Algorithmica (2015) 71:605–635
DOI 10.1007/s00453-014-9924-2

On Sparsification for Computing Treewidth

Bart M. P. Jansen

Received: 22 November 2013 / Accepted: 24 July 2014 / Published online: 14 August 2014

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 ℓ that do not exceed the number of vertices cannot have kernels with $\mathcal{O}(\ell^{2-\epsilon})$ bits for any $\epsilon > 0$, unless NP $\subseteq$ coNP/poly. Motivated by the question of determining the optimal kernel size for TREEWIDTH parameterized by the size of a vertex cover X, we improve the $\mathcal{O}(|X|^3)$-vertex kernel from Bodlaender et al. (SIDMA 2013) to a kernel with $\mathcal{O}(|X|^2)$ vertices. Our improved kernel is based on the novel notion of *treewidth-invariant set*. We use the q-expansion lemma of Fomin et al. (STACS 2011) to find such sets efficiently in graphs whose order is superquadratic in their vertex cover number. We believe that our new reduction rule will be useful in practice.

Keywords Sparsification · Lower bounds · Treewidth · Kernelization

Mathematics Subject Classification F.2.2 · G.2.2

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

B. M. P. Jansen (✉)
Department of Informatics, University of Bergen, PO Box 7803, 5020 Bergen, Norway
e-mail: Bart.Jansen@ii.uib.no

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. [28] has proven to be an important asset for studying subexponential-time algorithms. The work of Dell and van Melkebeek [17] 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 $\not\subseteq$ coNP/poly, there is no polynomial-time algorithm that maps an instance x of q-CNF-SAT on n variables to an equivalent instance x' on $\mathcal{O}(n^{q-\epsilon})$ bits—not even if x' 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. Preprocessing procedures for TREEWIDTH have been studied in applied [9,10,20] 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 as input, and produces in polynomial time a graph G' such that $\text{TW}(G) \leq k$ if and only if $\text{TW}(G') \leq k$, with the guarantee that $|V(G')| \in \mathcal{O}(\text{VC}^3)$ (VC denotes the size of a smallest vertex cover of the input graph). A similar algorithm was given that reduces the number of vertices of G' to $\mathcal{O}(\text{FVS}^4)$, where 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 [3,19]. These results give an indication of how far the order 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?

1.1 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 [34]. We prove this result by giving a particularly efficient form of OR-cross-composition [8]. We embed the OR of t distinct 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 [16], we use the layout of a $2 \times \sqrt{t}$ table to embed t instances into a graph on $\mathcal{O}(n\sqrt{t})$ 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. [6] 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, allowing us to apply the OR-cross-composition framework. 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 [31], 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 ℓ that do not exceed the number of vertices, such as the vertex cover number or the feedback vertex number. Our result shows the impossibility of obtaining kernels of bitsize $\mathcal{O}(\ell^{2-\epsilon})$ 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. [7] contains $\mathcal{O}(\text{VC}^3)$ vertices, and therefore has bitsize $\Omega(\text{VC}^4)$. Motivated by the impossibility of obtaining kernels with $\mathcal{O}(\text{VC}^{2-\epsilon})$ 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 the new notion of *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 [33] and Fomin et al. [24] (cf. [25]), 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 $\mathcal{O}(\text{VC}^2)$ vertices, allowing them to be encoded in $\mathcal{O}(\text{VC}^3)$ 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 [11]: by exploiting local optimality considerations, our reduction rule does not need to know the value of k. The kernel can be obtained by computing a maximum matching in a bipartite graph in a graph whose order is linear in the size of the input, and the number of non-edges induced by the vertex cover from which the process is initiated. Based on the fact that the parameter-independent reduction algorithm is based on this well-understood matching subroutine, we expect that the new reduction rule will be useful in practice. Consequently, we pay some attention to implementation details and running time optimization in the exposition. As a byproduct of the exploited insights into the structure of treewidth instances, we also obtain a new bound on the order of minor-minimal treewidth obstructions in terms of their vertex cover number.

1.2 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 [17], packing problems [16,26], and for domination problems on degenerate graphs [15]. There are a number of experimental results on preprocessing for Treewidth [9,10,20]. Theoretical guarantees for preprocessing Treewidth [7] and Pathwidth [6] have also been obtained.

1.3 Organization

In Sect. 2 we give preliminaries on parameterized complexity, kernel lower bounds, graphs, and treewidth. In Sect. 3 we prove the sparsification lower bound for Treewidth $[n]$. The improved kernel for Treewidth [vc] is developed in Sect. 4.

2 Preliminaries

2.1 Parameterized Complexity and Kernels

A parameterized problem $\mathcal{Q}$ is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. The second component of a tuple $(x, k) \in \Sigma^* \times \mathbb{N}$ is called the *parameter* [18,23]. A parameterized problem $\mathcal{Q}$ is (strongly uniformly) *fixed-parameter tractable* if there is an algorithm that decides whether $(x, k) \in \mathcal{Q}$ that runs in time $f(k)|x|^{\mathcal{O}(1)}$ for some computable function f. 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. The concept of kernelization, that can be expressed in the language of parameterized complexity, will be used to capture various forms of efficient preprocessing.

Definition 1 (*Generalized kernelization*) Let $\mathcal{Q}, \mathcal{Q}' \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems and let $h : \mathbb{N} \to \mathbb{N}$ 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.

2.2 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. It was introduced in the journal version of the article on cross-composition [8].

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 ([8, 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.*

2.3 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 $N_G[S] := N_G(S) \cup S$. The degree of a vertex v in G is denoted by $\deg_G(v)$. If $S \subseteq V(G)$ then $G[S]$ denotes the subgraph of G induced by S. We use $G - S$ to denote the graph $G[V(G) \setminus S]$ that results after deleting all vertices of S and their incident edges from G. For a graph G, we denote its *edge-complement* by $\overline{G}$: this is the graph with vertex set $V(G)$ and edge set $\binom{V(G)}{2} \setminus E(G)$. 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 vertex 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 $N_G(v) \setminus \{u\}$ 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. If, additionally, H is not isomorphic to G, then H is a *proper minor* of G. We say

that an unordered pair $\{p, q\} \in \binom{V(G)}{2}$ is a *non-edge* in graph G if $\{p, q\} \notin E(G)$. A graph is *subcubic* if the degree of any of its vertices is at most three. The *cutwidth* of an n-vertex graph G with respect to a permutation $\pi : V(G) \to [n]$ of its vertices is defined as $\text{CUTW}(G, \pi) := \max_{i \in [n]} |\{\{u, v\} \in E(G) \mid \pi(u) \leq i < \pi(v)\}|$. The cutwidth of a graph is the minimum of $\text{CUTW}(G, \pi)$ over all permutations π of its vertex set. It is easy to see that the cutwidth of a disconnected graph is the maximum cutwidth of any of its connected components. The cutwidth of a graph does not exceed the number of edges.

When analyzing the running time of algorithms on graphs, we assume that the input graph is given by an adjacency-list representation. During the exposition we assume familiarity with basic algorithmic procedures such as radix sort and merge sort. The textbook by Cormen et al. [14] contains all the relevant background information.

2.4 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 $\pi : V(G) \to [n]$ 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.** [29, 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.

TREEWIDTH $[n]$
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* TREEWIDTH *$[n]$ 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 giving a cross-composition of cost $\mathcal{O}(\sqrt{t})$ of t instances of an NP-hard problem into TREEWIDTH $[n]$. We therefore first define a suitable source problem for the composition in Sect. 3.1, give the construction of the composed instance in Sect. 3.2, analyze its properties in Sect. 3.3, and finally put it all together in Sect. 3.4.

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.*

Proof Membership in NP is trivial. To establish completeness, we use the connection between treewidth and elimination orders. The instances created by the NP-completeness proof for TREEWIDTH due to Arnborg et al. [1] are close to satisfying the desired conditions. In Sect. 3 of their paper, Arnborg et al. [1] reduce the CUTWIDTH problem to TREEWIDTH. They show how to transform an n-vertex graph G with maximum degree Δ into a cobipartite graph G' with partite sets A and B of size $(\Delta+1)n$ each, such that G has cutwidth at most k if and only if G' has treewidth at most $(\Delta+1)(n+1)+k-1 = |A|+\Delta+k$. By Theorem 2 the latter happens if and only if G' has an elimination order of cost at most $|A|+\Delta+k+1$.

We briefly describe their construction to show that it results in a graph with a perfect matching between the sets A and B. Given a graph G of maximum degree Δ,

the cobipartite graph G' is constructed as follows. Each vertex $x \in V(G)$ is represented by $\Delta + 1$ vertices in A, and $\Delta - \deg_G(x) + 1$ vertices in B. Let A_x (resp. B_x) denote the set of vertices in A (resp. B) that represents x. Each edge $e \in E(G)$ is represented by two vertices in B; this set of vertices is denoted B_e. All vertices in A_x are adjacent to all vertices in B_x. All vertices in A are adjacent to both vertices in B_e if x is an endpoint of edge e. The set A is turned into a clique, as is the set B. A perfect matching between the sets A and B may be obtained as follows. For each $x \in V(G)$, for each edge e incident on x in G, match one vertex in A_x to a vertex in B_e. Match the remaining $(\Delta + 1) - \deg_G(x) = |B_x|$ vertices in A_x to B_x.

