





Bridging Treewidth and Clique-Width via Cograph-Modular-Treewidth

Václav Blažej  

University of Warsaw, Poland

Satyabrata Jana  

University of Warwick, Coventry, UK

M. S. Ramanujan  

University of Warwick, Coventry, UK

Peter Strulo  

University of Warwick, Coventry, UK

Abstract

Many classical graph problems – such as MAX CUT, CHROMATIC NUMBER, EDGE DOMINATING SET, and HAMILTONIAN CYCLE – are polynomial-time solvable on cographs, fixed-parameter tractable (FPT) when parameterized by treewidth, but $W[1]$ -hard when parameterized by clique-width. In contrast, GRAPH ISOMORPHISM is FPT parameterized by treewidth, but for clique-width it is known to be in XP; whether it is FPT or $W[1]$ -hard is open.

This reveals a sharp tractability gap between treewidth and clique-width. In this work, we propose a new structural graph parameter, \mathcal{C} -modular-treewidth, which lies between treewidth and clique-width. The parameter leverages modular decomposition and restricts modules to induce graphs from a fixed class \mathcal{C} (e.g., cographs or edgeless graphs). By exploiting true and false twins – a hallmark of cograph-like structure – our parameter allows the design of efficient algorithms for several hard problems beyond the reach of treewidth-based methods. In this work, we show that \mathcal{C} -modular-treewidth enables efficient solutions under suitable choices of \mathcal{C} , opening a new pathway in the parameterized complexity landscape between treewidth and clique-width. In particular we show that

- When parameterized by cograph-modular-treewidth, ISOMORPHISM admits an FPT algorithm, whereas CHROMATIC NUMBER remains $W[1]$ -hard.
- When parameterized by independent-modular-treewidth, HAMILTONIAN CYCLE and EDGE DOMINATING SET remain $W[1]$ -hard.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Treewidth, Clique-width, Cograph, FPT, $W[1]$ -hard

Digital Object Identifier 10.4230/LIPIcs.IPEC.2025.18

Funding *Václav Blažej*: Supported by project BOBR that received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 948057).

Satyabrata Jana: Supported by the Engineering and Physical Sciences Research Council (grant number EP/V044621/1).

M. S. Ramanujan: Supported by the Engineering and Physical Sciences Research Council (grant number EP/V044621/1).

Peter Strulo: Supported by the Engineering and Physical Sciences Research Council (grant number EP/V044621/1).



© Václav Blažej, Satyabrata Jana, M. S. Ramanujan, and Peter Strulo;
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Parameterized and Exact Computation (IPEC 2025).

Editors: Akanksha Agrawal and Erik Jan van Leeuwen; Article No. 18; pp. 18:1–18:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The field of parameterized complexity offers powerful tools for tackling computationally hard problems by identifying structural parameters that govern the complexity of input instances. Among these parameters, *treewidth* has played a central role due to its ability to capture the tree-like structure of graphs. Many graph problems that are NP-hard in general become tractable when parameterized by treewidth. This is because a wide range of problems that are solvable in polynomial time on trees can also be efficiently solved on graphs of bounded treewidth [5, 15, 16, 18, 19, 28, 32]. A foundational result in this area is that any graph problem expressible in monadic second-order logic (MSO) can be solved in linear time on graphs of bounded treewidth [2, 10, 11, 13]. This result captures a large family of natural problem such as HAMILTONIAN CYCLE, and optimization versions of the problems like VERTEX COVER, DOMINATING SET, FEEDBACK VERTEX SET, MAX CUT, CHROMATIC NUMBER and EDGE DOMINATING SET. For a comprehensive survey on treewidth and its algorithmic implications, we refer the reader to Bodlaender [3]. At the other end of the spectrum lies *clique-width*, a more general graph parameter introduced by Courcelle and Olariu [14]. Clique-width generalizes treewidth in the sense that every graph class of bounded treewidth also has bounded clique-width [8]. Importantly, for graph problems expressible in monadic second-order logic without edge set quantification (so-called MS_1 logic), Courcelle, Makowsky, and Rotics [12] showed that such problems are fixed-parameter tractable (FPT) when parameterized by clique-width. However, this generality comes at a cost. Many problems that are tractable for bounded treewidth become W[1]-hard or even para-NP-hard when parameterized by clique-width [20, 21, 22]. This reveals a crucial algorithmic gap and motivates the need for *intermediate parameters* – those that extend beyond treewidth but remain more structured than clique-width to retain algorithmic tractability.

In this work, we introduce such an intermediate parameter: *\mathcal{C} -modular-treewidth*. This parameter lies strictly between treewidth and clique-width and is inspired by the classical notion of modular decomposition, which partitions a graph into *modules* – vertex sets that share uniform adjacency to the rest of the graph. The idea is to restrict the induced subgraphs formed by these modules to belong to a fixed class \mathcal{C} , thereby allowing controlled generalization beyond treewidth. We consider the following five problems in our framework.

ISOMORPHISM

Input: Two graphs G_1 and G_2 .

Question: Are G_1 and G_2 isomorphic, i.e., is there a bijection $\phi: V(G_1) \rightarrow V(G_2)$ such that $uv \in E(G_1)$ if and only if $\phi(u)\phi(v) \in E(G_2)$?

MAX CUT

Input: A graph G , a positive integer r .

Question: Can we partition $V(G)$ into two subsets such that at least r edges have endpoints in both subsets?

CHROMATIC NUMBER

Input: A graph G , a positive integer r .

Question: Can we color the vertices of G using at most r colors such that no adjacent vertices receive the same color?

EDGE DOMINATING SET

Input: A graph G , a positive integer r .

Question: Is there an edge subset $X \subseteq E(G)$ of size at most r such that every edge in G is either in X or adjacent to an edge in X ?

HAMILTONIAN CYCLE

Input: A graph G .

Question: Is there a cycle that passes through every vertex of G exactly once?

A central case of interest is when \mathcal{C} consists of *cographs* (i.e., P_4 -free graphs) or *edgeless graphs*. Cographs, defined recursively via disjoint union and join operations, are well-known for their structural simplicity: they have clique-width at most two, admit cotree representations, and are closed under modular decomposition. Many difficult problems become polynomial-time solvable on cographs, including CHROMATIC NUMBER and HAMILTONIAN CYCLE. A key feature of cographs is the occurrences of *twin vertices* – pairs of vertices with identical neighborhoods (true twins if adjacent, false twins if not). These twins play a foundational role in modular decomposition and in our parameter. By leveraging the presence of twin sets, we define a decomposition that generalizes tree decompositions while enforcing that the modules belong to a restricted class \mathcal{C} .

Motivation and Applications. Our motivation is to bridge the algorithmic tractability gap between treewidth and clique-width. The same question was asked by Sæther and Telle [30] who introduced the graph parameter *sm-width*, which lies between treewidth and clique-width. Notably, some graph classes with unbounded treewidth, like distance-hereditary graphs, have bounded sm-width. In [30] the authors showed that MAX CUT, CHROMATIC NUMBER, HAMILTONIAN CYCLE, and EDGE DOMINATING SET are FPT when parameterized by sm-width.

Another parameter between treewidth and clique-width is called *modular-treewidth* and was introduced by Bodlaender and Jansen [4]. Notably Lampis [25] studied CHROMATIC NUMBER parameterized by modular-treewidth (mtw), showing an algorithm that decides k -coloring in $\mathcal{O}^*\left(\binom{k}{\lfloor k/2 \rfloor}^{\text{mtw}}\right)$ time; this is FPT when parameterized by $k + \text{mtw}$. Hegerfeld and Kratsch [23] recently studied connectivity problems on modular-width. Notably, they define *twinclass-treewidth* which is equivalent to ours \mathcal{C} -modular treewidth when \mathcal{C} is set to be (clique \cup edgeless), i.e., the class of all cliques and edgeless graphs. Several papers [25, 27, 29] use the name modular-treewidth to refer twinclass-treewidth.

