Better Polynomial Algorithms on Graphs of Bounded Rank-width

Robert Ganian and Petr Hliněný *

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
ganian@mail.muni.cz, hlineny@fi.muni.cz

Abstract. Although there exist many polynomial algorithms for NP-hard problems running on a bounded clique-width expression of the input graph, there exists only little comparable work on such algorithms for rank-width. We believe that one reason for this is the somewhat obscure and hard-to-grasp nature of rank-decompositions. Nevertheless, strong arguments for using the rank-width parameter have been given by recent formalisms independently developed by Courcelle and Kanté, by the authors, and by Bui-Xuan et al. This article focuses on designing formally clean and understandable "pseudopolynomial" (XP) algorithms solving "hard" problems (non-FPT) on graphs of bounded rank-width. Those include computing the chromatic number and polynomial or testing the Hamiltonicity of a graph and are extendable to many other problems.

Keywords: Rank-width, rank-decomposition, chromatic number, chromatic polynomial, Hamiltonian path.

1 Introduction

We postpone all formal definitions till the next section. Rank-width, introduced by Oum and Seymour [20], is a relatively new graph complexity measure which is quickly receiving attention over the past few years. Compared to the (perhaps better known) clique-width measure, rank-width has two major advantages: First, an optimal rank-decomposition can be efficiently constructed if the rank-width is bounded [17]. Second, a rank-decomposition (actually, a suitable modification of it, see Section 3) allows for design of formally cleaner [11], and sometimes much faster parameterized algorithms [2, 12] than previously known ones running on a clique-width expression of the given graph.

The core of the new approach lies in an alternative characterization of rank-decompositions using bilinear graph products [4], or equivalently labeling parse trees [10, 11], or R_k-joins [2]. These new approaches have led to interesting new FPT algorithms [2, 12] which run much faster with respect to the rank-width parameter than previously known algorithms, as in [5].

* The authors have been supported by the the research grants GAČR 201/08/0308 and 201/09/J021, and P. Hliněný also by the research intent MSM0021622419 of the Czech Ministry of Education.

The aim of this paper is to extend the ideas of the mentioned algorithms to problems which likely do not have FPT algorithms with respect to rank-width, and hence their presented algorithms are "pseudopolynomial", a complexity class known as XP. We present some of the basic definitions in Section 2, and describe the parse-tree formalism for handling rank-decompositions of graphs in Section 3. The main new results are then presented and proved in Section 4.

We prove (Theorem 4.1) that the chromatic number of a graph of rank-width t can be determined in time $O(n^{h(t)})$ where $h(t) = O(2^{t(t+1)/2})$. This algorithm significantly improves over a previous algorithm of Kobler and Rotics [19] which runs in time $O\big(n^{4^k}\big)$ on graphs of clique-width k. When comparing these two algorithms, the readers should keep in mind that our parameter t is the rank-width of the input graph, and the clique-width k can reach up to $2^{t/2-1}$ by [3]. We, moreover, straightforwardly extend our algorithm (Theorem 4.8) to compute the chromatic polynomial, again improving runtime over previous [1]. Finally, we show (Theorem 4.9) how to decide Hamiltonian path in a graph of bounded rank-width.

2 Definitions and Basics

We only consider finite undirected simple graphs without loops. We will start by briefly introducing a few needed concepts and then define rank-decompositions and rank-width, while in Section 3 we continue by defining the concepts of t-labeled graphs and their parse trees. Many of the definitions in the latter section are taken or adapted from our [11].

The reader should be aware of the notion of *fixed-parameter tractable* [6] algorithms (FPT algorithms in short), which are the algorithms running in time $O(n^c \cdot 2^{f(k)})$ for a constant c, a parameter k (rank-width in our case) and any (computable) f. Some NP-hard problems such as deciding whether a graph is q-colourable do have FPT algorithms when parameterized by clique-width, see e.g. [5]. On the other hand, [9] have recently proved that various problems, such as the chromatic number or hamiltonicity, likely can not be solved by FPT algorithms parameterized by clique-/ rank-width.

In such cases, authors usually look for algorithms which are "pseudopolynomial"—formally in class *XP* or *uniform XP* [6]—i.e. running in time $O(n^{f(k)})$ for the parameter k and a computable function f. Many examples using the clique-width parameter can be found in [1, 7, 13, 19, 21]. Our goal in this paper is to design and use a mathematically precise and sound formalism for solving problems on graphs of bounded rank-width in XP time. This extends the Myhill–Nerode type formalism which we have introduced in [11] for FPT algorithms on such graphs.

Branch-width. A set function $f : 2^M \to \mathbb{Z}$ is called *symmetric* if $f(X) = f(M \setminus X)$ for all $X \subseteq M$. A tree is *subcubic* if all its nodes have degree at most 3. For a symmetric function $f : 2^M \to \mathbb{Z}$ on a finite set M, the branch-width of f is defined as follows.

A *branch-decomposition* of f is a pair (T, μ) of a subcubic tree T and a bijective function $\mu : M \to \{t : t \text{ is a leaf of } T\}$. For an edge e of T, the connected components of $T \setminus e$ induce a bipartition (X, Y) of the set of leaves of T. The *width* of an edge e of a branch-decomposition (T, μ) is $f(\mu^{-1}(X))$. The *width* of (T, μ) is the maximum width over all edges of T. The *branch-width* of f is the minimum of the width of all branch-decompositions of f. (If $|M| \leq 1$, then we define the branch-width of f as $f(\emptyset)$.)

A natural application of this definition is the branch-width of a graph, introduced by Robertson and Seymour along with better known tree-width. In that case we use $M = E(G)$, and f the connectivity function of G. There is, however, another interesting application of the aforementioned general notions, in which we consider the vertex set $V(G) = M$ of a graph G as the ground set.

