b-Coloring Parameterized by Clique-Width*

Lars Jaffke[1], Paloma T. Lima[1], and Daniel Lokshtanov[2]

[1]University of Bergen, Norway
{lars.jaffke,paloma.lima}@uib.no
[2]UC Santa Barbara, California, USA
daniello@ucsb.edu

July 19, 2022

Abstract

We provide a polynomial-time algorithm for b-Coloring on graphs of constant clique-width. This unifies and extends nearly all previously known polynomial time results on graph classes, and answers open questions posed by Campos and Silva [Algorithmica, 2018] and Bonomo et al. [Graphs Combin., 2009]. This constitutes the first result concerning structural parameterizations of this problem. We show that the problem is FPT when parameterized by the vertex cover number on general graphs, and on chordal graphs when parameterized by the number of colors. Additionally, we observe that our algorithm for graphs of bounded clique-width can be adapted to solve the Fall Coloring problem within the same runtime bound. The running times of the clique-width based algorithms for b-Coloring and Fall Coloring are tight under the Exponential Time Hypothesis.

1 Introduction

This paper settles open questions regarding the complexity of the b-Coloring problem on graph classes and initiates the study of its structural parameterizations. A *b-coloring* of a graph G with k colors is a partition of the vertices of G into k independent sets such that each of them contains a vertex that has a neighbor in all of the remaining ones. The *b-chromatic number* of G, denoted by $\chi_b(G)$, is the maximum integer k such that G admits a b-coloring with k colors. This notion was introduced by Irving and Manlove [34] to describe the behavior of the following color-suppressing heuristic for the Graph Coloring problem. We start with some proper coloring of the input graph G and try to iteratively suppress one of its colors. That is, for a given color c, we consider each vertex v of color c, and check if there is another color $c' \neq c$ available that does not appear in its neighborhood. If so, we assign vertex v the color c', observing that the coloring remains proper, and repeat this process for the remaining vertices of color c. If successful, we remove the color c from all vertices of G and decrease the number of colors by one. Once no color can be supressed by this procedure, the coloring at hand is a b-coloring of G, and in the worst case, this heuristic produces a coloring with $\chi_b(G)$ many colors.

Since then, the b-Coloring and b-Chromatic Number problems which given a graph G and an integer k ask whether G has a b-coloring with k colors and whether $\chi_b(G) \geq k$, respectively, have

*Based on an extended abstract that appeared STACS 2021 [36].

received considerable attention in the algorithms and complexity communities. Before we discuss these results, note that the b-Coloring and b-Chromatic Number problem are not as closely related as the Graph Coloring and Chromatic Number problems in terms of their (polynomial time) complexities. If we can solve Chromatic Number, then we can use this algorithm to solve Graph Coloring, since each n-vertex graph G has proper colorings with $\chi(G), \ldots, n$ colors. However, knowing $\chi_b(G)$ and $\chi(G)$ does not say anything about the existence of a b-coloring with $k \in \{\chi(G)+1, \ldots, \chi_b(G)-1\}$ colors. Therefore, the b-Coloring problem can be computationally harder on a graph class than the b-Chromatic Number problem. Trivially, if we know how to solve b-Coloring in polynomial time, we can solve b-Chromatic Number in polynomial time.

The b-Chromatic Number problem has been shown to be NP-complete in the general case [34], as well as on bipartite graphs [41], co-bipartite graphs [6], chordal graphs [29], and line graphs [8], and a lot of effort has been put into devising polynomial time algorithms for b-Coloring in various other classes of graphs.[1] These include trees [34], tree-cographs [6], and graphs with few P_4s, such as cographs and P_4-sparse graphs [5], P_4-tidy graphs [58], and $(q, q-4)$-graphs for constant q [10]. A common property shared by these graph classes is that they all have bounded *clique-width* [27, 28, 48, 57].[2]

The main contribution of this work is an algorithm that solves b-Coloring (and b-Chromatic Number) in polynomial time on graphs of constant clique-width. Besides unifying the above mentioned polynomial time cases, this extends the tractability landscape of these problems to larger graph classes, and answers two open problems stated in the literature.

Over a decade ago, Bonomo et al. [5] asked whether their polynomial time result for cographs can be extended to distance-hereditary graphs. Havet et al. [29] answered the question negatively by providing an NP-completeness proof for chordal distance-hereditary graphs. We observe, however, that their proof has a flaw and while it does prove the claimed statement for chordal graphs, it unfortunately fails to do so for distance-hereditary graphs. Our polynomial time algorithm for graphs of bounded clique-width in fact provides a positive answer to Bonomo et al.'s question, as distance-hereditary graphs have clique-width at most three [27]. In recent years, even subclasses of distance-hereditary graphs have received significant attention, for instance in the work of Campos and Silva [11]: they provide a polynomial time algorithm for claw-free block graphs, and ask whether this result can be generalized to block graphs. Our algorithm provides a positive answer to this question as well. Moreover, it extends the known algorithm for $(q, q-4)$-graphs [10] (for constant q) to all (q,t)-graphs for constants q and t with $q \geq 4$, $t \geq 0$, and either $q \leq 6$ and $t \leq q-4$, or $q \geq 7$ and $t \leq q-3$, by a theorem due to Makowsky and Rotics [48]. Similarly, it extends the polynomial time algorithm for P_4-tidy graphs [58] to the class of partner-limited graphs thanks to a result by Vanherpe [57]. We give an overview of the graph classes involved in the previous discussion in Figure 1.

Our algorithm runs in time $n^{2^{\mathcal{O}(w)}}$, where n denotes the number of vertices of the input graph which is given together with a clique-width w-expression. As consequences of results due to Fomin et al. [23] and Fomin et al. [24], we observe that b-Coloring parameterized by clique-width is W[1]-hard, and that the exponential dependence on w in the degree of the polynomial cannot be

[1] In many of the following references, the results are stated for b-Chromatic Number instead of b-Coloring; however the algorithms for b-Coloring follow from the algorithms for b-Chromatic Number together with the fact that these graph classes are *b-continuous* [6, 22, 58], meaning that they have b-colorings any number $k \in \{\chi(G), \ldots, \chi_b(G)\}$ of colors, and the fact that Chromatic Number is solvable in polynomial time on these graph classes (for instance via [21, 59]).

[2] To the best of our knowledge, the only polynomial time result for graphs of unbounded clique-width so far concerns graphs of large girth. In particular, Campos et al. [9] showed that b-Chromatic Number is polyomial-time solvable on graphs of girth at least 7.

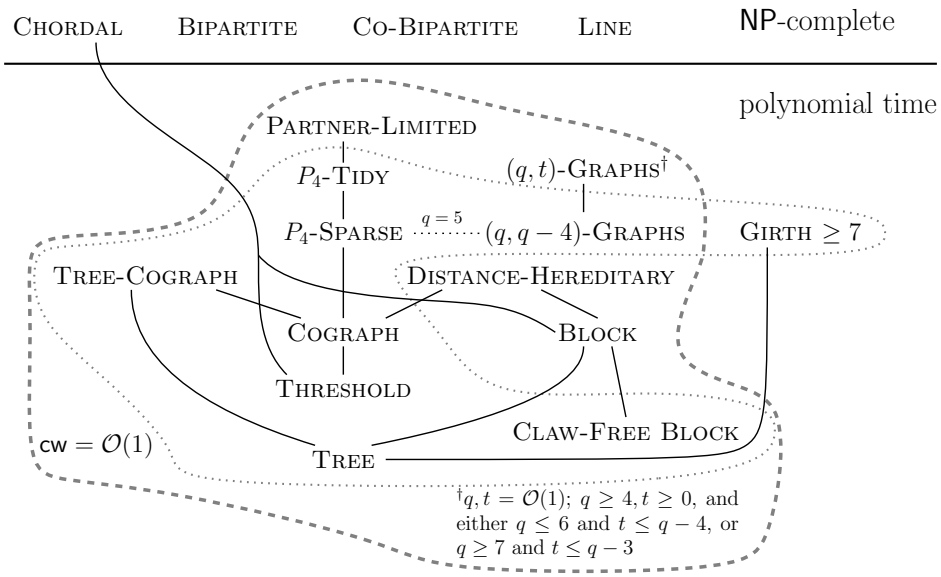

Figure 1: Some graph classes on which the complexities of b-Coloring and b-Chromatic Number problem were studied. Whenever two classes are connected by a line, the upper one contains the lower one. All NP-hardness results hold for b-Chromatic Number and all polynomial time results, except the one for graphs of girth at least seven, hold for b-Coloring. The inner bottom area (dotted line) shows classes for which polynomial time algorithms were previously known and the outer area (dashed line, labeled $\mathsf{cw} = \mathcal{O}(1)$) shows on which classes our algorithm can be applied.

avoided unless the Exponential Time Hypothesis (ETH) fails. Concretely, an algorithm running in time $n^{2^{o(w)}}$ would refute ETH.

From the point of view of parameterized complexity, Panolan et al. [50] showed that b-Chromatic Number parameterized by the number of colors is W[1]-hard. However, this problem may even be harder, since so far no XP-algorithm is known. Recently, Aboulker et al. [1] showed that the more restrictive b-Chromatic Core problem parameterized by the number of colors (which has a brute-force XP-algorithm, see e.g. [20]) remains W[1]-hard.

It is therefore natural to ask which additional restrictions can be imposed to obtain parameterized tractability results. For instance, an open problem posed by Sampaio [53] (see also [55]) asks whether b-Coloring parameterized by the number of colors is FPT on chordal graphs. We answer this question in the affirmative. Other restricted cases that have been considered in the literature target specific numbers of colors that depend on the input graph. The Dual b-Coloring problem, which asks if an input n-vertex graph has a b-coloring with $n-k$ colors, is FPT parameterized by k [30]. Moreover, deciding if a graph G has a b-coloring with $k = \Delta(G)+1$ colors, which is an upper bound on $\chi_b(G)$, is FPT parameterized by k [50, 53], while the case $k = \Delta(G)$ is XP and for every fixed $p \geq 1$, the case $k = \Delta(G) - p$ is NP-complete for $k = 3$ [35].

Another novelty aspect of our XP-algorithm parameterized by clique-width is that it is the first result about *structural parameterizations* of the b-Coloring and b-Chromatic Number problems. In all previously known polynomial time cases the algorithms only work if the input graph has some prescribed structure. Our algorithm works on all graphs, albeit with a prohibitively slow runtime on graphs of large clique-width. In this vein, we round off our work with an FPT-result for another lead player among structural parameterizations, the *vertex cover number* of a graph; a parameter often referred to as the *Drosophila* of parameterized complexity.

Fall Coloring. A *fall coloring* is a special type of b-coloring where *every* vertex needs to have at least one neighbor in all color classes except its own. In other words, it is a partition of the vertex set of a graph into independent dominating sets. As a standalone notion, fall coloring has been introduced by Dunbar et al. [19]. However, since the corresponding Fall Coloring problem falls in the category of locally checkable vertex partitioning problems, it has been shown in earlier work of Telle and Proskurowski [56] to be FPT parameterized by the tree-width of the input graph, as well as FPT parameterized by clique-width plus the number of colors by Gerber and Kobler [25] (see also [7]), and by Heggernes and Telle [31] to be NP-complete for fixed number of colors. Fall Coloring remains hard further restricted to bipartite [43, 44, 54], chordal [54], or planar [44] graphs. On the other hand, even with unbounded number of colors, it is known to be solvable in polynomial time on strongly chordal graphs [47, 26], threshold graphs and split graphs [49]. In all of these cases, one simply checks whether the chromatic number of the input graph is equal to its minimum degree plus one. To the best of our knowledge, these are the only known polynomial time cases.

We adapt our algorithm for b-Coloring on graphs of bounded clique-width to solve Fall Coloring, and therefore show that the latter problem is as well solvable in time $n^{2^{\mathcal{O}(w)}}$, where w denotes the clique-width of a given decomposition of the input graph. By a simple reduction, we show that Fall Coloring is also W[1]-hard in this parameterization and that an $n^{2^{o(w)}}$-time algorithm for it would refute ETH.

Vertex Coloring Problems Parameterized by Clique-Width. We briefly touch on differences in the complexities of vertex coloring problems of graphs when parameterized by clique-width. While the standard Graph Coloring problem, asking for a proper coloring of the input graph, is XP-time solvable parameterized by clique-width [21, 59], some of its generalizations are NP-complete on graphs of constant clique-width. In the List Coloring problem we are given a graph G and for each of its vertices v a list $L(v)$ of colors, and the question is whether G has a proper coloring such that each vertex is assigned a color from its list. This problem is NP-complete on the (not disjoint) union of two complete graphs [38]. We can see that such graphs have bounded clique-width for instance by observing that they do not contain a path on four vertices as an induced subgraph, and are therefore cographs, which have clique-width at most two [16]. In the related Precoloring Extension problem, we are given a graph, some of whose vertices already received a color, and the question is whether this coloring can be extended to a proper coloring of the entire graph. The following standard reduction from List Coloring, starting with a graph that is the union of two complete graphs, shows that this variant is NP-complete on graphs of constant clique-width as well. Take the graph G together with the lists $L(\cdot)$, and construct a graph H by adding to G, for each vertex $v \in V(G)$ and each color $c \notin L(v)$, a new vertex x_v^c which is adjacent only to v and assigned color c. It is not difficult to see that this precoloring of H can be extended to the remainder of its vertices if and only if G has a list coloring using the lists $L(\cdot)$. Moreover, adding pendant vertices to a graph does not increase its clique-width.

Belmonte et al. [3] showed that the Grundy Coloring problem, which asks for a linear order of the vertices that maximizes the number of colors used by the greedy coloring heuristic, is NP-complete on graphs of constant clique-width. This nicely contrasts our XP-algorithm for b-Coloring, since both the b-Coloring and the Grundy Coloring problems are rooted in the theoretical analysis of graph coloring heuristics.

Very recently, Jaffke et al. [37] showed that the Clique Coloring problem, asking for a vertex coloring without monochromatic maximal cliques, is XP parameterized by clique-width as well. The question whether Clique Coloring parameterized by clique-width is W[1]-hard remains open.

