Discrete Applied Mathematics 347 (2024) 1–22

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

On d-stable locally checkable problems parameterized by mim-width

Carolina Lucía Gonzalez [a], Felix Mann [b,*]

[a] CONICET-Universidad de Buenos Aires, Instituto de Investigación en Ciencias de la Computación (ICC), Buenos Aires, Argentina
[b] University of Fribourg, Department of Informatics, Fribourg, Switzerland

ARTICLE INFO

Article history:
Received 31 March 2023
Received in revised form 12 October 2023
Accepted 14 December 2023
Available online xxxx

Keywords:
Locally checkable problem
Mim-width
d-stability
Vertex partitioning problem
DN logic
Coloring
Conflict-free coloring
[k]-Roman domination
b-coloring

ABSTRACT

In this paper we continue the study of locally checkable problems under the framework introduced by Bonomo-Braberman and Gonzalez in 2020, by focusing on graphs of bounded mim-width. We study which restrictions on a locally checkable problem are necessary in order to be able to solve it efficiently on graphs of bounded mim-width. To this end, we introduce the concept of d-stability of a check function. The related locally checkable problems contain large classes of problems, among which we can mention, for example, LCVP problems. We give an algorithm showing that these problems are XP when parameterized by the mim-width of a given binary decomposition tree of the input graph , that is, that they can be solved in polynomial time given a binary decomposition tree of bounded mim-width. We explore the relation between d-stable locally checkable problems and the recently introduced DN logic (Bergougnoux, Dreier and Jaffke, 2022), and show that both frameworks model the same family of problems. We include a list of concrete examples of d-stable locally checkable problems whose complexity on graphs of bounded mim-width was open so far.

1. Introduction

There are many graph problems which can be expressed as vertex coloring problems or, equivalently, as vertex partition problems. Besides the well-known Chromatic Number and its variations, subset problems can be understood as a coloring problem with two colors, and path problems such as Hamiltonian cycle can be stated by asking for a vertex coloring with n colors $1, \ldots, n$ (where n is the number of vertices of some input graph) such that every vertex with color i has to be adjacent to a vertex with color $i-1$ and a vertex of color $i+1$. This broad variety of coloring problems makes it tempting not to try to solve each of these problems on its own but to find efficient algorithms for many of them at once. The probably best-known meta-theorem in graph theory is Courcelle's theorem [25], which states that every graph problem expressible in Monadic Second-Order graph logic (MSO_2) can be solved in linear time on graphs of bounded treewidth.

Locally checkable problems are a broad subclass of coloring problems. A locally checkable problem is associated with a *check function*, which is a function that takes as input a vertex v of a graph G and a coloring of the closed neighborhood of v and outputs a Boolean value. We are looking for a coloring c of the vertices of G such that the check function outputs True for every vertex v and the restriction of c to $N[v]$. Such a coloring will be called a *proper coloring*, since this is a generalization of the notion of proper colorings in the problem Chromatic Number. Indeed, determining the chromatic

* Corresponding author.
E-mail addresses: cgonzalez@dc.uba.ar (C.L. Gonzalez), felix.mann@unifr.ch (F. Mann).

https://doi.org/10.1016/j.dam.2023.12.015

number can be seen as a locally checkable problem, where the check function simply verifies that the color of a vertex v is unique in its closed neighborhood. Another example for a locally checkable problem is the domination problem. Here, we are coloring the graph with two colors, S and $\overline{S}$, and the check function verifies if the closed neighborhood of a vertex v contains at least one vertex of color S. Then, for any proper coloring, the vertices which have color S form a dominating set. Once a graph and a check function are given, we could ask not only whether there exists a proper coloring but also for an optimal proper coloring (for some notion of optimality).

Most locally checkable problems are NP-complete on general graphs. For a given family of graphs it is thus interesting to determine under which conditions on the check function and the color set we can efficiently solve the associated locally checkable problems on that family. In [20] Bonomo-Braberman and Gonzalez showed that, under mild conditions on the check function,[1] locally checkable problems can be solved in polynomial time in graphs of bounded treewidth.[2] This line of research has been continued in [8] in which Baghirova, Gonzalez, Ries and Schindl specified a class of check functions called *color-counting* check functions. These functions only depend on the vertex, the color it receives and, for each color a, the number of neighbors of v of color a. In other words, for a graph G and a set of q colors, the check function can be written as a function *check* which takes as input a vertex v, a color and, for each color a, a non-negative integer k_a. In order to determine if a coloring c is proper, we verify if $check(v, c(v), k_1, \ldots, k_q) = \textsc{True}$ for every $v \in V(G)$, where $k_a = |\{w \in V(G) : c(w) = a \text{ and } w \text{ is a neighbor of } v\}|$w. It is shown in [8] that the problems associated to a color-counting check function can be solved in polynomial time on graphs of bounded clique-width if the number of colors q is a constant. If we ease these restrictions on the check function and on q, we obtain locally checkable problems which are NP-hard even on complete graphs (see [20]) and thus, in this case, we do not expect to find a polynomial-time algorithm.

In this paper we turn our attention to graphs which are bounded in another width parameter, the *maximum induced matching width* (or *mim-width* for short), which was introduced by Vatshelle in [39] via the use of decomposition trees. A *binary decomposition tree* (T, δ) of G consists of a rooted binary tree T and a bijection δ between the leaves of T and the vertices of G (see [39, Definition 3.1.3]). For any node v of T there is an associated partition of the vertices into two sets: the vertices corresponding to leaves which are descendants of v, denoted T_v, and all other vertices, denoted $T_{\overline{v}}$. The *mim-width of a graph* is the smallest number k such that there is a binary decomposition tree (T, δ) of G such that, for every node v of T, the maximum induced matching in the bipartite graph $G[T_v, T_{\overline{v}}]$ has size at most k. A more detailed definition is given in Section 2.

In a sense, mim-width is a broader width parameter than treewidth or clique-width. Indeed, any graph class of bounded clique-width has bounded mim-width, but there are graph classes of bounded mim-width whose clique-width is unbounded, such as interval graphs and permutation graphs [10,29]. We observe that there are color-counting check functions with a bounded number of colors whose corresponding locally checkable problems are NP-hard on graphs of bounded mim-width (see Section 6). Thus, we study which further restrictions we have to impose on a check function such that the associated locally checkable problems can be solved in polynomial time on graphs of bounded mim-width.

Despite the broadness of families of bounded mim-width, there is still a surprising number of problems that can be solved in XP time when parameterized by mim-width. In one of the first works to identify a large family of such problems [22], Bui-Xuan, Telle and Vatshelle focused on the following problems, defined by Telle in [37], which are related to a special type of color-counting check functions. Given a positive integer q, a *degree constraint matrix* D is a $q \times q$-matrix whose entries are sets of non-negative integers. A *locally checkable vertex partitioning (LCVP)* problem is associated to a degree constraint matrix D and consists in deciding whether there exists a partition $\{V_1, V_2, \ldots, V_q\}$ of $V(G)$ such that for every $i, j \in [q]$ and every $v \in V_i$ it holds that $|N(v) \cap V_j| \in D[i, j]$. In [37] there is also a broad list of problems given which can be expressed as LCVP problems, amongst which are Dominating set and Independent set. In [31], Jaffke, Kwon, Strømme and Telle showed that a generalization of LCVP problems can be solved in polynomial time on graphs of bounded mim-width if q is a constant and every entry of the associated degree constraint matrix is either finite or co-finite (that is, the complement of a finite set). Observe that all algorithms whose runtime is polynomial when the mim-width is bounded only work under the assumption that a suitable binary decomposition tree is given with the graph. While for some graph classes there are efficient algorithms to obtain such a decomposition (see, for example, [10]), it is not known whether it is possible to obtain a binary decomposition tree of some graph G with the smallest possible mim-width in polynomial time. Even if we only ask for a binary decomposition tree of mim-width $f(mimw(G))$ for *any* function f, we do not know if this is possible.

Every LCVP problem can be considered as a locally checkable problem with a color-counting check function. Indeed, given a degree constraint matrix D, we can associate the following color-counting check function to its corresponding LCVP problem:

$$check(v, i, k_1, \ldots, k_q) = \left(\forall j \in [q],\ k_j \in D[i, j]\right).$$

Observe that there are color-counting check functions which do not correspond to any LCVP problem. For example, the k-domination problem asks for a coloring of the vertices of the graph with the numbers from 0 to k such that, for each vertex v, the sum of the colors assigned to the vertices in the closed neighborhood of v is at least k, and where we want to

[1] Namely, the existence of *polynomial partial neighborhood systems*. We omit the definition due to its technicality, but interested readers are encouraged to refer to [20].

[2] Some of these problems are in fact FPT parameterized by treewidth, while the others are simply XP.

minimize the sum of the colors of all the vertices in the graph. Another example is the conflict-free q-coloring, which asks for a coloring of the vertices with q colors such that in the neighborhood of each vertex there is a color which appears exactly once.

In order to generalize LCVP problems to include problems as the ones above, we introduce the notion of a *d-stable* check function.

Definition 1.1. Let $\{a_1, \ldots, a_q\}$ be a set of colors and let *check* be a color-counting check function. Let d be a positive integer. We say that *check* is a *d-stable check function* if for every $v \in V(G)$ and any coloring c of $V(G)$ we have

$$check(v, c(v), k_1, \ldots, k_q) = check(v, c(v), \min(d, k_1), \ldots, \min(d, k_q))$$

where $k_j = \left|\left\{w \in V(G) : c(w) = a_j \text{ and } w \text{ is a neighbor of } v\right\}\right|$w for every $j \in [q]$.

In other words, if a check function is d-stable that means that the "properness" of a coloring does not depend on the exact number of neighbors with a certain color as long as it is "large enough" (that is, larger than the threshold d). For example, for the domination problem, we only need to know if every vertex has zero or at least one neighbor in the dominating set. However, the exact number of neighbors in the set is not important if it is larger than one. Thus, this problem has a 1-stable check function.

Observe that in the result of [22] the condition that every entry of a degree constraint matrix D is either finite or co-finite guarantees that its associated check function is d-stable for some d (the value of d depends on the matrix D, see [22]). Thus, the notion of d-stability generalizes the set of LCVP problems which can be efficiently solved by the methods in [22].

We show that the d-stability of a check function suffices to yield an algorithm which solves the associated locally checkable problems in polynomial time on graphs of bounded mim-width (given a suitable binary decomposition tree). We also show that it is possible to solve optimization versions of these problems, by considering a function which assigns weights depending on the coloring of the graph. Given a coloring c of $V(G)$ with colors in $\{a_1, \ldots, a_q\}$, a *local weight function* assigns weights to vertices depending on the vertex and the coloring of its closed neighborhood. Then the weight of the coloring c is defined as the sum of the weights of all the vertices. We consider local weight functions which have properties analogous to the ones previously defined for the check functions. A *color-counting weight function* is a local weight function where the assigned weights depend on the vertex v, the color it receives and, for each color a, the number of neighbors of v which have color a. For a positive integer d, we say that a color-counting weight function w is d-stable if for every $v \in V(G)$ and every coloring c we have $\mathrm{w}(v, c(v), k_1, \ldots, k_q) = \mathrm{w}(v, c(v), \min(d, k_1), \ldots, \min(d, k_q))$, where $k_j = |N(v) \cap c^{-1}(a_j)|$ for every $j \in [q]$. A locally checkable problem with both a d-stable check function and a d-stable weight function will be called a *d-stable locally checkable problem*. We adapt the algorithm given in [22] to show that finding an optimal proper coloring for such a problem is possible in polynomial time in graphs of bounded mim-width. We also give the option to add global constraints on the sizes of the color classes and also connectivity constraints (among which we distinguish between *positive* and *negative* constraints, depending on whether they ask for a set of vertices to be connected or disconnected), for which we follow the ideas of Bergougnoux and Kanté in [15].

Theorem 1.2. *Let d and q be positive integers. For a d-stable locally checkable problem with q colors, size constraints $\mathcal{K}$, positive connectivity constraints $\mathcal{C}^+$ and negative connectivity constraints $\mathcal{C}^-$, there exists an algorithm that solves the problem on graphs with a given binary decomposition tree in time $n^{O(wdq(|\mathcal{C}^+|+1)2^{|\mathcal{C}^-|})}$, where n is the number of vertices of the input graph and w is the mim-width of the associated decomposition.*

We would like to analyze the relation of these d-stable functions with another subset of graph problems. The *distance neighborhood (DN) logic* was introduced by Bergougnoux, Dreier and Jaffke in [14] as an extension of existential MSO_1, that is, MSO_1 without universal quantifiers and without negation of terms containing existential quantifiers. DN logic introduces further the use of *neighborhood terms*, whose basic element is the following *neighborhood operator*: for a set S of vertices of a graph, $N_d^r(S)$ denotes the set of every vertex v for which there are at least d vertices in S whose distance to v is at most r and at least 1. This logic allows us to combine these terms using the usual set operators as well as constant sets of vertices. We can also use them in logical expressions by comparing sizes of these sets, asking for equality or if they are contained in each other. A precise definition is given in [14]. The A&C DN logic is an extension of DN logic with the terms con(t) and acy(t), which state that the subgraph induced by the neighborhood term t is connected or acyclic, respectively. In Section 4 we explore the modeling power of the frameworks. For any A&C DN expression there is an equivalent formulation using d-stable locally checkable problems with size, connectivity and acyclicity constraints. Conversely, we can model any d-stable locally checkable problem with such global constraints in A&C DN logic. It was also shown in [14] that any A&C DN formula can be solved in polynomial time on graphs of bounded mim-width, yielding another polynomial time algorithm solving d-stable locally checkable problems on graphs of bounded mim-width, besides Theorem 1.2. Since the two models look quite different, it may seem surprising that they can solve the same set of problems. However, this astonishment is dampened given the fact that they are both generalizations of the proof idea in [22].

The additional translation step comes at the expense of a worse runtime. For a d-stable coloring problem on q colors without connectivity or acyclicity constraints and an n-vertex graph G whose mim-width is w, the algorithm in

Section 5 has a runtime of $n^{O(dwq)}$. Using the proof in Section 4.2 and [14, Theorem 1.2] gives an algorithm of complexity $n^{O(wq^2d^2(d+1)^{2q})}$. Clearly, the first algorithm has a better exponent. In some cases, it is possible to find shorter DN-expressions for a locally checkable problem than the ones we obtain from Section 4.2. For example, in [14, Section 6.3], there is a DN-formula expressing the conflict-free q-coloring problem which has length $O(q^2)$. This is shorter than $O(q^2 2^{2q+1})$, the length we obtain from Section 4.2, but this still yields an algorithm of runtime $n^{O(dwq^2)}$ and thus a slower one than Section 5.

Finally, it is certainly of interest to determine if the restrictions we put on the check and weight function as well as on the number of colors could be eased. In Section 6 we show that this is not possible in general. In particular, we show that there are locally checkable problems with non d-stable color-counting check functions which are NP-hard even on interval graphs. Thus, we cannot expect to remove the condition of d-stability. We also give locally checkable problems with d-stable check function for which the associated optimization problem is NP-hard for certain non d-stable color-counting weight functions.

Our paper is structured as follows. We begin by explaining basic notions and definitions in Section 2. The definition and notation of d-stable locally checkable problems is given in Section 3. In Section 4 we analyze the relation between DN logic and d-stable locally checkable problems. We then give a proof for our main result, Lemma 5.2, in Section 5. Section 6 is dedicated to showing that the restrictions we give on the number of colors and on the check and weight functions are in some sense tight. Finally, Section 7 provides examples of problems which are associated to a d-stable check function and which have not been solved before on graphs of bounded mim-width.

2. Preliminaries

Let k be a positive integer and denote by $[k]$ the set $\{1, \dots, k\}$. We denote by $\mathbb{N}$ the set of non-negative integers, and by $\mathbb{F}_2$ the field with two elements. For a set S, $\mathcal{P}(S)$ denotes the set of all subsets of S.

Throughout this paper we will work with the set $\textsc{Bool} = \{\textsc{True}, \textsc{False}\}$ of Boolean values (truth values), and with all the usual logical operators, such as $\neg$, $\wedge$, $\vee$ and $\Rightarrow$, as well as with the quantifiers $\forall$, $\exists$. We use the following notation style for quantifiers: $\forall x \in X,\ P$ and $\exists x \in X,\ P$. Let $\mathbb{1} : \textsc{Bool} \to \mathbb{F}_2$ be the function such that $\mathbb{1}(\textsc{True}) = 1$ and $\mathbb{1}(\textsc{False}) = 0$.

Let k and q be non-negative integers. We denote by $\mathrm{part}_q(k)$ the set of all q-tuples of non-negative integers such that the sum of all entries is k. For a set S, we denote by $\mathrm{part}_q(S)$ the set of ordered partitions of S into q parts. For some $X \in \mathrm{part}_q(S)$, we denote by $|X|$ the q-tuple with $|X|_i = |X_i|$. If $X \in \mathrm{part}_q(S)$ and S' is another set then we denote by $S' \cap X$ the q-tuple with $(S' \cap X)_i = S' \cap X_i$. If S and S' are disjoint sets and $X \in \mathrm{part}_q(S)$, $X' \in \mathrm{part}_q(S')$ then we denote by $X \cup X'$ the entry-wise union of X and X', which is a partition of $S \cup S'$.

When S and S' are sets, an *S-tuple with entries from S'* is a map from S to S'. Given an S-tuple T for which every $x \in S$ is mapped to T_x, we sometimes write $T = (T_x)_{x \in S}$.

2.1. Basic graph theory definitions

Throughout the whole paper, we assume that all graphs are finite, simple and undirected, unless stated differently.

Given a graph G, we denote with $V(G)$ the vertex set of G and with $E(G)$ its edge set. For a vertex $v \in V(G)$, the *open neighborhood of v* ($N_G(v)$) is the set of neighbors of v in G, and the *closed neighborhood of v* is $N_G[v] = N_G(v) \cup \{v\}$. The *closed neighborhood of a set S* $\subseteq V(G)$ is $N_G[S] = \bigcup_{v \in S} N_G[v]$. The *degree* of a vertex v is $d_G(v) = |N_G(v)|$. We may omit the sub-index G when it is clear from the context.

Given two disjoint sets $S, S' \subseteq V(G)$, we denote by $G[S]$ the subgraph of G induced by S, and by $G[S, S']$ the bipartite graph of vertex set $S \cup S'$ and edge set $\left\{vv' \in E(G) : v \in S, v' \in S'\right\}$.

