Tight Algorithms for Connectivity Problems Parameterized by Clique-Width

Falko Hegerfeld
Humboldt-Universität zu Berlin, Germany

Stefan Kratsch
Humboldt-Universität zu Berlin, Germany

Abstract

The complexity of problems involving global constraints is usually much more difficult to understand than the complexity of problems only involving local constraints. In the realm of graph problems, connectivity constraints are a natural form of global constraints. We study connectivity problems from a fine-grained parameterized perspective. In a breakthrough result, Cygan et al. (TALG 2022) first obtained Monte-Carlo algorithms with single-exponential running time $\alpha^{\mathrm{tw}} n^{\mathcal{O}(1)}$ for connectivity problems parameterized by treewidth by introducing the cut-and-count-technique, which reduces many connectivity problems to locally checkable counting problems. Furthermore, the obtained bases α were shown to be optimal under the Strong Exponential-Time Hypothesis (SETH).

However, since only sparse graphs may admit small treewidth, we lack knowledge of the fine-grained complexity of connectivity problems with respect to dense structure. The most popular graph parameter to measure dense structure is arguably clique-width, which intuitively measures how easily a graph can be constructed by repeatedly adding bicliques. Bergougnoux and Kanté (TCS 2019) have shown, using the rank-based approach, that also parameterized by clique-width many connectivity problems admit single-exponential algorithms. Unfortunately, the obtained running times are far from optimal under SETH.

We show how to obtain optimal running times parameterized by clique-width for two benchmark connectivity problems, namely Connected Vertex Cover and Connected Dominating Set. These are the first tight results for connectivity problems with respect to clique-width and these results are obtained by developing new algorithms based on the cut-and-count-technique and novel lower bound constructions. Precisely, we show that there exist one-sided error Monte-Carlo algorithms that given a k-clique-expression solve

- Connected Vertex Cover in time $6^k n^{\mathcal{O}(1)}$, and
- Connected Dominating Set in time $5^k n^{\mathcal{O}(1)}$.

Both results are shown to be tight under SETH.

2012 ACM Subject Classification Mathematics of computing → Paths and connectivity problems; Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Combinatorial algorithms

Keywords and phrases Parameterized Complexity, Connectivity, Clique-width, Cut&Count, Lower Bound

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.59

Related Version *Full Version*: `https://arxiv.org/abs/2302.03627`

Funding *Falko Hegerfeld*: Partially supported by DFG Emmy Noether-grant (KR 4286/1).

1 Introduction

One way to cope with the NP-hardness of a problem is the theory of parameterized complexity, where we seek to solve structured instances faster than worst-case instances; an additional parameter quantifies how structured an instance is. Ideally, we obtain fixed-parameter tractable algorithms with running time[1] $\mathcal{O}^*(f(k))$, where k is the parameter and f some computable function. Having established the existence of such an algorithm, the next natural step is to take a fine-grained perspective and to determine the smallest possible function f, which quantifies the precise impact of the considered structure on problem complexity.

We apply this approach to connectivity problems. Our investigation starts with the breakthrough results of Cygan et al. [16], who for the first time obtained Monte-Carlo algorithms with running time $\mathcal{O}^*(\alpha^{\mathrm{tw}})$, for some constant *base* $\alpha > 1$, for connectivity problems parameterized by *treewidth* (tw). These algorithms are obtained via the so-called cut-and-count-technique, which reduces connectivity problems to locally checkable counting problems. In addition, the obtained bases α were proven to be optimal assuming the Strong Exponential-Time Hypothesis (SETH) [14].

As only sparse graphs may have small treewidth, we lack knowledge of the precise complexity of connectivity problems with respect to dense structure. In the regime of dense graphs, *clique-width* (cw) is one of the most popular parameters. Bergougnoux [2] applied cut-and-count to several width-parameters based on structured neighborhoods with clique-width among these. Moreover, Bergougnoux and Kanté [4], building upon the rank-based approach of Bodlaender et al. [8], obtain single-exponential running times $\mathcal{O}^*(\alpha^{\mathrm{cw}})$ for a large class of connectivity problems parameterized by clique-width. As both articles are aimed at obtaining a breadth of single-exponential algorithms for a large class of problems, the Connected (Co-)(σ, ρ)-Dominating Set problems, the obtained bases for particular problems are far from being optimal. For example, the former article implies an $\mathcal{O}^*(128^{\mathrm{cw}})$-time algorithm for Connected Dominating Set and the latter yields an $\mathcal{O}^*((27 \cdot 2^{\omega+1})^{\mathrm{cw}})$-time algorithm for Connected Vertex Cover and an $\mathcal{O}^*((8 \cdot 2^{\omega+1})^{\mathrm{cw}})$-time algorithm for Connected Dominating Set, where ω is the matrix multiplication exponent, see e.g. Alman and Vassilevska W. [1]. Even if $\omega = 2$, this only yields the large bases 216 and 64 respectively.

We show that the running times for Connected Vertex Cover and Connected Dominating Set parameterized by clique-width can be considerably optimized by providing novel algorithms. These faster algorithms again rely on the cut-and-count-technique and are fine-tuned by precisely analyzing which cut-and-count states are necessary to consider. Moreover, we use further techniques such as fast subset convolution, inclusion-exclusion states, and distinguishing between live and dead labels to obtain the improved running times.

▶ **Theorem 1.1.** *There are one-sided error Monte-Carlo algorithms that, given a k-expression[2] for a graph G, can solve*

- *Connected Vertex Cover in time $\mathcal{O}^*(6^k)$,*
- *Connected Dominating Set in time $\mathcal{O}^*(5^k)$.*

We show that these algorithms are essentially the correct ones for these problem-parameter-combinations by proving that the obtained running times are optimal under SETH. To prove these lower bounds, we follow the by now standard construction principle of Lokshtanov et al. [35] for lower bounds relative to width-parameters. To apply this principle for clique-width, we closely investigate the problem behavior across *joins*, i.e., the edge-structures via which clique-width is defined, and the results of this investigation strongly guide us in designing appropriate gadgets. Precisely, we obtain the following tight lower bounds:

[1] The $\mathcal{O}^*$-notation hides polynomial factors in the input size.
[2] A k-expression witnesses that the clique-width of G is at most k.

Table 1 Optimal running-times of several connectivity problems with respect to various width-parameters listed in increasing generality. The results in the last column are obtained in this paper. Between modular-treewidth and clique-width, we only have the relationship $\mathrm{cw}(G) \leq 3 \cdot 2^{\text{mod-tw}(G)-1}$, but the same results are also tight for the more restrictive modular-pathwidth, where we have $\mathrm{cw}(G) \leq \text{mod-pw}(G) + 2$ by Hegerfeld and Kratsch [26]. The "?" marks problem-parameter combinations, where an algorithm with single-exponential running time is known by Bergougnoux and Kanté [4], but a gap between the lower bound and algorithm remains.

Parameters	cutwidth	treewidth	modular-tw	clique-width
Connected Vertex Cover	$\mathcal{O}^*(2^k)$	$\mathcal{O}^*(3^k)$	$\mathcal{O}^*(5^k)$	$\mathcal{O}^*(6^k)$
Connected Dominating Set	$\mathcal{O}^*(3^k)$	$\mathcal{O}^*(4^k)$	$\mathcal{O}^*(4^k)$	$\mathcal{O}^*(5^k)$
Steiner Tree	$\mathcal{O}^*(3^k)$	$\mathcal{O}^*(3^k)$	$\mathcal{O}^*(3^k)$	?
Feedback Vertex Set	$\mathcal{O}^*(2^k)$	$\mathcal{O}^*(3^k)$	$\mathcal{O}^*(5^k)$	?
References	[9]	[15, 16]	[26]	here

▶ **Theorem 1.2.** *Assuming SETH*[3], *the following statements hold for all $\varepsilon > 0$:*
- *Connected Vertex Cover (with unit costs) cannot be solved in time $\mathcal{O}^*((6 - \varepsilon)^{\mathrm{cw}})$.*
- *Connected Dominating Set (with unit costs) cannot be solved in time $\mathcal{O}^*((5 - \varepsilon)^{\mathrm{cw}})$.*

This work is part of a research program to determine the optimal running times for connectivity problems relative to several width-parameters ranging from restrictive to more and more general ones, hence yielding a *fine-grained* understanding of the *price of generality*. We summarize the known results in Table 1. The cut-and-count-technique by Cygan et al. [15, 16] together with their lower bounds settle the complexity relative to treewidth (and pathwidth) for many connectivity problems. Bojikian et al. [9] consider the more restrictive *cutwidth* and combine cut-and-count with the rank-based approach to improve upon the treewidth-algorithms or provide new lower bound constructions with low cutwidth when no improved algorithm exists. Hegerfeld and Kratsch [26] consider the parameter *modular-treewidth* which lifts treewidth into the dense regime by combining tree decompositions with modular decompositions and thus serves as a natural intermediate step between treewidth and clique-width. The results on modular-treewidth are obtained by reducing directly to the treewidth-case or by applying the cut-and-count-technique and the modular structure to essentially reduce to a more involved problem parameterized by treewidth; in the latter case, new lower bound constructions are provided that follow similar high-level principles as here, but that have to adhere to different design restrictions. Cygan et al. [16] observe that connectivity increases the base by at most 1 in the sparse setting, e.g., Vertex Cover has optimal base 2, see Lokshtanov et al. [35], and Connected Vertex Cover has optimal base 3 parameterized by treewidth. For clique-width, this increase can be larger and the impact of connectivity can even flip the relative complexities, e.g., the optimal bases of Vertex Cover and Dominating Set are 2 and 4 [29, 32] which increase to 6 and 5, respectively, upon adding the connectivity constraint.

