

A Faster Polynomial-Space Algorithm for Hamiltonian Cycle Parameterized by Treedepth

Stefan Kratsch 

Humboldt-Universität zu Berlin, Germany

Abstract

A large number of NP-hard graph problems can be solved in $c^w n^{\mathcal{O}(1)}$ time and space when the input graph is provided together with a tree decomposition of width w . Moreover, assuming the Strong Exponential-Time Hypothesis (SETH) we have essentially matching lower bounds for many such problems. The main drawback of these results is that the corresponding dynamic programming algorithms use exponential space, which makes them infeasible for larger w , and there is some evidence that this cannot be avoided.

This motivates using somewhat more restrictive structure/decompositions of the graph to also get good (exponential) dependence on the corresponding parameter but use only polynomial space. A number of papers have contributed to this quest by studying problems relative to treedepth, and have obtained fast polynomial space algorithms, often matching the dependence on treewidth in the time bound. E.g., a number of connectivity problems could be solved by adapting the cut-and-count technique of Cygan et al. (FOCS 2011, TALG 2022) to treedepth, but this excluded well-known path and cycle problems such as HAMILTONIAN CYCLE (Hegerfeld and Kratsch, STACS 2020).

Recently, Nederlof et al. (SIDMA 2023) showed how to solve HAMILTONIAN CYCLE, and several related problems, in $5^\tau n^{\mathcal{O}(1)}$ randomized time and polynomial space when provided with an elimination forest of depth τ . We present a faster (also randomized) algorithm, running in $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space, for the same set of problems. We use ordered pairs of what we call *consistent* matchings, rather than perfect matchings in an auxiliary graph, to get the improved time bound.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Graph algorithms analysis

Keywords and phrases Parameterized algorithms, Hamiltonian cycle, polynomial space, treedepth

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.103

Related Version *Full Version:* <https://arxiv.org/abs/2603.24492>

1 Introduction

It is widely believed that NP-hard problems do not admit polynomial-time algorithms for solving them exactly (i.e., that $P \neq NP$). A core question of parameterized complexity is, therefore, to what extent input structure can be leveraged to nevertheless get reasonably fast algorithms, i.e., *how structure affects complexity*. It is well known, for example, that a large number of NP-hard graph problems can be solved even in linear time on graphs of bounded treewidth, i.e., in time $f(\text{tw})(n + m)$ via Courcelle's theorem [6] and its extensions. While the function $f(\text{tw})$ is enormous in general, much better bounds were obtained for a variety of problems.¹ More recently, initiated by Lokshtanov et al. [19], it was shown that many of these improved bounds were in fact optimal under the Strong Exponential-Time Hypothesis (SETH),² e.g., DOMINATING SET can be solved in $3^{\text{tw}} n^{\mathcal{O}(1)}$ but not in $(3 - \varepsilon)^{\text{tw}} n^{\mathcal{O}(1)}$ time for

¹ Often at the cost of a modest increase in the polynomial dependence on the graph size, and with the best bounds assuming that a tree decomposition of the graph is already known.

² SETH is the hypothesis that for all $\varepsilon > 0$ there is $q \in \mathbb{N}$ such that q -CNF SAT cannot be solved in $O((2 - \varepsilon)^n)$ time for formulas with n variables.



any $\varepsilon > 0$ (assuming SETH). In some cases, arriving at such tight bounds first required novel algorithmic tools, such as the *cut-and-count* technique of Cygan et al. [10], which enabled single-exponential dependence on the treewidth for many connectivity-related problems.

With these much better (and possibly optimal) bounds relative to the treewidth there still remains a caveat: All these algorithms rely on some form or other of dynamic programming (DP) and they essentially use as much space as time, i.e., exponential space relative to treewidth (and ditto for other width parameters). While the (conditionally) optimal bounds are very satisfying, this is quite an obstacle for applying the algorithms on a wider scale, as a lack of memory stops a computation whereas time is (somewhat) more readily available.³ Unfortunately, this does not come from a lack of algorithmic design capability, but it is widely believed that these time bounds are not obtainable using significantly less space. Moreover, there are both unconditional lower bounds and completeness for specific models [12, 5] as well as general, conditional lower bounds [24]. Thus, to get polynomial space, we will have to reduce the generality of the input structure that we can deal with, i.e., we need to use more restrictive graph parameters corresponding to more restrictive structure and decompositions.

A key parameter in this regard is the *treedepth* of the graph (along with several generalizations): The treedepth of a graph $G = (V, E)$, denoted $\text{td}(G)$, is defined as the minimum depth τ of a rooted forest $\mathcal{T} = (V, F)$ such that for each edge $\{u, v\} \in E$ one of u and v is an ancestor of the other in $\mathcal{T}$. Such a forest $\mathcal{T}$ is then also called an *elimination forest* of G , which comes from a different but equivalent way of defining treedepth. It is well known (and easy to see) that treedepth is at least as big as pathwidth, which is at least as big as treewidth; at the same time, the treedepth of a graph with n vertices is at most $\log n$ times its treewidth. Where path and tree decompositions lend themselves to dynamic programming (using exponential space), elimination forests also often allow branching algorithms with the same parameter dependence but using only polynomial space. E.g., DOMINATING SET can be solved in $3^\tau n^{\mathcal{O}(1)}$ time and polynomial space when given an elimination forest of depth τ as part of the input [24].

Generally, branching seems the method of choice to get fast polynomial-space algorithms, yet even simple dynamic programming approaches do not easily carry over to branching: Intuitively, in partial solutions maintained by DP algorithms we usually have changing roles/states of vertices, e.g., a vertex turning from undominated to dominated, which at first glance is hard to do in a branching algorithm that essentially gives each vertex a fixed state (though will make that “choice” many times in its run). Only for what one may call “static” partial solutions, e.g., for q -COLORING, is there an immediate way to transfer from DP to branching. In this way, given an elimination forest of depth τ , it is straightforward to solve INDEPENDENT SET and q -COLORING in $2^\tau n^{\mathcal{O}(1)}$ resp. $q^\tau n^{\mathcal{O}(1)}$ time and polynomial space.

