Unified Approach to Polynomial Algorithms on Graphs of Bounded (bi-)Rank-width

Robert Ganian[a], Petr Hliněný[a], Jan Obdržálek[a]

[a]*Faculty of Informatics, Masaryk University, Botanická 68a, 60200 Brno Czech Republic*

Abstract

In this paper we develop new algorithmic machinery for solving hard problems on graphs of bounded rank-width and on digraphs of bounded bi-rank-width in polynomial (XP, to be precise) time. These include, particularly, graph colouring and chromatic polynomial problems, the Hamiltonian path and c-min-leaf outbranching, the directed cut, and more generally MSOL-partitioning problems on digraphs. Our focus on a formally clean and unified approach for the considered algorithmic problems is in contrast with many previous published XP algorithms running on graphs of bounded clique-width, which mostly used ad hoc techniques and ideas. The new contributions include faster algorithms for computing the chromatic number and the chromatic polynomial on graphs of bounded rank-width, and new algorithms for solving the defective colouring, the min-leaf outbranching, and the directed cut problems.

Keywords: Rank-width, bi-rank-width, XP algorithm, chromatic number, chromatic polynomial, Hamiltonian path, min-leaf outbranching, directed max-cut.

1. Introduction

We postpone all formal definitions till the next section. Rank-width, introduced by Oum and Seymour [27] in a relation to graph clique-width, is a relatively new graph complexity measure which has been receiving considerable attention over the past few years. Compared to the (perhaps better known) clique-width measure, rank-width has some major advantages: First, an optimal rank-decomposition can be efficiently constructed if the rank-width is bounded [22]. Second, a rank-decomposition (actually, a suitable modification of it, see Section 3) allows for design of formally cleaner [13], and often significantly faster

Email addresses: `xganian1@fi.muni.cz` (Robert Ganian), `hlineny@fi.muni.cz` (Petr Hliněný), `obdrzalek@fi.muni.cz` (Jan Obdržálek)

[1]The authors have been supported by the the research grant GAČR 201/09/J021, and P. Hliněný also by the research intent MSM0021622419 of the Czech Ministry of Education.

parametrized (FPT) algorithms [3, 15] than previously known ones running on a clique-width expression of the given graph. Third, analogously to clique-width, there exist natural extensions of rank-width to digraphs – cf. bi-rank-width or $GF(4)$-rank-width introduced by Kanté [23] – enjoying similar nice algorithmic properties as shown in [16].

To recapitulate, several recently published papers have been dealing with FPT algorithms on graphs of bounded rank-width [3, 5, 13, 15] and of bounded bi-rank-width [16]. On the other hand, no papers dealing specifically with XP algorithms on graphs of bounded rank-width seem to be published to date, and related papers, e.g. [2, 9, 17, 24] working on graphs of bounded clique-width, seem to mostly use ad hoc techniques and ideas for designing their algorithms.

The aim of this paper is to extend some of the core ideas of [13, 15] (which are using mathematical tools of automata theory in designing FPT algorithms on graphs of bounded rank-width) also to problems which likely do not have FPT algorithms with respect to rank-width, and hence for which we would like to get parametrized algorithms belonging to the class XP ("pseudopolynomial"). Two main advantages of our *novel unified approach* are as follows.

Firstly, the underlying mathematical framework related to finite automata and language congruences provides an easier and more precise description of our algorithms and also their proofs. This is, simply speaking, due to the fact that we use precisely defined equivalence relations on the universe of all "partial solutions" instead of (often vaguely defined) dynamic processing data ("tables").

Secondly, by a careful analysis of the mentioned equivalences over all partial solutions, we are able to provide XP algorithms whose theoretical runtime is often much better than the runtime of the corresponding algorithms which are (or can be) formulated on graphs of bounded clique-width. For this we exploit the nice algebraic properties of rank-width and its labeling parse trees (cf. Section 3 and Lemma 4.2).

Section 2 presents some basic definitions. In Section 3 we describe the parse-tree formalism (analogous to [5, 12]) for handling rank-decompositions of graphs. Section 4 then presents three new exemplary XP algorithms for colouring problems on graphs of bounded rank-width. An extension of rank-width and the parse-tree formalism to directed graphs is shown in Section 5, which is then applied to solving selected interesting digraph problems parameterized by bi-rank-width in Section 6.

The most important new contributions of these sections can be summarized:

- [2] We provide algorithms computing the chromatic number and the chromatic polynomial of a graph of rank-width t in time $O\left(n^{h(t)}\right)$ where $h(t) = 2^{1+t(t+1)/2} + O(1)$, largely beating the previous algorithms by

[2]These two algorithms have already appeared in the conference proceedings version of this paper [14], and they have an improved presentation here.

Kobler and Rotics [24] and Averbouch et al [2], respectively. See Theorems 4.1 and 4.7.

- [3] We extend the recent XP algorithm for defective (ℓ, q)-colouring (with ℓ fixed) by Kolman et al [25] from tree-width bounded graphs to graphs of rank-width t, providing an XP algorithm of runtime $O\left(n^{k(t)}\right)$ where $k(t) = 4\ell \cdot 2^t + O(1)$. See Theorem 4.8.
- [3] We provide new XP algorithms for solving the unweighted max-directed-cut and the c-min-leaf (c fixed) outbranching problems on graphs of bi-rank-width t, for which no analogous algorithms have been known before. These run in time $O\big(n^{4\cdot 2^t + O(1)}\big)$ and $O\big(n^{4\cdot 2^{ct+c+t} + O(1)}\big)$, respectively. See Theorems 6.4 and 6.7.

Although above running times with terms like $2^{O(t)}$ in the exponent may look horrible at the first sight, we note that there exist interesting graph classes having *very small* rank-width, such as the graphs of rank-width $t = 1$ which have been known as "distance-hereditary" graphs for quite long time.

Final Section 7 then outlines some interesting generalizations of our algorithms (including, for instance, an alternative approach to the so called MSOL-partitioning problems of Rao [28]), and possible further research directions.

2. Definitions and Basics

For now we only consider finite undirected simple graphs, i.e. without loops or multiple edges. We 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 [15]. Further definitions related to digraphs are then presented in Section 5.

The reader should be aware of the notion of *fixed-parameter tractable* [8] algorithms (FPT algorithms in short), which are the algorithms running in time $O(n^p \cdot 2^{f(k)})$ for a constant p, a parameter k (rank-width in our case) and any (computable) function f. Some NP-hard problems such as deciding whether a graph is 3-colourable do have FPT algorithms when parameterized by clique-width, see e.g. [6]. On the other hand, [11] have recently proved that various problems, such as the chromatic number or Hamiltonicity, likely cannot 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* [8] – 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 [2, 9, 17, 24, 28]. Our goal in this paper is to design and practically use a mathematically precise and sound formalism for solving problems on graphs of bounded rank-width in XP time. This extends the

[3] All these algorithms are new here – not presented in the conference proceedings [14].

Myhill–Nerode type automata formalism which we have introduced in [13, 15] for FPT algorithms on such graphs.

Rank-width is usually defined in terms of generalized branch-width. We therefore start with the definition of branch-width:

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) where T is a subcubic tree and μ 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 put $M = E(G)$, and take f to be 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 ([27]). For a graph G, let $\boldsymbol{A}_G[U, W]$ be the 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 $a_{u,w}$, where $u \in U$ 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. The important property of rank-width is that it can be verified in polynomial time, as stated by the following theorem.

Theorem 2.1 ([22]). *For every parameter 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 adjacency matrices consists of all 1s. It is similar for 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 to 2. A rank-decomposition of the cycle C_5 is shown in Figure 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.

Rank-width is closely tied to another width parameter called *clique-width* [9]. A graph has bounded rank-width if and only if it has bounded clique-width.

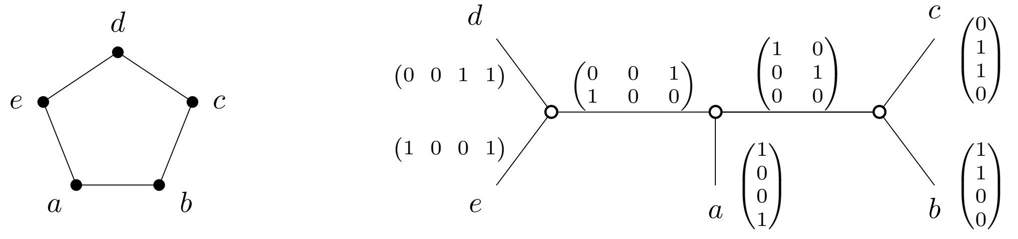

d
e
c
a
b
d
e
c
a
b
(0 0 1 1)
(1 0 0 1)
$\begin{pmatrix}0&0&1\\1&0&0\end{pmatrix}$
$\begin{pmatrix}1&0\\0&1\\0&0\end{pmatrix}$
$\begin{pmatrix}0\\1\\1\\0\end{pmatrix}$
$\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$
$\begin{pmatrix}1\\1\\0\\0\end{pmatrix}$

Figure 1: A rank-decomposition of the graph cycle C_5.

However, there is no equivalent of Theorem 2.1 for clique-width [10], and the value of clique-width can be up to exponentially larger than rank-width [4], both facts making 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 "bare" rank-decomposition of a graph.

3. Rank-width (labeling) Parse Trees

In a search for a "more suitable form" of a rank-decomposition, Courcelle and Kanté [5] 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 3.2). Here we introduce (following [12] and [13, 15]) 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 [3].

t-labeled graphs. A *t-labeling* of a graph is a mapping $lab : V(G) \to 2^{L_t}$ which to each vertex of G assigns a set of *labels* from $L_t = \{1, 2, \dots, t\}$. (This is equivalent to multiple-coloured graphs of [5].) 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. [5] 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.

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 $\boldsymbol{T}_f$. 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 $lab' = lab \times \boldsymbol{T}_f$ 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 $f \circ lab(v) = \{2, 4\} = \{1, 3, 4\} \vartriangle \{1, 2, 3\}$ (here the operator $\vartriangle$ stands for symmetric difference).

We will now define three different operators on t-labeled graphs. These operators resemble the operators for building k-expressions for clique-width, and were first introduced in [13]. The first operator is $\odot$, a nullary operator creating a single new graph vertex of label $\{1\}$.

Let $\bar{G}_1 = (G_1, lab_1)$ and $\bar{G}_2 = (G_2, lab_2)$ be t-labeled graphs. *t-labeling join* of $\bar{G}_1$ and $\bar{G}_2$, $\bar{G}_1 \otimes \bar{G}_2$, is defined as taking the disjoint union of G_1 and G_2 and adding all edges (u, v) such that $|lab_1(u) \cap lab_2(v)|$ is odd, where $u \in V(G_1), v \in V(G_2)$. Considering the scalar product $\cdot$ of vectors, $\{u, v\}$ is an edge of $\bar{G}_1 \otimes \bar{G}_2$ if and only if $lab_1(u) \cdot lab_2(v) = 1$ over GF(2). The resulting graph is unlabeled.

The third operator combines together t-labeling join and and t-relabeling. For t-relabelings $f_1, f_2, g : L_t \to 2^{L_t}$, let $\otimes[g \mid f_1, f_2]$ be a binary operator—called *t-labeling composition* (as bilinear product of [5])—over pairs of t-labeled graphs $\bar{G}_1 = (G_1, lab_1)$ and $\bar{G}_2 = (G_2, lab_2)$ defined as follows:

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

where a new labeling is $lab(v) = f_i \circ lab_i(v)$ for $v \in V(G_i)$, $i = 1, 2$. In other words, t-labeling composition performs t-labeling join of $\bar{G}_1$ and $g(\bar{G}_2)$, and then relabels the vertices originally from $V(G_1)$ using t-relabeling f_1 and vertices from $V(G_2)$ using f_2. The resulting graph $\bar{H}$ is again a t-labeled graph. Notice that $\{u, v\}$ is an edge of $\bar{H}$ if and only if $lab_1(u) \times \boldsymbol{T}_g^T \times lab_2(v)^T = 1$ over GF(2).