Rank-width ([20]). For a graph G, let $\boldsymbol{A}_G[U, W]$ be the bipartite adjacency matrix of a bipartition (U, W) of the vertex set $V(G)$ defined over the two-element field GF(2) as follows: the entry $a_{u,w}$, $u \in U$ and $w \in W$, of $\boldsymbol{A}_G[U, W]$ is 1 if and only if uw is an edge of G. The *cut-rank* function $\rho_G(U) = \rho_G(W)$ then equals the rank of $\boldsymbol{A}_G[U, W]$ over GF(2). A *rank-decomposition* and *rank-width* of a graph G is the branch-decomposition and branch-width of the cut-rank function ρ_G of G on $M = V(G)$, respectively.

Theorem 2.1 ([17]). *For every fixed t there is an $O(n^3)$-time FPT algorithm that, for a given n-vertex graph G, either finds a rank-decomposition of G of width at most t, or confirms that the rank-width of G is more than t.*

A few rank-width examples. Any complete graph of more than one vertex has clearly rank-width 1 since any of its bipartite adjacency matrices consists of all 1s. It is similar with complete bipartite graphs if we split the decomposition along the parts. We illustrate the situation with graph cycles: while C_3 and C_4 have rank-width 1, C_5 and all longer cycles have rank-width equal 2. A rank-decomposition of, say, the cycle C_5 is shown in Fig. 1. Conversely, every subcubic tree with at least 4 leaves has an edge separating at least 2 leaves on each side, and every corresponding bipartition of C_5 gives a matrix of rank 2.

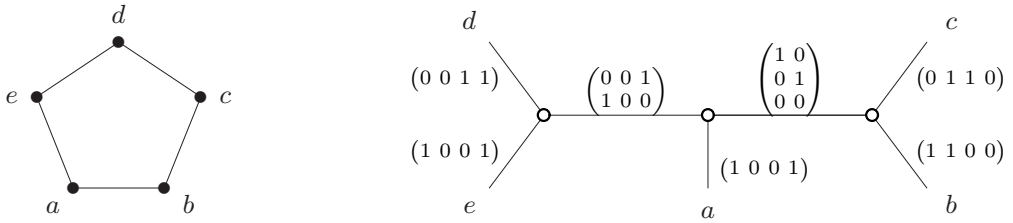

Fig. 1. A rank-decomposition of the graph cycle C_5.

Rank-width is closely tied to another width parameter called *clique-width* [7]. A graph has bounded rank-width if and only if it has bounded clique-width.

However, there is no equivalent of Theorem 2.1 for clique-width [8], and the value of clique-width can be up to exponentially larger than rank-width [3], both making the rank-width a more attractive parameter for designing algorithms. On the other hand, it appears really difficult to design dynamic-programming algorithms running on a given rank-decomposition of a graph.

3 Rank-width Parse Trees and Regularity

In a search for a "more suitable form" of a rank-decomposition, Courcelle and Kanté [4] defined the bilinear products of multiple-coloured graphs, and proposed algebraic expressions over these operators as an equivalent description of a rank-decomposition (cf. Theorem 3.1). Here we introduce (following [10, 11]) the same idea in terms of labeling join and parse trees which we propose as more convenient for the results in the next sections. One should note that an analogous idea also underlies the H-join decompositions of Bui-Xuan, Telle and Vatshelle [2].

A *t-labeling* of a graph is a mapping $lab : V(G) \to 2^{L_t}$ where $L_t = \{1, 2, \ldots, t\}$ is the set of *labels* (this notion is exactly equivalent to multiple-coloured graphs of [4]). Having a graph G with an (implicitly) associated t-labeling lab, we refer to the pair (G, lab) as to a *t-labeled graph* and use notation $\bar{G}$. Notice that each vertex of a t-labeled graph may have zero, one or more labels. We will often view (cf. [4] again) a t-labeling of G equivalently as a mapping $V(G) \to \mathrm{GF}(2)^t$ to the *binary vector space* of dimension t, where $\mathrm{GF}(2)$ is the two-element finite field.

Labeling join ([11]). Considering t-labeled graphs $\bar{G}_1 = (G_1, lab^1)$ and $\bar{G}_2 = (G_2, lab^2)$, a *t-labeling join* $\bar{G}_1 \otimes \bar{G}_2$ is defined on the disjoint union of G_1 and G_2 by adding all edges (u, v) such that $|lab^1(u) \cap lab^2(v)|$ is odd, where $u \in V(G_1), v \in V(G_2)$. The resulting graph is unlabeled.

A *t-relabeling* is a mapping $f : L_t \to 2^{L_t}$. In linear algebra terms, a t-relabeling f is in a natural one-to-one correspondence with a *linear transformation* $f : \mathrm{GF}(2)^t \to \mathrm{GF}(2)^t$, i.e. a $t \times t$ binary matrix $\boldsymbol{A}_f$. For a t-labeled graph $\bar{G} = (G, lab)$ we define $f(\bar{G})$ as the same graph with a vertex t-labeling $lab' = f \circ lab$. Here $f \circ lab$ stands for the linear transformation f applied to the labeling lab, or equivalently $lab' = lab \cdot \boldsymbol{A}_f$ as matrix multiplication. Informally, f is applied separately to each label in $lab(v)$ and the outcomes are summed up "modulo 2"; e.g. for $lab(v) = \{1, 2\}$ and $f(1) = \{1, 3, 4\}$, $f(2) = \{1, 2, 3\}$, we get $f \circ lab(v) = \{2, 4\} = \{1, 3, 4\} \triangle \{1, 2, 3\}$.

