obtained [7] a polynomial-time algorithm that takes an instance (G, k) of Treewidth, and produces in polynomial time a graph G' such that $\textsc{tw}(G) \leq k$ if and only if $\textsc{tw}(G') \leq k$, with the guarantee that $|V(G')| \in \mathcal{O}(\textsc{vc}^3)$ ($\textsc{vc}$ denotes the size of a smallest vertex cover of the input graph). A similar algorithm was given that reduces the vertex count of G' to $\mathcal{O}(\textsc{fvs}^4)$, where $\textsc{fvs}$ is the size of a smallest feedback vertex set in G. Hence polynomial-time data reduction can compress Treewidth instances to a number of vertices polynomial in their vertex cover (respectively feedback vertex) number. On the other hand, the natural parameterization of Treewidth is trivially AND-compositional, and therefore does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly [3,17]. These results give an indication of how far the vertex count of a Treewidth instance can efficiently be reduced in terms of various measures of its complexity. However, they do not tell us anything about the question of *sparsification*: can we efficiently make a Treewidth instance less dense, without changing its answer?

Our Results. Our first goal in this paper is to determine whether nontrivial sparsification is possible for Treewidth instances. As a simple graph G on n vertices can be encoded in n^2 bits through its adjacency matrix, Treewidth instances consisting of a graph G and integer k in the range $[1 \ldots n]$ can be encoded in $\mathcal{O}(n^2)$ bits. We prove that it is unlikely that this trivial sparsification scheme for Treewidth can be improved significantly: if there is a polynomial-time algorithm that reduces Treewidth instances on n vertices to equivalent instances of an arbitrary problem, with $\mathcal{O}(n^{2-\epsilon})$ bits, for some $\epsilon > 0$, then NP $\subseteq$ coNP/poly and the polynomial hierarchy collapses [27]. We prove this result by giving a particularly efficient form of OR-cross-composition [9]. We embed the OR of t n-vertex instances of an NP-complete graph problem into a Treewidth instance with $\mathcal{O}(n\sqrt{t})$ vertices. The construction is a combination of three ingredients. We carefully inspect the properties of Arnborg et al.'s [1] NP-completeness proof for Treewidth to obtain an NP-complete source problem called Cobipartite Graph Elimination that is amenable to composition. Its instances have a restricted form that ensures that good solutions to the composed Treewidth instance cannot be obtained by combining partial solutions to two different inputs. Then, like Dell and Marx [14], we use the layout of a $2 \times \sqrt{t}$ table to embed t instances into a graph on $\mathcal{O}(n^{\mathcal{O}(1)}\sqrt{t})$ vertices. For each way of choosing a cell in the top and bottom row, we embed one instance into the edge set induced by the vertices representing the two cells. Finally, we use ideas employed by Bodlaender et al. [8] in the superpolynomial lower bound for Treewidth parameterized by the vertex-deletion distance to a clique: we compose the input instances of Cobipartite Graph Elimination into a cobipartite graph to let the resulting Treewidth instance express a logical OR, rather than an AND. Our proof combines these three ingredients with an intricate analysis of the behavior of elimination orders on the constructed instance. As the treewidth of the constructed cobipartite graph equals its pathwidth [24], the obtained sparsification lower bound for Treewidth also applies to Pathwidth.

Our sparsification lower bound has immediate consequences for parameterizations of Treewidth by graph parameters that do not exceed the vertex count, such as the vertex cover number or the feedback vertex number. Our result shows the impossibility of obtaining kernels of bitsize $\mathcal{O}(k^{2-\epsilon})$ for such parameterized problems, assuming NP $\not\subseteq$ coNP/poly. The kernel for Treewidth parameterized by vertex cover (Treewidth [vc]) obtained by Bodlaender et al. [6] contains $\mathcal{O}(\textsc{vc}^3)$ vertices, and therefore has bitsize $\Omega(\textsc{vc}^4)$. Motivated by the impossibility of obtaining kernels with $\mathcal{O}(\textsc{vc}^{2-\epsilon})$ bits, and with the aim of developing new reduction rules that are useful in practice, we further investigate kernelization for Treewidth [vc]. We give an improved kernel based on *treewidth-invariant sets*: independent sets of vertices whose elimination from the graph has a predictable effect on its treewidth. While finding such sets seems to be hard in general, we show that the q-expansion lemma, previously employed by Thomassé [25] and Fomin et al. [19], can be used to find them when the graph is large with respect to its vertex cover number. The resulting kernel shrinks Treewidth instances to $\mathcal{O}(\textsc{vc}^2)$ vertices, allowing them to be encoded in $\mathcal{O}(\textsc{vc}^3)$ bits. Thus we reduce the gap between the upper and lower bounds on kernel sizes for Treewidth [vc]. Our new reduction rule for Treewidth [vc] relates to the old rules like the crown-rule for k-Vertex Cover relates to the high-degree Buss-rule [12]: by exploiting local optimality considerations, our reduction rule does not need to know the value of k.

Related Work. While there is an abundance of superpolynomial kernel lower bounds, few superlinear lower bounds are known for problems admitting polynomial kernels. There are results for hitting set problems [15], packing problems [14,20], and for domination problems on degenerate graphs [13].

