Structural Parameterizations of Clique Coloring

Lars Jaffke
University of Bergen, Norway
lars.jaffke@uib.no

Paloma T. Lima
University of Bergen, Norway
paloma.lima@uib.no

Geevarghese Philip
Chennai Mathematical Institute, India
UMI ReLaX, Chennai, India
gphilip@cmi.ac.in

Abstract

A clique coloring of a graph is an assignment of colors to its vertices such that no maximal clique is monochromatic. We initiate the study of structural parameterizations of the Clique Coloring problem which asks whether a given graph has a clique coloring with q colors. For fixed $q \geq 2$, we give an $\mathcal{O}^\star(q^{\mathsf{tw}})$-time algorithm when the input graph is given together with one of its tree decompositions of width tw. We complement this result with a matching lower bound under the Strong Exponential Time Hypothesis. We furthermore show that (when the number of colors is unbounded) Clique Coloring is XP parameterized by clique-width.

2012 ACM Subject Classification Mathematics of computing → Graph coloring

Keywords and phrases clique coloring, treewidth, clique-width, structural parameterization, Strong Exponential Time Hypothesis

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.49

Related Version A full version of this paper is available at `https://arxiv.org/abs/2005.04733`.

Funding *Lars Jaffke*: Supported by the Trond Mohn Foundation (TMS).

Acknowledgements The work was partially done while L. J. and P. T. L. were visiting Chennai Mathematical Institute.

1 Introduction

Vertex coloring problems are central in algorithmic graph theory, and appear in many variants. One of these is Clique Coloring, which given a graph G and an integer k asks whether G has a clique coloring with k colors, i.e. whether each vertex of G can be assigned one of k colors such that there is no monochromatic maximal clique. The notion of a clique coloring of a graph was introduced in 1991 by Duffus et al. [16], and it behaves quite differently from the classical notion of a proper coloring, which forbids monochromatic edges. Any proper coloring is a clique coloring, but not vice versa. For instance, a complete graph on n vertices only has a proper coloring with n colors, while it has a clique coloring with two colors. Moreover, proper colorings are closed under taking subgraphs. On the other hand, removing vertices or edges from a graph may introduce new maximal cliques, therefore a clique coloring of a graph is not always a clique coloring of its subgraphs, not even of its induced subgraphs.

45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020).
Editors: Javier Esparza and Daniel Kráľ; Article No. 49; pp. 49:1–49:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Also from a complexity-theoretic perspective, Clique Coloring behaves very differently from Graph Coloring. Most notably, while it is easy to decide whether a graph has a proper coloring with two colors, Bacsó et al. [2] showed that it is already coNP-hard to decide if a given coloring with two colors is a clique coloring. Marx [26] later proved Clique Coloring to be Σ_2^p-complete for every fixed number of (at least two) colors.

On the positive side, Cochefert and Kratsch showed that the Clique Coloring problem can be solved in $\mathcal{O}^\star(2^n)$ time,[1] and the problem has been shown to be polynomial-time solvable on several graph classes. Mohar and Skrekovski [27] showed that all planar graphs are 3-clique colorable, and Kratochvíl and Tuza gave an algorithm that decides whether a given planar graph is 2-clique colorable [24]. For several graph classes it has been shown that all their members except odd cycles on at least five vertices (which require three colors) are 2-clique colorable [2, 3, 6, 7, 14, 23, 28, 31]. Therefore, on these classes Clique Coloring is polynomial-time solvable. Duffus et al. [16] even conjectured in 1991 that perfect graphs are 3-clique colorable, which was supported by many subclasses of perfect graphs being shown to be 2- or 3-clique colorable [1, 2, 9, 14, 16, 27, 28]. However, in 2016, Charbit et al. [8] showed that there are perfect graphs whose clique colorings require an unbounded number of colors.

In this work, we consider Clique Coloring from the viewpoint of parameterized algorithms and complexity [13, 15]. In particular, we consider structural parameterizations of Clique Coloring by two of the most commonly used decomposition-based width measures of graphs, namely *treewidth* and *clique-width*. Informally speaking, the treewidth of a graph G measures how close G is to being a forest. On dense graphs, the treewidth is unbounded, and clique-width can be viewed as an extension of treewidth that remains bounded on several simply structured dense graphs.

Our first main result is a fixed-parameter tractable algorithm for q-Clique Coloring parameterized by treewidth. More precisely: we show that for any fixed $q \geq 2$, q-Clique Coloring (asking for a clique coloring with q colors) can be solved in time $\mathcal{O}^\star(q^{\mathsf{tw}})$, where tw denotes the width of a given tree decomposition of the input graph. We also show that this running time is likely the best possible in this parameterization; we prove that under the Strong Exponential Time Hypothesis (SETH), for any $q \geq 2$, there is no $\epsilon > 0$ such that q-Clique Coloring can be solved in time $\mathcal{O}^\star((q-\epsilon)^{\mathsf{tw}})$. In fact, we rule out $\mathcal{O}^\star((q-\epsilon)^t)$-time algorithms for a much smaller class of graphs than those of treewidth t, namely: graphs that have both *pathwidth* and *feedback vertex set number* simultaneously bounded by t.

Our second main result is an XP algorithm for Clique Coloring with clique-width as the parameter. The algorithm runs in time $k^{f(w)} \cdot n \leq n^{\mathcal{O}(f(w))}$, where w is the clique-width w of a given clique-width expression of the input n-vertex graph, k the number of colors, and $f(w) = 2^{2^{\mathcal{O}(w)}}$. The double-exponential dependence on w in the degree of the polynomial stems from the notorious property of clique colorings which we mentioned above; namely, that taking induced subgraphs does not necessarily preserve clique colorings. This results in a large amount of information that needs to be carried along as the algorithm progresses.

This algorithm raises two questions. First, if Clique Coloring is FPT parameterized by clique-width even if k is a priori unbounded. Second, if the triple exponential dependence on w can be avoided under for instance the Exponential Time Hypothesis (ETH), also in the case when k is fixed. Intuitively, a positive answer to the first question only seems feasible via a proof that all graphs of clique-width w can be clique colored with at most some $g(w)$

[1] The $\mathcal{O}^\star$-notation suppresses polynomial factors in the input size, i.e. for inputs of size n, we have that $\mathcal{O}^\star(f(n)) = \mathcal{O}(f(n) \cdot n^{\mathcal{O}(1)})$.

colors, for some function g. However, the current literature appears to be far from providing such a result. On the other hand, hardness proofs for Graph Coloring parameterized by clique-width [17, 18] rely on the fact that cliques require many colors while keeping the clique-width small; since cliques can be clique colored with two colors, these tricks are of no use in the setting of Clique Coloring. For the second (possibly more tangible) question, one could search for an algorithm for 2-Clique Coloring running in time $2^{2^{2^{o(w)}}} \cdot n^{\mathcal{O}(1)}$, or rule out the existence of such an algorithm under ETH.

2 Preliminaries

Throughout this work, proofs of statements marked with "♣" are deferred to the full version. For basic terminology in graph theory we refer the reader to [5] (or the full version). Let Ω be a set and $\sim$ an equivalence relation over Ω. For an element $x \in \Omega$ the *equivalence class of* x, denoted by $[x]$, is the set $\{y \in \Omega \mid x \sim y\}$. We denote the set of all equivalence classes of $\sim$ by $\Omega/\sim$.