Using this information we prove the NP-completeness of Cobipartite Graph Elimination. We reduce from Cutwidth3 (cf. [6, §5]), the cutwidth problem on subcubic graphs, which is known to be NP-complete [32, Corollary 2.10]. Given an instance (G, k) of Cutwidth3, which asks whether the subcubic graph G has cutwidth at most k, let n be the number of vertices in G. As the cutwidth of a graph does not exceed its edge count, and a subcubic n-vertex graph has at most $3n$ edges, we may output a constant-size yes-instance if $k \geq 3n$. In the remainder we therefore have $k < 3n$. Form a new graph G^* as the disjoint union of sufficiently many, say 20, copies of G. The resulting graph G^* has $20n$ vertices, and its maximum degree is $\Delta \leq 3$. As the cutwidth of a graph is the maximum cutwidth of its connected components, graph G^* has cutwidth at most k if and only if (G, k) is a yes-instance. Now apply the transformation by Arnborg et al. to the instance (G^*, k). It results in a cobipartite graph G' with partite sets A' and B' of size $(\Delta+1)20n$, such that G' has an elimination order of cost $|A'| + \Delta + k + 1$ if and only if (G, k) is a yes-instance. The construction ensures that G' has a perfect matching between A' and B'. Now put $k' := \Delta + k + 1 < 4 + 3n \leq \frac{|A'|}{2}$. It is easy to see that $|A'|$ is even. The resulting instance (G', A', B', k') of Cobipartite Graph Elimination therefore satisfies all constraints. As G' has an elimination order of cost at most $|A'| + k'$ if and only if (G, k) has cutwidth at most three, this completes the proof. □

3.2 The Construction

We start by defining an appropriate polynomial equivalence relation $\mathcal{R}$ as in Definition 2. Let all strings that do not encode valid instances of Cobipartite Graph Elimination be equivalent under $\mathcal{R}$. Let two valid instances of the problem 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. If the number of input instances t is not a square (i.e., $\sqrt{t}$ is not an integer) then we may duplicate some input instances to increase their number to the nearest even power of two. This does not affect the value of the or of the inputs, and increases the number of inputs by at most a factor four. If the cost of the cross-composition is $\mathcal{O}(\sqrt{t})$ with respect to the increased value of t, it will therefore also be $\mathcal{O}(\sqrt{t})$ with respect to the original value of t.

We may therefore 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 $(G_{i,j}, A_{i,j}, B_{i,j}, k_{i,j})$ of Cobipartite Graph Elimination 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* $C'_i = \{c^1_i, \ldots, c^n_i\}$, all adjacent to B'_i.
   - *n dummy vertices* $D'_i = \{d^1_i, \ldots, d^n_i\}$, all adjacent to $\bigcup_{j\in[r]} A'_j$ and to C'_i.
   - *$\frac{n}{2}$ blanker vertices* $X'_i = \{x^1_i, \ldots, x^{n/2}_i\}$, 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. Refer to Fig. 1 for an example.

3.2.1 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 $G'[A'_i \cup B'_j]$ is isomorphic to $G_{i,j}$ for $i, j \in [r]$. 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 afterward eliminating B'_{j^*} tells us something about the cost

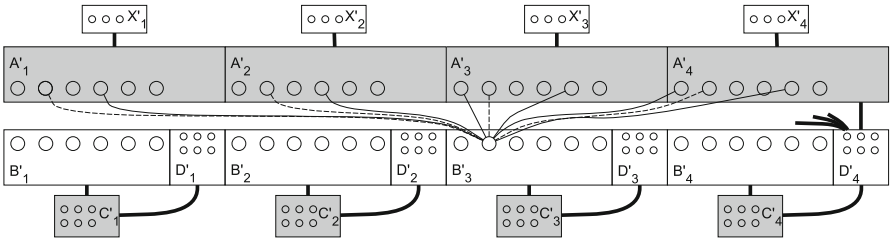


Fig. 1 Illustration of the construction for a composition of $t = r^2 = 4^2$ input instances whose partite sets all have $n = 6$ vertices each, into the structure of a 2×4 table. Vertices that are drawn within the same box form a clique. When two boxes are joined by a thick line, the graph contains all possible edges between the two groups of vertices. The vertices in the *gray boxes* form the clique A' in the cobipartite graph G'. The vertices in *white boxes* form the clique B'. Adjacencies between sets $D'_.$ and $\bigcup_{i\in[r]} A'_i$ are only visualized for D'_4. For all $i, j \in [r]$ the set A'_i has a perfect matching into B'_j. Matching edges are drawn with dashed lines. Each vertex in a set $A'_.$ or $B'_.$ has many neighbors in the opposite clique. The corresponding edges are only drawn for vertex $\hat{b}^2_3$ in the set B'_3. For each $i, j \in [r]$ the subgraph induced by $A'_i \cup B'_j$ is isomorphic to the input graph $G_{i,j}$

of eliminating G'_{i^*,j^*}. We can effectively blank out the adjacency between B' and A'_i by establishing complete adjacency between the two sets, which has the effect that the resulting graph forgets the original adjacency. We therefore 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 causes all possible edges to be added between A'_i and B', thereby blanking 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' that start 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

Observe that in the constructed graph G', all vertices in a set X'_i ($i \in [r]$) have exactly the same closed neighborhood. The same is true for vertices in sets C'_i or D'_i. By Observation 1 we may therefore assume that an optimal elimination order eliminates all vertices in one such group consecutively. This motivates our interest in the behavior of elimination orders with respect to *blocks* of blanker vertices, checker vertices, or dummy vertices. 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.

We call an elimination order canonical if it is (i^*, j^*)-canonical for some choice of i^* and j^*. 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 $3rn$.*

Proof (1) By Definition 4, all vertices eliminated before B'_{j^*} are blanker vertices. The elimination of a vertex v in a block X'_i turns $N(v)$ into a clique and removes v. As A' and B' are cliques from the start, no extra edges can be introduced between members of A' or between members of B'. When considering the effects of eliminating vertices from B', we therefore only have to consider which vertices of B' become adjacent to vertices in A'. As blanker vertices are not adjacent to checking vertices, no elimination of a blanker vertex introduces adjacencies to sets C'_j for any $j \in [r]$. Eliminating X'_i effectively makes all remaining vertices in B' adjacent to A'_i, as X'_i and A'_i are adjacent by construction. With these insights we prove the first item of the lemma.

Consider the situation when $0 \leq \ell < r - 1$ blocks of blankers $X'_{i_1}, \ldots, X'_{i_\ell}$ have already been eliminated, and we are about to eliminate a blanker vertex v_B in the next block $X'_{i_{\ell+1}}$. Then $N[v_B] \cap B'$ contains all remaining vertices in B', of which there are $|B'| - \ell \cdot \frac{n}{2} = 2rn + \frac{(r-\ell)n}{2}$. Now consider the neighborhood of v_B in A'. As observed above, $N[v_B] \cap A'$ does not contain checking vertices. When it comes to adjacencies into $\bigcup_{i\in[r]} A'_i$, vertex v_B in block $X'_{i_{\ell+1}}$ is adjacent to $A'_{i_{\ell+1}}$, and to the blocks $A'_{i_1}, \ldots, A'_{i_\ell}$ for the blankers $X'_{i_1}, \ldots, X'_{i_\ell}$ that have previously been eliminated. Hence v_B has $(\ell + 1)n$ neighbors in A'. Summing up the contributions from the two partite sets, we find $|N[v_B]| = 2rn + \frac{(r-\ell)n}{2} + (\ell + 1)n = 3rn + \frac{(\ell-r)n}{2} + n$. The largest value is attained when the last block of blankers unequal to X'_{i^*} is about to be eliminated; at that point $\ell = r - 2$ blocks have been eliminated already, which results in a cost of $3rn$ for the first vertex of the last block $X'_{i_{r-1}}$ that is eliminated. Having argued that the cost of the *first* vertex of a block to be eliminated never exceeds $3rn$, we show that this also holds for the other vertices in the same block. After the first vertex of a block of blankers $X'_{i_{\ell+1}}$ has been eliminated, the other vertices of $X'_{i_{\ell+1}}$ are eliminated immediately after v, by Definition 4. Hence their closed neighborhood at time of elimination is smaller than that of v: elimination of v does not introduce any new adjacencies for vertices with the same closed neighborhood as v. Hence the

remaining vertices of $X'_{i_{\ell+1}}$ cost less than v. Thus the cost of eliminating the vertices before B'_{j^*} does not exceed $3rn$.

(2) Let G'_B be the graph that is obtained from G' by eliminating according to π' until just after the last vertex of B'_{j^*}. Then G'_B contains exactly one block of blankers X'_{i^*}, as all other sets have been eliminated before B'_{j^*}. It does not contain B'_{j^*} as that was just eliminated. The elimination of B'_{j^*} has made the remainder of B' adjacent to $\bigcup_{i\in[r]} A'_i$, as B'_{j^*} has a perfect matching into A'_i for all $i \in [r]$. According to Definition 4, elimination order π' eliminates D'_{j^*} just after B'_{j^*}. At that point, the neighborhood of the dummy vertices D'_{j^*} into $\bigcup_{j\in[r]} C'_j$ is exactly C'_{j^*}: the set D'_{j^*} was initially adjacent to C'_{j^*}, the eliminated blanker vertices were not adjacent to any checking vertices and therefore did not introduce new adjacencies to checking vertices, and the eliminated vertices from B'_{j^*} see only the checking vertices C'_{j^*}, by construction. Hence the cost of eliminating the first dummy vertex in D'_{j^*} is $|\bigcup_{j\in[r]\setminus\{j^*\}} B'_{j^*} \cup D'_{j^*}| + |D'_{j^*}| + |X'_{i^*}| + |\bigcup_{i\in[r]} A'_i| + |C'_{j^*}|$, which is $2(r-1)n + n + \frac{n}{2} + rn + n = 3rn + \frac{n}{2}$. As the other dummy vertices in D'_{j^*} have exactly the same closed neighborhood, their elimination is not more expensive.

By Definition 4, order π' follows the elimination of D'_{j^*} by eliminating X'_{i^*}. At the time of elimination, $N[X'_{i^*}] \cap B'$ has size $|\bigcup_{j\in[r]\setminus\{j^*\}} B'_j \cup D'_j| + |X'_{i^*}| = 2(r-1)n + \frac{n}{2}$. Vertices in X'_{i^*} are adjacent to $\bigcup_{i\in[r]} A'_i$, and to exactly one set of checking vertices, namely C'_{j^*}; the elimination of B'_{j^*} has introduced these adjacencies. Hence the cost of eliminating the first vertex of X'_{i^*} is $2(r-1)n + \frac{n}{2} + rn + n = 3rn - \frac{n}{2}$. As the other vertices in X'_{i^*} have the same neighborhood, they are not more expensive. In summary, no vertex of $D'_{j^*} \cup X'_{i^*}$ costs more than $3rn + \frac{n}{2}$ when eliminated.

