Solving Connectivity Problems Parameterized by Treedepth in Single-Exponential Time and Polynomial Space

Falko Hegerfeld
Humboldt-Universität zu Berlin, Germany
hegerfeld@informatik.hu-berlin.de

Stefan Kratsch
Humboldt-Universität zu Berlin, Germany
kratsch@informatik.hu-berlin.de

Abstract

A breakthrough result of Cygan et al. (FOCS 2011) showed that connectivity problems parameterized by treewidth can be solved much faster than the previously best known time $\mathcal{O}^*(2^{\mathcal{O}(\mathrm{tw}\log \mathrm{tw})})$. Using their inspired Cut&Count technique, they obtained $\mathcal{O}^*(\alpha^{\mathrm{tw}})$ time algorithms for many such problems. Moreover, they proved these running times to be optimal assuming the Strong Exponential-Time Hypothesis. Unfortunately, like other dynamic programming algorithms on tree decompositions, these algorithms also require *exponential space*, and this is widely believed to be unavoidable. In contrast, for the slightly larger parameter called treedepth, there are already several examples of matching the time bounds obtained for treewidth, but using only polynomial space. Nevertheless, this has remained open for connectivity problems.

In the present work, we close this knowledge gap by applying the Cut&Count technique to graphs of small treedepth. While the general idea is unchanged, we have to design novel procedures for counting consistently cut solution candidates using only polynomial space. Concretely, we obtain time $\mathcal{O}^*(3^d)$ and polynomial space for Connected Vertex Cover, Feedback Vertex Set, and Steiner Tree on graphs of treedepth d. Similarly, we obtain time $\mathcal{O}^*(4^d)$ and polynomial space for Connected Dominating Set and Connected Odd Cycle Transversal.

2012 ACM Subject Classification Mathematics of computing → Paths and connectivity problems; Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Combinatorial algorithms

Keywords and phrases Parameterized Complexity, Connectivity, Treedepth, Cut&Count, Polynomial Space

Digital Object Identifier 10.4230/LIPIcs.STACS.2020.29

Related Version A full version of the paper is available at [18], `http://arxiv.org/abs/2001.05364`.

1 Introduction

The goal of parameterized complexity is to reign in the combinatorial explosion present in NP-hard problems with the help of a secondary parameter. This leads us to the search for fixed-parameter tractable (FPT) algorithms, i.e., algorithms with running time $\mathcal{O}(f(k)n^c)$ where n is the input size, k is the secondary parameter, f is a computable function, and c is a constant. There are several books giving a broad overview of parameterized complexity [10, 12, 13, 28]. One of the success stories of parameterized complexity is a graph parameter called treewidth. A large swath of graph problems admit FPT-algorithms when parameterized by treewidth as witnessed by, amongst other things, Courcelle's theorem [9]. However, the function f resulting from Courcelle's theorem is non-elementary [16]. Thus, a natural goal is to find algorithms with a smaller, or ideally minimal, dependence on the treewidth in the running time, i.e. algorithms where f is as small as possible. Problems only involving

local constraints usually permit a single-exponential dependence on the treewidth (tw) in the running time, i.e. time $\mathcal{O}^*(\alpha^{\mathrm{tw}})$ for some small constant α,[1] by means of dynamic programming on tree decompositions [1, 31, 32, 33]. For many of these problems we also know the optimal base α if we assume the strong exponential-time hypothesis (SETH) [22]. For a long time a single-exponential running time seemed to be out of reach for problems involving global constraints, in particular for connectivity constraints. This changed when Cygan et al. [11] introduced the Cut&Count technique, which allowed them to obtain single-exponential-time algorithms for many graph problems involving connectivity constraints. Again, many of the resulting running times can be shown to be optimal assuming SETH [11].

The issue with treewidth-based algorithms is that dynamic programming on tree decompositions seems to inherently require exponential space. In particular, Chen et al. [8] devised a model for single-pass dynamic programming algorithms on tree decompositions and showed that such algorithms require exponential space for Vertex Cover and 3-Coloring. Algorithms requiring exponential time and exponential space usually run out of available space before they hit their time limit [34]. Hence, it is desirable to reduce the space requirement while maintaining the running time. As discussed, this seems implausible for treewidth. Instead, we consider a different, but related, parameter called treedepth. Treedepth is a slightly larger parameter than treewidth and of great importance in the theory of sparse graphs [25, 26, 27]. It has been studied under several names such as minimum elimination tree height [7], ordered chromatic number [19], and vertex ranking [6]. Fürer and Yu [17] established an explicit link between treedepth and tree decompositions, namely that treedepth is obtained by minimizing the maximum number of forget nodes in a root-leaf-path over all nice tree decompositions (see [20] for a definition). Many problems parameterized by treedepth allow branching algorithms on *elimination forests*, also called *treedepth decompositions*, that match the running time of the treewidth-algorithms, but replacing the dependence on treewidth by treedepth, while only requiring polynomial space [8, 17, 29].

Our contribution. The Cut&Count technique reduces problems with connectivity constraints to counting problems of certain cuts, called *consistent cuts*. We show that for several connectivity problems the associated problem implied by the Cut&Count technique can be solved in time $\mathcal{O}^*(\alpha^d)$ and polynomial space, where α is a constant and d is the depth of a given elimination forest. Furthermore, the base α matches the base in the running time of the corresponding treewidth-algorithm. Concretely, given an elimination forest of depth d for a graph G we prove the following results:

- Connected Vertex Cover, Feedback Vertex Set, and Steiner Tree can be solved in time $\mathcal{O}^*(3^d)$ and polynomial space.
- Connected Dominating Set and Connected Odd Cycle Transversal can be solved in time $\mathcal{O}^*(4^d)$ and polynomial space.

Related work. The Cut&Count technique leads to randomized algorithms as it relies on the Isolation Lemma. At the cost of a worse base in the running time, Bodlaender et al. [5] present a generic method, called the rank-based approach, to obtain deterministic single-exponential-time algorithms for connectivity problems parameterized by treewidth; the rank-based approach is also able to solve counting variants of several connectivity problems. Fomin et al. [15] use matroid tools to, amongst other results, reobtain the deterministic running times of the rank-based approach. In a follow-up paper, Fomin et al. [14] manage to

[1] The $\mathcal{O}^*$-notation hides polynomial factors in the input size.

improve several of the deterministic running times using their matroid tools. Multiple papers adapt the Cut&Count technique and rank-based approach to graph parameters different from treewidth. Bergougnoux and Kanté [3] apply the rank-based approach to obtain single-exponential-time algorithms for connectivity problems parameterized by cliquewidth. The same authors [4] generalize, incurring a loss in the running time, this approach to a wider range of parameters including rankwidth and mim-width. Pino et al. [30] use the Cut&Count technique and rank-based approach to obtain fast deterministic and randomized algorithms for connectivity problems parameterized by branchwidth.

Lokshtanov and Nederlof [23] present a framework using algebraic techniques, such as Fourier, Möbius, and Zeta transforms, to reduce the space usage of certain dynamic programming algorithms from exponential to polynomial. Fürer and Yu [17] adapt this framework to the setting where the underlying set (or graph) is dynamic instead of static, in particular for performing dynamic programming along the bags of a tree decomposition, and obtain a $\mathcal{O}^*(2^d)$-time, where d is the depth of a given elimination forest, and polynomial-space algorithm for counting perfect matchings. Using the same approach, Belbasi and Fürer [2] design an algorithm counting the number of Hamiltonian cycles in time $\mathcal{O}^*((4k)^d)$, where k is the width and d the depth of a given tree decomposition, and polynomial space. Furthermore, they also present an algorithm for the traveling salesman problem with the same running time, but requiring pseudopolynomial space.

Organization. We describe the preliminary definitions and notations in Section 2. In Section 3 we first discuss the Cut&Count setup and give a detailed exposition for Connected Vertex Cover. Afterwards, we explain what general changes can occur for the other problems and then discuss Feedback Vertex Set; the remaining problems can be found in the full version. We conclude in Section 4. Proofs that are delegated to the full version [18] are denoted by $\star$.

