Tight Bounds for Clique-Packing Parameterized by Clique-Width

Narek Bojikian
Humboldt-Universität zu Berlin, Germany

Stefan Kratsch
Humboldt-Universität zu Berlin, Germany

Abstract

In the d-Clique Packing problem, given a graph G and an integer t, we need to decide whether G contains a set of t pairwise vertex-disjoint cliques of size d each. This generalizes Triangle Packing and it is NP-complete for all $d \geq 3$. For each such d, we show how to solve the problem in $n^{\mathcal{O}(k^{d-1})}$ time where k is the clique-width of the graph (with a k-expression of G given in the input). We complement this by showing that, assuming the Exponential-Time Hypothesis (ETH), there is no algorithm that solves the problem in $n^{o(k^{d-1})}$ time for any fixed $d \geq 3$, already for the special case of seeking a partition into cliques of size d. Our proof also entails $W[1]$-hardness of d-Clique Packing (and d-Clique Partitioning) parameterized by clique-width for each $d \geq 3$. Our work continues a series of results on ETH-tight bounds for fundamental graph problems started by Fomin et al. (SICOMP 2010+2014) who obtained tight bounds for Max-Cut and Edge Dominating Set.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized complexity, triangle packing, clique packing, clique-width

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.115

Related Version *Full Version*: https://arxiv.org/abs/2606.31873 [8]

1 Introduction

Parameterized complexity studies the influence of instance properties such as solution size or structural restrictions on problem complexity. To this end, such properties are quantified as parameters and the main goal is to find parameterized algorithms that are (reasonably) fast when said parameters are small. By now, there is an extensive toolbox of algorithmic techniques as well as a variety of structural parameters for dealing with problems that are NP-hard in general. These result in FPT-algorithms (fixed-parameter tractable) with running time $f(k) \cdot n^{\mathcal{O}(1)}$ and in (slower) XP-algorithms with running time $n^{f(k)}$, where n is the input size, k is the parameter, and f is any computable function. This is complemented by a well-developed understanding of lower bounds, proving conditional optimality of the dependence on k subject to, e.g., ETH or SETH[1], or conditionally ruling out FPT- or even XP-algorithms, e.g., conditioned on $P \neq NP$ or $FPT \neq W[1]$.[2]

In this work, we are interested in (conditionally) tight complexity relative to the graph parameter *clique-width*. While clique-width is not as ubiquitous as treewidth, for which there is a wealth of conditionally tight upper and lower bounds, it can also be small for

[1] The Exponential-Time Hypothesis (ETH) posits that there is $c > 1$ such that no algorithm solves 3-SAT in $\mathcal{O}(c^n)$ time. The Strong Exponential-Time Hypothesis (SETH) posits that for each $c < 2$ there is $q \in \mathbb{N}$ such that no algorithm solves q-SAT in $\mathcal{O}(c^n)$ time.

[2] The class FPT contains all parameterized problems that have an FPT-algorithm relative to their parameter. The class $W[1]$ contains all problems reducible to Clique parameterized by solution size.

dense graphs.[3] Recently, a number of works have given tight bounds under SETH for many fundamental graph problems parameterized by clique-width, most taking the form that an $\alpha^k n^{\mathcal{O}(1)}$ time (FPT-)algorithm is known/shown and any $(\alpha-\varepsilon)^k n^{\mathcal{O}(1)}$ time algorithm would refute SETH; here α is problem-specific [24, 19, 6, 18, 21, 7, 9]. Earlier, starting with Fomin et al. [13, 14], tight upper and (conditional) lower bounds were obtained, usually with $n^{f(k)}$ upper bound and ruling out $n^{o(f(k))}$ subject to ETH [14, 1, 15, 2]. In particular, for Edge Dominating Set, Max Cut, and Hamiltonian Cycle there are algorithms running in $n^{\mathcal{O}(k)}$ time when given a k-expression of the graph, while $n^{o(k)}$ time is ruled out under ETH [14, 15, 2]. For Graph Coloring, however, there is a $n^{\mathcal{O}(2^k)}$ time algorithm by Kobler and Rotics [23] and Fomin et al. [15] ruled out $n^{2^{o(k)}}$ time under ETH.[4]

The unusual dependence of 2^k in the exponent of n for Graph Coloring compared to $\mathcal{O}(k)$ for the other problems naturally leads to the question of what other (conditionally) tight complexities are open. Presumably, there are artificial problems for any dependence on k, but are there also natural ones with e.g. a polynomial of k in the exponent? Moreover, outside of the many problems that are MSO expressible and thereby FPT due to a result of Courcelle [11] our knowledge about the (conditionally) optimality of XP-algorithms for problems parameterized by clique-width is still quite limited.

With this motivation, we study the d-Clique Packing problem where, given a graph G and an integer t, the task is to decide whether there exists a family of at least t vertex-disjoint cliques of size exactly d each in G. We also consider the special case of d-Clique Partitioning where we seek a partition into cliques of size d each. These are natural generalizations of Triangle Packing and Partition into Triangles and they are NP-complete for each $d \geq 3$. They are also natural stepping stones for considering more general (induced) subgraph packing problems.

Our results. We provide tight bounds for the complexity of d-Clique Packing and d-Clique Partitioning relative to clique-width for each value $d \geq 3$ (modulo ETH). Our reductions also directly show the $W[1]$-hardness of these problems parameterized by clique-width. Since d-Clique Packing clearly solvable in polynomial time for $d \leq 2$ (the case $d = 2$ is equivalent to the maximum matching), this results in a full classification of the parameterized complexity of these problems with respect to clique-width for each value of d. We divide our work into two parts mainly proving the following two theorems.

▶ **Theorem 1.** *For each positive integer $d \geq 3$, the d-Clique Packing problem can be solved in time $n^{\mathcal{O}(k^{d-1})}$ when a k-clique-expression of the input graph is given.*

In the latter part, we show that for each value $d \geq 3$ even the d-Clique Partitioning problem cannot be solved in time $n^{o(k^{d-1})}$ under ETH, even when the input graph is provided with a linear k-expression.

▶ **Theorem 2.** *For any integer $d \geq 3$, the d-Clique Partitioning problem is $W[1]$-hard, and cannot be solved in time $n^{o(k^{d-1})}$ assuming ETH, even when the input graph is given together with a linear k-expression.*

---

[3] If treewidth is bounded then so is clique-width, but clique-width may be exponential in the treewidth. Conversely, there are families of graphs of bounded clique-width but unbounded treewidth. For a graph to have treewidth at most k, it cannot have more than kn edges.

[4] This would leave room for, e.g., a $n^{1.5^k}$ time algorithm but double exponential dependence is necessary.

Related work. Fürer proposed two generalizations of clique-width, called fusion width [16] and multi-clique-width [17], that in particular avoid the exponential blow-up compared to treewidth. Chekan and Kratsch [10] showed that several tight bounds carry over from clique-width to both new variants.

