Measuring what Matters: A Hybrid Approach to Dynamic Programming with Treewidth

Eduard Eiben
Department of Informatics, University of Bergen, Bergen, Norway
eduard.eiben@uib.no

Robert Ganian
Vienna University of Technology, Vienna, Austria
rganian@gmail.com

Thekla Hamm
Vienna University of Technology, Vienna, Austria
thekla.hamm@tuwien.ac.at

O-joung Kwon
Department of Mathematics, Incheon National University, Korea
ojoungkwon@inu.ac.kr

Abstract

We develop a framework for applying treewidth-based dynamic programming on graphs with "hybrid structure", i.e., with parts that may not have small treewidth but instead possess other structural properties. Informally, this is achieved by defining a refinement of treewidth which only considers parts of the graph that do not belong to a pre-specified tractable graph class. Our approach allows us to not only generalize existing fixed-parameter algorithms exploiting treewidth, but also fixed-parameter algorithms which use the size of a modulator as their parameter. As the flagship application of our framework, we obtain a parameter that combines treewidth and rank-width to obtain fixed-parameter algorithms for CHROMATIC NUMBER, HAMILTONIAN CYCLE, and MAX-CUT.

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

Keywords and phrases Parameterized complexity, treewidth, rank-width, fixed-parameter algorithms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2019.42

Funding E. Eiben was supported by Pareto-Optimal Parameterized Algorithms (ERC Starting Grant 715744). R. Ganian acknowledges support by the Austrian Grant Agency (FWF, project P31336 NFPC), and is also affiliated with FI MUNI, Czech Republic. O. Kwon was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education (No. NRF-2018R1D1A1B07050294).

1 Introduction

Over the past decades, the use of structural properties of graphs to obtain efficient algorithms for NP-hard computational problems has become a prominent research direction in computer science. Perhaps the best known example of a structural property that can be exploited in this way is the tree-likeness of the inputs, formalized in terms of the decomposition-based structural parameter *treewidth* [35]. It is now well known that a vast range of fundamental problems admit so-called *fixed-parameter* algorithms parameterized by the treewidth of the input graph – that is, can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ on n-vertex graphs of treewidth k (for some computable function f). We say that such problems are FPT parameterized by treewidth.

On the other hand, dense graphs are known to have high treewidth and hence require the use of different structural parameters; the classical example of such a parameter tailored to dense graphs is *clique-width* [8]. Clique-width is asymptotically equivalent to the structural parameter *rank-width* [34], which is nowadays often used instead of clique-width due to a number of advantages (rank-width is much easier to compute [26] and can be used to design more efficient fixed-parameter algorithms than clique-width [21, 22]). While rank-width (or, equivalently, clique-width) dominates[1] treewidth and can be used to "lift" fixed-parameter algorithms designed for treewidth to well-structured dense graphs for a number of problems, there are also important problems which are FPT parameterized by treewidth but W[1]-hard (and hence probably not FPT) parameterized by rank-width. The most prominent examples of such problems are Chromatic Number [18], Hamiltonian Cycle [18], and Max-Cut [19].

Another generic type of structure used in algorithmic design is based on measuring the size of a *modulator* (i.e., a vertex deletion set) [5] to a certain graph class. Basic examples of parameters based on modulators include the vertex cover number (a modulator to edgeless graphs) [16] and the feedback vertex set number (a modulator to forests)[3]. For dense graphs, modulators to graphs of rank-width 1 have been studied [13, 30], and it is known that for every constant c one can find a modulator of size at most k to graphs of rank-width c (if such a modulator exists) in time $f(k) \cdot n$ [29]. However, the algorithmic applications of such modulators have remained largely unexplored up to this point.

Our Contribution. We develop a class of *hybrid parameters* which combines the foremost advantages of treewidth and modulators to obtain a "best-of-both-worlds" outcome. In particular, instead of measuring the treewidth of the graph itself or the size of a modulator to a graph class $\mathcal{H}$, we consider the treewidth of a (torso of a) modulator to $\mathcal{H}$. This parameter, which we simply call $\mathcal{H}$-treewidth, allows us to lift previously established tractability results for a vast number of problems from treewidth and modulators to a strictly more general setting. As our first technical contribution, we substantiate this claim with a meta-theorem that formalizes generic conditions under which a treewidth-based algorithm can be generalized to $\mathcal{H}$-treewidth; the main technical tool for the proof is an adaptation of *protrusion replacement* techniques [2, Section 4].

As the flagship application of $\mathcal{H}$-treewidth, we study the case where $\mathcal{H}$ is the class $\mathcal{R}_c$ of graphs of rank-width at most c (an arbitrary constant). $\mathcal{R}_c$-treewidth hence represents a way of lifting treewidth towards dense graphs that lies "between" treewidth and rank-width. We note that this class of parameters naturally incorporates a certain scaling trade-off: $\mathcal{R}_c$-treewidth dominates $\mathcal{R}_{c-1}$-treewidth for each constant c, but the runtime bounds for algorithms using $\mathcal{R}_c$-treewidth are worse than those for $\mathcal{R}_{c-1}$-treewidth.

Our first result for $\mathcal{R}_c$-treewidth is a fixed-parameter algorithm for computing the parameter itself. We then develop fixed-parameter algorithms for Chromatic Number, Hamiltonian Cycle and Max-Cut parameterized by $\mathcal{R}_c$-treewidth; moreover, in 2 out of these 3 cases the parameter dependencies of our algorithms are essentially tight. These algorithms represent generalizations of:

1. classical fixed-parameter algorithms parameterized by treewidth [12],
2. polynomial-time algorithms on graphs of bounded rank-width [22], and
3. (not previously known) fixed-parameter algorithms parameterized by modulators to graph classes of bounded rank-width.

[1] Parameter α dominates parameter β if for each graph class with bounded β, α is also bounded.

The main challenge for all of these problems lies in dealing with the fact that some parts of the graph need to be handled using rank-width based techniques, while for others we use treewidth-based dynamic programming. We separate these parts from each other using the notion of *nice $\mathcal{H}$-tree decompositions*. The algorithm then relies on enhancing the known dynamic programming approach for solving the problem on treewidth with a subroutine that not only solves the problem on the part of the graph outside of the modulator, but also serves as an interface by supplying appropriate records to the treewidth-based dynamic programming part of the algorithm. At its core, each of these subroutines boils down to solving an "extended" version of the original problem parameterized by the size of a modulator to $\mathcal{R}_c$; in particular, each subroutine immediately implies a fixed-parameter algorithm for the respective problem when parameterized by a modulator to constant rank-width. To give a specific example for such a subroutine, in the case of Chromatic Number one needs to solve the problem parameterized by a modulator to $\mathcal{R}_c$ where the modulator is furthermore precolored.

To avoid any doubt, we make it explicitly clear that the runtime of all of our algorithms utilizing $\mathcal{R}_c$-treewidth has a polynomial dependency on the input where the degree of this polynomial depends on c (as is necessitated by the W[1]-hardness of the studied problems parameterized by rank-width).