2 Preliminaries

2.1 Notation

Let $G = (V, E)$ be an undirected graph. We denote the number of vertices by n and the number of edges by m. For a vertex set $X \subseteq V$, we denote by $G[X]$ the subgraph of G that is induced by X. The *open neighborhood* of a vertex v is given by $N(v) = \{u \in V \mid \{u, v\} \in E\}$, whereas the *closed neighborhood* is given by $N[v] = N(v) \cup \{v\}$. We extend these notations to sets $X \subseteq V$ by setting $N[X] = \bigcup_{v \in X} N[v]$ and $N(X) = N[X] \setminus X$. Furthermore, we denote by $\mathtt{cc}(G)$ the number of connected components of G.

A *cut* of a set $X \subseteq V$ is a pair (X_L, X_R) with $X_L \cap X_R = \emptyset$ and $X_L \cup X_R = X$, we also use the notation $X = X_L \uplus X_R$. We refer to X_L and X_R as the *left* and *right side* of the cut, respectively. Note that either side may be empty, although usually the left side is nonempty.

For two integers a, b we write $a \equiv b$ to indicate equality modulo 2, i.e., a is even if and only if b is even. We use Iverson's bracket notation: for a predicate p, we have that $[p]$ is 1 if p is true and 0 otherwise. For a function f we denote by $f[v \mapsto \alpha]$ the function $(f \setminus \{(v, f(v))\}) \cup \{(v, \alpha)\}$. By $\mathbb{F}_2$ we denote the field of two elements. For a field or ring $\mathbb{F}$ we denote by $\mathbb{F}[Z_1, Z_2, \ldots, Z_t]$ the ring of polynomials in the indeterminates $Z_1, Z_2, \ldots, Z_t$ with coefficients in $\mathbb{F}$. With $\mathcal{O}^*$ we hide polynomial factors, i.e. $\mathcal{O}^*(f(n)) = \mathcal{O}(f(n)\mathrm{poly}(n))$. For a natural number n, we denote by $[n]$ the set of integers from 1 to n.

2.2 Treedepth

▶ **Definition 2.1.** An *elimination forest* of an undirected graph $G = (V, E)$ is a rooted forest $\mathcal{T} = (V, E_{\mathcal{T}})$ such that for every edge $\{u, v\} \in E$ either u is an ancestor of v in $\mathcal{T}$ or v is an ancestor of u in $\mathcal{T}$. The *depth* of a rooted forest is the largest number of nodes on a path from a root to a leaf. The *treedepth* of G is the minimum depth over all elimination forests of G and is denoted by $\mathrm{td}(G)$.

We slightly extend the notation for elimination forests used by Pilipczuk and Wrochna [29]. For a rooted forest $\mathcal{T} = (V, E_{\mathcal{T}})$ and a node $v \in V$ we denote by $\texttt{tree}[v]$ the set of nodes in the subtree rooted at v, including v. By $\texttt{tail}[v]$ we denote the set of all ancestors of v, including v. Furthermore, we define $\texttt{tree}(v) = \texttt{tree}[v] \setminus \{v\}$, $\texttt{tail}(v) = \texttt{tail}[v] \setminus \{v\}$, and $\texttt{broom}[v] = \{v\} \cup \texttt{tail}(v) \cup \texttt{tree}(v)$. By $\texttt{child}(v)$ we denote the children of v.

Note that an elimination forest $\mathcal{T}$ of a connected graph consists only of a single tree.

2.3 Isolation Lemma

▶ **Definition 2.2.** A function $\mathbf{w} \colon U \to \mathbb{Z}$ *isolates* a set family $\mathcal{F} \subseteq 2^U$ if there is a unique $S' \in \mathcal{F}$ with $\mathbf{w}(S') = \min_{S \in \mathcal{F}} \mathbf{w}(S)$, where for subsets X of U we define $\mathbf{w}(X) = \sum_{u \in X} \mathbf{w}(u)$.

▶ **Lemma 2.3** (Isolation Lemma, [24])**.** *Let $\mathcal{F} \subseteq 2^U$ be a nonempty set family over a universe U. Let $N \in \mathbb{N}$ and for each $u \in U$ choose a weight $\mathbf{w}(u) \in [N]$ uniformly and independently at random. Then $\mathbb{P}[\mathbf{w}$ isolates $\mathcal{F}] \geq 1 - |U|/N$.*

When counting objects modulo 2 the Isolation Lemma allows us to avoid unwanted cancellations by ensuring with high probability that there is a unique solution. In our applications, we will choose N so that we obtain an error probability of less than $1/2$.

3 Cut&Count

In this section $G = (V, E)$ always refers to a connected undirected graph. For the sake of a self-contained presentation, we state the required results, with their proofs in the appendix, for the Cut&Count technique again, mostly following the presentation of Cygan et al. [11]. Our approach only differs from that of Cygan et al. [11] in the counting sub-procedure.

We begin by describing the Cut&Count setup and then present the counting sub-procedure for Connected Vertex Cover. Afterwards we explain how to adapt the counting sub-procedure for the other problems. Our exposition is the most detailed for Connected Vertex Cover, whereas the analogous parts of the other problems will not be discussed in such detail.

3.1 Setup

Suppose that we want to solve a problem on G involving connectivity constraints, then we can make the following general definitions. The solutions to our problem are subsets of a universe U which is related to G. Let $\mathcal{S} \subseteq 2^U$ denote the set of solutions and we want to determine whether $\mathcal{S}$ is empty or not. The Cut&Count technique consists of two parts:

- **The Cut part:** We relax the connectivity constraints to obtain a set $\mathcal{S} \subseteq \mathcal{R} \subseteq 2^U$ of possibly connected solutions. The set $\mathcal{Q}$ will contain pairs (X, C) consisting of a candidate solution $X \in \mathcal{R}$ and a consistent cut C of X, which is defined in Definition 3.1.
- **The Count part:** We compute $|\mathcal{Q}|$ modulo 2 using a sub-procedure. The consistent cuts are defined so that non-connected candidate solutions $X \in \mathcal{R} \setminus \mathcal{S}$ cancel, because they are consistent with an even number of cuts. Hence, only connected candidates $X \in \mathcal{S}$ remain.

If $|\mathcal{S}|$ is even, then this approach does not work, because the connected solutions would cancel out as well when counting modulo 2. To circumvent this difficulty, we employ the Isolation Lemma (Lemma 2.3). By sampling a weight function $\mathbf{w}\colon U \to [N]$, we can instead count pairs with a fixed weight and it is likely that there is a weight with a unique solution if a solution exists at all. Formally, we compute $|\mathcal{Q}_w|$ modulo 2 for every possible weight w, where $\mathcal{Q}_w = \{(X, C) \in \mathcal{Q} \mid \mathbf{w}(X) = w\}$, instead of computing $|\mathcal{Q}|$ modulo 2.

▶ **Definition 3.1** ([11]). A cut (V_L, V_R) of an undirected graph $G = (V, E)$ is *consistent* if $u \in V_L$ and $v \in V_R$ implies $\{u, v\} \notin E$. A *consistently cut subgraph* of G is a pair $(X, (X_L, X_R))$ such that $X \subseteq V$ and (X_L, X_R) is a consistent cut of $G[X]$. For $V' \subseteq V$, we denote the set of consistently cut subgraphs of $G[V']$ by $\mathcal{C}(V')$.

To ensure that connected solutions are not compatible with an even number of consistent cuts, we will usually force a single vertex to the left side of the consistent cut. This results in the following fundamental property of consistent cuts.

▶ **Lemma 3.2** ($\star$, [11]). *Let X be a subset of vertices such that $v_1 \in X \subseteq V$. The number of consistently cut subgraphs $(X, (X_L, X_R))$ such that $v_1 \in X_L$ is equal to $2^{\mathsf{cc}(G[X])-1}$.*

With Lemma 3.2 we can distinguish disconnected candidates from connected candidates by determining the parity of the number of consistent cuts for the respective candidate. We determine this number not for a single candidate but we determine the total for all candidates with a fixed weight. Corollary 3.3 encapsulates the Cut&Count technique for treedepth.

▶ **Corollary 3.3** ($\star$). *Let $\mathcal{S} \subseteq 2^U$ and $\mathcal{Q} \subseteq 2^{U \times (V \times V)}$ such that the following two properties hold for every weight function $\mathbf{w}\colon U \to [2|U|]$ and target weight $w \in \mathbb{N}$:*

1. $|\{(X, C) \in \mathcal{Q} \mid \mathbf{w}(X) = w\}| \equiv |\{X \in \mathcal{S} \mid \mathbf{w}(X) = w\}|$,
2. *There is an algorithm $\texttt{CountC}(\mathbf{w}, w, \mathcal{T})$ accepting weights $\mathbf{w}\colon U \to [N]$, a target weight w, and an elimination forest $\mathcal{T}$, such that $\texttt{CountC}(\mathbf{w}, w, \mathcal{T}) \equiv |\{(X, C) \in \mathcal{Q} \mid \mathbf{w}(X) = w\}|$.*

Then Algorithm 1 returns **false** *if $\mathcal{S}$ is empty and* **true** *with probability at least $1/2$ otherwise.*

Algorithm 1 Cut&Count.