Parameterized Complexity and SETH. We give the basic definitions of parameterized complexity that are relevant to this work and refer to [13, 15] for details. Let Σ be an alphabet. A *parameterized problem* is a set $\Pi \subseteq \Sigma^* \times \mathbb{N}$, the second component being the *parameter* which usually expresses a structural measure of the input. A parameterized problem Π is said to be *fixed-parameter tractable*, or in the complexity class FPT, if there is an algorithm that for any $(x, k) \in \Sigma^* \times \mathbb{N}$ correctly decides whether or not $(x, k) \in \Pi$, and runs in time $f(k) \cdot |x|^c$ for some computable function $f : \mathbb{N} \to \mathbb{N}$ and constant c. We say that a parameterized problem is in the complexity class XP, if there is an algorithm that for each $(x, k) \in \Sigma^* \times \mathbb{N}$ correctly decides whether or not $(x, k) \in \Pi$, and runs in time $f(k) \cdot |x|^{g(k)}$, for some computable functions f and g.

In 2001, Impagliazzo et al. conjectured that a brute force algorithm to solve the q-SAT problem which given a CNF-formula with clauses of size at most q, asks whether it has a satisfying assignment, is "essentially optimal". This conjecture is called the *Strong Exponential Time Hypothesis*, and can be formally stated as follows. (For a survey of conditional lower bounds based on SETH and related conjectures, see [32].)

▶ **Conjecture** (SETH, Impagliazzo et al. [19, 20]). *For every $\epsilon > 0$, there is a $q \in \mathcal{O}(1)$ such that q-SAT on n variables cannot be solved in time $\mathcal{O}^\star((2-\epsilon)^n)$.*

Treewidth. We now define the treewidth and pathwidth of a graph.

▶ **Definition 1** (Treewidth, Pathwidth). *Let G be a graph. A* tree decomposition *of G is a pair $(T, \mathcal{B})$ of a tree T and an indexed family of vertex subsets $\mathcal{B} = \{B_t \subseteq V(G)\}_{t \in V(T)}$, called* bags*, satisfying the following properties.*

(T1) $\bigcup_{t \in V(T)} B_t = V(G)$.

(T2) *For each $uv \in E(G)$ there exists some $t \in V(T)$ such that $\{u, v\} \subseteq B_t$.*

(T3) *For each $v \in V(G)$, let $U_v := \{t \in V(T) \mid v \in B_t\}$ be the nodes in T whose bags contain v. Then, $T[U_v]$ is connected.*

The width *of $(T, \mathcal{B})$ is $\max_{t \in V(T)} |B_t| - 1$, and the* tree-width *of a graph is the minimum width over all its tree decompositions. If T is a path, then $(T, \mathcal{B})$ is called a* path decomposition*, and the* path-width *of a graph is the minimum width over all its path decompositions.*

Branch Decompositions and Module-Width. We skip the definition of clique-width and refer to [11]; instead, we define the equivalent measure of *module-width* that we will use in our algorithm. The definition of module-width is based the notion of a rooted branch decomposition.

▶ **Definition 2** (Branch decomposition). *Let G be a graph. A* branch decomposition *of G is a pair $(T, \mathcal{L})$ of a subcubic tree T and a bijection $\mathcal{L}\colon V(G) \to L(T)$. If T is a caterpillar, then $(T, \mathcal{L})$ is called a* linear branch decomposition. *If T is rooted, then we call $(T, \mathcal{L})$ a* rooted branch decomposition. *In this case, for $t \in V(T)$, we denote by T_t the subtree of T rooted at t, and we define $V_t \coloneqq \{v \in V(G) \mid \mathcal{L}(v) \in \mathrm{L}(T_t)\}$, $\overline{V_t} \coloneqq V(G) \setminus V_t$, and $G_t \coloneqq G[V_t]$.*

Module-width is attributed to Rao [29, 30]. On a high level, the module-width of a rooted branch decomposition bounds, at each of its nodes t, the maximum number of subsets of $\overline{V_t}$ that make up the intersection of $\overline{V_t}$ with the neighborhood of some vertex in V_t.

▶ **Definition 3** (Module-width). *Let G be a graph, and $(T, \mathcal{L})$ be a rooted branch decomposition of G. For each $t \in V(T)$, let $\sim_t$ be the equivalence relation on V_t defined as follows:*

$$\forall u, v \in V_t\colon u \sim_t v \Leftrightarrow N_G(u) \cap \overline{V_t} = N_G(v) \cap \overline{V_t}$$

The module-width *of $(T, \mathcal{L})$ is $\mathsf{mw}(T, \mathcal{L}) \coloneqq \max_{t \in V(T)} |V_t/{\sim_t}|$. The* module-width of G, *denoted by $\mathsf{mw}(G)$, is the minimum module width over all rooted branch decompositions of G.*

We introduce some notation. For a node $t \in V(T)$ and a set $S \subseteq V(G_t)$, we let $\mathsf{eqc}_t(S)$ be the set of all equivalence classes of $\sim_t$ which have a nonempty intersection with S, and $\overline{\mathsf{eqc}}_t(S)$ be the remaining equivalence classes of $\sim_t$. Formally, $\mathsf{eqc}_t(S) \coloneqq \{Q \in V_t/{\sim_t} \mid Q \cap S \neq \emptyset\}$ and $\overline{\mathsf{eqc}}_t(S) \coloneqq V_t/{\sim_t} \setminus \mathsf{eqc}_t(S)$. Moreover, for a set of equivalence classes $\mathcal{Q} \subseteq V_t/{\sim_t}$, we let $V(\mathcal{Q}) \coloneqq \bigcup_{Q \in \mathcal{Q}} Q$.

Let $(T, \mathcal{L})$ be a rooted branch decomposition of a graph G and let $t \in V(T)$ be a node with children r and s. We now describe an operator associated with t that tells us how the graph G_t is formed from its subgraphs G_r and G_s, and how the equivalence classes of $\sim_t$ are formed from the equivalence classes of $\sim_r$ and $\sim_s$. Concretely, we associate with t a bipartite graph H_t on bipartition $(V_r/{\sim_r}, V_s/{\sim_s})$ such that:

(i) $E(G_t) = E(G_r) \cup E(G_s) \cup F$, where $F = \{uv \mid u \in V_r, v \in V_s, \{[u], [v]\} \in E(H_t)\}$, and
(ii) there is a partition $\mathcal{P} = \{P_1, \ldots, P_h\}$ of $V(H_t)$ such that $V_t/{\sim_t} = \{Q_1, \ldots, Q_h\}$, where for $1 \le i \le h$, $Q_i = \bigcup_{Q \in P_i} Q$. For each $1 \le i \le h$, we call P_i the *bubble* of the resulting equivalence class $\bigcup_{Q \in P_i} Q$ of $\sim_t$.

As auxiliary structures, for $p \in \{r, s\}$, we let $\eta_p\colon V_p/{\sim_p} \to V_t/{\sim_t}$ be the map such that for all $Q_p \in V_p/{\sim_p}$, $Q_p \subseteq \eta_p(Q_p)$, i.e. $\eta_p(Q_p)$ is the equivalence class of $\sim_t$ whose bubble contains Q_p. We call (H_t, η_r, η_s) the *operator* of t.