Let $\odot$ be a nullary operator creating a single new graph vertex of label $\{1\}$. For t-relabelings $f_1, f_2, g : L_t \to 2^{L_t}$, let $\otimes[g \,|\, f_1, f_2]$ be a binary operator—called *t-labeling composition* (as bilinear product of [4])—over pairs of t-labeled graphs $\bar{G}_1 = (G_1, lab^1)$ and $\bar{G}_2 = (G_2, lab^2)$ defined as follows:

$$\bar{G}_1 \otimes[g \,|\, f_1, f_2]\ \bar{G}_2 \ = \ \bar{H} \ = \ \big(\bar{G}_1 \otimes g(\bar{G}_2),\, lab\big)$$

where a new labeling is $lab(v) = f_i \circ lab^i(v)$ for $v \in V(G_i)$, $i = 1, 2$.

A *t-labeling parse tree* T, see also [10, Definition 6.11], is a finite rooted ordered subcubic tree (with the root degree at most 2) such that

– all leaves of T contain the $\odot$ symbol, and
– each internal node of T contains one of the t-labeling composition symbols.

A parse tree T then *generates* (parses) the graph G which is obtained by successive leaves-to-root applications of the operators in the nodes of T. See Fig. 2, 3.

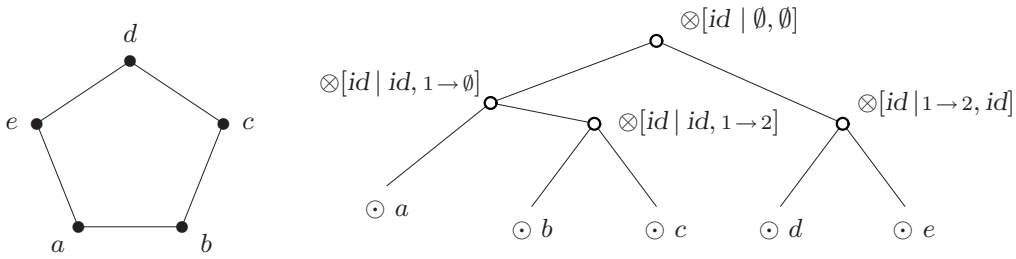

Fig. 2. An example of a labeling parse tree which generates a 2-labeled cycle C_5, with symbolic relabelings at the nodes (*id* denotes the relabeling preserving all labels, and $\emptyset$ is the relabeling "forgetting" all labels).

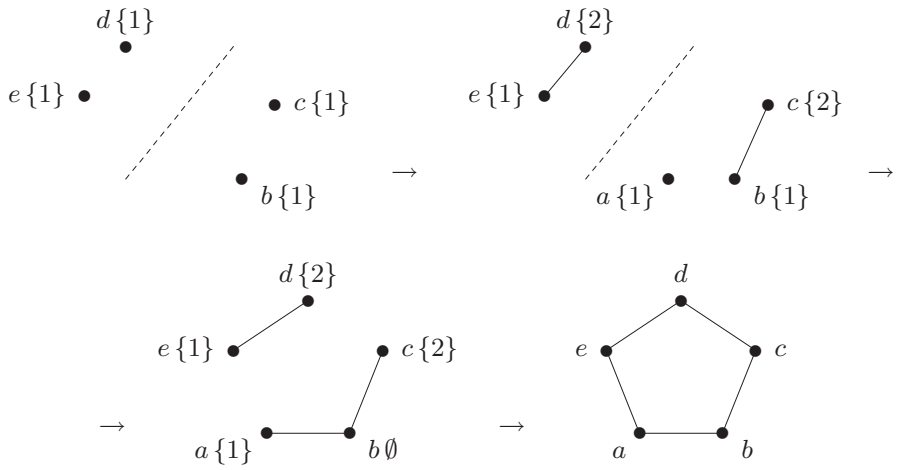

Fig. 3. "Bottom-up" generation of C_5 by the parse tree from Fig. 2.

Analogously to the work of Courcelle and Kanté we get a crucial statement:

Theorem 3.1 (Rank-width parsing theorem [4, 11]). *A graph G has rank-width at most t if and only if (some labeling of) G can be generated by a t-labeling parse tree. Furthermore, a width-t rank-decomposition of G can be transformed into a t-labeling parse tree on $\Theta(|V(G)|)$ nodes in time $O(t^2 \cdot |V(G)|^2)$.*

The tools we use are inspired by the Myhill–Nerode theorem in automata theory. We see this classical theorem as the starting point of formal understanding of dynamic algorithms: Such an algorithm typically collects "all relevant

information" about the studied problem on a local part, and then processes this information "through" the whole input. The task is to determine what the words "all relevant information" mean here.

The basic approach defines a so-called canonical equivalence, e.g. [16, 11], which is analogous to the "right congruence" known in theory of regular languages. A strongly enhanced formalism, called a PCE scheme, is given by the authors in [12]. The PCE scheme provides a very fine control over the runtime of FPT algorithms on graphs of bounded rank-width. On the other hand, in this paper we show a formally precise handling of problems for which we do not have FPT algorithms (wrt. rank-width), and which thus require a different, yet similar, formal approach.

4 Applications in Pseudopolynomial Algorithms

4.1 Computing the chromatic number

We illustrate our formalism on the graph chromatic number problem, for which we strongly improve the previous algorithm of [19] running on graphs of bounded clique-width. For the purposes of this section, it is useful to think about colouring not as a function from vertices to colours but rather as a vertex-partition of G. Formally, a *colour partition* of G is a partition $\mathcal{N}$ of $V(G)$ into pairwise disjoint nonempty(!) sets such that each $X \in \mathcal{N}$ is independent in G. The *chromatic number* of a graph G is the minimum $|\mathcal{N}|$ such that $\mathcal{N}$ is a colour partition of G.

