Generalized Graph Packing Problems Parameterized by Treewidth

Barış Can Esmer
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
Saarbrücken Graduate School of Computer Science, Saarland Informatics Campus, Germany

Dániel Marx
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Abstract

H-Packing is the problem of finding a maximum number of vertex-disjoint copies of H in a given graph G. H-Partition is the special case of finding a set of vertex-disjoint copies that cover each vertex of G exactly once. Our goal is to study these problems and some generalizations on bounded-treewidth graphs. The case of H being a triangle is well understood: given a tree decomposition of G having treewidth tw, the K_3-Packing problem can be solved in time $2^{\mathrm{tw}} \cdot n^{O(1)}$, while Lokshtanov et al. [*ACM Transactions on Algorithms* 2018] showed, under the Strong Exponential-Time Hypothesis (SETH), that there is no $(2-\epsilon)^{\mathrm{tw}} \cdot n^{O(1)}$ algorithm for any $\epsilon > 0$ even for K_3-Partition. Similar results can be obtained for any other clique K_d for $d \geq 3$. We provide generalizations in two directions:

- We consider a generalization of the problem where every vertex can be used at most c times for some $c \geq 1$. When H is any clique K_d with $d \geq 3$, then we give upper and lower bounds showing that the optimal running time increases to $(c+1)^{\mathrm{tw}} \cdot n^{O(1)}$. We consider two variants depending on whether a copy of H can be used multiple times in the packing.
- If H is not a clique, then the dependence of the running time on treewidth may not be even single exponential. Specifically, we show that if H is any fixed graph where not every 2-connected component is a clique, then there is no $2^{o(\mathrm{tw} \log \mathrm{tw})} \cdot n^{O(1)}$ algorithm for H-Partition, assuming the Exponential-Time Hypothesis (ETH).

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases Graph Packing, Graph Partitioning, Parameterized Complexity, Treewidth, Pathwidth, pw-SETH, Single-Exponential Lower Bound, Slightly Superexponential Lower Bound

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.3

Related Version *Full Version*: `http://arxiv.org/abs/2509.06091`

1 Introduction

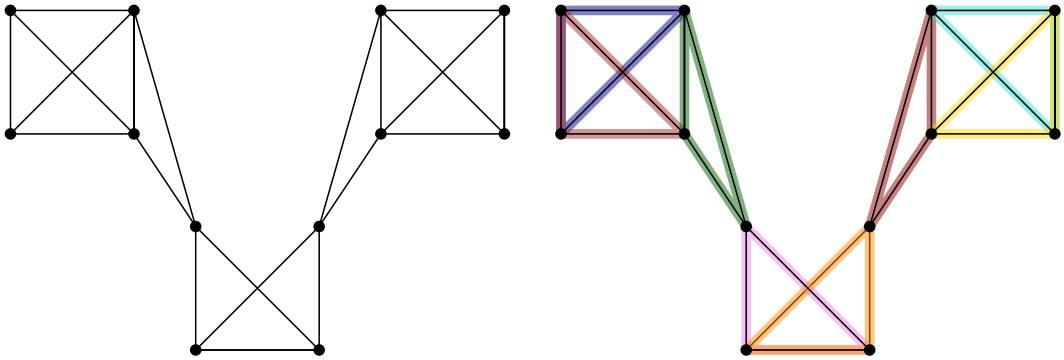
Parameterized complexity theory has proven instrumental in systematically understanding the computational complexity of various combinatorial problems under different parameterizations. Parameterization by treewidth implies tractability for a large number of fundamental algorithmic problems. A prominent line of research has emerged around classifying the complexity of classical NP-hard graph problems under this parameterization framework [8, 7].

In their influential work, Lokshtanov et al. [8] studied six classical combinatorial problems for which parameterized algorithms are known where the running-time dependence on treewidth is single exponential. They showed that, under Strong Exponential Time Hypothesis (SETH), the running times are optimal in the base of the exponent. Following this work, significant efforts have been devoted to generalizing and extending these results. In particular, five out of six problems in [8] have been put into a wider context and generalized to an infinite family of problems: q-Coloring was generalized into H-homomorphism problems [1, 10, 9], Independent Set (equivalent to Vertex Cover), MaxCut and Odd Cycle Transversal [2] into H-homomorphism deletion problem, and Dominating Set into general (σ, ρ)-dominating set problems [5]. Curticapean and Marx [3] showed tight lower bounds for the problem of

counting perfect matchings, which was later extended into general factor problems. However, from the initial six problems studied by Lokshtanov et al. [8], the triangle packing problem has remained unaddressed by prior generalization efforts.

The $2^{tw} \cdot n^{\mathcal{O}(1)}$ running time for triangle packing stated in [8] is actually valid for any clique packing problem, instead of triangle with just three vertices. In this paper, we investigate further generalizations of the triangle packing problem. This line of research naturally gives rise to two conceptually distinct directions:

1. **Allowing vertices to be covered multiple times:** The motivation for this generalization stems naturally from closely related problems such as fractional packing, where vertices can inherently contribute fractionally or multiple times to different packings. To illustrate, Figure 1.1 shows a graph that does not admit a triangle partition, yet allows a collection of triangles that cover each vertex exactly twice. Motivated by this observation, we formally define and explore a generalized packing problem, where each vertex of G can be covered at most c times, for any fixed integer $c \geq 1$. Unlike $c = 1$ case, this generalization demands careful consideration of the definition, specifically on whether a copy of H may appear multiple times in the solution.
2. **Considering packings of arbitrary graphs:** The second natural direction involves generalizing triangle packing by replacing triangles with an arbitrary fixed graph H. This problem has already been studied in the literature under the name of H-partition in [6]. Specifically, in [6] the authors prove the NP-Hardness of the H-partition problem where H has a connected component with at least three vertices. In this paper we let H be a connected graph with at least three vertices and study the complexity of the H-partition problem parameterized by treewidth.



(a) The graph under consideration.

(b) Each color represents a triangle in the collection. Observe that each vertex is covered exactly twice.

Figure 1.1 A graph that does not admit a triangle partition, but allows a collection of triangles such that each vertex is covered exactly twice.

1.1 Our Results

Let H be a fixed graph and $c \geq 1$ be an integer. Given a graph G, a subgraph Z of G is called a copy of H if Z is isomorphic to H. Moreover, if Z is a copy of H in G, we say that Z covers v if $v \in V(Z)$.