Further Related Work. Beyond these tight results, the cut-and-count-technique has also been applied to branchwidth [42] and treedepth [23, 38]. Due to its reliance on the isolation lemma, the cut-and-count-technique yields randomized algorithms. The rank-based approach of Bodlaender et al. [8] and the matroid-based techniques of Fomin et al. [19, 20] deal

[3] If we instead assume an appropriate random variant of SETH, then these reductions also rule out Monte-Carlo algorithms with such running times.

with this shortcoming at the cost of a higher running time and the rank-based approach can also be applied in other contexts. By combining the rank-based approach with other techniques to avoid Gaussian elimination, optimal running times can be obtained in some cases, such as for Hamiltonian Cycle parameterized by pathwidth [12, 13] or coloring problems parameterized by cutwidth [22, 31]. There are further applications of the rank-based approach to connectivity problems relative to dense width-parameters, such as rankwidth [5] and mim-width [3]. We also refer to the survey of Nederlof on rank-based methods [37].

Moving away from connectivity problems, we survey some more of the literature obtaining tight fine-grained parameterized algorithms for dense parameters. Iwata and Yoshida show that for any $\varepsilon > 0$ Vertex Cover can be solved in time $\mathcal{O}^*((2-\varepsilon)^{\mathrm{tw}})$ if and only if Vertex Cover can be solved in time $\mathcal{O}^*((2-\varepsilon)^{\mathrm{cw}})$ [29]; as the bases differ for treewidth and clique-width in our case, it seems difficult to transfer their techniques to our setting. Lampis [34] obtains the tight running time of $\mathcal{O}^*((2^q-2)^{\mathrm{cw}})$ for q-Coloring and a tight result for q-Coloring parameterized by a more restrictive variant of modular-treewidth. Generalizing to homomorphism problems, Ganian et al. [21] obtain tight results for parameterization by clique-width, where the obtained base depends on a special measure of the target graph. Katsikarelis et al. [32] obtain tight results for (k, r)-Center parameterized by clique-width and, in particular, the tight running time $\mathcal{O}^*(4^{\mathrm{cw}})$ for Dominating Set. Jacob et al. [30] and Hegerfeld and Kratsch [24] show that the running time $\mathcal{O}^*(4^{\mathrm{cw}})$ is tight for Odd Cycle Transversal, where the latter article also considers a generalization to more colors and contains tight results for parameters that do not fall into the class of width-parameters.

Organization. We discuss the preliminaries in Section 2. Section 3 covers the algorithms (Theorem 1.1) and Section 4 the lower bounds (Theorem 1.2); both sections first outline the used techniques and present more details for Connected Vertex Cover. For space reasons, we only give a few remarks regarding Connected Dominating Set. We conclude in Section 5. Proofs and sections that are delegated to the full version [25] are denoted by $\star$.

2 Preliminaries

For two integers a, b we write $a \equiv_c b$ to indicate equality modulo $c \in \mathbb{N}$. We use Iverson's bracket notation: for a boolean predicate p, we have that $[p]$ is 1 if p is true and 0 otherwise. For a function f we denote by $f[v \mapsto \alpha]$ the function $(f \setminus \{(v, f(v))\}) \cup \{(v, \alpha)\}$, viewing f as a set; we also write $f[v \mapsto \alpha, w \mapsto \beta]$ instead of $(f[v \mapsto \alpha])[w \mapsto \beta]$. By $\mathbb{Z}_2$ we denote the field of two elements. For $n_1, n_2 \in \mathbb{Z}$, we write $[n_1, n_2] = \{x \in \mathbb{Z} : n_1 \leq x \leq n_2\}$ and $[n_2] = [1, n_2]$. For a function $f \colon V \to \mathbb{Z}$ and a subset $W \subseteq V$, we write $f(W) = \sum_{v \in W} f(v)$. For functions $g \colon A \to B$, where $B \not\subseteq \mathbb{Z}$, and $A' \subseteq A$, we still denote the *image of A' under g* by $g(A') = \{g(v) : v \in A'\}$. If $f \colon A \to B$ is a function and $A' \subseteq A$, then $f\big|_{A'}$ denotes the *restriction* of f to A' and for a subset $B' \subseteq B$, we denote the *preimage of B' under f* by $f^{-1}(B') = \{a \in A : f(a) \in B'\}$. An ordered tuple of sets $(A_1, \ldots, A_\ell)$ is an *ordered subpartition* if $A_i \cap A_j = \emptyset$ for all $i \neq j \in [\ell]$. The *power set* of a set A is denoted by $\mathcal{P}(A)$.

Graph Notation. We use common graph-theoretic notation and the essentials of parameterized complexity. Let $G = (V, E)$ be an undirected graph. For a vertex set $X \subseteq V$, we denote by $G[X]$ the subgraph of G that is induced by X. The *open neighborhood* of a vertex v is given by $N_G(v) = \{u \in V : \{u, v\} \in E\}$, whereas the *closed neighborhood* is given by $N_G[v] = N_G(v) \cup \{v\}$. For sets $X \subseteq V$ we define $N_G[X] = \bigcup_{v \in X} N_G[v]$ and $N_G(X) = N_G[X] \setminus X$. For two disjoint vertex subsets $A, B \subseteq V$, we define $E_G(A, B) = \{\{a, b\} \in E(G) : a \in A, b \in B\}$

and adding a *join* between A and B means adding an edge between every vertex in A and every vertex in B. For a vertex set $X \subseteq V$, we define $\delta_G(X) = E_G(X, V \setminus X)$ and we write $\delta_G(v) = \delta_G(\{v\})$ for single vertices v. We denote the *number of connected components* of G by $\mathtt{cc}(G)$. A *cut* of G is a partition $V = V_L \cup V_R$, $V_L \cap V_R = \emptyset$, of its vertices into two parts.

Clique-Expressions and Clique-Width. A *labeled graph* is a graph $G = (V, E)$ together with a *label function* $\mathtt{lab}\colon V \to \mathbb{N} = \{1, 2, 3, \ldots\}$; we usually omit mentioning $\mathtt{lab}$ explicitly. A labeled graph is *k-labeled* if $\mathtt{lab}(v) \leq k$ for all $v \in V$. We consider the following four operations on labeled graphs:

- the *introduce*-operation $\ell(v)$ constructs a single-vertex graph whose vertex v has label ℓ,
- the *union*-operation $G_1 \oplus G_2$ constructs the disjoint union of labeled graphs G_1 and G_2,
- the *relabel*-operation $\rho_{i \to j}(G)$ changes the label of all vertices in G with label i to label j,
- the *join*-operation $\eta_{i,j}(G)$, $i \neq j$, which adds an edge between every vertex in G with label i and every vertex in G with label j.

A valid expression that only consists of introduce-, union-, relabel-, and join-operations is called a *clique-expression*. The graph constructed by a clique-expression μ is denoted G_μ and the constructed label function is denoted $\mathtt{lab}_\mu\colon V(G_\mu) \to \mathbb{N}$. We associate to a clique-expression μ the syntax tree T_μ in the natural way and to each node $t \in V(T_\mu)$ the corresponding operation. For any node $t \in V(T_\mu)$ the subtree rooted at t induces a *subexpression* μ_t. When μ is fixed, we define $G_t = G_{\mu_t}$, $V_t = V(G_t)$, $E_t = E(G_t)$, and $\mathtt{lab}_t = \mathtt{lab}_{\mu_t}$ for any $v \in V(T_\mu)$. We write $V_t^\ell = \mathtt{lab}_t^{-1}(\ell)$ for the set of all vertices with label ℓ at node t and we write $L_t = \{\ell \in \mathbb{N} : V_t^\ell \neq \emptyset\}$ for the set of *nonempty labels at node t*.

A clique-expression μ is a *k-clique-expression* or just *k-expression* if G_t is k-labeled for all $t \in V(T_\mu)$. The *clique-width* of a graph G, denoted by $\mathrm{cw}(G)$, is the minimum k such that a k-expression μ with $G = G_\mu$ exists. A clique-expression μ is *linear* if in every union-operation the second graph consists only of a single vertex. Accordingly, we define the *linear-clique-width* of a graph G, denoted $\text{lin-cw}(G)$, by only considering linear expressions.

Strong Exponential-Time Hypothesis. The *Strong Exponential-Time Hypothesis* (SETH) [10, 28] concerns the complexity of q-Satisfiability, i.e., every clause contains at most q literals. We define $c_q = \inf\{\delta : q\text{-Satisfiability can be solved in time } \mathcal{O}(2^{\delta n})\}$ for all $q \geq 3$. The weaker *Exponential-Time Hypothesis* (ETH) of Impagliazzo and Paturi [27] posits that $c_3 > 0$ and the Strong Exponential-Time Hypothesis states that $\lim_{q \to \infty} c_q = 1$. Equivalently, for every $\delta < 1$, there is some q such that q-Satisfiability cannot be solved in time $\mathcal{O}(2^{\delta n})$. For our lower bounds, the following weaker variant of SETH is sufficient.

▶ **Conjecture 2.1** (CNF-SETH). *For every $\varepsilon > 0$, there is no algorithm solving Satisfiability with n variables and m clauses in time $\mathcal{O}(\mathrm{poly}(m)(2 - \varepsilon)^n)$.*

Cut and Count. Let $G = (V, E)$ denote a connected graph. For easy reference, we repeat the key definition and lemmas of the cut-and-count-technique here.

▶ **Definition 2.2** ([16]). A cut (V_L, V_R) of an undirected graph $G = (V, E)$ is *consistent* if $u \in V_L$ and $v \in V_R$ implies $\{u, v\} \notin E$, i.e., $E_G(V_L, V_R) = \emptyset$. A *consistently cut subgraph* of G is a pair $(X, (X_L, X_R))$ such that $X \subseteq V$ and (X_L, X_R) is a consistent cut of $G[X]$. We denote the set of consistently cut subgraphs of G by $\mathcal{C}(G)$.