Sketch of the algorithm. Let us discuss how we obtain our XP-algorithm parameterized by clique-width. First, we consider a branch decomposition of the input graph G of bounded *module-width* w which is equivalent to clique-width and has the following property. At each node t of the branch decomposition we have a subgraph G_t of G whose vertex set can be partitioned into at most w equivalence classes with respect to their neighborhood outside of G_t. For the purpose of our dynamic programming algorithm, it suffices to describe colorings by the way each of their color classes interacts with these equivalence classes. In the Graph Coloring problem, it is enough to describe a color class according to its intersection with the equivalence classes of G_t alone [21, 59] (see also [24]). For the b-Coloring problem, we additionally have to ensure that eventually, each color class indeed has a b-vertex. The challenge is to do so without explicitly remembering which color classes a vertex has already seen in its neighborhood – this would result in prohibitively large tables. We overcome this difficulty by a symmetry breaking trick that instead stores, for each color class, a *demand* to the future neighbors of the equivalence classes which – if fulfilled – guarantees that the *other* color classes can have b-vertices in the end.

2 Preliminaries

Graphs. All graphs considered here are simple and finite. For a graph G we denote by $V(G)$ and $E(G)$ the vertex set and edge set of G, respectively. For an edge $e = uv \in E(G)$, we call u and v the *endpoints* of e and we write $u \in e$ and $v \in e$.

For two graphs G and H, we say that G is a *subgraph* of H, written $G \subseteq H$, if $V(G) \subseteq V(H)$ and $E(G) \subseteq E(H)$. For a set of vertices $S \subseteq V(G)$, the *subgraph of G induced by S* is $G[S] := (S, \{uv \in E(G) \mid u, v \in S\})$.

For a graph G and a vertex $v \in V(G)$, the set of its *neighbors* is $N_G(v) := \{u \in V(G) \mid uv \in E(G)\}$, and the *degree* of v is $\deg_G(v) := |N_G(v)|$. The *closed neighborhood* of v is $N_G[v] := \{v\} \cup N_G(v)$. For a set $X \subseteq V(G)$, we let $N_G(X) := \bigcup_{v \in X} N_G(v) \setminus X$ and $N_G[X] := X \cup N_G(X)$. In all these cases, we may drop G as a subscript if it is clear from the context. A graph is called *subcubic* if all its vertices have degree at most three.

A graph G is *connected* if for all 2-partitions (X, Y) of $V(G)$ with $X \neq \emptyset$ and $Y \neq \emptyset$, there is a pair $x \in X$, $y \in Y$ such that $xy \in E(G)$. A *connected component* of a graph is a maximal connected subgraph. A connected graph is called a *cycle* if all its vertices have degree two. A connected graph is called a *tree* if it has no cycle as a subgraph. In a tree T, the vertices of degree one are called the *leaves* of T, denoted by $\mathrm{L}(T)$, and the vertices in $V(T) \setminus \mathrm{L}(T)$ are the *internal vertices* of T. A tree of maximum degree at most two is a *path* and the leaves of a path are called its *endpoints*. If P is a path with endpoints u and v, then we say that P is a *path from u to v*. The *length* of a path is the number of its edges. For a graph G and a pair of vertices $u, v \in V(G)$, we denote by $\mathrm{dist}_G(u, v)$ the length of the shortest path between u and v in G.

A tree T is called *rooted*, if there is a distinguished vertex $r \in V(T)$, called the *root* of T, inducing an ancestral relation on $V(T)$: for a vertex $v \in V(T)$, if $v \neq r$, the neighbor of v on the path from v to r is called the *parent* of v, and all other neighbors of v are called its *children*. For a vertex $v \in V(T) \setminus \{r\}$ with parent p, the *subtree rooted at v*, denoted by T_v, is the subgraph of T induced by all vertices that are in the same connected component of $(V(T), E(T) \setminus \{vp\})$ as v. We define $T_r := T$. A tree T is called a *caterpillar* if it contains a path $P \subseteq T$ such that all vertices in $V(T) \setminus V(P)$ are adjacent to a vertex in P.

For a graph H, we say that a graph G is *H-free* if G does not contain H as an induced subgraph. For a set of graphs $\mathcal{H}$, we say that G is *$\mathcal{H}$-free* if G is H-free for all $H \in \mathcal{H}$. For an integer $k \geq 3$, let C_k denote a cycle on k vertices. A graph G is called *chordal* if it is $\{C_n \mid n \geq 4\}$-free. A graph

G is called *distance-hereditary* if for each connected induced subgraph H of G, and each pair of vertices $u, v \in V(H)$, $\text{dist}_H(u, v) = \text{dist}_G(u, v)$.

A set of vertices $S \subseteq V(G)$ of a graph G is called an *independent set* if $E(G[S]) = \emptyset$. A set of vertices $S \subseteq V(G)$ is a *vertex cover* in G if $V(G) \setminus S$ is an independent set in G. A set of vertices $S \subseteq V(G)$ is a *clique* in G if $E(G[S]) = \{uv \mid u, v \in S\}$.

A graph G is called *bipartite* if its vertex set can be partitioned into two nonempty independent sets, which we will refer to as a *bipartition* of G.

Notation for Equivalence Relations. 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 $[x]_\sim$ or simply $[x]$ if $\sim$ is clear from the context, 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. We give the basic definitions of parameterized complexity that are relevant to this work and refer to [17, 18] 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 $f \colon \mathbb{N} \to \mathbb{N}$ 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.

The concept analogous to NP-hardness in parameterized complexity is that of $\mathsf{W}[1]$-hardness, whose formal definition we omit. The basic assumption is that $\mathsf{FPT} \neq \mathsf{W}[1]$, under which no $\mathsf{W}[1]$-hard problem admits an FPT-algorithm. For more details, see [17, 18].

Exponential Time Hypothesis. The 3-SAT problem asks whether a given boolean formula in conjunctive normal form with clauses of size at most three has a truth assignment to its variables that lets the formula evaluate to true. In 2001, Impagliazzo, Paturi, and Zane [32, 33] conjectured that any algorithm for the 3-SAT problem requires exponential time. This conjecture is known as the *Exponential Time Hypothesis* (ETH) whose plausibility stems from the fact that despite numerous efforts, a subexponential-time algorithm for 3-SAT remains elusive. It can be stated as follows.

Conjecture (ETH [32, 33]). There is no algorithm that solves each instance of 3-SAT on n variables in time $2^{o(n)}$.

This conjecture initiated a rich theory of hardness results conditioned on ETH (see e.g. the survey [45] and [17, Chapter 14]), allowing for more precise lower bounds than the ones obtained from assumptions such as $\mathsf{P} \neq \mathsf{NP}$ or $\mathsf{FPT} \neq \mathsf{W}[1]$.

2.1 Clique-Width, Branch Decompositions, and Module-Width

We first define clique-width, introduced by Courcelle, Engelfriet, and Rozenberg [15], and then the equivalent measure of *module-width* that we will use in our algorithm. We keep the definition of clique-width slightly informal and refer to [15, 16] for more details. The reason why we choose module-width over clique-width is that module-width allows for a slightly more compact description of our algorithm, since it suffices to consider a single operation in the dynamic programming instead of several.

Let G be a graph. The *clique-width* of G, denoted by $\mathsf{cw}(G)$, is the minimum number of labels $\{1, \ldots, k\}$ needed to obtain G using the following four operations:

(i) Create a new graph consisting of a single vertex labeled i.

(ii) Take the disjoint union of two labeled graphs G_1 and G_2.

(iii) Add all edges between pairs of vertices of label i and label j.

(iv) Relabel every vertex labeled i to label j.

We now turn to the definition of module-width which is based on the notion of a rooted branch decomposition.

Definition 2.1 (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 $\mathcal{L}\colon V(G) \to \mathrm{L}(T)$. If T is a caterpillar, then $(T, \mathcal{L})$ is called *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 $V_t := \{v \in V(G) \mid \mathcal{L}(v) \in \mathrm{L}(T_t)\}$, $\overline{V_t} := V(G) \setminus V_t$, and $G_t := G[V_t]$.

Module-width is attributed to Rao [51, 52].[3] On a high level, the module-width of a rooted branch decomposition measures, at each of its nodes t, the 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. This naturally groups the vertices of V_t into equivalence classes.

Definition 2.2 (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 $\mathsf{mw}(T, \mathcal{L}) := \max_{t \in V(T)} |V_t/{\sim_t}|$. The *module-width of* G, denoted by $\mathsf{mw}(G)$, is the minimum module width over all rooted branch decompositions of G.

Theorem 2.3 (Rao, Thm. 6.6 in [51]). *For any graph G, $\mathsf{mw}(G) \leq \mathsf{cw}(G) \leq 2 \cdot \mathsf{mw}(G)$, 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 $n = |V(G)|$.*

The operator (H_t, η_r, η_s) of node t with children r and s. 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 $\sim_s$. First, it is clear that $V_t = V_r \cup V_s$. Since G_r and G_s are induced subgraphs of G_t, we furthermore know that $E(G_t[V_r]) = E(G_r)$ and $E(G_t[V_s]) = E(G_s)$, so it remains to describe the edges between V_r and V_s. By the definition of module-width, we know that each pair of vertices $u, v \in V_r$ with $u \sim_r v$ has the same neighborhood in $\overline{V_r} = V_s \cup \overline{V_t}$. Hence, for each vertex $z \in V_s$, we know that *either both or neither* of u and v are adjacent to z. In other words, for each pair $Q_r \in V_r/{\sim_r}$, $Q_s \in V_s{\sim_s}$, either all edges between each pair of a vertex from Q_r and a vertex from Q_s are present in G_t, or

[3] Note that in [52], module-width is referred to as *modular-width* which usually has a different meaning, see e.g. [13].

none of them. This can be described by a bipartite graph H_t on bipartition $(V_r/\sim_r, V_s/\sim_s)$ with $[u]_{\sim_r}[v]_{\sim_s} \in E(H_t)$ if and only if $uv \in E(G_t)$. To summarize,

$$\begin{aligned} E(G_t) &= E(G_r) \cup E(G_s) \cup F \text{ where} \\ F &= \{uv \mid u \in V_r, v \in V_s, \{[u]_{\sim_r}, [v]_{\sim_s}\} \in E(H_t)\}. \end{aligned}$$

By roughly the same reasoning, we can observe that the equivalence relation $\sim_t$ *coarsens* the equivalence relations $\sim_r$ and $\sim_s$. Consider again vertices $u, v \in V_r$ such that $u \sim_r v$. Then, $N(u) \cap \overline{V_r} = N(v) \cap \overline{V_r}$, and since $V_r \subseteq V_t$ we have that $\overline{V_t} \subseteq \overline{V_r}$, which implies that $N(u) \cap \overline{V_t} = N(v) \cap \overline{V_t}$, so $u \sim_t v$. However, it may well be that there are vertices $u, v \in V_r$ with $u \not\sim_r v$, but $u \sim_t v$: this is the case when u and v have the same neighbors in $\overline{V_t}$, but different neighbors in V_s. Lastly, note that there may also be vertices $v \in V_r$ and $z \in V_s$ such that $v \sim_t z$.

We have argued that each equivalence class of $\sim_t$ can be obtained by taking a subset of equivalence classes of $\sim_r$ and $\sim_s$, and joining them (in what we call a 'bubble' below). Formally, there is a partition $\mathcal{P} = \{P_1, \ldots, P_h\}$ of $V(H_t) = V_r/\sim_r \cup V_s/\sim_s$ such that $V_t/\sim_t = \{Q_1, \ldots, Q_h\}$, where for $1 \leq i \leq h$, $Q_i = \bigcup_{Q \in P_i} Q$. For each $1 \leq i \leq h$, 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.

2.2 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). (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. The *restriction* of a coloring $\mathcal{C} = (C_1, \ldots, C_k)$ to a vertex set $S \subseteq V(G)$, is $\mathcal{C}|_S \coloneqq (C_1 \cap S, \ldots, C_k \cap S)$. In this case we say conversely that $\mathcal{C}$ *extends* $\mathcal{C}|_S$.

Whenever convenient, we may 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 $\phi|_S$.

A proper coloring $(C_1, \ldots, C_k)$ is called a *b-coloring*, if for all $i \in \{1, \ldots, k\}$, there is a vertex $v_i \in C_i$, called *b-vertex of color* i, such that for all $j \in \{1, \ldots, k\} \setminus \{i\}$, $N_G(v_i) \cap C_j \neq \emptyset$. In this work, we study the following computational problem.

b-Coloring

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

We sometimes denote a b-coloring $\mathcal{C} = (C_1, \ldots, C_k)$ by $(\mathcal{C}, B = \{v_1, \ldots, v_k\})$, where for all $i \in \{1, \ldots, k\}$, v_i is a b-vertex of color i. In this case, B can be understood as the set containing a *witness b-vertex* for each color class.

The following definition will be key to the algorithms presented in the next sections.

Definition 2.4 (Partial *b*-Coloring). Let G be a graph and $k \in \mathbb{N}$. For an induced subgraph H of G, a *partial b-coloring* of H is a pair $(\mathcal{C}, B)$ of a proper coloring $\mathcal{C} = (C_1, \ldots, C_k)$ of H and a subset $B \subseteq V(H)$ such that for all $i \in [k]$, $|C_i \cap B| \leq 1$. We call the vertices in B the *partial b-vertices*.

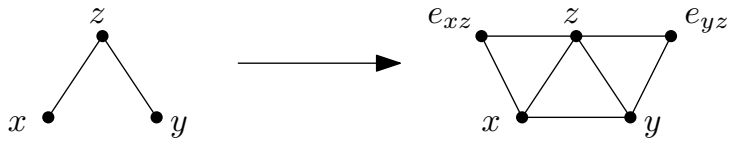

Figure 2

2.3 Distance-hereditary graphs

In their work on P_4-sparse graphs, Bonomo et al. [5] asked whether b-Coloring is polynomial-time solvable on the class of distance-hereditary graphs. Havet et al. [29] claimed to answer this question in the negative way, showing that b-Coloring is NP-complete on chordal distance-hereditary graphs. Their proof, however, contains a flaw and the graph constructed in their reduction, even though indeed chordal, fails to be distance-hereditary. In what follows, we briefly describe their reduction and argue that the graph constructed is not distance-hereditary.