(3) Let G'_X be the graph that is obtained from G' by eliminating according to π' until just after the last vertex of X'_{i^*}. Then G'_X does not contain any blanker vertices, as all such sets have been eliminated. Similarly, it does not contain B'_{j^*} or D'_{j^*}. The eliminations up until X'_{i^*} have made all of B' adjacent to $\bigcup_{i\in[r]} A'_i$. Vertices in a set $B'_j \cup D'_j$ for $j \neq j^*$ are adjacent to the checking vertices C'_j (by construction) and C'_{j^*} (because the elimination of B'_{j^*} introduced these adjacencies), but to no other checking vertices. Now consider the first vertex v that is eliminated after X'_{i^*}; by Definition 4 it is contained in some set B'_j with $j \neq j^*$. As no blanker vertex remains, and $B'_{j^*} \cup D'_{j^*}$ have been eliminated, there are exactly $2rn - 2n$ vertices left in B'. Vertex v is adjacent to all rn vertices in $\bigcup_{i\in[r]} A'_i$, to C'_{j^*}, and to the checking vertices corresponding to its own index. Hence the cost of v is $2rn - 2n + rn + n + n = 3rn$. The cost of the succeeding blankers D'_j in the same block is not more than that of v.

When eliminating the next group $B'_{j'}$, observe that $2n$ neighbors have been lost in B' (the set $B'_j \cup D'_j$ that was eliminated), whereas only n new neighbors have been introduced (the set C'_j). Hence the cost of later groups of vertices $B'_{j'}$ does not exceed the cost of v, and so does not exceed $3rn$. Finally, when all of B' has been eliminated then only the vertex set A' of size $2rn$ remains. At that point, no vertex can have cost more than $2rn$ as there are only $2rn$ vertices left in the graph. Thus the cost of

eliminating A' satisfies the claimed bound, after which the entire graph is eliminated. □

The next lemma links the cost of eliminating the set B'_{j^*} under a (i^*, j^*)-canonical elimination order for G', 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 during the construction of G' in Sect. 3.2, we created a one-to-one correspondence between $B_{i^*,j^*} = \{b^1_{i^*,j^*}, \ldots, b^n_{i^*,j^*}\}$ and $B'_{j^*} = \{\hat{b}^1_{j^*}, \ldots, \hat{b}^n_{j^*}\}$, 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'_i for $i \neq i^*$ 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 assumption 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 set B_{i^*,j^*} is a clique) 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 maximum cost of π' is attained when 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'.*

Proof Let π' be an elimination order for G' of cost at most k'. As A' is a clique in G', we may assume by Lemma 1 that π' eliminates all vertices of B' before any vertex of A' (Property 1). As each set X'_i forms a block in G', by Observation 1 we can adapt π' such that it eliminates the vertices of a set X'_i contiguously for all $i \in [r]$ (Property 2). Note that for $i \in [r]$ and vertices $d \in D'_i$ and $u \in B'_i$, we have $N_G[u] \subseteq N_G[d]$ by construction. Hence by Observation 1 we may assume that when a dummy $d \in D'_i$ is about to be eliminated, all vertices of the corresponding set B'_i are already eliminated (Property 3). Using these structural properties we proceed with the proof.

Consider the process of eliminating G' by π'. At some point, π' has eliminated $r-2$ distinct blocks of blankers $X'_{i_1}, \ldots, X'_{i_{r-2}}$. Consider the first vertex v_B of the blanker $X'_{i_{r-1}}$ that is eliminated after that point, and note that possibly non-blanker vertices are eliminated in between. Let G'_B be the graph obtained from G' by eliminating all vertices before v_B. □

Claim 1 *Graph G'_B still contains the vertices $\bigcup_{i\in[r]} B'_i \cup D'_i$: no vertex in this set is eliminated before v_B.*

Proof Assume for a contradiction that some vertex of $\bigcup_{i\in[r]} B'_i \cup D'_i$ is eliminated before v_B. We first show how to derive a contradiction when there is an index $j \in [r]$ such that B'_j is eliminated completely before v_B (Case 1). Afterward we show how to derive a contradiction when at least one vertex of B'_j remains in G'_B for all $j \in [r]$ (Case 2).

Case 1 If there is an index j such that no vertex of B'_j remains in G'_B, then let j^* be the index of the first set B'_j to be eliminated from G' completely. Consider the moment when the last vertex u of B'_{j^*} is eliminated. By our choice of j^* and Property 3, we know that no dummy vertex has been eliminated yet. So consider the closed neighborhood of u at the moment of its elimination. It contains all rn dummy vertices, as u is contained in the same clique B' as them. As at least two blocks of blankers remain, $N[u]$ contains at least $\frac{2n}{2}$ blanker vertices. We claim that u has become adjacent

to all vertices of $\bigcup_{i\in[r]} A'_i$. To see this, recall that u was the last vertex of B'_{j^*} to be eliminated. As we observed during the construction of G', there is a perfect matching between A'_i and B'_{j^*} for all $i \in [r]$. Hence for each vertex in $\bigcup_{i\in[r]} A'_i$, if u was not originally adjacent to it, then u has become adjacent to it by eliminating the vertex of B'_{j^*} that was matched to it. Thus u is indeed adjacent to all of $\bigcup_{i\in[r]} A'_i$. For each vertex in a set B'_j with $j \neq j^*$ that is eliminated before u, the elimination has made u adjacent to C'_j. By our choice of j^*, no such set B'_j is eliminated completely. Hence for each set B'_j with $j \neq j^*$ from which (less than n) vertices were eliminated, u has picked up n new neighbors in the set C'_j. Clearly, for each set B'_j with $j \neq j^*$ from which no vertices were eliminated, the neighborhood of u contains the n vertices in B'_j. So the number of neighbors of u in $\bigcup_{j\in[r]\setminus\{j^*\}} B'_j \cup C'_j$ is at least $(r-1)n$. Adding up the contribution of the blankers, the dummies, of $\bigcup_{i\in[r]} A'_i$, of $\bigcup_{j\in[r]\setminus\{j^*\}} B'_j \cup C'_j$, and of C'_{j^*}, to $N[u]$, we find that $|N[u]| \geq \frac{2n}{2} + rn + rn + (r-1)n + n \geq 3rn + n$. This value exceeds k', as $k < \frac{n}{2}$ by the definition of Cobipartite Graph Elimination. Hence we find a contradiction to the assumption that π' has cost at most k'.

Case 2 Assume now that for each $j \in [r]$ at least one vertex of B'_j remains in G'_B, which implies by Property 3 that all dummies are present in G'_B. Recall that we assumed, for a contradiction, that some vertex u of $\bigcup_{j\in[r]} B'_j \cup D'_j$ was eliminated before v_B. As u is no dummy, it is contained in some set B'_{j^*}. By the adjacency of B'_{j^*} to C'_{j^*}, the elimination has made the blanker $X'_{i_{r-1}}$ adjacent to C'_{j^*}. We will show that this causes the cost of v_B to exceed k'.

To see this, consider the neighbors of v_B in the various sets. For each set B'_j from which vertices were eliminated, we have eliminated less than n vertices (at least one vertex remains by the precondition to this case). For those sets B'_j, the blankers $X'_{i_{r-1}}$ have picked up adjacencies to the corresponding checkers C'_j. Thus $|N[v_B] \cap (\bigcup_{j\in[r]\setminus\{j^*\}} B'_j \cup C'_j)| \geq (r-1)n$; note carefully that the set $B'_{j^*} \cup C'_{j^*}$ is not counted here since its index is removed from $[r]$. As v_B is the first blanker vertex to be eliminated after $r-2$ blocks of blankers were already eliminated, there are two blocks of blankers left, giving $\frac{2n}{2}$ vertices in $N[v_B] \cap (\bigcup_{i\in[r]} X'_i)$. The prior eliminations of blankers $X'_{i_1}, \ldots, X'_{i_{r-2}}$ made $X'_{i_{r-1}}$ adjacent to the corresponding sets $A'_{i_1}, \ldots, A'_{i_{r-2}}$, and by construction $v_B \in X'_{i_{r-1}}$ is adjacent to $A'_{i_{r-1}}$. Now consider the remaining index $i_r \in [r] \setminus \{i_1, \ldots, i_{r-1}\}$, and let $i^* := i_r$ for convenience.

Recall that B'_{j^*} has a perfect matching into A'_{i^*} by the construction of G'. Hence for each vertex u that was eliminated from B'_{j^*}, vertex v_B has become adjacent to u's matching partner in the set A'_{i^*}. Hence, letting ℓ denote the number of vertices eliminated from B'_{j^*}, we know that v_B is adjacent to at least ℓ vertices in A'_{i^*}, showing that $|N[v_B] \cap (A'_{i^*} \cup B'_{j^*})| \geq (\ell + (n-\ell)) = n$. Summing up the contributions of the blankers, the dummies, the set $(\bigcup_{i\in[r]\setminus\{j^*\}} B'_i \cup C'_i)$, the set C'_{j^*}, the set $\bigcup_{i\in[r]\setminus\{i^*\}} A'_i$, and the set $A'_{i^*} \cup B'_{j^*}$ to $|N[v_B]|$, we find that $|N[v_B]| \geq \frac{2n}{2} + rn + (r-1)n + n + (r-1)n + n \geq 3rn + n > k'$, which is a contradiction to the assumption that the cost of π' is at most k'.

As the two cases are exhaustive, we have established that when v_B is eliminated, all vertices of $\bigcup_{i\in[r]} B'_i \cup D'_i$ still remain in the graph G'_B. □

We need two more claims to complete the proof of Lemma 5. As π' is block-contiguous with respect to the blankers (Property 2), after v_B it eliminates the rest of $X'_{i_{r-1}}$. Afterward only a single group of blankers remains, say X'_{i_r}.