▶ **Lemma 2.3** ([16]). *Let X be a subset of vertices such that $v_* \in X \subseteq V$. The number of consistently cut subgraphs $(X, (X_L, X_R))$ such that $v_* \in X_L$ is equal to $2^{\mathtt{cc}(G[X]) - 1}$.*

▶ **Corollary 2.4** ($\star$). *Let $\mathcal{R} \subseteq \mathcal{P}(V)$ be a family of vertex sets so that every $X \in \mathcal{R}$ contains v_*. If the set $\mathcal{A} = \{(X, (X_L, X_R)) \in \mathcal{C}(G) : X \in \mathcal{R}, v_* \in X_L\}$ has odd cardinality, then there exists an $X \in \mathcal{R}$ such that $G[X]$ is connected.*

With the isolation lemma we avoid unwanted cancellations in the cut-and-count-technique.

▶ **Definition 2.5.** A function $\mathbf{w}\colon U \to \mathbb{Z}$ *isolates* a set family $\mathcal{F} \subseteq \mathcal{P}(U)$ if there is a unique $S' \in \mathcal{F}$ with $\mathbf{w}(S') = \min_{S \in \mathcal{F}} \mathbf{w}(S)$; recall that $\mathbf{w}(X) = \sum_{u \in X} \mathbf{w}(u)$ for subsets X of U.

▶ **Lemma 2.6** (Isolation Lemma, [36]). *Let $\mathcal{F} \subseteq \mathcal{P}(U)$ be a nonempty set family over a universe U. Let $N \in \mathbb{N}$ and for each $u \in U$ choose a weight $\mathbf{w}(u) \in [N]$ uniformly and independently at random. Then $\mathbb{P}[\mathbf{w}$ isolates $\mathcal{F}] \geq 1 - |U|/N$.*

3 Dynamic Programming Algorithms

Cut and Count. The cut-and-count-technique by Cygan et al. [16] allows us to reduce the global connectivity constraint to a locally checkable counting problem. Consistent cuts, cf. Definition 2.2, are the main driver of the cut-and-count-technique. Any graph $G = (V, E)$ admits exactly $2^{\mathtt{cc}(G)}$ distinct consistent cuts, where $\mathtt{cc}(G)$ is the number of connected components of G. By fixing a vertex v_* and only considering consistent cuts (V_L, V_R) with $v_* \in V_L$, this number reduces to $2^{\mathtt{cc}(G)-1}$, so that G admits an odd number of such consistent cuts if and only if G is connected. This behavior implies that if we count pairs $(X, (X_L, X_R))$, where $v_* \in X \subseteq V$ is a partial solution and (X_L, X_R) a consistent cut of $G[X]$ with $v_* \in X_L$, modulo two, that all disconnected solutions cancel. When multiple connected solutions exist, they could also cancel modulo two, but this issue can be avoided at the cost of randomization by using the isolation lemma, cf. Lemma 2.6. So, if there exists a connected solution, then we can assume that Lemma 2.3 applies and let the algorithm answer accordingly.

Lifting Vertex States to Label States. For dynamic programming along clique-expressions, we have to characterize the relevant interactions of a partial solution with the *labels* which govern which *joins* can be constructed by the expression. In the considered problems, a single vertex v can take a constant number of different states with respect to a partial solution which we capture with a problem-dependent set Ω; e.g., for Connected Vertex Cover, we have $\Omega = \{\mathbf{0}, \mathbf{1}_L, \mathbf{1}_R\}$, representing $v \notin X$ (state $\mathbf{0}$), $v \in X_L$ (state $\mathbf{1}_L$), and $v \in X_R$ (state $\mathbf{1}_R$), respectively. A clique-expression repeatedly adds *joins* between pairs of vertex sets, say A and B, i.e., all possible edges between A and B are added, and the algorithm must check whether a partial solution remains feasible after adding a join and possibly update some states. A priori, each choice of vertex states in a label could yield different behaviors for partial solutions. However, for the considered problems the precise multiplicity of a vertex state in A or in B is irrelevant for a join; it suffices to distinguish for each side which vertex states appear and which do not. Therefore, the label states are captured by the subsets of Ω. The next two techniques will allow us to reduce the number of considered states further.

Nice Clique-Expressions. We refine and augment standard clique-expressions to distinguish between *live* and *dead* labels. When performing dynamic programming along a clique-expression, we consider the induced subgraphs defined by subexpressions of the given clique-expression. At a subexpression, we say that a label ℓ is *live* if in the remaining expression the vertices with label ℓ receive further edges that are not present in the current subexpression, otherwise ℓ is *dead*. First, we observe that we do not need to track the states of a partial solution at the dead labels, as they only have trivial interactions with the other states in the remaining expression. Hence, we only need to consider the states that can be attained at live labels which allows us to reduce the number of considered states for Connected Vertex Cover. To simplify the algorithms and avoid handling of edge cases, we transform the clique-expressions so that no degenerate cases occur and add a dead-operation $\perp_\ell$ which handles label ℓ turning from live to dead. The dead-operation is similar to *forget vertex nodes* in *nice tree decompositions* [33]. Distinguishing live and dead labels has been used before [21, 32, 34] to obtain improved running times, but handling the label types explicitly via an additional operation is new to the best of the authors' knowledge.

Inclusion-Exclusion States. For Connected Dominating Set, we transform to a different set of vertex states, called *inclusion-exclusion states*, which have proven helpful for domination problems before [23, 39, 41, 43]. These states do not track whether a vertex is *undominated* or *dominated* by a partial solution, but *allow* a vertex to be dominated or *forbid* it. A solution to the original problem can usually be recovered by an inclusion-exclusion argument, however when lifting to label states this argument does not directly transfer. We show that the argument can be adapted for label states when working modulo two, whereas for vertex states the argument is known to also work for non-modular counting. The advantage of the inclusion-exclusion states is that at joins we do not have to update vertex states from undominated to dominated, thus simplifying the algorithm and also allowing us to collapse several label states into a single one. The dead-operations of nice clique-expressions serve as suitable timepoints to apply the adapted inclusion-exclusion argument.

Fast Convolutions. To quickly compute the recurrences for union-operations, we utilize algorithms for *fast subset convolution*. We tailor the techniques developed by Björklund et al. [6] on trimmed subset convolutions to obtain a fast algorithm for the union-recurrence appearing in the Connected Vertex Cover algorithm. For Connected Dominating Set, we employ the lattice-based results of Björklund et al. [7].

3.1 Nice Clique-Expressions

Let μ be a k-expression for $G = (V, E)$; the associated syntax tree is T_μ. We say that a clique-expression μ is *irredundant* if for any join-operation $\eta_{i,j}(G_{t'}) = t \in V(T_\mu)$, it holds that $E_{G_{t'}}(V^i_{t'}, V^j_{t'}) = \emptyset$, i.e., no edge added by the join existed before.

▶ **Theorem 3.1** ([11]). *Any k-expression μ can be transformed into an equivalent, i.e., $G_{\mu'} = G_\mu$, irredundant k-expression μ' in polynomial time.*

Irredundancy still allows several edge cases regarding empty labels to occur, which require special handling in the dynamic programming algorithms. To avoid this extra effort in the algorithms, we transform any clique-expression so that these edge cases do not occur.

▶ **Definition 3.2.** A clique-expression μ of G is *nice* if μ satisfies the following properties:

- μ is irredundant,
- for every join-node $\eta_{i,j}(G_{t'}) = t \in V(T_\mu)$, where t' is the child of t, we have that $G_t \neq G_{t'}$, i.e., t adds at least one edge and $V^i_{t'} \neq \emptyset$ and $V^j_{t'} \neq \emptyset$,
- for every relabel-node $\rho_{i\to j}(G_{t'}) = t \in V(T_\mu)$, where t' is the child of t, we have that $V^i_{t'} \neq \emptyset$ and $V^j_{t'} \neq \emptyset$.

In the full version of the paper, we give a short proof how to transform a k-expression into an equivalent nice k-expression. However, Ducoffe [17] has also shown how to perform such a transformation in only linear time with a more involved proof.

▶ **Lemma 3.3** ($\star$, [17]). *Any k-expression μ can be transformed into an equivalent, i.e., $G_{\mu'} = G_\mu$, nice k-expression μ' in polynomial time.*

For nice clique-expressions, we will now present how we augment the associated syntax tree to distinguish between live and dead labels. For the remainder of this section, we assume that G is a connected graph with at least two vertices. We begin with the following definition.

▶ **Definition 3.4.** Given a clique-expression μ for $G = (V, E)$ and a node $t \in V(T_\mu)$, the set of *dead vertices at t* is defined by $D_t = \{v \in V_t : \delta_G(v) \subseteq E_t\}$. A vertex $v \in V_t \setminus D_t$ is called *live at t*.

The next lemma shows that in an irredundant clique-expression either none or all vertices in a label are dead, thus allowing us to sensibly speak of dead and live labels.

▶ **Lemma 3.5** ($\star$). *Given an irredundant k-expression μ for $G = (V, E)$, a node $t \in V(T_\mu)$, and a nonempty label $\ell \in L_t$, we have that either $V_t^\ell \cap D_t = \emptyset$ or $V_t^\ell \subseteq D_t$.*

The following definition formalizes the handling of live and dead labels and the dead nodes that are added when a label turns from live to dead.

▶ **Definition 3.6.** Given an irredundant k-expression μ for $G = (V, E)$, the *augmented syntax tree* $\hat{T}_\mu$ of μ is obtained from T_μ by inserting up to two *dead nodes* directly above every join node $t = \eta_{i,j}(G_{t'})$, where t' is the child of t in T_μ, based on the following criteria:

- if $V_t^i \subseteq D_t \setminus D_{t'}$, then the node $\bot_i$ is inserted,
- if $V_t^j \subseteq D_t \setminus D_{t'}$, then the node $\bot_j$ is inserted,
- if both nodes $\bot_i$ and $\bot_j$ are inserted, then we insert them in any order.