Bergougnoux et al. [3] showed that assuming ETH there is no $2^{o(k^2)} \cdot n^{\mathcal{O}(1)}$ time algorithm for Independent Set parameterized by rank-width, an even more general parameter. It is known that rank-width is upper-bounded by clique-width, but Independent Set parameterized by clique-width can be solved in $2^k \cdot n^{\mathcal{O}(1)}$ time, which is optimal under SETH [20, 26].

Organization. In Section 2 we give a formal definition of clique-width along with a useful variant called *NLC-width*. Section 3 presents our algorithm and proves Theorem 1. In Section 4 we prove Theorem 2 by giving a reduction from Multi-Colored Clique. We conclude in Section 5.

2 Preliminaries

For a positive integer d, a *d-clique* in a graph G is a subset of vertices of G of size d that induces a complete subgraph. A *d-clique packing* in G is a family of vertex-disjoint d-cliques in G. We define the d-Clique Packing problem as follows:

Problem: d-Clique Packing
Input: A graph $G = (V, E)$ and an integer $t \in \mathbb{N}$.
Output: Is there a d-clique packing of size at least t in G?

The d-Clique Partitioning problem is defined as the special case with $|V| = td$.

A *reduction* f between two parameterized problems A and B is a mapping that maps each instance (I_A, k_A) of A to an instance (I_B, k_B) of B such that f is computable, and $(I_A, k_A) \in A$ if and only if $(I_B, k_B) \in B$. We say that f is a *parameterized reduction* if it can be computed in FPT time and it holds that the size of the output parameter is bounded by some computable function of the input parameter. It is well-known that the class $W[1]$ is closed under parameterized reductions. We refer to [12, Chapter 13] for details.

Clique Expression. A *labeled graph* is a graph $G = (V, E)$ together with a *labeling function* $\mathrm{lab}\colon V \to \mathbb{N}$. We usually omit the function lab and assume that it is implicitly given with G. We say that G is *k-labeled*, if it holds that $\mathrm{lab}(v) \leq k$ for all $v \in V$. A *labeled forest* is a labeled graph that is a forest. We define a *clique expression* μ as a well-formed expression defined by the following operations on labeled graphs:

- *Introduce vertex* $i(v)$ for $i \in \mathbb{N}^+$. This operation constructs a graph containing a single vertex v labeled i.
- The *relabel* operation $\rho_{i,j}(G)$ for $i, j \in \mathbb{N}^+$, $i \neq j$. This operation changes the labels of all vertices in G labeled i to the label j.
- The *join* operation $\eta_{i,j}(G)$ for $i, j \in \mathbb{N}^+, i \neq j$. The constructed graph results from G by adding all edges between the vertices labeled i and the vertices labeled j.
- The *union* operation $G_1 \oplus G_2$. The resulting graph is the disjoint union of G_1 and G_2.

We denote the graph resulting from a clique expression μ by G_μ, and the constructed labeling function by lab_μ. We associate with a clique expression μ a syntax tree $\mathcal{T}_\mu$ (we omit μ when clear from context) in the natural way, and associate with each node $x \in V(\mathcal{T})$ the corresponding operation. For $x \in V(\mathcal{T})$, the subtree $\mathcal{T}_x$ rooted at x induces a subexpression μ_x. We define $G_x = G_{\mu_x}$, $V_x = V(G_x)$, $E_x = E(G_x)$ and $\mathrm{lab}_x = \mathrm{lab}_{\mu_x}$.

We say that a clique expression μ is a *k-expression* if G_x is a k-labeled graph for all $x \in V(\mathcal{T})$. We define the clique-width of a graph G (denoted by $\mathrm{cw}(G)$) as the smallest value k such that there exists a k-expression μ with G_μ isomorphic to G. A *linear k-expression* μ is a k-expression, such that $\mathcal{T}_\mu$ is a caterpillar, i.e. each union node has at least one child that corresponds to an introduce vertex operation.

NLC-Width. A closely related parameter to clique-width is NLC-width, where an NLC-expression is defined solely by the two following operations: The *introduce operation* $i(v)$, resulting in a graph consisting of a single vertex v labeled i. The second operation is called the *join operation* $\oplus_\alpha^\beta$ for a set $\alpha \subseteq [k]^2$ (called *edge mapping*) and a *relabeling function* $\beta : [k] \to [k]$. Given two k-labeled graphs G_1 and G_2, $G_1 \oplus_\alpha^\beta G_2$ is the k-labeled graph obtained by performing the following steps in order:

1. First, we construct the graph G^1, the disjoin union of G_1 and G_2.
2. After that we add to G^1 all edges between vertices v_1 of G_1 with label i_1 and vertices v_2 of G_2 of label i_2 such that $(i_1, i_2) \in \alpha$ resulting in the graph G^2.
3. Finally, for each vertex v of G^2, we set the label of v to $\beta(\mathrm{lab}(v))$ resulting in the graph $G_1 \oplus_\alpha^\beta G_2$.

The *NLC-width* of a graph G (denoted $\mathrm{nlcw}(G)$) is the smallest value k such that there exists an k-NLC-expression generating a graph isomorphic to G. We accompany each NLC-expression with a syntax tree $\mathcal{T}$ in a natural way, where each leaf corresponds to an introduce operation and each internal node corresponds to a join operation. We denote the nodes of $\mathcal{T}$ by $\mathcal{V} = V(\mathcal{T})$. For $x \in \mathcal{V}$, we denote by $\mathcal{T}_x$ the subtree of $\mathcal{T}$ rooted at x, and by $G_x = (V_x, E_x)$ the graph obtained by the subexpression corresponding to the $\mathcal{T}_x$.

It is well-known [22, 4] that it holds for every graph G that

$$\mathrm{nlcw}(G) \leq \mathrm{cw}(G) \leq 2 \cdot \mathrm{nlcw}(G).$$

Moreover, this relationship is constructive and so provided a k-clique-expression of a graph G of width k, a k-NLC-expression of G can be computed in polynomial time.

Due to space constraints, we only present the main ideas of our algorithms and proofs in the main body of the paper, and defer the full details to the full version.

3 Upper Bound

We build our algorithm over NLC-expressions rather than clique-expressions, since the former allow for a simpler presentation. We can do this without loss of generality, since an NLC-expression of width at most k can be computed in polynomial time from a given k-clique-expression. Along this upper bound, let us fix a constant integer $d \geq 3$, and let $(G = (V, E), t)$ be the given instance of the d-clique packing problem, where G is given together with an NLC-expression of width k. Let $\mathcal{T}$ be the syntax tree of the given NLC-expression, and let $\mathcal{V} := V(\mathcal{T})$ be its nodes.

