Computing the chromatic number using graph decompositions via matrix rank

Computing the Chromatic Number Using Graph Decompositions via Matrix Rank

Bart M. P. Jansen[1]
Eindhoven University of Technology, Eindhoven, The Netherlands
b.m.p.jansen@tue.nl

Jesper Nederlof[2]
Eindhoven University of Technology, Eindhoven, The Netherlands
j.nederlof@tue.nl

Abstract

Computing the smallest number q such that the vertices of a given graph can be properly q-colored is one of the oldest and most fundamental problems in combinatorial optimization. The q-Coloring problem has been studied intensively using the framework of parameterized algorithmics, resulting in a very good understanding of the best-possible algorithms for several parameterizations based on the structure of the graph. For example, algorithms are known to solve the problem on graphs of treewidth tw in time $\mathcal{O}^*(q^{\mathrm{tw}})$, while a running time of $\mathcal{O}^*((q-\varepsilon)^{\mathrm{tw}})$ is impossible assuming the Strong Exponential Time Hypothesis (SETH). While there is an abundance of work for parameterizations based on decompositions of the graph by *vertex separators*, almost nothing is known about parameterizations based on *edge separators*. We fill this gap by studying q-Coloring parameterized by cutwidth, and parameterized by pathwidth in bounded-degree graphs. Our research uncovers interesting new ways to exploit small edge separators.

We present two algorithms for q-Coloring parameterized by cutwidth ctw: a deterministic one that runs in time $\mathcal{O}^*(2^{\omega\cdot\mathrm{ctw}})$, where ω is the matrix multiplication constant, and a randomized one with runtime $\mathcal{O}^*(2^{\mathrm{ctw}})$. In sharp contrast to earlier work, the running time is *independent* of q. The dependence on cutwidth is optimal: we prove that even 3-Coloring cannot be solved in $\mathcal{O}^*((2-\varepsilon)^{\mathrm{ctw}})$ time assuming SETH. Our algorithms rely on a new rank bound for a matrix that describes compatible colorings. Combined with a simple communication protocol for evaluating a product of two polynomials, this also yields an $\mathcal{O}^*((\lfloor d/2\rfloor+1)^{\mathrm{pw}})$ time randomized algorithm for q-Coloring on graphs of pathwidth pw and maximum degree d. Such a runtime was first obtained by Björklund, but only for graphs with few proper colorings. We also prove that this result is optimal in the sense that no $\mathcal{O}^*((\lfloor d/2\rfloor+1-\varepsilon)^{\mathrm{pw}})$-time algorithm exists assuming SETH.

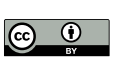
2012 ACM Subject Classification Mathematics of computing → Graph algorithms, Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Parameterized Complexity, Chromatic Number, Graph Decompositions

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.47

Related Version A full version is available at [28], https://arxiv.org/abs/1806.10501.

[1] NWO Veni grant "Frontiers in Parameterized Preprocessing" and NWO Gravitation grant "Networks"
[2] NWO Veni grant "Reducing small instances of complex tasks to large instances of simple ones" and NWO Gravitation grant "Networks"



1 Introduction

Graph coloring is one of the most fundamental combinatorial problems, studied already in the 1850s. Countless papers (cf. [38]) and several monographs [29, 30, 33] have been devoted to its combinatorial and algorithmic investigation. Since the graph coloring problem is NP-complete even in restricted settings such as planar graphs [21], considerable effort has been invested in finding polynomial-time approximation algorithms and exact algorithms that beat brute-force search [5, 6].

A systematic study of which characteristics of inputs govern the complexity of the graph coloring problem has been undertaken using the framework of parameterized algorithmics. The aim in this framework is to obtain algorithms whose running time is of the form $f(k) \cdot n^{\mathcal{O}(1)}$, where k is a parameter that measures the complexity of the instance and is independent of the number of vertices n in the input graph. Over the past decade, numerous parameters have been employed that quantify the structure of the underlying graph. In several settings, algorithms have been obtained that are *optimal* under the Strong Exponential Time Hypothesis (SETH) [25, 26]. For example, it has long been known (cf. [10, Theorem 7.9],[40]) that testing q-colorability on a graph that is provided together with a tree decomposition of width k can be done in time $\mathcal{O}(q^k \cdot k^{\mathcal{O}(1)} \cdot n)$. Lokshtanov, Marx, and Saurabh [34] proved a matching lower bound: an algorithm running in time $(q-\varepsilon)^k \cdot n^{\mathcal{O}(1)}$ for any $\varepsilon > 0$ and integer $q \geq 3$ would contradict SETH. Results are also known for graph coloring parameterized by the vertex cover number [27], pathwidth and the feedback vertex number [34], cliquewidth [17, 24, 31], twin-cover [20], modular-width [19], and split-matching width [39]. (See [16, Fig. 1] for relations between these parameters.)

A survey of these algorithmic results for graph coloring results in the following picture of the complexity landscape: For graph parameters that are defined in terms of the width of decompositions by vertex separators (pathwidth, treewidth, vertex cover number, etc.), one can typically obtain a running time of $\mathcal{O}^*(q^k)$ to test whether a graph that is given together with a decomposition of width k is q-colorable, but assuming (S)ETH there is no algorithm with running time $\mathcal{O}^*(c^k)$ for any constant c independent of q [27, Theorem 11]. (We use $\mathcal{O}^*(f(k))$ as a shorthand for $f(k) \cdot n^{\mathcal{O}(1)}$.)

The complexity of graph coloring parameterized by width measures based on vertex separators is therefore well-understood by now. However, only little attention has been paid to graph decompositions whose width is measured in terms of the number of *edges* in a separator. There is intriguing evidence that separators consisting of few edges (or, equivalently, consisting of a bounded number of bounded-degree vertices) can be algorithmically exploited in nontrivial ways when solving q-Coloring. In 2016, Björklund [4] presented a fascinating algebraic algorithm that decides q-colorability using an algorithmic variation on the Alon-Tarsi theorem [1]. Given a graph G of maximum degree d, a path decomposition of width k, and integers q and s, his algorithm runs in time $(\lfloor d/2 \rfloor + 1)^k n^{\mathcal{O}(1)} \cdot s$. If the graph is not q-colorable it always outputs no. If the graph has at most s proper q-colorings, then it outputs yes with constant probability. Hence when $q \geq (\lfloor d/2 \rfloor + 1)$ and s is small, it improves over the standard $\mathcal{O}^*(q^k)$-time dynamic program by exploiting the bounded-degree vertex separators encoded in the path decomposition. However, the dependence of the running time on the number of proper q-colorings in the graph is very undesirable, as that number may be exponentially large in n.

Björklund's algorithm hints at the fact that graph decompositions whose width is governed by the number of *edges* in a separator may yield an algorithmic advantage over existing approaches. In this work, we therefore perform a deeper investigation of how decompositions

by small edge separators can be exploited when solving q-Coloring. By leveraging a new rank upper bound for a matrix that describes the compatibility of colorings of subgraphs on two sides of a small edge separator, we obtain a number of novel algorithmic results. In particular, we show how to eliminate dependence on the number s of proper colorings.