Input: Set U, elimination forest $\mathcal{T}$, procedure `CountC` accepting $\mathbf{w}\colon U \to [N]$, $w \in \mathbb{N}$

```
for v ∈ U do
    Choose w(v) ∈ [2|U|] uniformly at random;
for w = 1, ..., 2|U|^2 do
    if CountC(w, w, T) ≡ 1 then return true;
return false;
```

We will use the same definitions as Cygan et al. [11] for $\mathcal{Q}$ and $\mathcal{S}$, hence it follows from their proofs that Condition 1 in Corollary 3.3 is satisfied. Our contribution is to provide the counting procedure `CountC` for problems parameterized by treedepth.

Given the sets $\mathcal{S}$, $\mathcal{R}$, and $\mathcal{Q}$, and a weight function $\mathbf{w}\colon U \to [N]$, we will define for every weight w the sets $\mathcal{S}_w = \{X \in \mathcal{S} \mid \mathbf{w}(X) = w\}$, $\mathcal{R}_w = \{X \in \mathcal{R} \mid \mathbf{w}(X) = w\}$, and $\mathcal{Q}_w = \{(X, C) \in \mathcal{Q} \mid \mathbf{w}(X) = w\}$.

3.2 Connected Vertex Cover

Connected Vertex Cover	
Input:	An undirected graph $G = (V, E)$ and an integer k.
Question:	Is there a set $X \subseteq V$, $\lvert X\rvert = k$, such that $G[X]$ is connected and X is a vertex cover of G, i.e., $e \cap X \neq \emptyset$ for all $e \in E$?

In the considered problems, one usually seeks a solution of size at most k. For convenience we choose to look for a solution of size exactly k and solve the other case in the obvious way. We define the objects needed for Cut&Count in the setting of Connected Vertex Cover. We let $U = V$ and define the candidate solutions by $\mathcal{R} = \{X \subseteq V \mid X \text{ is a vertex cover of } G \text{ and } \lvert X\rvert = k\}$, and the solutions are given by $\mathcal{S} = \{X \in \mathcal{R} \mid G[X] \text{ is connected}\}$.

To ensure that a connected solution is consistent with an odd number of cuts, we choose a vertex v_1 that is always forced to the left side of the cut (cf. Lemma 3.2). As we cannot be sure that there is a minimum connected vertex cover containing v_1, we take an edge $\{u, v\} \in E$ and run Algorithm 1 once for $v_1 := u$ and once for $v_1 := v$. Hence, for a fixed choice of v_1 we define the candidate-cut-pairs by $\mathcal{Q} = \{(X, (X_L, X_R)) \in \mathcal{C}(V) \mid X \in \mathcal{R} \text{ and } v_1 \in X_L\}$. We must check that these definitions satisfy the requirements of Corollary 3.3.

▶ **Lemma 3.4** ($\star$, [11]). *Let $\mathbf{w}\colon V \to [N]$ be a weight function, and let $\mathcal{Q}$ and $\mathcal{S}$ be as defined above. Then we have for every $w \in \mathbb{N}$ that $\lvert\mathcal{S}_w\rvert \equiv \lvert\mathcal{Q}_w\rvert$.*

Next, we describe the procedure `CountC` for Connected Vertex Cover.

Algorithm 2 `CountC` for Connected Vertex Cover.

Input: Elimination forest $\mathcal{T}$, weights $\mathbf{w}\colon V \to [2n]$, target weight $w \in [2n^2]$

1 Let r denote the root of $\mathcal{T}$;
2 $P :=$ `calc_poly_inc`$(r, \emptyset)$;
3 **return** the coefficient of $Z_W^w Z_X^k$ in P;

Algorithm 3 `calc_poly_exc`(v, f).

Input: Elimination forest $\mathcal{T}$, weights $\mathbf{w}\colon V \to [2n]$, vertex $v \in V$, previous choices $f\colon$ `tail`$[v] \to \{\mathbf{1}_L, \mathbf{1}_R, \mathbf{0}\}$

1 **if** *v is a leaf of $\mathcal{T}$* **then return** the result of equation (1);
2 **else**
3 $P := 1$;
4 **for** $u \in$ `child`(v) **do** `// cf. equation (2)`
5 $P := P \cdot$ `calc_poly_inc`(v, f);
6 **return** P;

▶ **Lemma 3.5.** *Given a connected graph $G = (V, E)$, a vertex $v_1 \in V$, an integer k, a weight function $\mathbf{w}\colon V \to [2n]$, and an elimination forest $\mathcal{T}$ of G of depth d, we can determine $\lvert\mathcal{Q}_w\rvert$ modulo 2 for every $0 \leq w \leq 2n^2$ in time $\mathcal{O}^*(3^d)$ and polynomial space. In particular, Algorithm 2 determines $\lvert\mathcal{Q}_w\rvert$ modulo 2 for a specified target weight w in the same time and space.*

Algorithm 4 `calc_poly_inc`(v, g).

Input: Elimination forest $\mathcal{T}$, weights $\mathbf{w}\colon V \to [2n]$, vertex $v \in V$, previous choices $g\colon \texttt{tail}(v) \to \{\mathbf{1}_L, \mathbf{1}_R, \mathbf{0}\}$

1 **for** $s \in \{\mathbf{1}_L, \mathbf{1}_R, \mathbf{0}\}$ **do**
2 $\quad P_s := \texttt{calc\_poly\_exc}(v, g[v \mapsto s])$;
3 **return** $P_{\mathbf{1}_L} Z_W^{\mathbf{w}(v)} Z_X + P_{\mathbf{1}_R} Z_W^{\mathbf{w}(v)} Z_X + P_{\mathbf{0}}$; // cf. equation (3)

Proof. For the discussion of the algorithm, it is convenient to drop the cardinality constraint in $\mathcal{R}$ and $\mathcal{Q}$ and to define these sets for every induced subgraph $G[V']$ of G. Hence, we define for every $V' \subseteq V$ the set $\widehat{\mathcal{R}}(V') = \{X \subseteq V' \mid X \text{ is a vertex cover of } G[V']\}$ and the set $\widehat{\mathcal{Q}}(V') = \{(X, (X_L, X_R)) \in \mathcal{C}(V') \mid X \in \mathcal{R}(V') \text{ and } (v_1 \in V' \to v_1 \in X_L)\}$.

Similar to Pilipczuk and Wrochna [29], our algorithm will compute a multivariate polynomial in the formal variables Z_W and Z_X, where the coefficient of $Z_W^w Z_X^i$ is the cardinality of $\widehat{\mathcal{Q}}_w^i(V) = \{(X, C) \in \widehat{\mathcal{Q}}(V) \mid \mathbf{w}(X) = w, |X| = i\}$ modulo 2, i.e., the formal variables track the weight and size of candidate solutions. In particular, we have that $\widehat{\mathcal{Q}}_w^k = \mathcal{Q}_w$ for every w. Polynomials act as an appropriate data structure, because addition and multiplication of polynomials naturally updates the weight and size trackers correctly.

The output polynomial is computed by a branching algorithm (see Algorithm 2) that starts at the root r of the elimination forest $\mathcal{T}$ and proceeds downwards to the leaves. At every vertex we branch into several states, denoted $\texttt{states} = \{\mathbf{1}_L, \mathbf{1}_R, \mathbf{0}\}$. The interpretation of the states $\mathbf{1}_L$ and $\mathbf{1}_R$ is that the vertex is inside the vertex cover and the subscript denotes to which side of the consistent cut it belongs. Vertices that do not belong to the vertex cover have state $\mathbf{0}$.