Intuitively, a partial solution at a node x is defined as a partition of the vertices in G_x into cliques of size *at most* d, where a solution is a partition at the root node with at least t cliques of size *exactly* d. We will define fingerprints corresponding to different partial solutions, that will allow us to count solutions in the claimed running time. Crucially, a clique can be represented by the set of its labels, where we have $\mathcal{O}(k^d)$ different such sets. A fingerprint of a partial solution then counts the number of times each such set of labels appears as a clique in this partial solution.

At first glance, this implies an algorithm with running time at least $n^{\Omega(k^d)}$ since this is the number of different fingerprints we keep track of. However, our main observation in the upper bound is that, while keeping the sets of labels in a fingerprint is essential to extend cliques correctly along the expression – as it allows us to combine two smaller cliques in a partial solution correctly – once a clique hits the target size d, we can "forget" its set of labels, and keep an aggregate counter for all such cliques independent of their labels. This reduces the running time to $n^{\mathcal{O}(k^{d-1})}$, since each finger print can now be defined as a mapping from $\mathcal{O}(k^{d-1})$ different sets of labels to some integer bounded by n, with a single added integer that represents the total count of d-cliques. In the following, we formalize this idea and present the algorithm. We start with some notation.

▶ **Definition 3.** *Given a vertex set S of a labeled graph G, we define the* type *of S, denoted* $\text{lab}^M(S)$*, as the multiset of labels of the vertices in S.*

Let $\mathcal{M}_d$ be the family of all multisets of size at most $d-1$ over the set $[k]$. We define the family of states $\mathcal{S} \coloneqq [n]_0^{\mathcal{M}_d}$ of all mappings that assign an integer in $[n]_0$ to each multiset in $\mathcal{M}_d$. We use this family to index the tables of our dynamic programming algorithm.

▶ **Definition 4.** *Given a labeled graph G, a* partial solution *in G is a partition of the vertices in the subgraph G_x into cliques of size at most d. Given a partial solution P, we define the* state *of this partition as the mapping $s^*(P) \in \mathcal{S}$ where for each $M \in \mathcal{M}_d$, $s^*(M)$ is the number of cliques C of P with $\text{lab}^M(C) = M$. We define the* weight *of the partition as the number of cliques in P of size exactly d.*

Now we define the tables A_x^t which indicate the existence of a partial solution of a specific state. We show that our dynamic programming routine computes these tables exactly.

▶ **Definition 5.** *We define the tables $A_x^t \in \{0,1\}^{\mathcal{S}}$ for each node $x \in \mathcal{V}$ and value $t \in [n]_0$, with $A_x^t[s] = 1$ if and only if there exists a partial solution in G_x of weight t and state s.*

The following observation holds since in a solution S, any vertex of G not covered by S can be covered by a clique of size one in a partition P.

▶ **Observation 6.** *The graph G admits a d-clique packing of size t if and only if there exists a state $s \in \mathcal{S}$ such that $A_r^t[s] = 1$.*

Now we aim to describe a dynamic programming routine that computes exactly the tables A_x^t for each node $x \in \mathcal{V}$ and each value $t \in [n]_0$. When handling a join node, it is more convenient to assign distinct labels to the vertices of the two unified graphs. This extends our definition of states. Let $\mathcal{M}_d^{2k}$ be the family of all multisets of size at most $d-1$ over the set $[2k]$. Let $\mathcal{S}^{2k}$ be the family of all extensions of the mappings in $\mathcal{S}$ to $\mathcal{M}_d^{2k}$.

Before we present the algorithm, we define the following operation that describes how a state of a partial solution changes after relabeling the vertices in an NLC-expression. We also use this operation to shift the labels in a state corresponding to the right child of a join node before combining it with the state corresponding to the left child, achieving the distinction of labels mentioned above.

▶ **Definition 7.** *Given a multiset $M \in \mathcal{M}^{2k}$ and two integers $i, j \in [2k]$, we define the multiset $M_{i\to j}$ obtained from M by replacing each occurrence of i in M with j, i.e.*

$$\#_\ell(M_{i\to j}) = \begin{cases} 0 & ;\ell = i, \\ \#_j(M) + \#_i(M) & ;\ell = j, \\ \#_\ell(M) & ;\textit{otherwise.} \end{cases}$$

Given a mapping $s \in \mathcal{S}^{2k}$, we define the mapping $s_{i\to j}$ as $s_{i\to j}(M) = \sum_{\substack{M' \in \mathcal{M}_d \\ (M')_{i\to j} = M}} s(M')$.

Finally, for some integer k', a relabeling function $\gamma\colon [k'] \to [k']$ and a multiset $M \in \mathcal{M}_d^{k'}$, we define the multiset $M_{\to\gamma} \in \mathcal{M}_d^{k'}$ by replacing each occurrence of a label ℓ in M with $\gamma(\ell)$. We define the mapping $s_{\to\gamma}$ as

$$s_{\to\gamma}(M) = \sum_{\substack{M' \in \mathcal{M}_d^{2k} \\ (M')_{\to\gamma} = M}} s(M').$$

For $s \in \mathcal{S}$, let $s{\downarrow}$ be the extension of s to $\mathcal{M}_d^{2k}$ with $s{\downarrow}(M) = 0$ for each $M \in \mathcal{M}_d^{2k} \setminus \mathcal{M}_d$. Let $s{\uparrow} = (s{\downarrow})_{\to\gamma}$ for the relabeling $\gamma\colon [2k] \to [2k]$ defined by $\gamma(i) = i + k$ for $i \in [k]$ and $\gamma(i) = i$ for $i \in \{k+1, \dots, 2k\}$. Intuitively, $s{\downarrow}$ is the extension of s that preserves the labels, whereas $s{\uparrow}$ is the extension of s that shifts all labels by k.

▶ **Definition 8.** *Given two mappings $s_1, s_2 \in \mathcal{S}$, we define the* shifted union *of s_1 and s_2, where for $M \in \mathcal{M}^{2k}$ it holds that*

$$(s_1 \uplus s_2)(M) = s_1{\downarrow}(M) + s_2{\uparrow}(M).$$

Note that $s_1 \uplus s_2(M) = 0$ whenever M contains an element in $[k]$ and an element in $[2k] \setminus [k]$. Intuitively, this operation combines two states by shifting the labels of the latter by k. This allows unified handling of join nodes in a step-by-step manner.

Finally, let $(X_1, Y_1) \dots (X_h, Y_h)$ be an enumeration of all sets in $\mathcal{M}^{2k} \times \mathcal{M}^{2k}$ in an arbitrary ordering. The following notion of feasibility indicates when two cliques in a partial solution can be merged by a join operation.