The reduction presented in [29] is from 3-Edge Coloring restricted to the class of 3-regular graphs. Given an instance G for 3-Edge Coloring with $V(G) = \{v_1, \ldots, v_n\}$, they construct a graph H as follows. The vertex set of H contains a copy of $V(G)$ plus one vertex associated with each edge of G. We denote by e_{xy} the vertex corresponding to the edge xy. The vertices of $V(G)$ form a clique in H, the vertices corresponding to edges form an independent set, and for each edge $xy \in E(G)$, the vertex e_{xy} is adjacent to the copy of x and y in H. The connected component of H induced by these vertices is therefore a split graph. Finally, they add three disjoint copies of $K_{1,n+2}$ to H. It is thus easy to see that H is a chordal graph. However, let xz and yz be two edges of G sharing one endpoint. Then the subgraph of H induced by $\{x, y, z, e_{xz}, e_{yz}\}$ is isomorphic to a gem (see Figure 2). As shown by Bandelt and Mulder [2], distance-hereditary graphs are gem-free graphs. This shows that the graph H is not a distance-hereditary graph.

2.4 Parameterized by vertex cover

In this subsection we prove that b-Coloring is FPT when parameterized by vertex cover. We will do so by providing a $2^{\mathcal{O}(\mathsf{tw} \cdot k)}n$ time algorithm for the problem parameterized by the tree-width of the input graph plus number of colors. The result for vertex cover will then follow from the fact that the vertex cover number of a graph is always at most its tree-width, and a b-coloring of a graph with vertex cover ℓ can have at most $\ell + 1$ many colors. Indeed, either all b-vertices are contained in the vertex cover, in which case there are at most ℓ of them, or there is one outside, whose degree is at most ℓ, and hence it can see at most ℓ colors in its neighborhood.

Definition 2.5. Let G be a graph. A *tree decomposition* of G is a pair $(T, \mathcal{B} = \{B_t \mid t \in V(T)\})$, where T is a tree, and the sets in $\mathcal{B}$ are called *bags*, satisfying the following conditions.

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

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

(iii) For each $v \in V(G)$, $T[\{t \in V(T) \mid v \in B_t\}]$ is connected.

The *width* of a tree decomposition is $\max_{t \in V(T)} |B_t| - 1$ and the *tree-width* of G is the minimum width over all its tree decompositions.

Definition 2.6. A tree decomposition $(T, \mathcal{B} = \{B_t \mid t \in V(T)\})$ of a graph G is called *nice* if T is a rooted tree and each node $t \in V(T)$ is one of the following types:

Leaf: t is a leaf of T and $B_t = \emptyset$.

Introduce: t has a single child s and $B_t = B_s \cup \{v\}$ for some $v \in V(G)$; we say that v is *introduced* at t.

Forget: t has a single child s and $B_s = B_t \cup \{v\}$ for some vertex $v \in B_t$; we say that v is *forgotten* at t.

Join: t has two children, s_1 and s_2, and $B_t = B_{s_1} = B_{s_2}$.

For $t \in V(T)$, we let T_t denote the subtree of T rooted at t; we let $V_t = \bigcup_{s \in V(T_t)} B_s$ and $G_t = G[V_t]$.

Theorem 2.7 (Korhonen [40]). *There is an algorithm that given a graph G on n vertices and an integer k, in $2^{\mathcal{O}(w)}n$ time either outputs a tree decomposition of G of width at most $2k+1$ or concludes that the tree-width of G is more than k.*

Lemma 2.8 (Kloks [39], verbatim from [17]). *If a graph G admits a tree deecomposition of width at most k, then it also admits a nice tree decomposition of width at most k. Moreover, given a tree decomposition $(T, \mathcal{B})$ of G of width at most k, one can in time $\mathcal{O}(k^2 \cdot \max\{|V(G)|, |V(T)|\})$ find a nice tree decomposition of G that has at most $\mathcal{O}(k|V(G)|)$ nodes.*

Proposition 2.9. *b-Coloring can be solved in $2^{\mathcal{O}(\mathsf{tw} \cdot k)}n$ time, where n is the number of vertices and tw the tree-width of the input graph, and k the number of colors.*

Proof. By Theorem 2.7 and Lemma 2.8 we can assume that we have a nice tree decomposition $(T, \mathcal{B} = \{B_t \mid t \in V(T)\})$ of G of width $w \leq 2\mathsf{tw}+1$ and with $\mathcal{O}(wn)$ nodes after spending $2^{\mathcal{O}(\mathsf{tw})}n$ time. We do bottom-up dynamic programming along $(T, \mathcal{B})$.

The table entries of the dynamic programming and their invariant are as follows. Let $t \in V(T)$ be a node of $(T, \mathcal{B})$. Then, we let $\mathsf{tab}_t[\gamma, C, P, \sigma] = 1$ if there is a partial b-coloring γ_t of G_t with the following properties, and 0 otherwise:

(i) $\gamma \colon B_t \to [k]$ is a proper coloring with $\gamma = \gamma_t|_{B_t}$.

(ii) $P \subseteq B_t$ is the set of partial b-vertices of γ_t that are contained in B_t.

(iii) $\sigma \colon P \to 2^{[k]}$ is a map such that for each $p \in P$, $\sigma(p)$ is the set of colors that appear in the neighborhood of p in γ_t.

(iv) $C \subseteq [k]$, where $\gamma(P) \subseteq C$, is the set of colors that have a partial b-vertex in γ_t. Each partial b-vertex not contained in B_t is a b-vertex.

We observe that at each node $t \in V(T)$ there are at most $2^{\mathcal{O}(wk)}$ table entries; moreover, once the table entries have been computed correctly, we know that G has a b-coloring with k colors if and only if at the root $\mathfrak{r}$ of T there is a table entry $\mathsf{tab}_{\mathfrak{r}}[\gamma, P, \sigma, C] = 1$, where $C = [k]$, and for all $p \in P$, $\sigma(p) = [k]$. We discuss how to compute the table entries for each type of node in $(T, \mathcal{B})$; we assume that initially all table entries are set to 0.

Leaf. If t is a leaf, then it is trivial. For technical reasons, we assume that there is a table entry $\mathsf{tab}_t[\emptyset, \emptyset, \emptyset, \emptyset] = 1$.

Introduce. If t is an introduce node, let s be its child and v the vertex introduced at v. Let γ be a proper k-coloring of $G[B_t]$. Each neighbor of v that is a partial b-vertex for its color has to mark the color $\gamma(v)$ as seen in its neighborhood. To this end, for each $P \subseteq B_t$ and $\sigma\colon P \to 2^{[k]}$, we say that a map $\sigma_s\colon P \to 2^{[k]}$ is *compatible with* σ if for all $p \in P \cap N(v)$, $\sigma(p) = \sigma_s(p) \cup \{\gamma(v)\}$, and for all $p \in P \setminus N[v]$, $\sigma(p) = \sigma_s(p)$.

We first discuss how to deal with the case when v is not a partial b-vertex for its color. We consider each set $C \subseteq [k]$, each $P \subseteq B_t \setminus \{v\}$, and each map $\sigma\colon P \to 2^{[k]}$. We set $\mathsf{tab}_t[\gamma, P, \sigma, C]$ to 1 if there is a map $\sigma_s\colon P \to 2^{[k]}$ compatible with σ and such that $\mathsf{tab}_s[\gamma|_{B_s}, P, \sigma_s, C] = 1$.

Next, we consider the case when v is a partial b-vertex for its color. Then we consider each set $C \subseteq [k]$ with $\gamma(v) \in C$, and each $P \subseteq B_t$ with $v \in P$, and each map $\sigma\colon P \to 2^{[k]}$ where $\sigma(v) = \gamma(N(v))$. We set $\mathsf{tab}_t[\gamma, P, \sigma, C]$ to 1 if there is a map $\sigma_s\colon P \setminus \{v\} \to 2^{[k]}$ that is compatible with σ and such that $\mathsf{tab}_s[\gamma|_{B_s}, P \setminus \{v\}, \sigma_s, C \setminus \{\gamma(v)\}] = 1$.

Forget. If t is a forget node, let s be its child and v be the vertex forgotten at t. The only thing we have to ensure here is that if v was a partial b-vertex for its color, then in fact it was a b-vertex for its color. We proceed as follows. We set $\mathsf{tab}_t[\gamma, P, \sigma, C]$ to 1 if $\mathsf{tab}_s[\gamma_s, P_s, \sigma_s, C] = 1$ where γ_s is an extension of γ (assigning v a color), and either

- $v \notin P_s$, $P_s = P$, and $\sigma_s = \sigma$, or
- $v \in P_s$, $P = P_s \setminus \{v\}$, $\sigma_s|_{B_t} = \sigma$ and $\sigma_s(v) = [k]$.

Join. If t is a join node, let s_1 and s_2 be its children. Here we only have to mark, for each partial b-vertex contained in B_t, the colors it has seen in G_{s_1} and in G_{s_2}. Therefore we proceed as follows. We set $\mathsf{tab}_t[\gamma, P, \sigma, C]$ to 1 if there exist $C_1, C_2 \subseteq [k]$ with $C_1 \cup C_2 = C$; and for $i \in [2]$, $\sigma_i\colon P \to 2^{[k]}$ such that for all $p \in P$, $\sigma(p) = \sigma_1(p) \cup \sigma_2(p)$, and such that $\mathsf{tab}_{s_i}[\gamma, P_i, \sigma_i, C_i] = 1$ for all $i \in [2]$.

Correctness of the algorithm follows from its description. Regarding its run time, we observe that for each node $t \in V(T)$, all table entries $\mathsf{tab}_t[\cdot]$ can be computed in time $2^{\mathcal{O}(wk)}$. Since the number of nodes in T is at most $\mathcal{O}(wn)$, the algorithm runs in time $2^{\mathcal{O}(wk)}n = 2^{\mathcal{O}(\mathsf{tw}\cdot k)}n$. □

Corollary 2.10. b-Coloring *can be solved in* $2^{\mathcal{O}(\ell^2)}n$ *time where* n *is the number of vertices and* ℓ *the vertex cover number of the input graph.*

Proof. Let G be the input graph with vertex cover number ℓ. It is well-known that a vertex cover of size ℓ of G, which can be found in $\mathcal{O}(1.2738^{\ell}+\ell n)$ time [12], can be used to give a path decomposition of G of width (at most) ℓ in $\mathcal{O}(n)$ time. Together with the fact that each b-coloring of a graph with vertex cover number ℓ can have at most $\ell + 1$ colors, the result follows from Proposition 2.9. □

2.5 Chordal graphs

Another consequence of Proposition 2.9 is that b-Coloring is fixed-parameter tractable on chordal graphs parameterized by the number of colors; which answers an open question of Sampaio [53].

Corollary 2.11. b-Coloring *can be solved in* $2^{\mathcal{O}(k^2)}n$ *time on chordal graphs with* n *vertices.*

Proof. Let (G, k) be an instance of b-Coloring such that G is a chordal graph. If the maximum clique size in G is more than k, then G has no proper coloring, and therefore no b-coloring, with k colors. We may assume that the maximum clique size in G is at most k. This in turn implies that

the treewidth of G is at most k, since a clique tree of G (which can be found in linear time [4]) is in fact a tree decomposition of width at most k of G. We can therefore apply Proposition 2.9.

Note that even though the algorithm of [4] implies a linear dependence on the number of edges in the input graph, this can be avoided by the following observation. If an n-vertex graph has tree-width at most w, then it has at most wn edges. Therefore, if the number of edges in G is more than kn then we can report that (G, k) is a No-instance; otherwise, the dependence on the number of edges is subsumed by the run time of the algorithm from Proposition 2.9. □

3 Parameterized by Clique-Width

In this section, we consider the b-coloring problem parameterized by the clique-width of the input graph. We will work with decompositions of bounded *module-width*, which is equivalent for our purposes, see Theorem 2.3.

The main contribution of this section is an algorithm that given a graph G on n vertices and one of its rooted branch decompositions of module-width w, and an integer k, decides whether G has a b-coloring with k colors in time $n^{2^{\mathcal{O}(w)}}$. Before we proceed, we observe that b-Coloring is W[1]-hard in this parameterization, and that the exponential dependence on w of the degree of the polynomial in the runtime is probably difficult to avoid.

Proposition 3.1. *The b-Coloring problem on graphs on n vertices parameterized by their module-width w is* W[1]*-hard and cannot be solved in time $n^{2^{o(w)}}$, unless* ETH *fails. Moreover, the hardness holds even when a linear branch decomposition of width w is provided.*

Proof. Fomin et al. [24] showed that the Graph Coloring problem which given a graph G of module-width w and an integer k asks for a proper coloring of G with k colors cannot be solved in time $n^{2^{o(w)}}$ unless ETH fails, even when a linear branch decomposition of module-width w is provided. Using Graph Coloring in this setting as a starting point of a reduction, we can add a k-clique to the input graph. The resulting graph has a b-coloring with k colors if and only if the original graph has a proper coloring with k colors (take the vertices in the k-clique as the b-vertices). It is not difficult to see that the given branch decomposition can be extended to include the vertices of the added k-clique without increasing its module-width by too much. W[1]-hardness parameterized by w can be observed using the same argument, even as a consequence of an earlier result [23]. □

3.1 Outline of the Algorithm

Throughout the following, we are given a graph G and one of its rooted branch decompositions $(T, \mathcal{L})$ of module-width $w = \mathsf{mw}(T, \mathcal{L})$ and we want to find a b-coloring of G with k colors, if it exists. In particular, our algorithm will find a b-coloring $\mathcal{C}$ together with a set of *witness b-vertices*, containing precisely one b-vertex for each color class of $\mathcal{C}$, if it exists. This will be done via dynamic programming along T, and for each node $t \in V(T)$, the partial solutions associated with t are partial b-colorings of G_t (recall Definition 2.4).

To obtain an efficient algorithm, we require a compact representation of the partial b-colorings of each subgraph G_t associated with a node $t \in V(T)$. To that end, we introduce the notion of a *t-signature* of a partial b-coloring. Two partial b-colorings with the same t-signature will be interchangeable for the sake of our algorithm, therefore the number of table entries at each node t will be bounded by the number of t-signatures.

Let $(\mathcal{C}, B)$ be a partial b-coloring of G_t. For $(\mathcal{C}, B)$ to be extended to a b-coloring $(\mathcal{C}', B')$ of the entire graph G, we have to ensure that two things happen for each color class $C \in \mathcal{C}$:

(i) The extension of C in $\mathcal{C}'$ is an independent set in G.

(ii) There is a witness b-vertex in B' for the extension of C in $\mathcal{C}'$.