Theorem 4.1. *Assume that an input graph G is given in the form of a t-labeling parse tree T. Then the chromatic number of G can be computed in time*

$$O\left(|V(G)|^{h(t)}\right) \text{ where } h(t) = O(2^{t(t+1)/2}).$$

We will also need a few preliminary technical results. Let, for $X \subseteq V(G)$, $\gamma(\bar{G}, X) = \{lab(u) : u \in X\}$. Notice that this set of labelings— vectors in $\mathrm{GF}(2)^t$ —generates a vector subspace $\langle\gamma(\bar{G}, X)\rangle$. Considering a t-labeled graph $\bar{G} = (G, lab)$ with a proper colour partition $\{C_1, C_2 \ldots C_j\}$, the core idea is that we only need to record the subspaces $\langle\gamma(\bar{G}, C_i)\rangle$ for $i = 1, \ldots, j$.

Lemma 4.2 (also [2, 12]). *Assume t-labeled graphs $\bar{G}$ and $\bar{H}$, and arbitrary sets $X \subseteq V(\bar{G})$, $Y \subseteq V(\bar{H})$. In the join graph $\bar{G} \otimes \bar{H}$, there is no edge between a vertex of X and a vertex of Y if and only if the subspace $\langle\gamma(\bar{G}, X)\rangle$ is orthogonal to the subspace $\langle\gamma(\bar{H}, Y)\rangle$ in $\mathrm{GF}(2)^t$.*

Proof. Let $\bar{G} = (G, lab)$ and $\bar{H} = (H, lab')$. We consider arbitrary $y \in Y$. Then (as a scalar product in $\mathrm{GF}(2)$) $lab(x) \cdot lab'(y) = 0$ for all $x \in X$; henceforth $\alpha \cdot lab'(y) = 0$ for all $\alpha \in \langle\gamma(\bar{G}, X)\rangle$ by means of elementary linear algebra. By a symmetrical argument, we get $\alpha \cdot \beta = 0$ for all $\beta \in \langle\gamma(\bar{H}, Y)\rangle$, which means these subspaces indeed are mutually orthogonal. Conversely, if $\langle\gamma(\bar{G}, X)\rangle \perp \langle\gamma(\bar{H}, Y)\rangle$, then there is obviously no edge between $x \in X$ and $y \in Y$ by the definition of $\otimes$ as $lab(x) \cdot lab'(y) = 0$. ∎

Corollary 4.3. *Assume t-labeled graphs $\bar{G}_1, \bar{G}_2$ and $\bar{H}$, and independent sets $X_i \subseteq V(\bar{G}_i)$, $i = 1, 2$ and $Y \subseteq V(\bar{H})$. If $\langle \gamma(\bar{G}_1, X_1) \rangle = \langle \gamma(\bar{G}_2, X_2) \rangle$, then $X_1 \cup Y$ is independent in $\bar{G}_1 \otimes \bar{H}$ if and only if $X_2 \cup Y$ is independent in $\bar{G}_2 \otimes \bar{H}$.*

Lemma 4.4 ([15], cf. [12, Proposition 6.1]). *The number $S(t)$ of subspaces of the binary vector space $\mathrm{GF}(2)^t$ satisfies $S(t) \leq 2^{t(t+1)/4} - 2$ for all $t \geq 12$.* ∎

Proof of Theorem 4.1. Given a graph G, we will write $G \models \nu(\mathcal{N})$ to say that a set family $\mathcal{N} \subseteq 2^{V(G)} \setminus \emptyset$ is a proper colour partition of G. Also, having two set families $\mathcal{N}, \mathcal{N}'$, we denote by $\mathcal{I}(\mathcal{N}', \mathcal{N})$ the set of all injective mappings $p : \mathcal{N}' \to \mathcal{N}$, and we write $\mathcal{N} \overset{p}{\leftarrow} \mathcal{N}'$ for $p \in \mathcal{I}(\mathcal{N}', \mathcal{N})$ to denote the family

$$\mathcal{N} \overset{p}{\leftarrow} \mathcal{N}' \;=\; \{X \cup p(X) : X \in \mathcal{N}'\} \cup \big(\mathcal{N} \setminus p(\mathcal{N}')\big)\,.$$

Informally, $\mathcal{N} \overset{p}{\leftarrow} \mathcal{N}'$ expands the colour partition $\mathcal{N}$ of G by merging some of its colour classes with those of $\mathcal{N}'$ as prescribed by p.

For any t-labeled graphs $\bar{G}_1, \bar{G}_2$ and any colour partitions $\mathcal{N}_i \subseteq 2^{V(G_i)} \setminus \emptyset$ where $i = 1, 2$, we define $(\bar{G}_1, \mathcal{N}_1) \approx_{\nu,t} (\bar{G}_2, \mathcal{N}_2)$ if and only if $|\mathcal{N}_1| = |\mathcal{N}_2| = q$ and the following holds true: For all t-labeled graphs $\bar{H}$ and all colour partitions $\mathcal{N} \subseteq 2^{V(H)}$, and for all $\mathcal{N}_0 \subseteq \mathcal{N}$ such that $|\mathcal{N}_0| \leq q$, it holds

$$\begin{aligned} &\exists p_1 \in \mathcal{I}(\mathcal{N}_0, \mathcal{N}_1) : \big(\bar{G}_1 \otimes \bar{H}\big) \models \nu\big((\mathcal{N}_1 \overset{p_1}{\leftarrow} \mathcal{N}_0) \cup (\mathcal{N} \setminus \mathcal{N}_0)\big) \\ \iff\; &\exists p_2 \in \mathcal{I}(\mathcal{N}_0, \mathcal{N}_2) : \big(\bar{G}_2 \otimes \bar{H}\big) \models \nu\big((\mathcal{N}_2 \overset{p_2}{\leftarrow} \mathcal{N}_0) \cup (\mathcal{N} \setminus \mathcal{N}_0)\big)\,. \end{aligned} \tag{4.5}$$

In informal words, $(\bar{G}_1, \mathcal{N}_1) \approx_{\nu,t} (\bar{G}_2, \mathcal{N}_2)$ means that there is no real difference between the q-colour-partitioned $(\bar{G}_1, \mathcal{N}_1)$ and $(\bar{G}_2, \mathcal{N}_2)$ with respect to the possibility of merging prescribed colour classes $\mathcal{N}_0$ of any joined graph $\bar{H}$ with some existing colour classes in $\mathcal{N}_1, \mathcal{N}_2$. Hence, $\approx_{\nu,t}$ captures all information necessary to decide which subcolourings of $\bar{G}_i$ extend to colourings of any larger $\bar{G}_i \otimes \bar{H}$.