Many natural problems are polynomial-time solvable on cographs, FPT when parameterized by treewidth, yet become intractable (e.g., W[1]-hard) under clique-width. Examples include MAX CUT, CHROMATIC NUMBER, EDGE DOMINATING SET, HAMILTONIAN CYCLE. Although ISOMORPHISM is FPT when parameterized by treewidth [26], for clique-width it is only known to be in XP [22] and whether it is FPT or W[1]-hard remains open.

W[1]-hardness reductions for many of these problems exhibit unbounded treewidth due to a simple obstruction, e.g. big cliques or bi-cliques, which additionally form modules in the constructed graph. This, together with their tractability on treewidth, raises the question of whether simple modules can be exploited to design FPT algorithms.

Our Results. In the following sections, we formally define \mathcal{C} -modular-treewidth, analyze its structural and algorithmic properties, and show its applicability to the above problems. Our results contribute to the broader goal of identifying and utilizing structural graph parameters that support efficient algorithms, lying between the extremes of treewidth and clique-width.

In Section 2.1 we clarify relations to other parameters and why results for modular-treewidth extend to our parameters. In particular, note that for CHROMATIC NUMBER the number of colors is naturally bounded for graphs of treewidth, which is also true for independent-modular-treewidth. On the other hand, the number of colors is unbounded in cographs, and we support untractability in this case by showing coloring is W[1]-hard by cograph-modular-treewidth.

■ **Table 1** Overview of results for the \mathcal{C} -modular-treewidth (-mt) parameter variants.

problem ↓ parameter →	treewidth	independent-mt	cograph-mt	clique-width
ISOMORPHISM	FPT [26]	FPT	FPT [Sec. 4]	XP [22]
MAX CUT	FPT [4]	FPT	FPT [4]	W[1]-hard [21]
CHROMATIC NUMBER	FPT [20]	FPT [25]	W[1]-h. [Sec. 5]	W[1]-hard [20]
EDGE DOMINATING SET	FPT [20]	W[1]-h. [Sec. 6]	W[1]-hard	W[1]-h. [20, 21]
HAMILTONIAN CYCLE	FPT [20]	W[1]-h. [Sec. 7]	W[1]-hard	W[1]-hard [20]

To simplify the notation in this paper, we employ the following convention while discussing problems. We use the following shorthand for width parameters: **tw** for treewidth, **cw** for clique-width, **imtw** for independent-modular-treewidth (i.e., \mathcal{C} -modular-treewidth with \mathcal{C} as edgeless graphs), and **cmtw** for cograph-modular-treewidth (i.e., \mathcal{C} -modular-treewidth with \mathcal{C} as cographs). For a graph problem Π and a width parameter $\mu \in \{\mathbf{tw}, \mathbf{cw}, \mathbf{imtw}, \mathbf{cmtw}\}$, when we represent it as Π/μ it indicates that the problem Π is considered with parameter μ . For example, ISOMORPHISM/**tw** indicates ISOMORPHISM parameterized by treewidth.

2 Preliminaries

We consider undirected unweighted simple graphs, unless stated otherwise. A graph G has n vertices $V(G)$ and m edges $E(G)$. For a set of vertices $X \subseteq V(G)$ we use $\binom{X}{2}$ for $\{uv \mid u, v \in X, u \neq v\}$, i.e., all the possible edges with both endpoints in X . For adjacency we use $u \sim v \iff uv \in E(G)$; similarly, $u \not\sim v$ denotes $uv \notin E(G)$. If one side (or both sides) of an adjacency relation contains a set $u \sim X$ we mean $u \sim x$ for every $x \in X$, similarly for $u \not\sim X$. We use \sim_G if the graph of the adjacency is not clear from the context. Let $G[X]$ for $X \subseteq V(G)$ denote the graph induced on vertices X , i.e., $G[X] = (X, E(G) \cap \binom{X}{2})$. Open neighborhood of u is denoted $N(u)$ and its closed neighborhood $N[u] = \{u\} \cup N(u)$. Two vertices u, v are called *true twins* if $N[u] = N[v]$ (implying $u \sim v$), and *false twins* if $N(u) = N(v)$ (implying $u \not\sim v$). We use $G_1 \approx G_2$ to denote that G_1 is isomorphic to G_2 .

► **Definition 1 (Module).** For a graph G , a subset of vertices $M \subseteq V(G)$ is said to be a *module* in G if for each $v \notin M$ either $v \sim M$ or $v \not\sim M$.

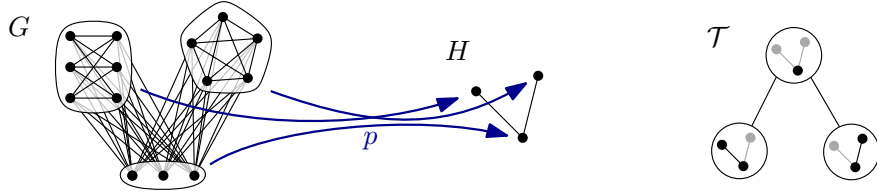
Treewidth. A *tree decomposition* of an undirected graph G is a pair $(T, \{X_t\}_{t \in V(T)})$ where T is a tree and $X_t \subseteq V(G)$ such that (i) for all edges $uv \in E(G)$ there exists a node $t \in V(T)$ such that $\{u, v\} \subseteq X_t$ and (ii) for all $v \in V(G)$ the subgraph induced by $\{t \in V(T) \mid v \in X_t\}$ is a non-empty tree. The *width* of a tree decomposition is $\max_{t \in V(T)} |X_t| - 1$. The *treewidth* of G is the minimum width of a tree decomposition of G .

► **Definition 2 (\mathcal{C} -modular tree-decomposition).** \mathcal{C} -modular tree-decomposition of a graph G is a triple (H, p, \mathcal{T}) where H is a graph, $p: V(G) \rightarrow V(H)$ is a surjective function such that $u \sim_H v \iff p^{-1}(u) \sim_G p^{-1}(v)$ and for each $u \in V(H)$ we have $G[p^{-1}(u)] \in \mathcal{C}$, and \mathcal{T} is a tree-decomposition of H . Width of a \mathcal{C} -modular tree-decomposition is defined as the width of the tree decomposition \mathcal{T} .

Observe that Definition 2 is equivalent to partitioning G into modules with each part inducing a graph from \mathcal{C} .

► **Definition 3 (\mathcal{C} -modular-treewidth).** \mathcal{C} -modular-treewidth (\mathcal{C} -mtw) of a graph G is the minimum k such that G has a \mathcal{C} -modular tree-decomposition of width k .

In other words, by the above definition we can construct G by taking a graph H of bounded treewidth and replace every vertex $u \in V(H)$ with a graph $p^{-1}(u) \in \mathcal{C}$, making all the replacement vertices adjacent to the prior neighbors of u , see Figure 1. Observe that for each $v \in V(H)$ every $p^{-1}(v)$ forms a module of G . We call H the *module graph* and the function p the *replacement function*.



■ **Figure 1** Example of a graph G which has \mathcal{C} -modular-treewidth 1 for \mathcal{C} containing complete graphs, independent sets, and complete bipartite graphs. \mathcal{C} -modular-treewidth is 1 because H has a tree-decomposition \mathcal{T} of width 1.

Note that if \mathcal{C} contains a single-vertex graph then every graph has a \mathcal{C} -modular tree-decomposition as we can set $H = G$, p to be identity and \mathcal{T} to be the tree-decomposition of G .

We now show basic properties of this general definition.

► **Observation 4.** \mathcal{S} -modular-treewidth, with \mathcal{S} being a class that consists of only one graph that contains a single vertex, is equivalent to treewidth.

Proof. With the class \mathcal{S} being this restricted the replacement function p cannot change the graph at all, hence, $G \approx H$ and we conclude that G has treewidth k if and only if it has \mathcal{S} -modular-treewidth k . ◀

► **Observation 5.** Any graph $G \in \mathcal{C}$ has \mathcal{C} -modular-treewidth 0.

Proof. As G is from the class \mathcal{C} it follows that it can be created from a single vertex u by replacing u with G . The graph with a single vertex has treewidth 0. ◀

► **Corollary 6.** Every edgeless graph G has $\text{imtw}(G) = 0$. Every cograph H has $\text{cmtw}(H) = 0$.