▶ **Definition 1.** *We say that a multiset (respectively, set)* $\mathcal{S} = \{(V_1, E_1), \ldots, (V_k, E_k)\}$ *of subgraphs of* G *is a* (c, H)*-multi-packing (respectively,* (c, H)*-single-packing) in* G *if*

1. *each* (V_i, E_i) *is isomorphic to* H *for* $1 \leq i \leq k$
2. *each vertex* v *of* G *is covered at most* c *times by the subgraphs in* $\mathcal{S}$*.*

The collection $\mathcal{S}$ *is called a* (c, H)*-multi-partition (respectively,* (c, H)*-single-partition) of* G *if each vertex* $v \in V(G)$ *is covered exactly* c *times.*

Observe that when c is equal to 1, two copies of H are not allowed to have any common vertices. Therefore, in this case, the notions of (c, H)-single-packing and $\big(c, H\big)$-multi-packing coincide, and we write H-packing to simpify the notation. We define H-partition in a similar way. The first problem we introduce, `Generalized Graph Packing`(H), asks the maximum size of an H-packing in G.

`Generalized Graph Packing`(H)
Input: A graph G
Output: The size of a largest H-packing in G.

For a fixed H, we show that `Generalized Graph Packing`(H) can be solved in time $2^{\mathcal{O}(w \cdot \log w)} \cdot n^{\mathcal{O}(1)}$ for graphs of treewidth w.

▶ **Theorem 2.** *Let H be an arbitrary graph such that it contains at least 3 vertices. Then,* `Generalized Graph Packing`(H) *can be solved in time $2^{\mathcal{O}(w \cdot \log(w))} \cdot n^{\mathcal{O}(1)}$ for all n-vertex graphs G where w is the treewidth of G.*

Then, we define partitioning problems in which each vertex of G must be covered the same number of times. For an arbitrary graph H, we let $c = 1$ which results in vertex disjoint copies of H.

`Generalized Graph Partitioning`(H)
Input: A graph G
Question: Is there an H-packing of G such that each vertex is covered exactly once?

Observe that `Generalized Graph Partitioning`(H) is a special case of the problem `Generalized Graph Packing`(H), which implies an algorithm with running time $2^{\mathcal{O}(w \cdot \log w)} \cdot n^{\mathcal{O}(1)}$ for the `Generalized Graph Partitioning`(H) problem on graphs of treewidth w. We show that this running time cannot be improved for many choices of H.

▶ **Theorem 3.** *Let H be an arbitrary graph with at least 3 vertices such that H is not a block graph. Then, there is no algorithm for* `Generalized Graph Partitioning`(H) *problem, that solves all instances G in time $2^{o(w \cdot \log w)} \cdot n^{\mathcal{O}(1)}$ where w is the treewidth of G, unless ETH fails.*

When H is a clique, we consider the more general problem in which a vertex $v \in V(G)$ can be covered more than once, at most c times for some $c \geq 1$. Therefore, we distinguish between two variants of the problem: one where each clique can be selected at most once, and another without restriction. The `Multi-Clique Packing`(c, d) problem asks the maximum size of a $\big(c, K_d\big)$-multi-packing in G.

`Multi-Clique Packing`(c, d)
Input: A graph G
Output: The size of a largest $\big(c, K_d\big)$-multi-packing in G.

Similar to the `Multi-Clique Packing`(c, d) problem, the `Single-Clique Packing`(c, d) problem asks the maximum size of a (c, K_d)-single-packing in G.

`Single-Clique Packing`(c, d)
Input: A graph G
Output: The size of a largest (c, K_d)-single-packing in G.

In the full version of the paper we show that `Single-Clique Packing`(c, d) admits a single-exponential time algorithm where the same algorithmic result also applies to `Multi-Clique Packing`(c, d) with slight modifications.

▶ **Theorem 4.** *Let $c \geq 1$ and $d \geq 3$ be integers. Then,* `Single-Clique Packing`(c, d) *can be solved in time $(c+1)^w \cdot n^{\mathcal{O}(1)}$ for all n-vertex graphs G given together with a tree decomposition of width at most w.*

Moreover, we define partitioning problems where H is a clique.

`Multi-Clique Partitioning`(c, d)
Input: A graph G
Question: Is there a $\big(c, K_d\big)$-multi-partition of G?

`Single-Clique Partitioning`(c, d)
Input: A graph G
Question: Is there a (c, K_d)-single-partition of G?

Similarly, we prove that this running time is optimal up to polynomial factors in the size of the input graph.

▶ **Theorem 5.** *Let $c \geq 1$ and $d \geq 3$ be integers. If there exists an $\varepsilon > 0$ such that* `Multi-Clique Partitioning`(c, d) *can be solved in time $(c+1-\varepsilon)^w \cdot n^{\mathcal{O}(1)}$ for all n-vertex graphs G given together with a path decomposition of width at most w, then the* **pw**-*SETH fails.*

Finally, we show that the lower bound result in Theorem 5 can also be transferred similarly.

▶ **Theorem 6.** *Let $c \geq 1$ and $d \geq 3$ be integers. If there exists an $\varepsilon > 0$ such that* `Single-Clique Partitioning`(c, d) *can be solved in time $(c+1-\varepsilon)^w \cdot n^{\mathcal{O}(1)}$ for all n-vertex graphs G given together with a path decomposition of width at most w, then the* **pw**-*SETH fails.*

2 Technical Overview

In this section, we will give an overview of the techniques and ideas presented in the paper.

2.1 Preliminaries

A graph H is k-connected if for each $A \subseteq V(H)$ such that $|A| = k-1$ it holds that $H \setminus A$ is connected. A graph is also called biconnected if it is 2-connected, and a block is a maximal 2-connected component of H. Similarly, a graph H is called a block graph if every block of H is a clique.

A vertex v of a connected graph H is a cutvertex if $G \setminus v$ is disconnected. In this paper, we refer to both the blocks of a graph and the nodes of the corresponding block tree interchangeably, with a slight abuse of notation. While formally distinct, we find it convenient to treat them as equivalent entities for the sake of clarity and brevity. For a vertex $h \in V(H)$ and a block $B \in \mathcal{B}_H$, we write $h \in B$ if $h \in V(B)$.

▶ **Definition 7.** *For a function f and a set X, we let $f\restriction_X$ denote the restriction of f to $X \cap \operatorname{dom}(f)$. Similarly, we let $f\restriction_{-X}$ denote the restriction of f to $\operatorname{dom}(f) \setminus X$. Finally, for $v \notin \operatorname{dom}(f)$ and a value y, we let $f\restriction_{[v\to y]}$ denote a function g which is defined as*

$$g(x) = \begin{cases} f(x) & \text{if } x \in \operatorname{dom}(f), \\ y & \text{if } x = v. \end{cases}$$

We use the Iverson bracket $[\![P]\!]$, which is defined to be 1 if the proposition P is true and 0 otherwise.

2.2 Gadgets

In this paper, we derive our lower bounds via reductions from well-studied base problems whose intractability is established under standard complexity hypotheses. Each reduction makes use of compact, purpose-built gadgets – small graphs whose behavior can be precisely engineered. Embedding these gadgets into our constructions yields intuitive, transparent proofs that highlight the underlying ideas without excessive technical overhead.