Related Work. Previous works have used a combination of treewidth with *backdoors*, a notion that is closely related to modulators, in order to solve non-graph problems such as Constraint Satisfaction [25], Boolean Satisfiability [24] and Integer Programming [23]. Interestingly, the main technical challenge in all of these papers is the problem of computing the parameter, while using the parameter to solve the problem is straightforward. In the algorithmic results presented in this contribution, the situation is completely reversed: the main technical challenge lies in developing the algorithms (and most notably the subroutines) for solving our targeted problems. Moreover, while the aforementioned three papers focus on solving a single problem, here we aim at identifying and exploiting structural properties that can be used to solve a wide variety of graph problems.

Other parameters which target inputs with hybrid structure include *sm-width* [36] and well-structured modulators [14]. It is not difficult to show that these are different (both conceptually and factually) from $\mathcal{R}_c$-treewidth.

2 Preliminaries

For $i \in \mathbb{N}$, let $[i]$ denote the set $\{1, \ldots, i\}$. All graphs in this paper are simple and undirected. We refer to the standard textbook [11] for basic graph terminology. For $S \subseteq V(G)$, let $G[S]$ denote the subgraph of G induced by S. For $v \in V(G)$, the set of neighbors of v in G is denoted by $N_G(v)$ (or $N(v)$ when G is clear from the context). For $A \subseteq V(G)$, let $N_G(A)$ denote the set of vertices in $G - A$ that have a neighbor in A. For a vertex set A, an *A-path* is a path whose endpoints are contained in A and all the internal vertices are contained in $G - A$.

A set M of vertices in a graph G is called a *modulator* to a graph class $\mathcal{H}$ if $G - M \in \mathcal{H}$. The operation of *collapsing* a vertex set X, denoted $G \circ X$, deletes X from the graph and adds an edge between vertices $u, v \in V(G - X)$ if $uv \notin E(G)$ and there is an u-v path with all internal vertices in $G[X]$. We assume that the reader is familiar with *parameterized complexity* [9, 12], notably with notions such as FPT, W[1], treewidth and Courcelle's Theorem.

Rank-width. For a graph G and $U, W \subseteq V(G)$, let $\boldsymbol{A}_G[U,W]$ denote the $U \times W$-submatrix of the adjacency matrix over the two-element field GF(2), i.e., the entry $a_{u,w}$, $u \in U$ and $w \in W$, of $\boldsymbol{A}_G[U,W]$ is 1 if and only if $\{u,w\}$ is an edge of G. The *cut-rank* function ρ_G of a graph G is defined as follows: For a bipartition (U,W) of the vertex set $V(G)$, $\rho_G(U) = \rho_G(W)$ equals the rank of $\boldsymbol{A}_G[U,W]$.

A *rank-decomposition* of a graph G is a pair (T,μ) where T is a tree of maximum degree 3 and $\mu : V(G) \to \{t \mid t \text{ is a leaf of } T\}$ is a bijective function. For an edge e of T, the connected components of $T - e$ induce a bipartition (X,Y) of the set of leaves of T. The *width* of an edge e of a rank-decomposition (T,μ) is $\rho_G(\mu^{-1}(X))$. The *width* of (T,μ) is the maximum width over all edges of T. The *rank-width* of G, $\mathbf{rw}(G)$ in short, is the minimum width over all rank-decompositions of G. We denote by $\mathcal{R}_i$ the class of all graphs of rank-width at most i. A *rooted* rank-decomposition is obtained from a rank-decomposition by subdividing an arbitrarily chosen edge, and the newly created vertex is called the *root*.

Unlike clique-width, rank-width can be computed exactly by a fixed-parameter algorithm (which also outputs a corresponding rank-decomposition) [26].

Monadic Second-Order Logic. Counting Monadic Second-Order Logic (CMSO_1) is a basic tool to express properties of vertex sets in graphs. The syntax of CMSO_1 includes logical connectives $\wedge, \vee, \neg, \Leftrightarrow, \Rightarrow$, variables for vertices and vertex sets, quantifiers $\exists, \forall$ over these variables, and the relations $a \in A$ where a is a vertex variable and A is a vertex set variable; $\mathrm{adj}(a,b)$, where a and b are vertex variables and the interpretation is that a and b are adjacent; equality of variables representing vertices and sets of vertices; $\mathrm{Parity}(A)$, where A is a vertex set variable and the interpretation is that $|A|$ is even.

The CMSO_1 Optimization problem is defined as follows:

CMSO_1-OPT	
Instance:	A graph G, a CMSO_1 formula $\phi(A)$ with a free set variable A, and opt $\in \{\min, \max\}$.
Task:	Find an interpretation of the set A in G such that G models $\phi(A)$ and A is of minimum/maximum (depending on opt) cardinality.

From the fixed-parameter tractability of computing rank-width [26], the equivalence of rank-width and clique-width [34] and Courcelle's Theorem for graphs of bounded clique-width [7] it follows that:

▶ **Fact 1** ([21]). CMSO_1-OPT *is* FPT *parameterized by* $\mathbf{rw}(G) + |\phi|$*, where* G *is the input graph and* ϕ *is the* CMSO_1 *formula.*

3 $\mathcal{H}$-Treewidth

The aim of $\mathcal{H}$-treewidth is to capture the treewidth of a modulator to the graph class $\mathcal{H}$. However, one cannot expect to obtain a parameter with reasonable algorithmic applications by simply measuring the treewidth of the graph *induced* by a modulator to $\mathcal{H}$ – instead, one needs to measure the treewidth of a so-called *torso*, which adds edges to track how the vertices in the modulator interact through $\mathcal{H}$. To substantiate this, we observe that HAMILTONIAN CYCLE would become NP-hard even on graphs with a modulator that (1) induces an edgeless graph, and (2) is a modulator to an edgeless graph, and where (3) each connected component outside the modulator has boundedly many neighbors in the modulator [1].

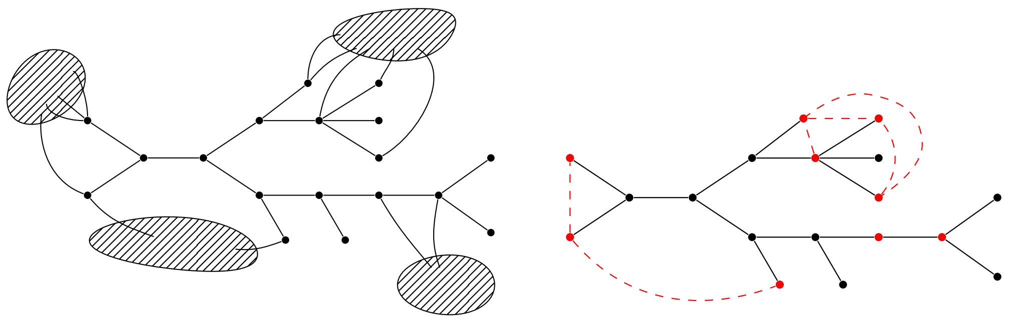

Figure 1 *Left:* A graph G with a tree as a modulator to $\mathcal{H}$ (the part in $\mathcal{H}$ is depicted hatched). *Right:* The corresponding $\mathcal{H}$-torso.

The notion of a *torso* has previously been algorithmically exploited in other settings [23, 25, 33], and its adaptation is our first step towards the definition of $\mathcal{H}$-treewidth (see also Figure 1).