Definition 3.1 ([12], Definition 6.11)**.** *A t-*labeling parse tree *T 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.*

For an illustration see Figure 2, 3.

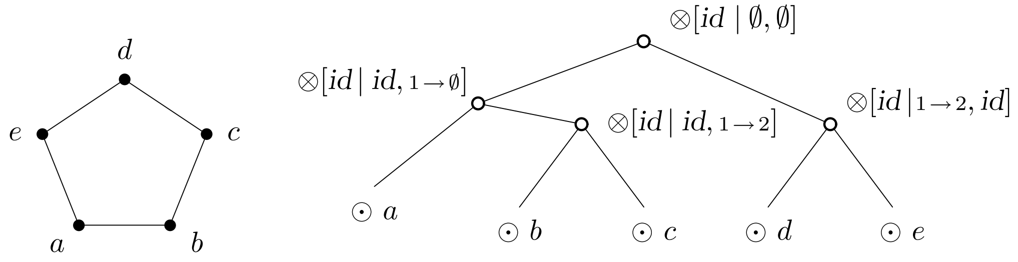

d
e
c
a
b
$\otimes[id \mid \emptyset, \emptyset]$
$\otimes[id \mid id, 1 \to \emptyset]$
$\otimes[id \mid id, 1 \to 2]$
$\otimes[id \mid 1 \to 2, id]$
$\odot\ a$
$\odot\ b$
$\odot\ c$
$\odot\ d$
$\odot\ e$

Figure 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).

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

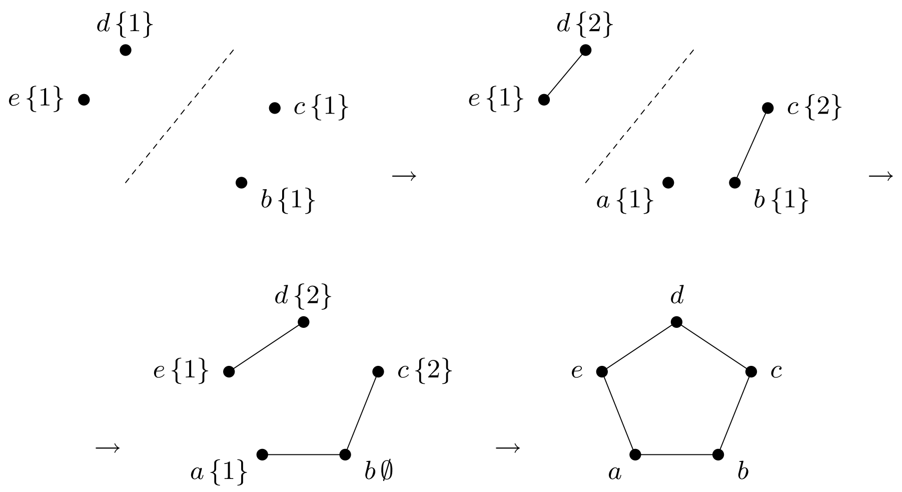


Figure 3: "Bottom-up" generation of C_5 by the parse tree from Figure 2.

Theorem 3.2 (Rank-width parsing theorem [5, 15]). *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 $O(t^2 \cdot |V(G)|^2)$.*

4. XP Algorithms for some Graph Colouring Problems

We start with an informal explanation of our unified approach to designing XP algorithms on graphs of bounded rank-width. Importantly, we see the classical Myhill–Nerode theorem in automata theory as the starting point of formal understanding of dynamic programming algorithms: Such an algorithm typically collects "all relevant information" about the studied problem on a local part of the input, and then processes this information "through" the whole input. The task is to determine what the words "all relevant information" mean here, and to prove at the same time that this information is not too large.

In the easier case of typical FPT dynamic algorithms, we can simply stipulate that a certain "formal language congruence" (the *canonical equivalence* as in, e.g. [15, 21]) has finitely many classes, and thus there is a finite tree automaton associated with our problem by the Myhill–Nerode theorem. For an explanation of the relation of our t-labeled graphs to classical formal languages, we note that word concatenation is replaced with the t-labeling join operation $\otimes$ of Section 3. Then the "relevant information" needed in dynamic processing would be precisely described by the states of the associated finite automaton.

For the problems we are considering here, however, the relevant congruence has infinitely many classes in general, but we can at least "nicely estimate" the number of classes relatively to the input graph size.

4.1. Computing the chromatic number

We illustrate the diverse aspects of our formalism on the graph chromatic number problem, for which we strongly improve runtime over the previous algorithm of Kobler and Rotics [24] that runs in time $O\big(n^{4^k}\big)$ on graphs of clique-width k. When comparing this with our Theorem 4.1, 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 [4]. Since [11] have shown that computing the chromatic number is a $W[2]$-hard problem when parameterized by the clique-width / rank-width, we "cannot hope" for an FPT algorithm here.

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 by an XP algorithm running in time*

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

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 an ordered partition $\mathcal{N}$ of $V(G)$ into pairwise disjoint sets (possibly empty) such that each $X \in \mathcal{N}$ is independent in G. The *chromatic number* of a graph G is the minimum number of nonempty classes in a colour partition of G.

Since we do not know the number of necessary colours in advance, we formally allow ordered partitions with countably many parts, padding the rest with empty classes. We, however, always have only finite number of nonempty parts in $\mathcal{N} = (X^0, X^1, X^2, \dots)$ and this number of nonempty parts we denote by $\|\mathcal{N}\|$. Having ordered colour partitions $\mathcal{N}_i = (X_i^0, X_i^1, \dots)$ for $i = 1, 2$, we denote by $\mathcal{N}_1 \uplus \mathcal{N}_2 = \big(X_1^0 \cup X_2^0, X_1^1 \cup X_2^1, \dots\big)$, and for a permutation $\pi : \omega \to \omega$ we write $\pi(\mathcal{N}) = \big(X^{\pi(0)}, X^{\pi(1)}, \dots\big)$.

We will also need a few preliminary technical results. Considering a t-labeled graph $\bar{G} = (G, lab)$, 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 $\langle\gamma(\bar{G}, X)\rangle$. The core idea is that one only needs to remember this subspace for making a future decision whether merging X with another set Y maintains the property being independent:

Lemma 4.2 (also [3, 15]). *Assume t-labeled graphs $\bar{G}$ and $\bar{H}$, and arbitrary nonempty 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 any vertex of X and any 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 ([19], cf. [15, Proposition 6.1])**.** *The number $S(t)$ of subspaces of the binary vector space* $\mathrm{GF}(2)^t$ *satisfies $S(t) \le 2^{t(t+1)/4} - 2$ for all $t \ge 12$.* ∎

The algorithm for Theorem 4.1, presented below, follows the typical dynamic programming paradigm on the parse tree of an input graph. Its core novel contribution (besides faster runtime) is in using an equivalence relation $\approx_{\nu,t}$ over all possible coloured subgraphs, which is analogous to classical language congruence in automata theory, for defining the “information” a dynamic programming algorithm needs to remember about all the colourings of a subgraph processed so far. This information is then easily “made explicit” in Claim 4.5, and the way the information can be efficiently processed is formally described in Claim 4.6. As a result, our algorithm (as compared to [24]) has quite short, and mathematically very precise at the same time, description and proof.

Proof of Theorem 4.1. Given a graph G, we write $G \models \nu(\mathcal{N})$ to say that an ordered set family $\mathcal{N} \in 2^{V(G)\times\omega}$ is a proper colour partition of G. For any t-labeled graphs $\bar{G}_1, \bar{G}_2$ and any colour partitions $\mathcal{N}_1 \in 2^{V(G_1)\times\omega}$, $\mathcal{N}_2 \in 2^{V(G_2)\times\omega}$, we define

$$(\bar{G}_1, \mathcal{N}_1) \approx_{\nu,t} (\bar{G}_2, \mathcal{N}_2) \tag{1}$$

if and only if $\|\mathcal{N}_1\| = \|\mathcal{N}_2\|$ and, for all t-labeled graphs $\bar{H}$ and all colour partitions $\mathcal{N} \in 2^{V(H)\times\omega}$, the following holds true:

$$\begin{aligned} &\exists \text{ permutation } \pi_1 : \omega \to \omega \text{ such that } \big(\bar{G}_1 \otimes \bar{H}\big) \models \nu\big(\pi_1(\mathcal{N}_1) \uplus \mathcal{N}\big) \\ \iff\ &\exists \text{ permutation } \pi_2 : \omega \to \omega \text{ s.t. } \big(\bar{G}_2 \otimes \bar{H}\big) \models \nu\big(\pi_2(\mathcal{N}_2) \uplus \mathcal{N}\big) \end{aligned}$$

Note that $(\bar{G}_1, \mathcal{N}_1) \approx_{\nu,t} (\bar{G}_2, \mathcal{N}_2)$ means that there is no real difference between $(\bar{G}_1, \mathcal{N}_1)$ and $(\bar{G}_2, \mathcal{N}_2)$ concerning the possibility of merging their colour classes with any joined graph $\bar{H}$. Hence, $\approx_{\nu,t}$ captures all information necessary to decide which colourings of $\bar{G}_i$ extend to colourings of any larger $\bar{G}_i \otimes \bar{H}$.

Let $\Gamma(\bar{G}, \mathcal{N}) = \{\langle\gamma(\bar{G}, X)\rangle \mid \emptyset \ne X \in \mathcal{N}\}$ denote a multiset of subspaces of $\mathrm{GF}(2)^t$ which, informally, stores the information about the (unordered) subspaces generated by the colour classes of $\mathcal{N}$ (cf. Lemma 4.2). Our crucial new finding, inspired by the colouring algorithm in [24], reads:

Claim 4.5. *For any t-labeled graphs $\bar{G}_1, \bar{G}_2$ and any $\mathcal{N}_i \in 2^{V(G_i)\times\omega}$, $i = 1, 2$ 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 $\Gamma(\bar{G}_1, \mathcal{N}_1) = \Gamma(\bar{G}_2, \mathcal{N}_2)$.*

To prove Claim 4.5 let $\mathcal{N}_i = (X_i^0, X_i^1, \dots)$ where $i = 1, 2$, let $\mathcal{N} = (Y^0, Y^1, \dots)$, and let $\mathcal{N}_i^+$ denote $\pi_i(\mathcal{N}_i) \uplus \mathcal{N}$, $i = 1, 2$. Let us assume $\Gamma(\bar{G}_1, \mathcal{N}_1) = \Gamma(\bar{G}_2, \mathcal{N}_2)$, and take any π_1 and $(\bar{H}, \mathcal{N})$ such that $\big(\bar{G}_1 \otimes \bar{H}\big) \models \nu(\mathcal{N}_1^+)$. (The case for π_2 and G_2 is symmetric.)

We choose any permutation $\rho : \omega \to \omega$ such that $\langle \gamma(\bar{G}_1, X_1^j) \rangle = \langle \gamma(\bar{G}_2, X_2^{\rho(j)}) \rangle$ for all $X_1^j \in \mathcal{N}_1$ and corresponding $X_2^{\rho(j)} \in \mathcal{N}_2$, and set $\pi_2 = \rho \circ \pi_1$. Since $\mathcal{N}_2^+$ is a partition of the vertices of $\bar{G}_2 \otimes \bar{H}$, to show $\bar{G}_2 \otimes \bar{H} \models \nu(\mathcal{N}_2^+)$ it suffices to verify that all parts of $\mathcal{N}_2^+$ are independent in $\bar{G}_2 \otimes \bar{H}$. We take any $X_2^{\pi_2(i)} \cup Y^i \in \mathcal{N}_2^+$. Then also $X_1^{\pi_1(i)} \cup Y^i \in \mathcal{N}_1^+$ and, moreover, $\pi_2(i) = \rho(\pi_1(i))$. Hence from Corollary 4.3 applied to $X_1 = X_1^{\pi_1(i)}$, $X_2 = X_2^{\pi_2(i)}$ and $Y = Y^i$, it follows that $X_2^{\pi_2(i)} \cup Y^i$ really is independent since $X_1^{\pi_1(i)} \cup Y^i$ is independent by the assumption $\big(\bar{G}_1 \otimes \bar{H}\big) \models \nu(\mathcal{N}_1^+)$, which finishes the proof of Claim 4.5.