Our results. We present efficient algorithms for q-Coloring parameterized by the width of various types of graph decompositions by small edge separators. Our first results are phrased in terms of the graph parameter *cutwidth*. A decomposition in this case corresponds to a linear ordering of the vertices; the cutwidth of this ordering is given by the maximum number of edges that connect a vertex in a prefix of the ordering to a vertex in the complement (see Section 2 for formal definitions). Cutwidth is one of the classic graph layout parameters (cf. [14]). It takes larger values than treewidth [32], and has been the subject of frequent study [23, 41, 42].

Informally speaking, we prove that interactions of partial solutions on low-cutwidth graphs are much simpler than interactions of partial solutions on low-pathwidth graphs. The rank-based approach developed in earlier work [8, 11, 18] can be used by setting up matrices whose rank determines the complexity of these interactions in low-cutwidth graphs. These are different from the matrices associated to partial solutions in low-pathwidth graphs, and admit better rank bounds. This is exploited by two different algorithms: a deterministic algorithm that employs fast matrix multiplication and therefore has the matrix-multiplication constant ω in its running time, and a faster randomized Monte Carlo algorithm.

▶ **Theorem 1.** *There is a deterministic algorithm that, for any q, solves q-Coloring on a graph G with a given linear layout of cutwidth* ctw *in $\mathcal{O}^*(2^{\omega \cdot \mathrm{ctw}})$ time, where $\omega \leq 2.373$ is the matrix multiplication constant.*

▶ **Theorem 2.** *There is a randomized Monte Carlo algorithm that, for any q, solves q-Coloring on a graph G with a given linear layout of cutwidth* ctw *in $\mathcal{O}^*(2^{\mathrm{ctw}})$ time.*

These results show a striking difference between cutwidth and parameterizations based on vertex separators such as treewidth and vertex cover number: we obtain single-exponential running times where the base of the exponent is *independent* of the number of colors q, which (assuming ETH) is impossible even parameterized by vertex cover [27]. The assumption that a decomposition is given in the input is standard in this line of research [8, 12, 11, 18] and decouples the complexity of *finding* a decomposition from that of *exploiting* a decomposition.

The ideas underlying Theorems 1 and 2 can also be used to eliminate the dependence on the number of proper colorings from Björklund's algorithm. We prove the following theorem:

▶ **Theorem 3.** *There is a randomized Monte Carlo algorithm that, for any q, solves q-Coloring on a graph G with maximum degree d and given path decomposition of width* pw *in $\mathcal{O}^*((\lfloor d/2 \rfloor + 1)^{\mathrm{pw}})$ time.*

Our approach uses the first step of the proof of the Alon-Tarsi theorem (i.e. rewrite the problem into evaluating the graph polynomial) and also relates colorability to certain orientations, but deviates from the previous algorithm otherwise: to evaluate the appropriate graph polynomial we extend a fairly simple communication-efficient protocol to evaluate a product of two polynomials.

We also prove that the randomized algorithms of Theorem 2 and Theorem 3 are conditionally *optimal*, even when restricted to special cases:

▶ **Theorem 4 (★).** *Assuming SETH, there is no $\varepsilon > 0$ such that 3-Coloring on a planar graph G given along with a linear layout of cutwidth* ctw *can be solved in time $\mathcal{O}^*((2-\varepsilon)^{\mathrm{ctw}})$.*

▶ **Theorem 5 (★).** *Let $d \geq 5$ be an odd integer and let $q_d := \lfloor d/2 \rfloor + 1$. Assuming SETH, there is no $\varepsilon > 0$ such that q_d-Coloring on a graph of maximum degree d given along with a path decomposition of pathwidth* pw *can be solved in time $\mathcal{O}^*((\lfloor d/2 \rfloor + 1 - \varepsilon)^{\mathrm{pw}})$.*

These results are obtained by building on the techniques of Lokshtanov et al. [34] that propagate 'partial assignments' throughout graphs of small cutwidth or pathwidth.

Organization

In Section 2 we provide preliminaries. In Section 3 we present algorithms for graph coloring, proving Theorems 1, 2, and 3. In Section 4 we give briefly sketch the main ideas of the proofs of Theorems 4 and 5, showing that our randomized algorithms cannot be improved significantly assuming SETH. Finally, we provide some conclusions in Section 5. Due to space restrictions, proofs for statements marked (★) have been deferred to the full version [28].

2 Preliminaries

We use $\mathbb{N}$ to denote the natural numbers, including 0. For a positive integer n and a set X we use $\binom{X}{n}$ to denote the collection of all subsets of X of size n. The *power set* of X is denoted 2^X. The set $\{1, \ldots, n\}$ is abbreviated as $[n]$. The $\mathcal{O}^*$ notation suppresses polynomial factors in the input size n, such that $\mathcal{O}^*(f(k))$ is shorthand for $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$. All our logarithms have base two. For sets S, T we denote by S^T the set of vectors indexed by elements of T whose entries are from S. If $T = [n]$, we use S^n instead of $S^{[n]}$.

We consider finite, simple, and undirected graphs G, consisting of a vertex set $V(G)$ and edge set $E(G) \subseteq \binom{V(G)}{2}$. The neighbors of a vertex v in G are denoted $N_G(v)$. The closed neighborhood of v is $N_G[v] := N_G(v) \cup \{v\}$. The degree $d(v)$ equals $|N_G(v)|$ and if $X \subseteq E(G)$, then $d_X(v)$ denotes the number of edges of X incident to v. This notation is extended to $d^-(v), d^+(v), d^-_X(v), d^+_X(v)$ for directed graphs in the natural way (e.g. $d^+_X(v)$ denotes the number of w such that $(v, w) \in X$). For a vertex set $S \subseteq V(G)$ the open neighborhood is $N_G(S) := \bigcup_{v \in S} N_G(v) \setminus S$ and the closed neighborhood is $N_G[S] := N_G(S) \cup S$, while $G[S]$ denotes the graph induced by S.

A q-coloring of a graph G is a function $f \colon V(G) \to [q]$. A coloring is *proper* if $f(u) \neq f(v)$ for all edges $\{u, v\} \in E(G)$. For a fixed integer q, the q-Coloring problem asks whether a given graph G has a proper q-coloring. The q-SAT problem asks whether a given Boolean formula, in conjunctive normal form with clauses of size at most q, has a satisfying assignment.

▶ **Strong Exponential Time Hypothesis** ([25, 26]). *For every $\varepsilon > 0$, there is a constant q such that q-SAT on n variables cannot be solved in time $\mathcal{O}^*((2 - \varepsilon)^n)$.*

Cutwidth. For an n-vertex graph G, a *linear layout* of G is a linear ordering of its vertex set, given by a bijection $\pi \colon V(G) \to [n]$. The *cutwidth* of G with respect to the layout π is:

$$\mathrm{ctw}_\pi(G) = \max_{1 \leq i < n} \left| \left\{ \{u, v\} \in E(G) \,\middle|\, \pi(u) \leq i \wedge \pi(v) > i \right\} \right|,$$

and the cutwidth $\mathrm{ctw}(G)$ of a graph G is the minimum cutwidth attained by any linear layout. It is well-known (cf. [7]) that $\mathrm{ctw}(G) \geq \mathrm{pw}(G) \geq \mathrm{tw}(G)$, where the latter denote the pathwidth and treewidth of G, respectively. An intuitive way to think about cutwidth is to consider the vertices as being placed on a horizontal line in the order dictated by the layout π, with edges drawn as x-monotone curves. For any position i we consider the gap between vertex $\pi^{-1}(i)$ and $\pi^{-1}(i+1)$, and count the edges that *cross* the gap by having one endpoint at position at most i and the other at position after i. The cutwidth of a layout is the maximum number of edges crossing any single gap.