▶ **Definition 1** ($\mathcal{H}$-Torso). *Let G be a graph and $X \subseteq V(G)$. For a graph class $\mathcal{H}$, $G \circ X$ is an* $\mathcal{H}$-torso *of G if each connected component C of $G[X]$ satisfies $C \in \mathcal{H}$.*

▶ **Definition 2** ($\mathcal{H}$-Treewidth). *The $\mathcal{H}$-treewidth of a graph G is the minimum treewidth of an $\mathcal{H}$-torso of G. We denote the $\mathcal{H}$-treewidth of G by $\mathbf{tw}_{\mathcal{H}}(G)$.*

Typically, we will want to consider a graph class $\mathcal{H}$ for which certain problems are polynomial-time tractable. Hence, we will assume w.l.o.g. that $(\emptyset, \emptyset) \in \mathcal{H}$. From the definition we easily observe that $\mathbf{tw}_{\mathcal{H}}(G) \leq \mathbf{tw}(G)$ for every G and $\mathcal{H}$.

3.1 Nice $\mathcal{H}$-Tree-Decompositions

Just like for tree decompositions, we can also define a canonical form of decompositions which has properties that are convenient when formulating dynamic programs using $\mathcal{H}$-treewidth. Intuitively, a *nice $\mathcal{H}$-tree decomposition* behaves like a nice tree decomposition on the torso graph (see points 1-3), with the exception that the neighborhoods of the collapsed parts must occur as special *boundary* leaves (see points 4-5).

▶ **Definition 3** (Nice $\mathcal{H}$-Tree-Decomposition). *A nice $\mathcal{H}$-tree decomposition of a graph G is a triple $(X, T, \{B_t \mid t \in V(T)\})$ where $X \subseteq V(G)$ such that $G \circ X$ is an $\mathcal{H}$-torso, $(T, \{B_t \mid t \in V(T)\})$ is a rooted tree decomposition of $G \circ X$, and:*

1. *Every node in T has at most two children.*
2. *If a node t has children $t_1 \neq t_2$, then $B_t = B_{t_1} = B_{t_2}$ and we call t a* join node.
3. *If a node t has exactly one child t', then either (a) there exists $x \in V(G) \setminus B_{t'}$ such that $B_t = B_{t'} \cup \{x\}$ and we call t an* introduce node, *or (b) there exists $x \in V(G) \setminus B_t$ such that $B_{t'} = B_t \cup \{x\}$ and we call t a* forget node.
4. *If a node t is a leaf, then (a) $|B_t| = 1$ and we call t a* simple leaf, *or (b) $B_t = N(C)$ for some connected component C of $G[X]$ and we call t a* boundary leaf.
5. *For each connected component C of $G[X]$ there is a unique leaf t with $B_t = N(C)$.*

An illustration of a nice $\mathcal{H}$-tree decomposition showcasing how it differs from a nice tree decomposition is provided in Figure 2; in line with standard terminology for treewidth, we call the sets B_t *bags*. The width of a nice $\mathcal{H}$-tree decomposition is simply the width of $(T, \{B_t \mid t \in V(T)\})$. Given a node t in a nice $\mathcal{H}$-tree decomposition T, we let Y_t be the set of all vertices

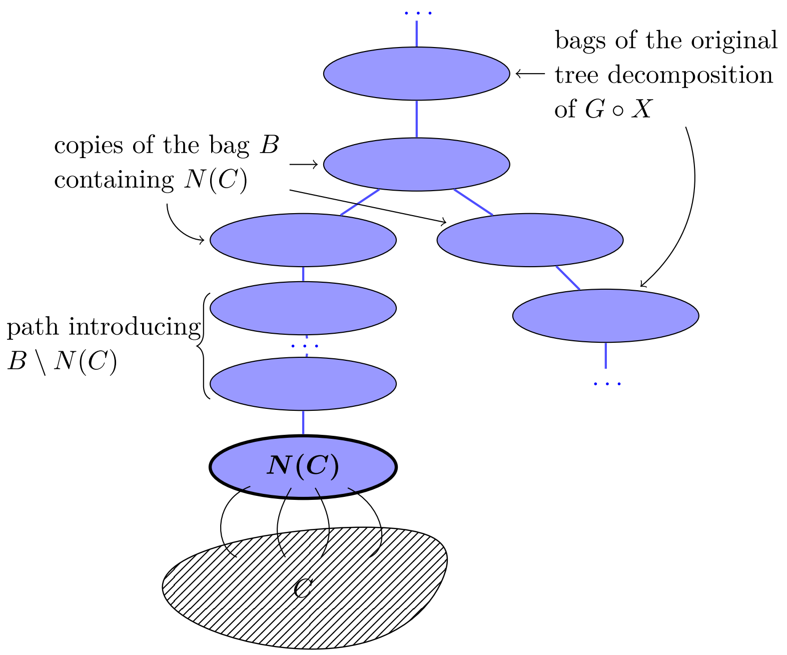

Figure 2 Part of a nice $\mathcal{H}$-tree-decomposition (blue) including a boundary leaf (bold) and a connected component C (hatched) of X.

contained in the bags of the subtree rooted at t, i.e., $Y_t = B_t \cup \bigcup_{p \text{ is separated from the root by } t} B_p$. It is possible to show that computing a nice $\mathcal{H}$-tree decomposition of bounded width can be reduced to finding an appropriate $\mathcal{H}$-torso (this is because a nice $\mathcal{H}$-tree decomposition can be obtained straightforwardly from a nice tree decomposition of the torso).

Hence, we can state the problem of computing a decomposition as follows:

$\mathcal{H}$-Treewidth	*Parameter:* k
Instance:	A graph G, an integer k.
Task:	Find an $\mathcal{H}$-torso U of G such that $\mathbf{tw}(U) \leq k$, or correctly determine that no such $\mathcal{H}$-torso exists.

An Algorithmic Meta-Theorem. Before proceeding to the flagship application of $\mathcal{H}$-treewidth where $\mathcal{H}$ is the class of graphs of bounded rank-width, here we give a generic set of conditions that allow fixed-parameter algorithms for problems parameterized by $\mathcal{H}$-treewidth. Specifically, we consider graph problems that are *finite-state* [6] or have *finite integer index* [2, 4, 20]. Informally speaking, such problems only transfer a limited amount of information across a small separator in the input graph and hence can be solved "independently" on both sides of such a separator. Since these notions are only used in this section, we provide concise definitions below.

First of all, we will need the notion of boundaried graphs and gluing. A graph $\bar{G}$ is called *t-boundaried* if it contains t distinguished vertices identified as $b_1^G, \ldots, b_t^G$. The *gluing operation* $\oplus$ takes two t-boundaried graphs $\bar{G}$ and $\bar{H}$, creates their disjoint union, and then alters this disjoint union by identifying the boundaries of the two graphs (i.e. by setting $b_i^G = b_i^H$ for each $i \in [t]$).

Consider a decision problem $\mathcal{P}$ whose input is a graph. We say that two t-boundaried graphs $\bar{C}$ and $\bar{D}$ are *equivalent*, denoted by $\bar{C} \sim_{\mathcal{P},t} \bar{D}$, if for each t-boundaried graph $\bar{H}$ it holds that $\bar{C} \oplus \bar{H} \in \mathcal{P}$ if and only if $\bar{D} \oplus \bar{H} \in \mathcal{P}$. We say that $\mathcal{P}$ is *finite-state* (or *FS*, in brief) if, for each $t \in \mathbb{N}$, $\sim_{\mathcal{P},t}$ has a finite number of equivalence classes.