With these preliminary results, we can proceed to the algorithm itself. 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. By Claim 4.5, 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.

If z is a leaf, then this information is trivial to construct. So let z be a node with a left son x and a right son y. We now show how to obtain the set $M_T(z)$ from the sets $M_T(x)$ and $M_T(y)$. Since the number of classes of $\approx_{\nu,t}$ is not finite, this is not a completely trivial task, but it is not difficult either. Let z carry the composition operator $\otimes[g \mid f_1, f_2]$ in T, i.e. $\bar{G}_z = \bar{G}_x \otimes[g \mid f_1, f_2]\ \bar{G}_y$.

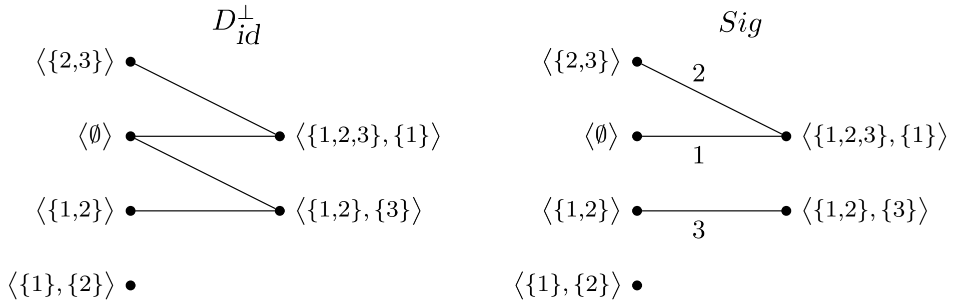
For any colour partitions $\mathcal{N}_1$ of $\bar{G}_x$ and $\mathcal{N}_2$ of $\bar{G}_y$, the multiset $\Gamma(\bar{G}_z, \mathcal{N}_1 \uplus \mathcal{N}_2)$ can be straightforwardly computed (without knowing explicitly $\mathcal{N}_1, \mathcal{N}_2$) from known $\Gamma(\bar{G}_x, \mathcal{N}_1)$, $\Gamma(\bar{G}_y, \mathcal{N}_2)$ and the relabelings f_1, f_2 if we specify also a "signature" of $\mathcal{N}_1, \mathcal{N}_2$: The *signature* $Sig(\mathcal{N}_1, \mathcal{N}_2)$ is an edge-weighted bipartite graph 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 $Sig(\mathcal{N}_1, \mathcal{N}_2)$ iff there is an index $i \in \omega$ (witness) such that, for $\emptyset \neq X_1^i \in \mathcal{N}_1$, $\emptyset \neq X_2^i \in \mathcal{N}_2$, it is $\Psi = \langle \gamma(\bar{G}_x, X_1^i) \rangle$ and $\Psi' = \langle \gamma(\bar{G}_y, X_2^i) \rangle$. The weight of the edge f is then the number of such witnesses i.

Informally, the signature $Sig(\mathcal{N}_1, \mathcal{N}_2)$ "loosely specifies" the way the (nonempty) colour classes of $\mathcal{N}_1$ are merged with those of $\mathcal{N}_2$ — the edge weights represent the number of pairs of colour classes which will be merged. Notice that although looping through all possible bijections between $\mathcal{N}_1$ and $\mathcal{N}_2$ is not feasible, the number of distinct signatures $Sig(\mathcal{N}_1, \mathcal{N}_2)$ is relatively small – polynomial in $|V(G)|$ as specified later.

We furthermore have to decide whether the join $\mathcal{N}_1 \uplus \mathcal{N}_2$ is a proper colour partition of G_z. For that purpose we define on $\mathcal{S} \dot\cup \mathcal{S}$ a bipartite graph $D_g^\perp$ (depending only on the relabeling g) with $E(D_g^\perp) = \{\Psi\Psi' \in \mathcal{S} \times \mathcal{S} \mid \Psi \perp g(\Psi')\}$. See Figure 4 left. From Lemma 4.2 we immediately conclude:

Claim 4.6. *$\bar{G}_z \models \nu(\mathcal{N}_1 \uplus \mathcal{N}_2)$ for any partitions $\mathcal{N}_1, \mathcal{N}_2$ such that $\bar{G}_x \models \nu(\mathcal{N}_1)$ and $\bar{G}_y \models \nu(\mathcal{N}_2)$ if, and only if, $Sig(\mathcal{N}_1, \mathcal{N}_2)$ is a subgraph of $D_g^{\perp}$.*

With Claim 4.6 at hand it is easy 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 bipartite graph $D_g^{\perp}$ by Claim 4.6) conforming to a simple consistency condition, and then add the resulting Γ_z to $M_T(z)$. The 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. See Figure 4.



$D_{id}^{\perp}$
$\langle\{2,3\}\rangle$
$\langle\emptyset\rangle$
$\langle\{1,2\}\rangle$
$\langle\{1\},\{2\}\rangle$
$\langle\{1,2,3\},\{1\}\rangle$
$\langle\{1,2\},\{3\}\rangle$
Sig
$\langle\{2,3\}\rangle$
2
$\langle\emptyset\rangle$
1
$\langle\{1,2\}\rangle$
3
$\langle\{1\},\{2\}\rangle$
$\langle\{1,2,3\},\{1\}\rangle$
$\langle\{1,2\},\{3\}\rangle$

Figure 4: Illustration of a fragment (on the left) of the graph $D_g^{\perp}$ which, by Claim 4.6, encodes admissible compositions of colour partitions of $\bar{G}_x$ and $\bar{G}_y$ in the composed graph $\bar{G}_z = \bar{G}_x \otimes[g \mid f_1, f_2]\ \bar{G}_y$. Here it is $g = id$ for simplicity. Since, for instance, the labeling $\{1\}$ vector is not orthogonal to $\{1,2\}$, there is no edge between $\langle\{1\},\{2\}\rangle$ and $\langle\{1,2\},\{3\}\rangle$. On the right, there is a corresponding fragment of a signature (weighted subgraph of $D_g^{\perp}$) $Sig = Sig(\mathcal{N}_1, \mathcal{N}_2)$ telling us that three of the $\mathcal{N}_1$-classes which generate the same subspace $\Psi = \langle\{1,2\}\rangle$ in $\bar{G}_x$ should be merged with three (no matter which) of the $\mathcal{N}_2$-classes which generate $\Psi' = \langle\{1,2\},\{3\}\rangle$. Analogously for the remaining weighted edges.

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 $S(t)^2 t^3$ steps where the number of subspaces $S(t)$ 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 $\Gamma(\bar{G}, \mathcal{N})$ is at most the number of nonempty colour classes. Analogously, the number of edges of $D_g^{\perp}$ is always at most $S(t)^2$, and so $w(G,t) \leq |V(G)|^{S(t)^2}$. These estimates then lead to a runtime bound of order $|V(G)|^{h(t)}$ where

$$h(t) \leq 2S(t) + S(t)^2 + O(1) \leq 2S(t)^2 + O(1) = 2^{1+t(t+1)/2} + O(1)\,.$$

■

4.2. 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 at most 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 [18] that the algorithm of [24] extends towards computing the chromatic polynomial on graphs of bounded clique-width, and the same statement occurs with a proof in [2]. We improve these results to match Theorem 4.1:

Theorem 4.7. *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 XP time*

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

Proof. We modify the algorithm of Theorem 4.1 so that it will compute the number of distinct proper colour partitions $\mathcal{N} \in 2^{V(G)\times c}$ of G having exactly c classes (including possible empty ones). The core tools are again the equivalence $\approx_{\nu,t}$ and Claim 4.5 telling us that it is enough to remember the numbers of partitions $\mathcal{N}$ determining the same values of $\Gamma(\bar{G}_z, \mathcal{N})$ at any node z.

Let $\vec{\alpha} = \big(\alpha_\Gamma \mid \Gamma$ is a multiset of subspaces of $\mathrm{GF}(2)^t, |\Gamma| \le c\big)$ be a vector of free variables. Our algorithm shall compute the linear multivariate (symbolic) polynomial $R(\bar{G}_z)[\vec{\alpha}] = \sum_\Gamma q_\Gamma \cdot \alpha_\Gamma$ where q_Γ stands for the number of distinct *unordered* colour partitions $\mathcal{U}$ of G_z such that $\Gamma(\bar{G}_z, \mathcal{U}) = \Gamma$.

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 these 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 (see Claim 4.6), of the terms $r_{Sig} \cdot \alpha_{\Gamma_{Sig}}$ where Γ_{Sig} is the multiset uniquely determined by Γ_1, Γ_2, the composition relabelings at z, and Sig. The number r_{Sig} is defined as follows.

Let (any) colour partitions $\mathcal{N}_1, \mathcal{N}_2$ of $\bar{G}_x, \bar{G}_y$ be such that $\Gamma(\bar{G}_x, \mathcal{N}_1) = \Gamma_1$ and $\Gamma(\bar{G}_y, \mathcal{N}_2) = \Gamma_2$. Then r_{Sig} is the number of distinct unordered partitions determined from the set $\big\{\mathcal{N}_1 \uplus \pi(\mathcal{N}_2) \mid \pi : \omega \to \omega \text{ s.t. } Sig(\mathcal{N}_1, \pi(\mathcal{N}_2)) = Sig\big\}$. One can check that this quantity does not depend on a particular choice of $\mathcal{N}_1, \mathcal{N}_2$ and can be easily computed from Γ_1, Γ_2 and Sig via the formula (which is in fact symmetric in Γ_1 and Γ_2):

$$r_{Sig} = \prod_{x\in\Gamma_1} \binom{m_{\Gamma_1}(x)}{w(xy_1), \ldots, w(xy_{d(x)}),\ m_{\Gamma_1}(x) - wd(x)} \cdot \prod_{y\in\Gamma_2} \binom{m_{\Gamma_2}(y)}{wd(y)} wd(y)!\,,$$

where $x \in \Gamma$ run over the elements of Γ without repetition, $m_\Gamma(x)$ is the multiplicity of x in Γ, w is the edge-weight function of Sig and wd means the "weighted degree" of a vertex of Sig – the sum of incident edge weights, and $y_1, \dots, y_{d(x)}$ run through the neighbours of x in Sig.

Finally, since the coefficient of α_Γ in the computed polynomial $R(\bar{G})[\vec{\alpha}]$ by definition counts certain unordered colour partitions with exactly $|\Gamma|$ colours, we get the resulting number of c-colourings of G (i.e. the number of ordered colour partitions from c colours) from $R(\bar{G})[\vec{\alpha}]$ by substituting $\alpha_\Gamma = c!/(c - |\Gamma|)!$. ∎

4.3. Defective colourings

Defective or improper colouring problems (partitions of graphs into vertex parts of bounded degree) appear in several research papers on graph theory, but do not seem to be widely considered from the algorithmic point of view (see, e.g., the extensive list of references provided in [25]).

Formally, a *defective (ℓ, q)-colouring* of a graph G is a partition of the vertex set of G into ℓ parts such that each part induces a subgraph of maximum degree at most q. Taking q as a fixed parameter (and thus optimizing on ℓ), this problem fits easily into the so called MSOL-partitioning framework of Rao [28] (see also Section 7), and hence this problem is solvable by an XP algorithm on graphs of bounded tree-width or clique-width. On the other hand, the situation gets very different if one fixes ℓ and optimizes on q, a case that does not fit into established frameworks. Kolman et al [25] have solved the defective (ℓ, q)-colouring problem (ℓ fixed) on graphs of bounded tree-width with an XP algorithm. What follows is our new generalization of their algorithm.