The t-signature has to represent a partial b-coloring faithfully enough so that we can keep track of all the ways in which the above two conditions can be satisfied for each of its color classes 'in the future'. At the same time, its definition has to enable us to significantly compress the information about partial b-colorings of G_t. This happens in the following way. We categorize color classes of partial b-colorings of G_t according to *t-types*. If two color classes C_1, C_2 of a partial b-coloring $(\mathcal{C}, B)$ have the same t-type, then the above two conditions can be satisfied for C_1 and C_2 by extensions of $(\mathcal{C}, B)$ in the exact same ways. This allows us to forget about the 'names' of the color classes in a partial b-coloring, but instead to only remember for each t-type how many color classes with that type there are. This is precisely the information that is stored in a t-signature.

Now, if we can bound the number of t-types by some function of the module-width w, say $f(w)$, then the number of t-signatures is upper bounded by $k^{f(w)} \leq n^{f(w)}$. (There are at most k colors, so in particular there are at most k colors with a given t-type.) This translates directly to an upper bound on the number of table entries in the dynamic programming algorithm, which, up to some constants in the degree of the polynomial, bounds the runtime of the resulting algorithm.

Let us discuss the information that goes into the definition of a t-type. Let C be a color class in a partial b-coloring $(\mathcal{C}, B)$ of G_t. To keep track of which vertices from $\overline{V_t}$ can be added to C without introducing a coloring conflict, it suffices to store which equivalence classes of $\sim_t$ have vertices in C,[4] since all vertices in a given equivalence class have the same neighbors in $\overline{V_t}$. This way we can ensure that condition (i) is satisfied.

To verify if condition (ii) is satisfied we have to store some information about the partial b-vertices. Naturally, we record whether or not B contains a partial b-vertex of C, but we need to store more information. Suppose that B contains the partial b-vertex v of C. In a straightforward approach, we would simply keep track of the color classes that already appear in the neighborhood of v. This way we could easily decide at which point during the execution of the algorithm, a partial b-vertex turns into a b-vertex. However, this results in prohibitively large table entries, since there are 2^{k-1} subsets of colors that we would have to consider, which for our purpose is no better than 2^n.

We overcome this issue with the following symmetry breaking trick: We do *not* record which color classes the partial b-vertex of C already sees/still needs to see. Instead, we record which equivalence classes $Q \in V_t/\sim_t$ contain a partial b-vertex w of *some other color class* such that $N(w) \cap C = \emptyset$. Suppose that some equivalence class $Q \in V_t/\sim_t$ contains the partial b-vertex $w \in B$ of another color class $C' \neq C$, such that w has no neighbor of color C in V_t. For w to become a b-vertex of its color, the color class C must be extended with a neighbor of w in the future, i.e. in $\overline{V_t}$. The neighborhood of w in $\overline{V_t}$ is precisely $N_G(Q) \cap \overline{V_t}$, therefore we can concisely model this situation as color class C requiring to contain a vertex among the future neighbors of Q. In this situation, we say that

color class C has demand to the future neighbors of Q.

The t-type records for each equivalence class Q of $\sim_t$, if a color class contains vertices of Q, or if it has demand to the future of Q, or none of the two. Note that if a color class both contains a vertex from Q and has demand to the future of Q, we already know that we can disregard the corresponding partial b-coloring: In the corresponding color class, we cannot add any future

[4] This is similar to the algorithm of Wanke for Graph Coloring on graphs of bounded NLC-width [59].

neighbors of Q without creating a coloring conflict, and if we do not add a future neighbor of Q, then there is some color class whose partial b-vertex will never become a b-vertex.

Now, if we have a partial b-coloring in which every color class has a partial b-vertex, and all demands have been fulfilled, meaning that there is no color class that has demand to the future of some equivalence class of $\sim_t$, then we know that we actually have a b-coloring. Moreover, the number of t-types is $2^{\mathcal{O}(w)}$, so the resulting algorithm runs in time $n^{2^{\mathcal{O}(w)}}$ (see above).

3.2 t-Types and t-Signatures

In this section we introduce the basic concepts that we alluded to in the above description, namely the notion of a *t-type* and of a *t-signature*, where t is some node in the given branch decomposition. A t-type is meant to capture the necessary information of a color class in a partial b-coloring of G_t. However, we cannot give the definition of a t-type as a property of a vertex set alone: a color class C may have demand to the future of an equivalence class, which is because there is a partial b-vertex of *another* color $C' \neq C$ that has no neighbor of color C yet. Therefore, we first give the definition of a t-type abstractly, i.e. absent of any partial b-coloring or color class, and then define what it means for a color class to be of a certain t-type *within a partial b-coloring*.

The t-type is a pair of a bit that is meant to tell us whether or not a coloring contains a partial b-vertex of that color, and a map that tells us for each equivalence class, whether there is a vertex of the color in the equivalence class (via the value cont), or if the color has demand to the future neighbors of the equivalence class (via the value dem), or none of the two (via the value none).

Definition 3.2 (t-Type). Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$ and let $t \in V(T)$. A *t-type* is a pair (ϕ, ξ) of a map $\phi\colon Q_t/\sim_t \to \{\mathsf{none}, \mathsf{cont}, \mathsf{dem}\}$ and a bit $\xi \in \{0, 1\}$. We denote the set of all t-types by types_t.

Before we proceed, we observe an upper bound on the number of t-types. For the component ξ, we clearly only have two choices, and for each equivalence class Q of $\sim_t$, the entry $\phi(Q)$ takes one of three values.

Observation 3.3. Let $(T, \mathcal{L})$ be a rooted branch decomposition of module-width $w = \mathsf{mw}(T, \mathcal{L})$. For each $t \in V(T)$, $|\mathsf{types}_t| = 2 \cdot 3^{|V_t/\sim_t|} \leq 2 \cdot 3^w$.

We now define what it means for a color class to be of a certain t-type within a partial b-coloring of G_t. This is basically a formalization of the above discussion, but it holds one aspect that is of importance of the algorithm and the arguments to follow. We discuss this after the following definition, which is illustrated in Figure 3.

Definition 3.4. Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$ and let $t \in V(T)$. Let $(\mathcal{C}, B)$ be a partial b-coloring of G_t, let $C \in \mathcal{C}$ be a color class, and let $\tau = (\phi, \xi) \in \mathsf{types}_t$ be a t-type. We say that C *has t-type τ in* $(\mathcal{C}, B)$ if

(i) $\xi = |C \cap B|$ and

(ii) for each $Q \in V_t/\sim_t$,

  (a) if $Q \cap C \neq \emptyset$, and there is no $v \in (B \setminus C) \cap Q$ such that $N(v) \cap C = \emptyset$, then $\phi(Q) = \mathsf{cont}$,

  (b) if $Q \cap C = \emptyset$ and there exists some $v \in (B \setminus C) \cap Q$ such that $N(v) \cap C = \emptyset$, then $\phi(Q) = \mathsf{dem}$, and

  (c) if $Q \cap C = \emptyset$, and there is no $v \in (B \setminus C) \cap Q$ such that $N(v) \cap C = \emptyset$, then $\phi(Q) = \mathsf{none}$.

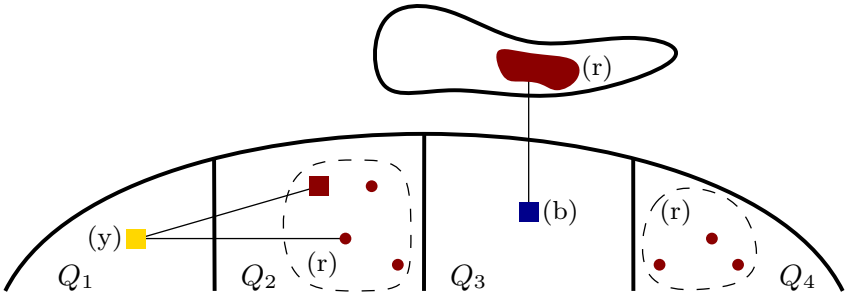

Figure 3: Illustration of the definition of a color class being of a certain *t-type* inside a partial b-coloring of G_t. The large square vertices are partial b-vertices for their color. The type of the red (r) color in the coloring is as follows. Since it has a b-vertex (the one in Q_2), we have that $\xi = 1$. Since Q_2 and Q_4 have red vertices, $\phi(Q_2) = \phi(Q_4) = \mathsf{cont}$. Q_1 and Q_3 do not have red vertices. Q_1 contains the b-vertex of color yellow (y), but this vertex already has a red neighbor. Therefore, $\phi(Q_1) = \mathsf{none}$. Finally, Q_3 has the b-vertex of color blue (b), and this vertex does not have a red neighbor yet. Therefore, there has to be a red vertex among the future neighbors of Q_3. Hence, $\phi(Q_3) = \mathsf{dem}$.

The reader may have observed that (ii) does not cover all the possibilities. The situation that is not covered is when $Q \cap C \neq \emptyset$ and there is some $v \in (B \setminus C) \cap Q$ such that $N(v) \cap C = \emptyset$. A priori, we can of course not exclude this as a possibility, but there is a simple reason that partial b-colorings that contain a color class in which this situation arises can be disregarded: For the vertex v to become a b-vertex for its color, we have to add a future neighbor of Q to C; but since Q already contains a vertex from C this means that the resulting set is not independent anymore.

We turn to the definition of a t-signature which again is first given in abstract terms.

Definition 3.5 (t-Signature). Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$, and let $t \in V(T)$. A *t-signature* is a map $\mathsf{sig}_t \colon \mathsf{types}_t \to \{0, 1, \ldots, k\}$ such that $\sum_{\tau \in \mathsf{types}_t} \mathsf{sig}_t(\tau) = k$.

The following bound on the number of t-signatures immediately follows from Observation 3.3: for each t-type, the function takes one of $k + 1 \leq n + 1$ values.

Observation 3.6. Let G be a graph on n vertices and $(T, \mathcal{L})$ be one of its branch decompositions of module-width $w = \mathsf{mw}(T, \mathcal{L})$. For each $t \in V(T)$, there are at most $n^{2^{\mathcal{O}(w)}}$ many t-signatures.

A t-signature *represents* a partial b-coloring $(\mathcal{C}, B)$ of G_t if for each t-type it counts correctly how many color classes in $\mathcal{C}$ are of that t-type in $(\mathcal{C}, B)$.

Definition 3.7. Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$, and let $t \in V(T)$. Let furthermore sig_t be a t-signature and $(\mathcal{C}, B)$ a partial b-coloring in G_t. We say that sig_t *represents* $(\mathcal{C}, B)$ if for each t-type $\tau \in \mathsf{types}_t$, there are precisely $\mathsf{sig}_t(\tau)$ color classes in $(\mathcal{C}, B)$ that have t-type τ in $(\mathcal{C}, B)$.

We call a partial b-coloring of G_t *representable* if there is a t-signature that represents it.

Since throughout this section, we only consider b-colorings and partial b-colorings with k (possibly empty) colors, Definitions 3.5 and 3.7 together imply that if a partial b-coloring is represented by a t-signature, then necessarily each of its color classes has a t-type: Definition 3.5 requires that for a t-signature sig_t, the sum of $\mathsf{sig}_t(\tau)$ over all t-types τ is k, and any partial b-coloring in G_t has k colors.

We would like to remark once more that not all partial b-colorings of G_t can be represented by a t-signature, since there is a case that a color class cannot be described by a t-type. In this

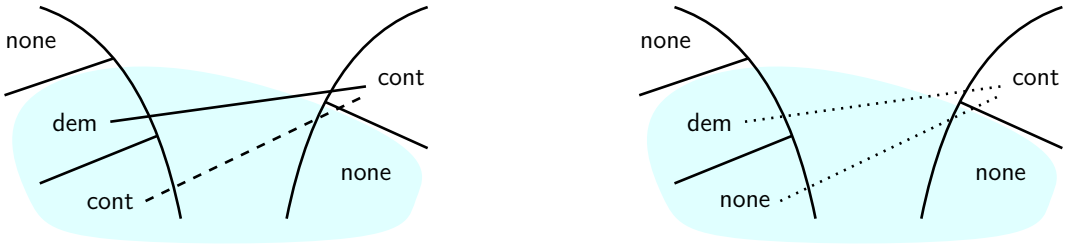

Figure 4: Illustration of Definition 3.9. The shaded area shows a bubble and the labels on the equivalence classes correspond to type labelings. For the left hand side, note that between a pair of classes that are both labeled 'cont', there can be no edge in the operator. Moreover, since the bubble contains a class labeled cont and one labeled dem, the demand of the latter has to be fulfilled at this node, i.e. there has to be an edge from this class to a 'cont'-class. The right side shows the situation when the 'cont'-class in the bubble is changed to 'none', in which case the dotted edges may or may not be present in the operator.

case the partial b-coloring is not representable. Conversely, we can make the following observation about representable partial b-colorings which is useful in several proofs and sometimes used without explicit reference.

Observation 3.8. Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$, and let $t \in V(T)$. Let $(\mathcal{C}, B)$ be a representable partial b-coloring of G_t, and let $C \in \mathcal{C}$ be a color class whose t-type in $(\mathcal{C}, B)$ is (ϕ, ξ). If for some equivalence class $Q \in V_t/\!\sim_t$, $Q \cap C \neq \emptyset$, then $\phi(Q) = \mathsf{cont}$.

3.3 Compatibility

Let $t \in V(T) \setminus \mathrm{L}(T)$ be an internal node of the given rooted branch decomposition, let r and s be its children, and let (H_t, η_r, η_s) be the operator of t. In our algorithm, we want to combine information about partial b-colorings of G_r and G_s to obtain information about partial b-colorings of G_t. We will try to obtain a color class of a partial b-coloring of G_t by taking the union of a color class C_r of a partial b-coloring of G_r and a color class C_s of a partial b-coloring of G_s.

However, in some cases this is not possible. For instance, when C_r contains vertices from some equivalence class $Q_r \in V_r/\!\sim_r$ and C_s contains vertices from some equivalence class $Q_s \in V_s/\!\sim_s$, and in the graph H_t of the operator of t, we have that $Q_rQ_s \in E(H_t)$. Then, in G_t all edges between the set Q_r and Q_s are present which means that $C_r \cup C_s$ is not an independent set in G_t.

Another condition is necessary to ensure that several demands that *have to be* met at node t are indeed met. Let $C_t = C_r \cup C_s$ and suppose there is an equivalence class $Q_t \in V_t/\!\sim_t$ that contains a vertex of C_t. Suppose furthermore that there is another equivalence class $Q_r \in V_r/\!\sim_r$ contained in the bubble of Q_t such that C_r has demand to the future neighbors of Q_r. Then, this demand must be fulfilled by a neighbor of Q_r in C_s for otherwise, the equivalence class Q_t both contains vertices of C_t and C_t has demand to the future neighbors of Q_t. The resulting partial b-coloring would not be representable.