We extend the notations G_t, V_t, D_t, V_t^ℓ, for $\ell \in [k]$, to dead nodes by inheriting the values of the child node. For every node $t \in V(\hat{T}_\mu)$, we inductively define the *live labels* $L_t^{live} \subseteq L_t$ by

$$\begin{array}{llll} L_t^{live} = \{\ell\} & \text{if } t = \ell(v), & L_t^{live} = L_{t'}^{live} \setminus \{i\} & \text{if } t = \rho_{i \to j}(G_{t'}), \\ L_t^{live} = L_{t'}^{live} & \text{if } t = \eta_{i,j}(G_{t'}), & L_t^{live} = L_{t'}^{live} \setminus \{\ell\} & \text{if } t = \bot_\ell(G_{t'}), \\ L_t^{live} = L_{t_1}^{live} \cup L_{t_2}^{live} & \text{if } t = G_{t_1} \oplus G_{t_2}. & & \end{array}$$

Dually, the set of *dead labels* $L_t^{dead} \subseteq L_t$ is given by $L_t^{dead} = L_t \setminus L_t^{live}$.

The next lemma shows that, up to pending dead nodes, L_t^{live} contains all nonempty labels that only consist of live vertices at t. Due to Lemma 3.5, no label of an irredundant k-expression can contain both live and dead vertices simultaneously.

▶ **Lemma 3.7** ($\star$). *Let μ be a nice k-expression of $G = (V, E)$ and $\hat{T}_\mu$ its augmented syntax tree. For any node $t \in V(\hat{T}_\mu)$ and $\ell \in L_t$, we have that $V_t^\ell \cap D_t = \emptyset$ implies $\ell \in L_t^{live}$. If t is not the child of a dead node, then we even have that $V_t^\ell \cap D_t = \emptyset$ if and only if $\ell \in L_t^{live}$.*

3.2 Connected Vertex Cover

CONNECTED VERTEX COVER

Input:	An undirected graph $G = (V, E)$, a cost function $\mathbf{c}\colon V \to \mathbb{N} \setminus \{0\}$ and an integer $\bar{b}$.
Question:	Is there a set $X \subseteq V$, $\mathbf{c}(X) \leq \bar{b}$, such that $G - X$ contains no edges and $G[X]$ is connected?

Let $(G = (V, E), \mathbf{c}, \bar{b})$ be a CONNECTED VERTEX COVER instance with $\mathbf{c}(v) \leq |V|^{\mathcal{O}(1)}$ for all $v \in V$. Furthermore, we assume that G is connected and contains at least two vertices.

Let μ be a nice k-expression for G. To apply the cut-and-count-technique, we first pick an edge in G, branch on one of its endpoints v_*, and in this branch only consider solutions containing v_*. Furthermore, we sample a weight function $\mathbf{w}\colon V \to [2|V|]$ for the isolation lemma, cf. Lemma 2.6. We perform bottom-up dynamic programming along the augmented syntax tree $\hat{T}_\mu$. At every node $t \in V(\hat{T}_\mu)$, we consider the following family of partial solutions

$$\mathcal{A}_t = \{(X, (X_L, X_R)) \in \mathcal{C}(G_t) : G_t - X \text{ contains no edges and } (v_* \in V_t \Rightarrow v_* \in X_L)\}.$$

In other words, $\mathcal{A}_t$ contains all consistently cut vertex covers of G_t such that v_* is on the left side of the cut if possible. For every $t \in V(\hat{T}_\mu)$, $\overline{c} \in [0, \mathbf{c}(V)]$, $\overline{w} \in [0, \mathbf{w}(V)]$, we define $\mathcal{A}_t^{\overline{c},\overline{w}} = \{(X, (X_L, X_R)) \in \mathcal{A}_t : \mathbf{c}(X) = \overline{c}, \mathbf{w}(X) = \overline{w}\}$. Let $\hat{r}$ denote the root node of the augmented syntax tree $\hat{T}_\mu$. By Corollary 2.4, there exists a connected vertex cover X of G with $\mathbf{c}(X) \leq \overline{b}$ if there exist $\overline{c} \in [0, \overline{b}]$ and $\overline{w} \in [0, \mathbf{w}(V)]$ such that $\mathcal{A}_{\hat{r}}^{\overline{c},\overline{w}}$ has odd cardinality.

We proceed by analyzing the behavior of a partial solution $(X, (X_L, X_R)) \in \mathcal{A}_t$ with respect to a label V_t^ℓ, $\ell \in L_t$. A vertex $v \in V_t^\ell$ can take one of the states $\Omega = \{\mathbf{0}, \mathbf{1}_L, \mathbf{1}_R\}$, meaning respectively $v \notin X$, or $v \in X_L$, or $v \in X_R$. To check the feasibility of $(X, (X_L, X_R))$, it suffices to store for each label which vertex states appear and which do not, as the constraints implied by $\mathcal{A}_t$ are "CSP-like", i.e., they apply to every edge, and they can be evaluated for every join by considering all pairs of involved vertex states. Hence, the power set $\mathcal{P}(\Omega)$ of Ω captures all possible label states.

The power set $\mathcal{P}(\Omega)$ a priori yields eight different states per label. However, we can exclude the state $\emptyset$ and the state $\Omega = \{\mathbf{0}, \mathbf{1}_L, \mathbf{1}_R\}$ from consideration. The former can be excluded, since we only need to store the state for nonempty labels, i.e., containing at least one vertex. The exclusion of the state $\Omega = \{\mathbf{0}, \mathbf{1}_L, \mathbf{1}_R\}$ is more subtle: any additional incident join would lead to an infeasible solution for this state, hence only dead labels, cf. Definition 3.6, may sensibly take this state. We return to this issue in a moment. Since it suffices to store the states of live labels, we set $\textbf{States} = \mathcal{P}(\Omega) \setminus \{\emptyset, \Omega\} = \{\{\mathbf{0}\}, \{\mathbf{1}_L\}, \{\mathbf{1}_R\}, \{\mathbf{0}, \mathbf{1}_L\}, \{\mathbf{0}, \mathbf{1}_R\}, \{\mathbf{1}_L, \mathbf{1}_R\}\}$.

Given a node $t \in V(\hat{T}_\mu)$, a *t-signature* is a function $f\colon L_t^{live} \to \textbf{States}$. For every node $t \in V(\hat{T}_\mu)$, $\overline{c} \in [0, \mathbf{c}(V)]$, $\overline{w} \in [0, \mathbf{w}(V)]$, and t-signature f, we define

$$\begin{aligned}\mathcal{A}_t^{\overline{c},\overline{w}}(f) = \{(X, (X_L, X_R)) \in \mathcal{A}_t^{\overline{c},\overline{w}} : \ & \mathbf{0} \in f(\ell) \Leftrightarrow V_t^\ell \not\subseteq X \text{ for all } \ell \in L_t^{live}, \\ & \mathbf{1}_L \in f(\ell) \Leftrightarrow X_L \cap V_t^\ell \neq \emptyset \text{ for all } \ell \in L_t^{live}, \\ & \mathbf{1}_R \in f(\ell) \Leftrightarrow X_R \cap V_t^\ell \neq \emptyset \text{ for all } \ell \in L_t^{live}\}.\end{aligned}$$

We claim that excluding the state Ω does not cause issues. Consider some node t whose parent is not a dead node, and $(X, (X_L, X_R)) \in \mathcal{A}_t^{\overline{c},\overline{w}}$ such that there is a live label $\ell \in L_t^{live}$ for which the three cases $V_t^\ell \not\subseteq X$, $X_L \cap V_t^\ell \neq \emptyset$, and $X_R \cap V_t^\ell \neq \emptyset$ simultaneously occur. Since ℓ is live, there is some $v \in N_G(V_t^\ell) \setminus N_{G_t}(V_t^\ell)$ by Lemma 3.7. We claim that $(X, (X_L, X_R))$ cannot be extended to a consistently cut vertex cover $(X', (X'_L, X'_R))$ of $G' = G[V_t \cup \{v\}]$ (hence also not of G). If $v \notin X'$, then there is an uncovered edge in G' between V_t^ℓ and v. If $v \in X'$, then some edge in G' crosses the cut (X'_L, X'_R) and so the cut cannot be consistent. Hence, partial solutions attaining the state $\{\mathbf{0}, \mathbf{1}_L, \mathbf{1}_R\}$ at a live label can be discarded.

Instead of computing the sets $\mathcal{A}_t^{\overline{c},\overline{w}}(f)$ directly, we only compute the quantities $A_t^{\overline{c},\overline{w}}(f) = |\mathcal{A}_t^{\overline{c},\overline{w}}(f)| \mod 2$. The recurrences for computing $A_t^{\overline{c},\overline{w}}(f)$, for every $t \in V(\hat{T}_\mu)$, t-signature f, $\overline{c} \in [0, \mathbf{c}(V)]$, $\overline{w} \in [0, \mathbf{w}(V)]$ depend on the type of the considered node t:

Introduce node. If $t = \ell(v)$ for some $\ell \in [k]$, then $L_t^{live} = \{\ell\}$ and

$$\begin{aligned}A_t^{\overline{c},\overline{w}}(f) = {} & [v \neq v_* \vee f(\ell) = \{\mathbf{1}_L\}] \\ & \cdot [(f(\ell) = \{\mathbf{0}\} \wedge \overline{c} = \overline{w} = 0) \vee (f(\ell) \in \{\{\mathbf{1}_L\}, \{\mathbf{1}_R\}\} \wedge \overline{c} = \mathbf{c}(v) \wedge \overline{w} = \mathbf{w}(v))],\end{aligned}$$

since in a singleton graph any choice of singleton state leads to a valid solution, but if $v = v_*$ then only the solution with v_* on the left side of the cut is allowed.