A graph G is *connected* if for every pair of vertices $u, v \in V(G)$ there exists a path in G from u to v. A *connected component* of a graph is the set of vertices of an inclusion-wise maximal connected subgraph. For two vertices u, v in a connected graph G, the *distance between u and v* is the number of edges in a shortest path from u to v in G, and we denote it by $\mathrm{dist}_G(u, v)$. Let $cc(G)$ denote the set of connected components of a graph G. We define $ccut(G) = \mathrm{part}_2(cc(G))$. An element of $ccut(G)$ is called a *connected cut*. For a set $S \subseteq V(G)$, let $ccut(S) = ccut(G[S])$.

Let r be a positive integer. The *rth power* of a graph G, denoted G^r, is the graph of vertex set $V(G)$ such that for all distinct vertices $u, v \in V(G)$, u is adjacent to v in G^r if and only if $\mathrm{dist}_G(u, v) \leq r$. The *$r$-neighborhood* $N_G^r(v)$ of a vertex v in G is the set of vertices different from v which are at distance at most r to v in G. Equivalently, this is the neighborhood of v in the rth graph power of G. That is, $N_G^r(v) = N_{G^r}(v)$.

A *rooted tree* is a tree with a distinguished vertex called *root*. We will refer to the vertices of rooted trees as *nodes*. A *rooted binary tree* is a rooted tree in which every non-leaf node has two children. For a graph G, a rooted binary tree T together with a bijection δ between $V(G)$ and the leafs of T is called a *binary decomposition tree* of G ([39, Definition 3.1.3]). Let (T, δ) be a binary decomposition tree of a graph G. Let $v \in V(T)$, we denote by T_v the set of vertices of G which correspond to the leaves which are descendants of v. We denote $T_{\overline{v}} = V(G) \setminus T_v$.

Let $mim(G, S)$ denote the maximum size of an induced matching in the bipartite graph $G[S, V(G) \setminus S]$ (measured as number of edges), and let $mimw(T, \delta) = \max_{v \in V(T)} \{mim(G, T_v)\}$. The mim-width of a graph G, denoted $mimw(G)$, is the minimum value of $mimw(T, \delta)$ over all binary decomposition trees (T, δ) of G. Given a graph class $\mathcal{G}$, we say that $\mathcal{G}$ is of *bounded mim-width* if $\sup\{mimw(G) : G \in \mathcal{G}\} < \infty$. All graphs classes of bounded clique-width have bounded mim-width, but there exist graphs classes (for example, interval graphs) of mim-width 1 and unbounded clique-width [29,39].

2.2. Weight sets

Let (Weights, $\preceq$) be a totally ordered set with a maximum element (called Error), together with the minimum operation of the order $\preceq$ (called min) and a closed binary operation on Weights (called $\oplus$) that is commutative and associative, has a neutral element and an absorbing element that is equal to Error, and the following property is satisfied: $s_1 \preceq s_2 \Rightarrow s_1 \oplus s_3 \preceq s_2 \oplus s_3$ for all $s_1, s_2, s_3 \in$ Weights. In such a case, we say that (Weights, $\preceq$, $\oplus$) is a *weight set*.

A classic example of a weight set is $(\mathbb{N} \cup \{+\infty\}, \leq, +)$. Notice that the maximum element is $+\infty$ in this case. We could also consider the reversed order of natural weights: $(\mathbb{N} \cup \{-\infty\}, \geq, +)$, where the maximum element is now $-\infty$. Another simple example worth mentioning is $(\{0, 1\}, \leq, \max)$.

3. *d*-stable locally checkable problems

In this paper we will consider the following type of problems on graphs, which we will call *d-stable locally checkable problems*. Let d and q be positive integers and suppose we are given:

- a simple undirected graph G,
- a set Colors $= \{a_1, \ldots, a_q\}$ of q colors,
- a weight set (Weights, $\preceq$, $\oplus$),
- a d-stable weight function $\mathrm{w}: V(G) \times$ Colors $\times \mathbb{N}^q \to$ Weights, and
- a d-stable check function $check: V(G) \times$ Colors $\times \mathbb{N}^q \to$ Bool.

We say that a coloring $c: V(G) \to$ Colors of the input graph G is a *proper coloring* if for every vertex $v \in V(G)$ we have that $check(v, c(v), k_1, \ldots, k_q) =$ True, where $k_j = |\left\{u \in N_G(v) : c(u) = a_j\right\}|$ for all $j \in [q]$. The *weight of a coloring* c is defined as $\mathrm{w}(c) = \bigoplus_{v \in V(G)} \mathrm{w}(v, c(v), k_1, \ldots, k_q)$ where $k_j = |\left\{u \in N_G(v) : c(u) = a_j\right\}|$ for all $j \in [q]$. The goal is to find the minimum[3] weight of a proper coloring of the input graph. Notice that if no such coloring exists then the answer is the maximum element of the weight set.

Observe that a d-stable check function has $|V(G)|q(d+1)^q$ different inputs, so its runtime could be dependent on $|V(G)|$. However, throughout this paper we implicitly demand that the runtime of every check function only depends on d and q. Furthermore, we require that applying $\oplus$ is a constant time operation.

We can further consider size, connectivity and acyclicity constraints:

- a set $\mathcal{K} \subseteq \mathrm{part}_q(|V(G)|)$ of size constraints,
- a set $\mathcal{C}^+ \subseteq \mathcal{P}$(Colors) of positive connectivity constraints and a set $\mathcal{C}^- \subseteq \mathcal{P}$(Colors) of negative connectivity constraints,
- a set $\mathcal{A}^+ \subseteq \mathcal{P}$(Colors) of positive acyclicity constraints and a set $\mathcal{A}^- \subseteq \mathcal{P}$(Colors) of negative acyclicity constraints,

and we ask for the minimum weight of a proper coloring c of the vertices of G such that:

- the q-tuple $(|\{v \in V(G) : c(v) = a_1\}|, \ldots, |\left\{v \in V(G) : c(v) = a_q\right\}|)$ is in $\mathcal{K}$ (that is, the sizes of the q color classes are "acceptable"),
- the subgraph of G induced by the set $\{v \in V(G) : c(v) \in C\}$ is connected (respectively not connected) for every $C \in \mathcal{C}^+$ (respectively $\mathcal{C}^-$), and also
- the subgraph of G induced by the set $\{v \in V(G) : c(v) \in A\}$ is acyclic (respectively not acyclic) for every $A \in \mathcal{A}^+$ (respectively $\mathcal{A}^-$).

Notice that $\mathcal{K}$ can capture *any* conditions on the sizes of the color classes. We can therefore more naturally implicitly define the set $\mathcal{K}$ by stating these conditions in words. The same holds for $\mathcal{C}$ and $\mathcal{A}$.

In order to illustrate these definitions better, we present the following examples of well-known problems modeled as d-stable locally checkable problems (with or without additional global constraints).

Example 3.1. Consider the Maximum Independent Set problem. This problem can be seen as a 1-stable locally checkable problem with two colors:

- Colors $= \{\mathrm{s}, \bar{\mathrm{s}}\}$,
- (Weights, $\preceq$, $\oplus$) $= (\mathbb{N} \cup \{-\infty\}, \geq, +)$,
- $\mathrm{w}(v, \mathrm{s}, k_{\mathrm{s}}, k_{\bar{\mathrm{s}}}) = 1$ and $\mathrm{w}(v, \bar{\mathrm{s}}, k_{\mathrm{s}}, k_{\bar{\mathrm{s}}}) = 0$ for all $v \in V(G)$ and $k_{\mathrm{s}}, k_{\bar{\mathrm{s}}} \in \mathbb{N}$,
- $check(v, a, k_{\mathrm{s}}, k_{\bar{\mathrm{s}}}) = (a = \bar{\mathrm{s}} \vee k_{\mathrm{s}} = 0)$ for all $v \in V(G)$, $a \in$ Colors and $k_{\mathrm{s}}, k_{\bar{\mathrm{s}}} \in \mathbb{N}$.

In a proper coloring c, the set of vertices receiving color s is independent by definition of the check function. Conversely, every independent set defines a proper coloring. In addition, by definition of the weights, $\mathrm{w}(c) = |\{v \in V(G) : c(v) = \mathrm{s}\}|$. Therefore, the minimum weight of a proper coloring equals the maximum size of an independent set.

[3] According to the order $\preceq$ of the weight set.

Example 3.2. The q-Equitable Coloring problem can be modeled as a 1-stable locally checkable problem with q colors:

- $\textsc{Colors} = [q]$,
- $(\textsc{Weights}, \preceq, \oplus) = (\{0, 1\}, \leq, \max)$,
- $\mathrm{w}(v, a, k_1, \ldots, k_q) = 0$ for all $v \in V(G)$, $a \in \textsc{Colors}$ and $k_1, \ldots, k_q \in \mathbb{N}$,
- $\mathit{check}(v, a, k_1, \ldots, k_q) = (k_a = 0)$ for all $v \in V(G)$, $a \in \textsc{Colors}$ and $k_1, \ldots, k_q \in \mathbb{N}$,
- for all $j \in [q]$, the size of the color class j is $\lfloor \frac{n}{q} \rfloor$ or $\lceil \frac{n}{q} \rceil$.

In this case, we are not interested in the weight of the coloring but only in the existence of it. Thus, the weights here serve the sole purpose of distinguishing if the solution is the maximum element 1 (there does not exist any proper coloring satisfying the size constraints) or 0 (such a coloring indeed exists). And it is easy to see that a q-equitable coloring exists if and only if a proper coloring exists.

Example 3.3. The Minimum Connected Dominating Set problem can also be expressed as a 1-stable locally checkable problem with two colors:

- $\textsc{Colors} = \{\mathrm{s}, \bar{\mathrm{s}}\}$,
- $(\textsc{Weights}, \preceq, \oplus) = (\mathbb{N} \cup \{+\infty\}, \leq, +)$,
- $\mathrm{w}(v, \mathrm{s}, k_{\mathrm{s}}, k_{\bar{\mathrm{s}}}) = 1$ and $\mathrm{w}(v, \bar{\mathrm{s}}, k_{\mathrm{s}}, k_{\bar{\mathrm{s}}}) = 0$ for all $v \in V(G)$ and $k_{\mathrm{s}}, k_{\bar{\mathrm{s}}} \in \mathbb{N}$,
- $\mathit{check}(v, a, k_{\mathrm{s}}, k_{\bar{\mathrm{s}}}) = (a = \mathrm{s} \vee k_{\mathrm{s}} = 1)$ for all $v \in V(G)$, $a \in \textsc{Colors}$ and $k_{\mathrm{s}}, k_{\bar{\mathrm{s}}} \in \mathbb{N}$,
- the color class $\{\mathrm{s}\}$ is connected.

In a proper coloring, the set of vertices receiving color s is dominating by definition of *check*. Furthermore, by considering $\mathcal{C} = \{\{\mathrm{s}\}\}$, we are asking that the set of vertices with color s induces a connected subgraph. Conversely, every connected dominating set defines a proper coloring satisfying the connectivity constraint. Since the weights of vertices with color s is 1 and the weight of those with color $\bar{\mathrm{s}}$ is 0, then the weight of the coloring is equivalent to the size of vertices receiving color s. Therefore, the minimum weight of a proper coloring satisfying the connectivity constraints equals the minimum size of a connected dominating set.

More examples of d-stable locally checkable problems with size and connectivity constraints can be found in Section 7. Many of the examples mentioned in [20, Table 1] are also d-stable (although this is not explicitly mentioned). It should further be noted that the "list versions" of these problems, where each vertex v has a list L_v of possible colors that it can receive, can be modeled by adding the condition "$a \in L_v$" in the definition of $\mathit{check}(v, a, k_1, \ldots, k_q)$ for every vertex v.

We can also consider problems where the local property involves not only the neighbors but vertices at larger distances. Let t be a fixed positive integer and let $r_1, \ldots, r_t$ be positive integers. We can modify the domain of d-stable check and weight functions to be $V(G) \times \textsc{Colors} \times \mathbb{N}^{qt}$ instead of $V(G) \times \textsc{Colors} \times \mathbb{N}^{q}$, where q is the number of colors. That is, instead of having an input vector $k = (k_1, \ldots, k_q)$ where each k_j indicates the number of neighbors of color a_j, we now have $\kappa = ((\kappa_1^1, \ldots, \kappa_q^1), \ldots, (\kappa_1^t, \ldots, \kappa_q^t))$ where each κ_j^i indicates the number of vertices of color a_j that are at distance at most r_i. In this way, we have a check function $\mathit{check}\colon V(G) \times \textsc{Colors} \times \mathbb{N}^{qt} \to \textsc{Bool}$ and say that a coloring c is a proper coloring if for every vertex $v \in V(G)$ we have that $\mathit{check}(v, c(v), \kappa) = \textsc{True}$, where $\kappa_j^i = |\left\{u \in N_G^{r_i}(v) : c(u) = a_j\right\}|$ for all $i \in [t]$ and $j \in [q]$. The weight function can be extended analogously. We say that these functions are *d-stable for t distances* $r_1, \ldots, r_t$. A locally checkable problem with such functions will be called *d-stable locally checkable for t distances* $r_1, \ldots, r_t$. We may omit mentioning $r_1, \ldots, r_t$ when they are irrelevant.

4. Comparison with A&C DN logic

In this section we show that the set of problems that can be modeled using A&C DN logic formulas of constant length is equivalent to the set of problems that can be modeled using locally checkable problems with constant number of colors, functions that are d-stable for t distances (for some constant positive integers d and t), and size, connectivity and acyclicity constrains.

The *distance neighborhood (DN) logic*, defined in [14], is obtained by extending existential MSO_1 with a *neighborhood operator* $N_d^r(\cdot)$: for a term t that represents a set U of vertices in a graph G, the term $N_d^r(t)$ represents the set of all vertices in G which are at distance at most r and at least 1 from at least d vertices in U. The DN logic also includes size measurement of terms ($|\cdot| \leq \cdot$) and comparison between terms (with = and $\subseteq$). For a DN logic formula φ, $d(\varphi)$ denotes the greatest integer d such that there is an operator $N_d^{\cdot}(\cdot)$ in φ. Further, $R(\varphi)$ denotes the set of integers r such that there is an operator $N_{\cdot}^r(\cdot)$ in φ.

The *A&C DN logic* is a further extension of DN logic with the formulas $\mathrm{con}(\cdot)$ and $\mathrm{acy}(\cdot)$. Here, the expression $\mathrm{con}(t)$ ($\mathrm{acy}(t)$, respectively) represents that $G[U]$ is connected (acyclic, respectively), where U is the set of vertices represented by t.

It was shown in [14] that any A&C DN formula φ can be expressed as an *A&C CDN formula* φ', which is a formula of the form $\exists X_1 \ldots X_k,\ \psi$ where ψ is a Boolean combination (that is, an expression using $\wedge$, $\vee$ and $\neg$) of the following types of *primitive formulas*:

1. $(P = X_i)$, for a fixed subset P and some variable X_i,
2. $(X_i = X_j)$, for some variables X_i, X_j,
3. $(X_i = \overline{X_j})$, for some variables X_i, X_j,
4. $(X_i \cap X_j = X_h)$, for some variables X_i, X_j, X_h,
5. $(N_d^r(X_i) = X_j)$, for some variables X_i, X_j and a positive integer d,
6. $(|X_i| \leq m)$, for some variable X_i and a positive integer m,
7. $\mathrm{acy}(X_i)$, for some variable X_i,
8. $\mathrm{con}(X_i)$, for some variable X_i.

Moreover, the length of φ' is at most ten times the length of φ, $d(\varphi') = d(\varphi)$ and $R(\varphi') = R(\varphi)$ ([14, Observation 3.1]).

4.1. DN logic formulas as d-stable locally checkable problems

In the following we will show how to model an A&C DN formula with d-stable locally checkable problems for t distances, for some constants t and d, with size, connectivity and acyclicity constraints.

Lemma 4.1. *Let Π be a problem that can be expressed by an A&C DN formula φ. Then, there exist some positive integers $p \leq 2^{10|\varphi|}$ and $t \leq |\varphi|$, and a set $\{\Pi_1, \ldots, \Pi_p\}$ of $d(\varphi)$-stable locally checkable problems for t distances with size, connectivity and acyclicity constraints, such that, for a given input, the optimal among all the optimal solutions of $\Pi_1, \ldots, \Pi_p$ is equivalent to the optimal solution of Π.*

Proof. Let G be the input graph and ϕ an A&C CDN formula equivalent to φ, where $\phi = (\exists X_1 \ldots X_k,\ \psi)$ and ψ is a Boolean combination of primitive formulas of the eight types given above. Let $A = A_1 \cup \cdots \cup A_8$ be the set of all of such primitive formulas in ψ, where A_i contains exactly the primitive formulas which are of type i, for every $i \in [8]$.

There are at most $|\psi|$ primitive formulas in A and thus at most $2^{|\psi|}$ truth assignments $T : A \to \textsc{Bool}$. For every truth assignment that satisfies ψ, we will show how to model an equivalent check function and size restrictions of color classes. Let T be a satisfying truth assignment. For every $i \in [8]$, let A_i^+ be the set of all primitive formulas from A_i which are set to $\textsc{True}$ by T. Set $A_i^- = A_i \setminus A_i^+$. In other words, the set A_i^+ contains all the primitive formulas of A_i which need to hold in order to satisfy ψ following the truth assignment T. Similarly, A_i^- contains all the formulas of A_i that must not hold according to T.

Let R be $R(\varphi)$ if this set is non-empty, otherwise we define R as $\{1\}$. Notice that the size of R is no longer than the length of φ.