Fürer and Yu [14] adapted the algebraization technique of Lokshtanov and Nederlof [20] to a dynamic setting to obtain an algorithm that counts the number of perfect matchings in $2^\tau n^{\mathcal{O}(1)}$ time and polynomial space. Chen et al. [5] (next to lower bounds above) solve DOMINATING SET in either $\tau^{\mathcal{O}(\tau^2)} n$ time and $\tau^{\mathcal{O}(1)} \log n$ space or in $3^\tau \log \tau n$ time and $\mathcal{O}(2^\tau \tau \log \tau + \tau \log n)$ space. Building on Fürer and Yu [14], Pilipczuk and Wrochna [24] design a $3^\tau n^{\mathcal{O}(1)}$ time and polynomial space algorithm for DOMINATING SET, and then improve its space usage to as little as $\mathcal{O}(\tau \log n)$ using Fourier transform and Chinese remaindering (next to foundational work above). Independently, Belbasi and Fürer [1] got the same time bound but use a little more space. A different work of Belbasi and Fürer [2]

³ In the words of Gerhard Woeginger [26]: “Note that algorithms with exponential space complexities are absolutely useless for real life applications.”

solves HAMILTONIAN CYCLE in $(4w)^\tau n^{\mathcal{O}(1)}$ time and $\mathcal{O}(w\tau n \log n)$ space where w is the width and τ the depth of a given tree decomposition.⁴ The same time but with an extra factor of sum of edge weights in space is also obtained for TRAVELING SALESPERSON [2].

Hegerfeld and Kratsch [15] adapted the cut-and-count technique of Cygan et al. [10] to obtain, for a number of connectivity problems, the same (randomized) time bound as relative to treewidth but using only polynomial space. The main omission in their work, and seemingly less amenable to cut-and-count relative to treedepth, are the path and cycle problems covered by Cygan et al. such as HAMILTONIAN CYCLE and MIN CYCLE COVER. This was addressed by Nederlof et al. [23] who used a different approach via perfect matchings in an auxiliary graph to solve PARTIAL CYCLE COVER in $5^\tau n^{\mathcal{O}(1)}$ randomized time and polynomial space; in this problem, given a graph G and numbers k, ℓ we need to decide whether exactly ℓ vertices of G can be covered with at most k vertex-disjoint cycles. From this, they directly get the same bounds for a bunch of related problems such as HAMILTONIAN CYCLE, LONG PATH, and MIN CYCLE COVER. They point out the gap to the dependence of $(2 + \sqrt{2})^p n^{\mathcal{O}(1)}$ time and space relative to the width p of a given path decomposition [9] and ask whether that time can be matched relative to treedepth but using polynomial space only. That being said, any improvement to the base 5 in the dependence on τ would be of interest [23].

Our work. We design a faster polynomial space Monte-Carlo algorithm for the PARTIAL CYCLE COVER problem (and thereby also for HAMILTONIAN CYCLE). Our algorithm runs in $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space when provided with an elimination forest of depth τ for the graph. Like the algorithm of Nederlof et al. [23] it does not make false positives and the chance of a false negative is at most $\frac{1}{2}$. As a corollary, observed by Nederlof et al. [23], we get the same bounds also for a number of classical path and cycle problems such as HAMILTONIAN CYCLE and LONGEST PATH.

► **Theorem 1.** *There is a $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space Monte-Carlo algorithm that, given a graph G , numbers $k, \ell \in \mathbb{N}$, and an elimination forest of depth τ for G , solves PARTIAL CYCLE COVER in $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space. Errors are limited to false negatives and occur with probability at most $\frac{1}{2}$.*

► **Corollary 2.** *Given a graph G , and k, ℓ as needed, and an elimination forest of depth τ for G , one can solve HAMILTONIAN CYCLE, HAMILTONIAN PATH, LONG CYCLE, LONG PATH, and MIN CYCLE COVER in $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space with false negatives only and error chance at most $\frac{1}{2}$.*

Our algorithm crucially relies on properties of what we call consistent pairs of matchings: Say that two matchings M_1 and M_2 are *consistent* if (1) $M_1 \cap M_2 = \emptyset$ and (2) $V(M_1) = V(M_2)$. Using well known relations between matchings and (partial) cycle covers one immediately observes that the union of two matchings M_1 and M_2 in G is a partial cycle cover in G if and only if M_1 and M_2 are consistent. Note, however, that the resulting partial cycle covers can only have even-length cycles. Fortunately, we can easily reduce PARTIAL CYCLE COVER to the special case where G is bipartite, while only adding 1 to the depth τ , so that all partial cycle covers consist of even-length cycles only.

⁴ Fürer and Yu [14] point out that treedepth can also be characterized as the maximum number of forget nodes in any nice tree decomposition of the graph, which they call the depth of the decomposition.

Our main technical contribution is a $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space deterministic algorithm that, given a graph $G = (V, E)$, $w, \ell \in \mathbb{N}$, edge weights $\mathbf{w}: E \rightarrow \{1, \dots, 2\ell\}$, and an elimination forest of depth τ for G , computes the number of *ordered pairs* of consistent matchings of cardinality $\frac{\ell}{2}$ each and total edge weight w , which we denote by $|\mathcal{M}_{w,\ell}|$. This itself is based on an inclusion-exclusion formula for $|\mathcal{M}_{w,\ell}|$ with $2n$ requirements.