More formally, we define a gadget G as a graph with designated portal vertices $\{p_1, \ldots, p_\ell\} \subseteq V(G)$, where the vertices $\operatorname{int}(G) := (G \setminus \{p_1, \ldots, p_\ell\})$ are called internal vertices. We say that a graph E is an extension of a gadget G if E contains G as an induced subgraph, where each internal vertex v of G satisfies

$$N_E(v) \subseteq V(G).$$

In other words, in an extension E, only the portal vertices of G can have neighbors outside of G.

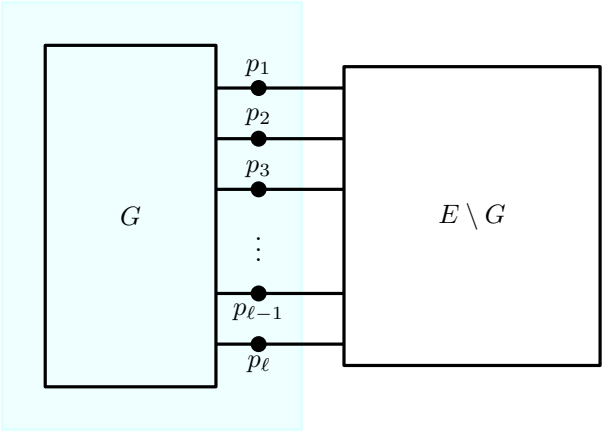


Figure 2.1 Illustration of an extension of gadget G with ℓ portal vertices: the blue region on the left highlights G along with its portals, while the rectangle on the right depicts the remainder $E \setminus G$.

We also require gadgets to behave in a structured way. Whenever a copy of H intersects the gadget, it must lie entirely within the gadget when connected to a larger graph. We formalize this in the following definition.

▶ **Definition 8.** *Let G be a gadget and P_G be the set of its portal vertices. For a graph H and $c \geq 1$, we say that G is (c, H)-internally coherent if, for any extension E of G and any (c, H)-multi-partition / (c, H)-single-partition $\mathcal{Z}$ of E, the following holds: If there exists $Z \in \mathcal{Z}$ that contains an internal vertex of G, then all vertices in Z must belong to G. Formally,*

$$\Big(V(Z) \cap \big(V(G) \setminus P_G\big)\Big) \neq \emptyset \implies V(Z) \subseteq V(G).$$

Next, we define the relation realized by a gadget.

▶ **Definition 9.** *Let H be a graph, $c \geq 1$ and G be a gadget with portal vertices $P_G = \{p_1, \ldots, p_\ell\}$. We say G dist-(c, H)-realizes (respectively, arb-(c, H)-realizes) a relation $R \subseteq \{0, \ldots, c\}^\ell$ if the following holds:*

$r \in R \iff$ *There exists a (c, H)-single-packing (respectively, $\big(c, H\big)$-multi-packing) $\mathcal{Z}$ of G such that $\mathcal{Z}$ covers each internal vertex of G exactly c times, and each portal vertex p_i exactly r_i times for $1 \leq i \leq \ell$.*

We say that G strict-(c, H)-realizes R if it both dist-(c, H) and arb-(c, H)-realizes R.

▶ Remark 10. When $c = 1$, the notions of dist-(c, H)-realization and arb-(c, H)-realization coincide. In this case, we simply say that the relation R is H-realized by G, instead of using the term "strict-$(1, H)$".

We say a relation $R \subseteq \{0, \ldots, c\}^\ell$ is (x, d)-regular for $d \geq 1$ and $0 \leq x \leq d - 1$ if for each $r \in R$, the weight of r is equivalent to $x \mod d$, i.e. $\mathbf{w}(r) \coloneqq \left(\sum_{i \in [\ell]} r_i\right) = x \mod d$. Moreover, a relation R has weight X if $\left(\max_{r \in R} \mathbf{w}(r)\right) = X$. Recall that a gadget is a small, engineered graph used to enforce specific behaviors in a reduction. We streamline the lower bound constructions by building general-purpose gadgets that can realize any relation. The descriptions of the gadgets are deferred to the full version of the paper. This way, we avoid repetitive constructions and keep the focus on the core ideas.

▶ **Lemma 11.** *Let H be an arbitrary graph, $\ell \geq 1$ be an integer and $R \subseteq \{0, 1\}^\ell$ be a relation that is $(x, |H|)$-regular for some $0 \leq x \leq |H| - 1$. Then, there exists a $(1, H)$-internally coherent gadget G that H-realizes the relation R and the size of G is bounded by some function of ℓ. Moreover, for relations with constant weight, it holds that $\mathbf{pw}(G) = \mathcal{O}(\ell)$.*

When $c \geq 1$, we require the relation to have more structure, i.e., the relation should be $(0, |H|)$-regular. However, this restriction can be easily handled in our lower bound constructions.

▶ **Lemma 12.** *Let $c \geq 1$, H be a clique, $\ell \geq 1$ be a constant and $R \subseteq \{0, \ldots, c\}^\ell$ be a $(0, |H|)$-regular relation. Then, there exists a (c, H)-internally coherent gadget G that strict-(c, H)-realizes the relation R. Moreover, the size of G is bounded by some function of ℓ.*

2.3 Single-Exponential Lower Bound

In this section, we give an overview of how the above-described gadgets are used to prove Theorem 6. Classically, when proving conditional lower bounds based on SETH, one needs to find a reduction from SAT. However, this usually involves carrying out repetitive, unnecessary work that is not specific to the problem one is working on. One of the strengths of the framework introduced by Lampis [7] is removing the need for such repetitive constructions.

The primal graph of a CSP instance ψ is a graph G that has a vertex for each variable of ψ, and there is an edge between $x, y \in V(G)$ if x and y appear together in a constraint. The pathwidth of the CSP instance ψ is defined to be the pathwidth of its primal graph. Similarly, by path decomposition of ψ, we mean a path decomposition of the primal graph of ψ. The following lemma from [7] implies that we can assume the path decomposition to be nice.

▶ **Lemma 13** (Lemma 2.1 in [7], restated for CSPs). *There is a linear-time algorithm that takes as input a CSP formula ψ with n variables and m constraints and a path decomposition of its primal graph of width p and outputs a nice path decomposition $B_1, \ldots, B_t$ of ψ containing at most $t = \mathcal{O}(p \cdot m)$ bags, as well as an injective function b from the set of constraints of ψ to $[t]$ such that for each constraint c, $B_{b(c)}$ contains all the variables of c.*

The following conjecture from [7], called **pw**-SETH, will form the basis of our hardness results.