Pathwidth and path decompositions. A *path decomposition* of a graph G is a path P in which each node x has an associated set of vertices $B_x \subseteq V(G)$ (called a *bag*) such that $\bigcup_{x \in V(P)} B_x = V(G)$ and the following properties hold:

1. For each edge $\{u, v\} \in E(G)$ there is a node x in P such that $u, v \in B_x$.
2. If $v \in B_x \cap B_y$ then $v \in B_z$ for all nodes z on the (unique) path from x to y in P.

The *width* of P is the size of the largest bag minus one, and the pathwidth of a graph G is the minimum width over all possible path decompositions of G. Since our focus here is on dynamic programming over a path decomposition we only mention in passing that the related notion of treewidth can be defined in the same way, except for letting the nodes of the decomposition form a tree instead of a path.

It is common for the presentation of dynamic-programming algorithms to use path- and tree decompositions that are normalized in order to make the description easier to follow. For an overview of tree decompositions and dynamic programming on tree decompositions see e.g. [9]. Following [12] we use the following path decompositions:

▶ **Definition 6** (Nice Path Decomposition)**.** A *nice path decomposition* is a path decomposition where the underlying path of nodes is ordered from left to right (the predecessor of any node is its left neighbor) and in which each bag is of one of the following types:

- **First (leftmost) bag**: the bag associated with the leftmost node x is empty, $B_x = \emptyset$.
- **Introduce vertex bag**: an internal node x of P with predecessor y such that $B_x = B_y \cup \{v\}$ for some $v \notin B_y$. This bag is said to *introduce* v.
- **Introduce edge bag**: an internal node x of P labeled with an edge $\{u, v\} \in E(G)$ with one predecessor y for which $u, v \in B_x = B_y$. This bag is said to *introduce* $\{u, v\}$.
- **Forget bag**: an internal node x of P with one predecessor y for which $B_x = B_y \setminus \{v\}$ for some $v \in B_y$. This bag is said to *forget* v.
- **Last (rightmost) bag**: the bag associated with the rightmost node x is empty, $B_x = \emptyset$.

It is easy to verify that any given path decomposition of pathwidth pw can be transformed in time $|V(G)| \cdot \text{pw}^{\mathcal{O}(1)}$ into a nice path decomposition without increasing the width. Let $B_1, \ldots, B_\ell$ be a nice path decomposition of G. We say B_i is *before* B_j if $i \leq j$. We denote $V_i = \bigcup_{j=1}^{i} B_i$ and let E_i denote the set of edges introduced in bags before i.

3 Upper bounds for Graph Coloring

In this section we outline algorithms for q-Coloring that run efficiently when given a graph and either a small-cutwidth layout or a good path decomposition on graphs with small maximum degree. We assume the input graph has no isolated vertices, as they are clearly irrelevant. We start by using the 'rank-based approach' as proposed in [8] to obtain deterministic algorithms, and afterward give a randomized algorithm with substantial speedup. In both approaches the idea is to employ dynamic programming to accumulate needed information about the existence of partial solutions, but use linear-algebraic methods to compress this information. Let us remark in passing that our approaches are robust in the sense that they directly extend to generalizations such as q-List Coloring in which for every vertex a set of allowed colors is given.[3]

A key quantity that determines the amount of information needed after compression in general is the rank of a *partial solutions matrix*. This matrix has its rows and columns

[3] In the deterministic approach we simply avoid partial solutions not satisfying these constraints, and in the randomized approach we assign sufficiently large weight to disallowed (vertex,color) combinations.

indexed by partial solutions (which could be defined in various ways) and an entry is 1 (or more generally, non-zero) if the two partial solutions combine to a solution. Previously, this method proved to be highly useful for connectivity problems parameterized by treewidth [8]. For q-Coloring parameterized by treewidth, partial solutions can naturally be defined as partial proper colorings of a subgraph whose boundary is formed by some vertex separator. Two partial colorings combine to a proper complete coloring if and only if the two partial colorings agree on the coloring of the separator. Unfortunately, the rank-based approach is not useful here as the partial solution matrices arising have large rank, as witnessed by induced identity submatrices of dimensions q^{tw}. Indeed, the lower bound under SETH by Lokshtanov, Marx, and Saurabh [34] shows that no algorithm can solve the problem much faster than $\mathcal{O}^*(q^{\mathrm{pw}})$, where pw denotes the pathwidth of the input graph.

Still, this does not exclude much faster running times parameterized by *cutwidth*. In our application of the rank-based approach for q-Coloring of a graph with a given linear layout of cutwidth ctw, the partial solutions are q-colorings of the first i and last $n-i$ vertices in the linear order, and clearly only the colors assigned to vertices incident to the edges going over the cut are relevant. If we let $X = X_i, Y = Y_i$ denote the endpoints of these edges occurring respectively not after and after i, and let $H = H_i$ denote the bipartite graph induced by the cut and these edges, we are set to study the rank of the following partial solutions matrix indexed by $x \in [q]^X$ and $y \in [q]^Y$:

$$M_H[x,y] = \begin{cases} 1, & \text{if } x \cup y \text{ is a proper } q\text{-coloring of } H, \\ 0, & \text{if otherwise.} \end{cases}$$

Here and below, we slightly abuse notation by viewing elements of V^I (i.e. vectors with values in V that are indexed by I) as sets of pairs in $I \times V$; that is, if $x \in V^I$ we also use x to denote the set $\{(i, x_i)\}_{i \in I}$. With this notation in mind, note that $x \cup y$ above can be interpreted as an element of $[q]^{X \cup Y}$ in the natural way as X and Y are disjoint. As the rank of M_H is generally high[4] and depends on q, we instead focus on the matrix M'_H defined by

$$M'_H[x,y] = \prod_{(v,w) \in E(H)} (x_v - y_w), \tag{1}$$

where all edges are directed from X to Y in $E(H)$. The crux is that the support (e.g. the set of non-zero entries) of M'_H equals the support of M_H:

▶ **Lemma 7.** *We have $M'_H[x,y] \neq 0$ if and only if $x \cup y$ is a proper q-coloring of H.*

Proof. If $x_v = y_w$ for some $(v,w) \in E(H)$ then the term $(x_v - y_w)$ is zero, implying the entire product on the right hand-side of (1) is zero. If x and y differ at every coordinate, then $M'_H[x,y]$ is a product of nonzero terms, and therefore non-zero itself. ◀

In Sections 3.1–3.2 this property will allow us to work with M'_H instead of M_H, when combined with the Isolation Lemma or Gaussian-elimination approach; similarly as in previous work [8, 11, 12].[5]

[4] For example, if H is a single edge M_H is the complement of an identity matrix of dimensions $q \times q$.