Set $\textsc{Colors} = \mathcal{P}([k]) \times \mathcal{P}(A_1) \times \mathcal{P}(A_5)$. Intuitively, if a vertex v obtains the color (S, F_1, F_5), then S represents the subset of the variables $X_1, \ldots, X_k$ in which v is contained, and F_1 and F_5 save which of the formulas in A_1 and A_5, respectively, are satisfied locally at v. We will define a check function that only makes sure that F_1 and F_5 are set correctly according to this idea. This function will be $d(\varphi)$-stable for $|R|$ distances $r \in R$. For every $v \in V(G)$, $(S, F_1, F_5) \in \textsc{Colors}$ and $\ell \in \mathbb{N}^{|\textsc{Colors}| \times |R|}$ (whose entries we denote by $\ell^r_{(S', F_1', F_5')}$ for any $r \in R$ and $(S', F_1', F_5') \in \textsc{Colors}$), set $\mathit{check}(v, (S, F_1, F_5), \ell) = \textsc{True}$ if and only if the following two statements hold:

- for all $(P = X_i) \in A_1$, we have that $(P = X_i) \in F_1$ if and only if $(i \in S)$ and $(v \in P)$ are either both true or both false,
- for every $(N_d^r(X_i) = X_j) \in A_5$, we have that $(N_d^r(X_i) = X_j) \in F_5$ if and only if $(j \in S)$ and $\left(\sum_{F_1' \subseteq A_1} \sum_{F_5' \subseteq A_5} \sum_{S_i \subseteq [k], i \in S_i} \ell^r_{S_i, F_1', F_5'} \geq d\right)$ are both true or both false.

We define the following size, connectivity and acyclicity constraints which then guarantee that ψ is satisfied:

- For every $(P = X_i) \in A_1^+$ and for every $(S, F_1, F_5) \in \textsc{Colors}$ where F_1 does not contain $(P = X_i)$, the size of the color class (S, F_1, F_5) is 0.
- For every $(P = X_i) \in A_1^-$, there exists a color $(S, F_1, F_5) \in \textsc{Colors}$ such that F_1 does not contain $(P = X_i)$ and the size of the color class (S, F_1, F_5) is at least 1.
- For every $(X_i = X_j) \in A_2^+$ and for every $(S, F_1, F_5) \in \textsc{Colors}$ where $|S \cap \{i, j\}| = 1$, the size of the color class (S, F_1, F_5) is 0.
- For every $(X_i = X_j) \in A_2^-$, there exists a color $(S, F_1, F_5) \in \textsc{Colors}$ such that $|S \cap \{i, j\}| = 1$ and the size of the color class (S, F_1, F_5) is at least 1.
- For every $(X_i = \overline{X_j}) \in A_3^+$ and for every $(S, F_1, F_5) \in \textsc{Colors}$ where $|S \cap \{i, j\}| \neq 1$, the size of the color class (S, F_1, F_5) is 0.
- For every $(X_i = \overline{X_j}) \in A_3^-$, there exists a color $(S, F_1, F_5) \in \textsc{Colors}$ such that $|S \cap \{i, j\}| \neq 1$ and the size of the color class (S, F_1, F_5) is at least 1.
- For every $(X_i \cap X_j = X_h) \in A_4^+$ and for every $(S, F_1, F_5) \in \textsc{Colors}$ where $S \cap \{i, j, h\}$ is either $\{i, j\}$, $\{i, h\}$, $\{j, h\}$, or $\{h\}$, the size of the color class (S, F_1, F_5) is 0.
- For every $(X_i \cap X_j = X_h) \in A_4^-$, there exists a color $(S, F_1, F_5) \in \textsc{Colors}$ such that $S \cap \{i, j, h\} \in \{\{i, j\}, \{i, h\}, \{j, h\}, \{h\}\}$ and the size of the color class (S, F_1, F_5) is at least 1.

- For every $(N_d^r(X_i) = X_j) \in A_5^+$ and for every $(S, F_1, F_5) \in \textsc{Colors}$ where F_5 does not contain $(N_d^r(X_i) = X_j)$, the size of the color class (S, F_1, F_5) is 0.
- For every $(N_d^r(X_i) = X_j) \in A_5^-$, there exists a color $(S, F_1, F_5) \in \textsc{Colors}$ such that F_5 does not contain $(N_d^r(X_i) = X_j)$ and the size of the color class (S, F_1, F_5) is at least 1.
- For every $(|X_i| \leq m) \in A_6^+$, the sum of the sizes of all color classes (S, F_1, F_5) with $i \in S$ is at most m.
- For every $(|X_i| \leq m) \in A_6^-$, the sum of the sizes of all color classes (S, F_1, F_5) with $i \in S$ is at least $m + 1$.
- For every $\mathrm{acy}(X_i) \in A_7^+$, the set of all vertices with colors (S, F_1, F_5), where $i \in S$, induces an acyclic subgraph.
- For every $\mathrm{acy}(X_i) \in A_7^-$, the set of all vertices with colors (S, F_1, F_5), where $i \in S$, induces a non-acyclic subgraph.
- For every $\mathrm{con}(X_i) \in A_8^+$, the set of all vertices with colors (S, F_1, F_5), where $i \in S$, induces a connected subgraph.
- For every $\mathrm{con}(X_i) \in A_8^-$, the set of all vertices with colors (S, F_1, F_5), where $i \in S$, induces a non-connected subgraph.

In [14], graphs are assumed to have an associated weight $w(v, S) \in \mathbb{N}$ for all vertices v and sets $S \subseteq [k]$, and the weight of a k-tuple $B \in \mathcal{P}(V(G))^k$ is defined as $\sum_{v \in V(G)} w(v, \{i : v \in B_i\})$. Therefore, we can consider the weight set $(\textsc{Weights}, \preceq, \oplus) = (\mathbb{N} \cup \{-\infty\}, \geq, +)$ and define the weight function as $\mathrm{w}(v, (S, F_1, F_5), \ell) = w(v, S)$ for every $v \in V(G)$, $(S, F_1, F_5) \in \textsc{Colors}$ and $\ell \in \mathbb{N}^{|\textsc{Colors}| \times |R|}$.

Now in order to compute an optimal solution of Π, we have to compute an optimal solution of the locally checkable problem associated to each truth assignment T of the primitive formulas which satisfies ψ, and then simply choose the best solution amongst them. As mentioned above, the number of these truth assignments is at most $2^{|\psi|}$ and, by [14, Observation 3.1], this is at most $2^{10|\varphi|}$. □

4.2. Formulation of d-stable locally checkable problems in DN logic

In this section we show that any d-stable locally checkable problem for t distances using q colors, size constraints $\mathcal{K}$, connectivity constraints $\mathcal{C}$ and acyclicity constraints $\mathcal{A}$, can be expressed as formulas in A&C DN logic whose lengths are bounded by a function in d, q and t. This result, together with Theorem 1.2 in [14], implies that such a problem is XP parameterized by the mim-width of a given binary decomposition tree of the input graph, when all d, q and t are bounded by a constant. Moreover, it allows us to combine locally checkable problems with connectivity and acyclicity constraints, as well as with any other property that can be potentially expressed in this logic. However, the complexity of using this method turns out to be slightly worse than the one of the algorithm in Section 5, which, on the other hand, only allows connectivity constraints but not acyclicity. Indeed, for a d-stable locally checkable problem with q colors and no other constraints, the complexity given by the algorithm in Section 5 is $n^{O(wqd)}$ while the complexity resulting from the next lemma is $n^{O(wq^2d^2(d+1)^{2q})}$, where w is the mim-width of the given binary decomposition tree. However, both results imply that the problems are XP parameterized by w.

Lemma 4.2. *Consider a locally checkable problem Π on a graph G, with color set $\textsc{Colors} = \{a_1, \ldots, a_q\}$, check and weight functions that are d-stable for t distances $r_1, \ldots, r_t$, size constraints $\mathcal{K} \subseteq \mathit{part}_q(|V(G)|)$, positive and negative connectivity constraints $\mathcal{C}^+, \mathcal{C}^- \subseteq \mathcal{P}([q])$, and positive and negative acyclicity constraints $\mathcal{A}^+, \mathcal{A}^- \subseteq \mathcal{P}([q])$. Then, there exists a set $\Phi = \{\varphi_1, \ldots, \varphi_{|\mathcal{K}|}\}$ of A&C DN formulas such that solving Π is equivalent to finding the optimal among all the optimal solutions of the problems associated to the formulas in Φ, and the length of each $\varphi \in \Phi$ is $O(tq^2d(d+1)^{2qt})$.*

Proof. Let $K \in \mathcal{K}$. We will show how to give an A&C DN formula φ_K that expresses the problem of finding a minimum weight proper coloring that satisfies the size constraint K and the connectivity and acyclicity constraints $\mathcal{C}^+$, $\mathcal{C}^-$, $\mathcal{A}^+$ and $\mathcal{A}^-$. The desired set Φ is then the set $\{\varphi_K : K \in \mathcal{K}\}$. Indeed, an optimal solution of Π has to satisfy one of the size constraints of $\mathcal{K}$. An optimal solution of Π must then be the optimal solution amongst a set of optimal solution for each size constraint in $\mathcal{K}$.

Let $\mathcal{T} = [q] \times \{0, \ldots, d\}^{qt}$. For every $(j, \kappa) \in \mathcal{T}$, with $j \in [q]$ and $\kappa = ((\kappa_1^1, \ldots, \kappa_q^1), \ldots, (\kappa_1^t, \ldots, \kappa_q^t)) \in \{0, \ldots, d\}^{qt}$, we define the set $P_{(j,\kappa)} = \{v \in V(G) : \mathit{check}(v, a_j, \kappa) = \textsc{True}\}$.

First, notice that finding a minimum weight proper coloring is equivalent to finding a minimum weight partition of $V(G)$ into sets X_T, for $T \in \mathcal{T}$, such that the following conditions are satisfied:

1. $\mathit{check}(v, a_j, \kappa) = \textsc{True}$ (or, equivalently, $v \in P_{(j,\kappa)}$) for all tuples $(j, \kappa) \in \mathcal{T}$ and $v \in X_{(j,\kappa)}$,
2. $\kappa_j^i = \min\left(d, \left|N^{r_i}(v) \cap \left(\bigcup_{\ell \in \{0,\ldots,d\}^{qt}} X_{(j,\ell)}\right)\right|\right)$ for all $i \in [t]$, all $j \in [q]$, all tuples $(j, \kappa) \in \mathcal{T}$ and all $v \in X_{(j,\kappa)}$,
3. for all $j \in [q]$, the number of vertices in the set $\bigcup_{\kappa \in \{0,\ldots,d\}^{qt}} X_{(j,\kappa)}$ equals K_j (that is, the size constraint is satisfied for every color class),
4. for all $C \in \mathcal{C}^+$ (respectively $\mathcal{C}^-$), the graph induced by the set $\bigcup_{j \in C} \bigcup_{\kappa \in \{0,\ldots,d\}^{qt}} X_{(j,\kappa)}$ is connected (respectively not connected), that is, the connectivity constraints are satisfied,
5. for all $A \in \mathcal{A}^+$ (respectively $\mathcal{A}^-$), the graph induced by the set $\bigcup_{j \in A} \bigcup_{\kappa \in \{0,\ldots,d\}^{qt}} X_{(j,\kappa)}$ is acyclic (respectively not acyclic), that is, the acyclicity constraints are satisfied,

where the weights are defined as $w_{(j,\kappa)}(v) = \mathrm{w}(v, a_j, \kappa)$. In other words, we partition the vertices according to the color they receive and, for each $i \in [q]$ and $r \in R$, the number (up to d) of vertices of color a_i that they have at distance at most r.

In the next formulas we will make use of the following formula defined in [14, Proposition 6.4]:

$$\text{part}(X_{T\in\mathcal{T}}) \equiv \overline{\bigcup_{T\in\mathcal{T}} X_T} = \emptyset \wedge \bigwedge_{T,T'\in\mathcal{T}} X_T \cap X_{T'} = \emptyset.$$

In other words, part$(X_{T\in\mathcal{T}})$ evaluates to TRUE if the vertex sets $X_{T\in\mathcal{T}}$ partition the set $V(G)$. We will, as well, employ the following function, which is defined for some positive integer r and some finite or co-finite set $\mu \subseteq \mathbb{N}$:

$$t_{r,\mu}(X) = \left\{v \in V(G) : |N^r(v) \cap X| \in \mu\right\}.$$

This function was given in [14, Lemma 6.1], where it was also shown that $t_{r,\mu}(X)$ can be expressed as a DN term of length $O(d(\mu)x)$, where x is the length of a term expressing X.

Consider the following A&C DN formula φ_K. We claim that it expresses the existence of a partition satisfying the above 5 conditions.

$$\begin{aligned}\varphi_K(X_{T\in\mathcal{T}}) \equiv\ & \text{part}(X_{T\in\mathcal{T}}) \wedge \text{proper}(X_{T\in\mathcal{T}}) \wedge \text{neigh}(X_{T\in\mathcal{T}}) \\ & \wedge \text{size}(X_{T\in\mathcal{T}}) \wedge \text{conn}(X_{T\in\mathcal{T}}) \wedge \text{acyc}(X_{T\in\mathcal{T}}),\end{aligned}$$

where:

- $\text{proper}(X_{T\in\mathcal{T}}) \equiv \bigwedge_{T\in\mathcal{T}} X_T \subseteq P_T$,
- $\text{neigh}(X_{T\in\mathcal{T}}) \equiv \bigwedge_{(j,\kappa)\in\mathcal{T}} X_{(j,\kappa)} \subseteq \left(\bigcap_{i\in[t]} \bigcap_{h\in[q]} t_{r_i,f(\kappa_h^i)}\left(\bigcup_{\ell\in\{0,\dots,d\}^{qt}} X_{(h,\ell)}\right)\right)$, where the function f is defined to be such that $f(x) = \{x\}$ if $x < d$ and $f(x) = \{y \in \mathbb{N} : y \geq x\}$ otherwise,
- $\text{size}(X_{T\in\mathcal{T}}) \equiv \left(\bigwedge_{j\in[q]} \left|\bigcup_{\kappa\in\{0,\dots,d\}^{qt}} X_{(j,\kappa)}\right| = K_j\right)$,
- $\text{conn}(X_{T\in\mathcal{T}}) \equiv \left(\bigwedge_{C\in\mathcal{C}^+} \text{con}\left(\bigcup_{j\in C} \bigcup_{\kappa\in\{0,\dots,d\}^{qt}} X_{(j,\kappa)}\right)\right) \wedge \left(\bigwedge_{C\in\mathcal{C}^-} \neg\text{con}\left(\bigcup_{j\in C} \bigcup_{\kappa\in\{0,\dots,d\}^{qt}} X_{(j,\kappa)}\right)\right)$,
- $\text{acyc}(X_{T\in\mathcal{T}}) \equiv \left(\bigwedge_{A\in\mathcal{A}^+} \text{acy}\left(\bigcup_{j\in A} \bigcup_{\kappa\in\{0,\dots,d\}^{qt}} X_{(j,\kappa)}\right)\right) \wedge \left(\bigwedge_{A\in\mathcal{A}^-} \neg\text{acy}\left(\bigcup_{j\in A} \bigcup_{\kappa\in\{0,\dots,d\}^{qt}} X_{(j,\kappa)}\right)\right)$.

By definition, φ_K expresses the existence of sets $X_{T\in\mathcal{T}}$ such that each of the subformulas part$(X_{T\in\mathcal{T}})$, proper$(X_{T\in\mathcal{T}})$, neigh$(X_{T\in\mathcal{T}})$, size$(X_{T\in\mathcal{T}})$, conn$(X_{T\in\mathcal{T}})$ and acyc$(X_{T\in\mathcal{T}})$ holds. As mentioned above, part$(X_{T\in\mathcal{T}})$ guarantees that the sets $X_{T\in\mathcal{T}}$ form a partition of $V(G)$. The subformula proper$(X_{T\in\mathcal{T}})$ expresses Condition 1 for all the vertices. By definition of $t_{r,\mu}$, we have that the subformula neigh$(X_{T\in\mathcal{T}})$ assures that for every $(j,\kappa) \in \mathcal{T}$ we have $X_{(j,\kappa)} \subseteq \bigcap_{i\in[t]} \bigcap_{h\in[q]} \{v \in V(G) : \left|N^{r_i}(v) \cap \left(\bigcup_{\ell\in\{0,\dots,d\}^{qt}} X_{(h,\ell)}\right)\right| \in f(\kappa_h^i)\}$. In other words, it assures that every vertex $v \in X_{(j,\kappa)}$ is such that $\left|N^{r_i}(v) \cap \left(\bigcup_{\ell\in\{0,\dots,d\}^{qt}} X_{(h,\ell)}\right)\right| \in f(\kappa_h^i)$ for every $i \in [t]$ and every $h \in [q]$, which implies Condition 2. Finally, size$(X_{T\in\mathcal{T}})$, conn$(X_{T\in\mathcal{T}})$ and acyc$(X_{T\in\mathcal{T}})$ ensure that the size, connectivity and acyclicity constraints are satisfied (Condition 3 to 5).

Notice that $\mathcal{T}$ has $q(d+1)^{qt}$ elements. Then, the length of the formula part$(X_{T\in\mathcal{T}})$ is $O(q^2(d+1)^{2qt})$ and the length of proper$(X_{T\in\mathcal{T}})$ is $O(q(d+1)^{qt})$. Since the length of a term $t_{r_i,f(\kappa_h^i)}(X)$ is $O(d\cdot\text{length}(X))$, the length of neigh$(X_{T\in\mathcal{T}})$ is $O(tq^2d(d+1)^{2qt})$. The lengths of size$(X_{T\in\mathcal{T}})$, conn$(X_{T\in\mathcal{T}})$ and acyc$(X_{T\in\mathcal{T}})$ are, respectively, $O(q(d+1)^{qt})$, $O((|\mathcal{C}^+|+|\mathcal{C}^-|)q(d+1)^{qt})$ and $O((|\mathcal{A}^+|+|\mathcal{A}^-|)q(d+1)^{qt})$, since there are at most $q(d+1)^{qt}$ possible tuples (j,κ) and for the last two formulas we have a conjunction of $|\mathcal{C}^+|+|\mathcal{C}^-|$ and $|\mathcal{A}^+|+|\mathcal{A}^-|$ terms of length $O(q(d+1)^{qt})$ each. Therefore, and since $|\mathcal{C}^+|+|\mathcal{C}^-| \leq 2^q$ and $|\mathcal{A}^+|+|\mathcal{A}^-| \leq 2^q$, the length of the complete formula is $O(tq^2d(d+1)^{2qt})$. □

By [14, Theorem 1.2], each of the problems associated to the formulas $\varphi_1, \dots, \varphi_{|\mathcal{K}|}$ of a problem Π as in Lemma 4.2 can be solved in time $n^{O(wt^2q^4d^3(d+1)^{4qt})}$ where w is the mim-width of the given binary decomposition tree. Since $|\mathcal{K}| \leq n^{q-1}$, we do not have to apply the algorithm more than n^{q-1} times. We can then look for an optimal solution among the $|\mathcal{K}|$ solutions we have obtained. Hence, the total complexity of solving the problem Π is $n^{O(wt^2q^4d^3(d+1)^{4qt})}$.