Theorem 4.8. *Assume that an input graph G is given in the form of a t-labeling parse tree T. Then the defective (ℓ, q)-colouring problem with a fixed parameter ℓ (i.e. minimizing q) can be solved on G with an XP algorithm running in time*

$$O\left(|V(G)|^{k(t,\ell)}\right) \text{ where } k(t, \ell) = 4\ell \cdot 2^t + O(1)\,.$$

Proof. According to definition, a partition $\mathcal{N}$ of $V(G)$ is a proper defective (ℓ, q)-colour partition if $\mathcal{N}$ has at most ℓ parts and each part induces a subgraph of G of maximum degree $\leq q$. Hence we will write $G \models \tau_q(\mathcal{N})$ to say that an ordered partition $\mathcal{N} \in 2^{V(G)\times\ell}$ is such that the induced subgraph $G \restriction X$ has maximum degree $\leq q$, for all $X \in \mathcal{N}$.

For integer q, any t-labeled graphs $\bar{G}_1, \bar{G}_2$, and any partitions $\mathcal{N}_i \in 2^{V(G_i)\times\ell}$, $i = 1, 2$, we define

$$(\bar{G}_1, \mathcal{N}_1) \approx_{\tau_q,t} (\bar{G}_2, \mathcal{N}_2)$$

if and only if, for all t-labeled graphs $\bar{H}$ and all $\mathcal{N} \in 2^{V(H)\times\ell}$, it holds

$$\left(\bar{G}_1 \otimes \bar{H}\right) \models \tau_q(\mathcal{N}_1 \uplus \mathcal{N}) \iff \left(\bar{G}_2 \otimes \bar{H}\right) \models \tau_q(\mathcal{N}_2 \uplus \mathcal{N})\,.$$

Compared to $\approx_{\nu,t}$, used in the proof of Theorem 4.1, this definition is obviously much simpler thanks to the fact that we are dealing with a fixed number ℓ

of parts, and so it is not necessary (though also possible) to factor out their permutations. Again, the equivalence classes of $\approx_{\tau_q,t}$ are defined as to capture all the information needed during the run of a dynamic programming algorithm for our problem.

Having a t-labeled graph $\bar{G} = (G, lab)$ and $X \subseteq V(G)$, we set $D(\bar{G}, X) = \big((d_a, |X_a|) \mid a \in \mathrm{GF}(2)^t\big)$ to be a vector of pairs $(d_a, |X_a|)$, where $X_a = \{v \in X \mid lab(v) = a\}$ are the vertices of X labeled a and d_a is the highest degree of a vertex of X_a in the induced subgraph $G \restriction X$. I.e. we restrict ourselves to the induced subgraph $G \restriction X$, and store the number and the highest degree of vertices with labeling a, for every possible labeling. (Remember that labeling is a point of $\mathrm{GF}(2)^t$). Then we define

$$\Theta(\bar{G}, \mathcal{N}) = \big(D(\bar{G}, X) \mid X \in \mathcal{N}\big) \text{ (as a sequence).} \qquad (2)$$

Claim 4.9. *For any q, any t-labeled graphs $\bar{G}_1, \bar{G}_2$, and any partitions $\mathcal{N}_i \in 2^{V(G_i)\times\ell}$, $i = 1, 2$, it holds $(\bar{G}_1, \mathcal{N}_1) \approx_{\tau_q,t} (\bar{G}_2, \mathcal{N}_2)$ if $\Theta(\bar{G}_1, \mathcal{N}_1) = \Theta(\bar{G}_2, \mathcal{N}_2)$.*

The proof of Claim 4.9 is quite straightforward. We assume the $\Theta(\bar{G}_1, \mathcal{N}_1) = \Theta(\bar{G}_2, \mathcal{N}_2)$, and fix arbitrary $\bar{H}$ and $\mathcal{N}$. We denote by $F_{i,j} = (\bar{G}_i \otimes \bar{H}) \restriction (X_i^j \cup Y^j)$ for $i = 1, 2$, $0 \le j < \ell$, where $\mathcal{N}_i = (X_i^0, X_i^1, \dots)$, and $\mathcal{N} = (Y^0, Y^1, \dots)$. Since $D(\bar{G}_1, X_1^j) = D(\bar{G}_2, X_2^j)$, the sets X_1^j and X_2^j have the same numbers of vertices of each label, and so every vertex of Y^j will have the same degree in $F_{1,j}$ as in $F_{2,j}$. On the other hand, every vertex of X_1^j of label $a \in \mathrm{GF}(2)^t$ has the same neighbourhood in Y^j of $F_{1,j}$ as any vertex of X_2^j of label a has in Y^j of $F_{2,j}$, and the highest degrees among them are equal. Therefore, $\big(\bar{G}_1 \otimes \bar{H}\big) \models \tau_q(\mathcal{N}_1 \uplus \mathcal{N})$ holds if and only if $\big(\bar{G}_2 \otimes \bar{H}\big) \models \tau_q(\mathcal{N}_2 \uplus \mathcal{N})$ holds.

Our dynamic programming algorithm for testing defective (ℓ, q)-colourability of the graph G then proceeds as follows. At a node z of the t-labeling parse tree T of G, the stored data is precisely the set $M_T(z)$ of those $\Theta(\bar{G}_z, \mathcal{N})$ coming from partitions $\mathcal{N} \in 2^{V(G_z)\times\ell}$ such that $G_z \models \tau_q(\mathcal{N})$. According to the same arguments as used in the proof of Claim 4.9, we see that $M_T(z)$ can be easily computed from $M_T(x)$, $M_T(y)$ at the sons x, y of z in T, and from the relabelings of the composition operator at z. Finally, at the root, we simply test whether $M_T(r)$ is nonempty.

We run this algorithm for $q = 0, 1, \dots, |V(G)| - 1$ to find the optimal value of q. Since the number of distinct possible $\Theta(\bar{G}, \mathcal{N})$ is at most $|V(G)|^{2\ell\cdot 2^t}$, the runtime bound for our algorithm is $O(|V(G)|^{4\ell\cdot 2^t + O(1)})$. ∎

5. Directed Rank-width and Parse Trees

From now on we are going to deal with directed graphs, or digraphs for short. We usually call the edges of a digraph arcs. A vertex v is a *sink* (a *source*) if no arc is leaving (entering) v. We again consider only simple digraphs, i.e. those without loops and multiple arcs of the same direction (2-cycles are allowed).

As noted above, the rank-width of undirected graphs was introduced by Oum and Seymour [27] in relation to graph clique-width. While the definition of clique-width works "as is" also on digraphs, the following straightforward generalization of rank-width to digraphs (related to clique-width again) has been proposed by Kanté [23] recently: Instead of using one adjacency matrix for every cut, we use two matrices (one for arcs going "left-to-right", and the other one for arcs going "right-to-left"), the cut-rank being the sum of their ranks. This is formalized by the following definition:

Definition 5.1 (Bi-rank-width)**.** Consider a digraph G, and vertex subsets $X \subseteq V(G)$ and $Y = V(G) \setminus X$. Let $\boldsymbol{A}_X^+$ denote the $X \times Y$ 0,1-matrix with the entries $a_{i,j} = 1$ ($i \in X$, $j \in Y$) iff $(i,j) \in E(G)$, and let $\boldsymbol{A}_X^- = (\boldsymbol{A}_Y^+)^T$. The *bi-cutrank* function of G is defined as the sum of the ranks of these two matrices $brk_G(X) = \mathrm{rank}(\boldsymbol{A}_X^+) + \mathrm{rank}(\boldsymbol{A}_X^-)$ over the binary field GF(2).

The *bi-rank-width* $brwd(G)$ and *bi-rank-decomposition* of G is the branch-width and branch-decomposition of the bi-cutrank function brk_G. See an illustration in Figure 5.

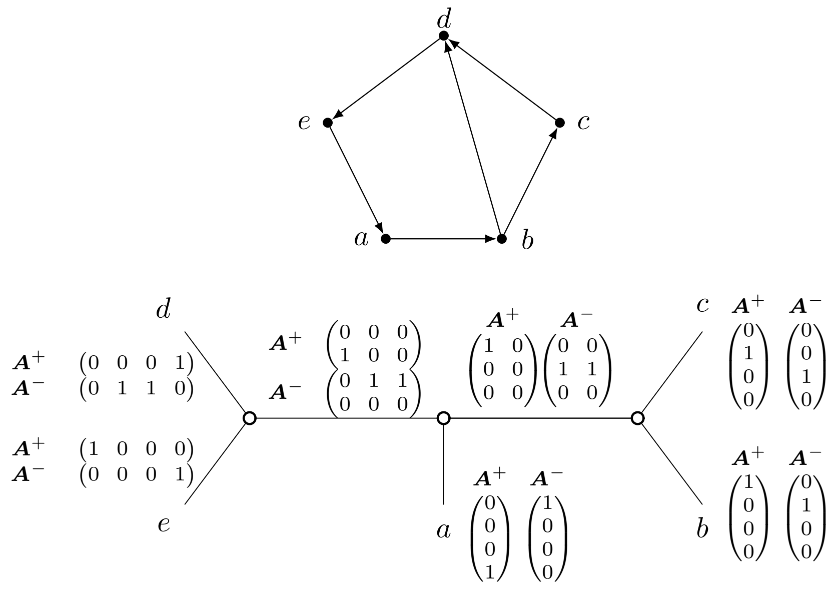

d
e
c
a
b
d
e
a
c
b
$\boldsymbol{A}^+$ $(0\ 0\ 0\ 1)$
$\boldsymbol{A}^-$ $(0\ 1\ 1\ 0)$
$\boldsymbol{A}^+$ $(1\ 0\ 0\ 0)$
$\boldsymbol{A}^-$ $(0\ 0\ 0\ 1)$
$\boldsymbol{A}^+$ $\begin{pmatrix}0&0&0\\1&0&0\end{pmatrix}$
$\boldsymbol{A}^-$ $\begin{pmatrix}0&1&1\\0&0&0\end{pmatrix}$
$\boldsymbol{A}^+$ $\begin{pmatrix}1&0\\0&0\\0&0\end{pmatrix}$
$\boldsymbol{A}^-$ $\begin{pmatrix}0&0\\1&1\\0&0\end{pmatrix}$
$\boldsymbol{A}^+$ $\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$
$\boldsymbol{A}^-$ $\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$
$\boldsymbol{A}^+$ $\begin{pmatrix}0\\1\\0\\0\end{pmatrix}$
$\boldsymbol{A}^-$ $\begin{pmatrix}0\\0\\1\\0\end{pmatrix}$
$\boldsymbol{A}^+$ $\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$
$\boldsymbol{A}^-$ $\begin{pmatrix}0\\1\\0\\0\end{pmatrix}$

Figure 5: A bi-rank-decomposition of a sample graph.

Importantly, Kanté [23] also proved that the rank-decomposition algorithm of Theorem 2.1 can be used to find an optimal bi-rank-decomposition of a digraph:

Theorem 5.2 ([23])**.** *For every parameter t there is an $O(n^3)$-time FPT algorithm that, for a given n-vertex digraph G, either finds a bi-rank-decomposition of G of width at most t, or confirms that the bi-rank-width of G is more than t.*

As in the case of ordinary rank-width, bi-rank-decomposition is not so suitable for designing dynamic programming algorithms. For rank-width, we showed the characterization using t-labeling parse trees in Section 3.

We remind the readers that, unlike with undirected tree-width, one cannot use an undirected rank-decomposition (or parse tree) of a digraph G to design a dynamic programming algorithm for a problem referring to the direction of arcs of G. That is because the parse tree produces large bipartite cliques, and one cannot exhaustively process all possible orientations of those.

However, an analogous "dynamic programming friendly" parse-tree view of bi-rank-width exists for digraphs: *Bi-labeling parse trees* (Definition 5.3), first defined by Kanté [23, Section 4.1] as terms over "algebraic operations for bi-rank-width", characterize bi-rank-width of digraphs up to a multiplicative factor 2 (cf. Lemma 5.4).