▶ **Conjecture 14** (Conjecture 1.1 from [7].). *For all $\epsilon > 0$ we have the following: there exists no algorithm which takes as input a $3-SAT$ instance ϕ on n variables and a path decomposition of its primal graph of width* **pw** *and correctly decides if ϕ is satisfiable in time $(2-\epsilon)^{\mathbf{pw}} n^{O(1)}$.*

Moreover, we have the following result from [7], which proves the equivalence of falsifying **pw**-SETH and finding a faster algorithm for 2-CSP.

▶ **Theorem 15** (Theorem 3.2 from [7], shortened). *For each $B \geq 3$ the following statements are equivalent:*

1. *The* **pw**-*SETH is false.*
2. *There exist $\epsilon > 0, b > 0$ and an algorithm that takes as input a 2-CSP instance ψ on alphabet $[B]$, together with a path decomposition of ψ, and decides if ψ is satisfiable in time $(B-\epsilon)^{\mathbf{pw}} \cdot |\psi|^b$.*

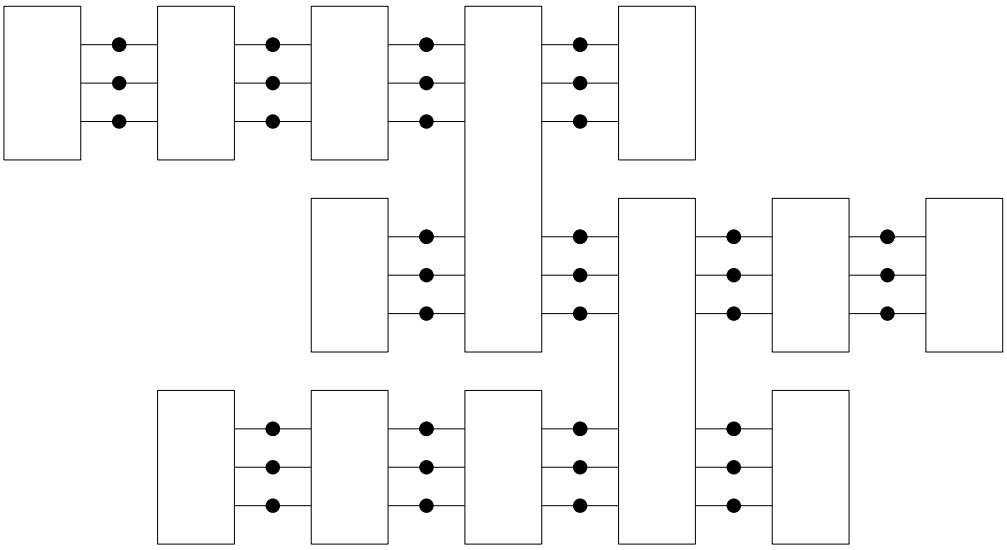


Figure 2.2 A high-level description of the lower bound construction used to prove Theorem 5. The coverage counts of all vertices in a column represent an assignment to the variables of the 2-CSP instance. The rectangles depict the gadgets that enforce the constraints; observe that they interact locally with the vertices, which allows us to bound the pathwidth of the constructed instance.

Motivated by Theorem 15, Section 3 presents a reduction from the 2-CSP problem to the `Single-Clique Partitioning`(c,d) problem. The core idea is to encode each variable's B possible assignments using a set of vertices whose coverage count in a (c, K_d)-single-packing represents the chosen value. We then introduce gadgets each enforcing a single constraint

of the 2-CSP instance via carefully specified relations so that only coverage patterns corresponding to satisfying assignments are possible. By designing these gadgets to interact locally, we ensure the pathwidth of the resulting instance increases by at most a constant, completing the reduction.

2.4 Slightly Superexponential Lower Bound

We now give a high-level description of the proof of Theorem 3. Let H be a graph with at least 3 vertices that is not a block graph. Next, we will describe the lower bound for the `Generalized Graph Partitioning`(H) problem which is stated in Theorem 3. Recall that `Generalized Graph Partitioning`(H) is solvable in slightly super-exponential time for any fixed graph H, i.e., there is an algorithm with running time $\mathcal{O}^*\left(2^{O(tw\cdot\log tw)}\right)$.[1] We show that this running time is optimal under ETH, i.e. there exists no algorithm for `Generalized Graph Partitioning`(H) with running time $\mathcal{O}^*\left(2^{o(tw\cdot\log(tw))}\right)$. To that end, we use an auxiliary problem for which such a lower bound was presented in [4]. Specifically, we define the $k \times k$ Permutation Independent Set problem where given a graph G on a vertex set $[k] \times [k]$, we ask whether there exists an independent set X in G that contains exactly one vertex from each row and each column.

$k \times k$ Permutation Independent Set
Input: A graph G on the vertex set $[k] \times [k]$
Question: Is there an independent set X of G such that X induces a permutation on $[k] \times [k]$?

The following hardness result was presented for the analogous $k \times k$ Permutation Clique problem in [4] which translates easily to $k \times k$ Permutation Independent Set. Below we state it for $k \times k$ Permutation Independent Set.

▶ **Theorem 16** (Theorem 14.14 in [4]). *Unless ETH fails, $k \times k$ Permutation Independent Set cannot be solved in time $2^{o(k\cdot\log k)}$.*

To prove Theorem 3, we reduce the $k \times k$ Permutation Independent Set problem to `Generalized Graph Partitioning`(H). The construction relies on structural properties of H. Let B be a block in H with a minimum separator S such that $B \setminus S$ has at least two connected components. Since H contains a non-clique block, such B and S exist. We partition S into two subsets, U and D, and create k copies of each. The construction in the proof of Theorem 3 ensures that any $|H|$-packing includes k copies of H, each covering exactly one copy of U and one of D. This yields $k! = 2^{\mathcal{O}(k\cdot\log k)}$ configurations, corresponding to all permutations of a set of size k. Additionally, for each edge e in the $k \times k$ Permutation Independent Set instance, we introduce a gadget to prevent e from being included in the set of vertices induced by this permutation. Moreover, these gadgets are designed to preserve pathwidth by interacting only locally. The detailed proof of Theorem 3 is deferred to the full version of the paper.

[1] The $O^*(f(k))$ notation suppresses polynomial factors in the input size n; that is, $O^*(f(k)) = O(f(k)\cdot n^c)$ for some constant c.

2.5 Algorithmic Results

Recall that a tree decomposition of a graph $G = (V, E)$ is a pair $(T, \{X_t\}_{t \in T})$, where T is a tree whose every node t is assigned a bag $X_t \subseteq V(G)$, satisfying the following properties:

1. For every vertex $v \in V$, there exists at least one bag X_t such that $v \in X_t$.
2. For every edge $(u, v) \in E$, there exists a bag X_t such that $\{u, v\} \subseteq X_t$.
3. For every vertex $v \in V$, the set of nodes $\{t \in V(T) \mid v \in X_t\}$ induces a connected subtree of T.