Next, consider a decision problem $\mathcal{Q}$ whose input is a graph and an integer. In this case we say that two t-boundaried graphs $\bar{C}$ and $\bar{D}$ are equivalent (denoted by $\bar{C} \sim_{\mathcal{Q},t} \bar{D}$) if there exists an *offset* $\delta(\bar{C}, \bar{D}) \in \mathbb{Z}$ such that for each t-boundaried graph $\bar{H}$ and each $q \in \mathbb{Z}$: $(\bar{C} \oplus \bar{H}, q) \in \mathcal{Q}$ if and only if $(\bar{D} \oplus \bar{H}, q + \delta(\bar{C}, \bar{D})) \in \mathcal{Q}$. We say that $\mathcal{Q}$ has *finite integer index* (or is *FII*, in brief) if, for each $t \in \mathbb{N}$, $\sim_{\mathcal{Q},t}$ has a bounded number of equivalence classes.

We note that a great number of natural graph problems are known to be FS or FII. For instance, all problems definable in Monadic Second Order logic are FS [2, Lemma 3.2], while examples of FII problems include Vertex Cover, Independent Set, Feedback Vertex Set, Dominating Set, to name a few [20]. We say that a FS or FII problem $\mathcal{P}$ is *efficiently extendable* on a graph class $\mathcal{H}$ if there is a fixed-parameter algorithm (parameterized by t) that takes as input a t-boundaried graph $\bar{G}$ such that the boundary is a modulator to $\mathcal{H}$ and outputs the equivalence class of $\bar{G}$ w.r.t. $\sim_{\mathcal{P},t}$.

▶ **Theorem 4.** *Let $\mathcal{P}$ be a FS or FII graph problem and $\mathcal{H}$ be a graph class such that (1) $\mathcal{P}$ is efficiently extendable on $\mathcal{H}$, (2) $\mathcal{P}$ is* FPT *parameterized by treewidth, and (3) $\mathcal{H}$-*treewidth *is* FPT*. Then $\mathcal{P}$ is* FPT *parameterized by $\mathcal{H}$-treewidth.*

Proof Sketch. We can solve $\mathcal{P}$ as follows. First of all, we use Point (3) to compute an $\mathcal{H}$-torso $G \circ X$ of treewidth k, where k is the $\mathcal{H}$-treewidth. Next, for each connected component C of $G[X]$, we use the fact that $C \in \mathcal{H}$ and Point (1) to compute the equivalence class of the boundaried graph $\bar{H} = \overline{G[C \cup N(C)]}$ where the boundary is $N(C)$. Note that since $\mathbf{tw}(G \circ X) \leq k$ and $N(C)$ forms a clique in $G \circ X$, $|N(C)| \leq k$ and hence this step takes only fixed-parameter time. Next, we use a brute-force enumeration argument to compute a bounded-size representative of the equivalence class of $\bar{H}$, and replace $\bar{H}$ with this representative. After doing this exhaustively, we obtain a graph G' of bounded treewidth, for which we can invoke Point (2). ◀

4 $\mathcal{R}_c$-Treewidth

This section focuses on the properties of $\mathcal{R}_c$-treewidth, a hierarchy of graph parameters that represent our flagship application of the generic notion of $\mathcal{H}$-treewidth.

Comparison to Known Parameters. It follows from the definition of $\mathcal{H}$-treewidth that $\mathcal{R}_c$-treewidth dominates treewidth (for every $c \in \mathbb{N}$). Similarly, it is obvious that $\mathcal{R}_c$-treewidth dominates the size of a modulator to $\mathcal{R}_c$ (also for every $c \in \mathbb{N}$). The following lemma shows that, for every fixed c, $\mathcal{R}_c$-treewidth is dominated by rankwidth.

▶ **Lemma 5.** *Let $c \in \mathbb{N}$. If $\mathbf{tw}_{\mathcal{R}_c}(G) = k$ then $\mathbf{rw}(G) \leq c + k + 1$.*

Proof. Let the $\mathcal{R}_c$-treewidth of G be witnessed by some nice $\mathcal{R}_c$-tree-decomposition $(X, T, \{B_t \mid t \in V(T)\})$ of width k.

We can obtain a rank-decomposition (T', μ) of G from $(X, T, \{B_t \mid t \in V(T))\}$ as follows:

For vertices v of $G \circ X$ such that there is no leaf node $t \in V(T)$ with $B_t = \{v\}$, let $t \in V(T)$ be a forget node with child t' such that $B_t \cup \{v\} = B_{t'}$. Turn t' into a join node by introducing t_1, t_2 with $B_{t_1} = B_{t_2} = B_{t'}$ as children of t', attaching the former child of t' to t_1 and a new leaf node t_v with $B_{t_v} = \{v\}$ below t_2. Note that this preserves the fact that for any $v \in V(G) \setminus X$, $T[\{u \in V(T) \mid v \in B_u\}]$ is a tree. Now we can choose for each $v \in V(G \circ X)$ some $\mu(v) \in V(T)$ such that $B_t = \{v\}$. This defines an injection from $V(G \circ X)$ to the leaves of T. However not every leaf of T is mapped to by μ. On one hand there are the boundary leaf nodes, below which we will attach subtrees to obtain a rank-decomposition of G. On the other hand there may be $v \in V(G \circ X)$ for which the choice of $\mu(v)$ was not unique, i.e. there is $t \neq \mu(v)$ with $B_t = \{v\}$. For all such v and t we delete all nodes on the root-t-path in T that do not lie on a path from the root to a vertex in $\{\mu(w) \mid w \in V(G \circ X)\} \cup \{t' \mid t' \text{ boundary leaf node}\}$. This turns μ into an injection from $V(G \circ X)$ to the leaves of T, that is surjective on the non-boundary leaf nodes.

Next, we extend (T, μ) to a rank-decomposition of G by proceeding in the following way for each connected component C of $G[X]$:
Let $t_C \in V(T)$ be the boundary leaf node with $B_t = N(C)$. Since $\mathbf{rw}(C) \leq c$, we find a rank-decomposition (T_C, μ_C) of C with width at most c. Attach T_C below t_C.
Let T' be the tree obtained by performing these modifications for all connected components C. Consider the rank-decomposition of G given by

$$\left(T', v \mapsto \begin{cases} \mu(v) & \text{if } v \in V(G \circ X) \\ \mu_C(v) & \text{if } v \in C \text{ for some } C \text{ as above} \end{cases}\right).$$

We show that its width is at most $c + k + 1$. Any edge e of T' is of one of the following types:

- e corresponds to an edge already contained in T: Then e induces a bipartition (X_G, Y_G) of $V(G)$. Fix $t \in V(T)$ to be the vertex in which e starts.
  Let $x \in X_G$ and $y \in Y_G$ be such that $xy \in E(G)$. Observe that e does not separate neighbors in X as these lie within the same connected component of $G[X]$ whose rank-decomposition is, by construction, attached completely within one of the two subtrees of T' separated by e. So, we consider $x \in V(G \circ X)$. If $y \in V(G \circ X)$ then x and y occur together in some bag of the original tree of the tree decomposition, and as remarked earlier this is still the case modified tree. This implies that at least one of $\{x, y\}$ must be present in B_t. If $y \notin V(G \circ X)$, by the construction of T', y corresponds to a leaf $t_y \in V(T')$ in a subtree attached to T rooted at $t_C \in V(T)$ with $B_{t_C} = N(C) \ni x$ and t_C and t_y are not separated by the removal of e. Also, x corresponds to a leaf $t_x \in V(T)$ which is in the subtree of $T' - e$ not containing t_y, i.e. the subtree not containing t_C. This means any subtree containing t_C and t_x also contains t, and since $(T, \{B_t \mid t \in V(T)\})$ is a tree-decomposition and $x \in B_{t_x} \cap B_{t_C}$ this means $x \in B_t$.
  In both cases at least one of $\{x, y\}$ is in B_t. Since this argument applies for every edge crossing the bipartition (X_G, Y_G), it follows that $\boldsymbol{A}_G[X_G, Y_G]$ may only contain “1” entries in rows and columns that correspond to the vertices in B_t. Since $|B_t| \leq k + 1$, it holds that $\boldsymbol{A}_G[X_G, Y_G]$ can be converted into a zero matrix by deleting at most $k + 1$ rows plus columns, which is a sufficient condition for $\boldsymbol{A}_G[X_G, Y_G]$ having rank at most $k + 1$.
- e corresponds to an edge in a rank-decomposition of some connected component C of $G[X]$: Then e induces a bipartition (X_G, Y_G) of $V(G)$ and a bipartition (X_C, Y_C), where $X_C = X_G \cap C$ and $Y_C = Y_G \cap C$, of C. Since vertices of C are only connected to vertices in $N(C)$ outside of C and $N(C) \subseteq B_t$ for some $t \in V(T)$, we have $\rho_G(X_G) \leq \rho_C(X_C) + |N(C)| \leq c + k + 1$.
- e corresponds to an edge connecting the rank-decomposition of some connected component C of $G[X]$ to T: Then the bipartition induced is $(C, V(G) \setminus C)$ and as $N(C) \subseteq B_t$ for some $t \in V(T)$ $\rho_G(C) \leq |N(C)| \leq k + 1$. ◀

Next, we compare $\mathcal{R}_c$-treewidth to Telle and Saether's *sm-width* [36].

▶ **Lemma 6.** *$\mathcal{R}_c$-treewidth and sm-width are incomparable.*

Computing $\mathcal{R}_c$-Treewidth. Our aim here is to determine the complexity of computing our parameters, i.e., finding a torso of small treewidth. Obtaining such a torso is a base prerequisite for our algorithms. We formalize the problem below.

$\mathcal{R}_c$-Treewidth	*Parameter:* k
Instance:	A graph G, an integer k.
Task:	Find a $\mathcal{R}_c$-torso U of G such that $\mathbf{tw}(U) \leq k$, or correctly determine that no such $\mathcal{R}_c$-torso exists.

▶ **Lemma 7.** *$\mathcal{R}_c$-Treewidth is* FPT*.*

5 Algorithms Exploiting Modulators to $\mathcal{R}_c$

For a problem $\mathcal{P}$ to be FPT parameterized by $\mathcal{R}_c$-treewidth, $\mathcal{P}$ must necessarily be FPT parameterized by treewidth and also FPT parameterized by the size of a modulator to $\mathcal{R}_c$. However, it is important to note that the latter condition is not sufficient; indeed, one can easily invent artificial problems that are defined in a way which make them trivial in both of the cases outlined above, but become intractable (or even undecidable) once parameterized by $\mathcal{R}_c$-treewidth. That is, after all, why we need the notion of *efficient extendability* in Theorem 4.

Hence, in order to develop fixed-parameter algorithms for Chromatic Number, Hamiltonian Cycle and Max-Cut parameterized by $\mathcal{R}_c$-treewidth, we first need to show that they are not only FPT parameterized by the size of a modulator to $\mathcal{R}_c$, but they are also efficiently extendable. Such a result would be sufficient to employ Theorem 4 together with Lemma 7 in order to establish the desired fixed-parameter tractability results. That is also our general aim in this section, with one caveat: in order to give explicit and tight upper bounds on the parameter dependency of our algorithms, we provide algorithms that solve generalizations of Chromatic Number, Hamiltonian Cycle and Max-Cut parameterized by the size of a modulator to $\mathcal{R}_c$, whereas it will become apparent in the next section that these generalizations precisely correspond to the records required by the treewidth-based dynamic program that will be used in the torso. In other words, the efficient extendability of our problems on $\mathcal{R}_c$ is not proved directly but rather follows as an immediate consequence of our proofs in this section and the correctness of known treewidth-based algorithms.

Chromatic Number. In Chromatic Number, we are given a graph G and asked for the smallest number $\chi(G)$ such that the vertex set of G can be properly colored using $\chi(G)$ colors, i.e., the smallest number $\chi(G)$ such that $V(G)$ can be partitioned into $\chi(G)$ independent sets. Our aim in this section is to solve a variant of Chromatic Number on graphs with a k-vertex modulator X to $\mathcal{R}_c$ where X is precolored:

$\mathcal{R}_c$-Precoloring Extension	*Parameter:* k
Instance:	A graph G, a k-vertex modulator $X \subseteq V(G)$ to $\mathcal{R}_c$ and a coloring of X.
Task:	Compute the smallest number of colors required to extend the coloring of X to a proper coloring of G.

▶ **Theorem 8.** *$\mathcal{R}_c$-Precoloring Extension can be solved in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$.*

Proof Sketch. Let G be a graph together with a k-vertex modulator X to $\mathcal{R}_c$ and a proper coloring of X by colors $[k]$; let col_X be the set of colors assigned to at least one vertex in X. For disjoint vertex sets S and Q of G, two vertices v and w in S are *twins* with respect to Q if $N(v) \cap Q = N(w) \cap Q$. A *twin class* of S with respect to Q is a maximal subset of S that consists of pairwise twins w.r.t. Q.

Our starting point is a rooted rank-decomposition (T, μ) of $G - X$ of width at most c, which may be computed in time $\mathcal{O}(n^3)$ [26]. On a high level, our algorithm will apply dynamic programming along (T, μ) where it will group colors together based on which twin classes they occur in (analogously as in the XP algorithm for Chromatic Number parameterized by clique-width, due to Kobler and Rotics [31]), but keep different (more detailed) records about the at most k colors used in X.

For each $t \in V(T)$, let S_t be the set of all vertices that are assigned to the descendants of t, and let $G_t := G[S_t \cup X]$. By our definition of (T, μ), recall that $\rho_{G-X}(S_t) \leq c$ and that there are at most $z = 2^c$ twin classes of S_t w.r.t. $V(G) \setminus (X \cup S_t)$. We will refer to these twin classes as $R^t_1, R^t_2, \ldots, R^t_z$.

We are now ready to formally define the dynamic programming table M_t that stores the information we require at a node t of T. For $b_1, b_2, \ldots, b_k \subseteq [z]$ and $\{\, d_Z \in [n] \mid Z \subseteq [z] \,\}$, we let $M_t(b_1, b_2, \ldots, b_k, \{\, d_Z \mid Z \subseteq [z] \,\}) = 1$ if there is a proper coloring of G_t such that (1) for every $i \in [k]$, the color i appears in twin classes in $\{R^t_j : j \in b_i\}$ and does not appear in other twin classes, and (2) for every $Z \subseteq [z]$, d_Z is the number of colors from $\{k+1, k+2, \ldots, n\}$ that appear in twin classes of $\{R^t_j : j \in Z\}$ and do not appear in other twin classes. On the other hand, if no such proper coloring exists then we let $M_t(b_1, b_2, \ldots, b_k, \{\, d_Z \mid Z \subseteq [z] \,\}) = 0$.