Bi-labeling parse trees present a natural generalization of labeling parse trees from Definition 3.1, performing actually two join operations at once in every composition node – one join adding the arcs from $\bar{G}_1$ to $\bar{G}_2$, and the other join adding the arcs in the opposite direction: $\bar{G}_2$ to $\bar{G}_1$. To formally capture this framework in one join operation, we hence have to define the join $\otimes$ with a bi-labeled graph on the right hand side (cf. the congruence relations in all the coming proofs).

t-labeled digraphs. A *t-labeled digraph* $\bar{G} = (G, lab)$ for a digraph G and t-relabeling are defined the same way as for ordinary graphs (cf. Section 3). A generalization is a *t-bi-labeled* digraph $\tilde{H} = (H, lab^+, lab^-)$, where each vertex $v \in V(H)$ has two labels $lab^+(v)$ and $lab^-(v)$. The operator $\odot$ is defined the same way as for undirected graphs, t-bi-labeling join and t-bi-labeling composition are defined as follows:

For a t-labeled digraph $\bar{G} = (G, lab)$ and a t-bi-labeled digraph $\tilde{H} = (H, lab^+, lab^-)$, the *$t$-bi-labeling join* $\bar{G} \otimes \tilde{H}$ is defined by taking a disjoint union of G and H, and adding, for $u \in V(G), v \in V(H)$, all arcs (u, v) such that $|lab(u) \cap lab^+(v)|$ is odd, and all arcs (v, u) such that $|lab(u) \cap lab^-(v)|$ is odd. The resulting digraph is unlabeled.

For two t-labeled digraphs $\bar{G}_i = (G_i, lab_i)$, $i = 1, 2$, and four relabelings $f_1, f_2, h^+, h^- : \mathrm{GF}(2)^t \to \mathrm{GF}(2)^t$, we define a *$t$-bi-labeling composition* operator $\otimes[h^+, h^- \mid f_1, f_2]$ as follows:

$$\bar{G}_1 \otimes[h^+, h^- \mid f_1, f_2] \; \bar{G}_2 \; := \; \bar{G}_3 = \big(\bar{G}_1 \otimes (h^+, h^-)(\bar{G}_2), \, lab_3\big)$$

where $(h^+, h^-)(\bar{G}_2)$ denotes the t-bi-labeled digraph $\big(G_2, \, h^+ \circ lab_2, \, h^- \circ lab_2\big)$ and the resulting labeling is $lab_3(v) = f_i \circ lab_i(v)$ for $v \in V(G_i)$, $i = 1, 2$. Note that we, in effect, add arcs from $\bar{G}_1$ to $h^+(\bar{G}_2)$, and from $h^-(\bar{G}_2)$ to $\bar{G}_1$, relabeling by f_1 and f_2 once all edges are added.

Definition 5.3 (Bi-labeling parse trees)**.** *A t-bi-labeling parse tree T, is a finite rooted ordered subcubic tree (with the root degree at most 2) such that*

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

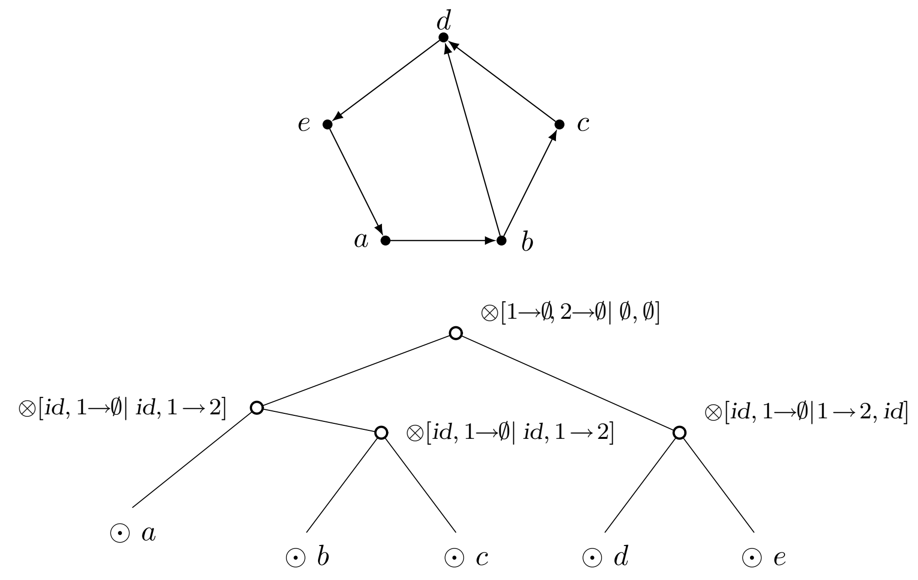

d
e
c
a
b
$\otimes[1\to\emptyset, 2\to\emptyset \mid \emptyset, \emptyset]$
$\otimes[id, 1\to\emptyset \mid id, 1\to 2]$
$\otimes[id, 1\to\emptyset \mid id, 1\to 2]$
$\otimes[id, 1\to\emptyset \mid 1\to 2, id]$
$\odot\ a$
$\odot\ b$
$\odot\ c$
$\odot\ d$
$\odot\ e$

Figure 6: An example of a bi-labeling parse tree with symbolic relabelings at the nodes (*id* denotes the relabeling preserving all labels, and $\emptyset$ is the relabeling "forgetting" all labels).

A parse tree T then generates *(parses) the digraph $\bar{G}$ which is obtained by successive leaves-to-root applications of the operators in the nodes of T.*

This definition is illustrated in Figure 6.

Lemma 5.4 (Kanté [23])**.** *Let G be a digraph of bi-rank-width t. If m is the smallest integer such that (some labeling of) G is produced by some m-bi-labeling parse tree, then $t \geq m \geq t/2$.*

For sake of completeness, we lastly remark that Kanté [23] considers also another directed generalization of rank-width, the so called $GF(4)$-rank-width. Since these two are within a constant factor, there is no need to consider the latter in our paper.

6. XP algorithms for directed problems

Having now the bi-labeling parse tree machinery at hand, it is straightforward to translate the formal tools of the previous sections to digraphs of bounded bi-rank-width, see e.g. the easy proof of Theorem 6.1. Moreover, we are able to provide new XP algorithms for other problems which have not been considered on digraphs of bounded clique-width / bi-rank-width before, such as the unweighted max-directed-cut and the min-leaf outbranching problems.

6.1. Hamiltonian path

This illustrating algorithm is based on the Hamiltonian path (i.e. a path visiting all vertices of the graph) algorithm for graphs of bounded clique-width by Espelage et al [9]. We include it in our paper for two main reasons; first, to show how our unified formalism works on digraphs and is able to capture the design ideas of previously known algorithms, and, second, to explicitly cover the directed variant of Hamiltonian path for a reference (as needed, e.g., in [16]). Moreover, this Theorem 6.1 has a useful generalization in Theorem 6.7.

Before we state the theorem, notice that handling of the Hamiltonian path problem within our formalism brings a new aspect – that we work with a set of edges (instead of vertex sets only). This means that we have to explicitly consider selections of edges "created" by the labeling join operation within our congruence (3), defined in the proof.

Theorem 6.1. *Assume an input digraph G given in the form of a t-bi-labeling parse tree T. Then one can decide whether G has a (directed) Hamiltonian path in time*

$$O\left(|V(G)|^{\ell(t)}\right) \text{ where } \ell(t) = 4^{t+1} + O(1)\,.$$

Proof. We say that a set of arcs $F \subseteq E(G)$ is *linear* if the subgraph $G \restriction F = \big(V(G), F\big)$ is a spanning collection of pairwise disjoint directed paths or isolated vertices. Notice that the restriction of a Hamiltonian path to an induced subgraph is always a linear set. We, moreover, write $G \models \kappa(F)$ to say that F is a directed Hamiltonian path in G.

Let $A(\bar{G}, \tilde{H}) = E(\bar{G} \otimes \tilde{H}) \setminus (E(G) \cup E(H))$ denote the set of arcs which are created, by the join operation (i.e. the set of arcs between G and H). Having t-labeled graphs $\bar{G}_1$ and $\bar{G}_2$, and linear subsets $F_1 \subseteq E(G_1)$ and $F_2 \subseteq E(G_2)$, we define the equivalence relation

$$(\bar{G}_1, F_1) \approx_{\kappa,t} (\bar{G}_2, F_2)$$

if and only if, for all t-bi-labeled digraphs $\tilde{H}$ and all linear $F \subseteq E(H)$, it holds

$$\begin{aligned} & \exists F_3 \subseteq A(\bar{G}_1, \tilde{H}) : \ \big(\bar{G}_1 \otimes \tilde{H}\big) \models \kappa(F_1 \cup F_3 \cup F) \\ \iff & \exists F_4 \subseteq A(\bar{G}_2, \tilde{H}) : \ \big(\bar{G}_2 \otimes \tilde{H}\big) \models \kappa(F_2 \cup F_4 \cup F)\,. \end{aligned} \tag{3}$$

Again, this definition captures all information necessary to decide which linear subsets of G_1 extend to Hamiltonian paths in any $\bar{G}_1 \otimes \tilde{H}$.

Similarly to [9], for $\bar{G} = (G, lab)$ and a linear subset $F \subseteq E(\bar{G})$, we define a multiset of labeling pairs $\Pi(\bar{G}, F) = \big\{(lab(u), lab(v)) \mid$ there is a component in $G \restriction F$, actually a path, from u to $v\big\}$. (An isolated vertex is a path with the ends $u = v$.)

Claim 6.2. *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 $(\bar{G}_1, F_1) \approx_{\kappa,t} (\bar{G}_2, F_2)$ if $\Pi(\bar{G}_1, F_1) = \Pi(\bar{G}_2, F_2)$.*

To prove Claim 6.2, we fix any $\tilde{H}$ and $F \subseteq E(H)$, and assume $(\bar{G}_1 \otimes \tilde{H}) \models \kappa(F_1 \cup F_3 \cup F)$ for some $F_3 \subseteq A(\bar{G}_1, \tilde{H})$. Since $\Pi(\bar{G}_1, F_1) = \Pi(\bar{G}_2, F_2)$, there is a partial injective mapping $b : V(G_1) \to V(G_2)$ such that b determines a bijection between the set of components of $G_1 \restriction F_1$ and that of $G_2 \restriction F_2$ (actually, between the ends of their components–paths) which preserves labelings of the ends. Hence for every arc $(u,v) \in F_3$ there is either the arc $(b(u),v) \in A(\bar{G}_2,\tilde{H})$ (if $v \in V(H)$) or the arc $(u,b(v)) \in A(\bar{G}_2,\tilde{H})$ (if $u \in V(H)$) by the definition of $\otimes$, and the set $F_4 \subseteq A(\bar{G}_2,\tilde{H})$ of all such arcs then fulfils $(\bar{G}_2 \otimes \tilde{H}) \models \kappa(F_2 \cup F_4 \cup F)$. This finishes the proof of Claim 6.2.

Therefore, our algorithm computes, in the leaves-to-root direction on T, the sets $M_T(z) = \{\Pi(\bar{G}_z, F) \mid F \subseteq E(G_z) \text{ linear}\}$ at the nodes z of T (parsing a t-labeled subdigraph $\bar{G}_z$). It holds that G has a Hamiltonian path with the arc set F if and only if $M_T(r)$ at the root r of T contains a multiset $\Pi(\bar{G}, F)$ of cardinality one.

First, it is easy to compute the set $M_T(x)$ when x is a leaf of T. So let z be a node of T with two sons x and y. The set $M_T(z)$ is computed from the sets $M_T(x)$ and $M_T(y)$ iteratively as outlined below. We start with the set $M_T^0(z)$ which is trivial to compute:

$$M_T^0(z) = \{\Pi(\bar{G}_z, F_x \cup F_y) \mid \Pi(\bar{G}_x, F_x) \in M_T(x),\, \Pi(\bar{G}_y, F_y) \in M_T(y)\}$$

Since both F_x and F_y are linear, so is their union and therefore $\Pi(\bar{G}_z, F_x \cup F_y)$ is defined. In the rest of the algorithm we assume we can distinguish between labellings of $M_T(x)$ and $M_T(y)$. This is easily achieved by using an extra label "$t+1$" for the vertices of $M_T(y)$.