To come back to partial cycles covers, it is easy to see (and detailed later) that a partial cycle cover with exactly p cycles visiting exactly ℓ vertices corresponds to 2^p ordered pairs of consistent matchings of cardinality $\frac{\ell}{2}$, which is similar to the symmetry used by Nederlof et al. [23]. By a standard application of the isolation lemma, we can reduce (with at least $\frac{1}{2}$ chance of success) to the case where there either is no partial cycle cover with at most k cycles on exactly ℓ vertices, or there is a unique such partial cycle cover of minimum total edge weight. Thus, counting the ordered pairs of consistent matchings modulo 2^{k+1} will let us detect which of the two cases is true, as all partial cycle covers with $p \geq k + 1$ cycles contribute a multiple of 2^{k+1} , which is congruent to 0.

Related work. Recently, Bergougnoux et al. [4] presented a logic-based meta-theorem for problems parameterized by treedepth: They show $2^{\mathcal{O}(\tau)} n^{\mathcal{O}(1)}$ time and polynomial space for problems expressible in what they call neighborhood operator logic with acyclicity, connectivity, and clique constraints. Dropping acyclicity and connectivity lets them improve the space to $\mathcal{O}(\tau \log n)$. This interesting result further establishes treedepth for obtaining fast parameterized algorithms with polynomial space, while still leaving room for hunting down better or even conditionally optimal time bounds (modulo SETH) for specific problems.

Speaking of conditional optimality (hence lower bounds), Jaffke and Jansen [17] showed that (modulo SETH) there is no $(q - \varepsilon)^{|X|} n^{\mathcal{O}(1)}$ time algorithm for q -COLORING when provided with X such that $G - X$ is a disjoint union of paths; this implies the same lower bound relative to treedepth using that X gives rise to an elimination forest of depth $|X| + \log n$. Koutis et al. [18] showed that under either of SETH or the Set Cover Conjecture (see [7]) DOMINATING SET parameterized by the size vc of a given vertex cover cannot be solved in $(2 - \varepsilon)^{vc} n^{\mathcal{O}(1)}$ time (and gave a matching algorithm relative to vc); the lower bound carries over to DOMINATING SET relative to treedepth but the best known algorithm takes $3^\tau n^{\mathcal{O}(1)}$ time. Hegerfeld and Kratsch [16] gave several lower bounds that carry over relative to treedepth, e.g., that (modulo SETH) there is no $(r + 1 - \varepsilon)^\tau n^{\mathcal{O}(1)}$ time algorithm for DELETION TO r -COLORABLE, which generalizes VERTEX COVER and ODD CYCLE TRANSVERSAL. This was pushed further by Esmer et al. [13] who cover also some packing problems and DOMINATING SET,⁵ and with a bit less structure required, but it remains that reproducing the bases relative to treewidth works best for “coloring-like” problems. E.g., we know nothing about lower bounds for HAMILTONIAN CYCLE relative to treedepth.

We would be remiss not to mention the state of the art for actually finding optimal elimination forests, i.e., with depth $\tau = \text{td}(G)$ (or testing whether $\text{td}(G) \leq \tau$, given τ): Reidl et al. [25] were able to do this in $2^{\mathcal{O}(\tau^2)} \cdot n$ time and exponential space. More recently, Nadara et al. [22] showed $2^{\mathcal{O}(\tau^2)} \cdot n^{\mathcal{O}(1)}$ time and polynomial space, but also $2^{\mathcal{O}(\tau^2)} \cdot n$ randomized time and $\tau^{\mathcal{O}(1)} \cdot n$ space. As pointed out by Nederlof et al. [23], we are “missing” a constant-factor approximation in $2^{\mathcal{O}(\tau)} n^{\mathcal{O}(1)}$ time, ideally with polynomial space.

Bergougnoux et al. [3] showed that the existence of fast polynomial-space algorithms can be pushed beyond treedepth by presenting single-exponential time and polynomial-space algorithms for INDEPENDENT SET, DOMINATING SET, and MAX CUT relative to shrubdepth, which is a dense generalization of treedepth and similar also to cliquewidth.⁶

⁵ Notably, for DOMINATING SET this does match base 3 as for treewidth, but only rules out base $2 - \varepsilon$.

⁶ We omit the time bounds here since they only make sense after introducing the necessary *tree models*.

Organization. Section 2 provides the necessary notation and recalls the isolation lemma. In Section 3 we derive an inclusion-exclusion formula for the number $|\mathcal{M}_{w,\ell}|$ of ordered pairs of consistent matchings of given cardinality and weight. Section 4 is the longest and most technical part, and shows how to recursively compute $|\mathcal{M}_{w,\ell}|$ given an elimination forest for the graph. Section 5 shows the algorithm for PARTIAL CYCLE COVER and the implied further applications. We conclude in Section 6 with a few questions for further research.

2 Preliminaries

We agree that $\mathbb{N} = \{0, 1, \dots\}$. We use $[n] := \{1, \dots, n\}$ and $[a, b] := \{a, a+1, \dots, b\}$. We use $\llbracket \dots \rrbracket$ for Iverson's bracket notation, where, for a proposition p , we have $\llbracket p \rrbracket = 1$ if p is true, and $\llbracket p \rrbracket = 0$ otherwise. Note, e.g., that $\llbracket p \wedge q \rrbracket = \llbracket p \rrbracket \llbracket q \rrbracket$.

Graphs. All graphs in this work are undirected and simple. We use standard graph notation (cf. [11]). For a graph $G = (V, E)$ and $U \subseteq V$ we denote by $\delta[U]$ the set of edges having *at least* one endpoint in U , formally $\delta[U] = \{e \in E \mid e \cap U \neq \emptyset\}$. A *matching* in $G = (V, E)$ is a set $M \subseteq E$ of pairwise disjoint edges. We recall (from the introduction) that two matchings M_1 and M_2 are *consistent* if (1) $M_1 \cap M_2 = \emptyset$ and (2) $V(M_1) = V(M_2)$. A *partial cycle cover* in $G = (V, E)$ is a set $C \subseteq E$ of edges such that each vertex of (V, C) has degree zero or two, i.e., it corresponds to a family of vertex-disjoint (simple) cycles in G .