The following definition formalizes this discussion and projects it down to the 'type level'; we illustrate this notion in Figure 4.

Definition 3.9 (Compatible types). Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$. Let furthermore $t \in V(T) \setminus \mathrm{L}(T)$ with children r and s, and let (H_t, η_r, η_s) be the operator of t. Let $(\phi_r, \xi_r) \in \mathsf{types}_r$ and $(\phi_s, \xi_s) \in \mathsf{types}_s$. We say that (ϕ_r, ξ_r) and (ϕ_s, ξ_s) are *compatible* if the following conditions hold.

(i) $\xi_r + \xi_s \leq 1$.

(ii) There is no pair $Q_r \in V_r/\sim_r$, $Q_s \in V_s/\sim_s$ such that $Q_rQ_s \in E(H_t)$ and $\phi_r(Q_r) = \phi_s(Q_s) = \mathsf{cont}$.

(iii) For each $Q \in V_t/\sim_t$ such that there exists a $p \in \{r, s\}$ and a $Q_p \in \eta_p^{-1}(Q)$ with $\phi_p(Q_p) = \mathsf{cont}$, the following holds.

  (a) For all $Q_r \in \eta_r^{-1}(Q)$ with $\phi_r(Q_r) = \mathsf{dem}$, there is a $Q_s \in V_s/\sim_s$ with $\phi_s(Q_s) = \mathsf{cont}$ and $Q_rQ_s \in E(H_t)$.

  (b) Similarly, for all $Q_s \in \eta_s^{-1}(Q)$ with $\phi_s(Q_s) = \mathsf{dem}$, there is a $Q_r \in V_r/\sim_r$ with $\phi_r(Q_r) = \mathsf{cont}$ and $Q_sQ_r \in E(H_t)$.

Given a pair of a color class C_r of a partial b-coloring of G_r and a color class C_s of a partial b-coloring of G_s whose types in the respective colorings are compatible, $C_r \cup C_s$, considered as a color class in a partial b-coloring of G_t, has a fixed type. We prove this later in the lemmas that attest the correctness of the algorithm, but we already describe the construction of this type here, mainly since the notion of compatibility of signatures that we give below, requires this 'merge type'.

Definition 3.10 (Merge Type). Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$. Let furthermore $t \in V(T) \setminus \mathrm{L}(T)$ with children r and s, and let (H_t, η_r, η_s) be the operator of t. Let $\rho = (\phi_r, \xi_r) \in \mathsf{types}_r$ and $\sigma = (\phi_s, \xi_s) \in \mathsf{types}_s$ be a pair of compatible types. The *merge type* of ρ and σ, denoted by $\mu(\rho, \sigma)$, is the following t-type (ϕ_t, ξ_t).

(i) $\xi_t = \xi_r + \xi_s$.

(ii) For each $Q \in V_t/\sim_t$:

  (a) If for some $p \in \{r, s\}$, there exists a $Q_p \in \eta_p^{-1}(Q)$ with $\phi_p(Q_p) = \mathsf{cont}$, then $\phi_t(Q) = \mathsf{cont}$.

  (b) If (ii.a) does not apply and for some $p \in \{r, s\}$ there exists a $Q_p \in \eta_p^{-1}(Q)$ with $\phi_p(Q_p) = \mathsf{dem}$ and for $o \in \{r, s\} \setminus \{p\}$ and all $Q_pQ_o \in E(H_t)$ we have $\phi_o(Q_o) \neq \mathsf{cont}$, then $\phi_t(Q) = \mathsf{dem}$.

  (c) If neither (ii.a) nor (ii.b) applies, then $\phi_t(Q) = \mathsf{none}$.

Towards a notion of compatibility of signatures, we first define a structure we call *merge skeleton*. Given a node $t \in V(T)$ with children r and s, the merge skeleton is an edge-labeled bipartite graph whose vertices are the r-types and the s-types, with the merge type of a compatible pair of types $\rho \in \mathsf{types}_r$, $\sigma \in \mathsf{types}_s$ written on the edge $\rho\sigma$. Such an edge is meant to represent the fact that taking the union of a color class C_r that has r-type ρ in a partial b-coloring of G_r with a color class C_s that has s-type σ in a partial b-coloring of G_s results in a color class of t-type $\mu(\rho, \sigma)$ in the partial b-coloring of G_t that results from merging the partial b-colorings of G_r and G_s.

Definition 3.11 (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 bipartite graph $(\mathfrak{J}, \mathfrak{m})$ where

- $V(\mathfrak{J}) = \mathsf{types}_r \cup \mathsf{types}_s$,
- for all $\rho \in \mathsf{types}_r$, $\sigma \in \mathsf{types}_s$, $\rho\sigma \in E(\mathfrak{J})$ if and only if ρ and σ are compatible, and
- $\mathfrak{m}\colon E(\mathfrak{J}) \to \mathsf{types}_t$ is such that for all $\rho\sigma \in E(\mathfrak{J})$, $\mathfrak{m}(\rho\sigma)$ is the merge type of ρ and σ.

Using the merge skeleton, we want to find out how to construct a t-signature of a partial b-coloring of G_t that is obtained from a pair of a partial b-coloring for G_r and one for G_s, knowing only their signatures. Any pair of an r-signature sig_r and an s-signature sig_s can 'flesh out' the merge skeleton $(\mathfrak{J}, \mathfrak{m})$ of r and s, in the following sense. We can obtain a map labeling the vertices of $\mathfrak{J}$ that follows sig_r on types_r and sig_s on types_s. Then, an edge-labeling $\mathfrak{n}$ of $\mathfrak{J}$ with integers from $\{0, 1, \ldots, k\}$, such that for each vertex of $\mathfrak{J}$, the sum over its incident edges e of $\mathfrak{n}(e)$ is equal to its vertex label, produces a t-signature sig_t. We can read off how many color classes of each type there are from the edge labeling $\mathfrak{n}$. In fact, each t-signature can be produced in such a way, as we prove below.

Definition 3.12 (Compatible signatures). Let $(T, \mathcal{L})$ be a rooted branch decomposition. Let furthermore $t \in V(T) \setminus \mathrm{L}(T)$ with children r and s. Let sig_t be a t-signature, let sig_r be an r-signature and sig_s be a s-signature. We say that $(\mathsf{sig}_t, \mathsf{sig}_r, \mathsf{sig}_s)$ is *compatible* if there is a triple $(\mathfrak{J}, \mathfrak{m}, \mathfrak{n})$ such that $(\mathfrak{J}, \mathfrak{m})$ is the merge skeleton of r and s, and $\mathfrak{n}\colon E(\mathfrak{J}) \to \{0, 1, \ldots, k\}$ is a map with the following properties.

(i) For all $p \in \{r, s\}$ and all $\pi \in \mathsf{types}_p$, $\sum_{e \in E(\mathfrak{J})\colon \pi \in e} \mathfrak{n}(e) = \mathsf{sig}_p(\pi)$.

(ii) For all $\tau \in \mathsf{types}_t$, $\sum_{e \in E(\mathfrak{J})\colon \mathfrak{m}(e) = \tau} \mathfrak{n}(e) = \mathsf{sig}_t(\tau)$.

We first show that we can test efficiently whether a triple of signatures is compatible.

Lemma 3.13. *Let G be a graph on n vertices and let $(T, \mathcal{L})$ be one of its rooted branch decomposition of module-width $w = \mathsf{mw}(T, \mathcal{L})$. Let $t \in V(T) \setminus \mathrm{L}(T)$ with children r and s. Let sig_t be a t-signature, sig_r be an r-signature, and sig_s be an s-signature. One can decide in time $n^{2^{\mathcal{O}(w)}}$ whether or not $(\mathsf{sig}_t, \mathsf{sig}_r, \mathsf{sig}_s)$ is compatible.*

Proof. We first observe that the merge skeleton can be constructed in $2^{\mathcal{O}(w)}$ time, where $w = \mathsf{mw}(T, \mathcal{L})$: It is easy to see that given two types $\rho \in \mathsf{types}_r$, $\sigma \in \mathsf{types}_s$, we can decide whether or not ρ and σ are compatible in time $w^{\mathcal{O}(1)}$. Moreover, by Observation 3.3, $|\mathsf{types}_r| \leq 2^{\mathcal{O}(w)}$ and $|\mathsf{types}_s| \leq 2^{\mathcal{O}(w)}$, therefore we have to check for $(2^{\mathcal{O}(w)})^2 = 2^{\mathcal{O}(w)}$ pairs of types if they are compatible, and if so, compute their merge type. (This also implies that $|E(\mathfrak{J})| = 2^{\mathcal{O}(w)}$.) Computing a merge type can be done in time $w^{\mathcal{O}(1)}$ as well, simply by following the construction given in Definition 3.10.

We brute-force all candidates for the labeling $\mathfrak{n}$. Given such a candidate, we can verify in time $2^{\mathcal{O}(w)}$ if it satisfies parts (i) and (ii) of the definition of compatible signatures. Since $|E(\mathfrak{J})| = 2^{\mathcal{O}(w)}$, a trivial upper bound on the number of such candidate labelings is $n^{2^{\mathcal{O}(w)}}$ and therefore the claimed bound follows. □

3.4 Merging and Splitting Partial b-Colorings

In this section we show that the notions introduced above work as desired, and the technical lemmas we prove here will be the cornerstone of the correctness proof of the resulting algorithm that we give later.

3.4.1 Bottom to Top

Lemma 3.14. *Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$ and let $t \in V(T) \setminus \mathrm{L}(T)$ be an internal node with children r and s. Let sig_r be an r-signature, sig_s be an s-signature, and sig_t be a t-signature such that:*

- *For all $p \in \{r, s\}$, there is a partial b-coloring $(\mathcal{C}_p, B_p)$ in G_p that is represented by sig_p, and*

- $(\mathsf{sig}_t, \mathsf{sig}_r, \mathsf{sig}_s)$ *is compatible.*

Then, there is a partial b-coloring $(\mathcal{C}_t, B_t)$ *of* G_t *that is represented by* sig_t.

Proof. Let $(\mathfrak{J}, \mathfrak{m}, \mathfrak{n})$ be the structure witnessing that $(\mathsf{sig}_t, \mathsf{sig}_r, \mathsf{sig}_s)$ is compatible. We use Algorithm 1 to create the pair $(\mathcal{C}_t, B_t)$. We first show that $(\mathcal{C}_t, B_t)$ is indeed a partial b-coloring of G_t, and then later that sig_t represents $(\mathcal{C}_t, B_t)$.

Input : $(\mathcal{C}_r, B_r)$, $(\mathcal{C}_s, B_s)$, $\mathfrak{J}$, and $\mathfrak{n}$ as above
Output: $(\mathcal{C}_t, B_t)$, where $\mathcal{C}_t$ is a partition of V_t and $B_t \subseteq V_t$.
$\mathcal{C}'_r \leftarrow \mathcal{C}_r$, $\mathcal{C}'_s \leftarrow \mathcal{C}_s$, $\mathcal{C}_t \leftarrow \emptyset$;
foreach $\rho \in \mathsf{types}_r$, $\sigma \in \mathsf{types}_s$ *with* $\rho\sigma \in E(\mathfrak{J})$ **do**
  Let $x \leftarrow \mathfrak{n}(\rho\sigma)$;
  for $i = 1, \ldots, x$ **do**
    Let $C_r \in \mathcal{C}'_r$ be of r-type ρ and $C_s \in \mathcal{C}'_s$ be of s-type σ;
    $\mathcal{C}_t \leftarrow \mathcal{C}_t \cup \{C_r \cup C_s\}$;
    $\mathcal{C}'_r \leftarrow \mathcal{C}'_r \setminus \{C_r\}$, $\mathcal{C}'_s \leftarrow \mathcal{C}'_s \setminus \{C_s\}$;
return $(\mathcal{C}_t, B_r \cup B_s)$;

Algorithm 1: Merging $(\mathcal{C}_r, B_r)$ and $(\mathcal{C}_s, B_s)$ according to $\mathfrak{J}$ and $\mathfrak{n}$.

Claim 3.14.1. $(\mathcal{C}_t, B_t)$ *as constructed above is a partial b-coloring of* G_t *with* k *colors.*

Proof. Since $\mathcal{C}_r$ is a partition of V_r and $\mathcal{C}_s$ is a partition of V_s, and each part of $\mathcal{C}_r$ and $\mathcal{C}_s$ is used precisely once to obtain a part of $\mathcal{C}_t$ in Algorithm 1, it is clear by Definition 3.12((i)) that $\mathcal{C}_t$ is a partition of V_t. Together with Definition 3.12((ii)) and the definition of a t-signature, this ensures that $\mathcal{C}_t$ has k parts.

We argue that each part $C \in \mathcal{C}_t$ is an independent set. Suppose for a contradiction that C is not an independent set and let $uv \in E(G_t)$ be an edge with $u, v \in C$. By construction, there are $C_r \in \mathcal{C}_r$ and $C_s \in \mathcal{C}_s$ such that $C = C_r \cup C_s$. Moreover, since C_r and C_s are color classes in a coloring, they are independent sets, so we may assume that $u \in C_r$ and $v \in C_s$ (up to renaming). For all $p \in \{r, s\}$, let $\tau_p = (\phi_p, \xi_p)$ be the p-type of C_p in $(\mathcal{C}_p, B_p)$. Let furthemore $Q_r \in V_r/{\sim_r}$ be the equivalence class of $\sim_r$ containing u and $Q_s \in V_s/{\sim_s}$ be the equivalence class of $\sim_s$ containing v. This means that $\phi_r(Q_r) = \phi_s(Q_s) = \mathsf{cont}$. For u and v to be adjacent, the edge Q_rQ_s has to be present in H_t. On the other hand, $\tau_r\tau_s$ is an edge of the merge skeleton which implies that τ_r and τ_s are compatible types; in which case Definition 3.9((ii)) forbids the presence of this edge in H_t, a contradiction.