▶ **Definition 9.** *Given a node $x \in \mathcal{V}$ corresponding to an operation $\oplus_\alpha^\beta$, and an integer $i \in [h]$, we call the pair (X_i, Y_i)* feasible *at x, if it holds that $|X_i \cup Y_i| \leq d$ and for each label $\ell \in X_i$ and each label $\ell' \in Y_i$, it holds that $(\ell, \ell') \in \alpha$.*

▶ **Algorithm 1.** *We define the tables $T_x^\omega \in \{0,1\}^{\mathcal{S}}$ for each node $x \in \mathcal{V}$ and each value $\omega \in [n]_0$ recursively over $\mathcal{T}$ where for an introduce vertex node $\mu_x = i(v)$: we define $T_x^\omega[s] = 1$ for the mapping s with $s(\langle i \rangle) = 1$ and $s(M) = 0$ for all other multisets $M \in \mathcal{M}_d$, and $T_x^\omega[s'] = 0$ for all other mappings $s' \in \mathcal{S}$ different from s.*

For a join node x corresponding to the operation $\oplus_\alpha^\beta$, we start by defining the auxiliary tables $T_i^\omega \in \{0,1\}^{\mathcal{S}^{2k}}$ for each value $i \in [h]_0$ and each $\omega \in [n]_0$. First, we define the tables T_0^ω as follows:

$$T_0^\omega[s] = \bigvee_{\omega_1 + \omega_2 = \omega} \bigvee_{\substack{s_1, s_2 \in \mathcal{S} \\ s_1 \uplus s_2 = s}} T_{x_1}^{\omega_1}[s1] \wedge T_{x_2}^{\omega_2}[s_2],$$

where $T_0^\omega[s] = 1$ if there exists two values $\omega_1, \omega_2 \in [n]_0$ with $\omega_1 + \omega_2 = \omega$ and two states $s_1, s_2 \in \mathcal{S}$ with $s_1 \uplus s_2 = s$ such that $T_{x_1}^{\omega_1}[s_1] = 1$ and $T_{x_2}^{\omega_2}[s_2] = 1$, and $T_0^\omega[s] = 0$ otherwise.

For $i \in [h]$, we compute the tables T_i^ω for all $\omega \in [n]_0$ as follows: If (X_i, Y_i) is not feasible at x, then we set $T_i = T_{i-1}$. Otherwise, we start by setting $T_i^\omega[s] = 0$ for all values of ω and all states $s \in \mathcal{S}^{2k}$. Then for each state $s \in \mathcal{S}^{2k}$ let $r = \min\{s(X_i), s(Y_i)\}$. For each value $r' \in [r]_0$, let s' be the state resulting from s by subtracting r' from $s(X_i)$ and $s(Y_i)$, and adding r' to $s(X_i \cup Y_i)$ if $|X_i \cup Y_i| \leq d - 1$. Then we set $T_i^\omega[s'] = T_i^\omega[s'] \vee T_{i-1}^\omega[s']$ if $|X_i \cup Y_i| \leq d_1$, or we set $T_i^{\omega + r'}[s'] = T_i^\omega[s'] \vee T_{i-1}^\omega[s']$ otherwise.

Let $\gamma^R\colon [2k] \to [2k]$ *be the relabeling function defined by* $\gamma^R(i) = \gamma^R(i+k) = i$ *for each* $i \in [k]$*. we define the auxiliary table* $\hat{T}^\omega \in \{0,1\}^{\mathcal{S}^{2k}}$ *where for* $s \in \mathcal{S}^{2k}$ *we define*

$$\hat{T}^\omega[s] = \bigvee_{\substack{s' \in \mathcal{S}^{2k} \\ (s')_{\to \gamma^R} = s}} T_h^\omega[s'].$$

Let $\hat{T}_0^\omega$ *be the restriction of* $\hat{T}^\omega$ *to* $\mathcal{S}$ *with* $\hat{T}_0^\omega[s] = \hat{T}^\omega[s\downarrow]$ *for each* $s \in \mathcal{S}$*. We define*

$$T_x^\omega[s] = \bigvee_{\substack{s' \in \mathcal{S} \\ (s')_{\to \beta} = s}} \hat{T}_0^\omega[s'].$$

Now we prove the correctness of the algorithm, where we show by induction over $\mathcal{T}$, that $T_x^\omega = A_x^\omega$ for each node $x \in \mathcal{V}$ and each value $\omega \in [n]_0$. In order to show this, we define the tables $\hat{A}^\omega$, and $\hat{A}_i^\omega$ for each value i corresponding to the tables $\hat{T}^\omega$ and T_i^ω respectively, and show their equality, again by induction over i. We refer to full version for the full details of the proof. We get the following lemma.

▶ **Lemma 10.** *It holds for each node* $x \in \mathcal{V}$ *and each value* $\omega \in [n]_0$ *that* $T_x^\omega = A_x^\omega$*.*

The following corollary follows then immediately from Observation 6.

▶ **Corollary 11.** *The graph* G *admits a* d*-clique packing of size at least* t *if and only if there exists a state* $s \in \mathcal{S}$ *and an integer* $t' \geq t$ *such that* $T_r^{t'}[s] = 1$*.*

Now we bound the running time of the algorithm.

▶ **Lemma 12.** *All tables* T_x^ω *can be computed in time* $n^{\mathcal{O}(k^{d-1})}$*.*

Proof sketch. Let $N := |\mathcal{S}^{2k}|$. Since $\mathcal{M}_d^{2k}$ consists of all multisets of size at most $d-1$ over $[2k]$, it holds that $N = (n+1)^{|\mathcal{M}_d^{2k}|} = n^{\mathcal{O}(k^{d-1})}$. It also holds that $|\mathcal{S}| \leq N$. But then all tables described in the algorithms can be computed, in the same described order, in time polynomial in N by iterating over all relevant states and values of ω. Since the size of $\mathcal{T}$ is polynomial in n, this implies that all tables T_x^ω can be computed in time $n^{\mathcal{O}(k^{d-1})}$. ◀

Proof of Theorem 1. Given the instance (G,t), together with a k-clique-expression of G for some positive integer k, we first compute a k-NLC-expression of G. As discussed in the preliminaries, this can be done in polynomial time. Let $\mathcal{T}$ be the syntax tree of the computed NLC-expression, and let $\mathcal{V}$ be the set of nodes of $\mathcal{T}$. Our algorithm then computes all tables T_x^ω for each node $x \in \mathcal{V}$ and each value $\omega \in [n]_0$ using Algorithm 1. It outputs YES, if there exists some mapping $s \in \mathcal{S}$ with $T_r^t[s] = 1$, where r is the root of $\mathcal{T}$, and NO otherwise. The correctness of the algorithm follows from Corollary 11. The algorithm runs in time $n^{\mathcal{O}(k^{d-1})}$ by Lemma 12. ◀

4 Lower Bound

Let us fix some integer $d \geq 3$. In this section, we aim to prove that the d-Clique Partitioning problem is $W[1]$-hard when parameterized by the linear clique-width, and that it cannot be solved in time $n^{o(k^{d-1})}$ assuming ETH, even when a linear k-clique-expression is given as part of the input. We base our lower bound on a reduction from the Multi-Colored Clique problem defined as follows.