2 Preliminaries

Parameterized Complexity and Kernels. A parameterized problem $\mathcal{Q}$ is a subset of $\Sigma^* \times \mathbb{N}$. The second component of a tuple $(x, k) \in \Sigma^* \times \mathbb{N}$ is called the *parameter* [16,18]. The set $\{1, 2, \ldots, n\}$ is abbreviated as $[n]$. For a finite set X and integer i we use $\binom{X}{i}$ to denote the collection of size-i subsets of X.

Definition 1 (Generalized kernelization). *Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems and let $h\colon \mathbb{N} \to \mathbb{N}$ be a computable function. A* generalized kernelization for $\mathcal{Q}$ into $\mathcal{Q}'$ of size $h(k)$ *is an algorithm that, on input $(x, k) \in \Sigma^* \times \mathbb{N}$, takes time polynomial in $|x| + k$ and outputs an instance (x', k') such that:*

- *$|x'|$ and k' are bounded by $h(k)$.*
- *$(x', k') \in \mathcal{Q}'$ if and only if $(x, k) \in \mathcal{Q}$.*

The algorithm is a kernelization*, or in short a* kernel*, for $\mathcal{Q}$ if $\mathcal{Q}' = \mathcal{Q}$. It is a* polynomial (generalized) kernelization *if $h(k)$ is a polynomial.*

Cross-Composition. To prove our sparsification lower bound, we use a variant of cross-composition tailored towards lower bounds on the degree of the polynomial in a kernel size bound. The extension is discussed in the journal version [9] of the extended abstract on cross-composition [6].

Definition 2 (Polynomial equivalence relation). *An equivalence relation $\mathcal{R}$ on Σ^* is called a* polynomial equivalence relation *if the following 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 time polynomial in $|x| + |y|$.*
2. *For any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of S into a number of classes that is polynomially bounded in the size of the largest element of S.*

Definition 3 (Cross-composition). *Let $L \subseteq \Sigma^*$ be a language, let $\mathcal{R}$ be a polynomial equivalence relation on Σ^*, let $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem, and let $f : \mathbb{N} \to \mathbb{N}$ be a function. An* OR-cross-composition of L into $\mathcal{Q}$ *(with respect to $\mathcal{R}$)* of cost $f(t)$ *is an algorithm that, given t instances $x_1, x_2, \ldots, x_t \in \Sigma^*$ of L belonging to the same equivalence class of $\mathcal{R}$, takes time polynomial in $\sum_{i=1}^{t} |x_i|$ and outputs an instance $(y, k) \in \Sigma^* \times \mathbb{N}$ such that:*

- *The parameter k is bounded by $\mathcal{O}(f(t) \cdot (\max_i |x_i|)^c)$, where c is some constant independent of t.*
- *$(y, k) \in \mathcal{Q}$ if and only if there is an $i \in [t]$ such that $x_i \in L$.*

Theorem 1 ([9, Theorem 6]). *Let $L \subseteq \Sigma^*$ be a language, let $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem, and let d, ϵ be positive reals. If L is NP-hard under Karp reductions, has an* OR*-cross-composition into $\mathcal{Q}$ with cost $f(t) = t^{1/d+o(1)}$, where t denotes the number of instances, and $\mathcal{Q}$ has a polynomial (generalized) kernelization with size bound $\mathcal{O}(k^{d-\epsilon})$, then NP $\subseteq$ coNP/poly.*

Graphs. All graphs we consider are finite, simple, and undirected. An undirected graph G consists of a vertex set $V(G)$ and an edge set $E(G) \subseteq \binom{V(G)}{2}$. The open neighborhood of a vertex v in graph G is denoted $N_G(v)$, while its closed neighborhood is $N_G[v]$. The open neighborhood of a set $S \subseteq V(G)$ is $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$, while the closed neighborhood is $N_G[S] := N_G(S) \cup S$. If $S \subseteq V(G)$ then $G[S]$ denotes the subgraph of G induced by S. We use $G - S$ to denote the graph $G[V(G) \setminus S]$. A graph is *cobipartite* if its edge-complement is bipartite. Equivalently, a graph G is cobipartite if its vertex set can be partitioned into two sets X and Y, such that both $G[X]$ and $G[Y]$ are cliques. A matching M in a graph G is a set of edges whose endpoints are all distinct. The endpoints of the edges in M are *saturated* by the matching. For disjoint subsets A and B of a graph G, we say that A has a perfect matching into B if there is a matching that saturates $A \cup B$ such that each edge in the matching has exactly one endpoint in each set. If $\{u, v\}$ is an edge in graph G, then *contracting* $\{u, v\}$ *into* u is the operation of adding edges between u and $N_G(v)$ while removing v. A graph H is a *minor* of a graph G, if H can be obtained from a subgraph of G by edge contractions.

Treewidth and Elimination Orders. While treewidth [2] is commonly defined in terms of tree decompositions, for our purposes it is more convenient to work with an alternative characterization in terms of *elimination orders. Eliminating* a vertex v in a graph G is the operation of removing v while completing its open neighborhood into a clique, i.e., adding all missing edges between neighbors of v. An elimination order of an n-vertex graph G is a permutation $\pi\colon V(G) \to [n]$ of its vertices. Given an elimination order π of G, we obtain a series of graphs by consecutively eliminating $\pi^{-1}(1), \ldots, \pi^{-1}(n)$ from G. The *cost* of eliminating a vertex v according to the order π, is the size of the *closed neighborhood* of v at the moment it is eliminated. The *cost of* π *on* G, denoted $c_G(\pi)$, is defined as the maximum cost over all vertices of G.