Claim 2 *After eliminating $X'_{i_{r-1}}$, order π' eliminates a vertex in $\bigcup_{i\in[r]} B'_i$.*

Proof By Property 1, all vertices of B' are eliminated before any vertex of A'. Recall that B' consists of blankers $\bigcup_{i\in[r]} X'_i$ and the vertices $\bigcup_{i\in[r]} B'_i \cup D'_i$. As $X'_{i_{r-1}}$ is the $(r-1)$-th block of blankers to be eliminated, afterward the only vertices in B' remaining are X'_{i_r} and $\bigcup_{i\in[r]} B'_i \cup D'_i$. By Property 3, π' eliminates all vertices of B'_i before eliminating a dummy in the corresponding set D'_i. Hence if π' does not follow the elimination of $X'_{i_{r-1}}$ by a vertex of $\bigcup_{i\in[r]} B'_i$, then it eliminates X'_{i_r}. If this is the case, then all blankers have been eliminated before eliminating any vertex of $\bigcup_{i\in[r]} B'_i \cup D'_i$. Now consider the first vertex u of $\bigcup_{i\in[r]} B'_i$ that is eliminated, and suppose it is contained in B'_{j^*}. Eliminating all blankers has made B'_{j^*} adjacent to all of $\bigcup_{i\in[r]} A'_i$. By construction B'_{j^*} is adjacent to the n checking vertices C'_{j^*}. By Property 3 it is adjacent to all dummies. Summing up the contributions of the dummies, of $\bigcup_{i\in[r]} B'_i$, of $\bigcup_{i\in[r]} A'_i$, and of the single set C'_{j^*}, to $N[u]$, we find that the cost of u is at least $rn + rn + rn + n > k'$; a contradiction. □

Before proving the next claim, we make an observation. Let u be the first vertex of $\bigcup_{i\in[r]} B'_i$ that is eliminated by π', and suppose that $u \in B'_{j^*}$. The elimination of u makes the last group of blankers X'_{i_r} adjacent to the checking vertices C'_{j^*}, as B'_{j^*} is adjacent to C'_{j^*}. This implies that after the elimination of $u \in B'_{j^*}$, the closed neighborhood of X'_{i^*} is a superset of the closed neighborhood of any remaining vertex in B'_{j^*}. To see this, note that at that stage, X'_{i^*} is adjacent to the remainder of B', to $\bigcup_{i\in[r]\setminus\{i^*\}} A'_i$ (by eliminating the previous blankers), to A'_{i^*} (by construction), and to C'_{j^*} (by eliminating u). On the other hand, vertices in B'_{j^*} see the remainder of B', they see $\bigcup_{i\in[r]\setminus\{i^*\}} A'_i$, a subset of A'_{i^*} that depends on the edges in the graph G_{i^*,j^*}, and C'_{j^*}. Hence, by the same reasoning as in Observation 1, if a vertex $z \in X'_{i^*}$ is eliminated after the *first* vertex of B'_{j^*} (i.e., u) but before the *last* vertex of B'_{j^*}, then the cost of π' does not increase when eliminating all vertices of B'_{j^*} just before z. Hence we may assume that π' eliminates all of B'_{j^*} before any vertex of X'_{i^*}; we call this Property 4. We use this in the proof of the following claim.

Claim 3 *All vertices of B'_{j^*} are eliminated before any vertex of $\bigcup_{j\in[r]\setminus\{j^*\}} B'_j$.*

Proof By Property 4, all vertices of B'_{j^*} are eliminated before the last blanker X'_{i^*}. Now suppose that before eliminating the last vertex of B'_{j^*}, order π' eliminates some vertex $v \in B'_{j'}$ with $j' \neq j^*$. Let v be the first vertex with this property. By Property 4, all vertices in X'_{i^*} remain in the graph when v is eliminated. This causes the cost of v to exceed k'. To see this, observe that at the time of elimination, the closed

neighborhood of v contains all rn dummy vertices (by Property 3), it contains the $\frac{n}{2}$ vertices of X'_{i^*}, it contains C'_{j^*} (by elimination of u) and $C'_{j'}$ (by construction), which contain n vertices each. Additionally, $N[v]$ contains $\bigcup_{i\in[r]\setminus\{j^*\}} B'_i$ by our choice of v, and $\bigcup_{i\in[r]\setminus\{i^*\}} A'_i$ by the eliminations of earlier groups of blankers, for a subtotal of $rn + \frac{n}{2} + 2n + (r-1)n + (r-1)n = 3rn + \frac{n}{2}$. If ℓ vertices have been eliminated from B'_{j^*} prior to elimination of v, then $N[v]$ contains $n-\ell$ vertices from B'_{j^*}, but has gained ℓ neighbors in A'_{i^*} by the perfect matching between A'_{i^*} and B'_{j^*} in G'. Hence the remaining vertices in $A'_{i^*} \cup B'_{j^*}$ contribute at least $\ell + (n-\ell)$ vertices to the cost of v. Thus the cost of v is at least $3rn + \frac{n}{2} + n$, which is more than k'; a contradiction. □

Using Claims 1, 2, and 3, we prove Lemma 5. By Property 1, elimination order π' eliminates B' before A'. By Claim 1, π' did not eliminate any vertex of $\bigcup_{i\in[r]} B'_i \cup D'_i$ when the first vertex of the $(r-1)$-th block of blankers is eliminated. As π' is block-contiguous with respect to the blankers, its initial behavior matches that of a canonical elimination order (Definition 4): it eliminates $r-1$ distinct blocks of blankers $X'_{i_1}, \ldots, X'_{i_{r-1}}$ before any vertex of $\bigcup_{j\in[r]} B'_j \cup D'_j$. By Claim 2 it then eliminates a vertex of $\bigcup_{j\in[r]} B'_j$, say a vertex in B'_{j^*}. By Claim 3 it completes the elimination of B'_{j^*} before touching vertices in $\bigcup_{j\in[r]\setminus\{j^*\}} B'_j$, by Property 3 it eliminates B'_{j^*} before any dummy, and by Property 4 it eliminates B'_{j^*} before the last blanker X'_{i_r}. Hence after the $r-1$ blocks of blankers, the vertices of B'_{j^*} are eliminated consecutively.

Once this is done, the closed neighborhoods of D'_{j^*} and X'_{i^*} coincide: by the perfect matchings between B'_{j^*} and A'_i (for all $i \in [r]$) in G', eliminating all of B'_{j^*} made D'_{j^*} and X'_{i^*} adjacent to $\bigcup_{i\in[r]} A'_i$. Furthermore, $N[D'_{j^*}] \cap \bigcup_{i\in[r]} C'_i = N[X'_{i^*}] \cap \bigcup_{i\in[r]} C'_i = C'_{j^*}$: the dummies see C'_{j^*} by construction, while X'_{i^*} sees it because of the elimination of B'_{j^*}. The closed neighborhoods of D'_{j^*} and X'_{i^*} are subsets of the closed neighborhoods of the other vertices that remain in B' at that point: vertices in a set $B'_j \cup D'_j$ for $j \neq j^*$ see $\bigcup_{i\in[r]} A'_i$ together with both A'_{j^*} and A'_j, while the latter set is not seen by $D'_{j^*} \cup X'_{i^*}$. Hence by Observation 1 we may assume that after finishing B'_{j^*}, order π' eliminates D'_{j^*} followed by X'_{i^*}.

Once that is done, the only vertices remaining in B' are $\bigcup_{i\in[r]\setminus\{j^*\}} B'_i \cup D'_i$. It is easy to see that for any $j \in [r] \setminus \{j^*\}$, all vertices in $B'_j \cup D'_j$ have the same closed neighborhood at that stage, consisting of the remainder of B' together with $C'_j \cup C'_{i^*}$ and $\bigcup_{i\in[r]} A'_i$. By Observation 1 we may assume that π' is block-contiguous after eliminating X'_{i^*}, which means it eliminates the sets $B'_j \cup D'_j$ one at a time. As we may shuffle the order within a set $B'_j \cup D'_j$ without changing the cost (all closed neighborhoods of vertices from such a set are identical), we may assume that the remaining actions of π' on B' are alternatingly eliminating a set B'_j followed by the corresponding set D'_j, until all of B' is eliminated. Then π' finishes by eliminating A' in some order. As this form exactly matches the definition of an (i^*, j^*)-canonical elimination order, we have proved that whenever an elimination order of G' exists that has cost at most k', then there is one that is canonical. This proves Lemma 5. □

3.4 Proof of Theorem 3

Having analyzed the connection 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 *Let (G', k') be the result of applying the construction of Sect. 3.2 to a series of inputs $G_{i,j}$ of* Cobipartite Graph Elimination *for $i, j \in [r]$. Then G' has an elimination order of cost at most k' if and only if there are indices $i, j \in [r]$ such that $G_{i,j}$ has an elimination order of cost at most $n + k$.*

Proof ($\Rightarrow$) Assume that G' has an elimination order π' of cost at most k'. By Lemma 5 we may assume that π' is (i^*, j^*)-canonical, for appropriate choices of i^* and j^*. Take an elimination order π for G_{i^*,j^*} that agrees with π' on B_{i^*,j^*}. By Lemma 4 this gives $c_{G'}(\pi') = 3rn + \frac{n}{2} - n + c_{G_{i^*,j^*}}(\pi)$. Hence $c_{G_{i^*,j^*}}(\pi) = c_{G'}(\pi') - 3rn - \frac{n}{2} + n \leq k' - 3rn - \frac{n}{2} + n = n + k$. Thus G_{i^*,j^*} has an elimination order of cost at most $n + k$.