Let $\Gamma(\bar{G}, \mathcal{N}) = \{\langle \gamma(\bar{G}, X) \rangle : X \in \mathcal{N}\}$ denote a multiset(!) of subspaces of $\mathrm{GF}(2)^t$. The crucial finding, inspired by the colouring algorithm in [19], reads:

(4.6) For any t-labeled graphs $\bar{G}_1, \bar{G}_2$ and any $\mathcal{N}_i \subseteq 2^{V(G_i)} \setminus \emptyset$, $i = 1, 2$ such that $\bar{G}_i \models \nu(\mathcal{N}_i)$, it holds $(\bar{G}_1, \mathcal{N}_1) \approx_{\nu,t} (\bar{G}_2, \mathcal{N}_2)$ if $\Gamma(\bar{G}_1, \mathcal{N}_1) = \Gamma(\bar{G}_2, \mathcal{N}_2)$.

We will use $\mathcal{N}_i^+$ to denote $(\mathcal{N}_i \overset{p_i}{\leftarrow} \mathcal{N}_0) \cup (\mathcal{N} \setminus \mathcal{N}_0)$, $i = 1, 2$ (cf. 4.5). To prove this claim, we assume $\Gamma(\bar{G}_1, \mathcal{N}_1) = \Gamma(\bar{G}_2, \mathcal{N}_2)$ and $\big(\bar{G}_1 \otimes \bar{H}\big) \models \nu(\mathcal{N}_1^+)$ for some fixed p_1. Since $\mathcal{N}_2^+$ is also a partition of the vertices of $\bar{G}_2 \otimes \bar{H}$, for claiming $\nu(\mathcal{N}_2^+)$ it suffices to verify that all $C \in \mathcal{N}_2^+$ are independent in the graph $\bar{G}_2 \otimes \bar{H}$. That is trivial if $C \in \mathcal{N}_2$ or $C \in (\mathcal{N} \setminus \mathcal{N}_0)$, since both $\bar{G}_2$ and $\bar{H}$ were properly coloured and in this case the whole C is present in one of the original graphs. For the rest, we consider any bijection $b : \mathcal{N}_1 \to \mathcal{N}_2$ preserving $\langle \gamma(\bar{G}_1, X) \rangle = \langle \gamma(\bar{G}_2, b(X)) \rangle$ for all $X \in \mathcal{N}_1$, and choose $p_2 = b \circ p_1$. Assume $C = Y \cup p_2(Y) \in \mathcal{N}_2^+$ where $Y \in \mathcal{N}_0$. Applying Corollary 4.3 with $X_1 = p_1(Y)$

and $X_2 = p_2(Y)$, we conclude that C is independent in $\bar{G}_2 \otimes \bar{H}$ if $p_1(Y) \cup Y$ is independent in $\bar{G}_1 \otimes \bar{H}$, and $p_1(Y) \cup Y \in \mathcal{N}_1^+$.

Considering the labeling parse tree T of G, and a node z of T, let $\bar{G}_z$ denote the t-labeled graph parsed by the subtree of T rooted at z. Thanks to previous Claim (4.6), a dynamic algorithm for computing the chromatic number of G has to remember only the set $M_T(z)$ of those multisets $\Gamma(\bar{G}_z, \mathcal{N})$ coming from proper colour partitions $\mathcal{N}$ of $V(G_z)$, at any particular node z of T. This information is trivial to construct at each leaf of T.

We now show how to obtain the set $M_T(z)$ from the sets $M_T(x)$ and $M_T(y)$ of the left son x and right son y of our z. We consider any proper colour partitions $\mathcal{N}_x$ and $\mathcal{N}_y$ of $\bar{G}_x$ and $\bar{G}_y$, respectively. Let $\mathcal{J}(\mathcal{N}_y, \mathcal{N}_x)$ denote the set of all partial injective mappings p from $\mathcal{N}_y$ into $\mathcal{N}_x$ (i.e. of injective mappings from any subset of $\mathcal{N}_y$ into $\mathcal{N}_x$), and let $\overset{p}{\rightleftharpoons}$ be a generalization of $\overset{p}{\leftarrow}$ such that

$$\mathcal{N}_x \overset{p}{\rightleftharpoons} \mathcal{N}_y \;=\; \{X \cup p(X) : X \in p^{-1}(\mathcal{N}_x)\} \cup \left(\mathcal{N}_y \setminus p^{-1}(\mathcal{N}_x)\right) \cup \left(\mathcal{N}_x \setminus p(\mathcal{N}_y)\right).$$

It is obvious that any colour partition of $\bar{G}_z$ induces colour partitions of $\bar{G}_x$ and $\bar{G}_y$, and so every $\bar{G}_z \models \nu(\mathcal{N}_z)$ is obtained as $\mathcal{N}_z = (\mathcal{N}_x \overset{p}{\rightleftharpoons} \mathcal{N}_y)$ for some $\bar{G}_x \models \nu(\mathcal{N}_x)$ and $\bar{G}_y \models \nu(\mathcal{N}_y)$, and suitable $p \in \mathcal{J}(\mathcal{N}_y, \mathcal{N}_x)$.