Theorem 2 ([2, Theorem 36]). *The treewidth of a graph G is exactly one less than the minimum cost of an elimination order for G.*

Lemma 1 ([4, Lemma 4], cf. [23, Lemma 6.13]). *Let G be a graph containing a clique $B \subseteq V(G)$, and let $A := V(G) \setminus B$. There is a minimum-cost elimination order π^* of G that eliminates all vertices of A before eliminating any vertex of B.*

Following the notation employed by Arnborg et al. [1] in their NP-completeness proof, we say that a *block* in a graph G is a maximal set of vertices with the same closed neighborhood. An elimination order π for G is *block-contiguous* if for each block $S \subseteq V(G)$, it eliminates the vertices of S contiguously. The following observation implies that every graph has a block-contiguous minimum-cost elimination order.

Observation 1. *Let G be a graph containing two adjacent vertices u, v such that $N_G[u] \subseteq N_G[v]$. Let π be an elimination order of G that eliminates v before u, and let the order π' be obtained by updating π such that it eliminates u just before v. Then the cost of π' is not higher than the cost of π.*

3 Sparsification Lower Bound for Treewidth

In this section we give the sparsification lower bound for TREEWIDTH. We phrase it in terms of a kernelization lower bound for the parameterization by the number of vertices, formally defined as follows.

n-TREEWIDTH
Input: An integer n, an n-vertex graph G, and an integer k.
Parameter: The number of vertices n.
Question: Is the treewidth of G at most k?

The remainder of this section is devoted to the proof of the following theorem.

Theorem 3. *If n-TREEWIDTH admits a (generalized) kernel of size $\mathcal{O}(n^{2-\epsilon})$, for some $\epsilon > 0$, then NP $\subseteq$ coNP/poly.*

We prove the theorem by cross-composition. We therefore first define a suitable source problem for the composition in Section 3.1, give the construction of the composed instance in Section 3.2, analyze its properties in Section 3.3, and finally put it all together in Section 3.4. The proofs of statements marked with a star (★) are deferred to the full version [22] of this work due to space restrictions.

3.1 The Source Problem

The sparsification lower bound for Treewidth will be established by cross-composing the following problem into it.

Cobipartite Graph Elimination
Input: A cobipartite graph G with partite sets A and B, and a positive integer k, such that the following holds: $|A| = |B|$, $|A|$ is even, $k < \frac{|A|}{2}$, and A has a perfect matching into B.
Question: Is there an elimination order for G of cost at most $|A| + k$?

The NP-completeness proof extends the completeness proof for Treewidth [1].

Lemma 2 (★). Cobipartite Graph Elimination *is NP-complete.*

3.2 The Construction

We start by defining an appropriate polynomial equivalence relationship $\mathcal{R}$. Let all malformed instances be equivalent under $\mathcal{R}$, and let two valid instances of Cobipartite Graph Elimination be equivalent if they agree on the sizes of the partite sets and on the value of k. This is easily verified to be a polynomial equivalence relation.

Now we define an algorithm that combines a sequence of equivalent inputs into a small output instance. As a constant-size NO-instance is a valid output when the input consists of solely malformed instances, in the remainder we assume that the inputs are well-formed. By duplicating some inputs, we may assume that the number of input instances t is a square, i.e., $t = r^2$ for some integer r. An input instance can therefore be indexed by two integers in the range $[r]$. Accordingly, let the input consist of instances $(G_{i,j}, A_{i,j}, B_{i,j}, k_{i,j})$ for $i, j \in [r]$, that are equivalent under $\mathcal{R}$. Thus the number of vertices is the same over all partite sets; let this be $n = |A_{i,j}| = |B_{i,j}|$ for all $i, j \in [r]$. Similarly, let k be the common target value for all inputs. For each partite set $A_{i,j}$ and $B_{i,j}$ in the input, label the vertices arbitrarily as $a^1_{i,j}, \ldots, a^n_{i,j}$ (respectively $b^1_{i,j}, \ldots, b^n_{i,j}$). We construct a cobipartite graph G' that expresses the OR of all the inputs, as follows.

1. For $i \in [r]$ make a vertex set A'_i containing n vertices $\hat{a}^1_i, \ldots, \hat{a}^n_i$.
2. For $i \in [r]$ make a vertex set B'_i containing n vertices $\hat{b}^1_i, \ldots, \hat{b}^n_i$.
3. Turn $\bigcup_{i \in [r]} A'_i$ into a clique. Turn $\bigcup_{i \in [r]} B'_i$ into a clique.
4. For each pair i, j with $i, j \in [r]$, we embed the adjacency of $G_{i,j}$ into G' as follows: for $p, q \in [n]$ make an edge $\{\hat{a}^p_i, \hat{b}^q_j\}$ if $\{a^p_{i,j}, b^q_{i,j}\} \in E(G_{i,j})$.

It is easy to see that at this point in the construction, graph G' is cobipartite. For any $i, j \in [r]$ the induced subgraph $G'[A'_i \cup B'_j]$ is isomorphic to $G_{i,j}$ by mapping $\hat{a}^\ell_i$ to $a^\ell_{i,j}$ and $\hat{b}^\ell_j$ to $b^\ell_{i,j}$. As $G_{i,j}$ has a perfect matching between $A_{i,j}$ and $B_{i,j}$ by the definition of Cobipartite Graph Elimination, this implies that G' has a perfect matching between A'_i and B'_j for all $i, j \in [r]$. These properties will be maintained during the remainder of the construction.