5. Solving *d*-stable locally checkable problems

In this section we give a more straightforward algorithm to solve d-stable locally checkable problems with size and connectivity constraints. Its complexity is slightly better than the one obtained with the DN logic formulation. To be precise, we will show the following theorem:

Theorem 1.2. *Let d and q be positive integers. For a d-stable locally checkable problem with q colors, size constraints $\mathcal{K}$, positive connectivity constraints $\mathcal{C}^+$ and negative connectivity constraints $\mathcal{C}^-$, there exists an algorithm that solves the problem on graphs with a given binary decomposition tree in time $n^{O(wdq(|\mathcal{C}^+|+1)2^{|\mathcal{C}^-|})}$, where n is the number of vertices of the input graph and w is the mim-width of the associated decomposition.*

We first define some necessary terms and notation. Let d be a positive integer. We define an equivalence relation $=_d$ on the natural numbers, where two non-negative integers i and j are equivalent if and only if $\min(d,i) = \min(d,j)$. For a binary decomposition tree (T,δ) of a graph G and a node $v \in V(T)$, we make use of the equivalence relation $\equiv_d^v$ presented in [22]. It is defined on subsets of T_v where two subsets $S, S' \subseteq T_v$ are equivalent if and only if for any $y \in T_{\overline{v}}$ we have $|N(y) \cap S| =_d |N(y) \cap S'|$. Intuitively, from a perspective of locally checkable problems, we could say that two subsets of T_v are equivalent if replacing one for the other would not influence the output of any d-stable check function when applied

to any vertex y in $T_{\overline{v}}$. The idea behind this is that we do not want to save all partial solutions in a dynamic programming algorithm, and (roughly speaking) it is enough to save one representative for each equivalence class. One of the key ideas when working with this equivalence class is then to give an upper bound for the number of equivalence classes which then leads to an upper bound of the number of partial solutions that we have to save. Such an upper bound in terms of the mim-width of the decomposition of the graph was given in [10], where it was shown that the number of equivalence classes of $\equiv_d^v$ is at most $O(n^{d\cdot mimw(G)})$.

Based on $\equiv_d^v$, we define the equivalence relation $\equiv_{d,q}^v$ on q-tuples of subsets of T_v, where two q-tuples $X, X' \in (\mathcal{P}(T_v))^q$ are equivalent if and only if they are entry-wise equivalent according to $\equiv_d^v$. Analogously, we define the equivalence relations $\equiv_d^{\overline{v}}$ and $\equiv_{d,q}^{\overline{v}}$ where subsets S, S' of $T_{\overline{v}}$ are equivalent if for every vertex w in T_v we have $|N(w) \cap S| =_d |N(w) \cap S'|$, and analogously for tuples in $(\mathcal{P}(T_{\overline{v}}))^q$. We will mainly use these relations when referring to partitions of T_v and not arbitrary tuples of subsets.

For an equivalence relation $\equiv$ on a set S and $X \in S$, we denote by $[X]^{\equiv}$ the equivalence class which consists of all elements $X' \in S$ such that $X' \equiv X$. When the equivalence relation is clear from context, we will drop the superscript and only write $[X]$. We denote by $\mathcal{R}_d^v$ and $\mathcal{R}_{d,q}^v$ the set of all equivalence classes of the relations $\equiv_d^v$ and $\equiv_{d,q}^v$, respectively, and analogously for $\overline{v}$. If $R \in \mathcal{R}_{d,q}^v$ and $X \in (\mathcal{P}(T_v))^q$ we write $X \equiv_{d,q}^v R$ if X is contained in the equivalence class R and say that X is a *representative* of R (and analogously for $\overline{v}$). If $R \in \mathcal{R}_d^v$ and $w \in T_{\overline{v}}$ we define $|N(w) \cap R|$ to be the unique number $i \in \{0, \dots, d\}$ such that $i =_d |N(w) \cap S|$ for some $S \subseteq T_v$ with $S \equiv_d^v R$. This notion is well-defined by the definition of $\equiv_d^v$. We say that w is *adjacent* to R if $|N(w) \cap R| \geq 1$ and we say that a set $S \subseteq T_{\overline{v}}$ is adjacent to R if it contains a vertex which is adjacent to R. If $C \subseteq [q]$ and $X \in (\mathcal{P}(T_v))^q$, we denote by X_C the set $\bigcup_{i \in C} X_i \subseteq T_v$ and we also define the equivalence class $[X]_C$ of the relation $\equiv_d^v$ by setting $[X]_C = [X_C]$. We do everything analogously for $T_{\overline{v}}$. We define $\mathrm{snec}_d(T) = \max_{v \in V(T)}(\max\{|\mathcal{R}_d^v|, |\mathcal{R}_d^{\overline{v}}|\})$ and $\mathrm{snec}_{d,q}(T) = \max_{v \in V(T)}(\max\{|\mathcal{R}_{d,q}^v|, |\mathcal{R}_{d,q}^{\overline{v}}|\})$.

Remark 5.1. It was shown in [22, Lemma 1] that for a node v of T we can compute a list which contains exactly one representative of each equivalence class of $\equiv_d^v$, in time $O(\mathrm{snec}_d(T)\log(\mathrm{snec}_d(T))|V(G)|^2)$. The same applies to $\equiv_d^{\overline{v}}$. By [10, Lemma 2], $\mathrm{snec}_d(T) \leq n^{d\cdot mimw(T,\delta)}$. Since the equivalence classes of $\equiv_{d,q}^v$ and $\equiv_{d,q}^{\overline{v}}$ are q-tuples of equivalence classes of $\equiv_d^v$ and $\equiv_d^{\overline{v}}$, respectively, their representatives are simply the q-tuples of the representatives of $\equiv_d^v$ and $\equiv_d^{\overline{v}}$, respectively. Thus, we can also efficiently compute a list of representatives and $\mathrm{snec}_{d,q}(T) \leq n^{q\cdot d\cdot mimw(T,\delta)}$.

The next lemma shows that it is possible to solve, in polynomial time for graphs of bounded mim-width with a given suitable binary decomposition tree, a locally checkable problem with a d-stable check function, a d-stable weight function, a set of positive connectivity constraints $\mathcal{C} \subseteq \mathcal{P}([q])$ and a size constraint $\mathcal{K}$ with $|\mathcal{K}| = 1$. We will later explain how this result can be easily extended to a set $\mathcal{K}$ of several size constraints and to negative connectivity constraints. For the complexity analysis, we will use the fact that d and q are constants and that $|\mathcal{C}| \leq 2^q$ (and hence is bounded by a constant).

Lemma 5.2. *Let d, q be integers, with $q \geq 2$. Consider a d-stable locally checkable problem Π with q colors, a set $\mathcal{K}$ containing only one size constraint K, and a set of positive connectivity constraints $\mathcal{C}$. Then, for an input graph G with an associated binary decomposition tree (T, δ), we can solve Π in $O\left(n^{2q} \cdot \mathrm{snec}_{d,q}(T)^{6|\mathcal{C}|+6} \cdot \left(n^2 + \mathrm{snec}_{d,q}(T)^{\omega-1}\right)\right)$ time, where $n = |V(G)|$ and ω is the matrix multiplication exponent ($\omega < 2.4$, [18]).*

Proof. This proof closely follows the ones given in [22] and [15]. Our focus will be on highlighting that the d-stability of the check function is enough to guarantee the correctness of the output, and how the weight function and the restriction on the size of color classes can be implemented.

First of all, notice that finding a coloring c of G with q colors $\{a_1, \dots, a_q\}$ is equivalent to finding a q-partition $(X_1, \dots, X_q)$ of $V(G)$, where $X_j = c^{-1}(a_j)$ for all $j \in [q]$. We will therefore show how to compute a q-partition $X \in \mathrm{part}_q(V(G))$ such that $check(v, a, |N(v) \cap X_1|, \dots, |N(v) \cap X_q|) = \textsc{True}$ for all $v \in V(G)$ and $a \in \textsc{Colors}$, $|X| = K$ and $G[X_C]$ is connected for all $C \in \mathcal{C}$. We can assume that G does not contain isolated vertices.

For every $v \in V(T)$, $X \in \mathrm{part}_q(T_v)$ and $R' \in \mathcal{R}_{d,q}^{\overline{v}}$ and some $Y \in \mathrm{part}_q(T_{\overline{v}})$ with $Y \equiv_{d,q}^{\overline{v}} R'$ we define the weight of X when complemented with Y, $\mathrm{w}(X, Y)$, as

$$\mathrm{w}(X, Y) = \bigoplus_{j \in [q]} \bigoplus_{w \in X_j} \mathrm{w}(w, a_j, |N(w) \cap (X_1 \cup Y_1)|, \dots, |N(w) \cap (X_q \cup Y_q)|).$$

We define $\mathrm{w}(X, [Y]) = \mathrm{w}(X, Y)$. It follows from the d-stability of w and the definition of the equivalence relation $\equiv_{d,q}^{\overline{v}}$ that this notion is well-defined.

We say that a tuple $(X, Y) \in \mathrm{part}_q(T_v) \times \mathrm{part}_q(T_{\overline{v}})$ is T_v-*valid* if for every vertex $w \in T_v$ we have that $check(w, j_w, |N(w) \cap (X_1 \cup Y_1)|, \dots, |N(w) \cap (X_q \cup Y_q)|) = \textsc{True}$, where j_w is such that $w \in X_{j_w}$, $|X| + |Y| = K$ and $G[X_C \cup Y_C]$ is connected for each $C \in \mathcal{C}$. We say that a tuple $(X, R') \in \mathrm{part}_q(T_v) \times \mathcal{R}_{d,q}^{\overline{v}}$ is T_v-*partially valid* if for any vertex $w \in T_v$ and for a representative $Y \equiv_{d,q}^{\overline{v}} R'$ we have $check(w, j_w, |N(w) \cap (X_1 \cup Y_1)|, \dots, |N(w) \cap (X_q \cup Y_q)|) = \textsc{True}$, where j_w is such that $w \in X_{j_w}$. Observe that all of these notions are well-defined by the definition of the equivalence relation $\equiv_{d,q}^v$.

We now explain the general idea of the algorithm. For every node $v \in V(T)$ and for every $R \in \mathcal{R}_{d,q}^v$, $R' \in \mathcal{R}_{d,q}^{\overline{v}}$, $K' \in \mathrm{part}_q(|T_v|)$, we will compute a set $M_v[R, R', K'] \subseteq \mathrm{part}_q(T_v)$ such that

- $|M_v[R, R', K']| \leq 2^{|\mathcal{C}|}\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$,
- for every $X \in M_v(R, R', K')$ we have that $X \equiv^v_{d,q} R$, $|X| = K'$ and that (X, R') is T_v-partially valid, and
- for every $X \in \mathrm{part}_q(T_v)$ with $X \equiv^v_{d,q} R$, $|X| = K'$ and every $Y \in \mathrm{part}_q(T_{\overline{v}})$ such that $Y \equiv^{\overline{v}}_{d,q} R'$ and (X, Y) is T_v-valid there is an $X' \in M_v[R, R', K']$ such that $\mathrm{w}(X', Y) \preceq \mathrm{w}(X, Y)$ and (X', Y) is T_v-valid.

Let r be the root of T. Observe that $T_{\overline{r}} = \emptyset$ and thus for every $X, X' \in \mathrm{part}_q(T_r)$, we have that $X \equiv^r_{d,q} X'$. We further have that $\mathcal{R}^r_{d,q}$ contains only one element, namely $[\emptyset^q]$. It follows that $M_r[[\emptyset^q], [\emptyset^q], K]$ has to contain an optimal solution of our problem, if and only if such a solution exists. In order to solve the problem we just need to search for an $X \in M_r[[\emptyset^q], [\emptyset^q], K]$ of minimum weight $\mathrm{w}(X, \emptyset^q)$ such that $(X, \emptyset^q)$ is T_r-valid. Notice that the size of $M_r[[\emptyset^q], [\emptyset^q], K]$ is at most $2^{|\mathcal{C}|}\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$, and that checking if $(X, \emptyset^q)$ is T_r-valid (knowing that $(X, [\emptyset^q])$ is T_r-partially valid) and computing $\mathrm{w}(X, \emptyset^q)$ can be done in $O(n^2)$ time. Therefore, given the set $M_r[[\emptyset^q], [\emptyset^q], K]$, finding an optimal solution there (or discovering that no solution exists) can be done in time $O(\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}n^2)$.

We will compute these sets $M_v[R, R', K']$ by the standard dynamic programming bottom-up approach. We start by computing them whenever v is a leaf of T. Then, T_v contains a single vertex w of G and for each equivalence class in $\mathcal{R}^v_{d,q}$ there is just one representative. Thus, for each $R \in \mathcal{R}^v_{d,q}$ and $R' \in \mathcal{R}^{\overline{v}}_{d,q}$ we let X be the unique representative of R and if $X \in \mathrm{part}_q(T_v)$ and (X, R') is T_v-partially valid then we set $M_v[R, R', |X|] = \{X\}$ and to $\emptyset$ if not. For every $K' \neq |X|$, we set $M_v[R, R', K'] = \emptyset$.

Now let v be a non-leaf node of T. Let a and b be the children of v, and assume we have already correctly computed the sets $M_a[R_a, R'_a, K_a]$ for every $R_a \in \mathcal{R}^a_{d,q}$, $R'_a \in \mathcal{R}^{\overline{a}}_{d,q}$ and $K_a \in \mathrm{part}_q(|T_a|)$, as well as $M_b[R_b, R'_b, K_b]$ for every $R_b \in \mathcal{R}^b_{d,q}$, $R'_b \in \mathcal{R}^{\overline{b}}_{d,q}$ and $K_b \in \mathrm{part}_q(|T_b|)$. For $R \in \mathcal{R}^v_{d,q}$, $R' \in \mathcal{R}^{\overline{v}}_{d,q}$ and $K' \in \mathrm{part}_q(|T_v|)$, we say that two tuples $(R_a, R'_a, K^a) \in \mathcal{R}^a_{d,q} \times \mathcal{R}^{\overline{a}}_{d,q} \times \mathrm{part}_q(|T_a|)$ and $(R_b, R'_b, K^b) \in \mathcal{R}^b_{d,q} \times \mathcal{R}^{\overline{b}}_{d,q} \times \mathrm{part}_q(|T_b|)$ are (R, R', K')-*compatible* if $K^a + K^b = K'$ and, for representatives $Y, Y', Y_a, Y'_a, Y_b, Y'_b$ of $R, R', R_a, R'_a, R_b, R'_b$ respectively, we have $Y_a \cup Y_b \equiv^v_{d,q} Y$, $Y_b \cup Y' \equiv^{\overline{a}}_{d,q} Y'_a$, $Y_a \cup Y' \equiv^{\overline{b}}_{d,q} Y'_b$. For some fixed $R \in \mathcal{R}^v_{d,q}$, $R' \in \mathcal{R}^{\overline{v}}_{d,q}$ and $K' \in \mathrm{part}_q(|T_v|)$ consider the set

$$\mathcal{M} = \bigcup_{\substack{(R_a, R'_a, K^a) \text{ and } (R_b, R'_b, K^b) \\ \text{are } (R, R', K')-\text{compatible}}} \left(\bigcup_{\substack{X_a \in M_a[R_a, R'_a, K^a] \\ X_b \in M_b[R_b, R'_b, K^b]}} (X_a \cup X_b) \right).$$

Here the union is indexed over all tuples (R_a, R'_a, K^a) and (R_b, R'_b, K^b) which are as above and (R, R', K')-compatible. We first show that this set has all the properties demanded from $M_v[R, R', K']$ except for the upper bound on the size, and then we show a way to reduce the size of the set without losing these other properties. Notice that the size of $\mathcal{M}$ could be as large as $n^q \cdot \mathrm{snec}_{d,q}(T)^4 \cdot \left(2^{|\mathcal{C}|}\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}\right)^2$ because we are assuming that $M_a[R_a, R'_a, K^a]$ and $M_b[R_b, R'_b, K^b]$ are of size at most $2^{|\mathcal{C}|}\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$ and there are at most $n^q \cdot \mathrm{snec}_{d,q}(T)^4$ pairs of tuples (R_a, R'_a, K^a) and (R_b, R'_b, K^b) that are (R, R', K')-compatible (there are at most n^q possible values for K^a and for each of them there is only one choice for K^b, and there are at most $\mathrm{snec}_{d,q}(T)$ possible values for each R_a, R'_a, R_b, R'_b).

Let (R, R', K') be as above and $X \in \mathrm{part}_q(T_v)$ be such that $X \equiv^v_{d,q} R$ and $|X| = K'$. Suppose there exists $Y \in \mathrm{part}_q(T_{\overline{v}})$ such that $Y \equiv^{\overline{v}}_{d,q} R'$ and (X, Y) is T_v-valid. Set $X^a = T_a \cap X$, $X^b = T_b \cap X$, $X'^a = X^b \cup Y$, $X'^b = X^a \cup Y$, $K^a = |X^a|$ and $K^b = |X^b|$. Clearly (X^a, X'^a) is T_a-valid, (X^b, X'^b) is T_b-valid and $K^a + K^b = K'$, hence $M_a[[X^a], [X'^a], K^a]$ and $M_b[[X^b], [X'^b], K^b]$ are not empty.