For $p \in \mathcal{J}(\mathcal{N}_y, \mathcal{N}_x)$, we define the "signature" $Sig(p)$ as an edge-weighted bipartite graph D_p on the vertex set $\mathcal{S} \dot{\cup} \mathcal{S}$, where $\mathcal{S}$ is the family of all subspaces of $\mathrm{GF}(2)^t$. $f = \Psi\Psi' \in \mathcal{S} \times \mathcal{S}$ is an edge of D_p iff there is $Y \in \mathcal{N}_y$ in the domain of p such that $\Psi' = \langle \gamma(\bar{G}_y, Y) \rangle$ and $\Psi = \langle \gamma(\bar{G}_x, p(Y)) \rangle$. The weight of the edge f is then the number of such witnesses $Y \in \mathcal{N}_y$.

Let z carry the composition operator $\otimes[g \mid f_1, f_2]$ in T, i.e. $\bar{G}_z = \bar{G}_x \otimes[g \mid f_1, f_2]\ \bar{G}_y$. Notice that $\Gamma(\bar{G}_z, \mathcal{N}_z)$ where $\mathcal{N}_z = (\mathcal{N}_x \overset{p}{\rightleftharpoons} \mathcal{N}_y)$ is uniquely determined by $\Gamma(\bar{G}_x, \mathcal{N}_x)$, $\Gamma(\bar{G}_y, \mathcal{N}_y)$, by the relabelings f_1, f_2, and by $Sig(p)$. We furthermore define on the same vertex set a special bipartite graph $D_g^{\perp}$ (depending on the relabeling g) with $E(D_g^{\perp}) = \{\Psi\Psi' \in \mathcal{S} \times \mathcal{S} : \Psi \perp g(\Psi')\}$. The purpose is to explicitly define which colour classes could be merged through $\otimes[g \mid f_1, f_2]$ without creating edges inside any class.

From Lemma 4.2 we immediately conclude:

(4.7) $\bar{G}_z \models \nu(\mathcal{N}_z)$ where $\mathcal{N}_z = (\mathcal{N}_x \overset{p}{\rightleftharpoons} \mathcal{N}_y)$ for some $\bar{G}_x \models \nu(\mathcal{N}_x)$, $\bar{G}_y \models \nu(\mathcal{N}_y)$ and $p \in \mathcal{J}(\mathcal{N}_y, \mathcal{N}_x)$ if, and only if, $Sig(p)$ is a subgraph of $D_g^{\perp}$.

With Claim (4.7) at hand it is straightforward how to compute the set $M_T(z)$ from the sets $M_T(x)$ and $M_T(y)$. We loop through all members $\Gamma_x \in M_T(x)$ and $\Gamma_y \in M_T(y)$, and all admissible signatures Sig (i.e. nonnegative integer weightings of the above bipartite graph $D_g^{\perp}$ by (4.7)), test a simple consistency condition, and then possibly add the resulting Γ_z (easily computable) to $M_T(z)$. This consistency condition on Sig is that, for each its vertex Ψ, the sum of the weights of the edges of Sig incident with Ψ is at most the multiplicity of Ψ in Γ_x or Γ_y, respectively.

Finally, the chromatic number of G equals the least cardinality of a member of $M_T(r)$ where r is the root of T. Such a leaves-to-root dynamic algorithm then

runs in time $O\big(m(G,t)^2 \cdot w(G,t) \cdot S(t)^2 \cdot t^3 \cdot |V(G)|\big)$, where $m(G,t)$ denotes the number of possible distinct $\Gamma(\bar{G},\mathcal{N})$, and $w(G,t)$ stands for the number of distinct weightings of the graph $D_g^{\perp}$. Each Γ_z is then determined from Γ_x, Γ_y and Sig in $S(t)^2t^3$ steps where $S(t)$ is estimated in Lemma 4.4.

For simplicity, we provide only short arguments giving rather weak (but sufficient) bounds on m, w here: $m(G,t)$ can be bounded from above by $|V(G)|^{S(t)}$— consider that the multiplicity of any subspace in the multiset $\Gamma(\bar{H},\mathcal{N})$ is at most the number of nonempty colour classes. With analogous arguments we also get $w(G,t) \leq |V(H)|^{S(t)^2}$. These estimates then lead to a runtime bound of order $|V(H)|^{h(t)}$ where $h(t) \leq 2S(t) + S(t)^2 + o(t^2) + 1 = O(S(t)^2) = O(2^{t(t+1)/2})$. ∎

4.2 The chromatic polynomial

The chromatic polynomial was first introduced by Birkhoff in the context of the Four Colour problem. Although the concept seems quite technical and obscure in nature, it has since become of independent interest. The *chromatic polynomial* of G is a polynomial $P_G(x)$ such that for every nonnegative integer x, $P_G(x)$ equals the number of distinct proper colourings of G which use x colours. It is a trivial observation that, given the values of all $P_G(x)$ for $x = 1, 2, \ldots, n = |V(G)|$, finding $P_G(x)$ simply becomes a matter of resolving n (independent) equations of n unknowns.

Computing the chromatic polynomial is generally $\#P$-complete. It has been noted by [14] that the algorithm of [19] extends towards computing the chromatic polynomial on graphs of bounded clique-width, and the same statement occurs with a proof in [1]. We improve those results with:

Theorem 4.8. *Assume that an input graph G is given in the form of a t-labeling parse tree T. Then the chromatic polynomial of G can be computed in time*

$$O\left(|V(G)|^{h(t)}\right) \text{ where } h(t) = O(2^{t(t+1)/2}).$$

Proof. As already explained, we need to modify the algorithm of Theorem 4.1 so that it will compute the number of distinct proper colourings of G having the prescribed number of colour classes. Fortunately, we do not need to process all possible colour partitions of G; thanks to (4.6), we only have to remember the numbers of partitions $\mathcal{N}$ determining the same value of $\Gamma(\bar{G}_z,\mathcal{N})$ at a node z.