5. For each $i \in [r]$, add the following vertices to G':
   - n *checking vertices* $C'_i = \{c_i^1, \ldots, c_i^n\}$, all adjacent to B'_i.
   - n *dummy vertices* $D'_i = \{d_i^1, \ldots, d_i^n\}$, all adjacent to $\bigcup_{j \in [r]} A'_j$ and to C'_i.
   - $\frac{n}{2}$ *blanker vertices* $X'_i = \{x_i^1, \ldots, x_i^{n/2}\}$, all adjacent to A'_i.
6. Turn $\bigcup_{i \in [r]} A'_i \cup C'_i$ into a clique A'. Turn $\bigcup_{i \in [r]} B'_i \cup D'_i \cup X'_i$ into a clique B'.

The resulting graph G' is cobipartite with partite sets A' and B'. Define $k' := 3rn + \frac{n}{2} + k$. Observe that $|A'| = 2rn$ and that $|B'| = 2rn + \frac{rn}{2}$. Graph G' can easily be constructed in time polynomial in the total size of the input instances.

Intuition. Let us discuss the intuition behind the construction before proceeding to its formal analysis. To create a composition, we have to relate elimination orders in G' to those for input graphs $G_{i,j}$. All adjacency information of the input graphs $G_{i,j}$ is present in G'. As A' is a clique in G', by Lemma 1 there is a minimum-cost elimination order for G' that starts by eliminating all of B'. But when eliminating vertices of some B'_{j^*} from G', they interact simultaneously with all sets A'_i ($i \in [r]$), so the cost of those eliminations is not directly related to the cost of elimination orders of a particular instance G_{i^*,j^*}. We therefore want to ensure that low-cost elimination orders for G' first "blank out" the adjacency of B' to all but one set A'_{i^*}, so that the cost of afterwards eliminating B'_{j^*} tells us something about the cost of eliminating G'_{i^*,j^*}. To blank out the other adjacencies, we need earlier eliminations to make B' adjacent to all vertices of $\bigcup_{i \in [r] \setminus \{i^*\}} A'_i$. These adjacencies will be created by eliminating the *blanker* vertices. For an index $i \in [r]$, vertices in X'_i are adjacent to A'_i and all of B'. Hence eliminating a vertex in X'_i indeed blanks out the adjacency of B' to A'_i. The weights of the various groups (simulated by duplicating vertices with identical closed neighborhoods) have been chosen such that low-cost elimination orders of G' starting with B', have to eliminate $r-1$ blocks of blankers $X'_{i_1}, \ldots, X'_{i_{r-1}}$ before eliminating any other vertex of B'. This creates the desired blanking-out effect. The checking vertices C'_i ($i \in [r]$) enforce that after eliminating $r-1$ blocks of blankers, an elimination order cannot benefit by mixing vertices from two or more sets $B'_i, B'_{i'}$: each set B'_i from which a vertex is eliminated, introduces new adjacencies between B' and C'_i. Finally, the dummy vertices are used to ensure that after one set $B'_i \cup D'_i$ is completely eliminated, the cost of eliminating the remainder is small because $|B'|$ has decreased sufficiently.

3.3 Properties of the Constructed Instance

The following type of elimination orders of G' will be crucial in the proof.

Definition 4. *Let $i^*, j^* \in [r]$. An elimination order π' of G' is (i^*, j^*)-*canonical *if π' eliminates $V(G)$ in the following order:*

1. *first all blocks of blanker vertices X'_i for $i \in [r] \setminus \{i^*\}$, one block at a time,*
2. *then the vertices of B'_{j^*}, followed by dummies D'_{j^*}, followed by blankers X'_{i^*},*
3. *alternatingly a block B'_i followed by the corresponding dummies D'_i, until all remaining vertices of $\bigcup_{i \in [r]} B'_i \cup D'_i$ have been eliminated,*
4. *and finishes with the vertices $\bigcup_{i \in [r]} A'_i \cup C'_i$ in arbitrary order.*

Lemma 3 shows that the crucial part of a canonical elimination order is its behavior on B'_{j^*}.

Lemma 3 (★). *Let π' be an (i^*, j^*)-canonical elimination order for G'.*

1. *No vertex that is eliminated before the first vertex of B'_{j^*} costs more than $3rn$.*
2. *When a vertex of $D'_{j^*} \cup X'_{i^*}$ is eliminated, its cost does not exceed $3rn + \frac{n}{2}$.*
3. *No vertex that is eliminated after X'_{i^*} costs more than $3rn$.*

The next lemma links this behavior to the cost of a related elimination order for G_{i^*,j^*}. Some terminology is needed. Consider an (i^*, j^*)-canonical elimination order π' for G', and an elimination order π for G_{i^*,j^*} that eliminates all vertices of B_{i^*,j^*} before any vertex of A_{i^*,j^*}. By numbering the vertices in B_{i^*,j^*} (a partite set of G_{i^*,j^*}) from 1 to n, we created a one-to-one correspondence between B_{i^*,j^*} and B'_{j^*}, the first set of non-blanker vertices eliminated by π'. Hence we can compare the relative order in which vertices of B_{i^*,j^*} are eliminated in π and π'. If both π and π' eliminate the vertices of B_{i^*,j^*} in the same relative order, then we say that the elimination orders *agree on* B_{i^*,j^*}.