For each vertex v there are multiple subproblems on $G[\texttt{broom}[v]]$. When solving a subproblem, we need to take into account the choices that we have already made, i.e., the branching decisions for the ancestors of v. At each vertex we compute two different types of polynomials, which correspond to two different kinds of partial solutions. Those that are subsets of $\texttt{tree}(v)$ and respect the choices made on $\texttt{tail}[v]$ and those that are subsets of $\texttt{tree}[v]$ and respect the choices made on $\texttt{tail}(v)$. Distinguishing these two types of partial solutions is important when v has multiple children in $\mathcal{T}$. Formally, the previous branching decisions are described by assignments f or g from $\texttt{tail}[v]$ or $\texttt{tail}(v)$ to $\{\mathbf{1}_L, \mathbf{1}_R, \mathbf{0}\}$ respectively.

For every vertex v and assignment $f\colon \texttt{tail}[v] \to \{\mathbf{1}_L, \mathbf{1}_R, \mathbf{0}\}$ we define the partial solutions at v, but excluding v, that respect f by

$$\mathcal{P}_{(v)}(f) = \{(X, (X_L, X_R)) \in \mathcal{C}(\texttt{tree}(v)) \mid X' = X \cup f^{-1}(\{\mathbf{1}_L, \mathbf{1}_R\}), \\ C' = (X_L \cup f^{-1}(\mathbf{1}_L), X_R \cup f^{-1}(\mathbf{1}_R)), (X', C') \in \widehat{\mathcal{Q}}(\texttt{broom}[v])\}.$$

So, $\mathcal{P}_{(v)}(f)$ consists of consistently cut subgraphs $(X, (X_L, X_R))$ of $G[\texttt{tree}(v)]$ that are extended by f to valid candidate-cut-pairs (X', C') for $G[\texttt{broom}[v]]$, meaning that X' is a vertex cover of $G[\texttt{broom}[v]]$ and C' is a consistent cut of X'.

Very similarly, for every vertex v and assignment $g\colon \texttt{tail}(v) \to \{\mathbf{1}_L, \mathbf{1}_R, \mathbf{0}\}$ we define the partial solutions at v, possibly including v, that respect g by

$$\mathcal{P}_{[v]}(g) = \{(X, (X_L, X_R)) \in \mathcal{C}(\texttt{tree}[v]) \mid X' = X \cup g^{-1}(\{\mathbf{1}_L, \mathbf{1}_R\}), \\ C' = (X_L \cup g^{-1}(\mathbf{1}_L), X_R \cup g^{-1}(\mathbf{1}_R)), (X', C') \in \widehat{\mathcal{Q}}(\texttt{broom}[v])\}.$$

Thus, for the root r of $\mathcal{T}$ we have $\mathcal{P}_{[r]}(\emptyset) = \widehat{\mathcal{Q}}(V)$.

We keep track of the partial solutions $\mathcal{P}_{(v)}(f)$ and $\mathcal{P}_{[v]}(g)$ using polynomials which we define now. For every vertex v and assignment $f\colon \mathtt{tail}[v] \to \{\mathbf{1}_L, \mathbf{1}_R, \mathbf{0}\}$ we will compute a polynomial $P_{(v)}(f) \in \mathbb{F}_2[Z_W, Z_X]$ where $P_{(v)}(f) = \sum_{w=0}^{2n^2} \sum_{i=0}^{n} c_{w,i} Z_W^w Z_X^i$ and

$$c_{w,i} = |\{(X, C) \in \mathcal{P}_{(v)}(f) \mid \mathbf{w}(X) = w \text{ and } |X| = i\}| \mod 2.$$

Similarly, for every vertex v and assignment $g\colon \mathtt{tail}(v) \to \{\mathbf{1}_L, \mathbf{1}_R, \mathbf{0}\}$ we will compute a polynomial $P_{[v]}(g) \in \mathbb{F}_2[Z_W, Z_X]$ where $P_{[v]}(g) = \sum_{w=0}^{2n^2} \sum_{i=0}^{n} c'_{w,i} Z_W^w Z_X^i$ and

$$c'_{w,i} = |\{(X, C) \in \mathcal{P}_{[v]}(g) \mid \mathbf{w}(X) = w \text{ and } |X| = i\}| \mod 2.$$

Algorithm 2 computes the polynomial $P = P_{[r]}(\emptyset)$, where r is the root of $\mathcal{T}$, and extracts the appropriate coefficient of P. To compute P we employ recurrences for $P_{(v)}(f)$ and $P_{[v]}(g)$. We proceed by describing the recurrence for $P_{(v)}(f)$.

In the case that v is a leaf node in $\mathcal{T}$, i.e., $\mathtt{tree}(v) = \emptyset$, we can compute $P_{(v)}(f)$ by

$$\begin{aligned} P_{(v)}(f) \quad &= \quad [f^{-1}(\{\mathbf{1}_L, \mathbf{1}_R\}) \text{ is a vertex cover of } G[\mathtt{tail}[v]]] \\ &\cdot \quad [(f^{-1}(\mathbf{1}_L), f^{-1}(\mathbf{1}_R)) \text{ is a consistent cut of } G[f^{-1}(\{\mathbf{1}_L, \mathbf{1}_R\})]] \\ &\cdot \quad [v_1 \in \mathtt{tail}[v] \to f(v_1) = \mathbf{1}_L], \end{aligned} \tag{1}$$

which checks whether the assignment f induces a valid partial solution. This is the only step in which we explicitly ensure that we are computing only vertex covers; in all other steps this will not be required. If v is not a leaf, then $P_{(v)}(f)$ is computed by the recurrence

$$P_{(v)}(f) = \prod_{u \in \mathtt{child}(v)} P_{[u]}(f), \tag{2}$$

which combines disjoint partial solutions. The equations (1) and (2) are used by Algorithm 3 to compute the polynomial $P_{(v)}(f)$.

We proceed by giving the recurrence that is used by Algorithm 4 to compute the polynomial $P_{[v]}(g)$:

$$P_{[v]}(g) = P_{(v)}(g[v \mapsto \mathbf{1}_L])\, Z_W^{\mathbf{w}(v)} Z_X + P_{(v)}(g[v \mapsto \mathbf{1}_R])\, Z_W^{\mathbf{w}(v)} Z_X + P_{(v)}(g[v \mapsto \mathbf{0}]). \tag{3}$$

Equation (3) tests all three possible states for v in a candidate-cut-pair and multiplies by $Z_W^{\mathbf{w}(v)} Z_X$ if v is in the vertex cover to update the weight and size of the partial solutions.

Correctness. We will now prove the correctness of equations (1) through (3). First of all, observe that when $v_1 \in \mathtt{tail}[v]$ but $f(v_1) \neq \mathbf{1}_L$ then we must have that $P_{(v)}(f) = 0$; similarly, we must have $P_{[v]}(g) = 0$ when $g(v_1) \neq \mathbf{1}_L$ for $v_1 \in \mathtt{tail}(v)$. This property is ensured by equation (1) and preserved by the recurrences (2) and (3). To see that equation (1) is correct, notice that when v is a leaf node in $\mathcal{T}$ we have that $\mathtt{tree}(v) = \emptyset$ and hence the only consistently cut subgraph of $\mathtt{tree}(v)$ is $(\emptyset, (\emptyset, \emptyset))$. Therefore, we only need to verify whether this is a valid partial solution in $P_{(v)}(f)$, which reduces to the predicate on the right-hand side of (1).

For equations (2) and (3), we have to establish bijections between the objects counted on either side of the respective equation and argue that size and weight are updated correctly. We proceed by proving the correctness of equation (2), which is the only equation where the proof of correctness requires the special properties of elimination forests. We consider any $(X, (X_L, X_R)) \in \mathcal{P}_{(v)}(f)$. We can uniquely partition X into subsets X^u of $\mathtt{tree}[u]$ for $u \in \mathtt{child}(v)$ by setting $X^u = X \cap \mathtt{tree}[u]$. Furthermore, by setting $X_L^u = X_L \cap \mathtt{tree}[u]$ and

$X^u_R = X_R \cap \texttt{tree}[u]$ we obtain $(X^u, (X^u_L, X^u_R)) \in \mathcal{P}_{[u]}(f)$, because we are only restricting the vertex cover $X' = X \cup f^{-1}(\{\mathbf{1}_L, \mathbf{1}_R\})$ and consistent cut $(X_L \cup f^{-1}(\mathbf{1}_L), X_R \cup f^{-1}(\mathbf{1}_R))$ to the induced subgraph $G[\texttt{broom}[u]]$ of $G[\texttt{broom}[v]]$. Vice versa, any combination of partial solutions $(X^u, (X^u_L, X^u_R)) \in \mathcal{P}_{[u]}(f)$ for each $u \in \texttt{child}(v)$ yields a partial solution $(X, (X_L, X_R)) \in \mathcal{P}_{(v)}(f)$ as there are no edges in G between $\texttt{tree}[u]$ and $\texttt{tree}[u']$ for $u \neq u' \in \texttt{child}(v)$ by the properties of an elimination forest. Since the sets X^u partition X, we obtain the size and weight of X by summing over the sizes and weights of the sets X^u respectively. Hence, these values are updated correctly by polynomial multiplication.