Let $\textsc{Conn}(Z^a, Z^b, Y, \mathcal{C})$ denote the statement "$(Z^a)_C \cup (Z^b)_C \cup Y_C$ is connected for each $C \in \mathcal{C}$". Observe that

$$\begin{aligned}
\mathrm{w}(X, Y) &= \mathrm{w}(X^a \cup X^b, Y) \\
&= \mathrm{w}(X^a, X'^a) \oplus \mathrm{w}(X^b, X'^b) \\
&\geq \min_{\substack{Z^a \in \mathrm{part}_q(T_a),\ Z^a \equiv^a_{d,q} X^a \\ Z^b \in \mathrm{part}_q(T_b),\ Z^b \equiv^b_{d,q} X^b \\ |Z^a| = K^a,\ |Z^b| = K^b \\ \textsc{Conn}(Z^a, Z^b, Y, \mathcal{C})}} \left(\mathrm{w}(Z^a, X'^a) \oplus \mathrm{w}(Z^b, X'^b)\right) \\
&= \min_{\substack{Z^a \in \mathrm{part}_q(T_a),\ Z^a \equiv^a_{d,q} X^a \\ |Z^a| = K^a}} \left(\mathrm{w}(Z^a, X'^a) \oplus \left(\min_{\substack{Z^b \in \mathrm{part}_q(T_b),\ Z^b \equiv^b_{d,q} X^b \\ |Z^b| = K^b \\ \textsc{Conn}(Z^a, Z^b, Y, \mathcal{C})}} \left(\mathrm{w}(Z^b, X'^b)\right) \right) \right) \\
&= \min_{\substack{Z^a \in \mathrm{part}_q(T_a),\ Z^a \equiv^a_{d,q} X^a \\ |Z^a| = K^a}} \left(\mathrm{w}(Z^a, X'^a) \oplus \left(\min_{\substack{Z^b \in M_b[[X^b], [X'^b], K^b] \\ \textsc{Conn}(Z^a, Z^b, Y, \mathcal{C})}} \left(\mathrm{w}(Z^b, X'^b)\right) \right) \right)
\end{aligned}$$

$$
\begin{aligned}
&= \min_{\substack{Z^a \in \mathrm{part}_q(T_a),\ Z^a \equiv^a_{d,q} X^a \\ |Z^a| = K^a \\ Z^b \in M_b[[X^b],[X'^b],K^b] \\ \textsc{Conn}(Z^a, Z^b, Y, \mathcal{C})}} \left(\mathrm{w}(Z^a, X'^a) \oplus \mathrm{w}(Z^b, X'^b)\right) \\
&= \min_{Z^b \in M_b[[X^b],[X'^b],K^b]} \left(\left(\min_{\substack{Z^a \in \mathrm{part}_q(T_a),\ Z^a \equiv^a_{d,q} X^a \\ |Z^a| = K^a \\ \textsc{Conn}(Z^a, Z^b, Y, \mathcal{C})}} \left(\mathrm{w}(Z^a, X'^a)\right) \right) \oplus \mathrm{w}(Z^b, X'^b) \right) \\
&= \min_{Z^b \in M_b[[X^b],[X'^b],K^b]} \left(\left(\min_{\substack{Z^a \in M_a[[X^a],[X'^a],K^a] \\ \textsc{Conn}(Z^a, Z^b, Y, \mathcal{C})}} \left(\mathrm{w}(Z^a, X'^a)\right) \right) \oplus \mathrm{w}(Z^b, X'^b) \right) \\
&= \min_{\substack{Z^a \in M_a[[X^a],[X'^a],K^a] \\ Z^b \in M_b[[X^b],[X'^b],K^b] \\ \textsc{Conn}(Z^a, Z^b, Y, \mathcal{C})}} \left(\mathrm{w}(Z^a, X'^a) \oplus \mathrm{w}(Z^b, X'^b)\right) \\
&= \min_{\substack{Z^a \in M_a[[X^a],[X'^a],K^a] \\ Z^b \in M_b[[X^b],[X'^b],K^b] \\ \textsc{Conn}(Z^a, Z^b, Y, \mathcal{C})}} \left(\mathrm{w}(Z^a, Z^b \cup Y) \oplus \mathrm{w}(Z^b, Z^a \cup Y)\right) \\
&= \min_{\substack{Z^a \in M_a[[X^a],[X'^a],K^a] \\ Z^b \in M_b[[X^b],[X'^b],K^b] \\ \textsc{Conn}(Z^a, Z^b, Y, \mathcal{C})}} \mathrm{w}(Z^a \cup Z^b, Y).
\end{aligned}
$$

It follows that there are $Z^a \in M_a[[X^a], [X'^a], K^a]$ and $Z^b \in M_b[[X^b], [X'^b], K^b]$ such that $\mathrm{w}(Z^a \cup Z^b, Y) \preceq \mathrm{w}(X^a \cup X^b, Y) = \mathrm{w}(X, Y)$. Further, $(Z^a \cup Z^b, Y)$ is T_v-valid. Indeed, the connectivity constraints are satisfied by the above and the other constraints follow from the definition of M_a and M_b.

Succinctly, for every $X \in \mathrm{part}_q(T_v)$ with $X \equiv^v_{d,q} R$ and $|X| = K'$, and every $Y \in \mathrm{part}_q(T_{\overline{v}})$ such that $Y \equiv^{\overline{v}}_{d,q} R'$ and (X, Y) is T_v-valid, there exists $Z \in \mathcal{M}$ such that $\mathrm{w}(Z, Y) \preceq \mathrm{w}(X, Y)$ and (Z, Y) is T_v-valid.

We are now going to show that we can find a subset of $\mathcal{M}$ which has cardinality at most $2^{|\mathcal{C}|}\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$ and keeps the property we have just shown. Observe that any subset of $\mathcal{M}$ can only contain partitions of T_v which are T_v-partially valid together with R' and who have the correct partition size. Thus, when choosing a subset of $\mathcal{M}$, we only have to focus on keeping the partial solutions which are part of an optimal solution.

Let $v \in V(T)$, $R' \in \mathcal{R}^{\overline{v}}_{d,q}$, $\mathcal{X} \subseteq \mathrm{part}_q(T_v)$ as well as $\mathcal{X}' \subseteq \mathcal{X}$. We say that $\mathcal{X}'$ does (v, R')-*represent* $\mathcal{X}$ if, for every $X \in \mathcal{X}$ and $Y \in \mathrm{part}_q(T_{\overline{v}})$ such that $Y \equiv^{\overline{v}}_{d,q} R'$ and $X_C \cup Y_C$ is connected for every $C \in \mathcal{C}$, there exists $X' \in \mathcal{X}'$ such that $X'_C \cup Y_C$ is connected for every $C \in \mathcal{C}$ and $\mathrm{w}(X', R') \preceq \mathrm{w}(X, R')$.

Claim 5.3. *Let $v \in V(T)$ and $R \in \mathcal{R}^v_{d,q}$, $R' \in \mathcal{R}^{\overline{v}}_{d,q}$. Let $\mathcal{X} \subseteq part_q(T_v)$ be such that $X \equiv^v_{d,q} R$ for every $X \in \mathcal{X}$. We can find a set (v, R')-representing $\mathcal{X}$ of cardinality at most $2^{|\mathcal{C}|}snec_{d,q}(T)^{2|\mathcal{C}|}$ in $O(|\mathcal{X}| \cdot snec_{d,q}(T)^{2|\mathcal{C}|} \cdot (n^2 + snec_{d,q}(T)^{\omega-1}))$ time, where ω is the matrix multiplication exponent ($\omega < 2.4$ [18]).*

Proof. Let $\mathcal{C}_\emptyset \subseteq \mathcal{C}$ be the set of all connectivity restraints C for which $\emptyset \equiv^{\overline{v}}_d R'_C$. For any $C \in \mathcal{C}_\emptyset$ and for any $Y \in \mathrm{part}_q(T_{\overline{v}})$ with $Y \equiv^{\overline{v}}_{d,q} R'$, notice that $Y_C \equiv^{\overline{v}}_d \emptyset$. Also, if $Y_C = \emptyset$ then Y can only be completed to a T_v-valid coloring with a partition $X \in \mathcal{X}$ when X_C is connected; on the other hand, if $Y_C \neq \emptyset$ then Y can only be completed to a T_v-valid coloring with a partition $X \in \mathcal{X}$ when X_C is empty, since otherwise $X_C \cup Y_C$ cannot be connected.

Given a map $\tau : \mathcal{C}_\emptyset \to \{0, 1\}$ we say that a partition $X \in \mathcal{X}$ is of *type* τ if $X_C = \emptyset$ for every $C \in \tau^{-1}(0)$ and X_C is connected for every $C \in \tau^{-1}(1)$. Observe that if $\mathcal{C}_\emptyset = \emptyset$ then τ can only be the empty map and then every X is of type τ. There are at most $2^{|\mathcal{C}_\emptyset|}$ possible maps from $\mathcal{C}_\emptyset$ to $\{0, 1\}$. For each map τ as above we will compute a set $\mathfrak{X}_\tau$ of size at most $\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$ that (v, R')-represents the set $\{X \in \mathcal{X} : X \text{ has type } \tau\}$. We claim that their union will yield a set as demanded. Indeed, for every $Y \equiv^{\overline{v}}_{d,q} R'$ we define a map τ_Y as above by mapping $C \in \mathcal{C}_\emptyset$ to 0 if Y_C is non-empty and to 1 otherwise. By the above, for any $X \in \mathcal{X}$ such that $X_C \cup Y_C$ is connected for every $C \in \mathcal{C}$, we have that X is of type τ_Y. This implies that there is $X' \in \mathfrak{X}_{\tau_Y}$ such that $X'_C \cup Y_C$ is connected for every $C \in \mathcal{C}$ and $\mathrm{w}(X', Y) \preceq \mathrm{w}(X, Y)$. Hence the union of all the $\mathfrak{X}_\tau$ is a (v, R')-representative of $\mathcal{X}$. Further, if the size of each $\mathfrak{X}_\tau$ is indeed at most $\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$, then their union has size at most $2^{|\mathcal{C}_\emptyset|}\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|} \leq 2^{|\mathcal{C}|}\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$.

It remains to show how to compute such a set $\mathfrak{X}_\tau$ for a fixed τ. Fix a map τ as above and consider the set $\mathcal{X}_\tau$ consisting of every $X \in \mathcal{X}$ which has type τ. Let $\overline{\mathcal{C}_\emptyset} = \mathcal{C} \setminus \mathcal{C}_\emptyset$. Now R'_C does not contain the empty set for any $C \in \overline{\mathcal{C}_\emptyset}$. Let $\mathcal{Y}$ be the set of all partitions $Y \equiv^{\overline{v}}_{d,q} R'$ such that $\tau_Y = \tau$.

If there is a partition $X \in \mathcal{X}_\tau$ such that there is a $C \in \overline{\mathcal{C}_\emptyset}$ and a connected component $A \subseteq X_C$ such that A is not adjacent to R'_C, then for any $Y \in \mathcal{Y}$ the set $X_C \cup Y_C$ is not connected. Hence, we remove all such partitions from $\mathcal{X}_\tau$. Analogously, if there is a partition $Y \in \mathcal{Y}$ which has a connected component A of Y_C such that A is not adjacent to R_C then $X_C \cup Y_C$ cannot be connected for any $X \in \mathcal{X}_\tau$. Hence, we remove these partitions from $\mathcal{Y}$.

If the size of $\mathcal{X}_\tau$ is at most $\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$, then we can set $\mathfrak{X}_\tau = \mathcal{X}_\tau$. Thus, we can assume from now on that $|\mathcal{X}_\tau| > \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$.

For every $S \subseteq T_{\overline{v}}$ let v_S be a vertex in S. Let $\mathcal{T}$ be the set of $\overline{\mathcal{C}_\emptyset}$-tuples whose entries are 2-tuples from $\mathcal{R}^{\overline{v}}_{d,q}$ (that is, $\mathcal{T}$ is the set of all elements $(R^1_C, R^2_C)_{C\in\overline{\mathcal{C}_\emptyset}}$ with $R^1, R^2 \in \mathcal{R}^{\overline{v}}_{d,q}$). Let us now define the following matrices over $\mathbb{F}_2$:

- J is the $(\mathcal{X}_\tau \times \mathcal{Y})$-matrix for which $J(X, Y) = 1$ if and only if $X_C \cup Y_C$ is connected for each $C \in \overline{\mathcal{C}_\emptyset}$.
- L is a $(\mathcal{P}(T_v) \times (\mathcal{R}^{\overline{v}}_d \times \mathcal{R}^{\overline{v}}_d))$-matrix such that $L(X, (R'_1, R'_2)) = 1$ if and only if no connected component of X is adjacent to both R'_1 and R'_2.
- $\overline{L}$ is a $((\mathcal{R}^{\overline{v}}_d \times \mathcal{R}^{\overline{v}}_d) \times \mathcal{P}(T_{\overline{v}}))$-matrix such that $\overline{L}((R'_1, R'_2), Y) = 1$ if and only if there is a connected cut (Y_1, Y_2) of Y such that $v_Y \in Y_1$, $Y_1 \equiv^{\overline{v}}_d R'_1$ and $Y_2 \equiv^{\overline{v}}_d R'_2$.
- L^* is an $(\mathcal{X}_\tau \times \mathcal{T})$-matrix such that $L^*(X, (R^1_C, R^2_C)_{C\in\mathcal{C}}) = 1$ if and only if $L(X_C, (R^1_C, R^2_C)) = 1$ for each $C \in \overline{\mathcal{C}_\emptyset}$.
- $\overline{L^*}$ is a $(\mathcal{T} \times \mathcal{Y})$-matrix such that $\overline{L^*}((R^1_C, R^2_C)_{C\in\mathcal{C}}, Y) = 1$ if and only if $\overline{L}((R^1_C, R^2_C), Y_C) = 1$ for every $C \in \overline{\mathcal{C}_\emptyset}$.

We will now prove that $L^*\overline{L^*} = J$. The operations that follow are all over $\mathbb{F}_2$. For all $X \in \mathcal{X}_\tau$ and all $Y \in \mathcal{Y}$, we have:

$$\begin{aligned}
L^*\overline{L^*}(X, Y) &= \sum_{(R^1_C, R^2_C)_{C\in\overline{\mathcal{C}_\emptyset}}\in\mathcal{T}} \mathbb{1}\left(\forall C \in \overline{\mathcal{C}_\emptyset},\ L\left(X_C, \left(R^1_C, R^2_C\right)\right) = 1 \wedge \overline{L}\left(\left(R^1_C, R^2_C\right), Y_C\right) = 1\right) \\
&= |\{(R^1_C, R^2_C)_{C\in\overline{\mathcal{C}_\emptyset}} \in \mathcal{T} : L\left(X_C, \left(R^1_C, R^2_C\right)\right) = 1 \wedge \overline{L}\left(\left(R^1_C, R^2_C\right), Y_C\right) = 1 \\
&\qquad \text{for every } C \in \overline{\mathcal{C}_\emptyset}\}| \\
&= |\{(R^1_C, R^2_C)_{C\in\overline{\mathcal{C}_\emptyset}} \in \mathcal{T} : \forall C \in \overline{\mathcal{C}_\emptyset},\ \exists (Z^1_C, Z^2_C) \in ccut(X_C \cup Y_C), \\
&\qquad v_{Y_C} \in Z^1_C \wedge Z^1_C \cap Y_C \equiv^{\overline{v}}_d R^1_C \wedge Z^2_C \cap Y_C \equiv^{\overline{v}}_d R^2_C\}| \\
&= \prod_{C\in\overline{\mathcal{C}_\emptyset}} |\{(R^1, R^2) \in \mathcal{R}^{\overline{v}}_d \times \mathcal{R}^{\overline{v}}_d : \exists (Z^1, Z^2) \in ccut(X_C \cup Y_C), \\
&\qquad v_{Y_C} \in Z^1 \wedge Z^1 \cap Y_C \equiv^{\overline{v}}_d R^1 \wedge Z^2 \cap Y_C \equiv^{\overline{v}}_d R^2\}| \\
&= \prod_{C\in\overline{\mathcal{C}_\emptyset}} \left|\left\{(Z^1, Z^2) \in ccut(X_C \cup Y_C) : v_{Y_C} \in Z^1\right\}\right| \\
&= \prod_{C\in\overline{\mathcal{C}_\emptyset}} 2^{|cc(X_C\cup Y_C)|-1}.
\end{aligned}$$

The first two equalities follow right from the definition, the third one follows from [15, Claim 4.3.1]. The fourth one holds since we can choose the pair of equivalence classes (R^1_C, R^2_C) independently for each $C \in \overline{\mathcal{C}_\emptyset}$. The fifth equality follows from [15, Claim 4.3.2], and the last one follows from the fact that any connected component of $X_C \cup Y_C$ which does not contain v_{Y_C} can arbitrarily be assigned to both Z^1 and Z^2. Since we defined our matrices to be over $\mathbb{F}_2$, and the only case where every factor of the last product is odd is when $X_C \cup Y_C$ is connected for every $C \in \overline{\mathcal{C}_\emptyset}$, then the claim follows.

Observe that we can compute the matrix L^* in $O(|\mathcal{X}_\tau| \cdot \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|} \cdot n^2)$ time. Indeed, the number of entries of L^* is at most $|\mathcal{X}_\tau| \cdot \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$. Also, for an entry indexed by $(X, (R^1_C, R^2_C)_{C\in\mathcal{C}})$ and for every $C \in \overline{\mathcal{C}_\emptyset}$, we can first compute the connected components of X_C and then check the adjacencies. All these steps can be done in $O(n^2)$ time.

Now, we assign each row of L^* a weight. More precisely, the row which is indexed by $X \in \mathcal{X}_\tau$ obtains the weight $\mathrm{w}(X, R')$. These assignments can be done in $O(|\mathcal{X}_\tau|n^2)$ time. Let $\mathcal{B} \subseteq \mathcal{X}_\tau$ be such that the rows of L^* which are indexed by the entries in $\mathcal{B}$ are a basis of the row-space of L^*, $row(L^*)$, and the sum of the weights of these rows is minimum among all bases of $row(L^*)$ which consist of row vectors of L^*. Observe that the dimension of $row(L^*)$ and thus the size of $\mathcal{B}$ is at most the number of columns of L^*, which is bounded from above by $\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$. Observe as well that, by [18, Lemma 3.15], $\mathcal{B}$ can be computed in time $O(|\mathcal{X}_\tau| \cdot \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|(\omega-1)})$ (where $\omega < 2.373$ is the matrix multiplication coefficient).