Let $A'(\bar{G}_x, \bar{G}_y) = \{\, (lab_x(u), lab_y(v)) \mid (u,v) \in A(\bar{G}_x, \bar{G}_y)\}$, as a multiset of "symbolic arcs" added by the composition operator at z. This definition gives us a natural bijection $b : A(\bar{G}_x, \bar{G}_y) \to A'(\bar{G}_x, \bar{G}_y)$. Now assume the arcs of $A'(\bar{G}_x, \bar{G}_y)$ are linearly ordered as $(f_1, f_2, \ldots, f_a)$ where $a = |A'(\bar{G}_x, \bar{G}_y)| = |A(\bar{G}_x, \bar{G}_y)|$. Then we put, for $i = 1, \ldots, a$,

$$M_T^i(z) = M_T^{i-1}(z) \cup \qquad (4)$$
$$\big\{\Pi(\bar{G}_z, F \cup \{b^{-1}(f_i)\}) \mid \Pi(\bar{G}_z, F) \in M_T^{i-1}(z) \wedge (F \cup \{b^{-1}(f_i)\}) \text{ is linear}\big\}.$$

Finally, let $M_T(z) = M_T^a(z)$. The following Claim 6.3, which is easy to prove, implies that we can efficiently compute each of the sets $M_T^i(z)$ (4) in an "abstract way", looking just at $M_T(x), M_T(y)$ and $A'(\bar{G}_x, \bar{G}_y)$.

Claim 6.3. *Let $\bar{G}$ be a t-labeled graph, $F \subseteq E(G)$ a linear set and $e, e' \in E(G) \setminus F$, $e = (u,v), e' = (u',v')$ arcs s.t. $lab(u) = lab(u')$, $lab(v) = lab(v')$, and both $F \cup \{e\}$ and $F \cup \{e'\}$ are linear. Then $\Pi(\bar{G}, F \cup \{e\}) = \Pi(\bar{G}, F \cup \{e'\})$.*

Complexity-wise there are $2^{2t} = 4^t$ distinct labeling pairs in $\mathrm{GF}(2)^t$, and so $m(\bar{G}_z, t) \leq |V(G_z)|^{4^t}$ distinct multisets are considered as elements of the set $M_T(z)$. The procedure for computing $M_T(z)$ for non-leaf vertices has at most $m(\bar{G}_z, t)$ distinct final outcomes, but we need the extra label distinguishing between $M_T(x)$ and $M_T(y)$, and so we have to deal with up to $m(\bar{G}_z, t+1)$ distinct partial outcomes as elements of $M_T^i(z)$ (4). Hence the recombination procedure takes time bounded by $\big(m(G,t)^2 + m(G,t+1)\big) \cdot 2^{O(t)} \leq m(G,t)^4 \cdot 2^{O(t)}$, and the total runtime of the algorithm thus is

$$O\big(|V(G)| \cdot m(G,t)^4 \cdot 2^{O(t)}\big) = O\big(|V(G)|^{4^{t+1} + O(1)}\big)\,.$$

∎

6.2. Maximum directed cut problem

Max-cut is an extensively studied NP-hard problem both on graphs and, in its natural directed variant, on digraphs. Formally, in the *maximum directed cut* problem on a given digraph G, the goal is to partition the vertex set $V(G)$ into V_0 and V_1 such that the cardinality of $\{\, (u, v) \in E(G) \mid u \in V_0, v \in V_1 \,\}$ is maximized. This problem is often stated also with edge weights, but we consider only the unweighted (cardinality max-cut) variant in our paper – for reasons explained in Section 7.3.

It is well known that the maximum directed cut problem is NP-hard, and it has been shown to stay NP-hard even on acyclic digraphs [26]. Further parameterized complexity related issues of this problem can be found in [16].

Theorem 6.4. *Assume an input digraph G given in the form of a t-bi-labeling parse tree T. Then one can solve the unweighted maximum directed cut problem on G with an XP algorithm running in time*

$$O\left(|V(G)|^{q(t)}\right) \text{ where } q(t) = 4 \cdot 2^t + O(1)\,.$$

Proof. Recall the set of "new edges" $A(\bar{G}, \tilde{H}) = E(\bar{G} \otimes \tilde{H}) \setminus (E(G) \cup E(H))$ from the previous proof. This time, the "information to capture" is how many arcs from $A(\bar{G}, \tilde{H})$ add to a (partial) solution of max-cut on $\bar{G}$ when joined with an arbitrary $\tilde{H}$. This is formally expressed as follows.

Let $cut(G; X, Y) = \{(u, v) \in E(G) \mid u \in X, v \in Y\}$. Given two t-labeled digraphs $\bar{G}_1, \bar{G}_2$ and $X_i \subseteq V(G_i)$ where $i = 1, 2$ we define an equivalence relation

$$(\bar{G}_1, X_1) \approx_{cut,t} (\bar{G}_2, X_2)$$

if and only if, for all t-bi-labeled digraphs $\tilde{H}$ and all $Y \subseteq V(H)$, it holds

$$\begin{aligned} &\left|cut(\bar{G}_1 \otimes \tilde{H};\ X_1,\ V(H) \setminus Y)\right| + \left|cut(\bar{G}_1 \otimes \tilde{H};\ Y,\ V(G_1) \setminus X_1)\right| = \\ = &\left|cut(\bar{G}_2 \otimes \tilde{H};\ X_2,\ V(H) \setminus Y)\right| + \left|cut(\bar{G}_2 \otimes \tilde{H};\ Y,\ V(G_2) \setminus X_2)\right|\,. \end{aligned}$$

Then we immediately have:

Claim 6.5. *Assume that $Z \subseteq V(G)$ maximizes $|cut(G;\, Z, V(G) \setminus Z)|$ where $G = \bar{G}_1 \otimes \tilde{H}$. Then, setting $Z_1 = Z \cap V(G_1)$, it is $|cut(G_1;\, Z_1, V(G_1) \setminus Z_1)| \geq |cut(G_1;\, X_1, V(G_1) \setminus X_1)|$ for all $X_1 \subseteq V(G_1)$ such that $(\bar{G}_1, X_1) \approx_{cut,t} (\bar{G}_1, Z_1)$.*

Defining $\mathcal{C}(\bar{G}, X) = \big(\{lab(v) \mid v \in X\}, \{lab(v) \mid v \in V(G) \setminus X\}\big)$ as a pair of multisets, we moreover easily conclude:

Claim 6.6. *For any t-labeled graphs $\bar{G}_1, \bar{G}_2$ and any $X_1 \subseteq V(G_1)$ and $X_2 \subseteq V(G_2)$, it holds $(\bar{G}_1, X_1) \approx_{cut,t} (\bar{G}_2, X_2)$ if $\mathcal{C}(\bar{G}_1, X_1) = \mathcal{C}(\bar{G}_2, X_2)$.*

Our algorithm processes the given parse tree T in the leaves to root direction. At every node z of T, parsing a t-labeled subdigraph $\bar{G}_z$, we define $M_T^{cut}(z)$ as the set of the pairs (c, X_0) where $X_0 = X$ is a uniquely chosen set achieving maximum cardinality of $cut(G_z;\, X, V(G_z) \setminus X)$ among all $X \subseteq V(G_z)$ such that $\mathcal{C}(\bar{G}_z, X) = c$. Thanks to Claims 6.5 and 6.6, the set $M_T^{cut}(z)$ can be straightforwardly computed from $M_T^{cut}(x)$ and $M_T^{cut}(y)$ of the sons x, y of z in T. The largest X such that $(c, X) \in M_T^{cut}(r)$ at the root r of T is then an optimal solution to the maximum directed cut problem on G.

The number of distinct possible multiset pairs $\mathcal{C}(\bar{G}, X)$ is at most $|V(G)|^{2^{t+1}}$, which bounds the cardinality of each $M_T^{cut}(z)$. Hence the runtime estimate for our algorithm is $O\big(|V(G)| \cdot |V(G)|^{2^{t+1}\cdot 2} \cdot 2^{O(t)}\big) = O\big(|V(G)|^{4\cdot 2^t + O(1)}\big)$. ∎

6.3. *Min-leaf outbranching*

A (directed) tree T in a digraph G is called an *outbranching* (of G) if T is spanning and has only one source vertex (hence this vertex is the root and all arcs of T are directed from it). A *leaf* of a directed tree is a degree-1 vertex which is a sink. The *min-leaf outbranching* problem asks for an outbranching in the given graph with the least possible number of leaves. Obviously, this is an NP-hard problem already when we ask for an outbranching with one leaf (i.e. a directed Hamiltonian path).

Considering restricted digraph classes, the min-leaf outbranching problem is polynomial on acyclic digraphs [20], but NP-hard already on digraphs of directed path-width one [7]. c-min-leaf outbranching is in XP when parametrized by c (the number of leaves) and the DAG-width [7] of the input digraph. To our knowledge, min-leaf outbranching problems have not been considered on digraphs of bounded bi-rank-width so far.

We note that in [11] it was shown that already deciding Hamiltonicity (i.e. 1-min-leaf outbranching) is a $W[2]$-hard problem when parameterized by rank-width, and so we again "cannot hope" for an FPT algorithm here. Our XP algorithm for this problem widely generalizes the approach taken by Theorem 6.1.

Theorem 6.7. *Assume that an input graph G is given in the form of a t-labeling parse tree T. Then the question whether G has an outbranching with at most c leaves, where c is a fixed parameter (the c-min-leaf outbranching problem), can be solved on G with an XP algorithm running in time*

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

Proof. If F is the edge set of an outbranching in a digraph G and H is an induced subgraph of G, then the restriction $F \cap E(H)$ always gives an *outforest* – a spanning forest in which every connected component has a unique source (root). The core of our algorithm is based on the following simple observation: If U is a component (an out-tree) induced by $F \cap E(H)$, then the number of vertices u of U such that u is a U-leaf or that some edge of $F \setminus E(H)$ starts in u, is at most the number c of the leaves of the outbranching F in G. This observation suggests a way how to "store" necessary information about the components of $F \cap E(H)$ during parse tree processing, as formalized by (5) and Claim 6.8 below.

Writing $G \models \varrho_c(F)$ to express that F is the edge set of an outbranching in G with $\leq c$ leaves, we define (cf. the proof of Theorem 6.1):

$$(\bar{G}_1, F_1) \approx_{\varrho_c,t} (\bar{G}_2, F_2)$$

for t-labeled $\bar{G}_1, \bar{G}_2$ and outforest edge sets $F_1 \subseteq E(G_1)$, $F_2 \subseteq E(G_2)$ if, and only if, for all t-bi-labeled digraphs $\tilde{H}$ and all outforest sets $F \subseteq E(H)$, it holds

$$\begin{aligned} & \exists F_3 \subseteq A(\bar{G}_1, \tilde{H}) : \ \big(\bar{G}_1 \otimes \tilde{H}\big) \models \varrho_c(F_1 \cup F_3 \cup F) \\ \iff & \exists F_4 \subseteq A(\bar{G}_2, \tilde{H}) : \ \big(\bar{G}_2 \otimes \tilde{H}\big) \models \varrho_c(F_2 \cup F_4 \cup F) \,. \end{aligned} \tag{5}$$

Let $\bar{G} = (G, lab)$ be a t-labeled digraph, and F be the edge set of an outforest $\bar{G} \restriction F = \big(V(G), F\big)$. If $\bar{U} = (U, lab)$ is a connected component of $\bar{G} \restriction F$ with the root s and $W = \{w_1, \dots, w_i\} \subseteq V(U)$, $i \leq c$ is a vertex set containing (besides other vertices) all the leaves of U, then we define

$$B(\bar{U}, W) = \big(lab(s), lab(w_1), \dots, lab(w_i)\big) \in \mathrm{GF}(2)^{t \,\times\, \leq (c+1)} \,. \tag{6}$$

The order of $w_1, \dots, w_i$ is irrelevant. Particularly, if $\bar{U}$ consists of a single vertex s, then $B(\bar{U}, \{s\}) = (lab(s), lab(s))$.