We have shown that $\mathcal{C}_t$ is a proper coloring of G_t, it remains to show that for all $C \in \mathcal{C}_t$, $|C \cap B_t| \leq 1$. Suppose for a contradiction that for some $C \in \mathcal{C}_t$, $|C \cap B_t| > 1$, and let $C_r \in \mathcal{C}_r$, $C_s \in \mathcal{C}_s$ be such that $C = C_r \cup C_s$, as per Algorithm 1. Since for all $p \in \{r, s\}$, $(\mathcal{C}_p, B_p)$ is a partial b-coloring of G_p, we have that $|C_p \cap B_p| \leq 1$, and clearly $C_r \cap B_s = C_s \cap B_r = \emptyset$. This means that $|C_r \cap B_r| = |C_s \cap B_s| = 1$; and in the r-type (ϕ_r, ξ_r) of C_r in $(\mathcal{C}_r, B_r)$ and the s-type (ϕ_s, ξ_s) of C_s in $(\mathcal{C}_s, B_s)$, $\xi_r = \xi_s = 1$. But again, (ϕ_r, ξ_r) and (ϕ_s, ξ_s) are compatible, so by Definition 3.9((i)), $\xi_r + \xi_s \leq 1$, a contradiction. ⌟

To prove the lemma, it remains to show that the t-signature sig_t represents $(\mathcal{C}_t, B_t)$. This is shown via the following claim, with Definition 3.12 ensuring that the numbers work out.

Claim 3.14.2. Let $C_r \in \mathcal{C}_r$ and $C_s \in \mathcal{C}_s$, and let $\tau_r = (\phi_r, \xi_r)$ be the r-type of C_r in $(\mathcal{C}_r, B_r)$, and let $\tau_s = (\phi_s, \xi_s)$ be the s-type of C_s in $(\mathcal{C}_s, B_s)$, such that $C_t = C_r \cup C_s$ is a color class in $(\mathcal{C}_t, B_t)$. Then, the t-type of C_t in $(\mathcal{C}_t, B_t)$ is $\mu(\tau_r, \tau_s)$.

Proof. First observe that if $C_t = C_r \cup C_s$ is a color class in $(\mathcal{C}_t, B_t)$, then τ_r and τ_s are compatible by construction. Let $\tau_t = (\phi_t, \xi_t) = \mu(\tau_r, \tau_s)$. We have to argue that the t-type of C_t in $(\mathcal{C}_t, B_t)$ is indeed (ϕ_t, ξ_t).

For the first item of the definition of the merge type, we observe that $\xi_r + \xi_s = |C_r \cap B_r| + |C_s \cap B_s|$ and since $B_t = B_r \cup B_s$ and $C_t = C_r \cup C_s$, we have $\xi_t = \xi_r + \xi_s = |C_t \cap B_t|$.

Now let $Q \in V_t/\sim_t$. Suppose that $\phi_t(Q) = \mathsf{cont}$; we have to argue that $C_t \cap Q \neq \emptyset$ and that there is no vertex $v \in (B_t \setminus C_t) \cap Q$ with $N(v) \cap C_t = \emptyset$. By the definition of the merge type, there is some $p \in \{r, s\}$ such that there is a $Q_p \in V_p/\sim_p$ with $\eta_p(Q_p) = Q$ and $\phi_p(Q_p) = \mathsf{cont}$. Since C_p has p-type (ϕ_p, ξ_p) in $(\mathcal{C}_p, B_p)$, $C_p \cap Q_p \neq \emptyset$ which implies that $C_t \cap Q \neq \emptyset$. Now suppose that there is some vertex $v \in (B_t \setminus C_t) \cap Q$ with $N(v) \cap C_t = \emptyset$. This means that there is some $p \in \{r, s\}$ and some $Q_p \in \eta_p^{-1}(Q)$ such that $v \in Q_p$, and $N(v) \cap C_p = \emptyset$. Since C_p has a p-type in $(\mathcal{C}_p, B_p)$, this means that $C_p \cap Q_p = \emptyset$ and therefore $\phi_p(Q_p) = \mathsf{dem}$. Assume wlog that $p = r$. Since (ϕ_r, ξ_r) and (ϕ_s, ξ_s) are compatible, we have by Definition 3.9((iii)) that there is some $Q_s \in V_s/\sim_s$ with $\phi_s(Q_s) = \mathsf{cont}$ and $Q_r Q_s \in E(H_t)$. But this implies that v has a neighbor in $C_s \subseteq C_t$, a contradiction.

Now suppose that $\phi_t(Q) = \mathsf{dem}$. By the definition of the merge type, we have that in this case:

(i) For any $p \in \{r, s\}$ and $Q_p \in V_p/\sim_p$ with $\eta_p(Q_p) = Q$, $\phi_p(Q_p) \neq \mathsf{cont}$.

(ii) We may assume (up to renaming) that for some $Q_r \in \eta_r^{-1}(Q)$, $\phi_r(Q_r) = \mathsf{dem}$,

(iii) and that for all $Q_r Q_s \in E(H_t)$, $\phi_s(Q_s) \neq \mathsf{cont}$.

From (i) we derive that $C_t \cap Q = \emptyset$. Next, (ii) implies that there is a vertex $v \in (B_r \setminus C_r) \cap Q_r$ with $N(v) \cap C_r = \emptyset$, and by (iii), we can conclude that v has no neighbor in C_s either. Therefore, v has no neighbor in C_t, as required.

Finally, suppose that $\phi_t(Q) = \mathsf{none}$. Again then there is no $Q_p \in \eta^{-1}(Q)$ such that $\phi_p(Q_p) = \mathsf{cont}$. If for all $p \in \{r, s\}$ and all $Q_p \in \eta_p^{-1}(Q)$, $\phi_p(Q_p) = \mathsf{none}$, then it is clear that $C_t \cap Q = \emptyset$, and that there is no $v \in (B_t \setminus C_t) \cap Q$ with $N(v) \cap C = \emptyset$. So suppose (up to renaming) that for some $Q_r \in \eta_r^{-1}(Q)$, $\phi_r(Q_r) = \mathsf{dem}$, implying that there is a vertex $v \in (B_r \setminus C_r) \cap Q_r$ with $N(v) \cap C_r = \emptyset$. Since we did not land in case (ii.b) of the definition of a merge type, there is some $Q_r Q_s \in E(H_t)$ such that $\phi_s(Q_s) = \mathsf{cont}$, which means v has a neighbor in $C_s \subseteq C_t$. Since this holds for any such Q_r (and Q_s), we can conclude that there is no vertex in $(B_t \setminus C_t) \cap Q$ with $N(v) \cap C_t = \emptyset$. This concludes the proof. ⌟

This concludes the proof of Lemma 3.14. □

3.4.2 Top to Bottom

Lemma 3.15. *Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$ and let $t \in V(T) \setminus \mathrm{L}(T)$ be an internal node with children r and s. Let sig_t be a t-signature, and suppose there is a partial b-coloring $(\mathcal{C}_t, B_t)$ of G_t which is represented by sig_t. Then, there exists an r-signature sig_r and an s-signature sig_s such that*

- *for all $p \in \{r, s\}$ there is a partial b-coloring $(\mathcal{C}_p, B_p)$ represented by sig_p, and*
- *$(\mathsf{sig}_t, \mathsf{sig}_r, \mathsf{sig}_s)$ is compatible.*

Proof. For all $p \in \{r, s\}$, we let $\mathcal{C}_p := \mathcal{C}_t|_{V_p}$ and $B_p := B_t \cap V_p$. It is clear that $(\mathcal{C}_p, B_p)$ is a partial b-coloring of G_p.

Claim 3.15.1. For all $p \in \{r, s\}$, $(\mathcal{C}_p, B_p)$ is represented by some p-signature.

Proof. Suppose the claim is false for $p = r$. Then there is some $C_r \in \mathcal{C}_r$ that has no r-type in $(\mathcal{C}_r, B_r)$, meaning that for some $Q_r \in V_r/\!\sim_r$, $Q_r \cap C_r \neq \emptyset$ and there is a vertex $v \in (B_r \setminus C_r) \cap Q_r$ with $N(v) \cap C_r = \emptyset$. By construction, there is a $C_t \in \mathcal{C}_t$ with $C_t = C_r \cup C_s$ for some $C_s \subseteq V_s$. Since $(\mathcal{C}_t, B_t)$ is representable, and $\eta_r(Q_r) \cap C_t \neq \emptyset$, we know that $N(v) \cap C_t \neq \emptyset$ (otherwise, C_t has no t-type in $(\mathcal{C}_t, B_t)$). Therefore, $N(v) \cap C_s \neq \emptyset$. But since all vertices in Q_r are twins with respect to V_s, and since $C_r \cap Q_r \neq \emptyset$ and $v \in Q_r$, this means that there is an edge between some vertex in C_r and some vertex in C_s, contradicting the fact that C_t is an independent set. ⌟

By the previous claim, we know that $(\mathcal{C}_r, B_r)$ is represented by some r-signature sig_r and that $(\mathcal{C}_s, B_s)$ is represented by some s-signature sig_s. It remains to show that $(\mathsf{sig}_t, \mathsf{sig}_r, \mathsf{sig}_s)$ is compatible. To be able to argue this, we show that each t-type with non-zero value in sig_t appears as an edge label of the merge skeleton $(\mathfrak{J}, \mathfrak{m})$ of r and s, in particular that it is the merge type of the r-type and s-type labeling the endpoints of this edge.

Claim 3.15.2. Let $C_t \in \mathcal{C}_t$ be a color class whose t-type in $(\mathcal{C}_t, B_t)$ is $\tau_t = (\phi_t, \xi_t)$. Let $\tau_r = (\phi_r, \xi_r)$ be the r-type of $C_r := C_t \cap V_r$ in $(\mathcal{C}_r, B_r)$, and let $\tau_s = (\phi_s, \xi_s)$ be the s-type of $C_s := C_t \cap V_s$ in $(\mathcal{C}_s, B_s)$. Then, τ_r and τ_s are compatible and $\tau_t = \mu(\tau_r, \tau_s)$.

Proof. We first show that τ_r and τ_s are compatible. We know that $\xi_t \in \{0, 1\}$ and that $\xi_t = 1$ if and only if $|C_t \cap B_t| = 1$ if and only if either $|C_r \cap B_r| = 1$ or $|C_s \cap B_s| = 1$ if and only if either $\xi_r = 1$ or $\xi_s = 1$, therefore $\xi_r + \xi_s \leq 1$, meaning that condition (i) of the definition of compatibility is satisfied. Since C_t is an independent set, there are no edges between C_r and C_s. This means that for any pair $Q_r \in V_r/\!\sim_r$, $Q_s \in V_s/\!\sim_s$ with $\phi_r(Q_r) = \phi_s(Q_s) = \mathsf{cont}$, $Q_rQ_s \notin E(H_t)$, otherwise there would be an edge between C_r and C_s, so condition (ii) is satisfied as well.

Now suppose that Definition 3.9((iii)) is violated. We may assume (up to renaming) that there is some $Q \in V_t/\!\sim_t$ with the following properties. There is a $Q_r^* \in \eta_r^{-1}(Q)$ with $\phi_r(Q_r^*) = \mathsf{cont}$, meaning that $C_r \cap Q_r^* \neq \emptyset$ and so $C_t \cap Q \neq \emptyset$. Moreover, there is some $Q_r \in \eta_r^{-1}(Q)$ with $\phi_r(Q_r) = \mathsf{dem}$, where for any $Q_rQ_s \in E(H_t)$, $\phi_s(Q_s) \neq \mathsf{cont}$. This means that there is a vertex $v \in (B_r \setminus C_r) \cap Q_r$ such that $N(v) \cap C_r = \emptyset$, and moreover that $N(v) \cap C_s = \emptyset$, implying that $N(v) \cap C_t = \emptyset$. Note that $v \in (B_t \setminus C_t) \cap Q_t$. In other words, we have argued that Q is an equivalence class of $\sim_t$ such that $C_t \cap Q \neq \emptyset$ and there is a vertex $v \in (B_t \setminus C_t) \cap Q$ such that $N(v) \cap C_t = \emptyset$. But this means that the color class C_t cannot have a t-type in $(\mathcal{C}_t, B_t)$, so $(\mathcal{C}_t, B_t)$ was not representable, a contradiction.

Now we argue that τ_t, the t-type of C_t in $(\mathcal{C}_t, B_t)$, is indeed the merge type of τ_r and τ_s. We already argued above that $\xi_t = \xi_r + \xi_s$. Now let $Q \in V_t/\!\sim_t$, and suppose that $\phi_t(Q) = \mathsf{cont}$. This means that $Q \cap C_t \neq \emptyset$. We may assume (up to renaming) that $u \in Q_r \cap C_r$ for some $Q_r \in \eta_r^{-1}(Q)$. Since $(\mathcal{C}_r, B_r)$ is representable by Claim 3.15.1 this already implies that $\phi_r(Q_r) = \mathsf{cont}$, therefore $\phi_t(Q)$ is set in accordance with the definition of the merge type.

Now suppose that for some $Q \in V_t/\!\sim_t$, $\phi_t(Q) = \mathsf{dem}$. Then, $Q \cap C_t = \emptyset$ and there is some $v \in (B_t \setminus C_t) \cap Q$ such that $N(v) \cap C_t = \emptyset$. First, since $Q \cap C_t = \emptyset$, this immediately implies that for all $p \in \{r, s\}$ and all $Q_p \in \eta_p^{-1}(Q)$, $Q_p \cap C_p = \emptyset$ and therefore $\phi_p(Q_p) \neq \mathsf{cont}$. Now for $p \in \{r, s\}$, let Q_p be the equivalence class of $\sim_p$ containing v. We may assume (up to renaming) that $p = r$. Clearly, $\eta_r(Q_r) = Q$, therefore $Q_r \cap C_r = \emptyset$. Moreover, $N(v) \cap C_r = \emptyset$, and we have that $\phi_r(Q_r) = \mathsf{dem}$. Now suppose for a contradiction that for some $Q_rQ_s \in E(H_t)$, $\phi_s(Q_s) = \mathsf{cont}$.

This implies that $N(v) \cap C_s \neq \emptyset$, and therefore $N(v) \cap C_t \neq \emptyset$, a contradiction. We have shown that also in this case, $\phi_t(Q)$ is set in accordance with the definition of the merge type.