► **Lemma 7.** For a graph G and graph classes \mathcal{C} and \mathcal{D} such that $\mathcal{C} \subseteq \mathcal{D}$ we have \mathcal{D} -modular-treewidth of G is at most \mathcal{C} -modular-treewidth of G .

Proof. Consider a \mathcal{C} -modular tree-decomposition (H, p, \mathcal{T}) , then (H, p, \mathcal{T}) is also a \mathcal{D} -modular tree-decomposition because any part of p induces a graph of \mathcal{C} which is a subset of \mathcal{D} ; the other conditions remain unchanged and still hold. ◀

It is clear that the usefulness of \mathcal{C} -modular-treewidth depends heavily on the chosen graph class \mathcal{C} . One challenge with general graph classes is to be able to find a \mathcal{C} -modular tree-decomposition; one central class in our work is the class of cograph and to overcome this challenge we exploit the nice structure implied by the presence of twins in the class of cographs.

► **Definition 8** (Disjoint union, complement, join). Let $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ be two distinct graphs. The disjoint union of G_1 and G_2 is the graph $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. The complement of G_1 is $\bar{G}_1 = (V_1, \binom{V_1}{2} \setminus E_1)$. The join of graphs G_1 and G_2 is $(V_1 \cup V_2, E_1 \cup E_2 \cup \{uv \mid u \in V_1, v \in V_2\})$.

► **Definition 9** (Cographs). Cographs (complement reducible graphs) are defined recursively as follows: 1) a single vertex is a cograph; 2) the disjoint union of cographs is a cograph; 3) the join of cographs is a cograph.

As one can replace join with a sequence of 3 operations: complement, disjoint union, complement; it is not difficult to see that replacing (3) with taking complement of a cograph yields an equivalent (original) definition.

► **Definition 10** (Cotree [9]). *Cotree* $T(G)$ of a cograph G is a rooted tree with unordered children whose set of leaves is $V(G)$, each internal vertex (except possibly root) has degree at least 2, each internal vertex is labeled by its depth plus one modulo 2, and for each $u, v \in V(G)$ we have $u \sim v$ if and only if the least common ancestor of the leaves that represent u and v is labeled with 1.

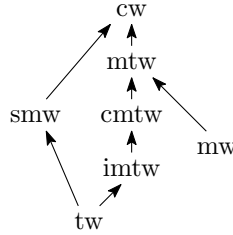
It was observed in [9] that each cograph has a unique representation by a cotree. We later use the fact that there is an algorithm to retrieve cotree and that tree isomorphism is a solved problem.

For a graph G , we denote its treewidth $\text{tw}(G)$, clique-width $\text{cw}(G)$, cograph-modular-treewidth $\text{cmtw}(G)$, and independent-modular-treewidth $\text{imtw}(G)$.

Parameterized Complexity. We refer to [15] for an introduction to the area. A parameterized problem (x, k) where x is a string over Σ and a parameter $k \in \mathbb{N}$ is fixed-parameter tractable (or FPT) if it can be solved in $f(k) \cdot |x|^{\mathcal{O}(1)}$ time for some computable function f . The standard notion of fixed-parameter intractability is $\text{W}[1]$ -hardness.

2.1 \mathcal{C} -modular-treewidth and other parameters

In this short section, we place the introduced parameters within the context of other parameters. Parameters we use solely in this section are modular-treewidth $\text{mtw}(G)$, modular-width $\text{mw}(G)$, and split-matching-width (sm-width) $\text{smw}(G)$. See the overview in Figure 2.



■ **Figure 2** Parameter hierarchy for treewidth (tw), sm-width (smw), modular-treewidth (mtw), modular-width (mw), clique-width (cw), independent-modular-treewidth (imtw), and cograph-modular-treewidth (cmtw). Arrow from A to B represents that if $A(G)$ is bounded, then $B(G)$ is bounded.

► **Lemma 11.** *For any graph G we have $\text{cmtw}(G) \leq \text{imtw}(G) \leq \text{tw}(G)$.*

Proof. Let us have graph classes: edgeless \mathcal{I} , cographs \mathcal{R} , graph with one vertex \mathcal{S} . Due to Observation 4 we know that treewidth is equivalent to \mathcal{S} -modular-treewidth. As $\mathcal{S} \subseteq \mathcal{I} \subseteq \mathcal{R}$ we can use Lemma 7 to conclude that $\text{cmtw}(G) \leq \text{imtw}(G) \leq \text{tw}(G)$. ◀

► **Lemma 12.** *For any graph G we have $\text{mtw}(G) \leq \text{cmtw}(G)$.*

Proof. Let $k = \text{cmtw}(G)$. Let (H, p, T) be the cograph-modular tree-decomposition that witnesses cograph-modular-treewidth of G is k . Observe, that one can alter this decomposition by taking modular decomposition of every co-graph and placing it in the stead of the cograph. This alteration results in a modular decomposition witnessing that $\text{mtw}(G) \leq k$ because every cograph will be decomposed into its cotree sturcture, resulting in tree nodes that are only cliques or independent sets that do not increase modular-treewidth of the decomposition. \blacktriangleleft

► **Lemma 13.** *mw is incomparable to cmtw , imtw , and tw .*

Proof. Note that due to Lemma 11 it suffices to show a graph class with bounded tw and unbounded mw , and another graph class with bounded mw and unbounded cmtw . We claim that the first graph class is a class of all paths – note that its treewidth is 1 as all paths are also trees; the modular-width cannot decompose any path because a path graph contains only trivial modules.

The second graph class we create inductively as follows. Let G_0 be a path on 4 vertices. Let G_{i+1} consists of 4 modules, each containing exactly G_i , and let the quotient graph over these modules be a path on 4 vertices. Let $\mathcal{G} = \cup_{i=0}^{\infty} \{G_i\}$. By design of \mathcal{G} , each graph $G \in \mathcal{G}$ has $\text{mw}(G) \leq 4$. On the other hand, notice that G contains no true nor false twins. We will later see in Lemma 19 that every non-trivial module of a cograph-modular tree-decomposition contains twins. Therefore, the decomposition of G is trivial (each module contains a single vertex) so the quotient graph contains a complete bipartite graph of unbounded size, hence, $\text{cmtw}(\mathcal{G})$ is unbounded. \blacktriangleleft

Later, we show that hardness of EDGE DOMINATING SET on independent-modular-treewidth; this problem is FPT on smw , hence we conclude the following.

► **Corollary 14.** *Unless $\text{FPT} = \text{W}[1]$, graph classes with bounded imtw do not necessarily have bounded smw .*

We did not find any reference for the relation between smw and mtw , nor a result that would prove incomparability of smw to our parameters. The above corollary shows one incomparability direction but it still remains to determine whether bounded smw implies bounded mtw , cmtw , or even imtw .

► **Lemma 15.** *There is a graph class with bounded smw and unbounded mtw .*

Proof. Consider a graph class \mathcal{G} that for each $n \geq 3$ contains a graph consisting of a clique K_n where each vertex of the clique $u \in V(K_n)$ has a private pendant leaf ℓ_u . We claim that \mathcal{G} has bounded smw but unbounded mtw . To show bounded smw it is not hard to find embedding of $V(G)$ into leaves of a ternary tree such that each subgraph induced by edges between parts of a bipartition that is implied by every edge of the ternary tree forms a complete bipartite graph. Unboundedness of mtw comes from the fact that every $G \in \mathcal{G}$ contains no non-trivial modules and so the first quotient graph must contain a clique K_n which lower bounds treewidth by $n - 1$. \blacktriangleleft

Due to Lemma 11, Lemma 15, and Corollary 14 we get the following corollary.

► **Corollary 16.** *Unless $\text{FPT} = \text{W}[1]$ we have that smw is incomparable to mtw , cmtw , and imtw .*

As mtw is defined as treewidth of quotient graphs within primal nodes of modular decomposition of the graph whereas mw is simply the maximum number of vertices within such nodes, implying $\text{mtw}(G) \leq \text{mw}(G)$ for any graph G .