Problem: Multi-Colored Clique
Input: A graph G with a partition of its vertex set into k color classes.
Output: Is there a clique in G containing one vertex from each partition?

We also use a result from Lokshtanov et al. [25] that states the following.

▶ **Lemma 13** (Theorem 5.2.). *Assuming ETH, there is no $f(k)n^{o(k)}$ time algorithm for the* Multi-Colored Clique, even on instances with the same number of vertices in each color class and the same number of edges between each pair of color classes, where k is the number of colors and n is the number of vertices in each color class.

However, we will restrict to a special case of this problem, that we call d-Binomial Multi-Colored Clique for some positive integer d, where the number of colors is a binomial coefficient $\binom{k}{d}$ for some integer $k \in \mathbb{N}$. We show that the lower bound holds for this restricted problem as well.

For each positive integer d, let $\mathcal{H}_d$ be the family of all $\binom{k}{d}$-partite graphs with n vertices in each partition, and m edges between each pair of partitions for some arbitrary integers $n, m, k \in \mathbb{N}$. We show that this problem stays hard even when restricted to instances in $\mathcal{H}_d$ for any positive integer d. Formally, we define the restricted version as follows:

Problem: d-Binomial Multi-Colored Clique
Input: A graph $H \in \mathcal{H}_d$.
Output: Is there a clique in H containing one vertex in each partition?

▶ **Corollary 14.** *For every positive integer d, the d-*Binomial Multi-Colored Clique problem is $W[1]$-hard parameterized by k, and cannot be solved in time $n^{o\left(\binom{k}{d}\right)}$ assuming ETH, where $\binom{k}{d}$ is the number of partitions of the input graph.

In the rest of this section, we provide a reduction from the $(d-1)$-Binomial Multi-Colored Clique problem to the d-Clique Partitioning problem on graphs of bounded clique-width that proves the claimed lower bounds. Let H be an instance of the $(d-1)$-Binomial Multi-Colored Clique problems with $r := \binom{k}{d-1}$ partitions. Let n be the number of vertices in each partition, and m be the number of edges between each pair of partitions.

We define the family of *colors* $R := \binom{[k]}{d-1}$, and we call each subset $\{i_1, \ldots, i_{d-1}\} \in R$ a *color*. We assign a distinct color to each partition of H (arbitrarily), where we denote the partition with color $\{i_1, \ldots, i_{d-1}\}$ by $V_{\{i_1,\ldots,i_{d-1}\}}$. We also call each partition a *color class*. For each color $C \in R$, let $V_C = \{u_C^1, \ldots, u_C^n\}$ be an arbitrary but fixed enumeration of the vertices in the color class V_C.

Our goal is to build an instance $\hat{G}$ of the d-Clique Partitioning problem with bounded clique-width $\mathcal{O}(k)$ such that H has a multi-colored clique if and only if the constructed instance is a YES-Instance. In order to achieve this, we first define an auxiliary graph G with the latter property. Then we define the graph $\hat{G}$ of bounded clique-width from G by adding edges to it, and we show that G has a d-clique partition if and only if $\hat{G}$ has one.

The graph G consists of a collection of the so called gadgets – graphs of constant clique-width that serve different functionalities. We add different copies of each gadget representing different aspects of the input instance. Mainly, we have two kinds of gadgets, each consisting of "simpler" gadgets. First, we define *color gadgets* that represent the different partitions of the multicolored clique instance. We add r color gadgets, each corresponding to a color in R,

and it encodes the id of the vertex selected in the corresponding color class in a hypothetical multi-colored clique. A color gadget consists of a sequence of *id gadget*s and *cuts* between them. An id gadget allows the selection of a specific vertex in a partition by its id (in $[n]$), while cuts ensure that consecutive id gadgets select the same id.

Second, we define *edge gadget*s, which ensure that the selected ids of different partitions are adjacent. An edge gadget consists of m *simple edge gadget*s that represent the single edges between two color classes, and a *selection gadget* that "consumes" all but a single simple edge gadget that will be "consumed" by the id gadgets adjacent to it. Now we provide a formal description of these gadgets:

Color gadget. The construction consists of r *color gadgets*, each corresponding to a color in R. A color gadget is a graph that encodes the choice of a vertex in the corresponding color class. It has a path-like structure, as it consists of a sequence of gadgets called *id gadgets* connected by small independent sets. While an id gadget encodes the choice of a vertex in the corresponding color class, the independent sets between id gadgets ensure that the choice is copied correctly from one id gadget to the next. This gives the graph G a grid-like structure, where the rows correspond to colors and each column corresponds to a pair of colors, and ensures that the chosen vertices in these two color classes are adjacent in H.

More concretely, let $\mathscr{L} \coloneqq \binom{R}{2}$ be the set of pairs of colors, and $\ell \coloneqq |\mathscr{L}|$. We define a *color gadget* as the graph consisting of a sequence of 2ℓ copies of a gadget called a *color column*, where a color column consists of n $(d-1)$-cliques. We add an independent set of size n between each two consecutive color columns, and make its vertices adjacent to all vertices of both columns on its sides. We also add an independent set of size n adjacent to all vertices of both the first and the last color columns. Hence, each color gadget consists of 2ℓ color columns and 2ℓ independent sets in an alternating manner. We partition a color gadget into ℓ segments, each consisting of two consecutive color columns together with the independent set between them. We call each such segment an *id gadget*. We call each independent set inside an id gadget an *inner cut*, and each independent set between two id gadgets an *outer cut*.

Note that each element of $\mathscr{L}$ is an unordered pair of subsets of $[k]$ of size $d-1$ each. We fix an arbitrary enumeration of the elements $\mathscr{L}$ as $\mathscr{L} = \{L_1, \ldots, L_\ell\}$. We denote the q-th id gadget in the color gadget $\mathscr{C}_{\{i_1,\ldots i_d\}}$ by $\mathscr{I}^q_{\{i_1,\ldots,i_d\}}$. Hence, each color gadget forms a row in the grid, and for each integer $q \in [\ell]$, the family consisting of the q-th id gadget of each color gadget forms the q-th column in the grid. On a more fine-grained view, each row of the grid consists of n finer rows, we call them *unit rows*, where each unit row consists of a sequence of 2ℓ $(d-1)$-cliques, and n single vertices between them in an alternating manner, where each such single vertex belongs to an independent set of the color gadget.

Intuitively, the choice of whether such a vertex is attached to the clique preceding it or the clique succeeding it in the unit row encodes a Boolean value, where the accumulation of all these values over an id gadget encodes the id of the vertex chosen in the corresponding color class.