The table M_t will be filled in a leaf-to-root fashion. Observe that by definition of d_Z's, for distinct subsets Z_1, Z_2 of $[z]$, d_{Z_1} and d_{Z_2} count disjoint sets of colors. This provides an easy way to count the total number of colors used. Since all vertices in $G - X$ appear below the root node r, the minimum number of colors required for a proper coloring of G will be the minimum value of $\left|\mathrm{col}_X \cup \{\, i \in [k] \mid b_i \neq \emptyset \,\}\right| + \sum_{Z \subseteq [z]} d_Z$, over all tuples $(b_1, b_2, \ldots, b_k, \{\, d_Z \mid Z \subseteq [z] \,\})$ whose M_r value is 1. ◀

Hamiltonian Cycle. In Hamiltonian Cycle, we are given an n-vertex graph G and asked whether G contains a cycle of length n as a subgraph. Note that if we restrict G to some subset of vertices $Y \subseteq V(G)$, then what remains from a Hamiltonian Cycle in G is a set of paths that start and end in the neighborhood of $V(G) \setminus Y$. Hence, the aim of this section is to solve the following generalization of Hamiltonian Cycle:

$\mathcal{R}_c$-Disjoint Paths Cover	*Parameter: k*
Instance:	A graph G, a k-vertex modulator $X \subseteq V(G)$ to $\mathcal{R}_c$, and $m \leq k$ pairs $(s_1, t_1), \ldots, (s_m, t_m)$ of vertices from X with $s_i \neq t_i$ for all $i \in [m]$.
Task:	Decide whether there are internally vertex-disjoint paths $P_1, P_2, \ldots, P_m$ in G such that P_i is a path from s_i to t_i and every vertex in $G - X$ belongs to precisely one path in $P_1, P_2, \ldots, P_m$.

▶ **Theorem 9.** *$\mathcal{R}_c$-Disjoint Paths Cover can be solved in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$.*

Proof Sketch. Let G be a graph, X be a k-vertex modulator to $\mathcal{R}_c$, and $(s_1, t_1), \ldots, (s_m, t_m)$ be m pairs of vertices from X. Our starting point is once again a rooted rank-decomposition (T, μ) of $G - X$ of width at most c, which may be computed in time $\mathcal{O}(n^3)$ [26]. We will obtain a fixed-parameter algorithm for checking the existence of such paths $P_1, \ldots, P_m$ in G by expanding the records used in Espelage, Gurski and Wanke's algorithm [15] for computing Hamiltonian Cycle parameterized by clique-width.

To follow partial solutions on each subgraph G_t, we consider certain generalizations of path-partitions of subgraphs of G. For a subgraph H of G, an *X-lenient path-partition* $\mathcal{P}$ of H is a collection of paths in H that are internally vertex-disjoint and share only endpoints in X such that $\bigcup_{P\in\mathcal{P}} V(P) = V(H)$. For convenience, we consider a path as an ordered sequence of vertices, and for a path $P = v_1v_2\cdots v_x$, we define $\ell(P) = v_1$ and $r(P) = v_x$.

We proceed by introducing our dynamic programming table. For each node t of T, we use the following tuples (D, SP) as indices of the table. Let $D = \{d_{b_1,b_2} \in \{0,1,\ldots,n\} \mid (b_1,b_2) \in [z] \times [z]\}$. The integer d_{b_1,b_2} will represent the number of paths in an X-lenient path-partition of G_t that are fully contained in $G_t - X$ and whose endpoints are contained in $R^t_{b_1}$ and $R^t_{b_2}$. Let SP be a set such that

- for each $i \in \{1,\ldots,m\}$, $(i,0,x)$, $(0,i,x)$, (i,i) with some $x \in [z]$ are the only possible tuples in SP,
- each integer in $\{1,\ldots,m\}$ appears at most once as an ℓ among all tuples $(\ell,0,p)$ or (ℓ,ℓ) in SP, and similarly, each integer in $\{1,\ldots,m\}$ appears at most once as an r among all tuples $(0,r,p)$ or (r,r) in SP.

In short, the tuple $(i,0,t)$ indicates the existence of a path starting in s_i and ending at a vertex in R^t_x. Similarly, $(0,i,t)$ indicates the existence of a path starting in t_i and ending in R^t_x. The tuple (i,i) indicates the existence of a path starting in s_i and ending in t_i. Note that there are at most $(n+1)^{z^2}$ possibilities for D. For an element of SP, there are $2mz+1$ possible elements in SP, and thus there are at most 2^{2kz+1} possibilities for SP. This implies that the number of possible tuples (D, SP) is bounded by $(n+1)^{z^2}2^{2kz+1}$.

We define a DP table M_t such that $M_t(D, SP) = 1$ if there is an X-lenient path-partition $\mathcal{P} = \mathcal{P}_1 \uplus \mathcal{P}_2$ of G_t such that

- $\mathcal{P}_1$ is the subset of $\mathcal{P}$ that consists of all paths fully contained in $G_t - X$,
- for every $d_{b_1,b_2} \in D$, there are exactly d_{b_1,b_2} distinct paths in $\mathcal{P}_1$ with endpoints in $R^t_{b_1}$ and $R^t_{b_2}$,
- for every (ℓ,r,p) or $(\ell,r) \in SP$, there is a unique path $P \in \mathcal{P}_2$ such that
  - if $\ell = i > 0$, then $\ell(P) = s_i$, and if $r = i > 0$, then $r(P) = t_i$,
  - if $\ell = 0$, then $\ell(P) \in R^t_p$, and if $r = 0$, then $r(P) \in R^t_p$,

In this case, we say that the X-lenient path-partition $\mathcal{P}$ is a partial solution with respect to (D, SP), and also (D, SP) is a characteristic of $\mathcal{P}$. We define $\mathcal{Q}_t$ as the set of all tuples (D, SP) where $M_t(D, SP) = 1$.

The table M_t is filled in a leaf-to-root fashion. Since all vertices in $G - X$ appear below the root node ro, to decide whether there is a desired X-lenient path-partition, it suffices to confirm that there are D and SP such that $M_{ro}(D, SP) = 1$, for every $d_{b_1,b_2} \in D$, $d_{b_1,b_2} = 0$, and for every $i \in \{1,2,\ldots,m\}$, $(i,i) \in SP$.

The proof can be completed by describing a dynamic program to fill in the table M_t for each node $t \in V(T)$ in a leaf-to-root fashion. ◀

Max-Cut. The third problem we consider is MAX-CUT, where we are given an integer ℓ together with an n-vertex graph G and asked whether $V(G)$ can be partitioned into sets V_1 and V_2 such that the number of edges with precisely one endpoint in V_1 (called the *cut size*) is at least ℓ.

$\mathcal{R}_c$-MAX-CUT EXTENSION	*Parameter:* k
Instance:	A graph G, a k-vertex modulator $X \subseteq V(G)$ to $\mathcal{R}_c$, $s \subseteq X$, and $\ell \in \mathbb{N}$.
Task:	Is there a partition of $V(G)$ into sets V_1 and V_2 such that $X \cap V_1 = s$ and the number of edges between V_1 and V_2 is at least ℓ.