[5] In the deterministic setting, the observation that one can work with a matrix different from a partial solution matrix but with the same support as the partial solution matrix was already used by Fomin et al. [18] in combination with a matrix factorization by Lovász [36].

3.1 A deterministic algorithm

We first show that M'_H has rank at most $\prod_{v\in X}(d_{E(H)}(v)+1)$ by exhibiting an explicit factorization. Here we use the shorthand $d_W(v)$ for the number of edges in W containing vertex v. For a bipartite graph H with parts X, Y and edges oriented from X to Y, we have:

$$\begin{aligned} M'_H[x,y] &= \prod_{(v,w)\in E(H)} (x_v - y_w) \\ &= \sum_{W\subseteq E(H)} \left(\prod_{v\in X} x_v^{d_W(v)}\right)\left(\prod_{v\in Y}(-y_v)^{d_{E(H)\setminus W}(v)}\right) \\ &= \sum_{(d_v\in\{0,\ldots,d_{E(H)}(v)\})_{v\in X}} \left(\prod_{v\in X} x_v^{d_v}\right)\left(\sum_{\substack{W\subseteq E(H)\\ \forall v\in X: d_W(v)=d_v}} \prod_{v\in Y}(-y_v)^{d_{E(H)\setminus W}(v)}\right), \quad (2) \end{aligned}$$

where the second equality follows by expanding the product and the third equality follows by grouping the summands on the number of edges incident to vertices in W included in X.

Expression (2) provides us with a matrix factorization $M'_H = L_H \cdot R_H$ where L_H is indexed by $x \in [q]^X$ and a sequence $s = (d_v \in \{0,\ldots,d_{E(H)}(v)\})_{v\in X}$ and R_H has columns indexed by $y \in [q]^Y$ (one such factorization sets $L_H[x,s] = \prod_{v\in X} x_v^{s_v}$). As the number of relevant sequences s is bounded by $\prod_{v\in X}(d_{E(H)}(v)+1)$, the factorization implies the claimed rank bound for M'_H.[6] The rank bound allows some partial solutions to be pruned from the dynamic-programming table without changing the answer. The following definition captures correct reduction steps.

▶ **Definition 8.** Fix a bipartite graph H with parts X and Y and let $\mathcal{S} \subseteq [q]^X$ be a set of q-colorings of X. We say $\mathcal{S}' \subseteq [q]^X$ *H-represents* $\mathcal{S}$ if $\mathcal{S}' \subseteq \mathcal{S}$, and for each $y \in [q]^Y$ we have:

$$(\exists x \in \mathcal{S}\colon x\cup y \text{ is a proper coloring of } H) \Leftrightarrow (\exists x' \in \mathcal{S}'\colon x'\cup y \text{ is a proper coloring of } H). \quad (3)$$

Note that the backward direction of (3) is implied by the property that $\mathcal{S}' \subseteq \mathcal{S}$, but we state both for clarity. If H is clear from context it will be omitted. For future reference we record the observation that the transitivity of this relation follows directly from its definition:

▶ **Observation 9.** *Let H be a bipartite graph with parts X and Y, and let $\mathcal{A}, \mathcal{B}, \mathcal{C} \subseteq [q]^X$. If $\mathcal{A}$ represents $\mathcal{B}$ and $\mathcal{B}$ represents $\mathcal{C}$, then $\mathcal{A}$ represents $\mathcal{C}$.*

Given the above matrix factorization, we can directly follow the proof of [8, Theorem 3.7] to get the following result (note that ω denotes the matrix multiplication constant):

▶ **Lemma 10.** *There is an algorithm* `reduce` *that, given a bipartite graph H with parts X, Y and a set $\mathcal{S} \subseteq [q]^X$, outputs in time $\left(\prod_{v\in X}(d_{E(H)}(v)+1)\right)^{\omega-1} \cdot |\mathcal{S}| \cdot \mathrm{poly}(|X|+|Y|)$ a set $\mathcal{S}'$ that represents $\mathcal{S}$ and satisfies $|\mathcal{S}'| \leq \prod_{v\in X}(d_{E(H)}(v)+1)$.*

Proof. The algorithm is as follows: compute explicitly the matrix $L_H[\mathcal{S},\cdot]$ (i.e. the submatrix of L_H induced by all rows in $\mathcal{S}$). As every entry of L_H can be computed in polynomial time, clearly this can be done within the claimed time bound. Subsequently, the algorithm finds a row basis of this matrix and returns that set as $\mathcal{S}'$. As the rank of a matrix is at most its

[6] This construction (first developed in this paper) has subsequently been used by the second author with Bansal et al. [3] in the completely different setting of online algorithms; see [3, Footnote 3].

number of columns, $|\mathcal{S}'| \leq \prod_{v \in X}(d_{E(H)}(v)+1)$. Using [8, Lemma 3.15], this step also runs in the promised running time.

To see that $\mathcal{S}'$ represents $\mathcal{S}$, note that clearly $\mathcal{S}' \subseteq \mathcal{S}$ and thus it remains to prove the forward implication of (3). To this end, suppose that $x \cup y$ is a proper q-coloring of H and $x \in \mathcal{S}$. As $\mathcal{S}'$ is a row basis of L_H, there exist $x^{(1)}, \ldots, x^{(\ell)} \in \mathcal{S}'$ and $\lambda_1, \ldots, \lambda_\ell$ such that

$$M'_H[x,y] = L_H[x,\cdot]R_H[\cdot,y] = \left(\sum_{i=1}^{\ell} \lambda_i L_H[x^{(i)},\cdot]\right) R_H[\cdot,y] = \sum_{i=1}^{\ell} \lambda_i M'_H[x^{(i)},y],$$

where $L_H[x,\cdot]$ and $R_H[\cdot,y]$ denote a row of L_H and column of R_H respectively. As $x \cup y$ is a proper coloring of H, Lemma 7 implies $M'_H[x,y]$ is non-zero. Therefore there must also exist $x^{(i)} \in \mathcal{S}'$ such that $M'_H[x^{(i)},y]$ is non-zero and hence $x^{(i)} \cup y$ is a proper coloring of H. ◀

Equipped with the algorithm `reduce` from Lemma 10 we are ready to present the algorithm for q-Coloring. On a high level, the algorithm uses a naïve dynamic-programming scheme, but by extensive use of the `reduce` procedure we efficiently represent sets of partial solutions and speed up the computation significantly.