Edge gadget. In order to ensure that the chosen vertices in the color classes are adjacent, for each pair of colors $L_q = \{C_1, C_2\}$, we modify the q-th id gadget of the color gadgets $\mathscr{C}_{C_1}$ and $\mathscr{C}_{C_2}$ as follows. First we remove the inner cuts from both gadgets. Then we build the so called *edge gadgets* between these two id gadgets as follows: we start by defining a *simple edge gadget* that corresponds to a single edge $\{u^{j^1}_{C_1}, u^{j^2}_{C_2}\}$ in H. A simple edge gadget consists of an independent set of size $2n$ with vertices $v_1, \ldots v_n$ and $w_1, \ldots w_n$. The vertices $v_1, \ldots v_{j_1}$

are adjacent to all vertices of the first color column of $\mathscr{C}^q_{C_1}$, and the vertices $v_{j_1+1}, \dots v_n$ are adjacent to all vertices of the second color column of $\mathscr{C}^q_{C_1}$. Similarly, the vertices $w_1, \dots w_{j_2}$ are adjacent to all vertices of the first color column of $\mathscr{C}^q_{C_2}$, and the vertices $w_{j_2+1}, \dots w_n$ are adjacent to all vertices of the second color column of $\mathscr{C}^q_{C_2}$.

As mentioned before, an edge gadget corresponding to the pair of colors $L_q = \{C_1, C_2\}$ consists of the m simple edge gadgets $\mathscr{E}_1, \dots \mathscr{E}_m$ corresponding to the m edges between the two color classes, together with a $\mathcal{F}$-selection gadget added to them, where $\mathcal{F}$ is the family $\big\{V(\mathscr{E}_1), \dots, V(\mathscr{E}_m)\big\}$. A selection gadget allows to select a single simple edge gadget from the family by "consuming" all but one of them. Now we provide a formal description of the selection gadget.

Selection gadget. Given a family $\mathcal{F}$ of m sets of vertices $X_1, \dots X_m$, of size r each where r is even, such that the union of all these sets forms an independent set, an $\mathcal{F}$-selection gadget $\mathscr{S}(\mathcal{F})$ consists of $m-1$ copies of a structure called *r-selection column* $C_1, \dots C_{m-1}$, where C_i is added between X_i and X_{i+1}.

The selection column C_i is a simple cycle on $2r$ vertices. We partition the vertices of this cycle into 4 sets by assigning the colors $c_0, \dots c_3$ to them, where the color assigned to each vertex is defined by its id modulo 4. Let $v_1, \dots v_r$ be the vertices of X_i and $w_1, \dots w_r$ be the vertices of X_{i+1} we add all edges between the vertices v_i for odd values of i and the vertices colored c_1 and c_2, and all edges between the vertices v_i for even values of i and the vertices colored c_3 and c_0. Similarly, we add all edges between the vertices w_i for odd values of i and the vertices colored c_2 and c_3, and all edge between the vertices w_i for even values of i and the vertices colored c_1 and c_0.

The even cycle structure of the selection column allows to decompose it into two perfect matchings, a "horizontal" one, containing the edges between the vertices colored c_0 and c_1 and the edges between the vertices colored c_2 and c_3, and a "vertical" one, containing the rest of the edges. Intuitively, we connect the endpoints of the horizontal edges to vertices of X_i, and the endpoints of the vertical edges to vertices of X_{i+1}. We separate the vertices of X_i and X_{i+1} into even and odd indices to avoid conflicts, where we connect the even indexed vertices to the edges of even indices of either matching, and the odd indexed vertices to the edges of odd indices.

Finally, we add r cliques of size $d-3$ each, and make each of these cliques adjacent to all vertices of the selection column and all vertices of the sets in $\mathcal{F}$. This ensures that any triangle formed between $\mathcal{F}$ and the selection column can be extended to a clique of size d and vice versa.

This gadget is inspired by a gadget that was recently introduced by Bojikian et al. [5] for triangle packing parameterized by cutwidth. Intuitively, any solution $\mathcal{P}$ must pack the edges of exactly one of the two perfect matchings of a selection column in its cliques, as once one edge of either matching is packed in some d-clique, all the other edges must agree. Hence, $\mathcal{P}$ either includes the edges of the horizontal matching together with the vertices of X_i, or the edges of the vertical matching together with the vertices of X_{i+1}. The endpoints of the matching edges then, together with the vertices of X_i or X_{i+1}, form triangles that can be extended to cliques of size d by adding the $(d-3)$-cliques adjacent to the selection columns. Since we add $m-1$ selection columns, $\mathcal{P}$ must pack all but one simple edge gadget with the selection columns, keeping exactly one simple edge gadget free to be packed with the id gadgets adjacent to it. Note that connecting each edge of a selection column to a single vertex of X_i or X_{i+1} results in a "cleaner" construction. However, we add "redundant" edges (between these two sets and the selection column between them) to ensure that the resulting graph has bounded clique-width, since this allows to divide the vertices of the selection column into four equivalent classes as described above.

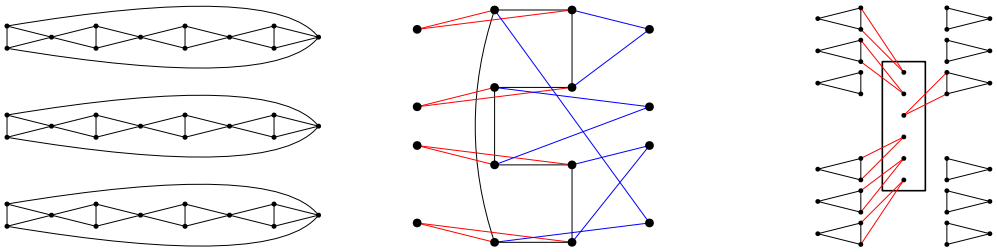

Figure 1 In (a) we depict three "unit rows" of a *color gadget* for $d = 3$. Each unit row consists of an alternation of 2-cliques and single vertices. In (b) we depict a part of a *selection gadget*. We depict two simple edge gadgets (the independent sets on the left and right) and a selection column between them. We omit the redundant edges for clarity, connecting each vertex of a simple edge gadget to the endpoints of a single edge. Note that the vertices of the first simple edge gadget build triangles with the edges of the horizontal matching of the selection column, while the vertices of the second simple edge gadget build triangles with the edges of the vertical matching. Finally, in (c) we depict a simple edge gadget replacing the inner cuts of two id gadgets. In this example, the simple edge gadget corresponds to the edge $\{u^2_{C_1}, u^3_{C_2}\}$ in H.

This concludes the description of a selection gadget, and thus the description of the graph G. Now we prove the correctness of our reduction.