Let $\vec{\alpha} = \big(\alpha_\Gamma : \Gamma \text{ is a multiset of subspaces of } \mathrm{GF}(2)^t\big)$ be a vector of free variables. Although $\vec{\alpha}$ is infinite, we shall actually use a finite part of it. Our algorithm shall compute the linear multivariete (symbolic) polynomial $R(\bar{G}_z)[\vec{\alpha}] = \sum_\Gamma q_\Gamma \cdot \alpha_\Gamma$ where q_Γ stands for the number of distinct colour partitions $\mathcal{N}$ of G_z such that $\Gamma(\bar{G}_z,\mathcal{N}) = \Gamma$.

The algorithm generally proceeds as that of Theorem 4.1. Specifically, at a node z of T with the sons x and y, we compute straightforwardly

$$R(\bar{G}_z)[\vec{\alpha}] = R(\bar{G}_x)[\vec{\alpha}] \cdot R(\bar{G}_y)[\vec{\alpha}]$$

and then apply all the necessary substitutions: For every pair $\alpha_{\Gamma_1}, \alpha_{\Gamma_2} \in \vec{\alpha}$, we replace the term $\alpha_{\Gamma_1} \cdot \alpha_{\Gamma_2}$ with a sum, over all admissible signatures Sig (4.7), of the terms $r_{Sig} \cdot \alpha_{\Gamma_{Sig}}$ where Γ_{Sig} is the multiset uniquely determined by Γ_1, Γ_2, Sig, and the composition relabelings at z. The number r_{Sig} is defined as follows.

Let $\mathcal{N}_1, \mathcal{N}_2$ be such that $\Gamma(\bar{G}_x, \mathcal{N}_1) = \Gamma_1$ and $\Gamma(\bar{G}_x, \mathcal{N}_2) = \Gamma_2$. Then r_{Sig} is the number of distinct partial injective mappings $p \in \mathcal{J}(\mathcal{N}_2, \mathcal{N}_1)$ such that $Sig(p) = Sig$. One can check that this quantity does not depend on a particular choice of $\mathcal{N}_1, \mathcal{N}_2$, and that it can be straightforwardly computed from Γ_1, Γ_2 and Sig along with computing Γ_{Sig}.

Finally, we extract the numbers of colourings of G from the above computed $R(\bar{G})[\vec{\alpha}]$. We note two points: First, a c-colouring does not have to use all c colours, and so we have to count with all colour partitions of at most c classes. Second, a traditional colouring distinguishes between the colours, while our colour classes do not. Hence we obtain the total number of distinct c-colourings of G from $R(\bar{G})[\vec{\alpha}]$ if we substitute $\alpha_\Gamma = c!/(c - |\Gamma|)!$ and $\alpha_\Gamma = 0$ if $|\Gamma| > c$. ∎

4.3 Hamiltonian path

The last algorithm illustrating the strength our approach is based on a Hamiltonian path algorithm for graphs of bounded clique-width [7]. While the goal of the previous two subsections was to demonstrate how one can design new more efficient algorithms on labeling parse trees with tools of linear algebra, here we show how to translate an existing clique-width-based algorithm to a formally better setting within our scheme.

Theorem 4.9. *Assume that an input graph G is given in the form of a t-labeling parse tree T. Then one can determine whether G has a Hamiltonian path in time*

$$O\left(|V(G)|^{\ell(t)}\right) \text{ where } \ell(t) = O(4^t)\,.$$

Proof. We say that a set of edges $F \subseteq E(G)$ is *linear* if the subgraph $G \restriction F = \big(V(G), F\big)$ is a collection of disjoint paths. We also write $G \models \lambda(F)$ if F is a Hamiltonian path in G. Having two linear subsets $F_1 \subseteq E(G_1)$ and $F_2 \subseteq E(G_2)$, we define the equivalence relation $(\bar{G}_1, F_1) \approx_{\lambda,t} (\bar{G}_2, F_2)$ if and only if, for all t-labeled graphs $\bar{H}$ and all linear $F \subseteq E(H)$, it holds

$$\begin{aligned}(4.10)\qquad &\exists F_3 \subseteq E(\bar{G}_1 \otimes \bar{H}) \setminus (E(G_1) \cup E(H)) : \ \big(\bar{G}_1 \otimes \bar{H}\big) \models \lambda(F_1 \cup F_3 \cup F)\\ \iff\ &\exists F_4 \subseteq E(\bar{G}_2 \otimes \bar{H}) \setminus (E(G_2) \cup E(H)) : \ \big(\bar{G}_2 \otimes \bar{H}\big) \models \lambda(F_2 \cup F_4 \cup F)\,.\end{aligned}$$

Obviously, $\approx_{\lambda,t}$ captures all information necessary to decide which linear subsets of G_1 extend to Hamiltonian paths in a join graph.

Similarly to [7], for a linear subset $F \subseteq E(\bar{G})$, we define a multiset of labeling pairs $\Pi(\bar{G}, F) = \{(lab(x), lab(y)) : \ x, y \text{ are the ends of a path in } G \restriction F\}$ (an isolated vertex is a path with the ends $x = y$). Analogously to (4.6) we have:

(4.11) For any t-labeled graphs $\bar{G}_1, \bar{G}_2$ and any linear $F_1 \subseteq E(G_1)$ and $F_2 \subseteq E(G_2)$, it holds $(\bar{G}_1, F_1) \approx_{\lambda,t} (\bar{G}_2, F_2)$ if $\Pi(\bar{G}_1, F_1) = \Pi(\bar{G}_2, F_2)$.