Lemma 4. *Let π' be an (i^*, j^*)-canonical elimination order of G'. Let π be an elimination order for G_{i^*,j^*} that eliminates all vertices of B_{i^*,j^*} before any vertex of A_{i^*,j^*}. If π' and π agree on B_{i^*,j^*}, then $c_{G'}(\pi') = 3rn + \frac{n}{2} - n + c_{G_{i^*,j^*}}(\pi)$.*

Proof. Consider the graph G'_B obtained from G' by performing the eliminations according to π' until we are about to eliminate the first vertex of B'_{j^*}. By Definition 4 this means that all blocks of blankers X'_j for $j \neq j^*$ have been eliminated, and no other vertices. Using the construction of G' it is easy to verify that these eliminations have made all remaining vertices of B' adjacent to $\bigcup_{i \in [r] \setminus \{i^*\}} A'_i$, and that no new adjacencies have been introduced to $\bigcup_{i \in [r]} C'_i$ or to A'_{i^*}. Graph $G'[A'_{i^*} \cup B'_{j^*}]$ was initially isomorphic to G_{i^*,j^*} by the obvious isomorphism based on the numbers assigned to the vertices. As no vertex adjacent to A'_{i^*} has been eliminated yet, this also holds for $G'_B[A'_{i^*} \cup B'_{j^*}]$.

Consider what happens when eliminating the first vertex v' of B'_{j^*} according to π'. Let $v \in B_{i^*,j^*}$ be the corresponding vertex in G_{i^*,j^*}. By the fact that the elimination orders agree, v is the first vertex of B_{i^*,j^*} to be eliminated under π.

The set $N_{G'_B}[v']$ contains C'_{j^*}, $\bigcup_{j \neq j^*} B'_j \cup D'_j$, $\bigcup_{i \neq i^*} A'_i$, X'_{i^*}, D'_{j^*}, and the vertices of $G'[A'_{i^*} \cup B'_{j^*}]$ that correspond exactly to $N_{G_{i^*,j^*}}[v]$ by the isomorphism. So the cost of eliminating v' from G' exceeds the cost of eliminating v from G_{i^*,j^*} by exactly $|C'_{j^*}| + |\bigcup_{j \neq j^*} B'_j \cup D'_j| + |\bigcup_{i \neq i^*} A'_i| + |X'_{i^*}| + |D'_{j^*}| = n + 2(r-1)n + (r-1)n + \frac{n}{2} + n = 3rn + \frac{n}{2} - n$. Now observe that by the isomorphism, eliminating v' from G' has exactly the same effect on the neighborhoods of B'_{j^*} into A'_{i^*}, as eliminating v from G_{i^*,j^*} has on the neighborhoods of B_{i^*,j^*} into A_{i^*,j^*}. Thus after one elimination, the remaining vertices of $A'_{i^*} \cup B'_{j^*}$ and $A_{i^*,j^*} \cup B_{i^*,j^*}$ induce subgraphs of G' and G_{i^*,j^*} that are isomorphic. Hence we may apply the same argument to the next vertex that is eliminated. Repeating this argument we establish that for each vertex in B'_{j^*}, its elimination from G' costs exactly $3rn + \frac{n}{2} - n$ more than the corresponding elimination in G_{i^*,j^*}.

Now consider the cost of π on G_{i^*,j^*}: it is at least $n+1$, as the first vertex to be eliminated is adjacent to all of B_{i^*,j^*} (the graph is cobipartite) and to at least one vertex of A_{i^*,j^*} (since the COBIPARTITE GRAPH ELIMINATION instance G_{i^*,j^*} has a perfect matching between its two partite sets). After all vertices of B_{i^*,j^*} have been eliminated from G_{i^*,j^*}, the remaining vertices cost at most n; there are at most n vertices left in the graph at that point. Hence the cost of π on G_{i^*,j^*} is determined by the cost of eliminating B_{i^*,j^*}. For each vertex from that set that is eliminated, π' incurs a cost exactly $3rn + \frac{n}{2} - n$ higher. Hence $c_{G'}(\pi')$ is at least $(3rn + \frac{n}{2} - n) + (n + 1) = 3rn + \frac{n}{2} + 1$. By Lemma 3 the cost that π' incurs before eliminating the first vertex of B'_{j^*} is at most $3rn$, the cost of eliminating $D'_{j^*} \cup X'_{i^*}$ is at most $3rn + \frac{n}{2}$, and the cost incurred after eliminating the last vertex of B'_{j^*} is at most $3rn$. Hence the cost of π' is determined by the cost of eliminating the vertices of B'_{j^*}. As this is exactly $3rn + \frac{n}{2} - n$ more than the cost of π on G_{i^*,j^*}, this proves the lemma. □

The last technical step of the proof is to show that if G' has an elimination order of cost at most k', then it has such an order that is canonical.

Lemma 5 (★). *If G' has an elimination order of cost at most k', then there are indices $i^*, j^* \in [r]$ such that G' has an (i^*, j^*)-canonical elimination order of cost at most k'.*

3.4 Proof of Theorem 3

Having analyzed the relationship between elimination orders for G' and for the input graphs $G_{i,j}$ $(i, j \in [r])$, we can complete the proof. By combining the previous lemmata it is easy to show that G' acts as the logical OR of the inputs.