($\Leftarrow$) In the other direction, suppose that G_{i^*,j^*} has an elimination order π of cost at most $n + k$. As A_{i^*,j^*} is a clique in G_{i^*,j^*}, by Lemma 1 we may assume that π eliminates all vertices of B_{i^*,j^*} before any vertex of A_{i^*,j^*}. Using Definition 4 it is easy to see that a canonical elimination order π' for G' exists that agrees with π on B_{i^*,j^*}. By Lemma 4 the cost of π' on G' exceeds the cost of π on G_{i^*,j^*} by exactly $3rn + \frac{n}{2} - n$. So the cost of π' on G' is at most $3rn + \frac{n}{2} - n + (n + k) = k'$, which proves this direction of the claim. □

Lemma 7 *There is an* OR*-cross-composition of* Cobipartite Graph Elimination *into* Treewidth *[n] of cost $\sqrt{t}$.*

Proof In Sect. 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 at most 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 relation 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 Treewidth $[n]$ 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 Treewidth $[n]$ of cost $\sqrt{t}$. □

Theorem 3 follows from the combination of Lemmas 7, 2, and Theorem 1. Since the pathwidth of a cobipartite graph equals its treewidth [31] and the graph formed by the cross-composition is cobipartite, the same construction gives an OR-cross-composition of bounded cost into Pathwidth $[n]$.

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

The obtained quadratic (generalized) kernel lower bounds also apply to parameterizations of TREEWIDTH and PATHWIDTH by measures that do not exceed the number of vertices: if TREEWIDTH or PATHWIDTH parameterized by a measure ℓ with $\ell \leq n$ would have a generalized kernel of size $\mathcal{O}(\ell^{2-\epsilon})$ for some $\epsilon > 0$, then as $\ell \leq n$ this would immediately give a generalized kernel of size $\mathcal{O}(n^{2-\epsilon})$ for TREEWIDTH $[n]$ or PATHWIDTH $[n]$. The latter implies NP $\subseteq$ coNP/poly through Theorem 3 and Corollary 1.

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?

We remark that the vertex cover X does not have to be optimal, and may for example be obtained by a simple approximation algorithm. For a full discussion of this matter we refer the reader to the survey by Fellows et al. [22].

The presentation is organized as follows. In Sect. 4.1 we introduce the notion of a treewidth-invariant set, and derive some of its properties. In Sect. 4.2 we show how the q-expansion lemma can be used to obtain a simple kernel with $\mathcal{O}(|X|^2)$ vertices. Afterward, in Sect. 4.3, we show how the running time and size bounds of the kernelization algorithm can be optimized by bypassing the q-expansion lemma and extracting treewidth-invariant sets directly from a maximum matching in an auxiliary bipartite graph.

4.1 Treewidth-Invariant Sets

Our kernelization revolves around the new notion of a treewidth-invariant set.

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* $T \neq \emptyset$ *is a treewidth-invariant set in* G *and* $\Delta := \max_{v \in T} \deg_G(v)$*, then* $\mathrm{TW}(G) = \max(\Delta, \mathrm{TW}(\hat{G}_T))$.

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

($\geq$)**.** As $\hat{G}_T$ is a minor of G, we have $\mathrm{TW}(G) \geq \mathrm{TW}(\hat{G}_T)$ (cf. [2, Lemma 16]). If $\mathrm{TW}(\hat{G}_T) \geq \Delta$ then this implies the inequality. So assume that $\Delta > \mathrm{TW}(\hat{G}_T)$.

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 $\text{TW}(\hat{G}_T) + 1$ 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 $\text{TW}(\hat{G}_T) + 1$, the total cost of elimination order π on G is $\max(\Delta+1, \text{TW}(\hat{G}_T)+1)$. By Theorem 2 this completes this direction of the proof. □

Lemma 8 shows that when a treewidth-invariant set is eliminated from a graph, its treewidth changes in a controlled manner. For completeness we remark that we could also have defined a treewidth-invariant set as an independent set T such that $\hat{G}_T$ is a minor of $G - \{v^*\}$, where v^* is a vertex of maximum degree among T. For such sets, one can prove an analogue of Lemma 8: $\text{TW}(G) = \max(\deg_G(v^*), \text{TW}(\hat{G}_T))$. As the current definition is more uniform and sufficient for our purposes, we use that instead.

Based on Lemma 8 we can formulate the following reduction rule for deciding whether the treewidth of a graph G is at most k. Let T be a treewidth-invariant set in G. If $\max_{v \in T} \deg_G(v) > k$, then output NO as the treewidth of G exceeds k. Otherwise, reduce to the graph $\hat{G}_T$. It can be shown that exhaustively eliminating treewidth-invariant sets reduces the order of a graph to the square of its vertex cover number. If we were able to identify treewidth-invariant sets efficiently, this would therefore lead to a quadratic-vertex kernel for TREEWIDTH [VC]. Unfortunately, it seems difficult to detect such sets in all circumstances. To circumvent this issue, 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, based on the independent set $V(G) \setminus X$ and the non-edges in the graph $G[X]$, is needed for this procedure (see Fig. 2).

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 $E(\overline{G}[X])$, with an edge between a vertex $v \in V(G) \setminus X$ and a vertex $x_{\{p,q\}}$ representing $\{p, q\} \in E(\overline{G}[X])$ 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.*

Proof As T is a subset of the independent set $V(G) \setminus X$, the set T is independent in G. It remains to prove that for every $v \in T$, the graph $\hat{G}_T$ is a minor of $G - \{v\}$. So consider an arbitrary vertex $v^* \in T$. We give a series of minor operations that transforms $G - \{v^*\}$ into $\hat{G}_T$. The crucial part of the transformation consists of contracting vertices of $T \setminus \{v^*\}$ into vertices of X, to turn $N_G(v)$ into a clique for all $v \in T$; afterward we can simply delete all remaining vertices of $T \setminus \{v^*\}$. Let $f\colon S \to \binom{T}{2}$ be a mapping that assigns to each vertex $u \in S$ a set of two vertices in $N_{H_{G,X}}(u) \cap T$, such that the images of f are pairwise disjoint.

Consider a vertex $v \in T$ such that $N_G(v)$ is not a clique. Let $\{p, q\}$ be a non-edge in $G[N_G(v)]$. As v is adjacent to both p and q in G, vertex v is adjacent to the representative $x_{\{p,q\}}$ in $H_{G,X}$, implying that $x_{\{p,q\}} \in S$. Hence $x_{\{p,q\}}$ is saturated by a 2-star into T. Consider the two vertices $f(x_{\{p,q\}})$ assigned to $x_{\{p,q\}}$; at least one of them, say u, differs from v^*. As u is adjacent to $x_{\{p,q\}}$ in $H_{G,X}$ by definition of 2-star saturation, by definition of $H_{G,X}$ this implies that u is adjacent to both p and q. Hence contracting u into p creates the missing edge $\{p, q\}$. Now observe that as the images of f are pairwise disjoint, for each non-edge $\{p, q\}$ in the neighborhood of some vertex in T, there is a distinct vertex unequal to v^* that can be contracted to create the non-edge. Contracting all such vertices into appropriate neighbors therefore turns each set $N_G(v)$ for $v \in T$ into a clique. Hence we establish that $\hat{G}_T$ is indeed a minor of $G - \{v^*\}$, proving that T is treewidth-invariant. □

4.2 A Simple Kernelization Using the q-Expansion Lemma

The q-expansion lemma was formulated by Fomin et al. [24], and can be considered to be an algorithmic version of combinatorial insights dating back to the middle of the twentieth century [25]. It concerns sets that are saturated by q-stars in a bipartite graph. The special case of $q = 2$ corresponds to a reduction rule employed by Thomassé [33] in his quadratic kernel for k-Feedback Vertex Set.

q-Expansion Lemma ([24, 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 $A = E(\overline{G}[X])$ 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 $\text{TW}(G) \geq \deg_G(v) > k$.
- Otherwise we reduce to $(\hat{G}_T, X, k)$ and restart the algorithm: Lemma 8 guarantees that $\text{TW}(G) \leq k$ if and only if $\text{TW}(\hat{G}_T) \leq k$. Observe that X is a vertex cover of $\hat{G}_T$, since the only newly introduced edges have both endpoints in X.

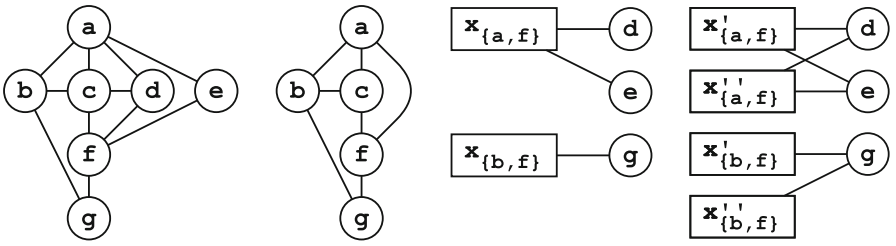
Each iteration takes polynomial time. As the number of vertices decreases in each iteration, there are at most n iterations until we fail to find a treewidth-invariant set. When that happens, we output the resulting instance. The q-expansion lemma ensures that at that point, $|B| \leq 2m$, where m is 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. □

4.3 Refined Analysis

The previous section showed how to obtain a quadratic-vertex kernel for TREEWIDTH [VC]. In this section we optimize the running time of the kernelization algorithm, and consider the effect of the reduction rule in more detail. The material in this section is stated for the optimization version of the problem, rather than the decision version, since the reduction rule does not depend on a threshold value k for the treewidth.

Lemma 10 *Let G be a graph with a vertex cover $X \subseteq V(G)$, and let $I := V(G) \setminus X$ be the corresponding independent set. Let $m_{\overline{G}[X]} := |E(\overline{G}[X])|$ be the number of non-edges in the graph $G[X]$. There is an algorithm that, given G and X, runs in time $\mathcal{O}(\min\{|I|, m_{\overline{G}[X]}, \sqrt{|I| + m_{\overline{G}[X]}}\} \cdot |I| \cdot m_{\overline{G}[X]} + |X|^2 + |V(G)| + |E(G)|)$ and computes a (possibly empty) treewidth-invariant set $T \subseteq I$ in G, such that $|T| \geq |I| - 2m_{\overline{G}[X]}$.*

Proof Given G and X, the goal is compute a set that is saturated by 2-stars in the graph $H_{G,X}$, which is treewidth-invariant by Lemma 9. For this purpose, we construct a bipartite graph $H'_{G,X}$. Intuitively, it is obtained from $H_{G,X}$ by replacing each vertex $x_{\{p,q\}}$ of $H_{G,X}$ (which corresponds to a non-edge $\{p,q\} \in E(\overline{G}[X])$ in the vertex cover) by two vertices $x'_{\{p,q\}}, x''_{\{p,q\}}$ with identical neighborhoods (see Fig. 2).