Relabel node. If $t = \rho_{i \to j}(G_{t'})$, where t' is the child of t, for some $i, j \in [k]$, then by niceness of μ it follows that $i \in L_{t'}$, $j \in L_{t'}$, $L_t = L_{t'} \setminus \{i\}$ and either $i, j \in L_{t'}^{live}$ or $i, j \in L_{t'}^{dead}$.

- If labels i and j are live at t', then label j is live at t and the recurrence is given by

$$A_t^{\overline{c},\overline{w}}(f) = \sum_{\substack{\mathbf{S}_1, \mathbf{S}_2 \in \mathbf{States}:\\ \mathbf{S}_1 \cup \mathbf{S}_2 = f(j)}} A_{t'}^{\overline{c},\overline{w}}(f[i \mapsto \mathbf{S}_1, j \mapsto \mathbf{S}_2]),$$

  since $V_t^j = V_{t'}^i \cup V_{t'}^j$ and we simply have to iterate over all possible combinations of previous states at labels i and j that yield the desired state $f(j)$.
- If labels i and j are dead at t', then label j is dead at t and since we do not track the state of dead labels, we can simply copy the previous table, i.e., $A_t^{\overline{c},\overline{w}}(f) = A_{t'}^{\overline{c},\overline{w}}(f)$.

Join node. To check whether two states can lead to a feasible solution after adding a join between their labels, we introduce a helper function feas: $\mathbf{States} \times \mathbf{States} \to \{0, 1\}$ defined by $\text{feas}(\mathbf{S}_1, \mathbf{S}_2) = [\mathbf{0} \notin \mathbf{S}_1 \vee \mathbf{0} \notin \mathbf{S}_2][\mathbf{1}_L \in \mathbf{S}_1 \Rightarrow \mathbf{1}_R \notin \mathbf{S}_2][\mathbf{1}_R \in \mathbf{S}_1 \Rightarrow \mathbf{1}_L \notin \mathbf{S}_2]$. There are two reasons for infeasibility: a join edge is not covered, i.e., $\mathbf{0}$ appears on both sides, or a join edge connects both sides of the cut, i.e., $\mathbf{1}_L$ appears on one side and $\mathbf{1}_R$ on the other.

We have $t = \eta_{i,j}(G_{t'})$ for some $i \neq j \in L_{t'}$, where t' is the child of t. We have $i, j \in L_{t'}^{live}$ and if vertices turn dead at t, i.e., $D_{t'} \subsetneq D_t$, then a future dead node will handle it. Hence, we simply filter out all partial solutions that become infeasible due to the new join:

$$A_t^{\overline{c},\overline{w}}(f) = \text{feas}(f(i), f(j)) A_{t'}^{\overline{c},\overline{w}}(f).$$

Dead node. We have that $t = \bot_\ell(G_{t'})$, where t' is the child of t, $\ell \notin L_t^{live}$, and $L_t^{live} = L_{t'}^{live} \setminus \{\ell\}$. Since the only change is that t-signatures do not track the state of label ℓ anymore, we add up the contributions of all previous states of label ℓ. Hence, the recurrence is given by

$$A_t^{\overline{c},\overline{w}}(f) = \sum_{\mathbf{S} \in \mathbf{States}} A_{t'}^{\overline{c},\overline{w}}(f[\ell \mapsto \mathbf{S}]).$$

Although this recurrence looks simple, its correctness proof is nontrivial as it relies on the previous argument why label state Ω can be excluded.

Union node. We omit the standard, but somewhat technical, description of the union-recurrence here. After handling labels that are nonempty at only one of the children, a trimmed subset convolution remains that we can solve in time $\mathcal{O}^*(6^{|L_t^{live}|})$ for all $\overline{c}$, $\overline{w}$, and f simultaneously via the convolution algorithms developed in the appendix of the full version.

▶ **Lemma 3.8** ($\star$). *Given a nice k-expression μ of $G = (V, E)$, the quantities $A_t^{\overline{c},\overline{w}}(f)$ for all nodes $t \in V(\hat{T}_\mu)$, t-signatures f, and appropriate $\overline{c}$, $\overline{w}$, can be computed in time $\mathcal{O}^*(6^k)$.*

Proof sketch. For introduce nodes, relabel nodes, or join nodes, the recurrence for $A_t^{\overline{c},\overline{w}}(f)$ can be computed in polynomial time, as additions and multiplications in $\mathbb{Z}_2$ take constant time. For union nodes t, we compute the recurrence for all f, $\overline{c}$, $\overline{w}$ simultaneously in time $\mathcal{O}^*(6^{|L_t^{live}|})$ as discussed. As μ is a k-expression, we have $|L_t^{live}| \leq k$ for all $t \in V(\hat{T}_\mu)$ and in particular at most 6^k t-signatures for any node $t \in V(T_\mu)$. Hence, the running time follows.

The proof of correctness for introduce nodes, relabel nodes, join nodes, and union nodes is straightforward. For dead nodes, the proof of correctness follows from the discussion on the exclusion of state $\{\mathbf{0}, \mathbf{1}_L, \mathbf{1}_R\}$ and the construction of the augmented syntax tree $\hat{T}_\mu$. ◀

▶ **Theorem 3.9** ($\star$)**.** *There is a randomized algorithm that given a nice k-expression μ for a graph $G = (V, E)$ can solve* Connected Vertex Cover *in time $\mathcal{O}^*(6^k)$. The algorithm does not return false positives and returns false negatives with probability at most $1/2$.*

Proof. We sample a weight function $\mathbf{w}\colon V \to [2n]$ uniformly at random. Then, we pick an edge in G and branch on its endpoints; the chosen endpoint takes the role of v_* in the current branch. Using Lemma 3.8, we compute the quantities $A_t^{\overline{c},\overline{w}}(f)$. At the root node $\hat{r}$, we have that $L_{\hat{r}}^{live} = \emptyset$. The algorithm returns true if in one of the branches there is some choice of $\overline{c} \le \overline{b}$, $\overline{w} \in [0, 2n^2]$, such that $A_{\hat{r}}^{\overline{c},\overline{w}}(\emptyset) \neq 0$, otherwise the algorithm returns false.

The running time directly follows from Lemma 3.8. The correctness and error probability follow from Corollary 2.4, Lemma 2.3 and the isolation lemma, Lemma 2.6. Since these arguments are standard for the cut-and-count-technique, we omit them here. ◀

3.3 Remarks on Connected Dominating Set Algorithm ($\star$)

Connected Dominating Set

Input:	An undirected graph $G = (V, E)$, a cost function $\mathbf{c}\colon V \to \mathbb{N} \setminus \{0\}$ and an integer $\overline{b}$.
Question:	Is there a set $X \subseteq V$, $\mathbf{c}(X) \le \overline{b}$, such that $N(X) \cup X = V$ and $G[X]$ is connected?

To obtain our algorithm for Connected Dominating Set, we transform to the inclusion-exclusion states and apply the cut-and-count-technique. We again have the vertex states $\mathbf{1}_L$ and $\mathbf{1}_R$, but the state $\mathbf{0}$ splits into the *allowed* ($\mathbf{A}$) and *forbidden* state ($\mathbf{F}$), which denote that a vertex is allowed or forbidden to be dominated. Lifting to label states, we see that the presence of allowed vertices is irrelevant, as they impose no constraint on future joins, and that all label states containing at least two distinct non-allowed states behave in the same way. This allows us to collapse the number of considered label states down to five.

To count solutions dominating a vertex v with the inclusion-exclusion states, one usually subtracts the number of solutions where v is forbidden from the solutions where v is allowed. This argument fails when applied to labels, i.e., *groups* of vertices. Instead, our dynamic program counts solutions with a label containing u undominated vertices exactly 2^u times, so that all solutions with $u > 0$ cancel modulo two. Whenever a label turns dead, we apply this argument to ensure that all vertices in dead labels are dominated.

4 Lower Bounds

Construction Principle. The high-level construction principle of the lower bounds follows the style of Lokshtanov et al. [35]. That means the resulting graphs can be interpreted as a *matrix of blocks*, where each block spans several rows and columns. Every row is a long *path-like gadget* that simulates a constant number of variables of the Satisfiability instance and which contributes 1 unit of clique-width. The number of simulated variables is tied to the running time that we want to rule out. For technical reasons, we consider bundles of rows simulating a *variable group* of appropriate size. Every column corresponds to a clause and consists of gadgets that *decode* the states on the path gadgets and check whether the resulting assignment satisfies the clause. As a consequence of the construction principle, the lower bounds already apply to *linear* clique-width.

Path Gadgets and State Transitions. Our main contribution is the design of the *path gadgets* that lie at the intersection of every row and column, whereas the *decoding gadgets* can be adapted from Cygan et al. [15]. To ensure that each row contributes one unit of clique-width, adjacent path gadgets in a row are connected by a join. Our goal is to design a path gadget admitting as many states as possible. An important issue is how the state of the path gadgets may transition along each row, as the reduction only works when the state transitions follow some *transition order*, i.e, state i can transition to state j only if $i \leq j$.

Determining the Transition Order. For sparse width-parameters such as pathwidth, determining an appropriate transition order is much simpler, as the number of possible states is very small, e.g., there are at most four vertex states for the considered benchmark problems. The possible set of states for clique-width is much larger and we must select a specific subset of states, as not all of them admit a transition order. Hence, we analyze the possible state transitions across a join, obtaining a *transition/compatibility* matrix showing which pairs of states can lead to a globally feasible solution and which cannot. After possibly reordering the rows and columns of the compatibility matrix, a transition order must induce a *triangular submatrix* with ones on the diagonal. From a largest possible such triangular submatrix of the compatibility matrix, we can then deduce an appropriate transition order which guides the design of the path gadget.