Therefore, our algorithm computes, in the leaves-to-root direction on T, the sets $N_T(z) = \{\Pi(\bar{G}_z, F) : F \subseteq E(G) \text{ linear}\}$. Since there are $2^{2t} = 4^t$ distinct labeling pairs in $\mathrm{GF}(2)^t$, there are at most $|V(G)|^{4^t}$ distinct multisets $\Pi(\bar{G}_z, F)$ to be considered in each set $N_T(z)$. Obviously, G has a Hamiltonian path F if and only if $N_T(r)$ contains a multiset $\Pi(\bar{G}, F)$ of cardinality one. The rest proceeds in the same way as the previous algorithms. ∎

Remark 4.12. One of the advantages of our new proof of Theorem 4.9 is that it immediately extends towards solving directed Hamiltonian path in digraphs of bounded *bi-rank-width* (a directed analogue of rank-width, cf. [18]).

5 Concluding Notes

The list of algorithms presented in this article is by no means exhaustive. Many pseudopolynomial algorithms designed for graphs of bounded clique-width (such as for the edge-dominating set [19]) can be straightforwardly translated into our parse tree approach on rank-width, similarly to Theorem 4.9. One can expect that the time complexity of such algorithms will have a "one-level higher" exponent, as we see in Theorem 4.9. As already mentioned, the reason is that rank-width generally has an exponentially smaller value than clique-width (not that the new algorithms would be slower).

Still, the main advantage of designing algorithms on rank-decompositions of graphs is that we can efficiently compute an optimal rank-decomposition by Theorem 2.1. Even better, sometimes it is possible to use our approach to design algorithms which are actually much faster than the best known ones on clique-width; this is the case of computing the chromatic numbers and polynomials in Theorem 4.1. Determining which algorithms designed for clique-width can be radically improved in such a way remains an open question, one that we believe deserves further study.

References

1. Averbouch, I., Godlin, B., Makowsky, J.A., and Rotics, U.: Computing graph polynomials on graphs of bounded clique-width . In: *Graph-theoretic concepts in computer science*. Volume 4271 of Lecture Notes in Comput. Sci., Springer, Berlin (2006) 191–204.
2. Bui-Xuan, B.-M., Telle, J.A., and Vatshelle, M.: H-join and algorithms on graphs of bounded rankwidth. Technical Report 378, Dept. of Informatics, University of Bergen, Norway (2008) `http://www.ii.uib.no/publikasjoner/texrap/pdf/2008-378.pdf`.
3. Corneil, D.G. and Rotics, U.: On the relationship between cliquewidth and treewidth. *SIAM J. Comput.* **34**(4) (2005) 825–847.
4. Courcelle, B. and Kanté, M.M.: Graph Operations Characterizing Rank-Width and Balanced Graph Expressions. In: *Graph-theoretic concepts in computer science*. Volume 4769 of Lecture Notes in Comput. Sci., Berlin, Springer (2007) 66–75.

5. Courcelle, B., Makowsky, J.A., and Rotics, U.: Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.* **33**(2) (2000) 125–150.
6. Downey, R.G. and Fellows, M.R.: *Parameterized complexity*. Monographs in Computer Science. Springer-Verlag, New York, 1999.
7. Espelage, W., Gurski, F., and Wanke, E.: How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In: *Graph-theoretic concepts in computer science*. Volume 2204 of Lecture Notes in Comput. Sci., Berlin, Springer (2001) 117–128.
8. Fellows, M.R., Rosamond, F.A., Rotics, U., Szeider, S.: Clique-width minimization is NP-hard. In: Proceedings of the *38th annual ACM Symposium on Theory of Computing*, ACM Press New York, NY, USA (2006) 354–362
9. Fomin, F., Golovach, P., Lokshtanov, D. and Saurabh, S.: Clique-width: On the Price of Generality. In: Proceedings of the *19th Annual ACM–SIAM Symposium on Discrete Algorithms*, ACM Press New York, USA (2009) 825–834.
10. Ganian, R.: *Automata formalization for graphs of bounded rank-width*. Master thesis. Faculty of Informatics of the Masaryk University, Brno, Czech republic (2008).
11. Ganian, R. and Hliněný, P.: Automata approach to graphs of bounded rank-width. In: *Proceedings of IWOCA 2008* (2008) 4–15.
12. Ganian, R. and Hliněný, P.: On Parse Trees and Myhill–Nerode–type Tools for handling Graphs of Bounded Rank-width. Manuscript (2009) 28 p.
13. Gerber, M.U. and Kobler, D.: Algorithms for vertex-partitioning problems on graphs with fixed clique-width. *Theoret. Comput. Sci.* **299**(1-3) (2003) 719–734.
14. Gimenez, O., Hliněný, P. and Noy, M.: Computing the Tutte Polynomial on Graphs of Bounded Clique-Width. *SIAM J. Discrete Math.* **20** (2006) 932–946.
15. Goldman, J. and Rota, G.-C.: The number of subspaces of a vector space. In: *Recent Progress in Combinatorics*, ed. W.T. Tutte. Academic Press (1969) 75–83.
16. Hliněný, P.: Branch-width, parse trees, and monadic second-order logic for matroids. *J. Combin. Theory Ser. B* **96**(3) (2006) 325–351.
17. Hliněný, P. and Oum, S.: Finding Branch-decomposition and Rank-decomposition. *SIAM J. Comput.* **38** (2008) 1012–1032.
18. Kanté, M.: The rank-width of directed graphs. arXiv:0709.1433v3 (2008).
19. Kobler, D. and Rotics, U.: Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Appl. Math.* **126**(2-3) (2003) 197–221.
20. Oum, S. and Seymour, P.: Approximating clique-width and branch-width. *J. Combin. Theory Ser. B* **96**(4) (2006) 514–528.
21. Rao, M.: MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theoretical Computer Science* **377** (2007) 260–267.