It remains to prove the correctness of (3). This time, consider any $(X, (X_L, X_R)) \in \mathcal{P}_{[v]}(g)$. Now, there are three possible cases depending on the state of v in this partial solution.

1. If $v \in X_L \subseteq X$, then we claim that $(X \setminus \{v\}, (X_L \setminus \{v\}, X_R)) \in \mathcal{P}_{(v)}(f)$, where $f = g[v \mapsto \mathbf{1}_L]$. This is true due to the identities $(X \setminus \{v\}) \cup f^{-1}(\{\mathbf{1}_L, \mathbf{1}_R\}) = X \cup g^{-1}(\{\mathbf{1}_L, \mathbf{1}_R\})$, and $(X_L \setminus \{v\}) \cup f^{-1}(\mathbf{1}_L) = X_L \cup g^{-1}(\mathbf{1}_L)$, and $X_R \cup f^{-1}(\mathbf{1}_R) = X_R \cup g^{-1}(\mathbf{1}_R)$, which mean that this implicitly defined mapping preserves the definition of X' and C' in the predicates of $\mathcal{P}_{[v]}(g)$ and $\mathcal{P}_{(v)}(f)$. Vice versa, any partial solution in $\mathcal{P}_{(v)}(f)$ can be extended to such a partial solution in $\mathcal{P}_{(v)}(g)$ by adding v to X_L. Since $|X| - |X \setminus \{v\}| = 1$ and $\mathbf{w}(X) - \mathbf{w}(X \setminus \{v\}) = \mathbf{w}(v)$, multiplication by $Z_W^{\mathbf{w}(v)} Z_X$ updates size and weight correctly.
2. If $v \in X_R \subseteq X$, the proof is analogous to case 1.
3. If $v \notin X$, then we have that $(X, (X_L, X_R)) \in \mathcal{P}_{(v)}(f)$, where $f = g[v \mapsto \mathbf{0}]$. Vice versa, any $(X, (X_L, X_R)) \in \mathcal{P}_{(v)}(f)$ must also be in $\mathcal{P}_{[v]}(g)$. Since X does not change, we do not need to update size or weight and do not multiply by further formal variables in this case.

If $v = v_1$, then equation (3) simplifies to $P_{[v]}(g) = P_{(v)}(g[v \mapsto \mathbf{1}_L]) Z_W^{\mathbf{w}(v)} Z_X$, because $\mathcal{P}_{(v)}(g[v \mapsto \mathbf{1}_R]) = \mathcal{P}_{(v)}(g[v \mapsto \mathbf{0}]) = \emptyset$ and hence only the first case occurs. Note that by establishing these bijections in the proofs of correctness, we have actually shown that equations (1) through (3) are also correct when working in $\mathbb{Z}$ instead of $\mathbb{F}_2$.

Time and Space Analysis. We finish the proof by discussing the time and space requirement. Observe that the coefficients of our polynomials are in $\mathbb{F}_2$ and hence can be added and multiplied in constant time. Furthermore, all considered polynomials consist of at most polynomially many monomials as the weight and size of a candidate solution are polynomial in n. Therefore, we can add and multiply the polynomials in polynomial time and hence compute recurrences (1), (2), and (3) in polynomial time. Every polynomial $P_{(v)}(f)$ and $P_{[v]}(g)$ is computed at most once, because $P_{(v)}(f)$ is only called by $P_{[v]}(g)$ where f is an extension of g, i.e., $f = g[v \mapsto s]$ for some $s \in \texttt{states}$, and $P_{[v]}(g)$ is only called by $P_{(w)}(g)$ where w is the parent of v. Hence, the recurrences only make disjoint calls and no polynomial is computed more than once. For a fixed vertex v there are at most 3^d choices for f and g. Thus, Algorithm 2 runs in time $\mathcal{O}^*(3^d)$ for elimination forests of depth d. Finally, Algorithm 2 requires only polynomial space, because it has a recursion depth of $2d+1$ and every recursive call needs to store at most a constant number of polynomials, which require by the previous discussion only polynomial space each. ◀

▶ **Theorem 3.6.** *There is a Monte-Carlo algorithm that given an elimination forest of depth d for a graph G solves* Connected Vertex Cover *on G in time $\mathcal{O}^*(3^d)$ and polynomial space. The algorithm cannot give false positives and may give false negatives with probability at most 1/2.*

Proof. We pick an edge $\{u, v\} \in E$ and branch on $v_1 := u$ and $v_1 := v$. We run Algorithm 1 with $U = V$ and the procedure `CountC` as given by Algorithm 2. Correctness follows from Corollary 3.3 and Lemma 3.4. Running time and space bound follow from Lemma 3.5. ◀

We remark that calling Algorithm 2 for each target weight $w \in [2n^2]$ (as in Algorithm 1) would redundantly compute the polynomial $P = P_{[r]}(\emptyset)$ several times, although it suffices to compute P once and then look up the appropriate coefficient depending on w.

If one is interested in solving Weighted Connected Vertex Cover, then it is straightforward to adapt our approach to polynomially-sized weights: instead of using Z_X to track the size of the vertex covers, we let it track their cost and change recurrence (3) accordingly.

3.3 Adapting to Other Problems

The high-level structure of the counting procedure for the other problems is very similar to that of Algorithm 2 for Connected Vertex Cover. One possible difference is that we might have to consider the solutions over a more complicated universe U than just the vertex set V. Also, we might want to keep track of more data of the partial solutions and hence use more than just two formal variables for the polynomials. Both of these changes occur for Feedback Vertex Set, which is presented in the next section. The equation for the base case (cf. equation (1)) and the recurrence for $P_{[v]}(g)$ (cf. equation (3)) are also problem-dependent.

Time and Space Analysis. The properties that we require of the polynomials and equations in the time and space analysis, namely that the equations can be evaluated in polynomial time and every polynomial is computed at most once, remain true by the same arguments as for Connected Vertex Cover. The running time essentially results from the number of computed polynomials, which increases when we use more states for the vertices. Again denoting the set of states by `states`, we obtain a running time of $\mathcal{O}^*(|\texttt{states}|^d)$ on elimination forests of depth d. The space analysis also remains valid, because the recursion depth remains $2d + 1$ and for each call we need to store only a constant number of polynomials each using at most polynomial space.

3.4 Feedback Vertex Set

Feedback Vertex Set	
Input:	An undirected graph $G = (V, E)$ and an integer k.
Question:	Is there a set $X \subseteq V$, $\lvert X\rvert = k$, such that $G - X$ is a forest?

Feedback Vertex Set differs from the other problems in that we do not have a positive connectivity requirement, but a negative connectivity requirement, i.e., we need to ensure that the remaining graph is badly connected in the sense that it contains no cycles. Cygan et al. [11] approach this via the well-known Lemma 3.7.

▶ **Lemma 3.7.** *A graph with n vertices and m edges is a forest if and only if it has at most $n - m$ connected components.*

Applying Lemma 3.7 requires that we count how many vertices and edges remain after deleting a set $X \subseteq V$ from G. We do not need to count exactly how many connected components remain, it suffices to enforce that there are not too many connected components. We will achieve this, like Cygan et al. [11], by the use of *marker vertices*. In this case, our solutions are pairs (Y, M) with $M \subseteq Y$, where we interpret Y as the forest that remains after removing a feedback vertex set X and the marked vertices are represented by the set M. To

bound the number of connected components, we want that every connected component of $G[Y]$ contains at least one marked vertex. By forcing the marked vertices to the left side of the cut, we ensure that candidates (Y, M) where $G[Y]$ has a connected component not containing a marked vertex, in particular those with more than $|M|$ connected components, cancel modulo 2. The formal definitions are $\mathcal{R} = \{(Y, M) \mid M \subseteq Y \subseteq V \text{ and } |Y| = n - k\}$, and $\mathcal{S} = \{(Y, M) \in \mathcal{R} \mid G[Y] \text{ is a forest, every connected component of } G[Y] \text{ intersects } M\}$, and $\mathcal{Q} = \{((Y, M), (Y_L, Y_R)) \mid (Y, M) \in \mathcal{R} \text{ and } (Y, (Y_L, Y_R)) \in \mathcal{C}(V) \text{ and } M \subseteq Y_L\}$.