Lemma 6 (★). *G' has an elimination order of cost $\leq k'$ $\Leftrightarrow$ there are $i, j \in [r]$ such that $G_{i,j}$ has an elimination order of cost $\leq n + k$.*

Lemma 7. *There is an* OR*-cross-composition of* COBIPARTITE GRAPH ELIMINATION *into n-*TREEWIDTH *of cost $\sqrt{t}$.*

Proof. In Section 3.2 we gave a polynomial-time algorithm that, given instances $(G_{i,j}, A_{i,j}, B_{i,j}, k_{i,j})$ of COBIPARTITE GRAPH ELIMINATION that are equivalent under $\mathcal{R}$ for $i, j \in [r]$, constructs a cobipartite graph G' with partite sets A' and B', and an integer k'. By Lemma 6 the resulting graph G' has an elimination order of cost k' if and only if there is a YES-instance among the inputs. By the correspondence between treewidth and bounded-cost elimination orders of Theorem 2, this shows that G' has treewidth at most $k'-1$ if and only if there is a YES-instance among the inputs. The polynomial equivalence relationship ensured that all partite sets of all inputs have the same number of vertices. For partite sets of size n, the constructed graph G' satisfies $|A'| = 2rn$ and $|B'| = \frac{5rn}{2}$. The number of vertices in G' is $n' = \frac{9rn}{2}$. Consider the n-TREEWIDTH instance $(G', n', k'-1)$. It expresses the logical OR of a series of $r^2 = t$ COBIPARTITE GRAPH ELIMINATION instances using a parameter value of $\frac{9n\sqrt{t}}{2} \in \mathcal{O}(n\sqrt{t})$. Hence the algorithm gives an OR-cross-composition of COBIPARTITE GRAPH ELIMINATION into n-TREEWIDTH of cost $\sqrt{t}$. □

Theorem 3 follows from the combination of Lemma 7, Lemma 2, and Theorem 1. Since the pathwidth of a cobipartite graph equals its treewidth [24] and the graph formed by the cross-composition is cobipartite, the same construction gives an OR-cross-composition of bounded cost into n-PATHWIDTH.

Corollary 1. *If n-*PATHWIDTH *admits a (generalized) kernel of size $\mathcal{O}(n^{2-\epsilon})$, for some $\epsilon > 0$, then NP $\subseteq$ coNP/poly.*

4 Quadratic-Vertex Kernel for Treewidth [VC]

In this section we present an improved kernel for TREEWIDTH [VC], which is formally defined as follows.

TREEWIDTH [VC]
Input: A graph G, a vertex cover $X \subseteq V(G)$, and an integer k.
Parameter: $|X|$.
Question: Is the treewidth of G at most k?

Our kernelization revolves around the following notion.

Definition 5. *Let G be a graph, let T be an independent set in G, and let $\hat{G}_T$ be the graph obtained from G by eliminating T; the order is irrelevant as T is independent. Then T is a* treewidth-invariant set *if for every $v \in T$, the graph $\hat{G}_T$ is a minor of $G - \{v\}$.*

Lemma 8. *If T is a treewidth-invariant set in G and $\Delta := \max_{v \in T} \deg_G(v)$, then $\textsc{tw}(G) = \max(\Delta, \textsc{tw}(\hat{G}_T))$.*

Proof. We prove that $\textsc{tw}(G)$ is at least, and at most, the claimed amount.

($\geq$). As $\hat{G}_T$ is a minor of G, we have $\textsc{tw}(G) \geq \textsc{tw}(\hat{G}_T)$ (cf. [2]). If $\textsc{tw}(\hat{G}_T) \geq \Delta$ then this implies the inequality. So assume that $\Delta > \textsc{tw}(\hat{G}_T)$. Let $v \in T$ have degree Δ. By assumption, $\hat{G}_T$ is a minor of $G - \{v\}$. It contains all vertices of $N_G(v)$ since T is an independent set. As $N_G(v)$ is a clique in $\hat{G}_T$, there is a series of minor operations in $G - \{v\}$ that turns $N_G(v)$ into a clique. Performing these operations on G rather than $G - \{v\}$ results in a clique on vertex set $N_G[v]$ of size $\deg_G(v) + 1 = \Delta + 1$: the set $N_G(v)$ is turned into a clique, and v remains unchanged. Hence G has a clique with $\Delta + 1$ vertices as a minor, which is known to imply (cf. [2]) that its treewidth is at least Δ.

($\leq$). Consider an optimal elimination order $\hat{\pi}$ for $\hat{G}_T$, which costs $\textsc{tw}(\hat{G}_T) + 1$ by Theorem 2. Form an elimination order π for G by first eliminating all vertices in T in arbitrary order, followed by the remaining vertices in the order dictated by $\hat{\pi}$. Consider what happens when eliminating the graph G in the order given by π. Each vertex $v \in T$ that is eliminated incurs cost $\deg_G(v) + 1 \leq \Delta + 1$: as T is an independent set, eliminations before v do not affect v's neighborhood. Once all vertices of T have been eliminated, the resulting graph is identical to $\hat{G}_T$, by definition. As π matches $\hat{\pi}$ on the vertices of $V(G) \setminus T$, and $\hat{\pi}$ has cost $\textsc{tw}(\hat{G}_T) + 1$, the total cost of elimination order π on G is $\max(\Delta + 1, \textsc{tw}(\hat{G}_T) + 1)$. By Theorem 2 this completes this direction of the proof. $\square$

Lemma 8 shows that when a treewidth-invariant set is eliminated from a graph, its treewidth changes in a controlled manner. To exploit this insight in a kernelization algorithm, we have to find treewidth-invariant sets in polynomial time. While it seems difficult to detect such sets in all circumstances, we show that the q-expansion lemma can be used to find a treewidth-invariant set when the size of the graph is large compared to its vertex cover number. The following auxiliary graph is needed for this procedure.