▶ **Theorem 4** (Rao, Thm. 6.6 in [29]). *For any graph G, $\mathsf{mw}(G) \le \mathsf{cw}(G) \le 2 \cdot \mathsf{mw}(G)$, where $\mathsf{mw}(G)$ denotes the module-width of G and $\mathsf{cw}(G)$ denotes the clique-width of G, and given a decomposition of bounded clique-width, a decomposition of bounded module-width, and vice versa, can be constructed in time $\mathcal{O}(n^2)$, where $n = |V(G)|$.*

Colorings. Let G be a graph. An ordered partition $\mathcal{C} = (C_1, \ldots, C_k)$ of $V(G)$ is called a *coloring* of G with k colors, or a k*-coloring* of G. (Observe that for $i \in \{1, \ldots, k\}$, C_i may be empty.) For $i \in \{1, \ldots, k\}$, we call C_i the *color class* i, and say that the vertices in C_i *have color* i. $\mathcal{C}$ is called *proper* if for all $i \in \{1, \ldots, k\}$, C_i is an independent set in G.

A coloring $\mathcal{C} = (C_1, \ldots, C_k)$ of a graph G is called a *clique coloring (with k colors)* if there is no monochromatic maximal clique, i.e. no maximal clique X in G such that $X \subseteq C_i$ for some i. In this work, we study the following computational problem.

Clique Coloring

Input: Graph G, integer k
Question: Does G have a clique coloring with k colors?

For $q \geq 2$, we denote by q-Clique Coloring the problem when the number of colors q is fixed and not part of the input. The q-Coloring and q-List Coloring problems also make an appearance. In the former, we are given a graph G and the question is whether G has a proper coloring with q colors. In the latter, we are additionally given a list $L(V) \subseteq \{1, \ldots, q\}$ for each vertex $v \in V(G)$, and additionally require the color of each vertex to be from its list.

Whenever convenient, we alternatively denote a coloring of a graph with k colors as a map $\phi\colon V(G) \to \{1, \ldots, k\}$. In this case, a restriction of ϕ to S is the map $\phi|_S\colon S \to \{1, \ldots, k\}$ with $\phi|_S(v) = \phi(v)$ for all $v \in S$. For any $T \subseteq V(G)$ with $S \subseteq T$, we say that $\phi|_T$ extends $\phi|_S$.

3 Parameterized by Treewidth

In this section, we consider the q-Clique Coloring problem, for fixed $q \geq 2$, parameterized by treewidth. First, in Section 3.1, we show that if we are given a tree decomposition of width tw of the input graph, then q-Clique Coloring can be solved in time $\mathcal{O}^\star(q^{\mathsf{tw}})$. After that, in Section 3.2, we show that this is tight according to SETH, by providing one reduction ruling out $\mathcal{O}^\star((2-\epsilon)^{\mathsf{tw}})$-time algorithms for 2-Clique Coloring and another one ruling out $\mathcal{O}^\star((q-\epsilon)^{\mathsf{tw}})$-time algorithms for q-Clique Coloring when $q \geq 3$.

3.1 Algorithm

The algorithm is bottom-up dynamic programming along a nice tree decomposition $(T, \mathcal{B})$ of the input graph G. At each bag B_t, we enumerate all colorings of $G[B_t]$ and verify for each such coloring if it can be extended to G_t such that there are no monochromatic maximal cliques that use a vertex from $V_t \setminus B_t$. Necessarily, we have to allow monochromatic maximal cliques S that are contained inside $G[B_t]$, since further up in the tree decomposition, there may be a vertex v that is complete to S. Therefore, all vertices in S may receive the same color, as long as v (or another such vertex) receives a different color. If on the other hand a monochromatic maximal clique has a vertex that has already been "forgotten" at t, i.e. it is contained in $V_t \setminus B_t$, then this vertex has no neighbors in $V(G) \setminus V_t$; therefore, no vertex from $V(G) \setminus V_t$ can "fix" this monochromatic maximal clique, and we can disregard the coloring at hand.

As a subroutine, we will have to be able to check at each bag B_t, if some subset $S \subseteq B_t$ contains a maximal clique in G_t. Doing this by brute force would add a *multiplicative* factor of roughly $2^{\mathsf{tw}} \cdot n$ to the runtime which we cannot afford. To avoid this increase in the runtime, we use fast subset convolution[2] [4] to build an oracle that, once constructed, can tell us in constant time whether or not any subset $S \subseteq B_t$ contains a maximal clique in G_t, for each node t. We give a dynamic programming algorithm (♣) that constructs such oracles for all

[2] Similar ideas have been used by Cochefert and Kratsch [10] to give an $\mathcal{O}^\star(2^n)$-time algorithm for Clique Coloring.

nodes in the tree decomposition, to ensure that we can maintain a runtime that is linear in n. Since it suffices to construct this oracle once per node, this will infer only an *additive* factor of $2^{\mathsf{tw}} \cdot \mathsf{tw}^{\mathcal{O}(1)} \cdot n$ to the runtime, which does not increase the worst-case complexity for any $q \geq 2$.

▶ **Theorem 5 (♣).** *For any fixed $q \geq 2$, there is an algorithm that given an n-vertex graph G and a tree decomposition of G of width* tw *which has $\mathcal{O}(\mathsf{tw} \cdot n)$ nodes, decides whether G has a clique coloring with q colors in time $\mathcal{O}(q^{\mathsf{tw}} \cdot \mathsf{tw}^{\mathcal{O}(1)} \cdot n)$, and constructs one such coloring, if it exists.*

3.2 Lower Bound

In this section we show that the previously presented algorithm is optimal under SETH. In fact, we prove hardness for a much larger parameter, namely the distance to a linear forest (for $q = 2$), and the distance to a caterpillar forest (for $q \geq 3$). Note that both paths and caterpillars have pathwidth 1, and clearly, they do not contain any cycles. Therefore, a lower bound parameterized by the (vertex deletion) distance to a linear/caterpillar forest implies a lower bound for the parameter pathwidth plus feedback vertex set number.

We first give the lower bound for the case $q = 2$. We would like to remark that Kratochvíl and Tuza [24] gave a reduction from Not-All-Equal SAT to 2-Clique Coloring as well, but their reduction does not imply the fine-grained lower bound we aim for here: the resulting graph is at distance $2n$ to a disjoint union of cliques of constant size (at most s). This only rules out $\mathcal{O}^\star((\sqrt{2} - \epsilon)^t)$-time algorithms parameterized by pathwidth, and does not give any lower bound if the feedback vertex set number is another component of the parameter.

▶ **Theorem 6.** *For any $\epsilon > 0$, 2-Clique Coloring parameterized by the distance t to a linear forest cannot be solved in time $\mathcal{O}^\star((2 - \epsilon)^t)$, unless* SETH *fails.*

Proof. We give a reduction from the well-known s-NAE-SAT problem, in which we are given a boolean CNF formula ϕ whose clauses are of size at most s, and the question is whether there is a truth assignment to the variables of ϕ, such that in each clause, at least one literal evaluates to true and at least one literal evaluates to false.

Let ϕ be a boolean CNF formula on n variables $x_1, \ldots, x_n$ with maximum clause size s. We denote by $\texttt{clauses}(\phi)$ the set of clauses of ϕ and by $\texttt{vars}(C)$ the set of variables that appear in the clause C of ϕ.