First we need to introduce some notation. A vector $x \in V^I$ is *an extension* of a vector $x' \in V^{I'}$ if $I' \subseteq I$ and $x'_i = x_i$ for every $i \in I'$. If $x \in V^I$ and $P \subseteq I$ then the projection $x_{|P}$ is defined as the unique vector in V^P of which x is an extension. Let G be the graph for which we need to decide whether a proper q-coloring exists and fix an ordering $v_1, \ldots, v_n$ of $V(G)$. We denote all edges as directed pairs (v_i, v_j) with $i < j$. For $i = 1, \ldots, n$, define V_i as the i'th prefix of this ordering, C_i as the i'th cut in this ordering, and X_i and Y_i as the left and respectively right endpoints of the edges in this cut, i.e.

$$V_i = \{v_1, \ldots, v_i\}, \qquad C_i = \{(v_l, v_r) \in E(G) : l \leq i < r\},$$
$$X_i = \{v_l \in V(G) : \exists (v_l, v_r) \in C_i \wedge l < r\}, \qquad Y_i = \{v_r \in V(G) : \exists (v_l, v_r) \in C_i \wedge l < r\}.$$

Note that $X_i \subseteq X_{i-1} \cup \{v_i\}$ and $Y_{i-1} \subseteq Y_i \cup \{v_i\}$. We let H_i denote the bipartite graph with parts X_i, Y_i and edge set C_i. For $i = 1, \ldots, n$, let $T[i] \subseteq [q]^{X_i}$ be the set of all q-colorings of the vertices in X_i that can be extended to a proper q-coloring of $G[V_i]$. The following lemma shows that we can continuously work with a table T' that represents a table T:

▶ **Lemma 11.** *If $T'[i-1]$ H_{i-1}-represents $T[i-1]$, then $T'[i]$ H_i-represents $T[i]$, where*

$$T'[i] = \left\{ (x \cup (v_i, c))_{|X_i} : x \in T'[i-1], c \in [q], \big(\forall v \in N(v_i) \cap X_{i-1} \colon x_v \neq c\big) \right\}. \tag{4}$$

Proof. Assuming the hypothesis, we first show that $T'[i] \subseteq T[i]$. Let $x \in T'[i-1]$ and $c \in [q]$ such that $\forall v \in N(v_i) \cap X_{i-1} \colon x_v \neq c$. As $T'[i-1]$ represents $T[i-1]$, we have that $x \in T[i-1]$. By definition of $T[i-1]$, there exists a proper coloring w of $G[V_{i-1}]$ that extends x. Since all $v \in N(v_i) \cap X_{i-1} = N(v_i) \cap V_{i-1}$ satisfy $x_v \neq c$, it follows that $w \cup (v_i, c)$ is a proper coloring of $G[V_i]$, and thus $(x \cup (v_i, c))_{|X_i} \in T[i]$.

Thus, to prove the lemma it remains to show the forward implication of (3). To this end, let $x \in T[i]$ and let $w \in [q]^{V_i}$ be a proper coloring of $G[V_i]$ that extends x. Let $y \in [q]^{Y_i}$ be such that $x \cup y$ is a proper coloring of H_i. As $w_{v_i} \neq w_{v_j}$ for neighbors $v_j \in N(v_i) \cap V_{i-1}$ and $w_{v_i} \neq y_{v_j}$ for $v_j \in N(v_i) \setminus V_i$, it follows that $w \cup y$ extends a proper coloring of H_{i-1}.

Therefore $w_{|X_{i-1}} \cup (y \cup (v_i, w_{v_i}))_{|Y_{i-1}}$ must be a proper coloring of H_{i-1}, and $w_{|X_{i-1}} \in T[i-1]$ as it can be extended to a proper coloring of V_i, and thus also to a proper coloring of V_{i-1}. As $T'[i-1]$ H_{i-1}-represents $T[i-1]$, there exists $x' \in T'[i-1]$ such that $x' \cup (y \cup (v_i, w_{v_i}))_{|Y_{i-1}}$ is a proper coloring of H_{i-1}.

As no neighbor of v_i was assigned color w_{v_i} by y, it follows that $(x' \cup (v_i, w_{v_i})) \cup y$ is an extension of a proper coloring of H_i. As $x' \cup (y \cup (v, w_{v_i}))_{|Y_{i-1}}$ is a proper coloring of H_{i-1}, no neighbors of v_i are assigned color w_{v_i} by x', and by (4) we have that $(x' \cup (v_i, w_{v_i})) \in T'[i]$, as required. ◀

Now we combine Lemma 10 with Lemma 11 to obtain an algorithm to solve q-Coloring.

▶ **Lemma 12.** *q-Coloring can be solved in time $\mathcal{O}^*\left(\left(\max_i \prod_{v \in X_i}(d_{E(H_i)}(v)+1)\right)^{\omega}\right)$.*

Proof. Note $T'[0] = T[0] = \{\emptyset\}$ (where $\emptyset$ is the 0-dimensional vector). Using Lemma 11, we can use (4) for $i = 1, \ldots, n$ to iteratively compute a set $T'[i]$ representing $T[i]$ from a set $T'[i-1]$ representing $T[i-1]$, and replace $T'[i]$ after each step with $\texttt{reduce}(H_i, T'[i])$. By combining Lemma 11 and Observation 9, we may conclude that G has a q-coloring if and only if $T'[n]$ is not empty (that is, it contains a single element which is the empty vector).

The time required for the computation dictated by (4) is clearly $|T'[i]| \cdot \text{poly}(n)$. Since $|T'[i-1]| \leq \max_i \prod_{v \in X_i}(d_{E(H_i)}(v)+1)$, as it is the result of `reduce`, we have that $|T'[i]|$ is bounded by $q \cdot \max_i \prod_{v \in X_i}(d_{E(H_i)}(v)+1)$. Using this upper bound for $T'[i]$, the time of `reduce` will be $\mathcal{O}^*\left(\left(\max_i \prod_{v \in X_i}(d_{E(H_i)}(v)+1)\right)^{\omega}\right)$, which clearly is the bottleneck in the running time. ◀

Theorem 1 now follows directly from this more general statement.

Proof of Theorem 1. If $v_1, \ldots, v_n$ is a layout of cutwidth k, then $|E(H_i)| \leq k$ for every i, and the term $\prod_{v \in X_i}(d_{E(H_i)}(v)+1)$ is upper bounded by 2^k by the AM-GM inequality. Thus the theorem follows from Lemma 12. ◀

3.2 A randomized algorithm

In this section we use an idea similar to the idea from the matrix factorization of the previous section to obtain faster randomized algorithms. Specifically, our main technical result is as follows (recall that E_i denotes the set of edges introduced in bags before B_i).

▶ **Theorem 13.** *There is a Monte Carlo algorithm for q-Coloring that, given a graph G and a nice path decomposition $B_1, \ldots, B_\ell$, runs in time $\mathcal{O}^*(\max_i \prod_{v \in B_i}(\min\{d_{E_i}(v), d(v) - d_{E_i}(v)\}+1))$. The algorithm does not give false-positives and returns the correct answer with high probability.*

Let $V(G) = V = \{v_1, \ldots, v_n\}$ be ordered arbitrarily, and direct every edge $\{v_i, v_j\}$ as (v_i, v_j) with $i < j$. Define the *graph polynomial* f_G as $f_G(x_1, \ldots, x_n) = \prod_{(u,v) \in E(G)}(x_u - x_v)$. This polynomial has been studied intensively (cf. [2, 13, 35]), for example in the context of the Alon-Tarsi theorem [1]. Define $P_G = \sum_{x \in [q]^V} f_G(x)$. Similarly as in Lemma 7 we see that if $P_G \neq 0$ then G has a proper q-coloring, and if G has a unique q-coloring then $P_G \neq 0$ as it is the product of non-zero values. This is useful if the graph is guaranteed to have at most one proper q-coloring. To this end, we use a standard technique based on the Isolation Lemma, which we state now.