▶ **Lemma 15.** *If H has a multi-colored clique, then G has a d-clique partition.*

Proof sketch. Given a multi-colored clique, let C be some color, and assume that the clique contains the ith vertex of C. Then in each id gadget of the color gadget of C, we pack each of the first i vertices of the inner cut together with the $(d-1)$-cliques of the left color column, and we pack the remaining vertices of the inner cut together with the $(d-1)$-cliques of the right color column. The outer cut vertices are packed in the reverse way. Now let $L_q = \{C_1, C_2\}$ be a pair of colors, and let $\{u^i_{C_1}, u^j_{C_2}\}$ be the edge of the multi-colored clique between these two colors. Then in the column of the grid corresponding to L_q, there must exist a simple edge gadget corresponding to that edge. We pack the vertices of all the other simple edge gadgets together with the selection gadget.

By the packing defined above, in the column corresponding to L_q, i $(d-1)$-cliques of the left color column are still "free", and $n - i$ $(d-1)$-cliques of the right color column are still free, whereas all the other vertices of this id gadget are already packed with outer cut vertices. Since these are exactly the vertices adjacent to the simple edge gadget corresponding to the edge $\{u^i_{C_1}, u^j_{C_2}\}$, we can pack the vertices of this simple edge gadget together with these free $(d-1)$-cliques. This covers all vertices of G in cliques of size d, and thus yields a d-clique partition of G. ◀

Now we aim to to prove the converse implication. From now on, we assume that G admits a d-clique partition. Let $\mathcal{P}$ be one such partition, and $E(\mathcal{P})$ be the set of all edges in the cliques of $\mathcal{P}$. We first prove some properties that every d-clique partition of G must have. These properties will be useful to show that the selected vertices correspond to a multi-colored clique in H.

In the full version we show that in an edge gadget, all but exactly one of the simple edge gadgets are packed with the vertices of the selection gadget internally, keeping a single simple edge gadget "exposed" to be packed together with the two id gadgets adjacent to this edge gadget.

▶ **Lemma 16.** *Let $\mathscr{S}(\mathcal{F})$ be a selection gadget in G with $\mathcal{F} = \{X_1, \dots X_m\}$, where all sets of $\mathcal{F}$ have size r. Then there exists a unique value $i \in [m]$ such that for all $j \neq i$, all vertices of X_j belong to cliques of $\mathcal{P}$ that lie completely in the selection gadget, while the vertices of X_i are in cliques that only intersect the selection gadget in X_i itself.*

Now we define the "state" of an id gadget defined by a solution. We will show that the outer cuts carry the state correctly between consecutive id gadgets.

▶ **Definition 17.** *Given an id gadget $\mathscr{I}$ in G, we define the* state *of $\mathscr{I}$ defined by $\mathcal{P}$, denoted by $s(\mathscr{I})$, as the number of $(d-1)$-cliques in the left color column of $\mathscr{I}$ that are contained in cliques of $\mathcal{P}$ together with vertices of the inner cut of this gadget if such an inner cut exists, or with vertices of the edge gadget replacing the inner cut otherwise.*

The following lemma states that all id gadgets of the same color gadget must have the same state. Intuitively, this holds since the $(d-1)$-cliques of a color column that are not packed with the vertices of the inner cut must be packed with the vertices of the outer cut adjacent to it.

▶ **Lemma 18.** *Any two id gadgets on the same color gadget have the same state.*

Finally, we show that the edge gadgets ensure that the selected states in different color gadgets correspond to adjacent vertices in H. Since we add an edge gadget for each pair of colors, this ensures that $\mathcal{P}$ corresponds to a solution of H.

▶ **Lemma 19.** *Let $\mathscr{I}^q_{C_1}$ and $\mathscr{I}^q_{C_2}$ be two id gadgets with the edge gadget corresponding to the pair of colors $L_q = \{C_1, C_2\}$ added between them. Let $s_1 = s(\mathscr{I}^q_{C_1})$ and $s_2 = s(\mathscr{I}^q_{C_2})$. Then the edge $\{u^{s_1}_{C_1}, u^{s_2}_{C_2}\}$ exists in H.*

Given all these properties, we can show the following:

▶ **Lemma 20.** *If G has a d-clique partition $\mathcal{P}$, then H has a multi-colored clique.*

Bounded clique-width graph. Now we describe the graph $\hat{G}$. For each id gadget $\mathscr{I}^q_C$ in G, and each $(d-1)$-clique of this gadget, let us fix a bijective assignment from C to the vertices of this clique. We call the value assigned to a vertex by this assignment its *color*. Let us denote the left color column of an id gadget $\mathscr{I}^q_C$ by $Q^q_C(L)$ and its right color column by $Q^q_C(R)$. For $X \in \{L, R\}$ and $q \in [\ell]$, we call the collection for all $(d-1)$-cliques of all color columns $Q^q_C(X)$ for all colors $C \in R$ the (q, X)-*grid column*. For each vertex of an inner cut, out cut or simple edge gadget attached to a color column $Q^q_C(X)$, we add all edges between this vertex and all vertices of the (q, X)-grid column whose colors belong to C. We denote the resulting graph by $\hat{G}$.

▶ **Lemma 21.** *The graph $\hat{G}$ has a d-clique partition if and only if G has one.*

Proof sketch. If G has a d-clique partition, then $\hat{G}$ also has one since $\hat{G}$ is a supergraph of G with the same set of vertices. For the other direction, we make the following crucial observation: in a valid partition, each cut vertex must be packed with a $(d-1)$-clique of the color column preceding or following it on the same color gadget. This holds, since every $(d-1)$-clique of an id gadget of a different color contains at least one vertex of a some different color, not adjacent to this cut vertex, and since all $(d-1)$-cliques are pairwise non-adjacent. Therefore, the only way to pack a cut vertex together with a $(d-1)$-clique is to pack it with a $(d-1)$-clique of its own color gadget. This already suffices, since only the number of "consumed" $(d-1)$-cliques of the left and right color columns decides the state of an id gadget, and not their specific identities. ◀

Finally, we show in the full version that $\hat{G}$ has linear clique-width at most $\mathcal{O}(k)$ by describing an explicit linear $\mathcal{O}(k)$-clique-expression of $\hat{G}$. We prove the following lemma:

▶ **Lemma 22.** *The graph $\hat{G}$ has linear clique-width at most $\mathcal{O}(k)$. Moreover, given the instance H, the graph $\hat{G}$ together with a linear $\mathcal{O}(k)$-clique-expression of $\hat{G}$ can be constructed in polynomial time.*

Proof sketch. We construct the grid-like structure of the $\hat{G}$ column by column, dividing the linear clique-expression into phases. The crucial part of the proof is that in $\hat{G}$ we connect all cut vertices of a cut in a color gadget to all clique-vertices in the column preceding with the corresponding colors, regardless of the color gadget they belong to. Hence, we do not need to distinguish between the vertices of different $(d-1)$-cliques assigned the same color in each column, allowing us to construct each column using $k+c$ labels only, for some constant value c. We also preserve the labels of the clique-vertices of the first column, in order to connect it to the outer cuts following the last column, using additional k labels.