The *width* of a tree decomposition is the size of its largest bag minus one, and the *treewidth* of G is the minimum width over all possible tree decompositions of G. A nice tree decomposition is a rooted tree decomposition in which each node is one of the four types: leaf, introduce, forget or join. We refer the reader to [4] for more details.

The algorithmic results in Theorems 2 and 4 follow a fairly standard dynamic programming framework over tree decompositions. We define a suitable set of states for each bag in the decomposition, capturing the essential information needed to extend partial solutions. The main challenge of the approach lies in carefully designing these states and proving the correctness of the update rules that determine how states transition from one bag to the next. Usually what determines the running time is the state representation and transitions to the specific structural constraints of our problem.

In particular, the single-exponential algorithm for the `Single-Clique Packing`(c, d) problem defines the state of a bag based on how many times each vertex in the bag is covered by a partial solution. This yields $(c+1)^\ell$ states for a bag of size ℓ, naturally leading to a running time of $(c+1)^{tw} \cdot n^{\mathcal{O}(1)}$.

In contrast, the slightly superexponential algorithm for the `Generalized Graph Packing`(H) problem requires keeping track of a partition of the bag, in which each part corresponds to a partial copy of H. This results in $\mathcal{O}(\ell^\ell)$ possible states for a bag of size ℓ, leading to an overall running time of $2^{\mathcal{O}(tw \cdot \log tw)} \cdot n^{\mathcal{O}(1)}$. The algorithms are presented in detail in the full version of the paper.

3 Lower Bounds for Clique Partitioning Problems

In this section we prove Theorems 5 and 6. Theorem 15 says that in order to prove a conditional lower bound based on **pw**-SETH, one can start the reduction from the 2-CSP problem. In the following, we will present a reduction from the 2-CSP problem to the `Multi-Clique Partitioning`(c, d). Subsequently, we will describe another reduction from `Multi-Clique Partitioning`(c, d) to `Single-Clique Partitioning`(c, d) problem, and prove Theorems 5 and 6.

3.1 Lower Bounds

We first prove Theorem 5. To that end, intuitively, we show that a fast algorithm for the `Multi-Clique Partitioning`(c, d) problem implies a fast algorithm for the 2-CSP problem.

▶ **Lemma 17.** *Let $c \geq 1$ and $d \geq 3$ be integers. Suppose there exists an $\varepsilon > 0$ such that* `Multi-Clique Partitioning`(c, d) *can be solved in time $(c + 1 - \varepsilon)^{\mathbf{pw}(G)} \cdot n^{\mathcal{O}(1)}$ for all n-vertex graphs G given together with a path decomposition of width at most $\mathbf{pw}(G)$. Then, there exist $\epsilon', b' > 0$, an integer $B \geq 1$ and an algorithm that takes as input a 2-CSP instance ψ on alphabet $[B]$, together with a path decomposition of ψ, and decides if ψ is satisfiable in time $(B - \epsilon')^{\mathbf{pw}} \cdot |\psi|^{b'}$.*

Proof. Let ε, b be as in the lemma statement and let $\mathcal{A}$ be the hypothetical algorithm for `Multi-Clique Partitioning`(c, d). Moreover, we let H denote the clique K_d and let ℓ be the smallest integer that is a multiple of $|H|$ such that

$$(1-\varepsilon)\cdot|H| < \frac{\varepsilon}{2}\cdot(\ell-|H|)\,. \tag{3.1}$$

Observe that ℓ is a constant that only depends on ε and H.

Let $B = (c+1)^{\ell-|H|}$ and $\varepsilon' = \frac{\varepsilon}{2}$. We will now present a reduction from 2-CSP with alphabet size B to `Multi-Clique Partitioning`(c, d). The idea is as follows: in order to make use of regular relations, we will represent each variable of the 2-CSP instance by ℓ vertices. Note that each of the ℓ vertices can be covered between 0 and c times, which can be thought of as the state of a vertex. In total, ℓ vertices combined give rise to $(c+1)^\ell$ states. These states can also be visualized as vectors $r \in \{0,\ldots,c\}^\ell$. Next, we consider the following subset of vectors

$$Z \coloneqq \{x \in (c+1)^\ell \mid w(x) = 0 \mod |H|\}.$$

Observe that we have $|Z| \geq (c+1)^{\ell-|H|} = B$, because we can append to each vector in $r' \in \{0,\ldots,c\}^{\ell-|H|}$ at most $|H|$ many 1's so that the new vector $r \in \{0,\ldots,c\}^\ell$ constructed this way satisfies $w(r) = 0 \mod |H|$. Hence there exists an injective map $\Phi\colon [B] \to Z$ where we define $W \coloneqq \operatorname{im}(\Phi)$. We use the set W to simulate B many assignments to a variable of the 2-CSP instance. Observe that W, as a relation, is $(0, |H|)$-regular.

Construction of the `Multi-Clique Partitioning`(c, d) instance. Let ψ be a 2-CSP instance with variables $x_1,\ldots,x_n$, contraints $C_1,\ldots,C_m$ and alphabet $[D]$. Let $\mathcal{P} = (B_1,\ldots,B_t)$ be a nice path decomposition of width p. Finally, let $b\colon [m] \to [t]$ be a function that maps each constraint C_i of ψ to a bag such that $B_{b(i)}$ contains the variables occurring in C_i. We will construct an instance of the `Multi-Clique Partitioning`(c, d) problem as follows:

1. For each $1 \leq i \leq n$, define $l(i) \in [t]$ to be the smallest integer such that $x_i \in B_{l(i)}$. Similarly, let $r(i)$ be the largest integer such that $x_i \in B_{r(i)}$. For each $i \in [n]$ and $l(i) \leq j \leq r(i)+1$, introduce ℓ vertices $\{a_1^{i,j},\ldots,a_\ell^{i,j}\}$.
2. Define the relation

   $$W^C \coloneqq \operatorname{Compl}_c\big(W\big).$$

   Observe that W is $(0, |H|)$-regular by construction. The same also holds for W^C, because for each $x \in W^C$ such that $x = \operatorname{Compl}_c\big(s\big)$ for $s \in W$, we have

   $$w(x) = \Big(\ell\cdot c - w(s)\Big) = 0 \mod |H|,$$

   where the last equivalence holds because ℓ and $w(s)$ are both equivalent to $0 \mod |H|$. Introduce two gadgets L^i and R^i that arb-(c, H)-realize the relation W^C and W, respectively, which exist by Lemma 12. Then, identify the portal vertices of L^i with $(a_1^{i,l(i)},\ldots,a_\ell^{i,l(i)})$, and similarly, identify the portal vertices of R^i with the vertices $\{a_1^{i,r(i)+1},\ldots,a_\ell^{i,r(i)+1}\}$.
3. Let $j \in [t]$. We say that j represents s for $s \in [m]$ if $b(s) = j$. In that case, we define S_j to be the relation, and i_1 and i_2 to be the indices of the variables associated with the constraint C_s. We define the new relation

   $$R_j \coloneqq \left\{\Phi(u_1)\odot\Phi(u_2)\odot\operatorname{Compl}_c\big(\Phi(u_1)\big)\odot\operatorname{Compl}_c\big(\Phi(u_2)\big) \;\middle|\; (u_1,u_2)\in S_j\right\}.$$