Since our solutions (Y, M) are pairs of two vertex sets, we need a larger universe to make the Isolation Lemma, Lemma 2.3, work. We use $U = V \times \{\mathbf{F}, \mathbf{M}\}$, hence a weight function $\mathbf{w}\colon U \to [N]$ assigns two different weights $\mathbf{w}(v, \mathbf{F})$ and $\mathbf{w}(v, \mathbf{M})$ to a vertex v depending on whether v is marked or not. To make these definitions compatible with Corollary 3.3 we associate to each pair (Y, M) the set $Y \times \{\mathbf{F}\} \cup M \times \{\mathbf{M}\} \subseteq U$, which also allows us to extend the weight function to such pairs (Y, M), i.e. $\mathbf{w}(Y, M) = \mathbf{w}(Y \times \{\mathbf{F}\} \cup M \times \{\mathbf{M}\})$.

▶ **Lemma 3.8** (⋆, [11]). *Let (Y, M) be such that $M \subseteq Y \subseteq V$. The number of consistently cut subgraphs $(Y, (Y_L, Y_R))$ such that $M \subseteq Y_L$ is equal to $2^{\overline{\mathsf{cc}}_M(G[Y])}$, where $\overline{\mathsf{cc}}_M(G[Y])$ is the number of connected components of $G[Y]$ that do not contain any vertex from M.*

To apply Lemma 3.7, we need to distinguish candidates by the number of edges, and markers, in addition to the weight, hence we make the following definitions for $j, \ell, w \in \mathbb{N}$:

$$\begin{aligned}
\mathcal{R}_w^{j,\ell} &= \{(Y, M) \in \mathcal{R} \mid \mathbf{w}(Y, M) = w, \ |E(G[Y])| = j, \ |M| = \ell\},\\
\mathcal{S}_w^{j,\ell} &= \{(Y, M) \in \mathcal{S} \mid \mathbf{w}(Y, M) = w, \ |E(G[Y])| = j, \ |M| = \ell\},\\
\mathcal{Q}_w^{j,\ell} &= \{(Y, M, (Y_L, Y_R)) \in \mathcal{Q} \mid \mathbf{w}(Y, M) = w, \ |E(G[Y])| = j, \ |M| = \ell\}.
\end{aligned}$$

▶ **Lemma 3.9** (⋆, [11]). *Let $\mathbf{w}\colon U \to [N]$ be a weight function, and $\mathcal{Q}$ and $\mathcal{S}$ as defined above. Then we have for every $w \in \mathbb{N}$ and $j \in [n - k - 1]$ that $|\mathcal{S}_w^{j,n-k-j}| \equiv |\mathcal{Q}_w^{j,n-k-j}|$.*

Note that by Lemma 3.7 a Feedback Vertex Set instance has a solution X if and only if there is a choice of $w, j \in \mathbb{N}$ and $M \subseteq Y := V \setminus X$ such that $(Y, M) \in \mathcal{S}_w^{j,n-k-j}$.

▶ **Lemma 3.10.** *Given a connected graph $G = (V, E)$, an integer k, a weight function $\mathbf{w}\colon U \to [4n]$ and an elimination forest $\mathcal{T}$ of G of depth d, we can determine $|\mathcal{Q}_w^{j,n-k-j}|$ modulo 2 for every $0 \leq w \leq 4n^2$, $0 \leq j \leq m$, in time $\mathcal{O}^*(3^d)$ and polynomial space.*

Proof. Again, we drop the cardinality constraints from $\mathcal{R}$ and $\mathcal{Q}$ and define for induced subgraphs $G[V']$ the variants $\widehat{\mathcal{R}}(V') = \{(Y, M) \mid M \subseteq Y \subseteq V'\}$ and $\widehat{\mathcal{Q}}(V') = \{((Y, M), (Y_L, Y_R)) \mid (Y, M) \in \widehat{\mathcal{R}}(V') \text{ and } (Y, (Y_L, Y_R)) \in \mathcal{C}(V') \text{ and } M \subseteq Y_L\}$.

We will compute a multivariate polynomial in the formal variables Z_W, Z_Y, Z_E, Z_M, where the coefficient of $Z_W^w Z_Y^i Z_E^j Z_M^\ell$ is the cardinality modulo 2 of

$$\widehat{\mathcal{Q}}_w^{i,j,\ell} = \{((Y, M), C) \in \widehat{\mathcal{Q}}(V) \mid \mathbf{w}(Y, M) = w, |Y| = i, |E(G[Y])| = j, |M| = \ell\}.$$

The coefficients of $Z_W^w Z_Y^{n-k} Z_E^j Z_M^{n-k-j}$ for every w and j then yield the desired numbers.

For Feedback Vertex Set we require three states which are given by $\texttt{states} = \{\mathbf{1}, \mathbf{0}_L, \mathbf{0}_R\}$. The state $\mathbf{1}$ represents vertices inside the feedback vertex set; the states $\mathbf{0}_L$ and $\mathbf{0}_R$ represent vertices inside the remaining forest and the subscript denotes to which side of the consistent cut a vertex belongs. Perhaps surprisingly, there is no state to represent marked vertices. It turns out that it is not important which vertices are marked; it is sufficient to know the number of marked vertices.

For every vertex v and assignment $f\colon \texttt{tail}[v] \to \{\mathbf{1}, \mathbf{0}_L, \mathbf{0}_R\}$ we define the partial solutions at v, but excluding v, that respect f by

$$\mathcal{P}_{(v)}(f) = \{((Y, M), (Y_L, Y_R)) \in \widehat{\mathcal{Q}}(\texttt{tree}(v)) \mid Y' = Y \cup f^{-1}(\{\mathbf{0}_L, \mathbf{0}_R\}),\\ C' = (Y_L \cup f^{-1}(\mathbf{0}_L), Y_R \cup f^{-1}(\mathbf{0}_R)), ((Y', M), C') \in \widehat{\mathcal{Q}}(\texttt{broom}[v])\}.$$

The partial solutions in $\mathcal{P}_{(v)}(f)$ are consistently cut subgraphs $(Y, (Y_L, Y_R))$ of $G[\texttt{tree}(v)]$ where a subset M of the left side is marked and the extension to (Y', C') by f is a consistently cut subgraph of $G[\texttt{broom}[v]]$.

Similarly, for every vertex v and assignment $g\colon \texttt{tail}(v) \to \{\mathbf{1}, \mathbf{0}_L, \mathbf{0}_R\}$ we define the partial solutions at v, possibly including v, that respect g by

$$\mathcal{P}_{[v]}(g) = \{((Y, M), (Y_L, Y_R)) \in \widehat{\mathcal{Q}}(\texttt{tree}[v]) \mid Y' = Y \cup g^{-1}(\{\mathbf{0}_L, \mathbf{0}_R\}),\\ C' = (Y_L \cup g^{-1}(\mathbf{0}_L), Y_R \cup g^{-1}(\mathbf{0}_R)), ((Y', M), C') \in \widehat{\mathcal{Q}}(\texttt{broom}[v])\}.$$

For every vertex v and assignment $f\colon \texttt{tail}[v] \to \{\mathbf{1}, \mathbf{0}_L, \mathbf{0}_R\}$ we will compute a polynomial $P_{(v)}(f) \in \mathbb{F}_2[Z_W, Z_Y, Z_E, Z_M]$ where the coefficient of $Z_W^w Z_Y^i Z_E^j Z_M^\ell$ in $P_{(v)}(f)$ is given by

$$|\{((Y, M), C) \in \mathcal{P}_{(v)}(f) \mid \mathbf{w}(Y, M) = w, |Y| = i, |E(G[Y])| = j, |M| = \ell\}| \mod 2.$$

For every vertex v and assignment $g\colon \texttt{tail}(v) \to \{\mathbf{1}, \mathbf{0}_L, \mathbf{0}_R\}$ we will compute a polynomial $P_{[v]}(g) \in \mathbb{F}_2[Z_W, Z_Y, Z_E, Z_M]$ where the coefficient of $Z_W^w Z_Y^i Z_E^j Z_M^\ell$ in $P_{[v]}(g)$ is given by

$$|\{((Y, M), C) \in \mathcal{P}_{[v]}(g) \mid \mathbf{w}(Y, M) = w, |Y| = i, |E(G[Y])| = j, |M| = \ell\}| \mod 2.$$

The exponents of the monomials $Z_W^w Z_Y^i Z_E^j Z_M^\ell$ in $P_{(v)}(f)$ and $P_{[v]}(g)$ range between $0 \leq w \leq 4n^2$, $0 \leq i \leq n$, $0 \leq j \leq m$, and $0 \leq \ell \leq n$.