Anatomy of a Path Gadget. Our path gadgets consist of a *central clique* communicating with the decoding gadgets, and two *boundary* parts, i.e., the *left* and *right* part connecting to the previous and following join. In the central clique, each solution avoids exactly one vertex representing the state of the path gadget. To implement the transition order, the left and right part have to communicate appropriate states to the two adjacent path gadgets. By *pairing* states along the main diagonal of the triangular submatrix, we see which states must be communicated in each case. The central idea behind designing the left and right part is to isolate the constituent state properties of the boundary vertices, such as, whether they are contained in the partial solution or whether they are dominated. This simplifies the communication with the central clique and expedites implementing the transition order.

Root-Connectivity. To capture the connectivity constraint, we create a distinguished vertex $\hat{r}$ called the *root* and attach a vertex of degree 1 to ensure that every connected vertex cover or connected dominating set must contain $\hat{r}$. Given a vertex subset $X \subseteq V(G)$ with $\hat{r} \in X$, we say that a vertex $v \in X$ is *root-connected* in X if there is a $v, \hat{r}$-path in $G[X]$. We will just say *root-connected* if X is clear from the context. The graph $G[X]$ is connected if and only if all vertices of X are root-connected in X. For the state of a partial solution X, it is important to consider which vertices are root-connected in X and which are not.

4.1 Path Gadget for Connected Vertex Cover

This subsection is devoted to constructing and analyzing the path gadget used to prove that Connected Vertex Cover (with unit costs) cannot be solved in time $\mathcal{O}^*((6-\varepsilon)^{\text{lin-cw}(G)})$ for some $\varepsilon > 0$ unless the CNF-SETH fails. The remaining parts of the construction are standard and can be found in the full version. We build a path gadget admitting 6 distinct states which narrows down to a single label/join, so that each row contributes one unit of linear clique-width. Each single vertex v has one of 3 states with respect to a partial solution X: $v \notin X$ (state $\mathbf{0}$), $v \in X$ and v is root-connected (state $\mathbf{1}_1$) or not root-connected (state $\mathbf{1}_0$). The state of a label can be described as a subset of $\{\mathbf{0}, \mathbf{1}_0, \mathbf{1}_1\}$.

We proceed by studying the compatibility of theses label states across a join, but we will only give an informal description here. Essentially, we assume that the considered join is the final opportunity for two partial solutions $X_1, X_2 \subseteq V$ with $\hat{r} \in X_i$, $i \in [2]$, living on

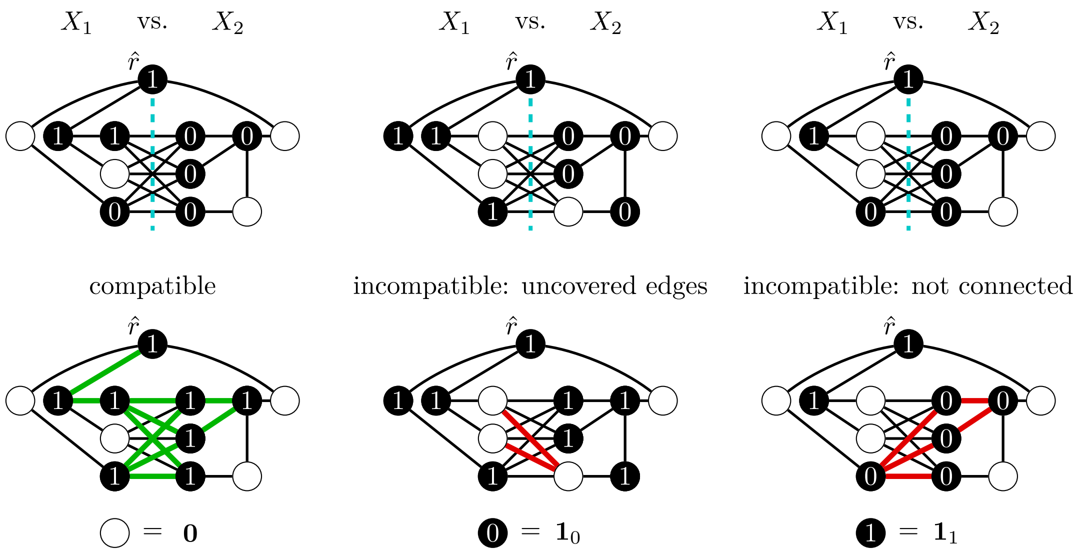

Figure 1 Several cases of partial solution compatibility across a join. The first row depicts the vertex states in X_1 and X_2, separated by the dashed line. The second row depicts the vertex states in $X_1 \cup X_2$ and highlights, from left to right, the induced edges, the uncovered edges, and a connected component not containing the root $\hat{r}$.

Table 2 A large triangular submatrix in the compatibility matrix of CONNECTED VERTEX COVER. The rows and columns have been reordered.

X_1 vs. X_2	$\{\mathbf{0}\}$	$\{\mathbf{1}_0, \mathbf{0}\}$	$\{\mathbf{1}_0\}$	$\{\mathbf{1}_1, \mathbf{0}\}$	$\{\mathbf{1}_1, \mathbf{1}_0\}$	$\{\mathbf{1}_1\}$
$\{\mathbf{1}_1\}$	1	1	1	1	1	1
$\{\mathbf{1}_1, \mathbf{1}_0\}$	0	1	1	1	1	1
$\{\mathbf{1}_1, \mathbf{0}\}$	0	0	1	0	1	1
$\{\mathbf{1}_0\}$	0	0	0	1	1	1
$\{\mathbf{1}_0, \mathbf{0}\}$	0	0	0	0	1	1
$\{\mathbf{0}\}$	0	0	0	0	0	1

separate sides of the join (with the exception of $\hat{r}$) to connect. Hence, the partial solutions X_1 and X_2 are considered to be *compatible* when in $X_1 \cup X_2$ every vertex incident to the considered join has state $\mathbf{0}$ or $\mathbf{1}_1$ and every edge of the join is covered by $X_1 \cup X_2$; see Figure 1. Since the interaction of X_i, $i \in [2]$, with the respective side of the join is captured by the aforementioned states, we obtain a compatibility matrix of size 7×7.

In this compatibility matrix, we find the triangular submatrix depicted in Table 2, after reordering rows and columns. Independent sets of size two are sufficient to generate the relevant label states and they are represented by the following ordered pairs of vertex states: $(\mathbf{0}, \mathbf{0})$, $(\mathbf{1}_0, \mathbf{0})$, $(\mathbf{1}_0, \mathbf{1}_0)$, $(\mathbf{1}_1, \mathbf{0})$, $(\mathbf{1}_1, \mathbf{1}_0)$, $(\mathbf{1}_1, \mathbf{1}_1)$. Pairing these states along the diagonal, we learn which states should be communicated to the left and right boundary in each case.

States. We define the three atomic states $\textbf{Atoms} = \{\mathbf{0}, \mathbf{1}_0, \mathbf{1}_1\}$, with their usual interpretation, and two predicates $\texttt{sol}, \texttt{conn} \colon \textbf{Atoms} \to \{0, 1\}$ by $\texttt{sol}(\mathbf{a}) = [\mathbf{a} \in \{\mathbf{1}_0, \mathbf{1}_1\}]$ and $\texttt{conn}(\mathbf{a}) = [\mathbf{a} = \mathbf{1}_1]$. We define six (gadget) states consisting of four atomic states each:

$$\mathrm{s}^1 = (\mathbf{0}\ , \mathbf{0}\ , \mathbf{1}_1, \mathbf{1}_1), \quad \mathrm{s}^2 = (\mathbf{1}_0, \mathbf{0}\ , \mathbf{1}_1, \mathbf{1}_0), \quad \mathrm{s}^3 = (\mathbf{1}_0, \mathbf{1}_0, \mathbf{1}_1, \mathbf{0}\ ),$$
$$\mathrm{s}^4 = (\mathbf{1}_1, \mathbf{0}\ , \mathbf{1}_0, \mathbf{1}_0), \quad \mathrm{s}^5 = (\mathbf{1}_1, \mathbf{1}_0, \mathbf{1}_0, \mathbf{0}\ ), \quad \mathrm{s}^6 = (\mathbf{1}_1, \mathbf{1}_1, \mathbf{0}\ , \mathbf{0}\ ).$$

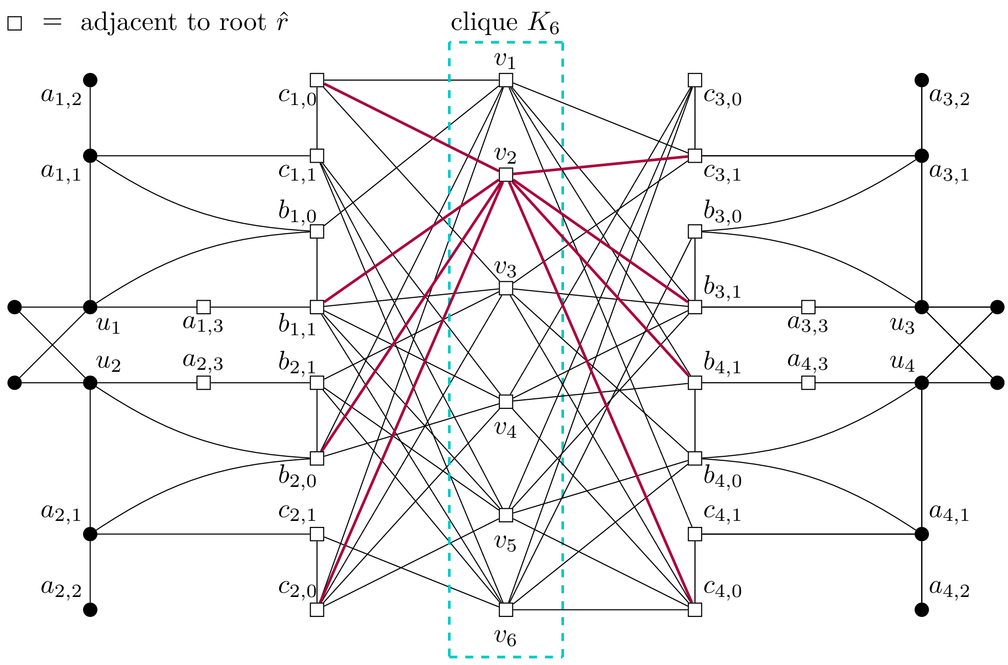

Figure 2 The path gadget P with the join vertices u_1, u_2 and u_3, u_4 joined to further vertices. All vertices depicted by a rectangle are adjacent to the root $\hat{r}$. The vertices inside the cyan dashed rectangle induce a clique. We have highlighted the edges incident to v_2 as an example.