(a) Graph G. **(b)** $\hat{G}_{\{d,e\}}$. **(c)** $H_{G,\{a,b,c,f\}}$. **(d)** $H'_{G,\{a,b,c,f\}}$.

Fig. 2 **a** A graph G with a treewidth-invariant set $\{d, e\}$. **b** The graph $\hat{G}_{\{d,e\}}$ obtained from G by eliminating d and e. **c** The bipartite non-edge connection graph $H_{G,X}$ with respect to the vertex cover $X = \{a, b, c, f\}$ of G, per Definition 6. **d** The doubled graph $H'_{G,X}$ used in the proof of Lemma 10

Formally, one partite set B' of $H'_{G,X}$ consists of $I = V(G) \setminus X$, and the other partite set A' consists of:

$$\left\{ x'_{\{p,q\}}, x''_{\{p,q\}} \;\middle|\; \{p, q\} \in E(\overline{G}[X]) \right\}.$$

There are edges in $H'_{G,X}$ between a vertex $v \in V(G) \setminus X$ and the vertices $\{x'_{\{p,q\}}, x''_{\{p,q\}}\}$ if and only if $v \in N_G(p) \cap N_G(q)$. The sizes of the partite sets A' and B' of $H'_{G,X}$ are $2m_{\overline{G}[X]}$ and $|I|$, respectively, giving a trivial bound of $2 \cdot |I| \cdot m_{\overline{G}[X]}$ on the number of edges in $H'_{G,X}$. □

Claim 4 *$H'_{G,X}$ can be constructed in $\mathcal{O}(|X|^2 + |I| \cdot m_{\overline{G}[X]} + |V(G)| + |E(G)|)$ time.*

Proof The construction algorithm proceeds as follows. We start by inserting all the vertices I into the graph $H'_{G,X}$, storing pointers from their copies in G to their copies in $H'_{G,X}$. We create an adjacency matrix for $G[X]$ in $\mathcal{O}(|X|^2 + |V(G)| + |E(G)|)$ time by scanning through all edges of G and marking the edges between members of X in the matrix. Then we number the vertices in G from 1 to $|V(G)|$. For each vertex in X we make a copy of its adjacency list containing only its neighbors in the set I. We sort all these truncated adjacency lists simultaneously by a single radix sort, in $\mathcal{O}(|V(G)| + |E(G)|)$ time.

For each pair $\{p, q\} \in \binom{X}{2}$, we test using the adjacency matrix of $G[X]$ whether there is an edge between p and q in G. If this is not the case, we add the two vertices $\{x'_{\{p,q\}}, x''_{\{p,q\}}\}$ to $H'_{G,X}$ and use the sorted lists of p's and q's neighbors in I to find their common I-neighbors in $\mathcal{O}(|I|)$ time, using a linear scan as in merge-sort. For each common neighbor $v \in N_G(p) \cap N_G(q) \cap I$, we use the pointer in G's copy of v to $H'_{G,X}$'s copy of v to insert the edges from $x'_{\{p,q\}}$ and $x''_{\{p,q\}}$ to v in constant time. After doing this for each pair $\{p, q\} \in \binom{X}{2}$ we have constructed the graph $H'_{G,X}$. Since we spend constant time for each edge in $G[X]$, and $\mathcal{O}(|I|)$ time for each non-edge in $G[X]$, the overall running time is $\mathcal{O}(|X|^2 + |I| \cdot m_{\overline{G}[X]} + |V(G)| + |E(G)|)$. □

To find the desired treewidth-invariant set in G, we need to compute a maximum matching in the bipartite graph $H'_{G,X}$. The fastest algorithmic approach for perform-

ing this step depends on the balance between the sizes of the partite sets of $H'_{G,X}$, which causes the complicated running time. The Hopcroft–Karp algorithm [27] runs in time $\mathcal{O}(\sqrt{n}m)$ time for an n-vertex graph with m edges. In our setting, this gives a running time of $\mathcal{O}(\sqrt{|I| + m_{\overline{G}[X]}} \cdot |I| \cdot m_{\overline{G}[X]})$. Alternatively, we may use the Ford–Fulkerson method of repeatedly finding simple augmenting paths. Finding one augmenting path takes $\mathcal{O}(n+m)$ time and increases the size of the matching. Since the size of a matching in $H'_{G,X}$ cannot exceed the size $\min\{2m_{\overline{G}[X]}, |I|\}$ of its smallest partite set, this number bounds the number of iterations of the Ford–Fulkerson method, giving a running time guarantee of $\mathcal{O}(\min\{m_{\overline{G}[X]}, |I|\} \cdot |I| \cdot m_{\overline{G}[X]})$. Choosing the algorithm that gives the best run-time guarantee, we compute a maximum matching M' in $H'_{G,X}$ in time $\mathcal{O}(\min\{|I|, m_{\overline{G}[X]}, \sqrt{|I| + m_{\overline{G}[X]}}\} \cdot |I| \cdot m_{\overline{G}[X]})$.

Let U be the set of vertices in $H'_{G,X}$ that are not incident to an edge in M', and let $U_{A'} := U \cap A'$ be the unsaturated vertices in the A'-side of $H'_{G,X}$. Let $R(U_{A'})$ be the vertices in $H'_{G,X}$ that are reachable from a vertex in $U_{A'}$ by an alternating path, i.e., a path that begins with a non-matching edge and alternates between edges in $E(H'_{G,X}) \setminus M'$ and edges of M'. By a suitable breadth-first search, the set $R(U_{A'})$ can be computed in time linear in the size of $H'_{G,X}$. Finally, put $T := I \setminus R(U_{A'})$, i.e., T consists of the vertices in the B'-partite set which are not reachable from $U_{A'}$ by alternating paths. We will prove that T is the desired treewidth-invariant set in G. Note that if $|I| \leq 2m_{\overline{G}[X]}$ then T might be empty, which means it is trivially treewidth-invariant.

Claim 5 $|T| \geq |I| - 2m_{\overline{G}[X]}$.

Proof We first show that all vertices in $R(U_{A'}) \cap I$ are saturated by M'. Suppose there is a vertex in $R(U_{A'}) \cap I$ that is not saturated by M'. Then there is an alternating path that starts at an unsaturated vertex in $U_{A'}$, and ends at an unsaturated vertex in the other partite set $B' = I$. Hence the matching can be augmented over this alternating path, contradicting the fact that M' is a maximum matching.

Since all vertices in $R(U_{A'}) \cap I$ are saturated by M', and the number of edges in M' is at most $2m_{\overline{G}[X]}$ since that is the size of one of the partite sets of $H'_{G,X}$, it follows that $R(U_{A'}) \cap I \leq 2m_{\overline{G}[X]}$ and therefore that $|T| = |I \setminus R(U_{A'})| \geq |I| - 2m_{\overline{G}[X]}$. □

Claim 6 *For each vertex in T, all its neighbors in $H'_{G,X}$ are matched into T by M'.*

Proof Consider a vertex $v \in T = I \setminus R(U_{A'})$. It is easy to see that all vertices in $N_{H'_{G,X}}(v)$ are saturated by M': if v has an unsaturated neighbor u, then the edge $\{u, v\}$ would be an alternating path from an unsaturated vertex in $U_{A'}$ to v, showing that $v \in R(U_{A'})$ and resulting in a contradiction.

Now suppose that there is a vertex $t \in T \subseteq B'$ with a neighbor $u \in N_{H'_{G,X}}(t) \subseteq A'$ such that the matching partner $u' \in B'$ of $u \in A'$ is not contained in T, and observe that this means that $\{t, u\} \notin M'$. By definition of T, there is an alternating path from an unmatched $x \in U_{A'} \subseteq A'$ to $u' \in B'$. Since the alternating path starts with a non-matching edge, and the vertices x and u' are in different partite sets, the last edge of this path from x to u' is a non-matching edge. Hence we may add the matching

edge $\{u, u'\}$ and the non-matching edge $\{t, u\}$ to this alternating path, to obtain an alternating path from x to t. But this shows that $t \in R(U_{A'})$, a contradiction to the assumption that $t \in T = I \setminus R(U_{A'})$. □

Claim 7 *T is a treewidth-invariant set.*

Proof By Lemma 9, it suffices to prove that in the graph $H_{G,X}$, the set $N_{H_{G,X}}(T)$ is saturated by 2-stars into T. Define a mapping $f\colon N_{H_{G,X}}(T) \to \binom{T}{2}$ as follows. For each vertex $x_{\{p,q\}} \in N_{H_{G,X}}(T)$ there are two copies $x'_{\{p,q\}}, x''_{\{p,q\}}$ in $N_{H'_{G,X}}(T)$, since $N_{H_{G,X}}(x_{\{p,q\}}) = N_{H'_{G,X}}(x'_{\{p,q\}}) = N_{H'_{G,X}}(x''_{\{p,q\}})$ by construction. The previous claim shows that for each vertex in $N_{H_{G,X}}(T)$, the two corresponding copies are both matched into T by M'. Now let $f(x_{\{p,q\}})$ consist of the two distinct matching partners of $x'_{\{p,q\}}$ and $x''_{\{p,q\}}$, for all $x_{\{p,q\}} \in N_{H_{G,X}}(T)$. Since the endpoints of a matching are all distinct, this gives the desired saturation by 2-stars into T. □

The claims show that T is a treewidth-invariant set of size at least $|I| - 2m_{\overline{G}[X]}$. Since it can be computed within the claimed time bounds, the lemma follows. □

The structure of q-expansions in graphs is related to that of crown decompositions, which are q-expansions for $q = 1$. We refer the reader to the paper by Chlebík and Chlebíková [12] for a detailed treatise on crowns. Using the previous lemma to extract treewidth-invariant sets from matchings in bipartite graphs, we obtain the following reduction algorithm.