▶ **Definition 14.** A function $\omega \colon U \to \mathbb{Z}$ *isolates* a set family $\mathcal{F} \subseteq 2^U$ if there is a unique $S' \in \mathcal{F}$ with $\omega(S') = \min_{S \in \mathcal{F}} \omega(S)$, where $\omega(S') := \sum_{v \in S'} \omega(v)$.

▶ **Lemma 15** (Isolation Lemma, [37])**.** *Let $\mathcal{F} \subseteq 2^U$ be a non-empty set family over universe U. For each $u \in U$, choose a weight $\omega(u) \in \{1, 2, \ldots, W\}$ uniformly and independently at random. Then $\Pr[\omega$ isolates $\mathcal{F}] \geq 1 - |U|/W$.*

We will apply Lemma 15 to isolate the set of proper colorings of G. To this end, we use the set $V(G) \times [q]$ of vertex/color pairs as our universe U, and consider a weight function $\omega\colon V(G) \times [q] \to \mathbb{Z}$.

▶ **Definition 16.** A *q-coloring* of G is a vector $x \in [q]^n$, and it is proper if $x_i \neq x_j$ for every $(i,j) \in E(G)$. The *weight* of x is $\omega(x) = \sum_{i=1}^{n} \omega((i, x_i))$.

Let $\omega\colon V(G) \times [q] \to [2nq]$ be a random weight function, i.e. for every $v \in V(G)$ and $c \in [q]$ we pick an integer from $[2nq]$ uniformly and independently at random. For every integer z we associate a number $P_G(z)$ with G, as follows:

$$P_G(z) = \sum_{\substack{x \in [q]^n \\ \omega(x) = z}} \prod_{(i,j) \in E(G)} (x_i - x_j)\,. \tag{5}$$

If G has no proper q-coloring, then $P_G(z) = 0$ since for every q-coloring x there will be an edge $(i,j) \in E$ for which $x_i = x_j$ and therefore the product in (5) vanishes. We claim that if G has a proper q-coloring, then with probability at least $1/2$ there exists $z \leq 2nq$ such that $P_G(z) \neq 0$, which means we get a correct algorithm with high probability by repeating a polynomial in n number of times. Let $\mathcal{F} = \{\{(i, x_i)\}_{i \in V} : x \text{ is a proper } q\text{-coloring of } G\} \subseteq 2^U$. As $\mathcal{F}$ is non-empty, we may apply Lemma 15 to obtain that ω isolates $\mathcal{F}$ with probability at least $1/2$. Conditioned on this event, there must exist an integer w such that there is exactly one proper q-coloring x of G satisfying $\omega(x) = z$. In this case, x is the only summand in (5) that can have a non-zero contribution. Moreover, as it is a proper coloring, its contribution is a product of non-zero entries and therefore non-zero itself. Thus $P_G(z)$ is non-zero with probability at least $1/2$.

We now continue by showing how to compute $P_G(z)$ for all $z \leq 2nq$ quickly using dynamic programming. Note that by expanding the product in (5) we have:

$$P_G(z) = \sum_{\substack{x \in [q]^n \\ \omega(x) = z}} \sum_{W \subseteq E(G)} \left(\prod_{(u,v) \in W} x_u \right) \left(\prod_{(u,v) \in E(G) \setminus W} -x_v \right). \tag{6}$$

If B_i is a bag of a path decomposition (Section 2), we need to define table entries T_i containing all information about the graph (V_i, E_i) needed to compute $P_G(z)$. Before we describe these table entries we make a small deviation to convey intuition about our approach. Specifically, we may interpret $P_G(z)$ as a polynomial in variables x_v for $v \in B_i$. Now suppose for simplicity that $|B_i| = 1$. Then the amount of information about E_i needed to compute $P_G(z)$ may be studied via a simple communication-complexity game that we now outline.

A One-way Communication Protocol. Alice has a univariate polynomial $P_A(x)$ of degree d_A, and Bob has a univariate polynomial $P_B(x)$ of degree d_B. Both parties know d_A, d_B and an additional integer q. Alice needs to send as few bits as possible to Bob after which Bob needs to output the quantity $\sum_{x \in [q]} P_A(x) P_B(x)$, where $q \in \mathbb{N}$ is known to both.

An easy strategy is that Alice sends the $d_A + 1$ coefficients of her polynomial to Bob. An alternative strategy for Alice is based on partial evaluations, which is useful when $d_B < d_A$. By expanding Bob's polynomial in coefficient form we can rewrite $\sum_{x \in [q]} P_A(x) P_B(x)$ into

$$\sum_{x \in [q]} P_A(x)(c_0 x^0 + c_1 x^1 + \ldots + c_{d_B} x^{d_B}) = c_0 \sum_{x \in [q]} P_A(x) x^0 + \ldots + c_{d_B} \sum_{x \in [q]} P_A(x) x^{d_B},$$

so as second strategy Alice may send the $d_B + 1$ values $\sum_{x \in [q]} P_A(x)x^i$ for $i = 0, \ldots, d^B$. So she can always send at most $\min\{d_A, d_B\} + 1$ integers.

In our setting for defining table entries T_i for evaluating $P_G(z)$, we think of $d_A(v)$ as the number of edges in E_i incident to v and of $d_B(v)$ as the number of edges incident to v not in E_i. Roughly speaking, the running time of Theorem 13 is obtained by defining table entries storing Alice's message, in which she chooses the best of the two strategies independently for every vertex.

Definition of the Table Entries. An *orientation* O of a subset $X \subseteq E(G)$ of edges is a set of directed pairs such that for every $\{u, v\} \in X$, either $(u, v) \in O$ or $(v, u) \in O$. If O is an orientation of X, we also say O *orients* X. The number of *reversals* $\text{rev}(O)$ of O is the number of $(v, u) \in O$ such that u is introduced in a bag before the bag in which v is introduced. An orientation is *even* if its number of reversals is even, and it is *odd* otherwise.

For a fixed path decomposition $B_1, \ldots, B_\ell$ of the input graph G, let $L_i \subseteq B_i$ consist of all vertices in B_i of which at most half of their incident edges are already introduced in B_i or a bag before B_i, and let $R_i = B_i \setminus L_i$. Let l^i be the vector indexed by L_i such that for every $v \in L_i$ the value l^i_v denotes the number of edges incident to v already introduced before or at bag B_i. Similarly, let r^i be the vector indexed by R_i such that for every $v \in R_i$ the value r^i_v denotes the number of edges incident to v introduced *after* bag B_i. So for every i we have $d(v) = l^i_v + r^i_v$.