The gadget states are numbered in the transition order. We collect the six gadget states in the set $\mathbf{States} = \{\mathbf{s}^1, \dots, \mathbf{s}^6\}$ and use the notation $\mathbf{s}_i^\ell \in \mathbf{Atoms}$, $i \in [4]$, $\ell \in [6]$, to refer to the i-th atomic component of state $\mathbf{s}^\ell$. Given a partial solution $Y \subseteq V(G)$, we associate to each vertex its state in Y with the map $\mathbf{state}_Y : V(G) \setminus \{\hat{r}\} \to \mathbf{Atoms}$, which is defined by $\mathbf{state}_Y(v) = \mathbf{0}$ if $v \notin Y$; $\mathbf{state}_Y(v) = \mathbf{1}_0$ if $v \in Y$ and v is not root-connected in $Y \cup \{\hat{r}\}$; $\mathbf{state}_Y(v) = \mathbf{1}_1$ if $v \in Y$ and v is root-connected in $Y \cup \{\hat{r}\}$.

Construction. The construction of the path gadget P is as follows. We create 4 *join* vertices $u_1, \dots, u_4$, 12 *auxiliary* vertices $a_{1,1}, a_{1,2}, a_{1,3}, a_{2,1}, \dots, a_{4,3}$, 8 *solution indicator* vertices $b_{1,0}, b_{1,1}, b_{2,0}, b_{2,1}, \dots, b_{4,1}$, 8 *connectivity indicator* vertices $c_{1,0}, c_{1,1}, c_{2,0}, c_{2,1}, \dots, c_{4,1}$ and 6 *clique* vertices $v_1, \dots, v_6$. We add edges so that the clique vertices v_ℓ, $\ell \in [6]$, induce a clique of size 6. Next, we explain how to connect the indicator vertices to the clique vertices. The clique vertex v_ℓ corresponds to choosing state $\mathbf{s}^\ell$ on the join vertices (u_1, u_2, u_3, u_4). The desired behavior of P is that a partial solution X of $P + \hat{r}$ contains $b_{i,1}$ if and only if X contains u_i and for the connectivity indicators, that X contains $c_{i,1}$ if and only if X contains u_i and u_i is root-connected in X. Accordingly, for all $i \in [4]$ and $\ell \in [6]$, we add the edges $\{v_\ell, b_{i,\mathtt{sol}(\mathbf{s}_i^\ell)}\}$ and $\{v_\ell, c_{i,\mathtt{conn}(\mathbf{s}_i^\ell)}\}$. For the remaining edges, we refer to Figure 2.

Behavior of a Single Path Gadget. We assume that G is a graph that contains $P + \hat{r}$ as an induced subgraph and that only the join vertices $u_i, i \in [4]$, and clique vertices $v_\ell, \ell \in [6]$, have neighbors outside this copy of $P + \hat{r}$. Furthermore, let X be a connected vertex cover of G with $\hat{r} \in X$; we abuse notation and write $X \cap P$ instead of $X \cap V(P)$.

We begin by showing a lower bound for $|X \cap P|$ via a vertex-disjoint packing of subgraphs.

▶ **Lemma 4.1** ($\star$). *We have that* $|X \cap P| \geq 21 = 4 \cdot 4 + 5$ *and more specifically* $a_{i,1} \in X$, $|X \cap \{u_i, a_{i,3}, b_{i,1}, b_{i,0}\}| \geq 2$, $|X \cap \{c_{i,0}, c_{i,1}\}| \geq 1$ *for all* $i \in [4]$ *and* $|X \cap \{v_1, \ldots, v_6\}| \geq 5$.

Using Lemma 4.1, we precisely analyze the solutions that match the lower bound of 21 on P and show that these have the desired state behavior. We define for any $Y \subseteq V(P)$ the 4-tuple $\mathbf{state}(Y) = (\mathbf{state}_Y(u_1), \mathbf{state}_Y(u_2), \mathbf{state}_Y(u_3), \mathbf{state}_Y(u_4))$ and show that the states communicated to the boundary depend on the state of the central clique as desired.

▶ **Lemma 4.2** ($\star$). *If* $|X \cap P| \leq 21$*, then* $|X \cap P| = 21$ *and* $a_{i,1} \in X$, $X \cap \{u_i, a_{i,3}, b_{i,1}, b_{i,0}\} \in \{\{u_i, b_{i,1}\}, \{a_{i,3}, b_{i,0}\}\}$, $|X \cap \{c_{i,0}, c_{i,1}\}| = 1$ *for all* $i \in [4]$ *and* $|X \cap \{v_1, \ldots, v_6\}| = 5$. *Furthermore, we have* $\mathbf{state}(X \cap P) = \mathbf{s}^\ell$ *for the unique integer* $\ell \in [6]$ *with* $v_\ell \notin X$.

Proof sketch. The first part follows by observing that the inequalities of Lemma 4.1 must be tight and that we must pick an antipodal pair in the cycle $\{u_i, a_{i,3}, b_{i,1}, b_{i,0}\}$. For the second part, we show that $\mathbf{state}_{X \cap P}(u_i) = \mathbf{s}_i^\ell$ for all $i \in [4]$. If $v_\ell \notin X$, then $b_{i,\mathtt{sol}(\mathbf{s}_i^\ell)}, c_{i,\mathtt{conn}(\mathbf{s}_i^\ell)} \in X$ by construction of P and because X is a vertex cover. Using the packing equations, we can propagate this information to u_i and obtain the desired state. ◀

Similarly, we also establish that for every state $\mathbf{s}^\ell \in \mathbf{States}$ a partial solution X_P^ℓ attaining $\mathbf{s}^\ell$ actually exists. This is made precise by the following Lemma 4.3, which also shows that for these partial solutions it is sufficient to establish root-connectivity for the join vertices.

▶ **Lemma 4.3.** *For every* $\ell \in [6]$*, there exists a vertex cover* X_P^ℓ *of* P *such that* $|X_P^\ell| = 21$, $X_P^\ell \cap \{v_1, \ldots, v_6\} = \{v_1, \ldots, v_6\} \setminus \{v_\ell\}$*, and* $\mathbf{state}(X_P^\ell) = \mathbf{s}^\ell$. *If* X *is a vertex cover of* G *with* $\hat{r} \in X$ *and* $X \cap P = X_P^\ell$ *and for every* $i \in [4]$ *either* $u_i \notin X$ *or* u_i *is root-connected in* X*, then every vertex of* X_P^ℓ *is root-connected in* X.

State Transitions. To study the state transitions, suppose that we have two copies P^1 and P^2 of P such that the vertices u_3 and u_4 in P^1 are joined to the vertices u_1 and u_2 in P^2. We denote the vertices of P^1 with a superscript 1 and the vertices of P^2 with a superscript 2, e.g., u_3^1 refers to the vertex u_3 of P^1. Again, suppose that P^1 and P^2 are embedded as induced subgraphs in a larger graph G with a root vertex $\hat{r}$ and that only the vertices $u_1^1, u_2^1, u_3^2, u_4^2$ and the clique vertices v_ℓ^1, v_ℓ^2, $\ell \in [6]$, have neighbors outside of $P^1 + P^2 + \hat{r}$.

Using the previous lemmas, we show that the state transitions respect the transition order and that it is also feasible for the state to remain stable.

▶ **Lemma 4.4** ($\star$). *Suppose that* $|X \cap P^1| \leq 21$ *and* $|X \cap P^2| \leq 21$*, then* $\mathbf{state}(X \cap P^1) = \mathbf{s}^{\ell_1}$ *and* $\mathbf{state}(X \cap P^2) = \mathbf{s}^{\ell_2}$ *with* $\ell_1 \leq \ell_2$. *Additionally, for each* $\ell \in [6]$*, the set* $X^\ell = X_{P^1}^\ell \cup X_{P^2}^\ell$ *is a vertex cover of* $P^1 + P^2$ *with* $\mathbf{state}_{X^\ell}(\{u_3^1, u_4^1, u_1^2, u_2^2\}) \subseteq \{\mathbf{0}, \mathbf{1}_1\}$.

Proof sketch. By Lemma 4.2, we have $\mathbf{state}(X \cap P^1) = \mathbf{s}^{\ell_1}$ and $\mathbf{state}(X \cap P^2) = \mathbf{s}^{\ell_2}$ for some $\ell_1, \ell_2 \in [6]$. Showing $\ell_1 \leq \ell_2$ corresponds to proving that all entries below the main diagonal of the chosen submatrix of the compatibility matrix are zero, i.e., that the submatrix is *triangular*. The second part corresponds to checking that all diagonal entries are ones. Both parts are proved by case checking. ◀

4.2 Remarks on Connected Dominating Set Lower Bound ($\star$)

The path gadget construction for Connected Dominating Set is very similar to Connected Vertex Cover as large parts of the gadget can be reused by subdividing edges. For Connected Dominating Set, we have to work with 4 vertex states instead of 3 due to tracking whether a vertex is dominated or not. Hence, we have to contend with, a priori,

$15 = 2^4 - 1$ possible label states instead of $7 = 2^3 - 1$. Surprisingly however, there are only five different behaviors for these label states, so that we obtain a smaller base, namely 5, for Connected Dominating Set compared to Connected Vertex Cover.