Finally, suppose that $\phi_t(Q) = \mathsf{none}$. Then, $Q \cap C_t = \emptyset$ and there is no $v \in (B_t \setminus C_t) \cap Q$ with $N(v) \cap C_t = \emptyset$. This immediately implies that for all $p \in \{r, s\}$ and all $Q_p \in \eta^{-1}(Q)$, $\phi_p(Q_p) \neq \mathsf{cont}$. Suppose that for some $p \in \{r, s\}$ and some $Q_p \in \eta_p^{-1}(Q)$, $\phi_p(Q_p) = \mathsf{dem}$, and assume (up to renaming) that $p = r$. This means that there is some vertex $u \in (B_r \setminus C_r) \cap Q_r$ with $N(u) \cap C_r = \emptyset$. On the other hand, we know that $N(u) \cap C_t \neq \emptyset$, so u has a neighbor in C_s. This means that there is a $Q_rQ_s \in E(H_t)$ such that $Q_s \cap C_s \neq \emptyset$, meaning that $\phi_s(Q_s) = \mathsf{cont}$. Therefore, $\phi_t(Q)$ is also set in accordance with the definition of the merge type. ⌟

To finish the proof, we have to construct an edge labeling $\mathfrak{n} \colon E(\mathfrak{J}) \to \{0, 1, \ldots, k\}$ satisfying the conditions of Definition 3.12. The previous claim tells us that we can construct $\mathfrak{n}$ in a straightforward way. Initially, set $\mathfrak{n}(\tau_t) = 0$ for all $\tau_t \in \mathsf{types}_t$. For each color class $C_t \in \mathcal{C}_t$ whose t-type in $(\mathcal{C}_t, B_t)$ is τ_t, we know that the r-type of $C_r := C_t \cap V_r$, say τ_r, and the s-type of $C_s := C_t \cap V_s$, say τ_s, are such that $\tau_t = \mathfrak{m}(\tau_r\tau_s)$, i.e. τ_t appears as the label of the edge between τ_r and τ_s in $\mathfrak{J}$. We therefore increase the value of $\mathfrak{n}(\tau_r\tau_s)$ by one. Once we did this for all color classes of $(\mathcal{C}_t, B_t)$, the tuple $(\mathfrak{J}, \mathfrak{m}, \mathfrak{n})$ satisfies the requirements of Definition 3.12, so $(\mathsf{sig}_t, \mathsf{sig}_r, \mathsf{sig}_s)$ is compatible. □

3.5 The Algorithm

As alluded to above, the algorithm is bottom-up dynamic programming along the given rooted branch decomposition $(T, \mathcal{L})$ of G. First, we define the table entries stored at each node.

Definition of the table entries. For a node $t \in V(T)$ and a t-signature sig_t, we let $\mathsf{tab}[t, \mathsf{sig}_t] = 1$ if and only if there exists a partial b-coloring of G_t that is represented by sig_t.

We now show that if all table entries have been computed correctly, then the solution can be read off the table entries stored at the root $\mathfrak{r}$ of the given rooted branch decomposition. Observe that since $V_\mathfrak{r} = V(G)$ and therefore $\overline{V_\mathfrak{r}} = \emptyset$, the equivalence relation $\sim_\mathfrak{r}$ has one equivalence class, namely $V(G)$.

Lemma 3.16. *Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$ and let $\mathfrak{r} \in V(T)$ be the root of T. Let ρ be the $\mathfrak{r}$-type $(\phi_\mathfrak{r}, \xi_\mathfrak{r})$ with $\xi_\mathfrak{r} = 1$ and $\phi_\mathfrak{r}(V(G)) = \mathsf{cont}$. Let $\mathsf{sig}_\mathfrak{r}$ be the $\mathfrak{r}$-signature letting $\mathsf{sig}_\mathfrak{r}(\rho) = k$. Then, G has a b-coloring with k colors if and only if $\mathsf{tab}[\mathfrak{r}, \mathsf{sig}_\mathfrak{r}] = 1$.*

Proof. Suppose that G has a b-coloring $(\mathcal{C}, B)$ with k colors. Then, $(\mathcal{C}, B)$ is also a partial b-coloring; but since all vertices in B are already b-vertices for their color, all demands have been fulfilled. This means that $(\mathcal{C}, B)$ is representable by an $\mathfrak{r}$-signature, denote this $\mathfrak{r}$-signature by sig. We argue that $\mathsf{sig} = \mathsf{sig}_\mathfrak{r}$, in particular that all color classes $C \in \mathcal{C}$ are of type $\rho = (\phi_\mathfrak{r}, \xi_\mathfrak{r})$ in $(\mathcal{C}, B)$ as in the statement of the lemma. Let $C \in \mathcal{C}$ be any color class. Since $(\mathcal{C}, B)$ is a b-coloring, B contains a b-vertex v of C, therefore also $C \neq \emptyset$ which implies that the $\mathfrak{r}$-type of C is indeed ρ. As this reasoning applies to all k color classes of $(\mathcal{C}, B)$, we can conclude that $\mathsf{tab}[\mathfrak{r}, \mathsf{sig}_\mathfrak{r}] = 1$.

Now suppose for the other direction that $\mathsf{tab}[\mathfrak{r}, \mathsf{sig}_\mathfrak{r}] = 1$. Then there is a partial b-coloring $(\mathcal{C}, B)$ of $G_\mathfrak{r} = G$ with k colors represented by $\mathsf{sig}_\mathfrak{r}$. Since $(\phi_\mathfrak{r}, \xi_\mathfrak{r})$ is the type of each color class and $\xi_\mathfrak{r} = 1$, each color class has a partial b-vertex; since no color class has demand to the future neighbors of $V(G)$ by $\phi_\mathfrak{r}$, each partial b-vertex is indeed a b-vertex for its color. Therefore, $\mathcal{C}$ is a b-coloring of G with k colors. □

We describe how to compute the table entries, starting with the leaves of T.

Leaves of T. Let $t \in V(T)$ be a leaf node of T and let $v \in V(G)$ be the vertex such that $\mathcal{L}(v) = t$. We show how to set the table entries $\mathsf{tab}[t, \cdot]$. The partial b-colorings of $G_t = (\{v\}, \emptyset)$ we have to consider are the following. The vertex v is colored with one of the k colors, and it is either the partial b-vertex for its color or not.

The t-signatures representing these colorings look as follows. Observe that $\sim_t$ has precisely one equivalence class, namely $\{v\}$. We let ϕ_{cont} be the map with $\phi_{\mathsf{cont}}(\{v\}) = \mathsf{cont}$. In the case that v is *not* the partial b-vertex of its color, we have

- one color of type $(\phi_{\mathsf{cont}}, 0)$, and
- $k-1$ colors of type $(\phi_\emptyset, 0)$ with $\phi_\emptyset(\{v\}) = \mathsf{none}$.

We denote this signature by sig_1, i.e. we let $\mathsf{sig}_1((\phi_{\mathsf{cont}}, 0)) = 1$ and $\mathsf{sig}_1((\phi_\emptyset, 0)) = k-1$.

In the case that v *is* the partial b-vertex of its color class, then the remaining $k-1$ color classes have demand to the future neighbors of $\{v\}$, so that v eventually becomes the b-vertex of its color. Therefore we have

- one color of type $(\phi_{\mathsf{cont}}, 1)$, and
- $k-1$ colors of type $(\phi_{\mathsf{dem}}, 0)$ with $\phi_{\mathsf{dem}}(\{v\}) = \mathsf{dem}$.

We denote this signature by sig_2, i.e. we let $\mathsf{sig}_2((\phi_{\mathsf{cont}}, 1)) = 1$ and $\mathsf{sig}_2((\phi_{\mathsf{dem}}, 0)) = k-1$. To summarize, for each t-signature sig, we let

$$\mathsf{tab}[t, \mathsf{sig}] := \begin{cases} 1, & \text{if } \mathsf{sig} \in \{\mathsf{sig}_1, \mathsf{sig}_2\} \\ 0, & \text{otherwise} \end{cases}$$

Next, the internal nodes of T.

Internal nodes of T. Now let $t \in V(T) \setminus \mathrm{L}(T)$ with children r and s. For each t-signature sig_t, we let $\mathsf{tab}[t, \mathsf{sig}_t] = 1$ if and only if there exists a pair $(\mathsf{sig}_r, \mathsf{sig}_s)$ of an r-signature sig_r and an s-signature sig_s such that

(i) $\mathsf{tab}[r, \mathsf{sig}_r] = 1$ and $\mathsf{tab}[s, \mathsf{sig}_s] = 1$, and

(ii) $(\mathsf{sig}_t, \mathsf{sig}_r, \mathsf{sig}_s)$ is compatible.

Equipped with the lemmas of the previous sections, we can prove correctness of the above algorithm.

Lemma 3.17. *For each $t \in V(T)$ and t-signature sig_t, the above algorithm computes the table entry $\mathsf{tab}[t, \mathsf{sig}_t]$ correctly.*

Proof. We prove the lemma by induction on the height of t. For the base case, when t is a leaf, it is easily verified. From now on we may assume that $t \in V(T) \setminus \mathrm{L}(T)$ with children r and s.

First, suppose that the algorithm set $\mathsf{tab}[t, \mathsf{sig}] = 1$. This means that there is a pair $(\mathsf{sig}_r, \mathsf{sig}_s)$ of an r-signature sig_r and an s-signature sig_s such that $\mathsf{tab}[r, \mathsf{sig}_r] = 1$ and $\mathsf{tab}[s, \mathsf{sig}_s] = 1$ and $(\mathsf{sig}_t, \mathsf{sig}_r, \mathsf{sig}_s)$ is compatible. By induction, we know that there is a partial b-coloring of G_r represented by the r-signature sig_r and a partial b-coloring of G_s represented by the s-signature sig_s. Then, by Lemma 3.14, there is a partial b-coloring of G_t represented by the t-signature sig_t.

Conversely, suppose that there is a partial b-coloring of G_t represented by the t-signature sig_t. Then, by Lemma 3.15, there is a partial b-coloring of G_r represented by an r-signature sig_r and a partial b-coloring of G_s represented by an s-signature sig_s, such that $(\mathsf{sig}_t, \mathsf{sig}_r, \mathsf{sig}_s)$ is compatible. By induction, the algorithm set $\mathsf{tab}[r, \mathsf{sig}_r] = 1$ and $\mathsf{tab}[s, \mathsf{sig}_s] = 1$, and therefore, by the above description, it set $\mathsf{tab}[t, \mathsf{sig}_t] = 1$. □

We wrap up. By Lemma 3.17, the algorithm computes all table entries correctly, and by Lemma 3.16, the solution to the instance can be determined upon inspecting the table entries associated with the root of the given branch decomposition. Correctness of the algorithm follows.

Regarding the runtime, we observe the following. Given an n-vertex graph with rooted branch decomposition $(T, \mathcal{L})$ of module-width $w = \mathsf{mw}(T, \mathcal{L})$, we have that $|V(T)| = \mathcal{O}(n)$. (T is a full binary tree on n leaves, so $|V(T)| = 2n - 1$.) Let $t \in V(T)$. If t is a leaf node, then computing the table entries $\mathsf{tab}[t, \cdot]$ takes constant time. If t is an internal node, then by Observation 3.6, we have to compute $n^{2^{\mathcal{O}(w)}}$ table entries. Assume by induction that the table entries associated with the children of t have been computed. For each t-signature sig_t we have to try for $\left(n^{2^{\mathcal{O}(w)}}\right)^2 = n^{2^{\mathcal{O}(w)}}$ pairs of one signature per child whether or not they form a compatible triple together with sig_t. For each triple, this can be done in time $n^{2^{\mathcal{O}(w)}}$ by Lemma 3.13. Therefore, the overall runtime of the algorithm is $n^{2^{\mathcal{O}(w)}}$.

Theorem 3.18. *There is an algorithm that solves b-Coloring in time $n^{2^{\mathcal{O}(w)}}$, where n denotes the number of vertices of the input graph, and w denotes the module-width of a given rooted branch decomposition of the input graph.*

3.6 Fall Coloring

Recall that a *fall coloring* is a special type of b-coloring where *every* vertex is a b-vertex for its color. In other words, it is a partition of the vertex set of a graph into independent dominating sets. We adapt our algorithm for b-Coloring on graphs of bounded clique-width to solve Fall Coloring, and therefore show that the latter problem is as well solvable in time $n^{2^{\mathcal{O}(w)}}$, where w denotes the clique-width of a given decomposition of the input graph.

Adaptation of the b-Coloring Algorithm

We now show how to adapt the algorithm of Theorem 3.18 to solve the Fall Coloring problem in time $n^{2^{\mathcal{O}(w)}}$ as well. This adaptation in some sense simplifies the algorithm for b-Coloring, since we do no have to keep track of whether or not a color class has a b-vertex in a partial coloring; *every* vertex has to be a b-vertex. Now, if we can construct a coloring such that each color class is nonempty, and each vertex is a b-vertex for its color, then clearly we have a fall coloring. With small modification, the mechanics of our algorithm for b-Coloring allow for checking if there is a coloring with this property. The main difference will be in the definition of the type of a color class.

Let $(C_1, \ldots, C_k)$ be a proper coloring of G_t for some node t, and C_i and C_j be two distinct color classes. If for some $Q \in V_t/\!\sim_t$, $C_i \cap Q = \emptyset$, and there is *any* vertex $v_j \in C_j$ such that $N(v_j) \cap C_i = \emptyset$, then C_i has demand to the future neighbors of Q: the vertex v_j needs to become a b-vertex of color j, and since it has no neighbor in color class i so far, one of its future neighbors (equivalently, a future neighbor of equivalence class Q), has to receive color i.

The definition of a *t-fall type* can be obtained from the definition of a t-type by dropping the bit ξ which becomes unnecessary in the context of Fall Coloring.

The definition of a color class being of a certain t-fall type becomes the following.

Definition 3.19 (t-Fall-type). Let G be a graph with rooted branch decomposition $(T, \mathcal{L})$, and let $t \in V(T)$. A *t-fall type* is a map $\phi\colon V_t/\!\sim_t \to \{\mathsf{none}, \mathsf{cont}, \mathsf{dem}\}$.

Let $\mathcal{C} = (C_1, \ldots, C_k)$ be a proper coloring of G_t, and let ϕ be a t-fall type. For $i \in \{1, \ldots, k\}$, we say that C_i *has t-fall type ϕ in $\mathcal{C}$* if for each $Q \in V_t/\!\sim_t$,

(i) if $Q \cap C_i \neq \emptyset$ and for all $v \in Q \setminus C_i$, $N(v) \cap C_i \neq \emptyset$, then $\phi(Q) = \mathsf{cont}$,

(ii) if $Q \cap C_i = \emptyset$ and there is a $v \in Q \setminus C_i$ with $N_{G_t}(v) \cap C_i = \emptyset$, then $\phi(Q) = \mathsf{dem}$, and

(iii) $\phi(Q) = \mathsf{none}$, otherwise.