Observe that R_j is $(0, |H|)$-regular because for each $x \in R_j$ we have

$$w(x) = (2 \cdot \ell \cdot c) = 0 \mod |H|$$

since ℓ is a multiple of $|H|$. Hence, we can introduce a gadget $N^j_{i_1,i_2}$ by Lemma 12 that arb-(c, H)-realizes R_j. Then, we identify the portal vertices of $N^j_{i_1,i_2}$ with the vertices

$$\left(a_1^{i_1,j}, \ldots, a_\ell^{i_1,j}\right) \odot \left(a_1^{i_2,j}, \ldots, a_\ell^{i_2,j}\right) \odot \left(a_1^{i_1,j+1}, \ldots, a_\ell^{i_1,j+1}\right) \odot \left(a_1^{i_2,j+1}, \ldots, a_\ell^{i_2,j+1}\right). \quad (3.2)$$

Next, we define the relation COPY $\subseteq \{0, \ldots, c\}^{2\cdot\ell}$ where

$$\text{COPY} \coloneqq \{\boldsymbol{u} \odot \text{Compl}_c(\boldsymbol{u}) \mid \boldsymbol{u} \in W\} \quad (3.3)$$

and let

$$\Gamma_j \coloneqq \begin{cases} \{i_1, i_2\} & \text{if } j \text{ represents } s, \\ \emptyset & \text{otherwise.} \end{cases}$$

Observe that COPY is $(0, |H|)$-regular because for each $x \in$ COPY we have

$$w(x) = \ell \cdot c = 0 \mod |H|.$$

By Lemma 12, there exists a gadget that arb-(c, H)-realizes COPY. Next, for each $i \in \left([n] \setminus \Gamma_j\right)$ such that $l(i) \leq j \leq r(i)$, we introduce a gadget F_i^j that arb-(c, H)-realizes the relation COPY and identifies the portal vertices of F_i^j with

$$\left(a_1^{i,j}, \ldots, a_\ell^{i,j}\right) \odot \left(a_1^{i,j+1}, \ldots, a_\ell^{i,j+1}\right). \quad (3.4)$$

Finally, for each $i \in [n]$ and $j \in [t]$ such that $l(i) \leq j \leq r(i)$, we define the gadget that covers i at step j as

$$K_i^j \coloneqq \begin{cases} N^j_{i_1,i_2} & \text{if } j \text{ represents } s \in [m] \text{ and } i \in \{i_1, i_2\} \\ F_i^j & \text{otherwise.} \end{cases}$$

This is the whole construction of the `Multi-Clique Partitioning`(c, d) instance which we call G.

Equivalence of the instances. We now prove that ψ is satisfiable if and only if G admits a $\left(c, K_d\right)$-multi-partition, by establishing each direction separately.

Suppose that there exists an assignment $(\alpha_1, \ldots, \alpha_n)$ to $(x_1, \ldots, x_n)$ such that ψ is satisfied. In the following, we will describe a $\left(c, K_d\right)$-multi-partition $\mathcal{K}$ of G. For each $i \in [n]$, define $\boldsymbol{a}_i \coloneqq \Phi(\alpha_i) \in W$. Now let $j \in [t]$. Observe that since $\boldsymbol{a}_i \in W$ for each $i \in [n]$, it holds that there exists a $\left(c, K_d\right)$-multi-packing of L^i that covers $(a_1^{i,l(i)}, \ldots, a_\ell^{i,l(i)})$ according to $\text{Compl}_c(\boldsymbol{a}_i)$. Moreover, there exists a $\left(c, K_d\right)$-multi-packing of R^i that covers $(a_1^{i,r(i)+1}, \ldots, a_\ell^{i,l(i)+1})$ according to $\boldsymbol{a}_i$. In both cases, the internal vertices of the gadgets are covered exactly c times.

Now let $j \in [t]$. For all $i \in \left([n] \setminus \Gamma_j\right)$, there exists a $\left(c, K_d\right)$-multi-packing of F_i^j that covers

$$\left(a_1^{i,j}, \ldots, a_\ell^{i,j}\right) \odot \left(a_1^{i,j+1}, \ldots, a_\ell^{i,j+1}\right) \quad (3.5)$$

according to $\boldsymbol{a}_i \odot \mathrm{Compl}_c(\boldsymbol{a}_i)$, because $\boldsymbol{a}_i \in W$. Moreover, if j represents s for some $s \in [m]$, and x_{i_1}, x_{i_2} are the variables corresponding to C_s, then there exists a (c, K_d)-multi-packing that covers

$$\left(a_1^{i_1,j}, \ldots, a_\ell^{i_1,j}\right) \odot \left(a_1^{i_2,j}, \ldots, a_\ell^{i_2,j}\right) \odot \left(a_1^{i_1,j+1}, \ldots, a_\ell^{i_1,j+1}\right) \odot \left(a_1^{i_2,j+1}, \ldots, a_\ell^{i_2,j+1}\right) \quad (3.6)$$

according to $\boldsymbol{a}_{i_1} \odot \boldsymbol{a}_{i_2} \odot \mathrm{Compl}_c(\boldsymbol{a}_{i_1}) \odot \mathrm{Compl}_c(\boldsymbol{a}_{i_2})$. Again, in both cases, the internal vertices of the gadgets are covered exactly c times. Next, we prove that the remaining vertices are covered c times as well.

▷ Claim 18. It holds that for each $i \in [n]$, $j \in [t]$ such that $l(i) \leq j \leq r(i) + 1$ and $x \in [\ell]$, $a_x^{i,j}$ is covered exactly c times by $\mathcal{K}$.

Proof. We prove the claim by induction on j. Let $j = 1$, $i \in [n]$ and suppose that $l(i) \leq j \leq r(i) + 1$. Since $1 \leq l(i) \leq j = 1$, we have that $l(i) = 1 = j$. Consider the vertices $\left(a_1^{i,1}, \ldots, a_\ell^{i,1}\right)$, which are covered according to $\mathrm{Compl}_c(\boldsymbol{a}_i)$ by L^i. Moreover, $\left(a_1^{i,1}, \ldots, a_\ell^{i,1}\right)$ is covered according to $\boldsymbol{a}_i$ by K_i^1 which follows from (3.5) or (3.6), depending on whether $i \in \Gamma_1$ or not, respectively. All in all, $a_x^{i,1}$ is covered exactly c times for $x \in [\ell]$.