If $b \in \mathbb{N}^I_{\geq 0}$ is a vector, we denote $\mathcal{P}(b)$ for the set of vectors a in $\mathbb{N}^I_{\geq 0}$ such that $a \preceq b$. Here $a \preceq b$ denotes that $a_v \leq b_v$ for every $v \in I$. For $d \in \mathcal{P}(l^i)$ and $e \in \mathcal{P}(r^i)$, define:

$$T^z_i[d, e] = \sum_{\substack{x \in [q]^{V_i \setminus L_i} \\ \omega(x) = z}} \sum_{\substack{O \text{ orients } E_i \\ \forall u \in L_i : d^+_O(u) = d_u}} (-1)^{\text{rev}(O)} \left(\prod_{u \in V_i \setminus L_i} x_u^{d^+_O(u)} \right) \left(\prod_{u \in R_i} x_u^{e_u} \right). \tag{7}$$

Intuitively, this could be seen as a partial evaluation of $P_G(z)$. Note we sum over all possible $x_v \in [q]$ for $v \in V_i \setminus L_i$, but let the values x_v for $v \in L_i$ be undetermined and store the coefficient in the obtained polynomial of a certain monomial $\prod_{u \in R_i} x_u^{e_u}$. Indeed, it is easily seen that $P_G(z)$ equals $T^z_\ell[\emptyset, \emptyset]$, where $\emptyset$ is the unique 0-dimensional vector. By combining the appropriate recurrence for all values $T^z_i[d, e]$ with dynamic programming, the following lemma is proved in the full version [28].

▶ **Lemma 17 (★).** *All values $T^z_i[d, e]$ can be computed in time $\text{poly}(n) \cdot \sum_{i=1}^{\ell} T_i$, where*

$$T_i = |\mathcal{P}(l^i)| \cdot |\mathcal{P}(r^i)| = \prod_{v \in B_i} (\min\{d_{E_i}(v), d(v) - d_{E_i}(v)\} + 1).$$

Thus $P_G(z)$ can be computed in the time stated in Theorem 13. As discussed, $P_G(z) = 0$ if G has no proper q-coloring. Otherwise, ω isolates the set of proper q-colorings of G with probability at least $1/2$. Conditioned on this event we have $P_G(z) \neq 0$, where z is the weight of the unique minimum-weight q-coloring. Therefore we output YES if $P_G(z) \neq 0$ for some z and obtain the claimed probabilistic guarantee. This concludes the proof of Theorem 13.

As special cases of Theorem 13 we obtain Theorems 2 and 3.

Proof of Theorem 2. Given a linear layout $v_1, \ldots, v_n$ of cutwidth k, define a nice path decomposition in which vertices are introduced in the order of the layout. After v_i is introduced, its incident edges to v_j with $j < i$ are introduced in arbitrary order. Forget v_i directly after the series of edge introductions that introduced its last incident edge.

As $v_1, \ldots, v_n$ has cutwidth at most k, for any bag B_i of this path decomposition the number of edges between V_i and $V \setminus V_i$ is at most k. Together with the edges incident on the most-recently introduced vertex v_j, these k edges are the only edges incident on B_i that are not in E_i. Consider the term $\prod_{v \in B_i} (\min\{d_{E_i}(v), d(v) - d_{E_i}(v)\} + 1)$. Vertex v_j contributes at most one factor n. For the remaining vertices in B_i, the only incident edges not in E_i are those in the cut of size at most k. By the AM-GM inequality, their contribution to the product is maximized when they are all incident to distinct vertices, in which case the algorithm of Theorem 13 runs in time $\mathcal{O}^*(2^k)$. ◀

Proof of Theorem 3. Follows from Theorem 13: $\min\{d_{E_i}(v), d(v) - d_{E_i}(v)\} \leq \lfloor d(v)/2 \rfloor$. ◀

4 Lower Bounds for Graph Coloring

In this section we discuss the main ideas behind our lower bounds, whose proofs are deferred to the full version [28]. We first start with Theorem 4, which rules out algorithms for solving 3-Coloring in time $\mathcal{O}^*((2-\varepsilon)^{\mathrm{ctw}})$, even on *planar graphs*. (We remark that a companion paper [22] was the first to present lower bounds for planar graphs of bounded cutwidth.) The overall approach is based on the framework by Lokshtanov et al. [34]. We prove that an n-variable instance of CNF-SAT can be transformed in polynomial time into an equivalent instance of 3-Coloring on a planar graph G with a linear layout of cutwidth $n + \mathcal{O}(1)$. Consequently, saving ε in the base of the exponent when solving graph coloring would violate SETH. By employing clause-checking gadgets in the form of a path [27], crossover gadgets [21], and a carefully constructed ordering of the graph, we get the desired reduction.

The second lower bound, Theorem 5, rules out algorithms with running time $\mathcal{O}^*((\lfloor d/2 \rfloor + 1 - \varepsilon)^{\mathrm{pw}})$ for solving q-Coloring for $q := \lfloor d/2 \rfloor + 1$ on graphs of maximum degree d and pathwidth pw, for any odd integer $d \geq 5$. The reduction employs *chains of cliques* to propagate assignments throughout a bounded-pathwidth graph. A t-chain of q-cliques is the graph obtained from a sequence of t vertex-disjoint q-cliques by selecting a distinguished *terminal* vertex in each clique and connecting it to the $(q-1)$ non-terminals in the previous clique. Any proper q-coloring of a chain assigns all terminals the same color, and terminals have $2(q-1)$ neighbors in the chain. Therefore, we can propagate a choice with q possibilities throughout a path decomposition. We encode truth assignments to variables of a CNF-SAT instance through colors given to the terminals of such chains. We enforce that the encoded truth assignment satisfies a clause, by enforcing that an assignment that does *not* satisfy the clause, is not the one encoded by the coloring. To check this, we take one terminal from each chain and connect it to a partner on a path gadget that forbids a specific coloring. Hence each vertex on a chain will receive at most one more neighbor, giving a maximum degree of $d := 2(q-1) + 1 = 2q - 1$ to represent a q-Coloring instance. Then solving this q-Coloring instance in $\mathcal{O}^*((\lfloor d/2 \rfloor + 1 - \varepsilon)^{\mathrm{pw}}) = \mathcal{O}^*(((q-1) + 1 - \varepsilon)^{\mathrm{pw}})$ time will contradict SETH for the same reason as in the earlier construction [34] showing the impossibility of $\mathcal{O}^*((q-\varepsilon)^{\mathrm{pw}})$-time algorithms.

5 Conclusion

We showed how graph decompositions using small edge separators can be used to solve q-Coloring. The exponential parts of the running times of our algorithms are independent of q, which is a significant difference compared to algorithms for parameterizations based on vertex separators. The deterministic $\mathcal{O}^*(2^{\omega \cdot \mathrm{ctw}})$ algorithm of Theorem 1 for the cutwidth parameterization follows cleanly from the bound on the rank of the partial solutions matrix.

It may serve as an insightful new illustration of the rank-based approach for dynamic-programming algorithms in the spirit of [8, 11, 12, 18].

One of the main take-away messages from this work from a practical viewpoint is the following. Suppose H is a subgraph of G connected to the remainder of the graph by k edges. Then any set of partial colorings $\mathcal{S}$ of H can be reduced to a subset $\mathcal{S}'$ of size 2^k, with the guarantee that if some coloring in $\mathcal{S}$ could be extended to a proper coloring of G, then this still holds for $\mathcal{S}'$. The reduction can be achieved by an application of Gaussian elimination, which has experimentally been shown to work well for speeding up dynamic programming for other problems [15]. We therefore believe the table-reduction steps presented here may also be useful when solving graph coloring over tree- or path decompositions, and can be applied whenever processing a separator consisting of few edges.