Finally, we show that an edge gadgets can also be constructed using a constant number of labels. It is essential here that in a selection gadget we only distinguish four types of vertices. We also use a special "forget" label to relabel vertices, whose neighbors have already been introduced, and are already adjacent to them, which allows us to reuse labels along the construction. ◀

Proof of Theorem 2. The $W[1]$-hardness follows from Corollary 14, since the reduction presented above fulfills all the requirements of a parameterized reduction. For the latter part, assume that there exists an algorithm solving d-Clique Partitioning in time $n^{o(k^{d-1})}$ on graphs of linear clique-width k when given together with a linear k-clique expressions. We show that this would contradict ETH by providing an algorithm solving $(d-1)$-Binomial Multi-Colored Clique in time $n^{o\left(\binom{k}{d}\right)}$.

Given an instance H of $(d-1)$-Binomial Multi-Colored Clique, we construct the graph $\hat{G}$ together with a linear k-clique-expression of $\hat{G}$ of width $\mathcal{O}(k)$ as described above. By Lemma 22 this can be done in polynomial time. It holds by Lemma 15 and Lemma 20 that H has a multi-colored clique if and only if $\hat{G}$ has a d-clique partition. Hence, we can solve the instance H by running the assumed algorithm on the instance $\hat{G}$. Since the size of $\hat{G}$ is bounded polynomially, this algorithm solves the instance H in time $n^{o(k^{d-1})}$, contradicting ETH. ◀

5 Conclusion

We have presented an algorithm that for each $d \geq 3$ solves d-Clique Packing in $n^{\mathcal{O}(k^{d-1})}$ time. We complemented this by ruling out algorithms running in $n^{o(k^{d-1})}$ time assuming ETH, even for the special case of d-Clique Partitioning.

It would be interesting to further explore the packing problems for (induced) H-subgraphs for fixed graph H relative to clique-width. For the induced case, we expect the same tight bound. Some preliminary results suggest that (unsurprisingly) things get complicated for packing H-subgraphs, depending on H. Are there other natural problems with optimal XP-algorithms with similar running times relative to clique-width?

References

1 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.

2 Benjamin Bergougnoux, Mamadou Moustapha Kanté, and O-joung Kwon. An optimal XP algorithm for hamiltonian cycle on graphs of bounded clique-width. *Algorithmica*, 82(6):1654–1674, 2020. doi:10.1007/S00453-019-00663-9.

3 Benjamin Bergougnoux, Tuukka Korhonen, and Jesper Nederlof. Tight lower bounds for problems parameterized by rank-width. 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 11:1–11:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.STACS.2023.11.

4 Hans L. Bodlaender and Jurriaan Hage. On switching classes, nlc-width, cliquewidth and treewidth. *Theor. Comput. Sci.*, 429:30–35, 2012. doi:10.1016/J.TCS.2011.12.021.

5 Narek Bojikian, Vera Chekan, and Stefan Kratsch. Tight bounds for some classical problems parameterized by cutwidth. In Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman, editors, *33rd Annual European Symposium on Algorithms, ESA 2025, Warsaw, Poland, September 15-17, 2025*, LIPIcs, pages 13:1–13:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ESA.2025.13.

6 Narek Bojikian and Stefan Kratsch. A tight monte-carlo algorithm for steiner tree parameterized by clique-width. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 29:1–29:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICALP.2024.29.

7 Narek Bojikian and Stefan Kratsch. Tight bounds for connected odd cycle transversal parameterized by clique-width. In Akanksha Agrawal and Erik Jan van Leeuwen, editors, *20th International Symposium on Parameterized and Exact Computation, IPEC 2025, Warsaw, Poland, September 17-19, 2025*, LIPIcs, pages 19:1–19:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.IPEC.2025.19.

8 Narek Bojikian and Stefan Kratsch. Tight bounds for clique-packing parameterized by clique-width. *arXiv preprint arXiv:2606.31873*, 2026. doi:10.48550/arXiv.2606.31873.

9 Narek Bojikian and Stefan Kratsch. Tight bounds for feedback vertex set parameterized by clique-width. In Sayan Bhattacharya, Danupon Nanongkai, Michael Benedikt, and Gabriele Puppis, editors, *53rd International Colloquium on Automata, Languages, and Programming, ICALP 2026, Royal Holloway, University of London, Egham, United Kingdom, July 7-10, 2026*, volume 374 of *LIPIcs*, pages 39:1–39:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. doi:10.4230/LIPIcs.ICALP.2026.39.

10 Vera Chekan and Stefan Kratsch. Tight algorithmic applications of clique-width generalizations. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, Bordeaux, France, August 28 - September 1, 2023*, LIPIcs, pages 35:1–35:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.MFCS.2023.35.

11 Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000. doi:10.1007/s002249910009.

12 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

13 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Intractability of clique-width parameterizations. *SIAM J. Comput.*, 39(5):1941–1956, 2010. doi:10.1137/080742270.

14 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM J. Comput.*, 43(5):1541–1563, 2014. doi:10.1137/130910932.

15 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Clique-width III: hamiltonian cycle and the odd case of graph coloring. *ACM Trans. Algorithms*, 15(1):9:1–9:27, 2019. doi:10.1145/3280824.

16 Martin Fürer. A natural generalization of bounded tree-width and bounded clique-width. In Alberto Pardo and Alfredo Viola, editors, *LATIN 2014: Theoretical Informatics - 11th Latin American Symposium, Montevideo, Uruguay, March 31 - April 4, 2014. Proceedings*, Lecture Notes in Computer Science, pages 72–83. Springer, 2014. doi:10.1007/978-3-642-54423-1_7.

17 Martin Fürer. Multi-clique-width. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, Berkeley, CA, USA, January 9-11, 2017*, LIPIcs, pages 14:1–14:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.14.

18 Robert Ganian, Thekla Hamm, Viktoriia Korchemna, Karolina Okrasa, and Kirill Simonov. The fine-grained complexity of graph homomorphism parameterized by clique-width. *ACM Trans. Algorithms*, 20(3):19, 2024. doi:10.1145/3652514.

19 Falko Hegerfeld and Stefan Kratsch. Tight algorithms for connectivity problems parameterized by clique-width. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, volume 274 of *LIPIcs*, pages 59:1–59:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.59.

20 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.

21 Lars Jaffke, Paloma T. Lima, and Daniel Lokshtanov. b-coloring parameterized by clique-width. *Theory Comput. Syst.*, 68(4):1049–1081, 2024. doi:10.1007/S00224-023-10132-0.

22 Ojvind Johansson. Clique-decomposition, nlc-decomposition, and modular decomposition-relationships and results for random graphs. *Congressus Numerantium*, pages 39–60, 1998.

23 Daniel Kobler and Udi Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discret. Appl. Math.*, 126(2-3):197–221, 2003. doi:10.1016/S0166-218X(02)00198-1.

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

25 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bull. EATCS*, 105:41–72, 2011. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/92.

26 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.