Given ϕ, we construct an instance G_ϕ for 2-Clique Coloring as follows. For each variable x_i, we create a vertex v_i in G. Let $V' = \{v_1, \ldots, v_n\}$. For each set S of variables, let $V_S = \{v_i \mid x_i \in S\}$. For each clause C_i of ϕ, we add the following clause gadget to G_ϕ. If C_i is monotone, add a path on four vertices to G_ϕ, the end vertices of which are a_i and b_i. Make $N(a_i) \cap V' = N(b_i) \cap V' = V_{\texttt{vars}(C_i)}$, and make $V_{\texttt{vars}(C_i)} \subset V'$ a clique. If C_i is not monotone, let $\texttt{pos}(C)$ (resp. $\texttt{neg}(C)$) denote the set of variables with positive (resp. negative) literals in C. Add a path on three vertices to G_ϕ, the end vertices of which are a_i and b_i, make $N(a_i) \cap V' = V_{\texttt{pos}(C)}$ and make $V_{\texttt{pos}(C)}$ a clique. Analogously, make $N(b_i) \cap V' = V_{\texttt{neg}(C)}$ and make $V_{\texttt{neg}(C)}$ a clique. Finally, add two adjacent vertices u, v to G_ϕ and make $N[u] = N[v] = \{u, v\} \cup V'$. See Figure 1.

▷ Claim (♣). G_ϕ is a yes-instance to 2-Clique Coloring if and only if ϕ is a yes-instance to s-NAE-SAT.

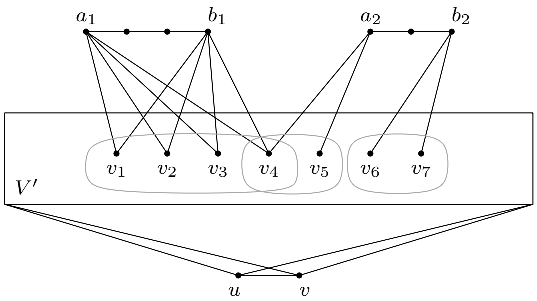

Figure 1 Depiction of G_ϕ with two clauses, namely a monotone clause $C_1 = \neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_4$ and a non-monotone clause $C_2 = x_4 \vee x_5 \vee \neg x_6 \vee \neg x_7$. Note that $G_\phi - V'$ is a linear forest.

Finally, note that $G - V'$ is a disjoint union of paths of length at most four. Hence, G is at distance n to a linear forest. Therefore, if for some $\epsilon > 0$, 2-Clique Coloring parameterized by the distance t to a linear forest can be solved in time $\mathcal{O}^\star((2-\epsilon)^t)$, then s-NAE-SAT can be solved in time $\mathcal{O}^\star((2-\epsilon)^n)$, which would contradict SETH [12]. This concludes the proof. ◀

We now turn to the case $q \geq 3$. Our reduction is from q-List-Coloring parameterized by the distance t to a linear forest, which has no $\mathcal{O}^\star((q-\epsilon)^t)$-time algorithms under SETH [21]. We require the lower bound to hold even when the input graphs are triangle-free, and in the full version we sketch that this is indeed the case, by the reduction presented in [21].

▶ **Theorem 7** (Jaffke and Jansen [21]). *For any $\epsilon > 0$ and any fixed $q \geq 3$, q-List Coloring on triangle-free graphs parameterized by the distance t to a linear forest cannot be solved in time $\mathcal{O}^\star((q-\epsilon)^t)$, unless* SETH *fails.*

The main ingredient in the proof of the next theorem is a construction based on Mycielski graphs. We would like to remark that also Marx [26] used Mycielski graphs and their properties in hardness proofs for the Clique Coloring problem.

▶ **Theorem 8.** *For any $\epsilon > 0$ and any fixed $q \geq 3$, q-Clique Coloring parameterized by the distance t to a caterpillar forest cannot be solved in time $\mathcal{O}^\star((q-\epsilon)^t)$, unless* SETH *fails.*

Proof. We give a reduction from q-List Coloring on triangle-free graphs parameterized by distance to linear forest. In this proof we use the phrases "q-colorable" as short for "can be properly colored with at most q colors", and "q-coloring" as short for "a proper coloring with at most q colors". To construct our instance of q-Clique Coloring, we will first describe the construction of a color selection gadget, and then describe how this gadget is attached to the rest of the graph. The description of the color selection gadget makes use of the famous Mycielski graphs (♣). For each $p \geq 2$, the graph M_p is p-colorable (but not $(p-1)$-colorable) and edge-critical, that is, the deletion of any edge of M_p leads to a $(p-1)$-colorable graph (see for instance [5, 25]). Moreover, $|V(M_p)| = 3 \cdot 2^{p-2} - 1$. We use the graph M'_p, obtained from M_p by the deletion of an arbitrary edge xy.

▷ Observation 9. Let M'_p be the graph obtained from M_p by the deletion of an edge xy. Then, M'_p is $(p-1)$-colorable, and in any $(p-1)$-coloring of M'_p, the vertices x and y receive the same color.

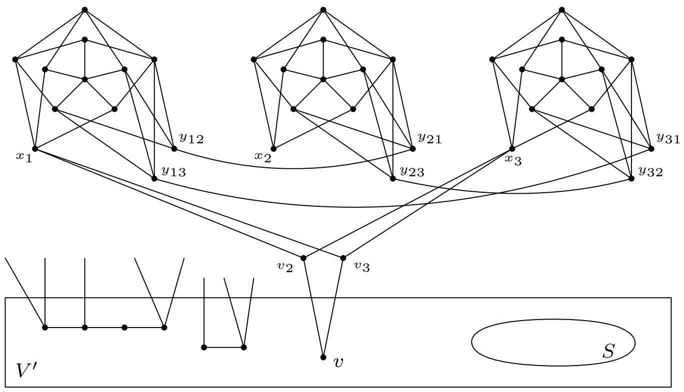

Figure 2 Here, $q = 3$ and $L(v) = \{1\}$. Note that $G' - (S \cup V(H_q))$ is a caterpillar forest.

Color selection gadget. We construct a gadget H_q in the following way. Consider q disjoint copies of M'_{q+1}. For $1 \leq i \leq q$, let $x_i y_i$ be the edge removed from M_{q+1} in order to obtain the ith copy of M'_{q+1}. For each i, add $q-1$ false twins to y_i. We denote these vertices by y_{ij}, with $1 \leq j \leq q$, $j \neq i$. Then delete the vertex y_i, for every i. Note that this graph is still q colorable and, by Observation 9, in every such q-coloring, for each i, the vertices x_i and y_{ij}, for all $j \neq i$, receive the same color. Now we add $\binom{q}{2}$ edges to connect the copies of M'_{q+1}: for $1 \leq i < j \leq q$, add the edge $y_{ij}y_{ji}$ to H_q. Note that H_q remains triangle-free after the addition of these edges, since for all $1 \leq i < j \leq q$, $N(y_{ij}) \cap N(y_{ji}) = \emptyset$. We will need the following property of the q-colorings of H_q.

▷ Claim (♣). The graph H_q is q-colorable. Moreover, in any q-coloring ϕ of H_q, $\phi(x_i) \neq \phi(x_j)$ for all $1 \leq i < j \leq q$.