The fact that bounded smw implies bounded cw is a combination of several facts: 1) clique-width is bounded by treewidth $\text{cw}(G) \leq 2^{\text{tw}(G)+1} + 1$ [14] – as each quotient graph of primal nodes has bounded treewidth one knows that there is an expression witnessing bounded clique-width of the quotient graph; 2) if we have an expression for a module, then we can change all vertices of the module to a single color and assume there is a single vertex instead – hence, we can easily attach expressions of modules to the expression of the parent quotient graph. Applying this recursively yields an expression of bounded clique-width.

3 Algorithm to find decompositions for cmtw and imtw

The usefulness of a parameter towards designing FPT algorithms is significantly impacted by our ability to retrieve a decomposition that witnesses that the parameter is bounded.

We first show an auxiliary lemma that shows structure of module graph of the decompositions. Then we focus on obtaining H and p of \mathcal{C} -modular tree-decompositions. We conclude with noting that \mathcal{T} of H can be found by showing how to obtain independent-modular tree-decomposition, which is quite simple, and then move to cograph-modular tree-decomposition for which we will need a number of auxiliary notions.

The below approaches are presented in order to have a self-contained contribution. We believe that using the linear-time algorithm for modular decomposition [31, 7] one can obtain independent- and cograph-modular-treewidth in linear time.

► **Lemma 17.** *For a graph G that has \mathcal{C} -modular-treewidth k there exists a \mathcal{C} -modular tree-decomposition (H, p, \mathcal{T}) of width k such that:*

1. *if \mathcal{C} is closed under disjoint unions, then H contains no false twins, and*
2. *if \mathcal{C} is closed under joins, then H contains no true twins.*

Proof. Towards a contradiction for (1), choose an H that contains the minimum number of false twins and suppose that this number is positive. Let $u, v \in V(H)$ be one such pair of false twins. The function p replaces each vertex of H with a graph from \mathcal{C} to obtain G . As u, v are false twins in H it follows that $\{u, v\}$ is a module in H . Let us alter the decomposition by removing v to create H' and observe that G can still be created from H' by creating $p' = p$ except for setting $(p')^{-1}(u)$ to be $G[p^{-1}(u) \cup p^{-1}(v)]$. This subgraph is in \mathcal{C} because both $p^{-1}(u)$ and $p^{-1}(v)$ are in \mathcal{C} and in G there is no edge between them as $uv \notin E(H)$, and because \mathcal{C} is closed under disjoint union. We removed v so the new decomposition $(H', p', \mathcal{T} - v)$ contains fewer twins, a contradiction.

We prove (2) by a similar contradiction; assuming \mathcal{C} is closed under joins and that u, v are true twins in H , we show that v can be removed because $G[p^{-1}(u) \cup p^{-1}(v)]$ is by $uv \in E(H)$ a join of the two graphs from \mathcal{C} . ◀

► **Theorem 18.** *Given a graph G on n vertices there is an algorithm that in $n^{\mathcal{O}(1)}$ time retrieves a graph H and a mapping $p: V(G) \rightarrow V(H)$ such that for every $u \in V(H)$ the vertices $p^{-1}(u)$ form an independent set of G .*

Proof. We aim to find maximal modules that induce independent sets in the graph. \mathcal{I} is closed under disjoint union so we invoke Lemma 17 and conclude that G has an \mathcal{I} -modular tree-decomposition (H, p, \mathcal{T}) where its module graph H contains no false twins. To find such H and p of such a decomposition we first retrieve the false twin relation by comparing open neighborhoods of all pairs of vertices; this implies a partition \mathcal{P} of $V(G)$.

From each part $P \in \mathcal{P}$ we pick an arbitrary single representative $u^P \in P$. Let the modular graph H be the graph induced on the representatives $H = G[\{u^P \mid P \in \mathcal{P}\}]$. The above can be easily done in $n^{\mathcal{O}(1)}$ time. For every $v \in V(G)$ let $p(v) = u^P$ where $P \in \mathcal{P}$ such that $v \in P$. As \mathcal{P} splits the graph into parts of equivalent neighborhoods, we have that the parts are modules of G . For any pair of vertices $v_1, v_2 \in V(G)$ we have $v_1 \sim_G v_2 \iff p(v_1) \sim_H p(v_2)$ by $N(p(v_1)) = N(v_1)$, $N(p(v_2)) = N(v_2)$, and H being an induced subgraph of G . ◀

To retrieve a cograph-modular tree-decomposition we use a very similar argument as for the independent-modular tree-decomposition; a number of notions needs to be introduced first.

Let us establish notations regarding partitions. Let each partition be represented as a family of sets that correspond to the parts, i.e., for a set $\{a, b, c, d, e, f\}$ we can have its partition $\{\{a, b, c\}, \{d, e\}, \{f\}\}$. The procedure described below creates nested partitions (partitions of partitions), resulting in e.g. $\{\{\{a, b, c\}\}, \{\{d, e\}, \{f\}\}\}$. Let *partition tree* $T(U)$ of a set U be a tree that corresponds to the bracket structure with labels of U replaced with \emptyset – defined inductively, if U is a label let $T(U)$ be a rooted tree with only the root vertex, otherwise $U = \{u_1, \dots, u_c\}$ and $T(U)$ is a rooted tree where the root has c children – roots of its subtrees $T(u_1), \dots, T(u_c)$.

Let G be a graph with cograph-modular-treewidth k . Let $\mathcal{P}_c(G)$ be a partition of $V(G)$ such that u, v belong to the same part if and only if $N[u] = N[v]$. Let $\mathcal{Q}_c(G)$ be the quotient graph over $\mathcal{P}_c(G)$, i.e., graph $\mathcal{Q}_c(G) = (\mathcal{P}_c(G), \{\{P, Q\} \mid P, Q \in \mathcal{P}_c(G), P \neq Q, u \in P, v \in Q, uv \in E(G)\})$. Similarly for open neighborhoods, let $\mathcal{P}_o(G)$ be a partition where $u, v \in P \in \mathcal{P}_o(G)$ if and only if $N(u) = N(v)$ and let $\mathcal{Q}_o(G)$ be the quotient graph over $\mathcal{P}_o(G)$. Note that taking the quotient graph results in a graph that is isomorphic to a graph where all but one vertex is removed from each part because the considered parts are always modules of the graph.

► **Lemma 19.** *Suppose G' is a cograph with at least 2 vertices and G' is an induced subgraph of G and $V(G')$ is a module of G , then G' contains a pair of vertices that are true or false twins in G .*

Proof. As G' is a cograph it contains a pair of vertices that in $G[V(G')]$ are true or false twins [9, Theorem 2]. Suppose G' contains $u, v \in V(G')$ that are true twins in $G[V(G')]$, so $N[u] \cap V(G') = N[v] \cap V(G')$. As u, v are part of the module their neighborhood in $V(G) \setminus V(G')$ is identical. We conclude that u, v are true twins in G as well. If G' contains false twins instead, the proof is identical. ◀

Now, we use the above lemma to find a tree decomposition of the module graph by first identifying twins of G , alternating between identifying true or false twins, and then invoking the algorithm to find the tree decomposition of the resulting graph.

► **Theorem 20.** *Given a graph G on n vertices there is an algorithm that in $n^{\mathcal{O}(1)}$ time retrieves a graph H and a mapping $p: V(G) \rightarrow V(H)$ such that for every $u \in V(H)$ the induced subgraph $G[p^{-1}(u)]$ is a cograph.*

Proof. We define a sequence of graphs G_0, G_1, \dots, G_{2q} where $G_0 = G$ and for each $i \in [q]$ the graph $G_{2i-1} = \mathcal{Q}_c(G_{2i-2})$ and $G_{2i} = \mathcal{Q}_o(G_{2i-1})$; let q be the minimum such that $G_{2q} = \mathcal{Q}_c(\mathcal{Q}_o(G_{2q+2}))$. Note that $q \leq n$ as by the definition for each $i \in [q]$ the graphs G_{2i} and G_{2i-2} are distinct; by the construction we have $|V(G_{2i})| < |V(G_{2i-2})|$. I.e., in the above we constructed a sequence of quotient graphs over alternating partitions \mathcal{P}_c and \mathcal{P}_o .

which ends at a point where the application does not change the graph, i.e., all vertices have distinct open neighborhoods and distinct closed neighborhoods. All of this takes at most $n^{\mathcal{O}(1)}$ time as $q \leq n$, and each step takes at most polynomial-time.