We again restrict ourselves to finding (partial) colorings that are *representable*, in the sense that there is no color class that both intersects an equivalence class and has demand to its future neighbors. In complete analogy, we define a t-signature as a function counting the number of color classes of each t-fall type.

We say that two fall-types are compatible, if they satisfy parts (ii) and (iii) of Definition 3.9, the definition of compatible types in the case of b-Coloring. Part (i) simply disappears since we do not have to keep track of whether or not a color class contains a partial b-vertex. With this in mind, the technical arguments given in Section 3.4 go through.

The definition of the table entries is analogous as well, and by an argument parallel to the proof of Lemma 3.16, we can conclude that this information is sufficient to solve the problem.

We discuss the resulting algorithm. For the leaf nodes, we only have to consider colorings with one color class whose fall-type is $\phi_v(\{v\}) = \mathsf{cont}$ and $k-1$ color classes whose fall-type is $\phi_{\mathsf{dem}}(\{v\}) = \mathsf{dem}$. This is because in any fall-coloring of G, the vertex v has to be a b-vertex for its color. The computation of the internal nodes remains the same. A correctness proof of the algorithm can now be given in the same way as in the proof of Lemma 3.17, and the discussion of the runtime of the algorithm still goes through. We have the following theorem.

Theorem 3.20. *There is an algorithm that solves* Fall Coloring *in time* $n^{2^{\mathcal{O}(w)}}$*, where* n *denotes the number of vertices of the input graph, and* w *denotes the module-width of a given rooted branch decomposition of the input graph.*

Hardness

We now show that the runtime of the algorithm from Theorem 3.20 is optimal in some sense. Specifically, we give a reduction that proves the same lower bounds as the ones we obtained for b-Coloring. Recall again that linear module-width and linear clique-width can be used interchangeably in this setting (Theorem 2.3).

Proposition 3.21. *The* Fall Coloring *problem on graphs on* n *vertices parameterized by the module-width* w *of the input graph is* W[1]*-hard and cannot be solved in time* $n^{2^{o(w)}}$*, unless* ETH *fails. Moreover, the hardness holds even when a linear branch decomposition of width* w *is provided.*

Proof. We give a reduction from Graph Coloring parameterized by the module-width w of the input graph which is W[1]-hard and has no $n^{2^{o(w)}}$-time algorithm under ETH [23, 24]. Given an instance (G, k) construct a instance (H, k) of Fall Coloring as follows. We obtain H from G by adding, for each vertex $v \in V(G)$, a clique X_v on $k-1$ vertices to the graph, and making X_v complete to v.

If H has a fall coloring with k colors, then clearly this is a proper coloring of G with k colors, since G is an induced subgraph of H. Suppose G has a proper coloring with k colors. For each vertex $v \in V(G)$, we can bijectively assign the $k-1$ remaining colors (i.e. all colors except the one appearing on v) to the vertices of X_v. The coloring constructed this way is a fall coloring of H with k colors: First, we immediately observe that the coloring is proper. Since we started from a proper coloring of G, there is no monochromatic edge in G. Since we colored the vertices of each X_v bijectively with all colors except the one appearing on v, and since $N_H(X_v) \cap V(G) = \{v\}$, we

did not introduce any monochromatic edge either. It remains to argue that each vertex of H is a b-vertex for its color. For each $v \in V(G)$, we have that v is a b-vertex since the remaining $k-1$ colors appear on X_v. For each $u \in X_v$, we have that u is a b-vertex since it sees $k-2$ colors on $X_v \setminus \{u\}$, plus the color of v; since $X_v \cup \{v\}$ is a clique, all of these colors are mutually distinct.

The size of H is polynomial in the size of G, and it is clear that adding the cliques X_v did not increase the module-width of G. □

4 Conclusion

In this work, we gave an XP-algorithm for b-Coloring parameterized by the clique-width of a given decomposition of the input graph, and an FPT-algorithm parameterized by the vertex cover number. This initiated the study of structural parameterizations of the b-Coloring and b-Chromatic Number problems. The most prominent parameter sitting between clique-width and the vertex cover number is arguably the treewidth of a graph. Since any graph of bounded treewidth has bounded clique-width, our algorithm implies that b-Coloring parameterized by treewidth is in XP. We therefore ask, is b-Coloring parameterized by the treewidth of the input graph FPT or W[1]-hard?

It would be interesting to obtain an FPT-algorithm for b-Coloring parameterized by the vertex cover number vc whose runtime is tight under ETH. Lokshtanov et al. [46] showed that Graph Coloring has no $2^{o(\mathsf{vc}\log\mathsf{vc})} \cdot n^{\mathcal{O}(1)}$ time algorithm unless ETH fails, and by the same argument[5] given in Proposition 3.1, this rules out $2^{o(\mathsf{vc}\log\mathsf{vc})} \cdot n^{\mathcal{O}(1)}$ time algorithms for b-Coloring under ETH. We therefore ask if the runtime of $2^{\mathcal{O}(\mathsf{vc}^2)} \cdot n^{\mathcal{O}(1)}$ in Corollary 2.10 can be improved to $2^{\mathcal{O}(\mathsf{vc}\log\mathsf{vc})} \cdot n^{\mathcal{O}(1)}$.

There are two main approaches for solving Graph Coloring parameterized by clique-width, one being efficient when the number of colors is small [42], and the other being efficient when the number of colors is large [21, 59]. Our algorithm for b-Coloring falls in the latter category. It would be interesting to obtain an efficient algorithm for b-Coloring parameterized by clique-width when the number is small, with a running time that is tight under the Strong Exponential Time Hypothesis as it was done for Graph Coloring by Lampis [42]. Moreover, Courcelle et al. [14] recently gave an algorithm that unifies both approaches into a single algorithm; is the same possible for b-Coloring?

Acknowledgements. We would like to thank the anonymous reviewers for many valuable suggestions that improved this work. We are particularly grateful for the suggestion to replace our initial vertex cover based algorithm and the use of Courcelle's Theorem by a single explicit DP algorithm on graphs of bounded tree-width. This led to a cleaner and more precise presentation of the result on chordal graphs as well as an improved algorithm parameterized by vertex cover.

References

[1] Pierre Aboulker, Édouard Bonnet, Eun Jung Kim, and Florian Sikora. Grundy coloring & friends, half-graphs, bicliques. In *STACS 2020*, volume 154 of *LIPIcs*, pages 58:1–58:18, 2020.

[2] Hans-Jürgen Bandelt and Henry Martin Mulder. Distance-hereditary graphs. *Journal of Combinatorial Theory Series B*, 41:182–208, 1986.

[5] Noting that we may assume that the number of colors is always linearly bounded in the vertex cover number; so adding the clique does not increase the number of colors in a prohibitive way.

[3] Rémy Belmonte, Eun Jung Kim, Michael Lampis, Valia Mitsou, and Yota Otachi. Grundy distinguishes treewidth from pathwidth. In *ESA 2020*, volume 173 of *LIPIcs*, pages 14:1–14:19, 2020.

[4] Jean R.S. Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. Springer, 1993.

[5] Flavia Bonomo, Guillermo Durán, Frederic Maffray, Javier Marenco, and Mario Valencia-Pabon. On the b-coloring of cographs and P_4-sparse graphs. *Graphs and Combinatorics*, 25(2):153–167, 2009.

[6] Flavia Bonomo, Oliver Schaudt, Maya Stein, and Mario Valencia-Pabon. b-Coloring is NP-hard on co-bipartite graphs and polytime solvable on tree-cographs. *Algorithmica*, 73(2):289–305, 2015.

[7] Binh-Minh Bui-Xuan, Jan Arne Telle, and Martin Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theoretical Computer Science*, 511:66–76, 2013.

[8] Victor A. Campos, Carlos V. Lima, Nicolas A. Martins, Leonardo Sampaio, Marcio C. Santos, and Ana Silva. The b-chromatic index of graphs. *Discrete Mathematics*, 338(11):2072–2079, 2015.

[9] Victor A. Campos, Carlos Vinicius G. C. Lima, and Ana Silva. Graphs of girth at least 7 have high b-chromatic number. *European Journal of Combinatorics*, 48:154–164, 2015.

[10] Victor A. Campos, Cláudia Linhares-Sales, Rudini Sampaio, and Ana Karolinna Maia. Maximization coloring problems on graphs with few P_4. *Discrete Applied Mathematics*, 164:539–546, 2014.

[11] Victor A. Campos and Ana Silva. Edge-b-coloring trees. *Algorithmica*, 80(1):104–115, 2018.

[12] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

[13] David Coudert, Guillaume Ducoffe, and Alexandru Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. *ACM Transactions on Algorithms*, 15(3):1–57, 2019.

[14] Bruno Courcelle, Irène Durand, and Michael Raskin. A unified algorithm for colouring graphs of bounded clique-width. *CoRR*, abs/2008.07468, 2020.

[15] Bruno Courcelle, Joost Engelfriet, and Grzegorz Rozenberg. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2):218–270, 1993.

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

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

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

[19] Jean E. Dunbar, Sandra M. Hedetniemi, S.T. Hedetniemi, David P. Jacobs, J. Knisely, R.C. Laskar, and Douglas F. Rall. Fall colorings of graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 33:257–274, 2000.

[20] Brice Effantin, Nicolas Gastineau, and Olivier Togni. A characterization of b-chromatic and partial grundy numbers by induced subgraphs. *Discrete Mathematics*, 339(8):2157–2167, 2016.

[21] Wolfgang Espelage, Frank Gurski, and Egon Wanke. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In *WG 2001*, pages 117–128, 2001.

[22] Taoufik Faik. About the b-continuity of graphs (extended abstract). *Electronic Notes in Discrete Mathematics*, 17:151–156, 2004.

[23] 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.

[24] 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.

[25] Michael U. Gerber and Daniel Kobler. Algorithms for vertex-partitioning problems on graphs with fixed clique-width. *Theoretical Computer Science*, 299(1-3):719–734, 2003.

[26] Wayne Goddard and Michael A. Henning. Independent domination in graphs: A survey and recent results. *Discrete Mathematics*, 313(7):839–854, 2013.

[27] Martin Charles Golumbic and Udi Rotics. On the clique-width of some perfect graph classes. *International Journal of Foundations of Computer Science*, 11(03):423–443, 2000.

[28] Frank Gurski. The behavior of clique-width under graph operations and graph transformations. *Theory of Computing Systems*, 60(2):346–376, 2017.

[29] Frédéric Havet, Claudia Linhares Sales, and Leonardo Sampaio. b-coloring of tight graphs. *Discrete Applied Mathematics*, 160(18):2709–2715, 2012.

[30] Frédéric Havet and Leonardo Sampaio. On the Grundy and b-chromatic numbers of a graph. *Algorithmica*, 65:885–899, 2013.

[31] Pinar Heggernes and Jan Arne Telle. Partitioning graphs into generalized dominating sets. *Nordic Journal on Computing*, 5(2):128–142, 1998.

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

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

[34] Robert W. Irving and David F. Manlove. The b-chromatic number of a graph. *Discrete Applied Mathematics*, 91(1-3):127–141, 1999.

[35] Lars Jaffke and Paloma T. Lima. A complexity dichotomy for critical values of the b-chromatic number of graphs. *Theoretical Computer Science*, 815:182–196, 2020.

[36] Lars Jaffke, Paloma T. Lima, and Daniel Lokshtanov. b-Coloring parameterized by clique-width. In *STACS 2021*, volume 187 of *LIPIcs*, pages 43:1–43:15. Schloss Dagstuhl, 2021.

[37] Lars Jaffke, Paloma T. Lima, and Geevarghese Philip. Structural parameterizations of clique coloring. In *MFCS 2020*, volume 170 of *LIPIcs*, pages 49:1–49:15. Schloss Dagstuhl, 2020.

[38] Klaus Jansen. Complexity results for the optimum cost chromatic partition problem, 1997.

[39] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994.

[40] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *FOCS 2021*, pages 184–192. IEEE, 2021.

[41] Jan Kratochvíl, Zsolt Tuza, and Margit Voigt. On the b-chromatic number of graphs. In *WG 2002*, pages 310–320, 2002.

[42] Michael Lampis. Finer tight bounds for coloring on clique-width. *SIAM Journal on Discrete Mathematics*, 34(3):1538–1558, 2020.

[43] Renu Laskar and Jeremy Lyle. Fall colouring of bipartite graphs and cartesian products of graphs. *Discrete Applied Mathematics*, 157(2):330–338, 2009.

[44] Juho Lauri and Christodoulos Mitillos. Complexity of fall coloring for restricted graph classes. In *IWOCA 2019*, pages 352–364. Springer, 2019.

[45] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

[46] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM Journal on Computing*, 47(3):675–702, 2018.

[47] Jeremy Lyle, Nate Drake, and Renu Laskar. Independent domatic partitioning or fall coloring of strongly chordal graphs. *Congressus Numerantium*, 172:149–159, 2005.

[48] Johann A. Makowsky and Udi Rotics. On the clique-width of graphs with few P_4's. *International Journal of Foundations of Computer Science*, 10(03):329–348, 1999.

[49] Christodoulos Mitillos. *Topics in Graph Fall-Coloring*. PhD thesis, Illinois Institute of Technology, 2016.

[50] Fahad Panolan, Geevarghese Philip, and Saket Saurabh. On the parameterized complexity of b-chromatic number. *Journal of Computer and System Sciences*, 84:120–131, 2017.

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

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

[53] Leonardo Sampaio. *Algorithmic aspects of graph colourings heuristics*. PhD thesis, Université Nice Sophia Antipolis, 2012.

[54] Ana Silva. Graphs with small fall-spectrum. *Discrete Applied Mathematics*, 254:183–188, 2019.

[55] Ana Shirley Ferreira da Silva. *The b-chromatic number of some tree-like graphs*. PhD thesis, Université Joseph-Fourier - Grenoble I, 2010.

[56] Jan Arne Telle and Andrzej Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM Journal on Discrete Mathematics*, 10(4):529–550, 1997.

[57] Jean-Marie Vanherpe. Clique-width of partner-limited graphs. *Discrete Mathematics*, 276(1-3):363–374, 2004.

[58] Clara Inés Betancur Velasquez, Flavia Bonomo, and Ivo Koch. On the b-coloring of P_4-tidy graphs. *Discrete Applied Mathematics*, 159(1):60 – 68, 2011.

[59] Egon Wanke. k-NLC graphs and polynomial algorithms. *Discrete Applied Mathematics*, 54:251–266, 1994.