We are now ready to describe the construction of our instance G' to q-Clique Coloring. Let (G, L) be an instance of q-List Coloring on triangle-free graphs that is at distance t from a linear forest. We construct G' as follows. Add a copy of G and a copy of H_q to G'. We denote by V' the set of vertices corresponding to $V(G)$ in G'. For each $v \in V'$, add $q - |L(v)|$ vertices adjacent to v. We denote these vertices by $\{v_j \mid j \notin L(v)\}$. Finally, make v_j adjacent to all the vertices of $\{x_\ell \mid \ell \neq j\}$. See Figure 2. Let $S \subseteq V(G)$ be a set such that $G - S$ is a linear forest and $|S| = t$. Then each connected component of $G' - (S \cup V(H_q))$ is a caterpillar and $|S \cup V(H_q)| = t + \mathcal{O}(1)$, since q is a constant.

▷ Claim (♣). (G, L) is a yes-instance to q-List Coloring if and only if G' is a yes-instance to q-Clique Coloring.

Now, if q-Clique Coloring admits an algorithm running in time $\mathcal{O}^\star((q-\epsilon)^{t'})$, for some $\epsilon > 0$, then we can solve q-List-Coloring in time $\mathcal{O}^\star((q-\epsilon)^{t+\mathcal{O}(1)}) = \mathcal{O}^\star((q-\epsilon)^t)$, contradicting SETH by Theorem 7, where t' and t denote the distance to a caterpillar forest and linear forest, respectively. ◀

4 Parameterized by Clique-width

In this section, we give an XP-time algorithm for Clique Coloring parameterized by clique-width, more precisely, parameterized by the equivalent measure module-width. We provide an algorithm that given an n-vertex graph G with one of its rooted branch decompositions $(T, \mathcal{L})$ of module-width w and an integer k, decides whether G has a clique coloring with k

colors in time $n^{f(w)}$, where $f(w) = 2^{2^{\mathcal{O}(w)}}$. Before we describe the algorithm, we give a high level outline of its main ideas. The algorithm is bottom-up dynamic programming along the given branch decomposition of the input graph. Let $t \in V(T)$. To keep the number of table entries bounded by something that is XP in the module-width, we group color classes into a number of *types* that is upper bounded by a function of w alone. Two color classes of the same type are interchangeable with respect to the underlying coloring being completable to a valid clique coloring of the whole graph. Partial solutions can then be described by remembering, for each type, how many color classes of that type there are. If the number of types is $f(w)$ for some function f, this gives an upper bound of $n^{f(w)}$ on the number of table entries at each node of the branch decomposition.

Let us discuss what kind of information goes into the definition of a type. We maintain information about cliques in G_t that are or may become monochromatic maximal cliques in some extension of the coloring at hand. It is not sufficient to consider only maximal cliques in G_t; given a maximal clique X in G_t, it may happen that in $\overline{V_t}$ there is a vertex v that is adjacent to a strict subset $Y \subset X$ of that clique, forming a maximal clique with Y – which does not fully contain X – in a supergraph of G_t. Considering the equivalence classes of $\sim_t$, this implies that the equivalence classes containing Y and the ones containing $X \setminus Y$ are disjoint. We therefore consider cliques X that are *maximal in the subgraph induced by the equivalence classes containing vertices of X*. We call such cliques X *eqc-maximal*, and observe that with a little extra information, we can keep track of the forming and disintegrating of eqc-maximal cliques along the branch decomposition. If an eqc-maximal clique is fully contained in some set of vertices (/color class) C, then we call it *potentially bad for C*. A potentially bad clique is described via its *profile*, which consists of the intersection pattern with the equivalence classes of $\sim_t$, and some extra information. At each node, there are at most $2^{\mathcal{O}(w)}$ profiles.

Equipped with this definition, we can define the notion of a t-type of a color class C, which is simply the subset of profiles at t, such that G_t contains a potentially bad clique with that C-profile. It immediately follows that the number of t-types is $2^{2^{\mathcal{O}(w)}}$. Now, colorings $\mathcal{C}_t$ of G_t are described by their *t-signature*, which records how many color classes of each type $\mathcal{C}_t$ has. There are at most $k^{f(w)}$ many t-signatures, where $f(w) = 2^{2^{\mathcal{O}(w)}}$, and this essentially bounds the runtime of the resulting algorithm by $n^{f(w)}$.

At the root node $\mathfrak{r} \in V(T)$, there is only one equivalence class, namely $V_{\mathfrak{r}} = V(G)$, and if in a coloring, there is a clique that is potentially bad for some color class, then it is indeed a monochromatic maximal clique. Therefore, at the root node, we only have to check whether there is a coloring all of whose color classes have no potentially bad cliques.

4.1 Potentially Bad Cliques

We now introduce the main concept used to describe color classes in partial solutions of our algorithms, namely potentially bad cliques. These are cliques that are monochromatic in some subgraph induced by a set of equivalence classes.

▶ **Definition 10** (Potentially Bad Clique). *Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$ and let $t \in V(T)$. A clique X in G_t is called* eqc-maximal (in G_t) *if it is maximal in $G_t[V(\mathsf{eqc}_t(X))]$. Let $C \subseteq V_t$ and let X be a clique in G_t. Then, X is called* potentially bad for C (in G_t)*, if X is eqc-maximal in G_t and $X \subseteq C$.*

Naturally, it is not feasible to keep track of *all* potentially bad cliques. We therefore capture the most vital information about potentially bad cliques in the following notion of a *profile*. For our algorithm, it is only important to know for a color class whether or not

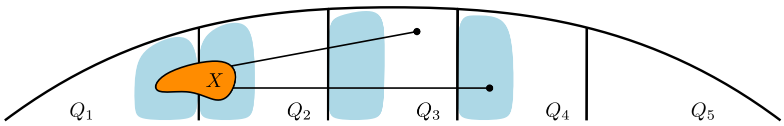

Figure 3 Illustration of the C-profile of a clique X that is potentially bad for a color class C, depicted as the shaded areas within the equivalence classes. In this case, we have that $\pi(X \mid C) = (\{Q_1, Q_2\}, \{Q_3, Q_4\})$.

it has some potentially bad clique with a given profile, rather than how many, or what its vertices are. This is key to reduce the amount of information we need to store about partial solutions. There are two components of a profile of a potentially bad clique X; the first one is the set of equivalence classes $\mathcal{Q}$ containing its vertices, and the second one consists of the equivalence classes $P \notin \mathcal{Q}$ that have a vertex that is complete to X. This is because, at a later stage, P may be merged with an equivalence class containing vertices of X (via the bubbles), in which case X is no longer potentially bad. We illustrate the following definition in Figure 3.

▶ **Definition 11** (Profile). *Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$ and let $t \in V(T)$. Let $C \subseteq V_t$ and let X be a clique in G_t that is potentially bad for C. The* C-profile *of X is a pair of subsets of $V_t/{\sim_t}$, $\pi(X \mid C) := (\mathcal{Q}, \mathcal{P})$, where*

$$\mathcal{Q} = \mathsf{eqc}_t(X) \text{ and } \mathcal{P} = \{P \in \overline{\mathsf{eqc}}_t(X) \mid \exists v \in P \colon X \subseteq N(v)\}.$$

We call the set of all pairs of disjoint subsets of $V_t/{\sim_t}$, where the first coordinate is nonempty, the profiles at t, *formally,* $\Pi_t := \{(\mathcal{Q}, \mathcal{P}) \mid \mathcal{Q}, \mathcal{P} \subseteq V_t/\sim_t \colon \mathcal{Q} \neq \emptyset \wedge \mathcal{Q} \cap \mathcal{P} = \emptyset\}$.

▷ Observation 12. Let $(T, \mathcal{L})$ be a rooted branch decomposition. For each $t \in V(T)$, there are at most $2^{\mathcal{O}(w)}$ profiles at t, where $w = \mathsf{mw}(T, \mathcal{L})$.