Note that due to Lemma 17 graph H should not contain twins; the above process gets rid of them. Moreover, due to Lemma 19, if there is a module that induces a cograph then the process eventually reduces it to a single vertex. We set $H = G_{2q}$ which is a module graph of G and p can be constructed according to Lemma 17 in a straight-forward manner. ◀

To finish, we apply the algorithm of Korhonen [24] which either returns a tree-decomposition \mathcal{T} of H of width $2k + 1$ or concludes that the treewidth of H is more than k in $2^{\mathcal{O}(k)} \cdot |V(H)|$ time. We return (H, p, \mathcal{T}) as the independent-modular tree-decomposition.

► **Corollary 21.** *Given an integer k and a graph G on n vertices there is an algorithm that retrieves an independent-modular tree-decomposition of G of width $2k + 1$ or determines that the independent-modular-treewidth of G is more than k in $n^{\mathcal{O}(1)} + 2^{\mathcal{O}(k)} \cdot n$ time.*

► **Corollary 22.** *Given an integer k and a graph G on n vertices there is an algorithm that retrieves a cograph-modular tree-decomposition of G of width $2k + 1$ or determines that the cograph-modular-treewidth of G is more than k in $n^{\mathcal{O}(1)} + 2^{\mathcal{O}(k)} \cdot n$ time.*

Note that in the proof of Theorem 20 the labels assigned to vertices correspond to cotrees of the particular modules. This could be utilized to retrieve cotrees immediately, but for simplicity we will retrieve the cotrees of $p^{-1}(u)$ for each $u \in V(H)$ independently using [6].

4 Isomorphism/cmtw is FPT

In short, to tackle ISOMORPHISM parameterized by **cmtw** we first retrieve the cograph-modular tree-decompositions. As every cograph is uniquely determined by its cotree (see Definition 10) we use the idea from the tree isomorphism checking algorithm to attach to each tree a unique number that represents its shape – two of these numbers are equal if and only if the two represented cotrees (and so their respective cographs) are isomorphic. Then, to each vertex v of the module graphs we attach a tree that represents the number for the cotree of $p^{-1}(v)$. These attached trees are shown not to occur in the module graph so we can safely use them to uniquely represent the cograph modules. In the end, we show that the attached trees are isomorphic if and only if the represented cographs are isomorphic. As attaching trees does not change treewidth of the graph it then suffices to check isomorphism of the two altered module graphs. Let us build these tools more formally now.

Let us first recall that isomorphism of rooted trees with unordered children.

► **Lemma 23** (see e.g. [1, Theorem 3.3]). *There is a procedure that assigns integers $\gamma: T \rightarrow [|T|]$ to each tree in a set of rooted trees T (with unordered children) so that two trees $t_1, t_2 \in T$ are isomorphic $t_1 \approx t_2$ if and only if their integers are equal $\gamma(t_1) = \gamma(t_2)$.*

Proof. The procedure assigns γ to every root of every subtree (i.e., every vertex of every tree) so that two subtrees are isomorphic if and only if their γ are equal. We can assume that the roots of all the trees in the set are children of a new dummy root so we can focus on processing a single tree. In essence, the vertices of the rooted tree are labelled bottom-up as follows. Each leaf gets label 0 and each inner vertex gets a label determined from the sorted labels of its children. More precisely, if label combination was never seen before it assigns the lowest unused \mathbb{N} ; hence, the maximum integer used for a label is no more than the total number of vertices. It follows that isomorphic subtrees get the same label, and that the used

labels are upper bound by the number of processed vertices. To compare two different trees one needs to preserve the label assigning function from one tree and use it to label the other tree – two trees are deemed isomorphic if they end up having the same-labelled root. ◀

We will use Lemma 23 to get integers for cotrees of cographs that make up modules of our graph. These numbers will be used as labels and we solve the labeled instance using transformation from the following lemma.

► **Lemma 24.** *Given a pair of graphs G, G' and a labeling function $\delta: V(G) \cup V(G') \rightarrow [n^{\mathcal{O}(1)}]$ there is an algorithm that in $n^{\mathcal{O}(1)}$ time transforms graphs G, G' into graphs \hat{G}, \hat{G}' so that $G \approx_\delta G'$, i.e., G has a δ -label-preserving isomorphism to G' if and only if $\hat{G} \approx \hat{G}'$, i.e., \hat{G} is isomorphic to \hat{G}' .*

Proof. Let us create \hat{G} from G by first subdividing all edges once and then to every vertex $u \in V(G)$ attaching $\delta(u) + 2$ new leaves. We create \hat{G}' from G' in the same way. This transformation runs in $n^{\mathcal{O}(1)}$ time. Due to the subdivision the transformation creates a bipartite graph. All of the new leaves are in one part of the bipartition. On the other hand, the old leaves (those in G) are not leaves in \hat{G} as to each of them we attached at least 2 new leaves. Note that the transformation is label agnostic and so if $G \approx_\delta G'$ then $\hat{G} \approx \hat{G}'$. In the other direction, assuming $\hat{G} \approx \hat{G}'$ we know that the parts with the leaves need to be mapped to each other. The number of adjacent leaves of a vertex is invariant under isomorphism and structure of the graph is clearly reflected in subdivision vertices, so it is straight-forward to lift the isomorphism to $G \approx_\delta G'$. ◀

► **Theorem 25.** *Given a pair of graphs G, G' and an integer k in $f(k) \cdot n^{\mathcal{O}(1)}$ time we can either decide ISOMORPHISM of G and G' or conclude that either G or G' has cograph-modular-treewidth more than k .*

Proof. By Corollary 22 we either get cograph-modular tree-decomposition (H, p, \mathcal{T}) of G and (H', p', \mathcal{T}') of G' , each of width at most $2k + 1$, or conclude that one of the graphs has cograph-modular-treewidth more than k . Then, for each vertex $u \in V(H)$ we take the induced subgraph $G[p^{-1}(u)]$ and retrieve its cotree C_u ; similarly we retrieve cotrees of all the vertices in G' . Now we create a labeling $\gamma: V(G) \cup V(G') \rightarrow \mathbb{N}$. Then, for every subgraph $\{G[p^{-1}(u)] \mid u \in V(H)\}$ we retrieve its cotree C_u [6]; similarly for H' we get cotrees C'_v . Now we run Lemma 23 for the set of all retrieved cotrees $\{C_u \mid u \in (V(G) \cup V(G'))\}$, let $\gamma(u) = \delta(C_u)$, i.e., we set $\gamma(u)$ to be the label that uniquely identifies the cotree C_u and subsequently also uniquely identifies the respective cograph. Note that γ has no values bigger than $|V(G)| + |V(G')|$. Next, we invoke Lemma 24 to transform H, H' to \hat{H}, \hat{H}' . Last, we use [26] to determine whether \hat{H} is isomorphic to \hat{H}' in $2^{\mathcal{O}(k^5 \log k)} \cdot (|V(\hat{H})| + |V(\hat{H}')|)^5$ time.