Now suppose that the claim holds for $1 < j \leq t$, and we will prove the claim for $j + 1$. Let $i \in [n]$ such that $l(i) \leq j + 1 \leq r(i) + 1$. Consider the vertices $\left(a_1^{i,j+1}, \ldots, a_\ell^{i,j+1}\right)$, which are covered according to $\mathrm{Compl}_c(\boldsymbol{a}_i)$ by K_i^j. This follows from (3.5) and (3.6). Now, observe that we have either $j < r(i)$ or $j = r(i)$. In both cases, by using the arguments in the $j = 1$ case, one can conclude that $\left(a_1^{i,j+1}, \ldots, a_\ell^{i,j+1}\right)$ is covered according to $\boldsymbol{a}_i$ by K_i^{j+1}. Therefore, all in all, it holds that each vertex $a_x^{i,j}$ is covered exactly c times for $x \in [\ell]$. ◁

Now for the reverse implication, suppose that G has a (c, K_d)-multi-partition. Since each gadget used in the construction of G is (c, K_d)-internally coherent, this implies that for each gadget there is a (c, K_d)-multi-packing that covers its interval vertices exactly c times, and its portal vertices according to the relation associated with it. In particular, for each $i \in [n]$, let $\boldsymbol{b}_i \in W^C$ denote the vector such that L^i covers the vertices $\{a_1^{i,l(i)}, \ldots, a_\ell^{i,l(i)}\}$ according to $\boldsymbol{b}_i$.

By induction, one can show that for each $j \in [t]$ and $i \in [n]$ such that $l(i) \leq j \leq r(i)$, the tuple $(a_1^{i,j}, \ldots, a_\ell^{i,j})$ is covered by K_i^j according to $\boldsymbol{z}_i$ where $\boldsymbol{z}_i = \mathrm{Compl}_c(\boldsymbol{b}_i)$. Hence, we let $\alpha_i \coloneqq \Phi^{-1}(\boldsymbol{z}_i) \in B$. Next, we will prove that $A \coloneqq (\alpha_1, \ldots, \alpha_n)$ is a satisfying assignment for ψ. To that end, let $s \in [m]$ and $j = b(s)$. To show that C_s is satisfied by A, let x_{i_1} and x_{i_2} be the variables associated with C_s. By the above discussion, we know that $\left(a_1^{i_1,j}, \ldots, a_\ell^{i_1,l}\right)$ and $\left(a_1^{i_2,j}, \ldots, a_\ell^{i_2,j}\right)$ is covered according to $\boldsymbol{z}_{i_1}$ and $\boldsymbol{z}_{i_2}$, respectively. By the definition of R_j, $a_{i_1} = \Phi^{-1}(\boldsymbol{z}_{i_1})$ and $a_{i_2} = \Phi^{-1}(\boldsymbol{z}_{i_2})$ satisfy C_s. Therefore, the assignment A satisfies all constraints of ψ, and ψ is satisfied if and only if G has a (c, K_d)-multi-partition.

Pathwidth and size of the constructed instance. To bound the pathwidth of G, we will create a path decomposition by following $\{B_j\}_{j \in [t]}$. Specifically, for each $j \in [t]$, we first create a new path decomposition $\mathcal{X} = (X_1, \ldots, X_t)$ where each X_j is a copy of B_j and we replace each $x_i \in B$ with the vertices $a_1^{i,j}, \ldots, a_\ell^{i,j}$. Note that the size of each X_j is at most $p \cdot \ell$ for $j \in [t]$.

In the following, we will add the remaining vertices of G to the bags in $\mathcal{X}$ such that $\mathcal{X}$ is a valid path decomposition of G. Let $j \in [t]$. For each $i \in [n] \setminus \Gamma_j$ such that $l(i) \leq j \leq r(i)$, we replace the vertices $\{a_x^{i,j}\}_{x \in [\ell]}$ with $\{a_x^{i,j+1}\}_{x \in [\ell]}$ as follows. After the bag X_j ,we first insert the bag $X'_{j,i} \coloneqq \left(X_j \cup V\left(F_i^j\right)\right)$, and then add another bag $X''_{j,i}$ where we replace

$\{a_x^{i,j}\}_{x\in[\ell]}$ in $X'_{j,i}$ with $\{a_x^{i,j+1}\}_{x\in[\ell]}$, and finally, another bag $X'''_{j,i}$ where we remove the vertices $V\left(F_i^j\right)$ from $X''_{j,i}$. For a fixed $j \in [t]$, we keep adding the bags iteratively for each $i \in [n] \setminus \Gamma_j$ such that $l(i) \leq j \leq r(i)$, until all vertices in all the gadgets are contained in the bag decomposition. Note that, by our construction, edges that are adjacent to a vertex in F_i^j are also covered by either $X'_{j,i}$ or $X''_{j,i}$.

Finally, for each $s \in [m]$ and $j = b(s)$, we add the gadgets $N^j_{i_1,i_2}$ to the tree decomposition in a similar way. Since the size of each $N^j_{i_1,i_2}$ and F_i^j is a function of ℓ, it is bounded by a constant. All in all, the pathwidth of G increases at most by a constant. We have

$$\mathbf{pw}(G) = l \cdot p + \mathcal{O}(1).$$

Finally, since t and m are bounded by a polynomial of n, it follows from the construction that

$$V(G) = n^{\mathcal{O}(1)}.$$

Running Time. Constructing the graph G from ψ takes time polynomial in n. By our assumption on $\mathcal{A}$, the whole reduction takes time

$$\begin{aligned}
(c+1)^{(1-\varepsilon)\cdot\mathbf{pw}(G)} \cdot V(G)^b &= (c+1)^{(1-\varepsilon)\cdot\ell\cdot p} \cdot n^{\mathcal{O}(1)} \\
&= (c+1)^{(1-\varepsilon)\cdot(\ell-|H|)\cdot p} \cdot (c+1)^{(1-\varepsilon)\cdot|H|\cdot p} \cdot n^{\mathcal{O}(1)} \\
&= B^{(1-\varepsilon)\cdot p} \cdot (c+1)^{(1-\varepsilon)\cdot|H|\cdot p} \cdot n^{\mathcal{O}(1)} \\
&< B^{(1-\varepsilon)\cdot p} \cdot (c+1)^{\frac{\varepsilon}{2}\cdot(\ell-|H|)\cdot p} \cdot n^{\mathcal{O}(1)} \\
&= B^{(1-\varepsilon)\cdot p} \cdot B^{\frac{\varepsilon}{2}\cdot p} \cdot n^{\mathcal{O}(1)} \\
&= B^{(1-\varepsilon')\cdot p} \cdot |\psi|^{b'}
\end{aligned}$$

where the inequality follows from (3.1) and b' is a constant. ◀

The proof of Theorem 5 follows from Theorem 15 and Lemma 17. We note here that the same construction can be used to prove Theorem 6, because the gadgets used in Lemma 17 strict-(c, K_d)-realize their relations (see Definition 9 and Lemma 12). However, by presenting a simple reduction, we demonstrate that a fast algorithm for `Single-Clique Partitioning`(c, d) implies a fast algorithm for `Multi-Clique Partitioning`(c, d), which is used to prove Theorem 6 in a more formal way.