Let $t \in V(T) \setminus \mathrm{L}(T)$ be an internal node with children r and s and operator (H_t, η_r, η_s), and let $\pi_r \in \Pi_r$ and $\pi_s \in \Pi_s$ be a pair of profiles. We are now working towards a notion that precisely captures when and how a potentially bad clique in G_r for some $C_r \subseteq V_r$ with C_r-profile π_r can be merged with a potentially bad clique in G_s for some $C_s \subseteq V_s$ with C_s-profile π_s to obtain a potentially bad clique for $C_r \cup C_s$ in G_t. As it turns out, if this is possible, then the profile of the resulting clique only depends on π_r, π_s, and the operator of t. Note that for now, we focus on the case when the cliques in G_r and G_s are both nonempty, and we discuss the case when one of them is empty below.

Before we proceed with this description, we need to introduce some more concepts. We illustrate all of the following concepts in Figure 4. For a set of equivalence classes $\mathcal{S} \subseteq V_r/{\sim_r} \cup V_s/{\sim_s}$, its *bubble buddies at* t, denoted by $\mathsf{bb}_t(\mathcal{S})$, are the equivalence classes of $V_r/{\sim_r} \cup V_s/{\sim_s}$ that are in the same bubble as some equivalence class in $\mathcal{S}$, i.e.

$$\mathsf{bb}_t(\mathcal{S}) := \bigcup_{p \in \{r,s\}} \{Q_p \in V_p/{\sim_p} \mid \eta_p(Q_p) \in \eta_p(\mathcal{S} \cap V_p/{\sim_p})\}\,.$$

We call $\pi_r = (\mathcal{Q}_r, \mathcal{P}_r)$ and $\pi_s = (\mathcal{Q}_s, \mathcal{P}_s)$ *compatible* if $\mathcal{Q}_r \cup \mathcal{Q}_s$ is a maximal biclique in $H'_t(\pi_r, \pi_s) := H_t[(\mathcal{Q}_r \cup \mathcal{Q}_s) \cup ((\mathcal{P}_r \cup \mathcal{P}_s) \cap \mathsf{bb}_t(\mathcal{Q}_r \cup \mathcal{Q}_s))]$. As we show below, the notion of compatibility precisely captures the "merging behavior" of potentially bad cliques. Moreover, for π_r and π_s compatible, we can immediately construct the profile of the resulting potentially bad clique: the *merge profile* of π_r and π_s is the profile $\mu(\pi_r, \pi_s) = (\mathcal{Q}_t, \mathcal{P}_t)$ such that

- $\mathcal{Q}_t = \eta_r(\mathcal{Q}_r) \cup \eta_s(\mathcal{Q}_s)$ and
- $\mathcal{P}_t = \bigcup_{\{o,p\}=\{r,s\}} \{\eta(Q_p) \mid Q_p \in \mathcal{P}_p \setminus \mathsf{bb}_t(\mathcal{Q}_r \cup \mathcal{Q}_s) \colon \mathcal{Q}_o \subseteq N_{H_t}(Q_p)\}$.

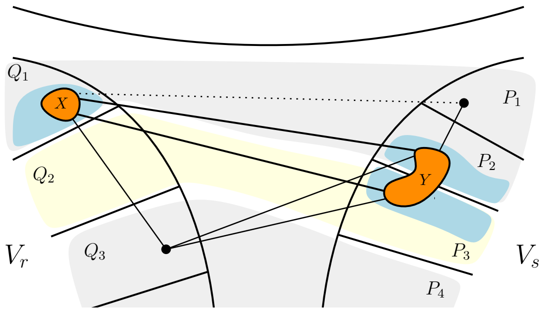

Figure 4 Merging a potentially bad clique X in G_r with a potentially bad clique Y in G_s to obtain a potentially bad clique in G_t. The color class at hand is depicted in blue and the gray and yellow areas show the (three) bubbles. Note that the equivalence classes P_1 and Q_2 are bubble buddies of $\mathsf{eqc}_r(X)$ and $\mathsf{eqc}_s(Y)$. Moreover, the types of X and Y are compatible, since $\{Q_1, P_2, P_3\}$ is a maximal biclique in $H_t[\{Q_1, P_1, P_2, P_3\}]$. Finally, note that the equivalence class of $\sim_t$ corresponding to the bubble containing Q_3 will have a vertex that is complete to the potentially bad clique $X \cup Y$.

▶ **Lemma 13 (♣).** *Let $t \in V(T) \setminus \mathrm{L}(T)$ be an internal node with children r and s and operator (H_t, η_r, η_s). For all $p \in \{r, s\}$, let $C_p \subseteq V_p$, let X_p be a clique in G_r that is potentially bad for C_p, and let $\pi_p := \pi(X_p \mid C_p) = (\mathcal{Q}_p, \mathcal{P}_p)$. If π_r and π_s are compatible, then $X_t := X_r \cup X_s$ is a clique that is potentially bad for $C_t := C_r \cup C_s$, and $\pi(X_t \mid C_t) = \mu(\pi_r, \pi_s)$.*

▶ **Lemma 14 (♣).** *Let $t \in V(T) \setminus \mathrm{L}(T)$ be an internal node with children r and s and operator (H_t, η_r, η_s). Let $C_t \subseteq V_t$, and let X_t be a clique in G_t that is potentially bad for C_t. For all $p \in \{r, s\}$, let $X_p := X_t \cap V_p$ and $C_p := C_t \cap V_p$. Suppose that for all $p \in \{r, s\}$, $X_p \neq \emptyset$. Then, for all $p \in \{r, s\}$, X_p is a potentially bad clique for C_p, and $\pi_r := \pi(X_r \mid C_r)$ and $\pi_s := \pi(X_s \mid C_s)$ are compatible.*

As mentioned above, we treat the case when a clique X_p in one of the children $p \in \{r, s\}$ remains potentially bad in G_t separately. This is because in that case, the notion of a maximal biclique in H_t' as defined above does not hold up very naturally. We formulate the analogous requirements for this case here, and we skip some of the details.

Let $t \in V(T) \setminus \mathrm{L}(T)$ be an internal node with children r and s and operator (H_t, η_r, η_s). Let $\pi_r \in \Pi_r$. We say that π_r is *liftable* if there is no $Q_s \in \mathsf{bb}_t(\mathcal{Q}_r)$ that is complete to $\mathcal{Q}_r$ in H_t, and $\mathsf{bb}_t(\mathcal{Q}_r) \cap \mathcal{P}_r = \emptyset$. The *lift profile* of π_r, denoted by $\lambda(\pi_r)$, is constructed as the merge profile of π_r with the empty set; i.e. we take $(\mathcal{Q}_s, \mathcal{P}_s) = (\emptyset, V_s/{\sim_s})$ and apply the definition given above, meaning $\lambda(\pi_r) = \mu(\pi_r, (\emptyset, V_s/{\sim_s}))$.