Theorem 5 *Let G be a graph with a vertex cover $X \subseteq V(G)$, and let $I := V(G) \setminus X$ be the corresponding independent set. Let $m_{\overline{G}[X]} := |E(\overline{G}[X])|$ be the number of non-edges in the graph $G[X]$. There is an algorithm that, given G and X, runs in time $\mathcal{O}(|V(G)| + |E(G)| + |X|^2 + \min\{\sqrt{|I|}, m_{\overline{G}[X]}\} \cdot |I| \cdot m_{\overline{G}[X]})$, and computes an integer Δ and a minor $\hat{G}_T$ of G with the same vertex cover X, such that:*

1. $\mathrm{TW}(G) = \max(\mathrm{TW}(\hat{G}_T), \Delta)$, *and*
2. *$\hat{G}_T$ has at most $|X| + 2m_{\overline{G}[X]} \leq |X|^2$ vertices.*

Proof Let the input be (G, X). The main idea is to apply Lemma 10 to find a treewidth-invariant set T and reduce to the graph $\hat{G}_T$, which is suitably small since T contains all but $2m_{\overline{G}[X]}$ vertices of I. We do a fine-grained analysis to optimize the overall running time.

If $|I| \leq 2m_{\overline{G}[X]}$ then the graph already satisfies the claimed size bounds, and we may simply output (G, X) unchanged and set $\Delta := 1$. In the remainder we therefore assume that $|I| > 2m_{\overline{G}[X]}$, which will be used to simplify the expressions of the running time.

The algorithm applies Lemma 10 to compute a treewidth-invariant set $T \subseteq I$ with the guarantee that $|T| \geq |I| - 2m_{\overline{G}[X]}$. Since $|I| > 2m_{\overline{G}[X]}$ the set T cannot be empty. Let $\Delta := \max_{v \in T} \deg_G(v)$. Using that $|I| > 2m_{\overline{G}[X]}$ we find that

$$\min\left\{m_{\overline{G}[X]}, |I|, \sqrt{m_{\overline{G}[X]} + |I|}\right\} \in \mathcal{O}\left(\min\{m_{\overline{G}[X]}, \sqrt{|I|}\}\right),$$

so the running time in Lemma 10 becomes $\mathcal{O}(\min\{\sqrt{|I|}, m_{\overline{G}[X]}\} \cdot |I| \cdot m_{\overline{G}[X]} + |X|^2 + |V(G)| + |E(G)|)$.

Once the treewidth-invariant set T is found, we compute $\Delta := \max_{v \in T} \deg_G(v)$. The last task is to build $\hat{G}_T$ as the output of the procedure. To avoid introducing a term $\Theta(|X|^2 \cdot |T|)$ in the running time, we do this carefully. To form $\hat{G}_T$ from G, we need to remove the vertices of T from G and turn $N_G(v)$ into a clique for each $v \in T$. To do the latter, we need to add the edges $E^* := \bigcup_{v \in T} \binom{N_G(v)}{2} \setminus E(G)$ to the graph. Since $T \subseteq V(G) \setminus X$, and X is a vertex cover, all new edges in E^* have both endpoints in X, implying that $|E^*| \leq m_{\overline{G}[X]}$. We utilize the graph $H'_{G,X}$ that was computed during the application of Lemma 10 to compute the set E^*. Since $N_{H'_{G,X}}(T)$ has exactly two vertices $x'_{\{p,q\}}, x''_{\{p,q\}}$ for each pair $\{p, q\} \in E^*$, we can find the set E^* in $\mathcal{O}(m_{\overline{G}[X]} \cdot |T|)$ time by finding the neighbors of T in $H'_{G,X}$.

Using E^* we build $\hat{G}_T$ as follows. We compute an adjacency matrix for $G[X]$ in $\mathcal{O}(|X|^2 + |V(G)| + |E(G)|)$ time. We make all the pairs in E^* new edges in the adjacency matrix, in $\mathcal{O}(|E^*|) \subseteq \mathcal{O}(m_{\overline{G}[X]})$ time. Then we remove the vertices in T from G, and compute an adjacency list representation of the resulting graph $\hat{G}_T$. Overall, the construction of $\hat{G}_T$ from T takes $\mathcal{O}(m_{\overline{G}[X]} \cdot |I| + |V(G)| + |E(G) + |X|^2)$ time (we use that $|T| \leq |I|$).

It is easy to verify that X is a vertex cover of the graph $\hat{G}_T$, as all edges that were introduced have both their endpoints in X. By Lemma 8 we know that $\text{TW}(G) = \max(\text{TW}(\hat{G}_T), \Delta)$. Since $|T| \geq |I| - 2m_{\overline{G}[X]}$, at most $2m_{\overline{G}[X]}$ vertices remain in I after all vertices of T have been removed. Hence the order of $\hat{G}_T$ is at most $|X| + 2m_{\overline{G}[X]} \leq |X| + 2\binom{X}{2} = |X|^2$. By the definition of a treewidth-invariant set, the graph $\hat{G}_T$ is a minor of $G - \{v\}$ for each $v \in T$, and is therefore a minor of G. Hence the pair $(\hat{G}_T, \Delta)$ is a valid output for the procedure. It is easy to see that the running time of the overall procedure is dominated by the invocation of Lemma 10, giving a bound of $\mathcal{O}(\min\{\sqrt{|I|}, m_{\overline{G}[X]}\} \cdot |I| \cdot m_{\overline{G}[X]} + |X|^2 + |V(G)| + |E(G)|)$. □

Theorem 5 should prove useful for building preprocessing algorithms that simplify the task of computing good tree decompositions. For that purpose, it is interesting to note that when a decomposition of the reduced graph is found, it is easy to lift this back to a decomposition of the original graph. The argument given in Lemma 8 shows that if the graph G is reduced to $\hat{G}_T$ by eliminating the treewidth-invariant set T, then an elimination order π' of $\hat{G}_T$ can be transformed into an elimination order of G of cost $\max(\Delta + 1, c_{\hat{G}_T}(\pi'))$ by prepending the vertices of T as a prefix in the new order. Hence even if a non-optimal elimination order of $\hat{G}_T$ is found, no additional loss is incurred when transforming it back to an elimination order of G. It is straight-forward to adapt this argument to transform tree decompositions of $\hat{G}_T$ into tree decompositions of G in polynomial time. Phrased in this language, the main insight is that for every eliminated vertex $v \in T$, any tree decomposition of $\hat{G}_T$ has a bag containing all vertices in $N_G(v)$, as $N_G(v)$ forms a clique in $\hat{G}_T$ [2, Lemma 4].

The vertex cover $X \subseteq V(G)$ needed in the input of the procedure may be obtained by a simple approximation or heuristic algorithm: the non-optimality of the vertex cover only affects the amount of data reduction that is achieved, and does not affect the fact that optimal tree decompositions of the original graph can be obtained from optimal tree decompositions of the reduced graph. Together with the fact that VERTEX

COVER has a simple 2-approximation algorithm (cf. [14]), the discussion above gives the following corollary.

Corollary 2 *There is a polynomial-time algorithm that, given a graph G, computes a minor $\hat{G}_T$ of G and an integer Δ such that:*

1. $|V(\hat{G}_T)| \leq 4 \cdot \text{VC}(G)^2$,
2. $\text{TW}(G) \geq \Delta$,
3. *a tree decomposition of width k for $\hat{G}_T$ can be transformed into a tree decomposition of G of width $\max(k, \Delta)$ in polynomial time.*

Here, $\text{VC}(G)$ denotes the size of a minimum vertex cover in G.

4.4 Consequences for Treewidth Obstructions

From the insights used to obtain the kernelization algorithm, we can also derive some properties of minor-minimal obstructions for treewidth (cf. [13,30]).

Corollary 3 *If G is a non-complete graph such that $\text{TW}(G') < \text{TW}(G)$ for all proper minors G' of G, and X is a vertex cover of G, then $|V(G)| \leq |X| + 2|E(\overline{G}[X])| \leq |X|^2$.*

Proof If G is not a complete graph (clique) and all proper minors of G have smaller treewidth than G itself, then G does not have a non-empty treewidth-invariant set. To see this, suppose that the non-complete graph G has a non-empty treewidth-invariant set T. By Lemma 8, we have $\text{TW}(G) = \max(\text{TW}(\hat{G}_T), \max_{v \in T} \deg_G(v))$. If $\text{TW}(G) = \text{TW}(\hat{G}_T)$, then the graph $\hat{G}_T$ is a minor of G (by Definition 5) with the same treewidth, and it is a proper minor since it does not contain T. If $\text{TW}(G) = \max_{v \in T} \deg_G(v)$, then let $v \in T$ be a vertex with maximum degree. As shown in the proof of Lemma 8 there is a clique minor of G on vertex set $N_G[v]$. This clique minor has treewidth $\deg_G(v) = \text{TW}(G)$, and is a proper minor since it is a clique, whereas G is not a clique by assumption. Hence in both cases we find that G has a proper minor with the same treewidth.