For a vertex set $W \subseteq V(G)$, we define $R(\bar{G}, F, W)$ as the multiset of all $B(\bar{U}, W \cap V(U))$ where $\bar{U}$ ranges over the components of $\bar{G} \restriction F$, provided that all these $B(\bar{U}, W \cap V(U))$ are defined. Finally, we define the set

$$\mathcal{R}(\bar{G}, F) := \big\{ R(\bar{G}, F, W) \mid W \subseteq V(G) \text{ such that } R(\bar{G}, F, W) \text{ is defined} \big\} \,, \tag{7}$$

and we claim:

Claim 6.8. *For any t-labeled graphs $\bar{G}_1, \bar{G}_2$ and any outforest sets $F_1 \subseteq E(G_1)$ and $F_2 \subseteq E(G_2)$, it holds $(\bar{G}_1, F_1) \approx_{\varrho_c,t} (\bar{G}_2, F_2)$ if $\mathcal{R}(\bar{G}_1, F_1) = \mathcal{R}(\bar{G}_2, F_2)$.*

To prove Claim 6.8, we fix any $\tilde{H}$ and F, and assume $(\bar{G}_1 \otimes \tilde{H}) \models \varrho_c(F_1 \cup F_3 \cup F)$ for some $F_3 \subseteq A(\bar{G}_1, \tilde{H})$. Let $W_1 \subseteq V(G_1)$ be the set containing all the leaves of $G_1 \restriction F_1$ and all the vertices u of G_1 such that some edge of F_3 starts in u. Then $|W_1 \cap V(U)| \leq c$ for every component U of $G_1 \restriction F_1$, and so $R(\bar{G}_1, F_1, W_1) \in \mathcal{R}(\bar{G}_1, F_1)$. Hence there exists $W_2 \subseteq V(G_2)$ such that $R(\bar{G}_1, F_1, W_1) = R(\bar{G}_2, F_2, W_2) \in \mathcal{R}(\bar{G}_2, F_2)$.

The latter equality, by definition of R, means that there exist bijections b, d between the sets S_1, S_2 of the roots of $G_1 \restriction F_1$ and $G_2 \restriction F_2$, and between W_1 and W_2, respectively, which satisfy: For a component root $s \in S_1$, it is $lab_1(s) = lab_2(b(s))$, and for $w \in W_1$ it is $lab_1(w) = lab_2(d(w))$. Moreover, these bijections preserve components, meaning that w belongs to the $(G_1 \restriction F_1)$-component with the root s iff $d(w)$ belongs to the $(G_2 \restriction F_2)$-component with the root $b(s)$. Thus each arc $(v, u) \in F_3$, $u \in S_1$ has a corresponding arc $(v, b(u)) \in A(\bar{G}_2, \tilde{H})$ by the definition of $\otimes$. Similarly every arc $(u, v) \in F_3$,

$u \in W_1$ has corresponding $(d(u), v) \in A(\bar{G}_2, \tilde{H})$. Let F_4 be the set of all these corresponding arcs. Then $F_2 \cup F_4 \cup F$ defines an outbranching in $\bar{G}_2 \otimes \tilde{H}$ with a set of leaves $S_4 \subseteq W_2 \cup V(H)$. Let also S_3 be the set of leaves of the outbranching $F_2 \cup F_3 \cup F$ in $\bar{G}_1 \otimes \tilde{H}$. Clearly, $S_3 \cap V(H) = S_4 \cap V(H)$, and for each $u \in S_4 \cap W_2$ we have $d^{-1}(u) \in W_1 \cap S_3$ since no edge of F_3 starts in this $d^{-1}(u)$. Hence $|S_4| \leq |S_3|$, which proves Claim 6.8.

The algorithm then, in the leaves-to-root direction on T, recursively computes the sets $M_T(z) = \bigcup\{\mathcal{R}(\bar{G}_z, F) \mid F \subseteq E(G_z) \text{ outforest}\}$ at the nodes z of T. In the root r of T, we get that G has an outbranching with arc set F and leaf set W iff $M_T(r)$ contains an element $R(\bar{G}, F, W)$ of cardinality one. It is trivial to compute the set $M_T(z)$ when z is a leaf of T.

So let z be a node of T with two sons x and z. To compute the set $M_T(z)$ from the sets $M_T(x)$ and $M_T(y)$ we proceed analogously to Theorem 6.1 and Claim 6.3. We start with computing

$$M_T^0(z) = \big\{R(\bar{G}_z, F_x \cup F_y, W_x \cup W_y) = R(\bar{G}_x, F_x, W_x) \cup R(\bar{G}_y, F_y, W_y) \mid \\ R(\bar{G}_x, F_x, W_x) \in M_T(x),\ R(\bar{G}_y, F_y, W_y) \in M_T(y)\,\big\}$$

which is always well-founded. For the rest, we again assume that the vertex labelings coming from elements of $M_T(y)$ are distinguished from those of $M_T(x)$, which can be implemented easily by using an extra label "$t+1$". Once again, let $A'(\bar{G}_x, \bar{G}_y) = \{\,(lab_x(u), lab_y(v)) \mid (u,v) \in A(\bar{G}_x, \bar{G}_y)\}$ be the multiset of "symbolic arcs" added by the composition operator at z, linearly ordered as $(f_1, f_2, \dots, f_a) = A'(\bar{G}_x, \bar{G}_y)$, and $b : A(\bar{G}_x, \bar{G}_y) \to A'(\bar{G}_x, \bar{G}_y)$ be a natural bijection. Let $b^{-1}(f_i) = (u_i, v_i)$. We further put, for $i = 1, \dots, a$,

$$M_T^i(z) = M_T^{i-1}(z) \cup \bigcup_{X=W,\ W\setminus\{u_i\}} \big\{R(\bar{G}_z, F \cup \{b^{-1}(f_i)\}, X) \mid \\ R(\bar{G}_z, F, W) \in M_T^{i-1}(z) \wedge u_i \in W \wedge \\ R(\bar{G}_z, F \cup \{b^{-1}(f_i)\}, X) \text{ is defined}\big\}. \tag{8}$$

We remind the readers that the multiset $R(\bar{G}_z, F \cup \{b^{-1}(f_i)\}, X)$ is defined iff $F \cup \{b^{-1}(f_i)\}$ is an outforest set (i.e. u_i, v_i belong to distinct components of $\bar{G}_z \restriction F$ and v_i is a root (source) of $\bar{G}_z \restriction F$), and all the sets $B(\bar{U}, X \cap V(U))$ are defined, where $\bar{U}$ ranges over the components of $\bar{G}_z \restriction F$.

In a straightforward analogy to Claim 6.3, this whole computation of $M_T(z) = M_T^a(z)$ (8) can be carried out from the knowledge of only $M_T(x)$, $M_T(y)$ and $A'(\bar{G}_x, \bar{G}_y)$. The number of possible distinct $B(\bar{U}, W)$ (6) is $2^{2t} + 2^{3t} + \cdots + 2^{(c+1)t} \leq 2^{(c+1)t+1}$. Hence the number of elements in every $M_T(z)$ is bounded by $m(G,c,t) \leq |V(G)|^{2^{(c+1)t+1}}$. Due to the extra label $t+1$, however, the number of possible distinct partial outcomes (elements) of $M_T^i(z)$ (8) is up to $m(G,c,t+1) \leq |V(G)|^{2^{(c+1)(t+1)+1}}$. The overall runtime estimate for our algorithm comes out:

$$O\big(|V(G)| \cdot 2^{O(ct)}\big(m(G,c,t)^2 + m(G,c,t+1)\big)\big) \leq\ O\big(|V(G)|^{2^{(c+1)(t+1)+1} + O(1)}\big)$$

∎

7. Concluding Notes

The list of algorithms presented in this article is by no means exhaustive. Other XP algorithms designed for graphs of bounded clique-width (e.g., for the edge-dominating set [24]) can similarly be translated into our parse tree approach for rank-width. One can expect that the time complexity of such algorithms will have a "one-level higher" exponent, as we see for instance in Theorem 6.1. 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, we have presented several nontrivial examples in which our unified formal approach gives new algorithms, or algorithms with an improved runtime bound. Determining which XP algorithms originally designed for clique-width can be radically improved using our approach remains an open question, one that we believe deserves further study. Finally, 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 and Theorem 5.2.

There are also a few more general considerations related to our presented algorithms which we would like to discuss briefly in the remainder of the paper.

Looking back at our aspiration for a unified formal approach to designing XP algorithms on graphs of bounded rank-width, the reader would likely ask; why did not we come up with one general framework including all the presented algorithmic problems? Say, one similar to the so called LinEMSOL framework of Arnborg et al [1] and Courcelle et al [6]. The truth is that this task does not seem easy at all. The diverse existing XP algorithms, even considering just those from our paper, have quite different setups — as witnessed for instance by the very different definitions of the congruence relations in our proofs here.

Nevertheless, we believe such a general "XP framework for width measures" might exist and we propose the unified design approach presented in our paper as a promising step towards finding it, which is one of the main targets of our future research. In particular, we briefly argue that our approach includes two existing specialized frameworks for XP algorithms parameterized by width measures.

7.1. On MSOL-partitioning problems

The *monadic second order logic* (MSOL) of one-sorted adjacency graphs, commonly abbreviated as MS_1, has variables for graph vertices (say $x, y, z \dots$) and for vertex sets ($X, Y, Z \dots$), common logic connectives and quantifiers. For an illustration, we can express that X is a dominating set in G as

$$G \models \forall y \left(y \in X \vee \exists z \left(z \in X \wedge \mathit{edge}(y,z) \right) \right).$$

See, e.g., [6, 15].

A problem is called MSOL-partitioning [28] if there exists an MSOL formula ϕ such that the problem can be (equivalently) formulated as follows: Find all integers s such that the vertex set of an input graph G can be partitioned into nonempty sets $X_1 \cup \cdots \cup X_s = V(G)$ where $G \models \phi(X_i)$ for $i = 1, \ldots, s$. A typical example of MSOL-partitioning problems is computing the chromatic number, and we have already shown how to compute it on graphs of bounded rank-width.

Moreover, as shown by the following theorem, a straightforward generalization of our Theorem 4.1 can solve any MSOL-partitioning problem, giving an alternative combinatorial proof of the main result of [28].

Theorem 7.1 (Rao [28])**.** *Let $\mathcal{P}$ be an MSOL-partitioning problem described by a formula $\phi(X)$. Then $\mathcal{P}$ can be solved on a graph (digraph) G by an XP algorithm with respect to the parameters ϕ, and t – the rank-width (bi-rank-width) of G.*

Proof. *(sketch)* For any property $\phi(X)$ there is a natural "congruence" relation, called the *canonical equivalence* in [15], defined as follows. Let $(\bar{G}, W)$ denote the t-labeled graph $\bar{G}$ equipped with a partial interpretation $W \subseteq V(G)$ of the variable X. Then we put $(\bar{G}_1, W_1) \approx_{\phi,t} (\bar{G}_2, W_2)$ iff, for all t-labeled graphs $\bar{H}$ equipped with $W_0 \subseteq V(H)$, it holds that

$$\bar{G}_1 \otimes \bar{H} \models \phi(W_1 \cup W_0) \quad \Longleftrightarrow \quad \bar{G}_2 \otimes \bar{H} \models \phi(W_2 \cup W_0)\,. \tag{9}$$

The crucial result of [15, Theorem 4.2] (implicitly following also from earlier work, e.g. [6]) claims that $\approx_{\phi,t}$ has finitely many classes over the universe of all t-labeled graphs equipped with a partial interpretation of X. Moreover, [15, Theorem 4.2] can easily be extended to t-labeled digraphs.