▶ **Lemma 15 (♣).** *Let $t \in V(T) \setminus \mathrm{L}(T)$ be an internal node with children r and s. Let $C_r \subseteq V_r$, $C_s \subseteq V_s$, let X_r be a clique in G_r, and let $\pi_r := \pi(X_r \mid C_r)$. Then, X_r is a potentially bad clique for $C_r \cup C_s$ in G_t if and only if X_r is a potentially bad clique for C_r in G_r and π_r is liftable, in which case $\pi_t(X_r \mid C_r \cup C_s) = \lambda(\pi_r)$.*

4.2 The type of a color class

We now describe the t-type of a color class C, which is the subset of profiles at t such that there is a clique in G_t that is potentially bad for C, with that C-profile. For our algorithm, two color classes with the same type will be interchangeable, therefore we only have to remember the number of color classes of each type.

▶ **Definition 16** (t-Type). *Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$, and let $t \in V(T)$. For a set $C \subseteq V_t$, the* t-type *of C, denoted by $\gamma_t(C)$ is*

$$\gamma_t(C) := \{\pi_t \in \Pi_t \mid \exists \text{ clique } X \text{ in } G_t \text{ which is potentially bad for } C \text{ and } \pi(X \mid C) = \pi_t\}.$$

With slight abuse of notation, we call the set $\Gamma_t = 2^{\Pi_t}$ of all subsets of profiles at t the t-types.

▷ Observation 17. Let $(T, \mathcal{L})$ be a rooted branch decomposition, and let $t \in V(T)$. There are at most $2^{2^{\mathcal{O}(w)}}$ many t-types, where $w := \mathsf{mw}(T, \mathcal{L})$.

The previous observation follows from Observation 12. In our algorithm we want to be able to determine the t-type of the union of a color class in G_r and a color class in G_s. This is done via the following notion of a merge type.

▶ **Definition 18** (Merge Type). *Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$, let $t \in V(T) \setminus \mathrm{L}(T)$ with children r and s. For a pair of an r-type $\gamma_r \in \Gamma_r$ and an s-type $\gamma_s \in \Gamma_s$, the* merge type *of γ_r and γ_s, denoted by $\mu(\gamma_r, \gamma_s)$, is the t-type obtained as follows.*

$$\begin{aligned}\mu(\gamma_r, \gamma_s) := \; & \{\mu(\pi_r, \pi_s) \mid \pi_r \in \gamma_r,\; \pi_s \in \gamma_s, \text{where } \pi_r \text{ and } \pi_s \text{ are compatible}\} \\ & \bigcup\nolimits_{p \in \{r,s\}} \{\lambda(\pi_p) \mid \pi_p \in \gamma_p, \text{where } \pi_p \text{ is liftable}\}\end{aligned}$$

The proof of the following lemma which shows that the merge type faithfully represents the merging of two color classes can be done using Lemmas 13, 14, and 15.

▶ **Lemma 19** (♣). *Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$, let $t \in V(T) \setminus \mathrm{L}(T)$ with children r and s. Let $C_r \subseteq V_r$ and $C_s \subseteq V_s$. Then, $\gamma_t(C_r \cup C_s) = \mu(\gamma_r(C_r), \gamma_s(C_s))$.*

4.3 The algorithm

We are now ready to describe the algorithm. As alluded to above, partial solutions at a node t, i.e. colorings of G_t, are described via the notion of a t-*signature* which records the number of color classes of each type in a coloring. If two colorings have the same t-signature, then they are interchangeable as far as our algorithm is concerned. We show that this information suffices to solve the problem in a bottom-up dynamic programming fashion.

▶ **Definition 20** (t-Signature). *Let k be a positive integer. Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$, let $t \in V(T)$, and let $\mathcal{C} = (C_1, \ldots, C_k)$ be a k-coloring of G_t. Then, $\sigma_{\mathcal{C}} \colon \Gamma_t \to \{0, 1, \ldots, k\}$ where $\forall \gamma_t \in \Gamma_t \colon \sigma_{\mathcal{C}}(\gamma_t) := |\{i \in \{1, \ldots, k\} \mid \gamma(C_i) = \gamma_t\}|$, is called the* t-signature *of $\mathcal{C}$. The set of* t-signatures *is defined as:*

$$\mathsf{sig}_t := \left\{\sigma_t \colon \Gamma_t \to \{0, 1, \ldots, k\} \;\middle|\; \sum\nolimits_{\gamma_t \in \Gamma_t} \sigma_t(\gamma_t) = k\right\}$$

The following bound on the number of t-signatures immediately follows from Observation 17, stating that the number of t-types is upper bounded by $2^{2^{\mathcal{O}(w)}}$.

▷ Observation 21. Let $(T, \mathcal{L})$ be a rooted branch decomposition of an n-vertex graph, and let $t \in V(T)$. There are at most $k^{2^{2^{\mathcal{O}(w)}}} \le n^{2^{2^{\mathcal{O}(w)}}}$ many t-signatures, where $w := \mathsf{mw}(T, \mathcal{L})$.

▷ Definition of the table entries. For each $t \in V(T)$ and $\sigma_t \in \mathsf{sig}_t$, we let $\mathsf{tab}[t, \sigma_t] = 1$ if and only if there is a k-coloring $\mathcal{C}$ of G_t such that $\sigma_{\mathcal{C}} = \sigma_t$.

▶ **Lemma 22** (♣). *Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$, and let $\mathfrak{r}$ be the root of T. G has a clique coloring with k colors if and only if $\mathsf{tab}[\mathfrak{r}, \sigma^\star] = 1$, where $\sigma^\star$ is the $\mathfrak{r}$-signature for which $\sigma^\star(\emptyset) = k$.*

▷ Leaves of T. Let $t \in \mathrm{L}(T)$ be a leaf node in T and let $v \in V(G)$ be the vertex such that $\mathcal{L}(v) = t$. We show how to compute the table entries $\mathsf{tab}[t, \cdot]$. Note that $G_t = (\{v\}, \emptyset)$, and that $\{v\}$ is the only equivalence class of $\sim_t$. To describe the types of color classes of G_t, observe that the only eqc-maximal clique in G_t is $\{v\} =: X_v$, which is potentially bad for $C_v := \{v\} = X_v$. In that case, we have that $\pi_v := \pi(X_v \mid C_v) = (\{v\}, \emptyset)$, and the type of color class C_v is $\{\pi_v\}$. The type of the remaining $k-1$ color classes is $\emptyset$, since they are all empty. Therefore, for each t-signature σ_t, we set $\mathsf{tab}[t, \sigma_t] := 1$ if and only if $\sigma_t(\{\pi_v\}) = 1$ and $\sigma_t(\emptyset) = k-1$.

Next, we move on to the computation of the table entries at internal nodes of the branch decomposition. To describe this part of the algorithm, we borrow the following notion of a *merge skeleton* from [22].

▶ **Definition 23** (Merge skeleton). *Let G be a graph and $(T, \mathcal{L})$ one of its rooted branch decompositions. Let $t \in V(T) \setminus \mathrm{L}(T)$ with children r and s. The* merge skeleton *of r and s is an edge-labeled complete bipartite graph $(\mathfrak{J}, \mathfrak{m})$ where*

- *$V(\mathfrak{J}) = \Gamma_r \cup \Gamma_s$, and*
- *for all $\gamma_r \in \Gamma_r$, $\gamma_s \in \Gamma_s$, $\mathfrak{m}(\gamma_r\gamma_s) = \mu(\gamma_r, \gamma_s)$.*

▷ Internal nodes of T. Let $t \in V(T) \setminus \mathrm{L}(T)$ be an internal node with children r and s. We discuss how to compute the table entries at t, assuming the table entries at r and s have been computed. Each coloring of G_t can be obtained from a coloring of G_r and a coloring of G_s, by merging pairs of color classes. Therefore, for each pair $\sigma_r \in \mathsf{sig}_r$, $\sigma_s \in \mathsf{sig}_s$ such that $\mathsf{tab}[r, \sigma_r] = 1$ and $\mathsf{tab}[s, \sigma_s] = 1$, we do the following. We enumerate all labelings of the edge set of the merge skeleton with numbers from $\{0, 1, \ldots, k\}$, with the following interpretation. If an edge $\gamma_r\gamma_s$ has label j, then it means that j color classes of r-type γ_r will be merged with j color classes of s-type γ_s; this gives j color classes of t-type $\mu(\gamma_r, \gamma_s) = \mathfrak{m}(\gamma_r\gamma_s)$. Each such labeling that respects the number of color classes available of each type will produce a coloring of G_t with some signature σ_t, which can then be read off the edge labeling. For all such σ_t, we set $\mathsf{tab}[t, \sigma_t] = 1$. We give the formal details in the full version.