Definition 6. *Given a graph G with a vertex cover $X \subseteq V(G)$, we define the bipartite* non-edge connection graph $H_{G,X}$. *Its partite sets are $V(G) \setminus X$ and $\binom{X}{2} \setminus E(G)$, with an edge between a vertex $v \in V(G) \setminus X$ and a vertex $x_{\{p,q\}}$ representing $\{p,q\} \in \binom{X}{2} \setminus E(G)$ if $v \in N_G(p) \cap N_G(q)$.*

For disjoint vertex subsets S and T in a graph G, we say that S *is saturated by q-stars into* T if we can assign to every $v \in S$ a subset $f(v) \subseteq N_G(v) \cap T$ of size q, such that for any pair of distinct vertices $u, v \in S$ we have $f(u) \cap f(v) = \emptyset$. Observe that an empty set can trivially be saturated by q-stars.

Lemma 9 (★). *Let (G, X, k) be an instance of* Treewidth [vc]. *If $H_{G,X}$ contains a set $T \subseteq V(G) \setminus X$ such that $S := N_{H_{G,X}}(T)$ can be saturated by 2-stars into T, then T is a treewidth-invariant set.*

***q*-Expansion Lemma ([19, Lemma 12]).** *Let q be a positive integer, and let m be the size of a maximum matching in a bipartite graph H with partite sets A and B. If $|B| > m \cdot q$ and there are no isolated vertices in B, then there exist nonempty vertex sets $S \subseteq A$ and $T \subseteq B$ such that S is saturated by q-stars into T and $S = N_H(T)$. Furthermore, S and T can be found in time polynomial in the size of H by a reduction to bipartite matching.*

Theorem 4. Treewidth [vc] *has a kernel with $\mathcal{O}(|X|^2)$ vertices that can be encoded in $\mathcal{O}(|X|^3)$ bits.*

Proof. Given an instance (G, X, k) of Treewidth [vc], the algorithm constructs the non-edge connection graph $H_{G,X}$ with partite sets $A = \binom{X}{2} \setminus E(G)$ and $B = V(G) \setminus X$. We attempt to find a treewidth-invariant set $T \subseteq B$. If B has an isolated vertex v, then by definition of $H_{G,X}$ the set $N_G(v)$ is a clique implying that $\{v\}$ is treewidth-invariant. If B has no isolated vertices, we apply the q-expansion lemma with $q := 2$ to attempt to find a set $S \subseteq A$ and $T \subseteq B$ such that S is saturated by 2-stars into T and $S = N_{H_{G,X}}(T)$. Hence such a set T is treewidth-invariant by Lemma 9. If we find a treewidth-invariant set T:

– If $\max_{v \in T} \deg_G(v) \geq k + 1$ then we output a constant-size no-instance, as Lemma 8 then ensures that $\textsc{tw}(G) \geq \deg_G(v) > k$.
– Otherwise we reduce to $(\hat{G}_T, X, k)$ and restart the algorithm.

Each iteration takes polynomial time. As the vertex count decreases in each iteration, there are at most n iterations until we fail to find a treewidth-invariant set. When that happens, we output the resulting instance. The q-expansion lemma ensures that at that point, $|B| \leq 2m$, where m is the size of a maximum matching in $H_{G,X}$. As m cannot exceed the size of the partite set A, which is bounded by $\binom{|X|}{2}$ as there cannot be more non-edges in a set of size $|X|$, we find that $|B| \leq 2\binom{|X|}{2}$ upon termination. As vertex set B of the graph $H_{G,X}$ directly corresponds to $V(G) \setminus X$, this implies that G has at most $|X|+2\binom{|X|}{2}$ vertices after exhaustive reduction. Thus the instance that we output has $\mathcal{O}(|X|^2)$ vertices. We can encode it in $\mathcal{O}(|X|^3)$ bits: we store an adjacency matrix for $G[X]$, and for each of the $\mathcal{O}(|X|^2)$ vertices v in $V(G) \setminus X$ we store a vector of $|X|$ bits, indicating for each $x \in X$ whether v is adjacent to it. $\square$

5 Conclusion

In this paper we contributed to the knowledge of sparsification for TREEWIDTH by establishing lower and upper bounds. Our work raises a number of questions.

We showed that TREEWIDTH and PATHWIDTH instances on n vertices are unlikely to be compressible into $\mathcal{O}(n^{2-\epsilon})$ bits. Are there natural problems on general graphs that do allow (generalized) kernels of size $\mathcal{O}(n^{2-\epsilon})$? Many problems admit $\mathcal{O}(k)$-vertex kernels when restricted to *planar* graphs [5], which can be encoded in $\mathcal{O}(k)$ bits by employing succinct representations of planar graphs. Obtaining subquadratic-size compressions for NP-hard problems on classes of potentially *dense* graphs, such as unit-disk graphs, is an interesting challenge.