▶ **Lemma 19.** *Let $c \geq 1$ and $d \geq 3$ be integers. Suppose there exists an $\varepsilon > 0$ such that* `Single-Clique Partitioning`(c, d) *can be solved in time $(c + 1 - \varepsilon)^{\mathbf{pw}(G)} \cdot n^{\mathcal{O}(1)}$ for all n-vertex graphs G given together with a path decomposition of width at most $\mathbf{pw}(G)$. Then, there exist $\epsilon' > 0$ and $b' > 0$ such that* `Multi-Clique Partitioning`(c, d) *can be solved in time $(c+1-\varepsilon')^{\mathbf{pw}(G)} \cdot n^{b'}$ for all n-vertex graphs G given together with a path decomposition of width at most $\mathbf{pw}(G)$.*

Proof. Let ε and b be as in the statement of the lemma. Moreover, let $\mathcal{A}$ be the hypothetical algorithm for `Single-Clique Partitioning`(c, d).

Construction of the `Single-Clique Partitioning`(c, d) instance. Let G be an instance of `Multi-Clique Partitioning`(c, d). Moreover, we let H denote the clique K_d and construct a `Single-Clique Partitioning`(c, d) instance G' as follows. Let G' have the same vertex set as G. We call these vertices the original vertices of G'. Moreover, for each clique X in G of size d, we add a gadget E_X that dist-(c, H)-realizes the relation $\mathrm{EQ}_d^{[0,c]}$. We identify the portal vertices of E_X with $V(X)$. This is the whole construction.

Equivalence of Instances. We will now prove that G admits a $\big(c, K_d\big)$-multi-partition if and only if G' admits a (c, K_d)-single-partition.

Suppose that G has a $\big(c, K_d\big)$-multi-partition which is denoted by $\mathcal{Z}$. For each clique X in G, let a_x denote the number of occurences of X in $\mathcal{Z}$. We construct a (c, K_d)-single-partition $\mathcal{K}$ for G' as follows. For each clique X in G, consider a (c, K_d)-single-packing of E_X such that the portal vertices of E_X are covered a_X times. Add this (c, K_d)-single-packing to $\mathcal{K}$. Since the gadgets are disjoint, the copies of K_d are also disjoint. The internal vertices of the gadgets are covered exactly c times. Moreover, the original vertices of G' are also covered c times, since each equality gadget simulates a clique. Hence $\mathcal{K}$ is a valid (c, K_d)-single-partition and therefore G' is a yes instance.

Now suppose that G' has a (c, K_d)-single-partition $\mathcal{K}$. Then, for each clique X in G, let a_X denote the number of times E_X covers its portal vertices. We construct a $\big(c, K_d\big)$-multi-packing $\mathcal{Z}$ by including each X exactly a_X times in $\mathcal{Z}$. Moreover, $\mathcal{Z}$ covers each vertex exactly c times, hence it is a valid $\big(c, K_d\big)$-multi-partition. Therefore, G is a yes-instance.

Running Time. First, we show that the pathwidth of G' is $p + \mathcal{O}(1)$. Take the path decomposition of G and for each clique X, let B_X denote the bag that contains the vertices of X. Note that since X is a clique, there exists such a bag. Moreover, we may assume each bag B_X is unique for each clique X, duplicating bags if necessary. Then, we simply add the vertices of E_X to the bag B_X, increasing its size at most by a constant. Therefore, we get a new path decomposition for G' where the size of a bag is at most $p + \mathcal{O}(1)$. Hence, it holds that $\mathbf{pw}(G') = p + \mathcal{O}(1)$.

Constructing the graph takes time polynomial in $n = V(G)$, and we also have $V(G') = n^{\mathcal{O}(1)}$. Therefore, the whole reduction takes time

$$\begin{aligned}(c+1-\varepsilon)^{\mathbf{pw}(G')} \cdot V(G')^b &= (c+1-\varepsilon)^{p+\mathcal{O}(1)} \cdot n^{\mathcal{O}(1)} \\ &= (c+1-\varepsilon)^{p} \cdot n^{\mathcal{O}(1)} \\ &= (c+1-\varepsilon)^{\mathbf{pw}(G)} \cdot n^{b'}\end{aligned}$$

where b' is a constant. ◀

The proof of Theorem 6 follows from Theorem 5 and Lemma 19.

References

1 Barış Can Esmer, Jacob Focke, Dániel Marx, and Paweł Rzążewski. Fundamental Problems on Bounded-Treewidth Graphs: The Real Source of Hardness. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2024.34.

2 Barış Can Esmer, Jacob Focke, Dániel Marx, and Paweł Rzążewski. List Homomorphisms by Deleting Edges and Vertices: Tight Complexity Bounds for Bounded-Treewidth Graphs. In Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman, editors, *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2024.39.

3 Radu Curticapean and D. Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In *SODA*, 2016. doi:10.1137/1.9781611974331.CH113.

4 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-21275-3.

5 Jacob Focke, Dániel Marx, Fionn Mc Inerney, Daniel Neuen, Govind S. Sankar, Philipp Schepper, and Philip Wellnitz. Tight Complexity Bounds for Counting Generalized Dominating Sets in Bounded-Treewidth Graphs Part II: Hardness Results. *ACM Transactions on Computation Theory*, page 3708509, January 2025. doi:10.1145/3708509.

6 P. Hell and D. G. Kirkpatrick. On generalized matching problems. *Information Processing Letters*, 12(1):33–35, February 1981. doi:10.1016/0020-0190(81)90073-9.

7 Michael Lampis. The Primal Pathwidth SETH. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 1494–1564. Society for Industrial and Applied Mathematics, January 2025. doi:10.1137/1.9781611978322.47.

8 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known Algorithms on Graphs of Bounded Treewidth Are Probably Optimal. *ACM Transactions on Algorithms*, 14(2):1–30, June 2018. doi:10.1145/3170442.

9 Karolina Okrasa, Marta Piecyk, and Pawel Rzążewski. Full complexity classification of the list homomorphism problem for bounded-treewidth graphs. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 74:1–74:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ESA.2020.74.

10 Karolina Okrasa and Pawel Rzążewski. Fine-grained complexity of the graph homomorphism problem for bounded-treewidth graphs. *SIAM J. Comput.*, 50(2):487–508, 2021. doi:10.1137/20M1320146.