We claim that $\mathcal{B}$ is a set which has the properties demanded from $\mathfrak{X}_\tau$. The size is already correct, then it remains to show that for every $X \in \mathcal{X}_\tau$, $Y \in \mathcal{Y}$, with $J(X, Y) = 1$, there is an $X' \in \mathcal{B}$ such that $J(X', Y) = 1$ and $\mathrm{w}(X', Y) \preceq \mathrm{w}(X, Y)$. Indeed, since $\mathcal{B}$ corresponds to a basis over $\mathbb{F}_2$, there is a unique $\mathcal{B}' \subseteq \mathcal{B}$ such that for any $\rho \in \mathcal{T}$ we have

$$L^*(X, \rho) = \sum_{B\in\mathcal{B}'} L^*(B, \rho).$$

We can use this to rewrite the row of J corresponding to X:

$$\begin{aligned}
J(X, Y) &= \sum_{\rho\in\mathcal{T}} L^*(X, \rho)\overline{L^*}(\rho, Y) \\
&= \sum_{B\in\mathcal{B}'} \sum_{\rho\in\mathcal{T}} L^*(B, \rho)\overline{L^*}(\rho, Y) \\
&= \sum_{B\in\mathcal{B}'} J(B, Y).
\end{aligned}$$

Hence, if $J(X, Y) = 1$ then there is an odd number of elements $B \in \mathcal{B}'$ such that $J(B, Y) = 1$. Let B be any of them which has maximum weight. If $\mathrm{w}(X, Y) \prec \mathrm{w}(B, Y)$ then $\mathcal{B} \setminus \{B\} \cup \{X\}$ corresponds to a basis of the row-space of L^* which has smaller weight than the one corresponding to $\mathcal{B}$. Indeed, it is clear that the weight would be smaller by the assumption on its weight and it does indeed correspond to a basis, since $L^*(B, \rho) = L^*(X, \rho) - \sum_{B' \in \mathcal{B} \setminus \{B\}} L^*(B', \rho)$ for every $\rho \in \mathcal{T}$.

It follows from Claim 5.3 that there is a set of size at most $2^{|\mathcal{C}|}\mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$ which (v, R')-represents $\mathcal{M}$. We can thus set $M_v[R, R', K']$ to be any such set. It remains to determine the runtime of the entire algorithm.

We can precalculate a representative for each equivalence class using Remark 5.1. For a fixed node $v \in V(T)$ the matrix M_v has at most $\mathrm{snec}_{d,q}(T)^2 \cdot n^{q-1}$ entries. We compute those entries using dynamic programming in a bottom-up fashion. When v is a leaf of T, each of those sets can be computed in constant time. When v is not a leaf, we first compute $\mathcal{M}$ in $O\left(n^q \cdot \mathrm{snec}_{d,q}(T)^4 \cdot \left(2^{|\mathcal{C}|} \cdot \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}\right)^2 \cdot n^2\right)$ time by using the pre-calculated sets over a and b (for each of the $O(n^q \cdot \mathrm{snec}_{d,q}(T)^4)$ pairs of compatible tuples, we iterate over all possible X_a, X_b, which are at most $\left(2^{|\mathcal{C}|} \cdot \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}\right)^2$ possibilities, and compute $X_a \cup X_b$ in $O(n^2)$ time), and then we apply Claim 5.3 in $O(n^q \cdot \mathrm{snec}_{d,q}(T)^4 \cdot (2^{|\mathcal{C}|} \cdot \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|})^2 \cdot \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|} \cdot (n^2 + \mathrm{snec}_{d,q}(T)^{\omega-1}))$ time, since in this case we have $|\mathcal{X}| = |\mathcal{M}| = n^q \cdot \mathrm{snec}_{d,q}(T)^4 \cdot (2^{|\mathcal{C}|} \cdot \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|})^2$. It follows that every entry of M_v can be computed in time $O\left(n^q \cdot \mathrm{snec}_{d,q}(T)^{6|\mathcal{C}|+4} \cdot \left(n^2 + \mathrm{snec}_{d,q}(T)^{\omega-1}\right)\right)$. Then we only need to multiply by the possible number of nodes $v \in V(T)$, which is $O(n)$, and by the number of possible entries of M_v, which is $O(\mathrm{snec}_{d,q}(T)^2 \cdot n^{q-1})$. We conclude by adding the complexity of the analysis at the root. Therefore, we obtain the claimed total runtime. □

In this proof, we only considered the case when we want one fixed size for the sizes of the partition sets. However, this is enough to handle *any* restrictions on the sizes of the partition sets. Indeed, let G be a graph with n vertices together with a check function *check* and a weight function w and some connectivity restraints $\mathcal{C} \subseteq \mathcal{P}([q])$. If we want to find an optimal q-partition of $V(G)$ which satisfies the constraints given by *check* and $\mathcal{C}$ for which the size of the partition classes is contained in some $\mathcal{K} \subseteq \mathrm{part}_q(n)$ then we can determine an optimal partition for each $K \in \mathcal{K}$ with the algorithm above. Since $|\mathcal{K}| \leq n^{q-1}$, we do not have to apply the algorithm more than a polynomial number of times. We can then look for an optimal solution among the $|\mathcal{K}|$ solutions we have determined.

In order to handle a set $\mathcal{C}^-$ of negative connectivity constraints, first notice that asking for a set S to be disconnected is equivalent to asking that S can be partitioned into two non-empty subsets S_1 and S_2 such that there are no edges between vertices in S_1 and vertices in S_2. Hence, we can do the following: replace the set of colors Colors with the set $\textsc{Colors} \times \left(\bigtimes_{C \in \mathcal{C}^-} \{0, 1\}\right)$. This can be seen as keeping the old colors but adding a flag for every negative connectivity constraint. We refer to the flag associated to some $C \in \mathcal{C}^-$ as the C-flag. The check and weight functions maintain their functionality while ignoring the newly introduced flags. However, if a vertex v receives color a (in the first entry) and this color is contained in some $C \in \mathcal{C}^-$, the check function now also makes sure that v has no neighbor with a color $a' \in C$ in the first entry and with a different C-flag. If a vertex v receives a color a (in the first entry) that is not contained in some $C \in \mathcal{C}^-$, its C-flag has no importance whatsoever. We now ask further that for every $C \in \mathcal{C}^-$ there is at least one vertex whose color is in C and whose C-flag is 0 and that there is at least one vertex whose color is in C and whose C-flag is 1. This last constraint is a size constraint on the color classes. The new number of colors is $q2^{|\mathcal{C}^-|}$, which is not larger than $q2^{2^q}$.

Since $\mathrm{snec}_{d,q}(T) \leq n^{qdw}$, where w is the mim-width of the given decomposition (see Remark 5.1), we can say that if we do not have negative connectivity constraints, then the time complexity of the algorithm is $O\left(n^{2q+wdq(6|\mathcal{C}|+\omega+5)}\right)$, which can also be written as $n^{O(wdq(|\mathcal{C}|+1))}$. However, in the case where we have a set $\mathcal{C}^-$ of negative connectivity constraints and a set $\mathcal{C}^+$ of positive ones, then the complexity is $O\left(n^{2q2^{|\mathcal{C}^-|}+wdq2^{|\mathcal{C}^-|}(6|\mathcal{C}^+|+\omega+5)}\right)$, which can also be written as $n^{O(wdq(|\mathcal{C}^+|+1)2^{\mathcal{C}^-})}$. Hence, our main result (Theorem 1.2) follows.

5.1. An extension to distance versions

In the setup we have just seen, the check and weight functions obtain as input only information about the colors of the vertices in the neighborhood of v. Similarly to the approach in [20], we can extend this to larger distances.

Let t be a fixed positive integer. Let $r_1, \ldots, r_t$ be positive integers. It is possible to modify the algorithm above to handle check and weight functions which are d-stable for t distances $r_1, \ldots, r_t$. We will only sketch the outlines of how to do this. We know from [31] that for any graph G, positive integer r and binary decomposition tree (T, δ) of G, the mim-width of (T, δ) can at most double when it is applied to G^r. We define $\equiv_d^{v,r}$ and $\equiv_{d,q}^{v,r}$ as before, where now r specifies the graph to which we refer, G^r. Analogously, define $\mathcal{R}_d^{v,r}$ and $\mathcal{R}_{d,q}^{v,r}$, as well as all the above for $\overline{v}$. We also define T_v^r-partial validity for a tuple $(X, R'^1, \ldots, R'^t)$ by only considering representatives $Y \subseteq \mathrm{part}_q(T_{\overline{v}})$ for which $Y \equiv_{d,q}^{\overline{v},r_i} R'^i$ for every $i \in [t]$. For every $v \in V(T)$, $R^i \in \mathcal{R}_{d,q}^{v,r_i}$, $R'^i \in \mathcal{R}_{d,q}^{\overline{v},r_i}$, with $i \in [t]$, and $K' \in \mathrm{part}_q(|T_v|)$ we compute $M_v[R^1, R'^1, \ldots, R^t, R'^t, K'] \subseteq \mathrm{part}_q(T_v)$ with the conditions as before and additionally for every X in this set we demand that $X \equiv_{d,q}^{v,r_i} R^i$ for every $i \in [t]$. The size constraint for M_v will remain the same. The rest of the algorithm will not be changed fundamentally. Observe in particular that Claim 5.3 can be used unchanged, since we would only have *more* restrictions to put on Y. When computing $\mathcal{M}$, we

have to consider more possibilities of combinations, but after the reduction step, the size will be the same as in the 1-distance version, only the computation time changes.

Let us explain the new time complexity. We first obtain the graphs $G^{r_1}, \ldots, G^{r_t}$ and the corresponding representatives. Each of the matrices M_v have $\mathrm{snec}_{d,q}(T)^{2t} n^q$ entries. The number of elements in each of them is still at most $2^{|\mathcal{C}|} \cdot \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|}$. The base case can still be done in constant time (since we are assuming that t is a constant). When v is not a leaf, we compute $\mathcal{M}$ and reduce it with Claim 5.3 in $O(n^q \cdot \mathrm{snec}_{d,q}(T)^{4t} \cdot (2^{|\mathcal{C}|} \cdot \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|})^2 \cdot (n^2 + \mathrm{snec}_{d,q}(T)^{2|\mathcal{C}|} \cdot (n^2 + \mathrm{snec}_{d,q}(T)^{\omega-1})))$ time. The overall time complexity is then $O\left(n^{2q} \cdot \mathrm{snec}_{d,q}(T)^{6|\mathcal{C}|+6t} \cdot \left(n^2 + \mathrm{snec}_{d,q}(T)^{\omega-1}\right)\right)$. Using the inequality on $\mathrm{snec}_{d,q}(T)$, we obtain $n^{O(wdq(|\mathcal{C}|+t))}$. Observe that we can handle negative connectivity constraints in the same way as in Theorem 1.2. The following corollary then follows.

Corollary 5.4. *Let d, q and t be positive integers. For a locally checkable problem with q colors, a check function and a weight function that are d-stable for t distances, size constraints $\mathcal{K}$, positive connectivity constraints $\mathcal{C}^+$ and negative connectivity constraints $\mathcal{C}^-$, there exists an algorithm that solves the problem on graphs with a given binary decomposition tree in time $n^{O(wdq(|\mathcal{C}^+|+t)2^{|\mathcal{C}^-|})}$, where n is the number of vertices of the input graph and w is the mim-width of the associated decomposition.*

We would like to highlight that only the number of different distances employed affects the complexity, but not the specific values of the distances. This allows us to consider problems where, for example, each vertex of color 1 has to be at distance at most $\frac{n}{2}$ of a vertex of color 2.

6. Limits of the framework

We have shown in the previous section that a d-stable locally checkable problem with a bounded number of colors can be solved on a family of graphs of bounded mim-width in polynomial time. In this section we show that it is unlikely that this result can be extended to locally checkable problems with less restrictions. In particular, we analyze here the following three situations: unbounded number of colors, weight functions which are not d-stable for any constant d, and also check functions which are not d-stable for any constant d.

6.1. Non-constant number of colors

First we show that it is unlikely that we can handle an unbounded number of colors, even when both the check and weight functions are 1-stable. As noted in [20], obtaining the chromatic number of a graph G can be modeled as a locally checkable problem with 1-stable functions but a linear number of colors:

- $\textsc{Colors} = \{1, \ldots, |V(G)|\}$,
- $(\textsc{Weights}, \preceq, \oplus) = (\mathbb{N} \cup \{+\infty\}, \leq, \max)$,
- $\mathrm{w}(v, a, k_1, \ldots, k_{|V(G)|}) = a$ for all $v \in V(G)$, $a \in \textsc{Colors}$ and $k_1, \ldots, k_{|V(G)|} \in \mathbb{N}$,
- $\mathit{check}(v, a, k_1, \ldots, k_{|V(G)|}) = (k_a = 0)$ for every $v \in V(G)$, $a \in \textsc{Colors}$ and $k_1, \ldots, k_{|V(G)|} \in \mathbb{N}$.

However, this problem was shown to be NP-complete on circular-arc graphs [28], a graph class of mim-width at most 2 [11] (this fact was also noted in [21]). Consequently, if we were able to find a method which could find optimal solutions of 1-stable locally checkable problems with unbounded number of colors in graphs of mim-width 2 in polynomial time, then we could solve any NP-complete problem in polynomial time.

6.2. Non-d-stable color-counting weight function

In this section we consider the Max-Cut problem. Given a graph G, a *cut* is a partition of the vertices into two disjoint sets S and T, and its size is defined as the number of edges of the induced bipartite graph $G[S, T]$. The Max-Cut problem asks for the maximum size of a cut in a graph. This problem was shown to be NP-hard on interval graphs [4], which all have mim-width 1 and for which a decomposition of mim-width 1 can be computed in linear time [10]. We can model Max-Cut as a locally checkable problem with two colors and color-counting functions. Indeed, let

- $\textsc{Colors} = \{\textsc{s}, \textsc{t}\}$,
- $(\textsc{Weights}, \preceq, \oplus) = (\mathbb{Q} \cup \{-\infty\}, \geq, +)$,
- $\mathrm{w}(v, \textsc{s}, k_\textsc{s}, k_\textsc{t}) = \frac{k_\textsc{t}}{2}$ and $\mathrm{w}(v, \textsc{t}, k_\textsc{s}, k_\textsc{t}) = \frac{k_\textsc{s}}{2}$ for all $v \in V(G)$, $k_\textsc{s}, k_\textsc{t} \in \mathbb{N}$, and
- $\mathit{check}(v, a, k_\textsc{s}, k_\textsc{t}) = \textsc{True}$ for all $v \in V(G)$, $a \in \{\textsc{s}, \textsc{t}\}$, $k_\textsc{s}, k_\textsc{t} \in \mathbb{N}$.

In other words, every coloring is proper, and for a given coloring, the sum of all the weights is exactly the number of edges whose endpoints are colored differently. Thus, a coloring of minimum weight (according to the order defined on the weight set) does indeed correspond to a maximum cut. Observe that its check function is 1-stable, but the weight function is not d-stable for any constant d. Thus, we cannot expect to eliminate this restriction on the weight function.

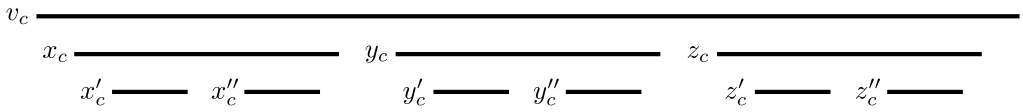

Fig. 1. Clause gadget for $c = (x, y, z)$.

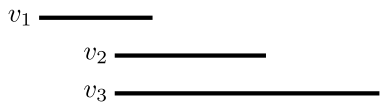

Fig. 2. Left gadget.

6.3. Non-d-stable color-counting check function

Finally, we show that for general color-counting check functions, we cannot expect to test existence of proper colorings unless P=NP. To this end, we define the following ad hoc locally checkable problem with constant number of colors and a color-counting check function. Let us set $\textsc{Colors} = \{1, \ldots, 7, \mathtt{T}, \mathtt{t}, \mathtt{F}, \mathtt{f}\}$. Given a vertex v, a color a and the number $k_{a'}$ of neighbors of v of color a', for every $a' \in \textsc{Colors}$, the check function is defined as follows:

$$
\begin{aligned}
&\mathit{check}(v, a, k_1, \ldots, k_7, k_{\mathtt{T}}, k_{\mathtt{t}}, k_{\mathtt{F}}, k_{\mathtt{f}}) = \\
&\quad (a = 1 \Leftrightarrow \deg(v) = 2) \\
&\wedge (a = 2 \Leftrightarrow \deg(v) > 2 \wedge k_1 \geq 1) \\
&\wedge (a = 3 \Leftrightarrow \deg(v) > 2 \wedge k_1 = 0 \wedge k_2 \geq 2) \\
&\wedge (a = 4 \Leftrightarrow \deg(v) > 2 \wedge k_1 = 0 \wedge k_2 = 1) \\
&\wedge (a = 5 \Leftrightarrow \deg(v) > 2 \wedge k_1 = 0 \wedge k_2 = 0 \wedge k_3 = k_4) \\
&\wedge (a = 6 \Leftrightarrow \deg(v) > 2 \wedge k_1 = 0 \wedge k_2 = 0 \wedge k_3 \neq k_4 \wedge k_5 > 1) \\
&\wedge (a = 7 \Leftrightarrow \deg(v) > 2 \wedge k_1 = 0 \wedge k_2 = 0 \wedge k_3 \neq k_4 \wedge k_5 = 1 \wedge \sum_{i \in \{7, \mathtt{T}, \mathtt{t}, \mathtt{F}, \mathtt{f}\}} k_i = 2) \\
&\wedge (a = 5 \Rightarrow k_{\mathtt{T}} = 1) \\
&\wedge (a = 6 \Rightarrow k_{\mathtt{T}} = k_{\mathtt{t}} \wedge k_{\mathtt{F}} = k_{\mathtt{f}}) \\
&\wedge (a = 7 \Rightarrow k_{\mathtt{t}} = k_{\mathtt{T}} = 1 \vee k_{\mathtt{f}} = k_{\mathtt{F}} = 1).
\end{aligned}
$$

Notice that it is justified to use the degree of the input vertex in the check function since the degree equals the sum of the sizes of all color classes in the neighborhood.

We claim that it is NP-hard to determine the existence of a proper coloring even when restricted to interval graphs (and thus in a graph class where the mim-width is bounded by 1). We show this by polynomially reducing from Positive 1-in-3 SAT, which is shown to be NP-complete in [38]. It is a 3-SAT variant in which all literals are positive and a truth assignment is satisfying if there is exactly one true literal in every clause.