Treewidth. Let $G = (V, E)$ be a graph. An *elimination forest* $\mathcal{T}$ for G is a rooted forest on the same vertex set V as G (but allowed any edges on G) such that for each edge $\{u, v\} \in E$ either u is an ancestor of v in $\mathcal{T}$ or vice versa. The *depth* τ of an elimination forest is the largest number of vertices (not edges) from any root of the forest to a leaf (of its tree). The *treewidth* of G is the smallest depth τ over all elimination forests for G . Though G and $\mathcal{T}$ have the same vertex set, we shall speak of vertex v when referring to v as a vertex of G , and of node v when referring to its role in $\mathcal{T}$. E.g., v can have child nodes (as a node of $\mathcal{T}$).

We use treewidth-related notation from [24], slightly extended by [15]: For a rooted forest $\mathcal{T} = (V, F)$ and node $v \in V$, we denote by $\text{tree}[v]$ the set of nodes in the subtree of v in $\mathcal{T}$, including v . Furthermore, we let $\text{tree}(v) := \text{tree}[v] \setminus \{v\}$. By $\text{tail}[v]$ we denote the set of nodes that are ancestors of v , including v . Furthermore, we let $\text{tail}(v) := \text{tail}[v] \setminus \{v\}$. Finally, we use $\text{broom}[v] := \{v\} \cup \text{tail}(v) \cup \text{tree}(v)$, and we will not define or use $\text{broom}(v)$.

Isolation lemma. We recall the well known isolation lemma due to Mulmuley et al. [21]:

► **Lemma 3** (Isolation Lemma, [21]). *Let $\mathcal{F} \subseteq 2^U$ be a nonempty set family over a universe U and let $N \in \mathbb{N}$. Let $\mathbf{w}: U \rightarrow [N]$ be chosen uniformly and independently at random. Then with probability at least $1 - \frac{|U|}{N}$ there is a unique $F \in \mathcal{F}$ of minimum size $\mathbf{w}(F) := \sum_{u \in F} \mathbf{w}(u)$.*

3 Inclusion-exclusion for ordered pairs of consistent matchings

Throughout this section let $G = (V, E)$ be a graph with vertex set $V = \{1, \dots, n\}$ let $\mathbf{w}: E \rightarrow \{1, \dots, N\}$ be a weight function, and let $w, \ell \in \mathbb{N}$, with ℓ even and $\ell \leq n$. We will derive an inclusion-exclusion formula for the number of ordered pairs of consistent matchings of cardinality $\frac{\ell}{2}$ and total edge weight equal to w . Formally, this is equal to $|\mathcal{M}_{w,\ell}|$ where

$$\mathcal{M}_{w,\ell} := \{(M_1, M_2) \mid M_1 \text{ and } M_2 \text{ are consistent matchings in } G, |M_1| = |M_2| = \ell/2, \mathbf{w}(M_1 \cup M_2) = w\}.$$

We will define a ground set $U_{w,\ell}$ and requirements $A_{w,\ell,i} \subseteq U_{w,\ell}$, for $i \in [2n]$ such that

$$\left| \bigcap_{i \in [2n]} A_{w,\ell,i} \right| = |\mathcal{M}_{w,\ell}|. \quad (1)$$

The set $U_{w,\ell}$ is defined to contain all triples (E_1, E_2, L) where E_1 and E_2 are disjoint edge subsets of G of cardinality $\frac{\ell}{2}$ each and total weight equal to w , and L is a vertex subset of G of cardinality $n - \ell$. Formally,

$$U_{w,\ell} := \left\{ (E_1, E_2, L) \mid E_1, E_2 \in \binom{E}{\ell/2}, E_1 \cap E_2 = \emptyset, \mathbf{w}(E_1 \cup E_2) = w, L \in \binom{V}{n-\ell} \right\}.$$

We define $2n$ requirements $A_{w,\ell,i} \subseteq U_{w,\ell}$. The first n enforce coverage of each $i \in V$ by E_1 or L , the second n enforce coverage by E_2 or L . Formally,

$$A_{w,\ell,i} := \begin{cases} \{(E_1, E_2, L) \in U_{w,\ell} \mid i \in V(E_1) \cup L\} & \text{if } i \in [n] \\ \{(E_1, E_2, L) \in U_{w,\ell} \mid i - n \in V(E_2) \cup L\} & \text{if } i \in [n+1, 2n]. \end{cases}$$

We show that this setup fulfills (1).

► **Lemma 4.** $\left| \bigcap_{i \in [2n]} A_{w,\ell,i} \right| = |\mathcal{M}_{w,\ell}|.$

Proof. Let $(E_1, E_2, L) \in \bigcap_{i \in [2n]} A_{w,\ell,i}$. By definition of $A_{w,\ell,i}$ we have $V \subseteq V(E_1) \cup L$ and $V \subseteq V(E_2) \cup L$. Since $|E_1| = |E_2| = \frac{\ell}{2}$, we have $|V(E_1)| \leq \ell$ and $|V(E_2)| \leq \ell$. Since furthermore $|L| = n - \ell$ and $|V| = n$, it follows that $V(E_1) = V(E_2) = V \setminus L$ and $|V(E_1)| = |V(E_2)| = \ell$. Hence, E_1 and E_2 are matchings on the same vertex set. Thus, all triplets in $\bigcap_{i \in [2n]} A_{w,\ell,i}$ consist of two such matchings together with the set L of $n - \ell$ uncovered vertices. Since $w = \mathbf{w}(E_1, E_2)$, we have $(E_1, E_2) \in \mathcal{M}_{w,\ell}$. Thus, for all $(E_1, E_2, L) \in \bigcap_{i \in [2n]} A_{w,\ell,i}$ we have $(E_1, E_2) \in \mathcal{M}_{w,\ell}$. For any two different $(E_1, E_2, L), (E'_1, E'_2, L') \in \bigcap_{i \in [2n]} A_{w,\ell,i}$ we must have $E_1 \neq E'_1$ or $E_2 \neq E'_2$, or else by the above it follows that $L = L'$. But then such different triplets correspond to different pairs (E_1, E_2) and (E'_1, E'_2) in $\mathcal{M}_{w,\ell}$, which implies that $\left| \bigcap_{i \in [2n]} A_{w,\ell,i} \right| \leq |\mathcal{M}_{w,\ell}|.$