We claim that the answer to whether G is isomorphic to G' is equivalent to answering whether \hat{H} is isomorphic to \hat{H}' . Indeed, observe that Corollary 22 obtains the cograph-modular tree-decomposition in a label-agnostic way so $G \approx G'$ if and only if the bijection that witnesses $H \approx H'$ maps to each other vertices that represent the same underlying cographs, i.e., a bijection $\alpha: H \rightarrow H'$ must satisfy $p^{-1}(u) \approx p'^{-1}(\alpha(u))$ for every $u \in V(H)$. We modeled this requirement by using bijection between cographs and cotrees, then using Lemma 23 to get the auxiliary labeling γ , and requiring γ -preserving isomorphism. Last, we used a transformation Lemma 24 which produces an equivalent instance. ◀

5 CHROMATIC NUMBER/cmtw is $W[1]$ -hard

A *coloring* of a graph G is an assignment $c: V(G) \rightarrow \mathbb{N}$ of a positive integer (color) to each vertex of G . The coloring c is *proper* if adjacent vertices receive distinct colors. The *chromatic number* $\chi(G)$ of a graph G is the smallest number of colors of a proper coloring of G . The CHROMATIC NUMBER problem asks whether $\chi(G) \leq r$ of G . Note that CHROMATIC NUMBER is not expressible in monadic second order logic. However, for every fixed r , checking whether $\chi(G) \leq r$ can be expressed in monadic second order logic even without edge set quantification. Since graphs of tree-width at most t are $t+1$ colorable, this implies that CHROMATIC NUMBER can be solved in linear time on graph classes of bounded treewidth. Moreover, this problem admits FPT algorithm when parameterized by tw , although it becomes $W[1]$ -hard when parameterized by cw [20]. In this section, we show that the CHROMATIC NUMBER problem is $W[1]$ -hard when the parameter is cograph-modular-treewidth. This hardness persists even when every cograph in the underlying decomposition is a clique. However, it admits an FPT algorithm when parameterized by independent-modular-treewidth [25]. In summary, we present the following result.

► **Theorem 26.** CHROMATIC NUMBER/cmtw is $W[1]$ -hard.

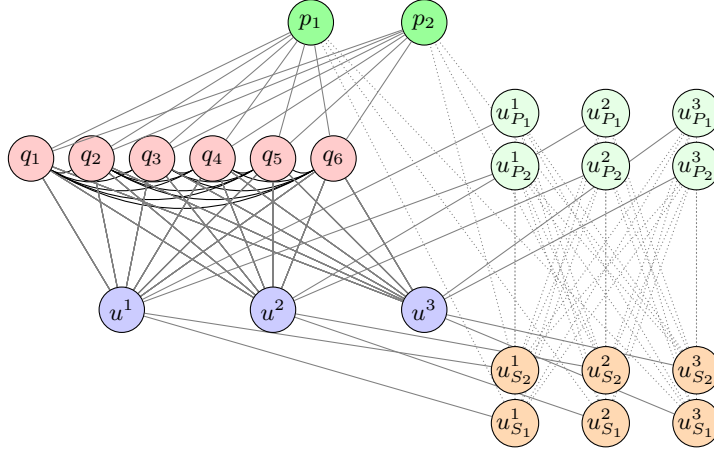
Proof. Our proof is based on a construction similar to that of [20], with specific modifications – namely, the addition of further gadget structures, which establishes the $W[1]$ -hardness of the CHROMATIC NUMBER when parameterized by clique-width. We give a reduction from EQUITABLE COLORING/ $\text{tw} + r$ which is known to be $W[1]$ -hard [18]. In the EQUITABLE COLORING problem one is given a graph G on n vertices and integer r and asked whether G can be properly r -colored in such a way that the number of vertices in any two color classes differs by at most 1 (such coloring is called an *equitable r -coloring*). Notice that if n is divisible by r this implies that all color classes must contain the same number of vertices. In our reduction, we will assume that in the instance we reduce from, n is divisible by r . For a justification of this assumption, if r does not divide n we can add a clique of size $r - (n - \lfloor n/r \rfloor r)$ to G . We reduce from the exact version of EQUITABLE COLORING, that is, the version where we are looking for an equitable coloring of G with exactly r colors.

Reduction. Given an input (G, r) to EQUITABLE COLORING, we construct an instance (H, r') of CHROMATIC NUMBER as follows (see Figure 3). Let $n = |V(G)|$ and $r' = r + nr$. We start with a copy G' of G . We add a clique Q of size nr and connect every vertex of Q to every vertex of G' by an edge. We add a clique P of size r' to H , called *palette*. The set P is partitioned into $r + 1$ parts: $P = P^M \cup P_1 \cup P_2 \cup \dots \cup P_r$, where $|P^M| = r$, $|P_i| = n$ (for $i \in [r]$). The vertices of P^M are denoted by p_1, p_2, \dots, p_r . For each vertex $v \in P_M$ and each vertex $w \in Q$, we add the edge vw . For each $i \in [r]$, we add a set S_i of n vertices which contains copies of all vertices of G . For each vertex $u \in V(G)$ and each $i \in [r]$, we assign vertices $u_{P_i} \in P_i$ and $u_{S_i} \in S_i$. For each vertex $u \in V(G)$ and each $i \in [r]$, we add the edge uu_{P_i} . For every vertex $u \in V(G)$ and each $i \in [r]$, we make u_{S_i} adjacent to u and the entire palette P except for u_{P_i} and p_i . For each $i \in [r]$, we add a clique C_i of $\frac{n(r-1)}{r}$ vertices and make every vertex of C_i adjacent to all the vertices of S_i and the entire palette P except for P_i . Now we show the correctness of the reduction.

► **Claim 27.** If G has an equitable r -coloring ψ , then H has a proper r' -coloring φ .

► **Claim 28.** If H has a proper r' -coloring φ , then G has an equitable r -coloring ψ .

► **Claim 29.** $\text{cmtw}(H) \leq 2r \cdot \text{tw}(G) + r + 2$



■ **Figure 3** Illustration of the construction in Theorem 26 for $n = 3$, $r = 2$.

Claims 27, 28 and 29 together give a parameterized reduction from **EQUITABLE COLORING** for $\text{tw} + r$ into **CHROMATIC NUMBER/cmtw**. Moreover, as per our construction, this hardness holds even when every cograph in the underlying decomposition is a clique or an independent set. This completes the proof of Theorem 26. ◀

6 EDGE DOMINATING SET/imtw is $\mathbf{W}[1]$ -hard

An *edge dominating set* of a graph G is a set $X \subseteq E(G)$ such that every edge of G is either in X or adjacent to at least one edge of X . **EDGE DOMINATING SET** problem asks to find an edge dominating set of size at most k . This problem admits an FPT algorithm parameterized by tw by Courcelle's Theorem since it is MSO_2 expressible but it is $\mathbf{W}[1]$ -hard when parameterized by cw [20]. We show that this problem remains $\mathbf{W}[1]$ -hard when parameterized by cmtw .

6.1 Exact Saturated Dominating Set

We first introduce a problem from which we will reduce. A *capacitated graph* is a pair (G, c) , where G is a graph and $c: V(G) \rightarrow \mathbb{N}$ is a capacity function such that $c(v) \in [\deg(v)]$ for every vertex $v \in V(G)$ (sometimes we simply say that G is a capacitated graph if the capacity function is clear from the context). A set $S \subseteq V(G)$ is called a *capacitated dominating set* if there is a domination mapping $f: V(G) \setminus S \rightarrow S$ which maps every vertex in $V(G) \setminus S$ to one of its neighbors in S in such a way that the total number of vertices mapped by f to any vertex $v \in S$ does not exceed its capacity $c(v)$, i.e., $|f^{-1}(v)| \leq c(v)$. For a vertex $v \in S$ we say that vertices in $f^{-1}(v)$ are dominated by v . In **CAPACITATED DOMINATING SET**, given a capacitated graph (G, c) and a positive integer k , the objective is to check whether G has a capacitated dominating set of size at most k for G . In the **EXACT CAPACITATED DOMINATING SET** problem, the size of such a set needs to be *exactly* k . It is known that both **CAPACITATED DOMINATING SET** and **EXACT CAPACITATED DOMINATING SET** are $\mathbf{W}[1]$ -hard when parameterized by the treewidth of the input graph and the solution size k [17, 20].

For a capacitated graph (G, c) , a capacitated dominating set $S \subseteq V(G)$ is called a *saturated dominating set* if there is a domination mapping f such that $|f^{-1}(v)| = c(v)$ for each vertex $v \in S$. Below we define the **EXACT SATURATED DOMINATING SET** problem.

EXACT SATURATED DOMINATING SET

Input: A capacitated graph (G, c) , a positive integer k .

Question: Does G have a saturated dominating set with *exactly* k vertices?