We now present the recurrences used to compute the polynomials $P_{(v)}(f)$ and $P_{[v]}(g)$. If v is a leaf node in $\mathcal{T}$, then we can compute $P_{(v)}(f)$ by

$$P_{(v)}(f) = [(f^{-1}(\mathbf{0}_L), f^{-1}(\mathbf{0}_R)) \text{ is a consistent cut of } G[f^{-1}(\{\mathbf{0}_L, \mathbf{0}_R\})]]. \tag{4}$$

If v is not a leaf node, then we compute $P_{(v)}(f)$ by

$$P_{(v)}(f) = \prod_{u \in \texttt{child}(v)} P_{[u]}(f). \tag{5}$$

To compute $P_{[v]}(g)$ we use the recurrence

$$\begin{aligned} P_{[v]}(g) \quad &= P_{(v)}(g[v \to \mathbf{1}]) \\ &+ P_{(v)}(g[v \to \mathbf{0}_L]) \quad Z_W^{\mathbf{w}(v,\mathbf{F})} \quad Z_Y \quad Z_E^{|N(v) \cap \texttt{tree}[v]|} \\ &+ P_{(v)}(g[v \to \mathbf{0}_L]) \quad Z_W^{\mathbf{w}(v,\mathbf{F}) + \mathbf{w}(v,\mathbf{M})} \quad Z_Y \quad Z_E^{|N(v) \cap \texttt{tree}[v]|} \quad Z_M \\ &+ P_{(v)}(g[v \to \mathbf{0}_R]) \quad Z_W^{\mathbf{w}(v,\mathbf{F})} \quad Z_Y \quad Z_E^{|N(v) \cap \texttt{tree}[v]|}. \end{aligned} \tag{6}$$

This recurrence tests all three possible states for the vertex v and whether it is marked. In the last case v has state $\mathbf{0}_L$, but the formal variables Z_W and Z_M must be updated differently from the case where v is not marked but has state $\mathbf{0}_L$.

We will now prove the correctness of the equations (4) to (6). For the correctness of equation (4), notice that $\texttt{tree}(v) = \emptyset$ when v is a leaf. Hence, $\widehat{\mathcal{Q}}(\texttt{tree}(v))$ degenerates to $\{((\emptyset, \emptyset), (\emptyset, \emptyset))\}$ and we must check whether $((\emptyset, \emptyset), (\emptyset, \emptyset)) \in \mathcal{P}_{(v)}(f)$ which means that $((f^{-1}(\{\mathbf{0}_L, \mathbf{0}_R\}), \emptyset), (f^{-1}(\mathbf{0}_L), f^{-1}(\mathbf{0}_R))) \in \widehat{\mathcal{Q}}(\texttt{broom}[v])$ and checking the consistency of the cut is the only nontrivial requirement in this case.

The proof of correctness for equation (5) is similar to the proof for equation (2) of Connected Vertex Cover. Any solution in $\mathcal{P}_{(v)}(f)$ uniquely partitions into solutions in $\mathcal{P}_{[u]}(f)$ for each $u \in \texttt{child}(v)$. Vice versa, any combination of solutions for the children u of v yields a unique solution in $\mathcal{P}_{(v)}(f)$. The properties of an elimination forest are needed to show that the union of consistent cuts remains a consistent cut. We omit further details.

To prove the correctness of equation (6) we consider a partial solution $((Y, M), (Y_L, Y_R)) \in \mathcal{P}_{[v]}(g)$ and distinguish between four cases depending on the state of v.

1. If $v \notin Y$, then $((Y, M), (Y_L, Y_R)) \in \mathcal{P}_{(v)}(f)$, where $f = g[v \mapsto \mathbf{1}]$, because there is no constraint involving vertices with state $\mathbf{1}$. Vice versa, we have that any partial solution $((Y, M), (Y_L, Y_R)) \in \mathcal{P}_{(v)}(f)$ must also be in $\mathcal{P}_{[v]}(g)$. Since Y and M do not change, we do not need to multiply by further formal variables.
2. If $v \in Y_L \subseteq Y$ and $v \notin M$, then $((Y \setminus \{v\}, M), (Y_L \setminus \{v\}, Y_R)) \in \mathcal{P}_{(v)}(f)$, where $f = g[v \mapsto \mathbf{0}_L]$, because the definition of Y' and C' in the predicate in the definition of $\mathcal{P}_{[v]}(g)$ and $\mathcal{P}_{(v)}(f)$ do not change. Hence, we can also extend any partial solution of $\mathcal{P}_{(v)}(f)$ to such a partial solution of $\mathcal{P}_{[v]}(g)$ by adding v to Y_L. The number of vertices in Y increase by 1, $|E(G[Y])|$ increases by $|N(v) \cap \texttt{tree}[v]|$, and the weight increases by $\mathbf{w}(v, \mathbf{F})$. Therefore, multiplication with $Z_W^{\mathbf{w}(v,\mathbf{F})} Z_Y Z_E^{|N(v)\cap \texttt{tree}[v]|}$ is the correct update.
3. If $v \in M \subseteq Y_L \subseteq Y$, then $((Y \setminus \{v\}, M \setminus \{v\}), (Y_L \setminus \{v\}, Y_R)) \in \mathcal{P}_{(v)}(f)$, where $f = g[v \mapsto \mathbf{0}_L]$. The argument is similar to case 2. Note that, again, the definition of Y' and C' do not change in the predicates. The set of marked vertices M does change, but we only need to ensure that M remains a subset of the left side of the cut, which we do by removing v from M. In addition to the changes in the number of vertices and edges from case 2, the number of marked vertices has increased by 1 and the weight increases by an additional $\mathbf{w}(v, \mathbf{M})$, to keep track of these changes we further multiply by $Z_W^{\mathbf{w}(v,\mathbf{M})} Z_M$.
4. If $v \in Y_R \subseteq Y$, then the proof is analogous to case 2.

The running time and space bound follows from the general discussion in Section 3.3. ◀

▶ **Theorem 3.11.** *There exists a Monte-Carlo algorithm that given an elimination forest of depth d solves Feedback Vertex Set in time $\mathcal{O}^*(3^d)$ and polynomial space. The algorithm cannot give false positives and may give false negatives with probability at most 1/2.*

Proof. We set $U = V \times \{\mathbf{F}, \mathbf{M}\}$, but we need to slightly adapt the definition of $\mathcal{S}$ and $\mathcal{Q}$ to be able to apply Corollary 3.3. We define $\widetilde{\mathcal{S}} = \cup_{j=0}^{n-k-1} \cup_{w=0}^{4n^2} \mathcal{S}_w^{j,n-k-j}$ and $\widetilde{\mathcal{Q}} = \cup_{j=0}^{n-k-1} \cup_{w=0}^{4n^2} \mathcal{Q}_w^{j,n-k-j}$. Note that $\mathcal{S}$ is nonempty if and only if $\widetilde{\mathcal{S}}$ is nonempty by Lemma 3.7. The procedure `CountC` is given by running the algorithm from Lemma 3.10 and for a given target weight w adding up (modulo 2) the values of $|\mathcal{Q}_w^{j,n-k-j}|$ for $j = 0, \ldots, n-k-1$, thereby obtaining the cardinality of $\widetilde{\mathcal{Q}}_w$ modulo 2. The desired algorithm is then given by running Algorithm 1. The correctness follows from Lemma 3.9 and Corollary 3.3 with $\widetilde{\mathcal{S}}$ and $\widetilde{\mathcal{Q}}$ instead of $\mathcal{S}$ and $\mathcal{Q}$. The running time and space bound follows from Lemma 3.10. ◀

Theorem 3.11 allows us to easily reobtain a result by Cygan et al. [11] on Feedback Vertex Set parameterized by Feedback Vertex Set. Recently, this result has been superseded by results of Li and Nederlof [21]; they present an $\mathcal{O}^*(2.7^k)$-time and exponential-space algorithm and an $\mathcal{O}^*(2.8446^k)$-time and polynomial-space algorithm for this problem.