Now consider a non-complete graph G whose proper minors have smaller treewidth than G, and a vertex cover $X \subseteq V(G)$. Since G does not have a non-empty treewidth-invariant set by the above argument, by Lemma 10 we have that $|I| = |V(G) \setminus X| \leq 2|E(\overline{G}[X])|$. Hence the number of vertices in G is bounded by $|X| + |I| \leq |X| + 2|E(\overline{G}[X])| \leq |X| + 2\binom{X}{2} = |X|^2$. □

5 Conclusion

In this paper we contributed to the knowledge of sparsification for TREEWIDTH by establishing lower and upper bounds. As a lower bound, we proved that instances of TREEWIDTH $[n]$ and PATHWIDTH $[n]$ cannot be compressed in polynomial time to $\mathcal{O}(n^{2-\epsilon})$ bits (for $\epsilon > 0$) unless NP $\subseteq$ coNP/poly. Our upper bound consisted of a new reduction rule for TREEWIDTH [VC] based on the novel notion of treewidth-invariant sets, resulting in a kernel with $|X|^2$ vertices when given a vertex cover X of the graph. Our work raises questions in three different areas: sparsification, kernelization, and graph minor obstruction sets.

5.1 Sparsification

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. For example, does VERTEX COVER on unit-disk graphs admit a polynomial-time compression into instances of size $\mathcal{O}(n^{2-\epsilon})$, for some $\epsilon > 0$? If so, does the natural parameterization k-VERTEX COVER admit a kernel with $\mathcal{O}(k^{2-\epsilon})$ bits when the problem is restricted to unit-disk graphs? The results of Dell and van Melkebeek [17] imply that this is not the case on general graphs, assuming NP $\not\subseteq$ coNP/poly.

5.2 Kernelization

In Sect. 4 we gave a quadratic-vertex kernel for TREEWIDTH [VC]. We showed that the kernelization algorithm can be adapted to the optimization setting of computing tree decompositions, since decompositions of the reduced graph can easily be transformed back to decompositions of the original graph. Many reduction rules formulated for decision problems depend on the threshold value k of the question, which makes them harder to use in optimization settings. Our new reduction rule does not depend on a threshold value in its input, and can therefore be easily applied for optimization purposes (cf. [10]).

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. The treewidth-invariant sets that are identified by the current algorithm have the following special form. On input (G, X), they consist of vertices T from the independent set $I := V(G) \setminus X$ such that for each non-edge $\{p, q\}$ that exists in $\overline{G}[N_G(v)]$ for some $v \in T$, there are two unique vertices in T assigned to the non-edge that can be contracted into p or q to realize it. Hence these treewidth-invariant sets do not utilize the possibility of creating many non-edges simultaneously by a single contraction. Furthermore, the reduction algorithm finds such treewidth-invariant sets with respect to a particular choice of vertex cover X. It would be interesting to determine whether treewidth-invariant sets of the described "conservative" form can be found in polynomial time without specifying a particular choice of vertex cover. We strongly suspect that testing whether a graph has a non-empty treewidth-invariant set is NP-complete in general.

The refined analysis in Sect. 4.3 reveals that the number of vertices in the independent set $V(G) \setminus X$ of reduced instances is bounded by twice the number of *non-edges* in the graph induced by X. On graphs that have a dense vertex cover, this bound may

be much better than the naive bound $\binom{|X|}{2}$ of the number of pairs in X. This also raises some interesting algorithm engineering challenges. At first glance, one would expect that the reduction algorithm performs best when the vertex cover supplied in the input is as small as possible. However, it might be the case that there is a larger vertex cover that induces fewer non-edges. In this case, the analysis suggests that the reduction algorithm is able to reduce the graph better with respect to the larger vertex cover than the smaller one. It would be interesting to determine if this is the case in practice, and to develop good (heuristic) algorithms for finding a vertex cover that induces a small number of non-edges.

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 [6]. 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].

5.3 Obstruction Sets

As a corollary to our insights on treewidth-invariant sets, we proved a bound on the sizes of graphs G for which every minor operation decreases their treewidth. The bound is expressed in terms of the vertex cover number. Such graphs G are easily seen to be the minor-minimal obstructions to having treewidth $\text{TW}(G) - 1$. Hence Corollary 3 shows that the order of minor-minimal treewidth obstructions is at most quadratic in their vertex cover number. It would be interesting to determine whether this bound is tight. We conjecture that it is not. This conjecture is motivated by our inability to construct quadratic-size treewidth obstructions. The right answer may well be that for minor-minimal obstructions to treewidth, the number of vertices is linear in the vertex cover number. If this is indeed the case, then using the approach described by Fellows and the present author [21] this would imply that TREEWIDTH [VC] has a coNP-kernel with $\mathcal{O}(|X|^2)$ bits, and therefore that no lower bound better than $\mathcal{O}(|X|^{2-\epsilon})$ can be proven using the current frameworks [8,17].[1] As insights into obstruction sets and kernels often come in tandem, such research might also lead to an improved kernel for TREEWIDTH [VC].

[1] A coNP-kernel of size $f(k)$ is a nondeterministic polynomial-time algorithm that, given a YES-instance (x, k), outputs a YES-instance (x', k') with $|x'|, k' \leq f(k)$ on each computation path. Given a NO-instance, it outputs an instance of size $f(k)$ on each computation path, and at least one of the computation paths yields a NO-instance.

References

1. Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM J. Algebr. Discret. Methods **8**, 277–284 (1987). doi:10.1137/0608024
2. Bodlaender, H.L.: A partial k-arboretum of graphs with bounded treewidth. Theoret. 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: Proceedings of the 14th Annual European Symposium on Algorithms, Lecture Notes in Computer Science, vol. 4168, pp. 672–683 (2006). doi:10.1007/11841036_60
5. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (Meta) Kernelization. In: Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, pp. 629–638 (2009). doi:10.1109/FOCS.2009.46
6. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernel bounds for structural parameterizations of pathwidth. In: Proceedings of the 13th Scandinavian Symposium and Workshops on Algorithm Theory, Lecture Notes in Computer Science, vol. 7357, pp. 352–363 (2012). doi:10.1007/978-3-642-31155-0_31
7. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Preprocessing for treewidth: a combinatorial analysis through kernelization. SIAM J. Discret. Math. **27**(4), 2108–2142 (2013). doi:10.1137/120903518
8. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. SIAM J. Discret. Math. **28**(1), 277–305 (2014). doi:10.1137/120880240
9. Bodlaender, H.L., Koster, A.M.C.A.: Safe separators for treewidth. Discret. Math. **306**(3), 337–350 (2006). doi:10.1016/j.disc.2005.12.017
10. 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
11. Buss, J.F., Goldsmith, J.: Nondeterminism within P. SIAM J. Comput. **22**(3), 560–572 (1993). doi:10.1137/0222038
12. Chlebík, M., Chlebíková, J.: Crown reductions for the minimum weighted vertex cover problem. Discret. Appl. Math. **156**(3), 292–312 (2008). doi:10.1016/j.dam.2007.03.026
13. Chlebíková, J.: The structure of obstructions to treewidth and pathwidth. Discret. Appl. Math. **120**(1–3), 61–71 (2002). doi:10.1016/S0166-218X(01)00281-5
14. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)
15. Cygan, M., Grandoni, F., Hermelin, D.: Tight kernel bounds for problems on graphs with small degeneracy. In: Proceedings of the 21st Annual European Symposium on Algorithms, Lecture Notes in Computer Science, vol. 8125, pp. 361–372 (2013). doi:10.1007/978-3-642-40450-4_31
16. Dell, H., Marx, D.: Kernelization of packing problems. In: Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 68–81 (2012)
17. Dell, H., van Melkebeek, D.: Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In: Proceedings of the 42nd ACM Symposium on Theory of Computing, pp. 251–260 (2010). doi:10.1145/1806689.1806725
18. Downey, R., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, New York (1999)
19. Drucker, A.: New limits to classical and quantum instance compression. In: Proceedings of the 53rd Annual Symposium on Foundations of Computer Science, pp. 609–618 (2012). doi:10.1109/FOCS.2012.71
20. van den Eijkhof, F., Bodlaender, H.L., Koster, M.C.A.: Safe reduction rules for weighted treewidth. Algorithmica **47**(2), 139–158 (2007). doi:10.1007/s00453-006-1226-x
21. Fellows, M.R., Jansen, B.M.P.: FPT is characterized by useful obstruction sets. In: Proceedings of the 39th International Workshop on Graph-Theoretic Concepts in Computer Science, pp. 261–273 (2013). doi:10.1007/978-3-642-45043-3_23
22. Fellows, M.R., Jansen, B.M.P., Rosamond, F.: Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. Eur. J. Comb. **34**(3), 541–566 (2013). doi:10.1016/j.ejc.2012.04.008
23. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, New York (2006)

24. Fomin, F.V., Lokshtanov, D., Misra, N., Philip, G., Saurabh, S.: Hitting forbidden minors: Approximation and kernelization. In: Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science, pp. 189–200 (2011). doi:10.4230/LIPIcs.STACS.2011.189
25. Halmos, P.R., Vaughan, H.E.: The marriage problem. Am. J. Math. **72**(1), 214–215 (1950)
26. Hermelin, D., Wu, X.: Weak compositions and their applications to polynomial lower bounds for kernelization. In: Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 104–113 (2012)
27. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. **2**(4), 225–231 (1973). doi:10.1137/0202019
28. 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
29. Jansen, B.M.P.: The power of data reduction: Kernels for Fundamental Graph Problems. Ph.D. Thesis, Utrecht University, The Netherlands (2013)
30. Lucena, B.: Achievable sets, brambles, and sparse treewidth obstructions. Discret. Appl. Math. **155**(8), 1055–1065 (2007). doi:10.1016/j.dam.2006.11.006
31. Möhring, R.H.: Triangulating graphs without asteroidal triples. Discret. Appl. Math. **64**(3), 281–287 (1996). doi:10.1016/0166-218X(95)00095-9
32. Monien, B., Sudborough, I.H.: Min cut is NP-complete for edge weighted trees. Theoret. Comput. Sci. **58**, 209–229 (1988). doi:10.1016/0304-3975(88)90028-X
33. Thomassé, S.: A $4k^2$ kernel for feedback vertex set. ACM Trans. Algorithms **6**(2) (2010). doi:10.1145/1721837.1721848
34. Yap, C.K.: Some consequences of non-uniform conditions on uniform classes. Theoret. Comput. Sci. **26**, 287–300 (1983). doi:10.1016/0304-3975(83)90020-8