References

1 Noga Alon and Michael Tarsi. Colorings and orientations of graphs. *Combinatorica*, 12(2):125–134, 1992. doi:10.1007/BF01204715.

2 Noga Alon and Michael Tarsi. A note on graph colorings and graph polynomials. *J. Comb. Theory, Ser. B*, 70(1):197–201, 1997. doi:10.1006/jctb.1997.1753.

3 Nikhil Bansal, Marek Eliás, Grigorios Koumoutsos, and Jesper Nederlof. Competitive algorithms for generalized k-server in uniform metrics. In *Proc. 29th SODA*, pages 992–1001, 2018. doi:10.1137/1.9781611975031.64.

4 Andreas Björklund. Coloring graphs having few colorings over path decompositions. In *Proc. 15th SWAT*, volume 53 of *LIPIcs*, pages 13:1–13:9, 2016. doi:10.4230/LIPIcs.SWAT.2016.13.

5 Andreas Björklund and Thore Husfeldt. Exact graph coloring using inclusion-exclusion. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*. Springer, 2008. doi:10.1007/978-0-387-30162-4_134.

6 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009. doi:10.1137/070683933.

7 Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.

8 Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.

9 Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, 2008. doi:10.1093/comjnl/bxm037.

10 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

11 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. In *Proc. 45th STOC*, pages 301–310. ACM, 2013. doi:10.1145/2488608.2488646.

12 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proc. 52nd FOCS*, pages 150–159, 2011. doi:10.1109/FOCS.2011.23.

13 J. A. de Loera. Gröbner bases and graph colorings. *Contributions to Algebra and Geometry*, 35(1):89–96, 1995.

14 Josep Díaz, Jordi Petit, and Maria J. Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, 2002. doi:10.1145/568522.568523.

15 Stefan Fafianie, Hans L. Bodlaender, and Jesper Nederlof. Speeding up dynamic programming with representative sets: An experimental evaluation of algorithms for Steiner tree on tree decompositions. *Algorithmica*, 71(3):636–660, 2015. doi:10.1007/s00453-014-9934-0.

16 Michael R. Fellows, Bart M. P. Jansen, and Frances Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *European J. Combin.*, 34(3):541–566, 2013. doi:10.1016/j.ejc.2012.04.008.

17 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Intractability of clique-width parameterizations. *SIAM J. Comput.*, 39(5):1941–1956, 2010. doi:10.1137/080742270.

18 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Efficient computation of representative families with applications in parameterized and exact algorithms. *J. ACM*, 63(4):29:1–29:60, 2016. doi:10.1145/2886094.

19 Jakub Gajarský, Michael Lampis, and Sebastian Ordyniak. Parameterized algorithms for modular-width. In *Proc. 8th IPEC*, volume 8246 of *Lecture Notes in Computer Science*, pages 163–176. Springer, 2013. doi:10.1007/978-3-319-03898-8_15.

20 Robert Ganian. Twin-cover: Beyond vertex cover in parameterized algorithmics. In Dániel Marx and Peter Rossmanith, editors, *Proc. 6th IPEC*, volume 7112 of *Lecture Notes in Computer Science*, pages 259–271. Springer, 2011. doi:10.1007/978-3-642-28050-4_21.

21 M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976. doi:10.1016/0304-3975(76)90059-1.

22 Bas A.M. van Geffen, Bart M.P. Jansen, Arnoud A.W.M. de Kroon, and Rolf Morel. Lower bounds for dynamic programming on planar graphs of bounded cutwidth. *CoRR*, 2018. arXiv:1806.10513.

23 Archontia C. Giannopoulou, Michal Pilipczuk, Jean-Florent Raymond, Dimitrios M. Thilikos, and Marcin Wrochna. Cutwidth: Obstructions and algorithmic aspects. In *Proc. 11th IPEC*, volume 63 of *LIPIcs*, pages 15:1–15:13, 2016. doi:10.4230/LIPIcs.IPEC.2016.15.

24 Petr A. Golovach, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Cliquewidth III: The odd case of graph coloring parameterized by cliquewidth. In *Proc. 29th SODA*, pages 262–273, 2018. doi:10.1137/1.9781611975031.19.

25 Russel Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

26 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

27 Lars Jaffke and Bart M. P. Jansen. Fine-grained parameterized complexity analysis of graph coloring problems. In *Proc. 10th CIAC*, Lecture Notes in Computer Science, pages 345–356, 2017. doi:10.1007/978-3-319-57586-5_29.

28 Bart M. P. Jansen and Jesper Nederlof. Computing the chromatic number using graph decompositions via matrix rank. *CoRR*, 2018. arXiv:1806.10501.

29 T.R. Jensen and B. Toft. *Graph Coloring Problems*. Wiley interscience publication. Wiley, 1995.

30 David S. Johnson, Anuj Mehrotra, and Michael A. Trick. Special issue on computational methods for graph coloring and its generalizations. *Discrete Applied Mathematics*, 156(2):145–146, 2008. doi:10.1016/j.dam.2007.10.007.

31 Daniel Kobler and Udi Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Applied Mathematics*, 126(2-3):197–221, 2003. doi:10.1016/S0166-218X(02)00198-1.

32 Ephraim Korach and Nir Solel. Tree-width, path-width, and cutwidth. *Discrete Applied Mathematics*, 43(1):97–101, 1993. doi:10.1016/0166-218X(93)90171-J.

33 R.M. R. Lewis. *A Guide to Graph Colouring: Algorithms and Applications*. Springer Publishing Company, 2015.

34 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. In *Proc. 22nd SODA*, pages 777–789, 2011. doi:10.1137/1.9781611973082.61.

35 L. Lovász. Bounding the independence number of a graph. In Achim Bachem, Martin Grötschel, and Bemhard Korte, editors, *Bonn Workshop on Combinatorial Optimization*, volume 66, pages 213–223. North-Holland, 1982. doi:10.1016/S0304-0208(08)72453-8.

36 László Lovász. Flats in matroids and geometric graphs. In *Combinatorial surveys (Proc. Sixth British Combinatorial Conf.)*, pages 45–86. Academic Press London, 1977.

37 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

38 P.M. Pardalos, T. Mavridou, and J. Xue. *The graph coloring problem: A bibliographic survey*, volume 2, pages 331–395. Kluwer Academic Publishers, Boston, 1998.

39 Sigve Hortemo Sæther and Jan Arne Telle. Between treewidth and clique-width. *Algorithmica*, 75(1):218–253, 2016. doi:10.1007/s00453-015-0033-7.

40 Jan Arne Telle and Andrzej Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM J. Discrete Math.*, 10(4):529–550, 1997. doi:10.1137/S0895480194275825.

41 Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth I: A linear time fixed parameter algorithm. *J. Algorithms*, 56(1):1–24, 2005. doi:10.1016/j.jalgor.2004.12.001.

42 Dimitrios M. Thilikos, Maria J. Serna, and Hans L. Bodlaender. Cutwidth II: algorithms for partial w-trees of bounded degree. *J. Algorithms*, 56(1):25–49, 2005. doi:10.1016/j.jalgor.2004.12.003.