The proof of the following lemma which asserts the correctness of our algorithm can be done via induction on the height of t and using Lemma 19.

▶ **Lemma 24** (♣). *Let G be a graph and $(T, \mathcal{L})$ one of its rooted branch decompositions, and let $t \in V(T)$. The above algorithm computes the table entries $\mathsf{tab}[t, \cdot]$ correctly, i.e. for each $\sigma_t \in \mathsf{sig}_t$, it sets $\mathsf{tab}[t, \sigma_t] = 1$ if and only if G_t has a k-coloring $\mathcal{C}$ with $\sigma_{\mathcal{C}} = \sigma_t$.*

The details of the runtime discussion (based on Observation 21 and the fact that $|V(T)| = \mathcal{O}(n)$) are deferred to the full version; correctness is shown in Lemma 24, and by Lemma 22, the solution to the problem can be read off the table entries at the root. Using memoization techniques, the above algorithm can return a coloring if one exists.

▶ **Theorem 25.** *There is an algorithm that given an n-vertex graph G together with one of its rooted branch decompositions $(T, \mathcal{L})$ and a positive integer k, decides whether G has a clique coloring with k colors in time $k^{2^{2^{\mathcal{O}(w)}}} \cdot n \le n^{2^{2^{\mathcal{O}(w)}}}$, where $w := \mathsf{mw}(T, \mathcal{L})$. If such a coloring exists, the algorithm can construct it.*

References

1 Thomas Andreae, Martin Schughart, and Zsolt Tuza. Clique-transversal sets of line graphs and complements of line graphs. *Discrete Mathematics*, 88(1):11–20, 1991.

2 Gábor Bacsó, Sylvain Gravier, András Gyárfás, Myriam Preissmann, and András Sebo. Coloring the maximal cliques of graphs. *SIAM Journal on Discrete Mathematics*, 17(3):361–376, 2004.

3 Gábor Bacsó and Zsolt Tuza. Clique-transversal sets and weak 2-colorings in graphs of small maximum degree. *Discrete Mathematics and Theoretical Computer Science*, 11(2):15–24, 2009.

4 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC 2007)*, pages 67–74, San Diego, California, USA, June 11–13 2007. ACM.

5 J. A. Bondy and U. S. R. Murty. *Graph Theory*. Springer, 2008.

6 C. N. Campos, Simone Dantas, and Célia Picinin de Mello. Colouring clique-hypergraphs of circulant graphs. *Electronic Notes in Discrete Mathematics*, 30:189–194, 2008.

7 Márcia R. Cerioli and André L. Korenchendler. Clique-coloring circular-arc graphs. *Electronic Notes in Discrete Mathematics*, 35:287–292, 2009.

8 Pierre Charbit, Irena Penev, Stéphan Thomassé, and Nicolas Trotignon. Perfect graphs of arbitrarily large clique-chromatic number. *Journal of Combinatorial Theory, Series B*, 116:456–464, 2016.

9 Maria Chudnovsky and Irene Lo. Decomposing and clique-coloring (diamond, odd-hole)-free graphs. *Journal of Graph Theory*, 86(1):5–41, 2017.

10 Manfred Cochefert and Dieter Kratsch. Exact algorithms to clique-colour graphs. In Viliam Geffert, Bart Preneel, Branislav Rovan, Julius Stuller, and A Min Tjoa, editors, *Proceedings of the 40th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2014)*, volume 8327 of *LNCS*, pages 187–198. Springer, 2014.

11 Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

12 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Transactions on Algorithms*, 12(3):41:1–41:24, 2016.

13 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

14 David Défossez. Clique-coloring some classes of odd-hole-free graphs. *Journal of Graph Theory*, 53(3):233–249, 2006.

15 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

16 Dwight Duffus, Bill Sands, Norbert Sauer, and Robert E. Woodrow. Two-colouring all two-element maximal antichains. *Journal of Combinatorial Theory, Series A*, 57(1):109–116, 1991.

17 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Intractability of clique-width parameterizations. *SIAM Journal on Computing*, 39(5):1941–1956, 2010.

18 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Clique-width III: Hamiltonian cycle and the odd case of graph coloring. *ACM Transactions on Algorithms*, 15(1):9:1–9:27, 2019.

19 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

20 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

21 Lars Jaffke and Bart M. P. Jansen. Fine-grained parameterized complexity analysis of graph coloring problems. In Dimitris Fotakis, Aris Pagourtzis, and Vangelis Th. Paschos, editors, *Proceedings of the 10th International Conference on Algorithms and Complexity, (CIAC 2017)*, volume 10236 of *LNCS*, pages 345–356, Athens, Greece, May 24–26 2017. Springer.

22 Lars Jaffke, Paloma T. Lima, and Daniel Lokshtanov. b-Coloring parameterized by clique-width. *CoRR*, abs/2003.04254, 2020.

23 Sulamita Klein and Aurora Morgana. On clique-colouring of graphs with few P_4's. *Journal of the Brazilian Computer Society*, 18(2):113–119, 2012.

24 Jan Kratochvíl and Zsolt Tuza. On the complexity of bicoloring clique hypergraphs of graphs. *Journal of Algorithms*, 45(1):40–54, 2002.

25 Lásló Lovász. *Combinatorial Problems and Exercises*. North-Holland Publishing Co., 1993.

26 Dániel Marx. Complexity of clique coloring and related problems. *Theoretical Computer Science*, 412(29):3487–3500, 2011.

27 Bojan Mohar and Riste Skrekovski. The Grötzsch theorem for the hypergraph of maximal cliques. *Electronic Journal of Combinatorics*, 6(1):128, 1999.

28 Irena Penev. Perfect graphs with no balanced skew-partition are 2-clique-colorable. *Journal of Graph Theory*, 81(3):213–235, 2016.

29 Michaël Rao. *Décompositions de graphes et algorithmes efficaces*. PhD thesis, University of Metz, 2006.

30 Michaël Rao. Clique-width of graphs defined by one-vertex extensions. *Discrete Mathematics*, 308(24):6157–6165, 2008.

31 Erfang Shan, Zuosong Liang, and Liying Kang. Clique-transversal sets and clique-coloring in planar graphs. *European Journal of Combinatorics*, 36:367–376, 2014.

32 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In Thore Husfeldt and Iyad A. Kanj, editors, *Proceedings of the 10th International Symposium on Parameterized and Exact Computation (IPEC 2015)*, volume 43 of *LIPIcs*, pages 17–29. Schloss Dagstuhl, 2015.