5 Conclusion and Open Problems

We have provided the first tight results under SETH for connectivity problems parameterized by clique-width, namely the problems Connected Vertex Cover and Connected Dominating Set. For several important benchmark problems such as Steiner Tree, Connected Odd Cycle Transversal, and Feedback Vertex Set, we are not able to achieve tight results with the current techniques. For Steiner Tree, our algorithmic techniques readily yield an $\mathcal{O}^*(4^{\mathrm{cw}})$-time algorithm, but the compatibility matrix for the lower bound only contains a triangular submatrix of size 3×3, hence we are not able to prove a larger lower bound than for treewidth. Similarly for Connected Odd Cycle Transversal, the techniques for Connected Vertex Cover yield an $\mathcal{O}^*(14^{\mathrm{cw}})$-time algorithm and a larger lower bound can be proven by adapting the gadgets for Connected Vertex Cover and adding a gadget to detect the used color at join-vertices, however there is again no large enough triangular submatrix that would allow us to show that $\mathcal{O}^*(14^{\mathrm{cw}})$ is optimal. For Feedback Vertex Set, a problem with a negative connectivity constraint in the form of acyclicity, the usual cut-and-count approach involves counting the edges induced by a partial solution, but this immediately leads to an XP-algorithm parameterized by clique-width as already noted by Bergougnoux and Kanté [4, 5]. Hence, a different approach is required to obtain plausible running times for tight results.

A big caveat in applying algorithms parameterized by clique-width is that we are lacking good algorithms for computing clique-expressions. The currently best algorithms rely on approximating clique-width via rankwidth, see Oum and Seymour [40] for the first such algorithm and Fomin and Korhonen [18] for the most recent one. However, the approximation via rankwidth introduces an exponential error, therefore all single-exponential algorithms parameterized by clique-width become double-exponential algorithms unless we are given a clique-expression by other means. A first step towards better approximation algorithms for clique-width could be a fixed-parameter tractable algorithm with subexponential error.

References

1 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 522–539. SIAM, 2021. doi:10.1137/1.9781611976465.32.

2 Benjamin Bergougnoux. *Matrix decompositions and algorithmic applications to (hyper)graphs*. PhD thesis, University of Clermont Auvergne, Clermont-Ferrand, France, 2019. URL: https://tel.archives-ouvertes.fr/tel-02388683.

3 Benjamin Bergougnoux, Jan Dreier, and Lars Jaffke. *A logic-based algorithmic meta-theorem for mim-width*, pages 3282–3304. SIAM, 2023. doi:10.1137/1.9781611977554.ch125.

4 Benjamin Bergougnoux and Mamadou Moustapha Kanté. Fast exact algorithms for some connectivity problems parameterized by clique-width. *Theor. Comput. Sci.*, 782:30–53, 2019. doi:10.1016/j.tcs.2019.02.030.

5 Benjamin Bergougnoux and Mamadou Moustapha Kanté. More applications of the d-neighbor equivalence: Acyclicity and connectivity constraints. *SIAM J. Discret. Math.*, 35(3):1881–1926, 2021. doi:10.1137/20M1350571.

6 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Trimmed moebius inversion and graphs of bounded degree. *Theory Comput. Syst.*, 47(3):637–654, 2010. doi:10.1007/s00224-009-9185-7.

7 Andreas Björklund, Thore Husfeldt, Petteri Kaski, Mikko Koivisto, Jesper Nederlof, and Pekka Parviainen. Fast zeta transforms for lattices with few irreducibles. *ACM Trans. Algorithms*, 12(1):4:1–4:19, 2016. doi:10.1145/2629429.

8 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

9 Narek Bojikian, Vera Chekan, Falko Hegerfeld, and Stefan Kratsch. Tight bounds for connectivity problems parameterized by cutwidth. In Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté, editors, *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany*, volume 254 of *LIPIcs*, pages 14:1–14:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.STACS.2023.14.

10 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*, pages 75–85. Springer, 2009. doi:10.1007/978-3-642-11269-0_6.

11 Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discret. Appl. Math.*, 101(1-3):77–114, 2000. doi:10.1016/S0166-218X(99)00184-5.

12 Radu Curticapean, Nathan Lindzey, and Jesper Nederlof. A tight lower bound for counting hamiltonian cycles via matrix rank. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1080–1099. SIAM, 2018. doi:10.1137/1.9781611975031.70.

13 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, 2018. doi:10.1145/3148227.

14 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.23.

15 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *CoRR*, abs/1103.0534, 2011. arXiv:1103.0534.

16 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Trans. Algorithms*, 18(2):17:1–17:31, 2022. doi:10.1145/3506707.

17 Guillaume Ducoffe. Maximum matching in almost linear time on graphs of bounded clique-width. *Algorithmica*, 84(11):3489–3520, 2022. doi:10.1007/s00453-022-00999-9.

18 Fedor V. Fomin and Tuukka Korhonen. Fast fpt-approximation of branchwidth. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 886–899. ACM, 2022. doi:10.1145/3519935.3519996.

19 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

20 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative families of product families. *ACM Trans. Algorithms*, 13(3):36:1–36:29, 2017. doi:10.1145/3039243.

21 Robert Ganian, Thekla Hamm, Viktoriia Korchemna, Karolina Okrasa, and Kirill Simonov. The fine-grained complexity of graph homomorphism parameterized by clique-width. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 66:1–66:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.66.

22 Carla Groenland, Isja Mannens, Jesper Nederlof, and Krisztina Szilágyi. Tight bounds for counting colorings and connected edge sets parameterized by cutwidth. In Petra Berenbrink and Benjamin Monmege, editors, *39th International Symposium on Theoretical Aspects of Computer Science, STACS 2022, March 15-18, 2022, Marseille, France (Virtual Conference)*, volume 219 of *LIPIcs*, pages 36:1–36:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.STACS.2022.36.

23 Falko Hegerfeld and Stefan Kratsch. Solving connectivity problems parameterized by treedepth in single-exponential time and polynomial space. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 29:1–29:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.29.

24 Falko Hegerfeld and Stefan Kratsch. Towards exact structural thresholds for parameterized complexity. In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation, IPEC 2022, September 7-9, 2022, Potsdam, Germany*, volume 249 of *LIPIcs*, pages 17:1–17:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.IPEC.2022.17.

25 Falko Hegerfeld and Stefan Kratsch. Tight algorithms for connectivity problems parameterized by clique-width. *CoRR*, abs/2302.03627, 2023. doi:10.48550/arXiv.2302.03627.

26 Falko Hegerfeld and Stefan Kratsch. Tight algorithms for connectivity problems parameterized by modular-treewidth. *CoRR*, abs/2302.14128, 2023. doi:10.48550/arXiv.2302.14128.

27 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

28 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

29 Yoichi Iwata and Yuichi Yoshida. On the equivalence among problems of bounded width. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 754–765. Springer, 2015. doi:10.1007/978-3-662-48350-3_63.

30 Hugo Jacob, Thomas Bellitto, Oscar Defrain, and Marcin Pilipczuk. Close relatives (of feedback vertex set), revisited. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8-10, 2021, Lisbon, Portugal*, volume 214 of *LIPIcs*, pages 21:1–21:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.21.

31 Bart M. P. Jansen and Jesper Nederlof. Computing the chromatic number using graph decompositions via matrix rank. *Theor. Comput. Sci.*, 795:520–539, 2019. doi:10.1016/j.tcs.2019.08.006.

32 Ioannis Katsikarelis, Michael Lampis, and Vangelis Th. Paschos. Structural parameters, tight bounds, and approximation for (k, r)-center. *Discrete Applied Mathematics*, 264:90–117, 2019. doi:10.1016/j.dam.2018.11.002.

33 Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. doi:10.1007/BFb0045375.

34 Michael Lampis. Finer tight bounds for coloring on clique-width. *SIAM J. Discret. Math.*, 34(3):1538–1558, 2020. doi:10.1137/19M1280326.

35 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.

36 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

37 Jesper Nederlof. Algorithms for np-hard problems via rank-related parameters of matrices. In Fedor V. Fomin, Stefan Kratsch, and Erik Jan van Leeuwen, editors, *Treewidth, Kernels, and Algorithms - Essays Dedicated to Hans L. Bodlaender on the Occasion of His 60th Birthday*, volume 12160 of *Lecture Notes in Computer Science*, pages 145–164. Springer, 2020. doi:10.1007/978-3-030-42071-0_11.

38 Jesper Nederlof, Michal Pilipczuk, Céline M. F. Swennenhuis, and Karol Wegrzycki. Hamiltonian cycle parameterized by treedepth in single exponential time and polynomial space. In Isolde Adler and Haiko Müller, editors, *Graph-Theoretic Concepts in Computer Science - 46th International Workshop, WG 2020, Leeds, UK, June 24-26, 2020, Revised Selected Papers*, volume 12301 of *Lecture Notes in Computer Science*, pages 27–39. Springer, 2020. doi:10.1007/978-3-030-60440-0_3.

39 Jesper Nederlof, Johan M. M. van Rooij, and Thomas C. van Dijk. Inclusion/exclusion meets measure and conquer. *Algorithmica*, 69(3):685–740, 2014. doi:10.1007/s00453-013-9759-2.

40 Sang-il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B*, 96(4):514–528, 2006. doi:10.1016/j.jctb.2005.10.006.

41 Michal Pilipczuk and Marcin Wrochna. On space efficiency of algorithms working on structural decompositions of graphs. *ACM Trans. Comput. Theory*, 9(4):18:1–18:36, 2018. doi:10.1145/3154856.

42 Willem J. A. Pino, Hans L. Bodlaender, and Johan M. M. van Rooij. Cut and count and representative sets on branch decompositions. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 27:1–27:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.27.

43 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.