▶ **Theorem 10.** $\mathcal{R}_c$-MAX-CUT EXTENSION *can be solved in polynomial time.*

6 Algorithmic Applications of $\mathcal{R}_c$-Treewidth

In this section, we show that Chromatic Number, Hamiltonian Cycle and Max-Cut are FPT parameterized by $\mathcal{R}_c$-treewidth. As our starting point, recall that each of these problems admits a fixed-parameter algorithm when parameterized by treewidth which is based on leaf-to-root dynamic programming along the nodes of a nice tree decomposition. Notably, the algorithms are based on defining a certain *record* $\delta(P, Q)$ (for vertex sets P, Q) such that $\delta(B_t, Y_t)$ captures all the relevant information required to solve the problem on $G[Y_t]$ and to propagate this information from a node t to its parent. The algorithms compute these records on the leaves of the tree decomposition by brute force, and then dynamically update these records while traversing a nice tree decomposition towards the root; once the record $\delta(B_r, Y_r)$ is computed for the root r of the decomposition, the algorithm outputs the correct answer.

Our general strategy for solving these problems will be to replicate the records employed by the respective dynamic programming algorithm $\mathbb{A}$ used for treewidth, but *only for the nice $\mathcal{R}_c$-tree decomposition of the torso* of the input graph G. Recall that aside from the "standard" simple leaf nodes, nice $\mathcal{R}_c$-tree decompositions also contain boundary leaf nodes, which serve as separators between the torso and a connected component C with rank-width at most c. For $\mathbb{A}$ to work correctly with the desired runtime, we need to compute the record for each boundary leaf node using a subprocedure that exploits the bounded rank-width of C; in particular, we will see that this amounts to solving the problems defined in Section 5. Before proceeding to the individual problems, we provide a formalization and proof for the general ideas outlined above.

▶ **Lemma 11.** *Let $\mathcal{P}$ be a graph problem which can be solved via a fixed-parameter algorithm $\mathbb{A}$ parameterized by treewidth, where $\mathbb{A}$ runs in time $f(k') \cdot n'^a$ and operates by computing a certain record δ in a leaves-to-root fashion along a provided nice width-k' tree decomposition of the n'-vertex input graph.*

Let $\mathcal{Q}$ be obtained from $\mathcal{P}$ by receiving the following additional information in the input: (1) a nice $\mathcal{R}_c$-tree decomposition $(X, T, \{B_t \mid t \in V(T)\})$ of width k for the input n-vertex graph G, and (2) for each boundary leaf node t corresponding to the neighborhood of a connected component C of $G[X]$, the record $\delta(B_t, B_t \cup C)$.

Then, $\mathcal{Q}$ can be solved in time $f(k) \cdot n^a$.

Chromatic Number. Chromatic Number is W[1]-hard parameterized by rank-width [17] but can be solved in time $2^{\mathcal{O}(\mathbf{tw}(G) \cdot \log \mathbf{tw}(G))} \cdot n$ on n-vertex graphs when a minimum-width tree-decomposition is provided with the input [28]; moreover, it is known that this runtime is essentially tight [32].

It is well known that the chromatic number is at most $\mathbf{tw}(G) + 1$. One possible way of defining records in order to achieve a runtime of $2^{\mathcal{O}(\mathbf{tw}(G) \cdot \log \mathbf{tw}(G))} \cdot n$ is to track, for each proper coloring of vertices in a bag B_t, the minimum number of colors required to extend such a coloring to Y_t [28]. Formally, let S_t be the set of all colorings of B_t with colors $[\mathbf{tw}(G) + 1]$, and let $\alpha(B_t, Y_t) : S_t \to \mathbb{Z}$ be defined as follows:

- $\alpha(B_t, Y_t)(s) = -1$ if s is not a proper coloring of $G[B_t]$.
- $\alpha(B_t, Y_t)(s) = q$ if q is the minimum number of colors used by any proper coloring of $G[Y_t]$ which extends s.

Using Theorem 8, we can compute such $\alpha(B_t, Y_t)(s)$ for every proper coloring s of B_t. Hence, combining Lemma 11 and Theorem 8, we obtain:

▶ **Theorem 12.** CHROMATIC NUMBER *can be solved in time* $2^{\mathcal{O}(k\log(k))}\cdot n^{\mathcal{O}(1)}$ *if a nice* $\mathcal{R}_c$*-tree decomposition of width* k *is provided on the input.*

Hamiltonian Cycle. HAMILTONIAN CYCLE is W[1]-hard parameterized by rank-width [17] but can be solved in time $2^{\mathcal{O}(\mathbf{tw}(G)\cdot\log\mathbf{tw}(G))}\cdot n$ on n-vertex graphs when a minimum-width tree-decomposition is provided with the input via standard dynamic programming. This algorithm can be improved to run in time $2^{\mathcal{O}(\mathbf{tw}(G))}\cdot n)$ by applying the advanced *rank-based approach* of Cygan, Kratsch and Nederlof [10] to prune the number of records. To simplify our exposition, here we focus on extending the standard dynamic programming algorithm which yields a slightly super-exponential runtime.

One possibility for defining the records for HAMILTONIAN CYCLE is to track all possible ways one can cover Y_t by paths that start and end in B_t (intuitively, this corresponds to what remains of a hypothetical solution if we "cut off" everything above Y_t) [12]. Formally, let ${B_t}^\diamond$ be defined as follows:

- if $|B_t|>2$, then ${B_t}^\diamond$ is the set of graphs with at most $|B_t|$ edges and degree at most 2 over vertex set B_t;
- if $|B_t|=2$, then ${B_t}^\diamond$ contains three (multi)graphs over vertex set B_t: the edgeless graph, the graph with one edge, and the multigraph with two edges and no loops;
- if $|B_t|=1$, then ${B_t}^\diamond$ contains an edgeless graph and a graph with a single loop, both over the single vertex in B_t;
- if $|B_t|=0$, then ${B_t}^\diamond=\{\text{YES, NO}\}$.

We let $\beta(B_t,Y_t):{B_t}^\diamond\to\{0,1\}$, where for $Q\in{B_t}^\diamond$ we set $\beta(B_t,Y_t)(Q)=1$ if and only if there exists a set P of paths in $G[Y_t]$ and a bijection that maps each $(v_1,\dots,v_\ell)\in P$ to an edge $(v_1,v_\ell)\in E(Q)$ such that each vertex $v\in G[Y_t\setminus B_t]$ is contained in precisely one path in P. In the special case where $B_t=\emptyset$, our records explicitly state whether $G[Y_t]$ contains a Hamiltonian cycle or not.

As before, we can now shift our attention to the problem of computing our records in boundary leaf nodes. We do so by looping over all of the at most k^{2k}-many graphs $Q\in{B_t}^\diamond$; for each such Q we check whether $G[Y_t]-B_t$ can be covered by internally vertex-disjoint paths connecting the pairs of vertices in B_t that form the endpoints of the edges in Q. Hence, we are left with the $\mathcal{R}_c$-DISJOINT PATHS COVER problem. From Theorem 9 and Lemma 11, we obtain:

▶ **Theorem 13.** HAMILTONIAN CYCLE *can be solved in time* $2^{\mathcal{O}(k\log(k))}\cdot n^{\mathcal{O}(1)}$ *if a nice* $\mathcal{R}_c$*-tree decomposition of width* k *is provided on the input.*

Max-Cut. MAX-CUT is another problem that is W[1]-hard parameterized by rank-width [19] but admits a simple fixed-parameter algorithm parameterized by treewidth – notably, it can be solved in time $2^{\mathcal{O}(\mathbf{tw}(G))}\cdot n$ on n-vertex graphs when a minimum-width tree-decomposition is provided with the input via standard dynamic programming [9, 12].