It is known that EXACT SATURATED DOMINATING SET/**cw** is $W[1]$ -hard [20]. We expand the aforementioned result and show that EXACT SATURATED DOMINATING SET/**tw** is $W[1]$ -hard.

► **Theorem 30.** *The EXACT SATURATED DOMINATING SET/**tw** is $W[1]$ -hard.*

6.2 Edge Dominating Set

In this section we show the following result.

► **Theorem 31.** *EDGE DOMINATING SET/**intw** is $W[1]$ -hard.*

Proof. We give a reduction from EXACT SATURATED DOMINATING SET/**tw**. Our proof follows a construction similar to that in [20], with minor modifications, which establishes the $W[1]$ -hardness of the EDGE DOMINATING SET problem when parameterized by clique-width. We start with a description of an auxiliary gadget.

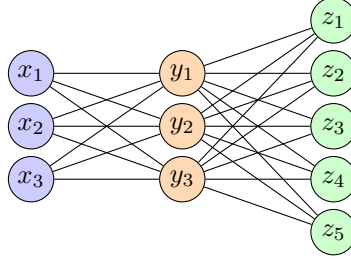
Auxiliary gadget. Let $s \leq t$ be positive integers. We construct a graph $F_{s,t}$ with the vertex set $\{x_1, \dots, x_s, y_1, \dots, y_s, z_1, \dots, z_t\}$ and edges $x_i y_j$ for $1 \leq i, j \leq s$, and $y_i z_j$ for $1 \leq i \leq s, 1 \leq j \leq t$. Basically, we have a complete bipartite graph between the x_i 's and the y_j 's as well as between the y_i 's and the z_j 's. The vertices z_1, \dots, z_t are called the *roots* of $F_{s,t}$, see Figure 4.

Reduction. Let (G, c) be a capacitated graph with the vertex set $\{u_1, \dots, u_n\}$, and k be a positive integer. We introduce three vertex sets $\{v_1, \dots, v_n\}$, $\{v'_1, \dots, v'_n\}$, and $\{w_1, \dots, w_n\}$. Then for each $i \in [n]$, we add the edges (v_i, w_i) and (v_i, v'_i) . Now for every vertex u_i , we introduce a set U_i with $c(u_i)$ vertices. For every edge $u_i u_j \in E(G)$, we connect all vertices of U_i with v_j and all vertices of U_j to v_i . We introduce vertex sets $\{a_1, \dots, a_n\}$ and $\{b_1, \dots, b_n\}$. Vertices a_i are made adjacent to all vertices of U_i , w_i , and b_i . For every vertex b_i we add a set R_i of $c(u_i) + 1$ vertices and make b_i adjacent to all the vertices of R_i . Then for each R_i , $i \in [n]$, we add vertex sets P_i and Q_i both of size $|R_i|$ and we make a complete bipartite graph between the vertices of P_i and Q_i as well as P_i and R_i . Let $P = P_1 \cup P_2 \cup \dots \cup P_n$. We denote the obtained graph by G' , see Figure 5. Let $r = \sum_{v \in V(G)} c(v)$. Finally, we introduce three copies of $F_{s,t}$: (i) a copy of $F_{n-k,n}$ with roots $\{a_1, \dots, a_n\}$, (ii) a copy of $F_{k,n}$ with roots $\{b_1, \dots, b_n\}$, and (iii) a copy of $F_{n,r+n}$ with roots in P . Let H be this final resulting graph.

► **Claim 32.** Any set of s edges incident with vertices y_1, \dots, y_s forms an edge dominating set in $F_{s,t}$. Furthermore, let G be a graph obtained by the union of $F_{s,t}$ with some other graph H such that $V(F_{s,t}) \cap V(H) = \{z_1, \dots, z_t\}$. Then every edge dominating set of G contains at least s edges from $F_{s,t}$.

The proof of Claim 32 follows from the fact that every edge dominating set includes at least one edge from $E(y_i)$ for $i \in \{1, \dots, s\}$ where $E(y_i)$ denotes the set of edges with one endpoint at y_i .

► **Claim 33.** If the capacitated graph (G, c) has an exact saturated dominating set of size k then H has an edge dominating set of cardinality $3n + r$.



■ **Figure 4** The graph $F_{3,5}$.

▷ **Claim 34.** If H has an edge dominating set of cardinality at most $3n + r$ then (G, c) has an exact saturated dominating set of size k .

▷ **Claim 35.** $\text{imtw}(H) \leq 7\text{tw}(G) + 6$

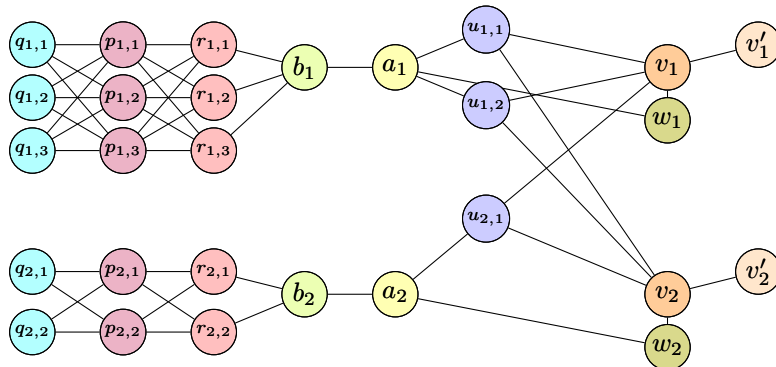
Claims 33, 34 and 35 together give a parameterized reduction from EXACT SATURATED DOMINATING SET/ tw to EDGE DOMINATING SET/ imtw . This completes the proof of Theorem 31. ◀

7 HAMILTONIAN CYCLE/ imtw is $\text{W}[1]$ -hard

A *Hamiltonian cycle* in a graph G is defined as a cycle that includes every vertex of G . The HAMILTONIAN CYCLE problem asks finding such a cycle in G . Although the problem can be solved using an FPT algorithm with the parameterization by tw (which is expressible in MSO_2), it becomes $\text{W}[1]$ -hard when parameterized by cw [20]. We demonstrate that the problem also remains $\text{W}[1]$ -hard with the cmtw parameterization, and even $\text{W}[1]$ -hard when using imtw as a parameter. In this section, we present the following finding.

► **Theorem 36.** HAMILTONIAN CYCLE/ imtw is $\text{W}[1]$ -hard.

Proof. In the proof of $\text{W}[1]$ -hardness of HAMILTONIAN CYCLE/ cw by Fomin et al. [20], the constructed gadget contains large independent modules. These modules have the property that, if each were contracted into a single vertex, the resulting graph would have bounded treewidth. We exhibit this property towards proving Theorem 36. For the sake of completeness, we present the same construction below, omitting the proof of the reduction (as it remains identical), but we include a proof demonstrating that the underlying graph has bounded treewidth.



■ **Figure 5** Illustration of the construction in Theorem 31 with $n = 2$, $c(u_1) = 2$, $c(u_2) = 1$.

We give a reduction from CAPACITATED DOMINATING SET/tw which is known to be W[1]-hard [20]. We start with descriptions of auxiliary gadgets.

First auxiliary gadget L_1 . The first auxiliary gadget is the graph L_1 with the vertex set $\{x, y, z, a, b, c, d\}$ and the edge set $\{xa, ab, bc, cd, dy, bz, cz\}$. Let P_1 be the path $xabzcdy$ and $P_2 = xabcdy$.

Second auxiliary gadget L_2 . The second auxiliary gadget is the graph L_2 with the vertex set $\{x, y, z, s, t, a, b, c, d, e, f, g, h\}$. Initially, the edges $\{xa, ab, bz, cz, cd, dy, se, ef, fb, ch, hg, gt\}$ in are added to its edge set. Then an x - y path $xw_1 \cdots w_9y$ of length 10 is added, and edges $fw_3, w_1w_6, w_4w_9, w_7h$ are included in the set of edges. Let $P = xabzcdy$, $R_1 = sefbaxw_1w_2 \cdots w_9ydchgt$, and $R_2 = sefw_3w_2w_1w_6w_5w_4w_9w_8w_7hgt$.