Now, let $(M_1, M_2) \in \mathcal{M}_{w,\ell}$. Let $L := V \setminus V(M_1)$. Clearly, $L = V \setminus V(M_2)$ and $|L| = n - \ell$. Moreover, for each $i \in V$ we have both $i \in V(M_1) \cup L$ and $i \in V(M_2) \cup L$. Hence $(M_1, M_2, L) \in \bigcap_{i \in [2n]} A_{w,\ell,i}$. Here, for different $(M_1, M_2), (M'_1, M'_2) \in \mathcal{M}_{w,\ell}$ it is obvious that the corresponding triplets (M_1, M_2, L) and (M'_1, M'_2, L') in $\bigcap_{i \in [2n]} A_{w,\ell,i}$ are different. Hence $|\mathcal{M}_{w,\ell}| \leq \left| \bigcap_{i \in [2n]} A_{w,\ell,i} \right|$. This completes the proof. ◀

Using a well-known variation of the inclusion-exclusion principle, expressing the size of the intersection by an alternating sum of sizes of intersections of complements (cf. [8]), we arrive at the following expression for $|\mathcal{M}_{w,\ell}|$; here $\bar{A}_{w,\ell,i} = U_{w,\ell} \setminus A_{w,\ell,i}$.

$$|\mathcal{M}_{w,\ell}| = \left| \bigcap_{i \in [2n]} A_{w,\ell,i} \right| = \sum_{I \subseteq [2n]} (-1)^{|I|} \left| \bigcap_{i \in I} \bar{A}_{w,\ell,i} \right| \quad (2)$$

4 Computing $|\mathcal{M}_{w,\ell}|$ for graphs of small treedepth

In this section, we show how to compute $|\mathcal{M}_{w,\ell}|$ in $4^\tau \cdot n^{\mathcal{O}(1)}$ time and polynomial space when the edge weights are polynomially bounded and given an elimination forest $\mathcal{T}$ of depth τ for the graph. Our algorithm relies on equation (2) for $|\mathcal{M}_{w,\ell}|$. Since we do not have time

proportional to 2^{2n} available, we will evaluate the contributions to the right-hand side while following the structure of $\mathcal{T}$. To this end, we will use polynomials in few indeterminates to conveniently handle different (partial) contributions like in previous work [15, 23]. Zoomed out, it looks a bit like the algorithm of Nederlof et al. [23] but we use inclusion-exclusion rather than cut-and-count for the main computation.

Throughout this section, let $G = (V, E)$ be a connected graph with vertex set $V = \{1, \dots, n\}$, let $\mathbf{w}: E \rightarrow \{1, \dots, N\}$ with $N = n^{\mathcal{O}(1)}$, and let $w, \ell \in \mathbb{N}$ with ℓ even and $\ell \leq n$. Moreover, let $\mathcal{T} = (V, E_{\mathcal{T}})$ be an elimination tree of G of depth τ and with root r . We show how to compute $|\mathcal{M}_{w, \ell}|$ in time $4^{\tau} \cdot n^{\mathcal{O}(1)}$ and polynomial space. It is straightforward to then use this approach to compute $|\mathcal{M}_{w, \ell}|$ for disconnected graphs. (If so desired, the algorithm works for larger weights at the cost of a factor N appearing in the running time.)

Additional notation. Some additional notation will be helpful to handle the fact that in equation (2) there are two possible requirements for each vertex. Accordingly, for $W \subseteq V$ we let $W^+ := \{v, v + n \mid v \in W\}$. We also extend this to $\text{tail}(v)$ etc., e.g., $\text{tail}^+(v) := \{u, u + n \mid u \in \text{tail}(v)\}$. Furthermore, we define **tpl**, **mon**, and **dsj**, which are short for tuple, monomial, and disjoint:

$$\begin{aligned} \text{tpl}(W) &:= \{(E_1, E_2, L) \mid E_1 \subseteq \delta[W], E_2 \subseteq \delta[W], L \subseteq W, E_1 \cap E_2 = \emptyset\} \\ \text{mon}(E_1, E_2, L) &:= x^{|E_1|} y^{|E_2|} z^{|L|} \omega^{\mathbf{w}(E_1 \cup E_2)} \\ \text{dsj}(I) &:= \{(E_1, E_2, L) \mid \forall \{s, t\} \in E_1 : \{s, t\} \cap I = \emptyset, \\ &\quad \forall \{s, t\} \in E_2 : \{s + n, t + n\} \cap I = \emptyset, L^+ \cap I = \emptyset\} \end{aligned}$$

We will use, e.g., $\text{tpl}(\text{tree}(v))$ to succinctly refer to the possible parts of tuples in $U_{w, \ell}$ whose edges have at least one endpoint in $\text{tree}(v)$; the weight is tracked through the arising polynomials (to be defined in a moment). Using $\text{mon}(E_1, E_2, L)$ we get a monomial whose degrees track the cardinalities of E_1 , E_2 , and L as well as the total weight of edges in $E_1 \cup E_2$ (which will always be disjoint). Using $(E_1, E_2, L) \in \text{dsj}(I)$ we express whether the tuple in question would be eligible for contributing for a given $I \subseteq \text{broom}^+[v]$, which corresponds to a selection of requirements in equation (2). (Note that $I \subseteq \text{broom}^+[v]$ will always be implicit through combination of subsets of $\text{tail}^+(v)$ and $\text{tree}^+[v]$ resp. $\text{tail}^+[v]$ and $\text{tree}^+(v)$.)