For the rest, we closely follow the proof of Theorem 4.1. Let the input (di)graph $\bar{G}$ be given by a t-labeling parse tree T (cf. Theorem 3.2). Having an ordered vertex partition $\mathcal{N}$ of $\bar{G}$, we define a multiset

$$\Gamma^{\phi}(\bar{G}, \mathcal{N}) \;:=\; \left\{ \mathcal{C} \text{ a class of } \approx_{\phi,t} \mid \emptyset \neq W \in \mathcal{N} \text{ and } (\bar{G}, W) \in \mathcal{C} \right\}.$$

As one can prove analogously to Claim 4.5, a dynamic programming algorithm solving our MSOL-partitioning problem $\mathcal{P}$ along T needs to remember, at a node z of T, only the set $M_T(z)$ of those multisets $\Gamma^{\phi}(\bar{G}, \mathcal{N})$ that result from some ordered vertex partition $\mathcal{N}$ of G_z. Furthermore, the sets $M_T(z)$ can be computed in XP time from the leaves to the root of T in a way which naturally combines Claim 4.6 and the transition function of a tree automaton associated with the classes of $\approx_{\phi,t}$.

Finally, at the root r of T, the solution to our problem $\mathcal{P}$ is the set of those integers s such that $s = |\Gamma_0^{\phi}|$ for some $\Gamma_0^{\phi} \in M_T(r)$ satisfying the following: For each $\approx_{\phi,t}$-class $\mathcal{C}_0 \in \Gamma_0^{\phi}$ and $(\bar{G}, W_0) \in \mathcal{C}_0$, it is $G \models \phi(W_0)$ (which is consistently defined thanks to (9) for $\bar{H} = \emptyset$). ∎

Our alternative proof has one noticeable advantage over original Rao's proof [28] (which was based on formal logic tools) — we can give for any particular MSOL-partitioning problem a fine runtime estimate based just on the number of classes of $\approx_{\phi,t}$.

Furthermore, an analogous generalization of Theorem 4.7 shows that we can even count the solutions to an MSOL-partitioning problem in XP time.

7.2. Max-degree optimization problems

Another short note relates to Theorem 4.8. Kolman et al [25], observing that some natural graph optimization problems do not fit into previously known formalisms (for graphs of bounded tree-width in their case), came with the following approach. An MSOL formula of two-sorted incidence graphs, abbreviated as MS_2 formula, has variables for graph vertices, edges, and their sets. (The expressive power of MS_2 is stronger than that of previously discussed MS_1.) Let, for a graph G and $F \subseteq E(G)$, the operator $\Delta_G(F)$ returns the maximum degree of the subgraph $(V(G), F)$.

Theorem 7.2 (Kolman, Lidický, and Sereni [25]). *Let $\psi(F)$ be an MS_2 formula. Then the problem*

$$\min_{F\subseteq E(G)} \ \Delta_G(F) \ : \ G \models \psi(F)$$

is solvable by an XP algorithm on graphs of bounded tree-width.

Unfortunately, this result cannot be extended to graphs of bounded rank-width in a straightforward manner since MS_1 formulae are not capable of dealing with edge sets. We can, however, define an operator $\Delta_G(W_1, \dots, W_c)$ which, for $W_1, \dots, W_c \subseteq V(G)$, equals to the maximum degree over the collection of induced subgraphs $G \restriction W_1, \dots, G \restriction W_c$. Then, extending Theorem 4.8, we can prove:

Theorem 7.3. *Let $\psi(X_1, \dots, X_c)$ be an MS_1 formula. Then the problem*

$$\min_{X_1,\dots,X_c\subseteq V(G)} \ \Delta_G(X_1, \dots, X_c) \ : \ G \models \psi(X_1, \dots, X_c)$$

is solvable by an XP algorithm on graphs of bounded rank-width.

Proof. *(sketch)* For simplicity we consider a decision version, i.e. the problem to decide, for an arbitrary integer q, whether there exist $W_1, \dots, W_c \subseteq V(G)$ such that $G \models \psi(W_1, \dots, W_c)$ and $\Delta_G(W_1, \dots, W_c) \le q$. We write shortly $\mathcal{X} : \{X_1, \dots, X_c\} \to 2^{V(G)}$ for a partial interpretation of the set variables X_i in G. As in the proof of Theorem 7.1, we use the claim that the canonical equivalence $\approx_{\psi,t}$ has finitely many classes over all $\mathcal{X}$-equipped t-labeled graphs by [15, Theorem 4.2].

Besides a novel use of the previous claim, the whole proof proceeds in the same way as that of Theorem 4.8. We start from a parse tree T for t-labeled $\bar{G}$. The novel idea is that we "separately process" the partial solutions belonging to each one of the (fixed number of) classes of $\approx_{\psi,t}$. Specifically, we define $\Theta(\bar{G}, \mathcal{X}) = \big(D(\bar{G}, \mathcal{X}(X_i)) \mid i = 1, \dots, c\big)$ as in (2). Being at a node z of T (parsing an induced subgraph $\bar{G}_z$), we define a sequence of sets $M'_T(z, \mathcal{C})$ for each of the classes $\mathcal{C}$ of $\approx_{\psi,t}$, where $M'_T(z, \mathcal{C})$ consists of all $\Theta(\bar{G}_z, \mathcal{X})$ such that the variable interpretation $\mathcal{X}$ gives $(\bar{G}_z, \mathcal{X}) \in \mathcal{C}$ and $\Delta_{G_z}(\mathcal{X}(X_1), \dots, \mathcal{X}(X_c)) \le q$.

It again follows from Claim 4.9 and [15, Theorem 4.2] that the sets $M'_T(z, \mathcal{C})$ can be computed recursively from the leaves of T to the root, and that the result of the computation is correct. Namely, at the root r of T, we simply check whether some of the sets $M'_T(r,\mathcal{C})$ for an "ψ-accepting" class $\mathcal{C}$ of $\approx_{\psi,t}$ is nonempty. Since the number of elements of $M'_T(z,\mathcal{C})$ is at most $|V(G)|^{2c\cdot 2^t}$, and the number of classes $\mathcal{C}$ is independent of $|V(G)|$, the whole algorithm runs in time $|V(G)|^{p(c,\psi,t)}$ for some function p (depending mainly on the number of classes of $\approx_{\psi,t}$). ∎

7.3. Rank-width and edge-weighted graphs

Our last remark concerns optimization problems on edge-weighted graphs. An example is the weighted version of maximum directed cut (Section 6.2).

With tree-width, it is often the case that whenever an unweighted version of a problem has an efficient solution on graphs of bounded tree-width, so does the corresponding weighted version. On the other hand, this is not the typical case with edge-weighted problems on graphs of bounded rank-width. An informal explanation might be that, even on a complete graph of rank-width 1, edge weights can "model" arbitrarily complex subgraphs (for which no efficient solution is likely). That is why, for instance, no efficient solution to weighted maximum directed cut likely exists on all graphs of bounded rank-width (compared to Theorem 6.4).

If one aims at designing parameterized algorithms for optimization problems on edge-weighted graphs G, the following idea might be a good starting point: Let $\boldsymbol{B}_G[U,W]$ be the weighted adjacency matrix of a bipartition (U,W) of $V(G)$, i.e. the matrix whose entries are the edge weights, or 0. The weighted cut-rank function then equals the rank of $\boldsymbol{B}_G[U,W]$ over the reals, and the weighted rank-width is the branch-width of it. Then, for instance, Theorem 6.4 can be extended to computing weighted maximum directed cut on digraphs of bounded weighted bi-rank-width. Again, we believe that this mentioned aspect of rank-width deserves deeper theoretical investigation in the future.

References

[1] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991.

[2] I. Averbouch, B. Godlin, J.A. Makowsky, and U. Rotics. Computing graph polynomials on graphs of bounded clique-width. In *WG'06*, volume 4271 of *LNCS*, pages 191–204. Springer, 2006.

[3] B.-M. Bui-Xuan, J.A. Telle, and M. Vatshelle. H-join and algorithms on graphs of bounded rank-width. submitted, November 2008.

[4] D.G. Corneil and U. Rotics. On the relationship between cliquewidth and treewidth. *SIAM J. Comput.*, 34(4):825–847, 2005.

[5] B. Courcelle and M. Kanté. Graph operations characterizing rank-width and balanced graph expressions. In *WG'07*, volume 4769 of *LNCS*, pages 66–75. Springer, 2007.

[6] B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.*, 33(2):125–150, 2000.

[7] P. Dankelmann, G. Gutin, and E. Kim. On complexity of minimum leaf out-branching problem. *Discrete Appl. Math.*, 157(13):3000 – 3004, 2009.

[8] R.G. Downey and M.R. Fellows. *Parameterized complexity.* Monographs in Computer Science. Springer, 1999.

[9] W. Espelage, F. Gurski, and E. Wanke. How to solve NP-hard graph problems on clique-width bounded graphs in polynomial time. In *WG'01*, volume 2204 of *LNCS*, pages 117–128. Springer, 2001.

[10] M.R. Fellows, F.A. Rosamond, U. Rotics, and S. Szeider. Clique-width minimization is NP-hard. In *STOC'06*, pages 354–362. ACM Press, 2006.

[11] F. Fomin, P. Golovach, D. Lokshtanov, and S. Saurab. Clique-width: On the price of generality. In *SODA'09*, pages 825–834. SIAM, 2009.

[12] R. Ganian. Automata formalization for graphs of bounded rank-width. Master's thesis, Faculty of Informatics of the Masaryk University, Brno, Czech Republic, 2008.

[13] R. Ganian and P. Hliněný. Automata approach to graphs of bounded rank-width. In *IWOCA'08*, pages 4–15, 2008.

[14] R. Ganian and P. Hliněný. Better polynomial algorithms on graphs of bounded rank-width. In *IWOCA'09*, volume 2874 of *LNCS*, pages 266–277. Springer, 2009.

[15] R. Ganian and P. Hliněný. On parse trees and Myhill–Nerode–type tools for handling graphs of bounded rank-width. *Discrete Appl. Math.*, 2009. To appear.

[16] R. Ganian, P. Hliněný, J. Kneis, A. Langer, J. Obdržálek, and P. Rossmanith. On digraph width measures in parametrized algorithmics. In *IWPEC'09*, volume 5917 of *LNCS*, pages 161–172. Springer, 2009.

[17] M.U. Gerber and D. Kobler. Algorithms for vertex-partitioning problems on graphs with fixed clique-width. *Theoret. Comput. Sci.*, 299(1-3):719–734, 2003.

[18] O. Gimenez, P. Hliněný, and M. Noy. Computing the Tutte polynomial on graphs of bounded clique-width. *SIAM J. Discrete Math.*, 20:932–946, 2006.

[19] J. Goldman and G.-C. Rota. The number of subspaces of a vector space. In W.T. Tutte, editor, *Recent Progress in Combinatorics*, pages 75–83. Academic Press, 1969.

[20] G. Gutin, I. Razgon, and E.J. Kim. Minimum leaf out-branching and related problems. *Theoret. Comput. Sci.*, 410(45):4571–4579, 2009.

[21] P. Hliněný. Branch-width, parse trees, and monadic second-order logic for matroids. *J. Combin. Theory Ser. B*, 96(3):325–351, 2006.

[22] P. Hliněný and S. Oum. Finding branch-decomposition and rank-decomposition. *SIAM J. Comput.*, 38:1012–1032, 2008.

[23] M. Kanté. The rank-width of directed graphs. arXiv:0709.1433v3, March 2008.

[24] D. Kobler and U. Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Appl. Math.*, 126(2-3):197–221, 2003.

[25] P. Kolman, B. Lidický, and J.-S. Sereni. A framework for fair optimization problems on tree-decomposable graphs. Manuscript.

[26] M. Lampis, G. Kaouri, and V. Mitsou. On the algorithmic effectiveness of digraph decompositions and complexity measures. In *ISAAC*, volume 5369 of *LNCS*, pages 220–231. Springer, 2008.

[27] S. Oum and P. Seymour. Approximating clique-width and branch-width. *J. Combin. Theory Ser. B*, 96(4):514–528, 2006.

[28] M. Rao. MSOL partitioning problems on graphs of bounded treewidth and clique-width. *Theoret. Comput. Sci.*, 377:260–267, 2007.