▶ **Corollary 3.12** (⋆, [11]). *There is a Monte-Carlo algorithm that given a feedback vertex set of size s solves Feedback Vertex Set in time $\mathcal{O}^*(3^s)$ and polynomial space. The algorithm cannot give false positives and may give false negatives with probability at most 1/2.*

4 Conclusion

The Cut&Count technique of Cygan et al. [11] has provided single-exponential-time and -space algorithms for many connectivity problems parameterized by treewidth. We have shown that this technique is just as useful for parameterization by treedepth, where we have obtained single-exponential-time and *polynomial-space* algorithms. Our algorithms run in time $\mathcal{O}^*(\alpha^d)$, where α is a small constant and d is the depth of a given elimination forest. The base α matches that obtained by Cygan et al. [11] for parameterization by treewidth. Assuming SETH, this base is optimal for treewidth, or even pathwidth [11]. In principle, since treedepth is a larger parameter than both treewidth and pathwidth, it may be possible to obtain better running times when parameterizing by treedepth, possibly at the cost of using exponential space. The style of construction, used to obtain lower bounds relative to treewidth, used by Lokshtanov et al. [22] and Cygan et al. [11], necessitates long paths and is thereby unsuitable for bounds relative to treedepth. Thus, the question remains whether our running times are optimal; it is tempting to conjecture that they are.

While we have not given the proofs, our techniques also extend to other problems like Connected Feedback Vertex Set and Connected Total Dominating Set. However, there are several problems, including Cycle Cover and Longest Cycle, for which Cygan et al. [11] obtain efficient algorithms, where it is yet unclear how to solve them in polynomial space when parameterizing by treedepth. In particular, Hamiltonian Path and Hamiltonian Cycle share the same issues, namely that the algorithms parameterized by treewidth keep track of the degrees in the partial solutions and it is not clear how to do that when branching on the elimination forest while only using polynomial space. Belbasi and Fürer [2] can count Hamiltonian cycles in polynomial space, but their running time also depends on the width of a given tree decomposition. An algorithm for any of these problems parameterized by treedepth, with single-exponential running time and requiring only polynomial space, would be quite interesting.

References

1 Jochen Alber and Rolf Niedermeier. Improved tree decomposition based algorithms for domination-like problems. In Sergio Rajsbaum, editor, *LATIN 2002: Theoretical Informatics, 5th Latin American Symposium, Cancun, Mexico, April 3-6, 2002, Proceedings*, volume 2286 of *Lecture Notes in Computer Science*, pages 613–628. Springer, 2002. doi:10.1007/3-540-45995-2_52.

2 Mahdi Belbasi and Martin Fürer. A space-efficient parameterized algorithm for the Hamiltonian cycle problem by dynamic algebraization. In René van Bevern and Gregory Kucherov, editors, *Computer Science - Theory and Applications - 14th International Computer Science Symposium in Russia, CSR 2019, Novosibirsk, Russia, July 1-5, 2019, Proceedings*, volume 11532 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2019. doi:10.1007/978-3-030-19955-5_4.

3 Benjamin Bergougnoux and Mamadou Moustapha Kanté. Fast exact algorithms for some connectivity problems parameterized by clique-width. *Theor. Comput. Sci.*, 782:30–53, 2019. doi:10.1016/j.tcs.2019.02.030.

4 Benjamin Bergougnoux and Mamadou Moustapha Kanté. More applications of the d-neighbor equivalence: Connectivity and acyclicity constraints. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany.*, volume 144 of *LIPIcs*, pages 17:1–17:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.17.

5 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.

6 Hans L. Bodlaender, Jitender S. Deogun, Klaus Jansen, Ton Kloks, Dieter Kratsch, Haiko Müller, and Zsolt Tuza. Rankings of graphs. *SIAM J. Discrete Math.*, 11(1):168–181, 1998. doi:10.1137/S0895480195282550.

7 Hans L. Bodlaender, John R. Gilbert, Hjálmtyr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, 1995. doi:10.1006/jagm.1995.1009.

8 Li-Hsuan Chen, Felix Reidl, Peter Rossmanith, and Fernando Sánchez Villaamil. Width, depth, and space: Tradeoffs between branching and dynamic programming. *Algorithms*, 11(7):98, 2018. doi:10.3390/a11070098.

9 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990.

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

11 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michał Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.23.

12 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

13 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006. doi:10.1007/3-540-29953-X.

14 Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Representative sets of product families. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 443–454. Springer, 2014. doi:10.1007/978-3-662-44777-2_37.

15 Fedor V. Fomin, Daniel Lokshtanov, and Saket Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 142–151. SIAM, 2014. doi:10.1137/1.9781611973402.10.

16 Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004. doi:10.1016/j.apal.2004.01.007.

17 Martin Fürer and Huiwen Yu. Space saving by dynamic algebraization based on tree-depth. *Theory Comput. Syst.*, 61(2):283–304, 2017. doi:10.1007/s00224-017-9751-3.

18 Falko Hegerfeld and Stefan Kratsch. Solving connectivity problems parameterized by treedepth in single-exponential time and polynomial space. *arXiv e-prints*, page arXiv:2001.05364, January 2020. arXiv:2001.05364.

19 Meir Katchalski, William McCuaig, and Suzanne M. Seager. Ordered colourings. *Discrete Mathematics*, 142(1-3):141–154, 1995. doi:10.1016/0012-365X(93)E0216-Q.

20 Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. doi:10.1007/BFb0045375.

21 Jason Li and Jesper Nederlof. Detecting feedback vertex sets of size k in $\mathcal{O}^*(2.7^k)$ time. *CoRR*, abs/1906.12298, 2019. arXiv:1906.12298.

22 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs of bounded treewidth are probably optimal. *ACM Trans. Algorithms*, 14(2):13:1–13:30, 2018. doi:10.1145/3170442.

23 Daniel Lokshtanov and Jesper Nederlof. Saving space by algebraization. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC*

2010, Cambridge, Massachusetts, USA, 5-8 June 2010, pages 321–330. ACM, 2010. doi:10.1145/1806689.1806735.

24 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.

25 Jaroslav Nešetřil and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006. doi:10.1016/j.ejc.2005.01.010.

26 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.

27 Jaroslav Nešetřil and Patrice Ossona de Mendez. On low tree-depth decompositions. *Graphs and Combinatorics*, 31(6):1941–1963, 2015. doi:10.1007/s00373-015-1569-7.

28 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. doi:10.1093/ACPROF:OSO/9780198566076.001.0001.

29 Michał Pilipczuk and Marcin Wrochna. On space efficiency of algorithms working on structural decompositions of graphs. *TOCT*, 9(4):18:1–18:36, 2018. doi:10.1145/3154856.

30 Willem J. A. Pino, Hans L. Bodlaender, and Johan M. M. van Rooij. Cut and count and representative sets on branch decompositions. In Jiong Guo and Danny Hermelin, editors, *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, volume 63 of *LIPIcs*, pages 27:1–27:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.27.

31 Jan Arne Telle and Andrzej Proskurowski. Practical algorithms on partial k-trees with an application to domination-like problems. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, Nicola Santoro, and Sue Whitesides, editors, *Algorithms and Data Structures, Third Workshop, WADS '93, Montréal, Canada, August 11-13, 1993, Proceedings*, volume 709 of *Lecture Notes in Computer Science*, pages 610–621. Springer, 1993. doi:10.1007/3-540-57155-8_284.

32 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.

33 Johan M. M. van Rooij, Hans L. Bodlaender, Erik Jan van Leeuwen, Peter Rossmanith, and Martin Vatshelle. Fast dynamic programming on graph decompositions. *CoRR*, abs/1806.01667, 2018. arXiv:1806.01667.

34 Gerhard J. Woeginger. Space and time complexity of exact algorithms: Some open problems (invited talk). In Rodney G. Downey, Michael R. Fellows, and Frank K. H. A. Dehne, editors, *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004, Proceedings*, volume 3162 of *Lecture Notes in Computer Science*, pages 281–290. Springer, 2004. doi:10.1007/978-3-540-28639-4_25.