Given an instance I (with at least one clause) of positive 1-in-3-SAT with clause set C and variable set X, we construct an interval graph G_I as follows. For each clause $c = (x, y, z)$ we create a copy of the *clause gadget* illustrated in Fig. 1. We call v_c a *clause vertex*, x_c, y_c and z_c *variable vertices*, and x'_c, x''_c, y'_c, y''_c, z'_c and z''_c *value vertices*. All the clause gadgets are placed next to each other in an arbitrary order such that they do not overlap. For every variable $x \in X$, do the following. Consider every two clauses c and c' that both contain x and whose gadgets are positioned in such a way that no other gadget corresponding to a clause containing x is positioned between the gadgets of c and c'. Without loss of generality, assume that the gadget corresponding to c is positioned to the left of the gadget of c'. We add an interval whose left endpoint is in the interval corresponding to x''_c and whose right endpoint is in the interval of $x'_{c'}$. The vertices corresponding to these intervals will be called *connector vertices*. Also, add the *left gadget* illustrated in Fig. 2 to the very left, not intersecting any of the intervals of the clause gadgets nor any of the connector vertices. Now, for each clause $c \in C$ we add two more intervals as follows. Add an interval whose left endpoint lies in the interval of v_2 but not in the interval of v_1, and whose right endpoint lies in the interval corresponding to v_c but to the left of any variable vertex in the clause gadget of c. The corresponding vertex of this interval is called the *opening vertex of* c. Add another interval whose left endpoint lies in the interval corresponding to v_3 but not in the interval of v_2, and whose right endpoint lies in the interval corresponding to v_c but to the right of every variable vertex in the clause gadget of c. We call the corresponding vertex the *closing vertex of* c. In Fig. 3, we show a simple example with three clauses in which only one variable appears twice (the right-most positioned variable in the middle clause and the middle variable in the right-most clause).

Notice that G_I can be constructed in polynomial time from a given instance I. We claim that G_I has a proper coloring if and only if I is satisfiable.

Fig. 3. Example of the interval representation of G_I, where I consists of three clauses $c_1 = (x_1, x_2, x_3)$, $c_2 = (x_4, x_5, x_6)$ and $c_3 = (x_7, x_6, x_8)$.

First, assume we have a proper coloring *col* of G_I. It is easy to see that v_1 is the only vertex colored with 1 (because it is the only vertex of degree 2) and the vertices v_2 and v_3 are the only ones colored with 2 (because they are the only neighbors of a vertex of color 1). Thus, all the opening vertices are colored with 3, and all the closing vertices are colored with 4, and no other vertices can get these colors. Now notice that every clause vertex is adjacent to the same number of opening vertices and closing vertices, but this property does not hold for any variable, value or connector vertex (the connector vertices are adjacent to the same opening and closing vertices as the value vertex in which its left endpoint is contained). Thus, all clause vertices are colored with 5 and no other vertices are. Since each connector vertex has at least two clause vertices in its neighborhood, these vertices receive color 6. All the variable and value vertices have exactly one neighbor of color 5, therefore, they are colored with 7, T, t, F or f. Then we have that each of the value vertices has exactly one neighbor which is not colored with a color from 1 to 6, while the variable vertices have two of them. Thus, variable vertices are colored with 7, and value vertices are colored with T, t, F or f. For every variable vertex, the set of colors assigned to their two neighboring value vertices is either {T, t} or {F, f}. Now consider the value vertices in the neighborhood of some fixed connector vertex w. We have shown that all of them, except for those whose intervals contain the endpoints of w, appear in pairs colored either with T and t, or with F and f. Since the number of value vertices in $N(w)$ which receive color T must be the same as the number of value vertices in $N(w)$ with color t (and analogously for F and f), it follows that the two value vertices containing the endpoints of w must either receive the colors T and t or the colors F and f. This implies that for every variable $x \in X$ either for every clause c which contains x we have that $col(x'_c), col(x''_c) \in \{\mathtt{T}, \mathtt{t}\}$, or for every clause c containing x we have that $col(x'_c), col(x''_c) \in \{\mathtt{F}, \mathtt{f}\}$. Finally, every vertex of color 5 has exactly one neighbor of color T. This means that in the neighborhood of every clause vertex there is exactly one variable vertex whose adjacent value vertices are colored with {T, t}, and the remaining two variables vertices have their adjacent value vertices colored with {F, f}. We set a variable $x \in X$ to True if all value vertices adjacent to the variable vertices corresponding to a literal of x are colored with T or t and to False otherwise. We claim that this yields a satisfying assignment of I. Indeed, each variable gets a single value (either True or False, since the set of colors of corresponding value vertices is either {T, t} or {F, f}), and each clause has exactly a variable set to True.

Assume now that the given instance I of positive 1-in-3 SAT is satisfiable. Let f be a function from the set of variables to {True, False} that assigns values to the variables satisfying the instance. We will construct a proper coloring of the graph G_I. Assign color 1 to v_1, color 2 to v_2 and v_3, color 3 to every opening vertex, color 4 to every closing vertex, color 5 to every clause vertex, color 6 to every connector vertex, and color 7 to every variable vertex. For every value vertex x'_c, if $f(x) =$ True then assign x'_c the color T, otherwise assign x'_c the color F. Analogously, for every value vertex x''_c, if $f(x) =$ True assign x''_c color t, otherwise assign x''_c color f. It is easy to verify that this is a proper coloring.

7. Applications

In this section, we give some examples of problems that can be modeled as d-stable locally checkable problems with a bounded number of colors, some of them with size and connectivity constraints on the color classes. Their complexity was previously unknown on graphs of bounded mim-width and, as a consequence of the results in Section 5 (as well as in Section 4.2), all these problems are XP parameterized by the mim-width of a given binary decomposition tree of the input graph.

Independently, Bergougnoux, Dreier and Jaffke [14] give a DN logic formula for conflict-free coloring and b-coloring with a bounded number of colors. Their expression is shorter and more elegant than the one that can be obtained from Section 4.2. However, applying the algorithm of [14] to these formulas yields a slower runtime than the algorithm in Section 5. Although every problem that is expressible as a d-stable locally checkable problem can be expressed in DN-logic too, we believe that for some of these problems it is more natural to express them as a d-stable locally checkable problem.

7.1. [k]-Roman domination

This problem was first defined in [1] as a generalization of Roman domination and double Roman domination [9,23]. Let k be a positive integer. A [k]-*Roman dominating function* on a graph G is a function $f : V(G) \to \{0, \ldots, k+1\}$ which has the property that if $f(v) < k$ then $\sum_{u \in N_G[v]} f(u) \geq |AN_G^f(v)| + k$, where $AN_G^f(v) = \{u \in N_G(v) : f(u) \geq 1\}$ (this set is called the *active neighborhood of* v). The *weight* of a [k]-Roman dominating function f is $\sum_{v \in V(G)} f(v)$, and the minimum weight of a [k]-Roman dominating function on G is the [k]-*Roman domination number* of G. The [k]−Roman domination problem consists in computing the [k]-Roman domination number of a given graph.

In [8], [k]−Roman domination was shown to be solvable in linear time on graphs of bounded clique-width by modeling it as a locally checkable problem with a color-counting check function, as follows:

- $\textsc{Colors} = \{0, \ldots, k+1\}$,
- $(\textsc{Weights}, \preceq, \oplus) = (\mathbb{N} \cup \{+\infty\}, \leq, +)$,
- $\mathrm{w}(v, a, \ell_0, \ldots, \ell_{k+1}) = a$ for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_0, \ldots, \ell_{k+1} \in \mathbb{N}$,
- $\mathit{check}(v, a, \ell_0, \ldots, \ell_{k+1}) = \left(a + \sum_{j=0}^{k+1} j\ell_j \geq k + \sum_{j=1}^{k+1} \ell_j\right)$ for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_0, \ldots, \ell_{k+1} \in \mathbb{N}$.

Notice that w is 1-stable and *check* is k-stable, so this model also suffices our purposes. Furthermore, as mentioned in [20], by making slight changes in the model (that do not affect the k-stability of the functions or the number of colors by more than a constant factor) we can also express several other variants of Roman domination, such as perfect [26], independent [34], outer independent [2], total [36] and maximal [20]. For the signed [3] and signed total [40] Roman domination, notice that they are color-counting, therefore FPT parameterized by clique-width [8], however they are not d-stable for any d.

7.2. Dual domination

In [24], three recent variations of the domination problem were defined. Given a graph partitioned into the *positive* vertices (V^+) and the *negative* vertices (V^-), in each problem we want to find a subset of positive vertices $D \subseteq V^+$ with conditions as follow:

- Maximum Bounded Dual Domination (MBDD): for an integer k given as input, $|N(D) \cap V^-| \leq k$ and $|N[D] \cap V^+|$ is as large as possible.
- Maximum Dual Domination (MDD): D maximizes $|N[D] \cap V^+| - |N(D) \cap V^-|$.
- Minimum Negative Dual Domination (mNDD): D dominates V^+ (that is, $V^+ \subseteq N[D]$) and minimizes $|N(D) \cap V^-|$.

All three problems are NP-hard and there exist linear-time algorithms to solve MDD and mNDD on trees, as well as a $O(k^2|V(G)|)$-time algorithm to solve MBDD on trees [24].

We will show that all three problems can be modeled as 1-stable locally checkable problems. In these models, the set D which we are searching for is the color class $\textsc{d}$.

Maximum Bounded Dual Domination (MBDD):

- $\textsc{Colors} = \left\{\textsc{d}, \textsc{n}^+, \textsc{n}^-, \overline{\textsc{n}}\right\}$,
- $(\textsc{Weights}, \preceq, \oplus) = (\mathbb{N} \cup \{-\infty\}, \geq, +)$,
- for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}} \in \mathbb{N}$ we have:

$$\mathrm{w}(v, a, \ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}}) = \begin{cases} 1 & \text{if } a \in \left\{\textsc{d}, \textsc{n}^+\right\} \\ 0 & \text{otherwise,} \end{cases}$$

- for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}} \in \mathbb{N}$ we have:
  - $\mathit{check}(v, \textsc{d}, \ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}}) = (v \in V^+)$
  - $\mathit{check}(v, \textsc{n}^+, \ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}}) = (v \in V^+ \wedge \ell_{\textsc{d}} \geq 1)$
  - $\mathit{check}(v, \textsc{n}^-, \ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}}) = (v \in V^- \wedge \ell_{\textsc{d}} \geq 1)$
  - $\mathit{check}(v, \overline{\textsc{n}}, \ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}}) = (\ell_{\textsc{d}} = 0)$,
- Size constraint: the number of vertices of color $\textsc{n}^-$ is at most k.

Maximum Dual Domination (MDD):

- $\textsc{Colors} = \left\{\textsc{d}, \textsc{n}^+, \textsc{n}^-, \overline{\textsc{n}}\right\}$,
- $(\textsc{Weights}, \preceq, \oplus) = (\mathbb{Z} \cup \{-\infty\}, \geq, +)$,
- for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}} \in \mathbb{N}$ we have:

$$\mathrm{w}(v, a, \ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}}) = \begin{cases} 1 & \text{if } a \in \left\{\textsc{d}, \textsc{n}^+\right\} \\ -1 & \text{if } a = \textsc{n}^- \\ 0 & \text{otherwise,} \end{cases}$$

- for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}} \in \mathbb{N}$ we have:
  - $\mathit{check}(v, \textsc{d}, \ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}}) = (v \in V^+)$
  - $\mathit{check}(v, \textsc{n}^+, \ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}}) = (v \in V^+ \wedge \ell_{\textsc{d}} \geq 1)$
  - $\mathit{check}(v, \textsc{n}^-, \ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}}) = (v \in V^- \wedge \ell_{\textsc{d}} \geq 1)$
  - $\mathit{check}(v, \overline{\textsc{n}}, \ell_{\textsc{d}}, \ell_{\textsc{n}^+}, \ell_{\textsc{n}^-}, \ell_{\overline{\textsc{n}}}) = (\ell_{\textsc{d}} = 0)$.

Minimum Negative Dual Domination (mNDD):

- $\textsc{Colors} = \left\{\textsc{d}, \textsc{n}^+, \textsc{n}^-, \overline{\textsc{n}}\right\}$,
- $(\textsc{Weights}, \preceq, \oplus) = (\mathbb{N} \cup \{+\infty\}, \leq, +)$,

- for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_{\mathrm{D}}, \ell_{\mathrm{N}^+}, \ell_{\mathrm{N}^-}, \ell_{\overline{\mathrm{N}}} \in \mathbb{N}$ we have:

$$\mathrm{w}(v, a, \ell_{\mathrm{D}}, \ell_{\mathrm{N}^+}, \ell_{\mathrm{N}^-}, \ell_{\overline{\mathrm{N}}}) = \begin{cases} 1 & \text{if } a = \mathrm{N}^- \\ 0 & \text{otherwise,} \end{cases}$$

- for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_{\mathrm{D}}, \ell_{\mathrm{N}^+}, \ell_{\mathrm{N}^-}, \ell_{\overline{\mathrm{N}}} \in \mathbb{N}$ we have:
  - $\mathit{check}(v, \mathrm{D}, \ell_{\mathrm{D}}, \ell_{\mathrm{N}^+}, \ell_{\mathrm{N}^-}, \ell_{\overline{\mathrm{N}}}) = (v \in V^+)$
  - $\mathit{check}(v, \mathrm{N}^+, \ell_{\mathrm{D}}, \ell_{\mathrm{N}^+}, \ell_{\mathrm{N}^-}, \ell_{\overline{\mathrm{N}}}) = (v \in V^+ \wedge \ell_{\mathrm{D}} \geq 1)$
  - $\mathit{check}(v, \mathrm{N}^-, \ell_{\mathrm{D}}, \ell_{\mathrm{N}^+}, \ell_{\mathrm{N}^-}, \ell_{\overline{\mathrm{N}}}) = (v \in V^- \wedge \ell_{\mathrm{D}} \geq 1)$
  - $\mathit{check}(v, \overline{\mathrm{N}}, \ell_{\mathrm{D}}, \ell_{\mathrm{N}^+}, \ell_{\mathrm{N}^-}, \ell_{\overline{\mathrm{N}}}) = (v \in V^- \wedge \ell_{\mathrm{D}} = 0)$.

The models above imply that these three problems are XP when parameterized by the mim-width of a given binary decomposition tree of the input graph. Moreover, by the results in [8], they are also FPT when parameterized by clique-width.

7.3. Happy colorings

In a vertex coloring c of a graph G, a vertex v is called *happy* if all its neighbors receive the color $c(v)$. Given a partial coloring of a graph G using k colors (that is, a coloring p of domain P for some $P \subseteq V(G)$), we want to determine the maximum number of happy vertices we can obtain in an extension of this coloring (that is, in a coloring c of the whole graph G, where c restricted to P is the coloring p). This problem is known as k-Maximum Happy Vertices (k-MHV) and was first defined in [41]. In the same paper it is proven that, for every $k \geq 3$, this problem is NP-complete. Algorithmic results for this problem so far include: an FPT algorithm parameterized by treewidth and neighborhood diversity [5], and an FPT algorithm parameterized by clique-width [17]. Also, a natural generalization to weighted graphs is defined in [5] (that is, there is a function $w : V(G) \to \mathbb{N}$ and we want to maximize the sum of $w(v)$ over all the happy vertices v). We can model weighted k-MHV as a 1-stable locally checkable problem, as follows:

- $\textsc{Colors} = \{1, \ldots, k\}$,
- for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_1, \ldots, \ell_k \in \mathbb{N}$:

$$\mathit{check}(v, a, \ell_1, \ldots, \ell_k) = \begin{cases} \textsc{False} & \text{if } v \in P \text{ and } a \neq p(v) \\ \textsc{True} & \text{otherwise,} \end{cases}$$

  where P is the set of precolored vertices and $p(v)$ is the color received by v in the precoloring,
- $(\textsc{Weights}, \preceq, \oplus) = (\mathbb{N} \cup \{-\infty\}, \geq, +)$,
- for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_1, \ldots, \ell_k \in \mathbb{N}$ we have:

$$\mathrm{w}(v, a, \ell_1, \ldots, \ell_k) = \begin{cases} w(v) & \text{if } \ell_i = 0 \text{ for all } i \in [k] - \{a\} \\ 0 & \text{otherwise.} \end{cases}$$

  Notice that this function is 1-stable.

The Maximum Happy Set problem (MaxHS), defined in [7], asks for a subset S of k vertices that maximizes the number of vertices $v \in S$ such that $N[v] \subseteq S$. This problem is NP-hard, as well as FPT when parameterized by either tree-width, by neighborhood diversity, or by cluster deletion number [7]. It is also FPT when parameterized by modular-width or by clique-width [35], and polynomial-time solvable on interval graphs [6]. With the following model we show that MaxHS is 1-stable with 2 colors and a size constraint.

- $\textsc{Colors} = \{\mathrm{s}, \overline{\mathrm{s}}\}$,
- for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_{\mathrm{s}}, \ell_{\overline{\mathrm{s}}} \in \mathbb{N}$: $\mathit{check}(v, a, \ell_{\mathrm{s}}, \ell_{\overline{\mathrm{s}}}) = \textsc{True}$,
- $(\textsc{Weights}, \preceq, \oplus) = (\mathbb{N} \cup \{-\infty\}, \geq, +)$,
- for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_1, \ldots, \ell_k \in \mathbb{N}$ we have:

$$\mathrm{w}(v, a, \ell_{\mathrm{s}}, \ell_{\overline{\mathrm{s}}}) = \begin{cases} 1 & \text{if } a = \mathrm{s} \text{ and } \ell_{\overline{\mathrm{s}}} = 0 \\ 0 & \text{otherwise.} \end{cases}$$

- The size of the color class s is k.

7.4. Locally bounded coloring

Suppose we are given a graph G, two positive integers p and k, a partition of the vertex set $V(G)$ into p subsets $V_1, \ldots, V_p$, and pk integral bounds $n_{i,j}$, with $i \in [p]$ and $j \in [k]$, such that $\sum_{j=1}^{k} n_{i,j} = |V_i|$ for each $i \in [p]$. The Locally Bounded k-Coloring problem, defined in [13], consists in deciding if there exists a coloring of G using k colors such that no two adjacent vertices get the same color and that, for each $i \in [p]$ and for each $j \in [k]$, the number of vertices having

color j in V_i is $n_{i,j}$. For fixed values of p and k, this problem is NP-complete [13] and polynomial-time solvable on graphs of bounded tree-width and on cographs [12].

We propose the following model for Locally Bounded k-Coloring, with fixed p and k, as a 1-stable locally checkable problem using pk colors and using size constraints.

- $\textsc{Colors} = [p] \times [k]$,
- $(\textsc{Weights}, \preceq, \oplus) = (\{0, 1\}, \leq, \max)$,
- $\mathrm{w}(v, (i, j), \ell_{(1,1)}, \ldots, \ell_{(p,k)}) = 0$ for all $v \in V(G)$, $(i, j) \in \textsc{Colors}$ and all $\ell_{(i',j')} \in \mathbb{N}$ with $(i', j') \in \textsc{Colors}$,
- $\mathit{check}(v, (i, j), \ell_{(1,1)}, \ldots, \ell_{(p,k)}) = \left(v \in V_i \wedge \sum_{i' \in [p]} \ell_{(i',j)} = 0\right)$ for all $v \in V(G)$, $a \in \textsc{Colors}$ and all $\ell_{(i',j')} \in \mathbb{N}$ with $(i', j') \in \textsc{Colors}$,
- Size constraints: for each pair $(i, j) \in \textsc{Colors}$, the number of vertices of color (i, j) is $n_{i,j}$.