Two polynomials. We are now ready to define two types of polynomials that will be used in our branching algorithm. For $v \in V$ and $J \subseteq \text{tail}^+(v)$ let $P_{(v)}(J) \in \mathbb{Z}[x, y, z, \omega]$ with

$$P_{(v)}(J) = \sum_{\substack{(E_1, E_2, L) \in \\ \text{tpl}(\text{tree}[v])}} \text{mon}(E_1, E_2, L) \sum_{\substack{I = J \cup K \\ K \subseteq \text{tree}^+[v]}} (-1)^{|K|} \mathbb{I}[(E_1, E_2, L) \in \text{dsj}(I)]$$

Similarly, for $v \in V$ and $J' \subseteq \text{tail}^+[v]$ let $Q_{[v]}(J') \in \mathbb{Z}[x, y, z, \omega]$ with

$$Q_{[v]}(J') = \sum_{\substack{(E'_1, E'_2, L') \in \\ \text{tpl}(\text{tree}(v))}} \text{mon}(E'_1, E'_2, L') \sum_{\substack{I' = J' \cup K' \\ K' \subseteq \text{tree}^+(v)}} (-1)^{|K'|} \mathbb{I}[(E'_1, E'_2, L') \in \text{dsj}(I')]$$

Intuitively, these capture the contributions of partial tuples based on $\text{tree}[v]$ resp. $\text{tree}(v)$. The inner sums mimic the shape of equation (2). The difference between the two types is that $P_{(v)}(J)$ uses $J \subseteq \text{tail}^+(v)$, so does not specify a choice of v and/or $v + n$ to be included in J , whereas this is included in $Q_{[v]}(J)$ with $J \subseteq \text{tail}^+[v]$. It is crucial that the edges with exactly one endpoint in $\text{tree}[v]$ resp. $\text{tree}(v)$ have their other endpoint in $\text{tail}(v)$ resp. $\text{tail}[v]$ so that consistency with $I \subseteq \text{broom}^+[v]$ can be checked.

We now establish recurrences and base cases that hold for these polynomials. We begin with the base case of $Q_{[v]}(J')$ for leaf nodes v of $\mathcal{T}$. Then we prove recurrences that relate $Q_{[v]}(J)$ to $P_{(u_1)}(J), \dots, P_{(u_o)}(J)$ for internal nodes v with $\text{children}(v) = \{u_1, \dots, u_o\}$, and for relating $P_{(v)}(J)$ to $Q_{[v]}(J')$ (for arbitrary nodes). We then show how to get $|\mathcal{M}_{w,\ell}|$ from $P_{(r)}(\emptyset)$. When spelling out the algorithm, it is crucial to work directly with the coefficients of the polynomials, but here their definitions via sums over tuples are more convenient.

Base case for $Q_{[v]}$ for leaf nodes. Let v be a leaf node of $\mathcal{T}$ and let $J' \subseteq \text{tail}^+[v]$. Since $\text{tree}(v) = \emptyset$ we have $\text{tpl}(\text{tree}(v)) = \{(\emptyset, \emptyset, \emptyset)\}$ so the outer sum has only one summand. Similarly, $\text{tree}^+(v) = \emptyset$, so $K' = \emptyset$ is the only choice for the inner sum. Clearly, for the unique choice $(\emptyset, \emptyset, \emptyset) \in \text{tpl}(\text{tree}(v))$ we get the summand $\text{mon}(\emptyset, \emptyset, \emptyset) = x^0 y^0 z^0 \omega^0 = 1$ since $(-1)^{|K'|} = 1$ and $(\emptyset, \emptyset, \emptyset) \in \text{dsj}(I')$. Thus, $Q_{[v]}(J') = 1$.

Recurrence for $Q_{[v]}$ for internal nodes. Let v be an internal node of $\mathcal{T}$, let $J' \subseteq \text{tail}^+[v]$, and let $\text{children}(v) = \{u_1, \dots, u_o\}$. We show that $Q_{[v]}(J')$ is simply the product of the polynomials $P_{(u_i)}(J')$ of the child nodes of v . (In hindsight this includes the case of leaf nodes as a special case, but it is good to treat that one explicitly.)

► **Lemma 5.**

$$Q_{[v]}(J') = \prod_{i \in [o]} P_{(u_i)}(J').$$

Proof. We start from the right-hand side of the equation and as a first step plug in the definition of $P_{(u_i)}(J')$:

$$\begin{aligned} & \prod_{i \in [o]} P_{(u_i)}(J') \\ &= \prod_{i \in [o]} \sum_{\substack{(E_1^i, E_2^i, L^i) \in \\ \text{tpl}(\text{tree}[u_i])}} \text{mon}(E_1^i, E_2^i, L^i) \sum_{\substack{I^i = J' \cup K^i \\ K^i \subseteq \text{tree}^+[u_i]}} (-1)^{|K^i|} \llbracket (E_1^i, E_2^i, L^i) \in \text{dsj}(I^i) \rrbracket \end{aligned}$$

We rewrite the expression by expanding out the product over $i \in [o]$:

$$\begin{aligned} &= \sum_{\substack{(E_1^1, E_2^1, L^1) \in \text{tpl}(\text{tree}[u_1]) \\ \vdots \\ (E_1^o, E_2^o, L^o) \in \text{tpl}(\text{tree}[u_o])}} \text{mon}(E_1^1, E_2^1, L^1) \cdot \dots \cdot \text{mon}(E_1^o, E_2^o, L^o) \\ &\quad \cdot \sum_{\substack{I^1 = J' \cup K^1 \\ K^1 \subseteq \text{tree}^+[u_1] \\ \vdots \\ I^o = J' \cup K^o \\ K^o \subseteq \text{tree}^+[u_o]}} (-1)^{|K^1| + \dots + |K^o|} \llbracket (E_1^1, E_2^1, L^1) \in \text{dsj}(I^1) \rrbracket \dots \llbracket (E_1^o, E_2^o, L^o) \in \text{dsj}(I^o) \rrbracket \end{aligned}$$

Note that the sets $\text{tree}^+[u_i]$, with $i \in [o]$, are a partition of $\text{tree}^+(v)$ since $\text{children}(v) = \{u_1, \dots, u_o\}$. We use this along with the structure of an elimination tree and the definitions of $\text{tpl}(\text{tree}[u_i])$ and $\text{dsj}(I^i)$ to see that we arrive at the definition of $Q_{[v]}(J')$.