The simplest way of defining the records for MAX-CUT is to keep track of all possible ways the bag B_t can be partitioned into V_1 and V_2, and for each entry in our table we keep track of the maximum number of crossing edges in Y_t compatible with that entry. Formally, let $\gamma(B_t,Y_t):2^{B_t}\to\mathbb{N}_0$, where for each $s\in 2^{B_t}$ it holds that $\gamma(B_t,Y_t)(s)$ is the maximum cut size that can be achieved in $G[Y_t]$ by any partition (V_1,V_2) satisfying $V_1\cap B_t=s$. As before, from Theorem 10 and Lemma 11, we obtain:

▶ **Theorem 14.** MAX-CUT *can be solved in time* $2^k\cdot n^{\mathcal{O}(1)}$*, if a nice* $\mathcal{R}_c$*-tree decomposition of width* k *is provided on the input.*

7 Concluding Remarks

While the technical contribution of this paper mainly focused on $\mathcal{R}_c$-treewidth, a parameter that allows us to lift fixed-parameter algorithms parameterized by treewidth to well-structured dense graph classes, it is equally viable to consider $\mathcal{H}$-treewidth for other choices of $\mathcal{H}$. Naturally, one should aim at graph classes where problems of interest become tractable, but it is also important to make sure that a (nice) $\mathcal{H}$-tree decomposition can be computed efficiently (i.e., one needs to obtain analogues to our Lemma 7). Examples of graph classes that may be explored in this context include split graphs, interval graphs, and more generally graphs of bounded mim-width [27].

References

1 Takanori Akiyama, Takao Nishizeki, and Nobuji Saito. $\mathcal{NP}$-completeness of the Hamiltonian cycle problem for bipartite graphs. *J. Inform. Process.*, 3:73–76, January 1980.

2 Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) Kernelization. *J. ACM*, 63(5):44:1–44:69, November 2016. doi:10.1145/2973749.

3 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Preprocessing for Treewidth: A Combinatorial Analysis through Kernelization. *SIAM J. Discrete Math.*, 27(4):2108–2142, 2013.

4 Hans L. Bodlaender and Babette van Antwerpen-de Fluiter. Reduction Algorithms for Graphs of Small Treewidth. *Inf. Comput.*, 167(2):86–119, 2001.

5 Leizhen Cai. Parameterized complexity of vertex colouring. *Discrete Applied Mathematics*, 127(3):415–429, 2003.

6 Bruno Courcelle. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Inf. Comput.*, 85(1):12–75, 1990.

7 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.

8 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.

9 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Texts in Computer Science. Springer, 2013.

10 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast Hamiltonicity Checking Via Bases of Perfect Matchings. *J. ACM*, 65(3):12:1–12:46, 2018.

11 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer Verlag, New York, 2nd edition, 2000.

12 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

13 Eduard Eiben, Robert Ganian, and O-joung Kwon. A single-exponential fixed-parameter algorithm for distance-hereditary vertex deletion. *J. Comput. Syst. Sci.*, 97:121–146, 2018.

14 Eduard Eiben, Robert Ganian, and Stefan Szeider. Solving Problems on Graphs of High Rank-Width. *Algorithmica*, 80(2):742–771, 2018.

15 Wolfgang Espelage, Frank Gurski, and Egon Wanke. Deciding Clique-Width for Graphs of Bounded Tree-Width. *J. Graph Algorithms Appl.*, 7(2):141–180, 2003.

16 Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Frances A. Rosamond, and Saket Saurabh. Graph Layout Problems Parameterized by Vertex Cover. In *Proc. ISAAC 2008*, volume 5369 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2008.

17 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Clique-width: on the price of generality. In *Proc. SODA 2009*, pages 825–834. SIAM, 2009.

18 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Intractability of Clique-Width Parameterizations. *SIAM J. Comput.*, 39(5):1941–1956, 2010.

19 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.

20 Jakub Gajarský, Petr Hlinený, Jan Obdrzálek, Sebastian Ordyniak, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. Kernelization using structural parameters on sparse graph classes. *J. Comput. Syst. Sci.*, 84:219–242, 2017.

21 Robert Ganian and Petr Hlinený. On parse trees and Myhill-Nerode-type tools for handling graphs of bounded rank-width. *Discrete Applied Mathematics*, 158(7):851–867, 2010.

22 Robert Ganian, Petr Hlinený, and Jan Obdrzálek. A unified approach to polynomial algorithms on graphs of bounded (bi-)rank-width. *Eur. J. Comb.*, 34(3):680–701, 2013.

23 Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. Going Beyond Primal Treewidth for (M)ILP. In *Proc. AAAI 2017*, pages 815–821. AAAI Press, 2017.

24 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Backdoor Treewidth for SAT. In *Proc. SAT 2017*, volume 10491 of *Lecture Notes in Computer Science*, pages 20–37. Springer, 2017.

25 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Combining Treewidth and Backdoors for CSP. In *Proc. STACS 2017*, volume 66 of *LIPIcs*, pages 36:1–36:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

26 Petr Hliněný and Sang-il Oum. Finding Branch-Decompositions and Rank-Decompositions. *SIAM J. Comput.*, 38(3):1012–1032, June 2008. `doi:10.1137/070685920`.

27 Lars Jaffke, O-joung Kwon, and Jan Arne Telle. A Unified Polynomial-Time Algorithm for Feedback Vertex Set on Graphs of Bounded Mim-Width. In *Proc. STACS 2018*, volume 96 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

28 Klaus Jansen and Petra Scheffler. Generalized Coloring for Tree-like Graphs. *Discrete Applied Mathematics*, 75(2):135–155, 1997.

29 Mamadou Moustapha Kanté, Eun Jung Kim, O-joung Kwon, and Christophe Paul. An FPT Algorithm and a Polynomial Kernel for Linear Rankwidth-1 Vertex Deletion. *Algorithmica*, 79(1):66–95, 2017. `doi:10.1007/s00453-016-0230-z`.

30 Eun Jung Kim and O-joung Kwon. A Polynomial Kernel for Distance-Hereditary Vertex Deletion. In *Proc. WADS 2017*, volume 10389 of *Lecture Notes in Computer Science*, pages 509–520. Springer, 2017.

31 Daniel Kobler and Udi Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Applied Mathematics*, 126(2-3):197–221, 2003.

32 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly Superexponential Parameterized Problems. *SIAM J. Comput.*, 47(3):675–702, 2018.

33 Dániel Marx and Paul Wollan. Immersions in Highly Edge Connected Graphs. *SIAM J. Discrete Math.*, 28(1):503–520, 2014.

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

35 Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B*, 36(1):49–64, 1984.

36 Sigve Hortemo Sæther and Jan Arne Telle. Between Treewidth and Clique-Width. In *Proc. WG 2014*, pages 396–407, 2014.