In Section 4 we gave a quadratic-vertex kernel for TREEWIDTH [VC]. While the algorithm is presented for the decision problem, it is easily adapted to the optimization setting (cf. [11]). The key insight for our reduction is the notion of treewidth-invariant sets, together with the use of the q-expansion lemma to find them when the complement of the vertex cover has superquadratic size. A challenge for future research is to identify treewidth-invariant sets that are not found by the q-expansion lemma; this might decrease the kernel size even further. As the sparsification lower bound proves that TREEWIDTH [VC] is unlikely to admit kernels of bitsize $\mathcal{O}(|X|^{2-\epsilon})$, while the current kernel can be encoded in $\mathcal{O}(|X|^3)$ bits, an obvious open problem is to close the gap between the upper and the lower bound. Does TREEWIDTH [VC] have a kernel with $\mathcal{O}(|X|)$ vertices? If not, then is there at least a kernel with $\mathcal{O}(|X|^2)$ rather than $\mathcal{O}(|X|^3)$ edges?

For PATHWIDTH [VC], a kernel with $\mathcal{O}(|X|^3)$ vertices is known [8]. Can this be improved to $\mathcal{O}(|X|^2)$ using an approach similar to the one used here? The obvious pathwidth-analogue of Lemma 8 fails, as removing a low-degree simplicial vertex may decrease the pathwidth of a graph. Finally, one may consider whether the ideas of the present paper can improve the kernel size for TREEWIDTH parameterized by a feedback vertex set [7].

References

1. Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM J. Algebra. Discr. 8, 277–284 (1987), doi:10.1137/0608024
2. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theor. Comput. Sci. 209(1-2), 1–45 (1998), doi:10.1016/S0304-3975(97)00228-4

3. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. J. Comput. Syst. Sci. 75(8), 423–434 (2009), doi:10.1016/j.jcss.2009.04.001
4. Bodlaender, H.L., Fomin, F.V., Koster, A.M.C.A., Kratsch, D., Thilikos, D.M.: On exact algorithms for treewidth. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 672–683. Springer, Heidelberg (2006)
5. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M. (Meta) Kernelization. In: Proc. 50th FOCS, pp. 629–638 (2009), doi:10.1109/FOCS.2009.46
6. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Cross-composition: A new technique for kernelization lower bounds. In: Proc. 28th STACS, pp. 165–176 (2011), doi:10.4230/LIPIcs.STACS.2011.165
7. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Preprocessing for treewidth: A combinatorial analysis through kernelization. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 437–448. Springer, Heidelberg (2011)
8. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernel bounds for structural parameterizations of pathwidth. In: Fomin, F.V., Kaski, P. (eds.) SWAT 2012. LNCS, vol. 7357, pp. 352–363. Springer, Heidelberg (2012)
9. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. CoRR, abs/1206.5941, arXiv:1206.5941 (2012)
10. Bodlaender, H.L., Koster, A.M.C.A.: Safe separators for treewidth. Discrete Math. 306(3), 337–350 (2006), doi:10.1016/j.disc.2005.12.017
11. Bodlaender, H.L., Koster, A.M.C.A., van den Eijkhof, F.: Preprocessing rules for triangulation of probabilistic networks. Comput. Intell. 21(3), 286–305 (2005), doi:10.1111/j.1467-8640.2005.00274.x
12. Buss, J.F., Goldsmith, J.: Nondeterminism within P. SIAM J. Comput. 22(3), 560–572 (1993), doi:10.1137/0222038
13. Cygan, M., Grandoni, F., Hermelin, D.: Tight kernel bounds for problems on graphs with small degeneracy. CoRR, abs/1305.4914, arXiv:1305.4914 (2013)
14. Dell, H., Marx, D.: Kernelization of packing problems. In: Proc. 23rd SODA, pp. 68–81 (2012)
15. Dell, H., van Melkebeek, D.: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In: Proc. 42nd STOC, pp. 251–260 (2010), doi:10.1145/1806689.1806725
16. Downey, R., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, New York (1999)
17. Drucker, A.: New limits to classical and quantum instance compression. In: Proc. 53rd FOCS, pp. 609–618 (2012), doi:10.1109/FOCS.2012.71
18. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer-Verlag New York, Inc. (2006)
19. Fomin, F.V., Lokshtanov, D., Misra, N., Philip, G., Saurabh, S.: Hitting forbidden minors: Approximation and kernelization. In: Proc. 28th STACS, pp. 189–200 (2011), doi:10.4230/LIPIcs.STACS.2011.189
20. Hermelin, D., Wu, X.: Weak compositions and their applications to polynomial lower bounds for kernelization. In: Proc. 23rd SODA, pp. 104–113 (2012)
21. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. 63(4), 512–530 (2001), doi:10.1006/jcss.2001.1774
22. Jansen, B.M.P.: On sparsification for computing treewidth. CoRR, abs/1308.3665, arXiv:1308.3665 (2013)

23. Jansen, B.M.P.: The Power of Data Reduction: Kernels for Fundamental Graph Problems. PhD thesis, Utrecht University, The Netherlands (2013)
24. Möhring, R.H.: Triangulating graphs without asteroidal triples. Discrete Appl. Math. 64(3), 281–287 (1996), doi:10.1016/0166-218X(95)00095-9
25. Thomassé, S.: A $4k^2$ kernel for feedback vertex set. ACM Trans. Algorithms 6(2) (2010), doi:10.1145/1721837.1721848
26. van den Eijkhof, F., Bodlaender, H.L., Koster, A.M.C.A.: Safe reduction rules for weighted treewidth. Algorithmica 47(2), 139–158 (2007), doi:10.1007/s00453-006-1226-x
27. Yap, C.-K.: Some consequences of non-uniform conditions on uniform classes. Theor. Comput. Sci. 26, 287–300 (1983), doi:10.1016/0304-3975(83)90020-8