Reduction. Let (G, c) be a capacitated graph with the vertex set $\{v_1, \dots, v_n\}$ and m edges, and let k be a positive integer. The construction of H is as follows:

- For every vertex v_i , four vertices a_i, b_i, c_i , and w_i are introduced, and the vertices b_i and c_i are joined by $c(v_i) + 1$ paths of length two. Let C_i denote the set of middle vertices of these paths, and $X_i = C_i \cup \{a_i, b_i, c_i\}$. Then a copy L_i^2 of the graph L_2 with $z = w_i$ is added, and vertices x and y of this gadget are joined by edges to a_i and b_i , respectively. By s_i and t_i we denote the vertices s and t , respectively, of L_i^2 . The paths corresponding to P in L_i^2 is called P^i .
- For every edge $v_i v_j \in E(G)$ with $i < j$, a copy L_{ij}^2 of L_2 is attached with $z = w_j$ and vertices x and y made adjacent to all the vertices of C_i . The vertices corresponding to s and t are called s_{ij} and t_{ij} , respectively, in L_{ij}^2 . Furthermore, let x_{ij} and y_{ij} denote the vertices corresponding to x and y , respectively, in L_{ij}^2 . The paths corresponding to P , R_1 , and R_2 are called P^{ij} , R_1^{ij} , and R_2^{ij} , respectively, in L_{ij}^2 .
- Next we add two vertices g and h which are joined by $\sum_{i=1}^n (c(v_i) + 4) + n + 2m + 1$ paths of length two. Let Y be the set of middle vertices of these paths. All vertices s_i, t_i, s_{ij} , and t_{ij} are joined by edges with all vertices of Y . For every vertex $r \in X_i, i \in [n]$, a copy L_r^1 of L_1 with $z = r$ is attached and the vertices x and y of this gadget are joined to all vertices of Y . We let x_r and y_r denote the vertices corresponding to x and y , respectively, in L_r^1 . Similarly, P_1^r and P_2^r denote paths in L_r^1 corresponding to P_1 and P_2 , respectively.
- Finally, we add $k + 1$ vertices, namely $\{p_1, \dots, p_{k+1}\}$, and make them adjacent to all the vertices $\{a_i, c_i \mid 1 \leq i \leq n\}$ and to g and h .

▷ **Claim 37** ([20, Lemma 10]). (G, c) has a capacitated dominating set of size at most k if and only if H has a Hamiltonian cycle.

▷ **Claim 38.** $\text{imtw}(H) \leq 24\text{tw}(G)$

Claims 37 and 38 together give a parameterized reduction from CAPACITATED DOMINATING SET/tw to EDGE DOMINATING SET/imtw. This completes the proof of Theorem 36. ◀

8 Conclusion

We introduced \mathcal{C} -modular-treewidth, a new graph parameter generalizing classical treewidth by leveraging structure from a fixed graph class \mathcal{C} . We established the parameterized complexity of several problems: both HAMILTONIAN CYCLE and EDGE DOMINATING SET are W[1]-hard for independent-modular-treewidth, while GRAPH ISOMORPHISM is FPT for cograph-modular-treewidth. We also showed that CHROMATIC NUMBER is W[1]-hard for cograph-modular-treewidth. Future work includes studying these and related problems under clique-modular-treewidth and exploring how different choices of the base class \mathcal{C} affect the algorithmic tractability of classic problems.

References

- 1 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1st edition, 1974.
- 2 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991. doi:10.1016/0196-6774(91)90006-K.
- 3 Hans L. Bodlaender. A partial k -arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.
- 4 Hans L. Bodlaender and Klaus Jansen. On the complexity of the maximum cut problem. *Nord. J. Comput.*, 7(1):14–31, March 2000.
- 5 Hans L. Bodlaender and Arie M.C.A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008. doi:10.1093/COMJNL/BXM037.
- 6 Anna Bretscher, Derek Corneil, Michel Habib, and Christophe Paul. A simple linear time LexBFS cograph recognition algorithm. *SIAM Journal on Discrete Mathematics*, 22(4):1277–1296, 2008. doi:10.1137/060664690.
- 7 Derek G. Corneil, Michel Habib, Christophe Paul, and Marc Tedder. A recursive linear time modular decomposition algorithm via LexBFS, 2024. arXiv:0710.3901.
- 8 Derek G. Corneil and Udi Rotics. On the relationship between clique-width and treewidth. *SIAM J. Comput.*, 34(4):825–847, 2005. doi:10.1137/S0097539701385351.
- 9 D.G. Corneil, H. Lerchs, and L. Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981. doi:10.1016/0166-218X(81)90013-5.
- 10 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
- 11 Bruno Courcelle. The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues. *RAIRO Theor. Informatics Appl.*, 26:257–286, 1992. doi:10.1051/ITA/1992260302571.
- 12 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.
- 13 Bruno Courcelle and Mohamed Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theor. Comput. Sci.*, 109(1&2):49–82, 1993. doi:10.1016/0304-3975(93)90064-Z.
- 14 Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77–114, 2000. doi:10.1016/S0166-218X(99)00184-5.
- 15 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015. doi:10.1007/978-3-319-21275-3.
- 16 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Trans. Algorithms*, 18(2):17:1–17:31, 2022. doi:10.1145/3506707.
- 17 Michael Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Capacitated domination and covering: A parameterized perspective. In *Parameterized and Exact Computation, Third International Workshop, IWPEC*, volume 5018 of *Lecture Notes in Computer Science*, pages 78–90. Springer, 2008. doi:10.1007/978-3-540-79723-4_9.
- 18 Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Inf. Comput.*, 209(2):143–153, 2011. doi:10.1016/J.IC.2010.11.026.
- 19 Jacob Focke, Dániel Marx, and Paweł Rżazewski. Counting list homomorphisms from graphs of bounded treewidth: Tight complexity bounds. *ACM Trans. Algorithms*, 20(2):11, 2024. doi:10.1145/3640814.

- 20 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.
- 21 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.
- 22 Martin Grohe and Pascal Schweitzer. Isomorphism testing for graphs of bounded rank width. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1010–1029. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.66.
- 23 Falko Hegerfeld and Stefan Kratsch. Tight algorithms for connectivity problems parameterized by modular-treewidth. In Daniël Paulusma and Bernard Ries, editors, *Graph-Theoretic Concepts in Computer Science - 49th International Workshop, WG 2023, Fribourg, Switzerland, June 28-30, 2023, Revised Selected Papers*, volume 14093 of *Lecture Notes in Computer Science*, pages 388–402. Springer, 2023. doi:10.1007/978-3-031-43380-1_28.
- 24 Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 184–192, 2022. doi:10.1109/FOCS52979.2021.00026.
- 25 Michael Lampis. Finer tight bounds for coloring on clique-width. *SIAM Journal on Discrete Mathematics*, 34(3):1538–1558, 2020. doi:10.1137/19M1280326.
- 26 Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth. *SIAM J. Comput.*, 46(1):161–189, 2017. doi:10.1137/140999980.
- 27 Stefan Mengel. Parameterized compilation lower bounds for restricted cnf-formulas. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 3–12. Springer, 2016. doi:10.1007/978-3-319-40970-2_1.
- 28 Karolina Okrasa, Marta Piecyk, and Pawel Rżazewski. 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*, volume 173 of *LIPICs*, pages 74:1–74:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ESA.2020.74.
- 29 Daniël Paulusma, Friedrich Slivovsky, and Stefan Szeider. Model counting for CNF formulas of bounded modular treewidth. *Algorithmica*, 76(1):168–194, 2016. doi:10.1007/S00453-015-0030-X.
- 30 Sigve Hortemo Sæther and Jan Arne Telle. Between treewidth and clique-width. *Algorithmica*, 75(1):218–253, 2016. doi:10.1007/S00453-015-0033-7.
- 31 Marc Tedder, Derek G. Corneil, Michel Habib, and Christophe Paul. Simpler linear-time modular decomposition via recursive factorizing permutations. In *Automata, Languages and Programming*, pages 634–645, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-70575-8_52.
- 32 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *Algorithms – ESA 2009*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.