7.5. Conflict-free k-coloring

Conflict-free coloring problems are a group of problems in all of which we look for a coloring of a graph such that in the neighborhood of every vertex there is a color which appears exactly once. The first such problem was defined in [27]. The following variations have been studied as well (for example in [16]). A *CFON k-coloring* of a graph G is a coloring $c : V(G) \to [k]$ such that for every vertex $v \in V(G)$ there is at least one vertex $u \in N(v)$ such that no other vertex in $N(v)$ has color $c(u)$. A *CFCN k-coloring* of a graph G is a coloring $c : V(G) \to [k]$ such that for every vertex $v \in V(G)$ there is at least one vertex $u \in N[v]$ such that no other vertex in $N[v]$ has color $c(u)$. A *CFON* k-coloring* of a graph G is a coloring $c : V(G) \to \{0, \ldots, k\}$ such that for every vertex $v \in V(G)$ there exists at least one vertex $u \in N(v)$ with $c(u) \neq 0$ and such that no other vertex in $N(v)$ has color $c(u)$. A *CFCN* k-coloring* of a graph G is a coloring $c : V(G) \to \{0, \ldots, k\}$ such that for every vertex $v \in V(G)$ there exists at least one vertex $u \in N[v]$ with $c(u) \neq 0$ and such that no other vertex in $N[v]$ has color $c(u)$. For each of these variants we want to determine the smallest k for which such a k-coloring exists. Each of these problems is NP-complete on general graphs.

We can model the CFON* k-coloring problem as a 2-stable locally checkable problem with $k + 1$ colors:

- $\textsc{Colors} = \{0, \ldots, k\}$,
- $(\textsc{Weights}, \preceq, \oplus) = (\{0, 1\}, \leq, \max)$,
- $\mathrm{w}(v, a, \ell_0, \ldots, \ell_k) = 0$ for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_0, \ldots, \ell_k \in \mathbb{N}$,
- $\mathit{check}(v, a, \ell_0, \ldots, \ell_k) = (\exists j \in [k],\ \ell_j = 1)$ for all $v \in V(G)$, $a \in \textsc{Colors}$ and $\ell_0, \ldots, \ell_k \in \mathbb{N}$.

For all the other variants of this problem we can easily modify the check function to model them correctly while preserving the 2-stability.

It was shown in [16] that the problem of finding a CFCN*- and a CFON*-coloring using the minimum number of colors is fixed-parameter tractable with respect to the combined parameters clique-width and number of colors k. Our result shows that the problems are solvable in polynomial time when mim-width and number of colors are bounded. In [16] they further show that any interval graph has a CFON* 3-coloring and ask for a polynomial-time algorithm determining the smallest number of colors needed for a CFON* coloring of an interval graph. Our result gives such an algorithm.

7.6. b-coloring with fixed number of colors

Given a graph G, a *b-coloring* of G is a coloring of $V(G)$ such that no pair of neighbors receive the same color and also every color class contains a vertex that has neighbors in all the other color classes. In [30], Irving and Manlove define the b-chromatic number of G, as the maximum k such that G admits a b-coloring with k colors, and prove that the problem of determining such number is NP-complete for general graphs. The b-coloring problem asks, for a given graph G and an integer k, if G admits a b-coloring with k colors. It is shown in [33] that this problem is NP-complete even for bipartite graphs. However, it has been shown in [32] that the b-coloring problem can be solved in polynomial time on graphs of bounded clique-width, and in [19] that for a fixed number of colors it can be solved in polynomial time on graphs of bounded proper thinness. Here we show that the b-coloring problem with a fixed number of colors k can be modeled with the following 1-stable locally checkable problem with $2k$ colors:

- $\textsc{Colors} = \{-k, \ldots, -1, 1, \ldots, k\}$,
- $(\textsc{Weights}, \preceq, \oplus) = (\{0, 1\}, \leq, \max)$,
- $\mathrm{w}(v, a, \ell_{-k}, \ldots, \ell_k) = 0$ for all $v \in V(G)$, $a \in \textsc{Colors}$ and all $\ell_b \in \mathbb{N}$ with $b \in \textsc{Colors}$,
- for every vertex $v \in V(G)$, every $a \in [k]$ and every $\ell_b \in \mathbb{N}$ with $b \in \textsc{Colors}$, we have that $\mathit{check}(v, a, \ell_{-k}, \ldots, \ell_k) = (\ell_a + \ell_{-a} = 0)$ and $\mathit{check}(v, -a, \ell_{-k}, \ldots, \ell_k) = (\ell_a + \ell_{-a} = 0 \wedge \forall i \in [k] \setminus \{a\},\ \ell_i + \ell_{-i} \geq 1)$,
- For each $a \in [k]$, the size of the color class $-a$ is at least 1.

It is easy to see that a proper coloring satisfying the size constraint corresponds to a b-coloring, where the vertices colored with negative integers represent the "b-vertices" (that is, the vertices that have at least one neighbor of each other color).

Data availability

No data was used for the research described in the article.

Acknowledgments

Carolina L. Gonzalez is partially supported by a CONICET doctoral fellowship, Argentina, CONICET, Argentina PIP 11220200100084CO, UBACyT, Argentina 20020170100495BA and ANPCyT, Argentina PICT-2021-I-A-00755. We would like to thank the anonymous reviewers for their valuable comments that helped improve our manuscript.

References

[1] H. Abdollahzadeh Ahangar, M. Álvarez, M. Chellali, S. Sheikholeslami, J. Valenzuela-Tripodoro, Triple roman domination in graphs, Appl. Math. Comput. 391 (2021) 125444, http://dx.doi.org/10.1016/j.amc.2020.125444, URL: https://www.sciencedirect.com/science/article/pii/S0096300320304057.

[2] H. Abdollahzadeh Ahangar, M. Chellali, S. Sheikholeslami, Outer independent double roman domination, Appl. Math. Comput. 364 (2020) 124617, http://dx.doi.org/10.1016/j.amc.2019.124617, URL: http://www.sciencedirect.com/science/article/pii/S0096300319306095.

[3] H. Abdollahzadeh Ahangar, M.A. Henning, C. Löwenstein, Y. Zhao, V. Samodivkin, Signed roman domination in graphs, J. Comb. Optim. 27 (2) (2014) 241–255, http://dx.doi.org/10.1007/s10878-012-9500-0.

[4] R. Adhikary, K. Bose, S. Mukherjee, B. Roy, Complexity of maximum cut on interval graphs, in: K. Buchin, E. Colin de Verdière (Eds.), 37th International Symposium on Computational Geometry, SoCG 2021, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 189, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021, pp. 7:1–7:11, http://dx.doi.org/10.4230/LIPIcs.SoCG.2021.7, URL: https://drops.dagstuhl.de/opus/volltexte/2021/13806.

[5] A. Agrawal, N. Aravind, S. Kalyanasundaram, A.S. Kare, J. Lauri, N. Misra, I.V. Reddy, Parameterized complexity of happy coloring problems, Theoret. Comput. Sci. 835 (2020) 58–81, http://dx.doi.org/10.1016/j.tcs.2020.06.002, URL: https://www.sciencedirect.com/science/article/pii/S0304397520303364.

[6] Y. Asahiro, H. Eto, T. Hanaka, G. Lin, E. Miyano, I. Terabaru, Complexity and approximability of the happy set problem, Theoret. Comput. Sci. 866 (2021) 123–144, http://dx.doi.org/10.1016/j.tcs.2021.03.023, URL: https://www.sciencedirect.com/science/article/pii/S0304397521001699.

[7] Y. Asahiro, H. Eto, T. Hanaka, G. Lin, E. Miyano, I. Terabaru, Parameterized algorithms for the happy set problem, Discrete Appl. Math. 304 (2021) 32–44, http://dx.doi.org/10.1016/j.dam.2021.07.005, URL: https://www.sciencedirect.com/science/article/pii/S0166218X21002626.

[8] N. Baghirova, C.L. Gonzalez, B. Ries, D. Schindl, Locally checkable problems parameterized by clique-width, in: S.W. Bae, H. Park (Eds.), 33rd International Symposium on Algorithms and Computation, ISAAC 2022, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 248, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2022, pp. 31:1–31:20, http://dx.doi.org/10.4230/LIPIcs.ISAAC.2022.31, URL: https://drops.dagstuhl.de/opus/volltexte/2022/17316.

[9] R.A. Beeler, T.W. Haynes, S.T. Hedetniemi, Double roman domination, Discrete Appl. Math. 211 (2016) 23–29, http://dx.doi.org/10.1016/j.dam.2016.03.017, URL: http://www.sciencedirect.com/science/article/pii/S0166218X1630155X.

[10] R. Belmonte, M. Vatshelle, Graph classes with structured neighborhoods and algorithmic applications, Theoret. Comput. Sci. 511 (2013) 54–65, http://dx.doi.org/10.1016/j.tcs.2013.01.011, URL: https://www.sciencedirect.com/science/article/pii/S0304397513000613. Exact and Parameterized Computation.

[11] R. Belmonte, M. Vatshelle, Graph classes with structured neighborhoods and algorithmic applications, Theoret. Comput. Sci. 511 (2013) 54–65, http://dx.doi.org/10.1016/j.tcs.2013.01.011, URL: https://www.sciencedirect.com/science/article/pii/S0304397513000613. Exact and Parameterized Computation.

[12] C. Bentz, Weighted and locally bounded list-colorings in split graphs, cographs, and partial k-trees, Theoret. Comput. Sci. 782 (2019) 11–29, http://dx.doi.org/10.1016/j.tcs.2019.02.029, URL: https://www.sciencedirect.com/science/article/pii/S0304397519301355.

[13] C. Bentz, C. Picouleau, Locally bounded k-colorings of trees, RAIRO-Oper. Res. 43 (1) (2009) 27–33, http://dx.doi.org/10.1051/ro/2009003.

[14] B. Bergougnoux, J. Dreier, L. Jaffke, A logic-based algorithmic meta-theorem for mim-width, in: N. Bansal, V. Nagarajan (Eds.), Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2023, pp. 3282–3304, http://dx.doi.org/10.1137/1.9781611977554.ch125, URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611977554.ch125.

[15] B. Bergougnoux, M.M. Kanté, More applications of the d-neighbor equivalence: acyclicity and connectivity constraints, SIAM J. Discrete Math. 35 (3) (2021) 1881–1926, http://dx.doi.org/10.1137/20M1350571.

[16] S. Bhyravarapu, T.A. Hartmann, S. Kalyanasundaram, I. Vinod Reddy, Conflict-free coloring: Graphs of bounded clique width and intersection graphs, in: P. Flocchini, L. Moura (Eds.), Combinatorial Algorithms, Springer International Publishing, Cham, 2021, pp. 92–106.

[17] I. Bliznets, D. Sagunov, Maximizing happiness in graphs of bounded clique-width, in: Y. Kohayakawa, F.K. Miyazawa (Eds.), LATIN 2020: Theoretical Informatics, Springer International Publishing, Cham, 2020, pp. 91–103, http://dx.doi.org/10.1007/978-3-030-61792-9_8.

[18] H.L. Bodlaender, M. Cygan, S. Kratsch, J. Nederlof, Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth, Inform. and Comput. 243 (2015) 86–111, http://dx.doi.org/10.1016/j.ic.2014.12.008, URL: https://www.sciencedirect.com/science/article/pii/S0890540114001606. 40th International Colloquium on Automata, Languages and Programming (ICALP 2013).

[19] F. Bonomo, D. de Estrada, On the thinness and proper thinness of a graph, Discrete Appl. Math. 261 (2019) 78–92, http://dx.doi.org/10.1016/j.dam.2018.03.072, URL: https://www.sciencedirect.com/science/article/pii/S0166218X1830180X. GO X Meeting, Rigi Kaltbad (CH), July 10–14, 2016.

[20] F. Bonomo-Braberman, C.L. Gonzalez, A new approach on locally checkable problems, Discrete Appl. Math. 314 (2022) 53–80, http://dx.doi.org/10.1016/j.dam.2022.01.019, URL: https://www.sciencedirect.com/science/article/pii/S0166218X22000348.

[21] N. Brettell, J. Horsfield, A. Munaro, G. Paesani, D. Paulusma, Bounding the mim-width of hereditary graph classes, J. Graph Theory 99 (1) (2022) 117–151, http://dx.doi.org/10.1002/jgt.22730, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jgt.22730.

[22] B.-M. Bui-Xuan, J.A. Telle, M. Vatshelle, Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems, Theoret. Comput. Sci. 511 (2013) 66–76, http://dx.doi.org/10.1016/j.tcs.2013.01.009, URL: http://www.sciencedirect.com/science/article/pii/S0304397513000595.

[23] E.J. Cockayne, P.A. Dreyer, S.M. Hedetniemi, S.T. Hedetniemi, Roman domination in graphs, Discrete Math. 278 (1) (2004) 11–22, http://dx.doi.org/10.1016/j.disc.2003.06.004, URL: http://www.sciencedirect.com/science/article/pii/S0012365X03004473.

[24] G. Cordasco, L. Gargano, A.A. Rescigno, Dual domination problems in graphs, J. Comput. System Sci. 128 (2022) 18–34, http://dx.doi.org/10.1016/j.jcss.2022.03.003, URL: https://www.sciencedirect.com/science/article/pii/S0022000022000277.

[25] B. Courcelle, The monadic second-order logic of graphs. I. Recognizable sets of finite graphs, Inform. and Comput. 85 (1) (1990) 12–75, http://dx.doi.org/10.1016/0890-5401(90)90043-H, URL: https://www.sciencedirect.com/science/article/pii/089054019090043H.
[26] A.T. Egunjobi, T.W. Haynes, Perfect double roman domination of trees, Discrete Appl. Math. (2020) http://dx.doi.org/10.1016/j.dam.2020.03.021, URL: http://www.sciencedirect.com/science/article/pii/S0166218X20301141.
[27] G. Even, Z. Lotker, D. Ron, S. Smorodinsky, Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks, in: The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings, 2002, pp. 691–700, http://dx.doi.org/10.1109/SFCS.2002.1181994.
[28] M.R. Garey, D.S. Johnson, G.L. Miller, C.H. Papadimitriou, The complexity of coloring circular arcs and chords, SIAM J. Algebr. Discrete Methods 1 (2) (1980) 216–227, http://dx.doi.org/10.1137/0601025.
[29] M.C. Golumbic, U. Rotics, On the clique—Width of perfect graph classes, in: P. Widmayer, G. Neyer, S. Eidenbenz (Eds.), Graph-Theoretic Concepts in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 135–147, http://dx.doi.org/10.1007/3-540-46784-X_14.
[30] R.W. Irving, D.F. Manlove, The b-chromatic number of a graph, Discrete Appl. Math. 91 (1) (1999) 127–141, http://dx.doi.org/10.1016/S0166-218X(98)00146-2, URL: http://www.sciencedirect.com/science/article/pii/S0166218X98001462.
[31] L. Jaffke, O. Kwon, T.J.F.S. mme, J.A. Telle, Generalized distance domination problems and their complexity on graphs of bounded mim-width, in: C. Paul, M. Pilipczuk (Eds.), 13th International Symposium on Parameterized and Exact Computation, IPEC, 2018, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 115, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2019, pp. 6:1–6:14, http://dx.doi.org/10.4230/LIPIcs.IPEC.2018.6, URL: http://drops.dagstuhl.de/opus/volltexte/2019/10207.
[32] L. Jaffke, P.T. Lima, D. Lokshtanov, B-coloring parameterized by clique-width, in: M. Bläser, B. Monmege (Eds.), 38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 187, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021, pp. 43:1–43:15, http://dx.doi.org/10.4230/LIPIcs.STACS.2021.43, URL: https://drops.dagstuhl.de/opus/volltexte/2021/13688.
[33] J. Kratochvíl, Z. Tuza, M. Voigt, On the b-chromatic number of graphs, in: G. Goos, J. Hartmanis, J. van Leeuwen, L. Kučera (Eds.), Graph-Theoretic Concepts in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 310–320, http://dx.doi.org/10.1007/3-540-36379-3_27.
[34] H. Maimani, M. Momeni, S. Nazari Moghaddam, F. Rahimi Mahid, S. Sheikholeslami, Independent double roman domination in graphs, Bull. Iranian Math. Soc. 46 (2020) 543–555, http://dx.doi.org/10.1007/s41980-019-00274-8.
[35] Y. Mizutani, B.D. Sullivan, Parameterized complexity of maximum happy set and densest k-subgraph, in: H. Dell, J. Nederlof (Eds.), 17th International Symposium on Parameterized and Exact Computation, IPEC, 2022, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 249, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2022, pp. 23:1–23:18, http://dx.doi.org/10.4230/LIPIcs.IPEC.2022.23, URL: https://drops.dagstuhl.de/opus/volltexte/2022/17379.
[36] Z. Shao, J. Amjadi, S.M. Sheikholeslami, M. Valinavaz, On the total double roman domination, IEEE Access 7 (2019) 52035–52041, http://dx.doi.org/10.1109/ACCESS.2019.2911659.
[37] J.A. Telle, Vertex Partitioning Problems: Characterization, Complexity and Algorithms on Partial *k*-Trees (Ph.D. thesis), University of Oregon, 1994.
[38] S. Tippenhauer, Freie Universität Berlin, 2016.
[39] M. Vatshelle, New width parameters of graphs (Ph.D. thesis), University of Bergen, 2012.
[40] L. Volkmann, On the signed total Roman domination and domatic numbers of graphs, Discrete Appl. Math. 214 (2016) 179–186, http://dx.doi.org/10.1016/j.dam.2016.06.006, URL: https://www.sciencedirect.com/science/article/pii/S0166218X16302712.
[41] P. Zhang, A. Li, Algorithmic aspects of homophyly of networks, Theoret. Comput. Sci. 593 (2015) 117–131, http://dx.doi.org/10.1016/j.tcs.2015.06.003, URL: https://www.sciencedirect.com/science/article/pii/S0304397515005058.