We begin with replacing the inner sum by the summation over $I' = J' \cup K'$ with $K' \subseteq \text{tree}^+(v)$ as it appears in $Q_{[v]}(J')$. The following arguments show that this is correct:

- Intuitively, we replace $K^1, \dots, K^o$ with $K^i \subseteq \text{tree}^+[u_i]$ by $K' = K^1 \cup \dots \cup K^o \subseteq \text{tree}^+(v)$. Since the sets $\text{tree}^+[u_i]$ are a partition of $\text{tree}^+(v)$ this is one-to-one. Moreover, we have $(-1)^{|K^1| + \dots + |K^o|} = (-1)^{|K'|}$.

- We replace $\llbracket (E_1^1, E_2^1, L^1) \in \mathbf{dsj}(I^1) \rrbracket \dots \llbracket (E_1^o, E_2^o, L^o) \in \mathbf{dsj}(I^o) \rrbracket$, which takes value 1 exactly when all o conditions are fulfilled (and zero otherwise) by

$$\llbracket (E_1^1 \cup \dots \cup E_1^o, E_2^1 \cup \dots \cup E_2^o, L^1 \cup \dots \cup L^o) \in \mathbf{dsj}(I') \rrbracket$$

where, as intended, $I' = J' \cup K' = J' \cup K^1 \cup \dots \cup K^o$. We check that these two terms give the same value, which is equivalent to checking that the conditions in brackets hold if and only if (recall that, in general, $\llbracket p \wedge q \rrbracket = \llbracket p \rrbracket \llbracket q \rrbracket$):

- Assume first that $(E_1^1 \cup \dots \cup E_1^o, E_2^1 \cup \dots \cup E_2^o, L^1 \cup \dots \cup L^o) \in \mathbf{dsj}(I')$ holds. By definition, this requires that $\forall \{s, t\} \in E_1^1 \cup \dots \cup E_1^o : \{s, t\} \cap I' = \emptyset$. Since $E_1^i \subseteq E_1^1 \cup \dots \cup E_1^o$ and $I^i = J' \cup K^i \subseteq J' \cup K' = I'$, it follows that $\forall \{s, t\} \in E_1^i : \{s, t\} \cap I^i = \emptyset$. The argument for $\forall \{s, t\} \in E_2^i : \{s + n, t + n\} \cap I^i = \emptyset$ is analogous. Finally, it is also required that $(L^1 \cup \dots \cup L^o)^+ \cap I' = \emptyset$. Since $L^i \subseteq L^1 \cup \dots \cup L^o$ and $I^i = J' \cup K^i \subseteq J' \cup K' = I'$, it follows that $(L^i)^+ \cap I^i = \emptyset$. Thus, $(E_1^i, E_2^i, L^i) \in \mathbf{dsj}(I^i)$ for all $i \in [o]$.
- Conversely, assume that $(E_1^i, E_2^i, L^i) \in \mathbf{dsj}(I^i)$ for all $i \in [o]$. Here it is crucial that the sets $\mathbf{tree}[u_i]$ for $i \in [o]$ are a partition of $\mathbf{tree}(v)$, and ditto for using $\mathbf{tree}^+$.
 - (1) We know that $\forall \{s, t\} \in E_1^i : \{s, t\} \cap I^i = \emptyset$; recall that $I^i = J' \cup K^i$. Let $j \neq i$ and let $\{s, t\} \in E_1^i$. Then $\{s, t\} \in \delta[\mathbf{tree}[u_o]]$ since $(E_1^i, E_2^i, L^i) \in \mathbf{tpl}(\mathbf{tree}[u_i])$. W.l.o.g., let $s \in \mathbf{tree}[u_i]$. Now $t \in \mathbf{tree}[u_i]$ too or $t \notin \mathbf{tree}[u_i]$. In the latter case, since $\mathcal{T}$ is an elimination $\mathcal{T}$ and t cannot be a descendant of s (or else $t \in \mathbf{tree}[s] \subseteq \mathbf{tree}[u_i]$), node t must be an ancestor of s ; then $t \in \mathbf{tail}(s) \setminus \mathbf{tree}[u_i] = \mathbf{tail}(u_i) = \mathbf{tail}[v]$. Either way, we have $K^j \subseteq \mathbf{tree}^+[u_j]$, so K^j is disjoint from $\mathbf{tree}^+[u_i] \supseteq \mathbf{tree}[u_i]$. It is also disjoint from $\mathbf{tail}(u_j) = \mathbf{tail}[v]$, so it contains neither s nor t . Thus, $\forall \{s, t\} \in E_1^i \forall j \neq i : \{s, t\} \cap K^j = \emptyset$. It follows that $\forall \{s, t\} \in E_1^i : \{s, t\} \cap I' = \emptyset$, since $I' = J' \cup K^1 \cup \dots \cup K^o$. And then this holds for all $\{s, t\} \in E_1^1 \cup \dots \cup E_1^o$.
 - (2) The argument for getting $\forall \{s, t\} \in E_2^1 \cup \dots \cup E_2^o : \{s + n, t + n\} \cap I' = \emptyset$ is fully analogous.
 - (3) We know that $(L^i)^+ \cap I^i = \emptyset$; recall that $I^i = J' \cup K^i$. Since $(E_1^i, E_2^i, L^i) \in \mathbf{tpl}(\mathbf{tree}[u_i])$, we have $L^i \subseteq \mathbf{tree}[u_i]$ and, hence, $(L^i)^+ \subseteq \mathbf{tree}^+[u_i]$. Now let $j \neq i$. Then $K^j \subseteq \mathbf{tree}^+[u_j]$ hence $(L^i)^+ \cap K^j \subseteq \mathbf{tree}^+[u_i] \cap \mathbf{tree}^+[u_j] = \emptyset$, as the sets $\mathbf{tree}^+[u_i]$ are a partition of $\mathbf{tree}^+(v)$. It follows that $(L^i)^+ \cap I' = (L^i)^+ \cap (J' \cup K^1 \cup \dots \cup K^o) = \emptyset$. Hence $(L^1 \cup \dots \cup L^o)^+ \cap I' = ((L^1)^+ \cup \dots \cup (L^o)^+) \cap I' = \emptyset$. Thus $(E_1^1 \cup \dots \cup E_1^o, E_2^1 \cup \dots \cup E_2^o, L^1 \cup \dots \cup L^o) \in \mathbf{dsj}(I')$.

Applying this replacement we arrive at

$$\begin{aligned}
 &= \sum_{\substack{(E_1^1, E_2^1, L^1) \in \mathbf{tpl}(\mathbf{tree}[u_1]) \\ \vdots \\ (E_1^o, E_2^o, L^o) \in \mathbf{tpl}(\mathbf{tree}[u_o])}} \mathbf{mon}(E_1^1, E_2^1, L^1) \cdot \dots \cdot \mathbf{mon}(E_1^o, E_2^o, L^o) \\
 &\quad \cdot \sum_{\substack{I' = J' \cup K' \\ K' \subseteq \mathbf{tree}^+(v)}} (-1)^{|K'|} \llbracket (E_1^1 \cup \dots \cup E_1^o, E_2^1 \cup \dots \cup E_2^o, L^1 \cup \dots \cup L^o) \in \mathbf{dsj}(I') \rrbracket
 \end{aligned}$$

Now we work on the outer summation. Since the inner sum only depends on, e.g., $E_1^1 \cup \dots \cup E_1^o$ but not on the separate sets, we can let the outer summation go over $(E_1, E_2, L) \in \mathbf{tpl}(v)$. The following arguments show that this is correct:

- The sets $\mathbf{tree}[u_1], \dots, \mathbf{tree}[u_o]$ are a partition of $\mathbf{tree}(v)$ as $\mathbf{children}(v) = \{u_1, \dots, u_o\}$. Moreover, there are no edges with endpoints in any two different sets $\mathbf{tree}[u_i]$ and $\mathbf{tree}[u_j]$, with $i \neq j$, by definition of an elimination tree. Thus, the sets $\delta[\mathbf{tree}[u_1]], \dots, \delta[\mathbf{tree}[u_o]]$ are a partition of $\delta[\mathbf{tree}(v)]$.

- First, this gives a canonical bijection between choices of o sets $L^1 \subseteq \text{tree}[u_1], \dots, L^o \subseteq \text{tree}[u_o]$ and single subsets $L \subseteq \text{tree}(v)$: Take union from former to latter, take projections $\text{tree}[u_1], \dots, \text{tree}[u_o]$ from latter to former.
- Second, we get an analogous bijection between choice of o pairs of disjoint subsets $E_1^1, E_2^1 \subseteq \delta[\text{tree}[u_1]], \dots, E_1^o, E_2^o \subseteq \delta[\text{tree}[u_o]]$ and single pairs of disjoint subsets $E_1, E_2 \subseteq \delta[\text{tree}(v)]$; union one way, projections the other.
- Together, we get a bijection between choices of o tuples

$$(E_1^1, E_2^1, L^1) \in \text{tpl}(\text{tree}[u_1]), \dots, (E_1^o, E_2^o, L^o) \in \text{tpl}(\text{tree}[u_o])$$

and single tuples $(E_1, E_2, L) \in \text{tpl}(\text{tree}(v))$. Thus, we get the same number of summands whether we go by the o tuples or by the single tuple. It remains to check that the values are the same along our bijection.

- Compare $\text{mon}(E_1^1, E_2^1, L^1) \cdot \dots \cdot \text{mon}(E_1^o, E_2^o, L^o)$ and $\text{mon}(E_1, E_2, L)$ for corresponding choices. It is easy to see that the product of o monomials on the left is exactly the single monomial on the right. E.g., we have that $E_1^1, \dots, E_1^o$ are a partition of E_1 , so either way we arrive at $x^{|E_1|}$ in the resulting monomial.
- In the inner sum, it is clear that nothing changes when we replace, e.g., $E_1^1 \cup \dots \cup E_1^o$ by E_1 , since these are identical (along our bijection).

With this second replacement we arrive at the definition of $Q_{[v]}(J')$, as desired.

$$\begin{aligned} &= \sum_{\substack{(E_1, E_2, L) \in \\ \text{tpl}(\text{tree}(v))}} \text{mon}(E_1, E_2, L) \sum_{\substack{I' = J' \cup K' \\ K' \subseteq \text{tree}^+(v)}} (-1)^{|K'|} \llbracket (E_1, E_2, L) \in \text{dsj}(I') \rrbracket \\ &= Q_{[v]}(J') \end{aligned}$$

This completes the proof. ◀

Recurrence for $P_{(v)}$. Now let v be an arbitrary node of $\mathcal{T}$ and let $J \subseteq \text{tail}^+(v)$. We show that $P_{(v)}(J)$ can be expressed using sums and products of polynomials, mainly certain $Q_{[v]}(J')$. Intuitively, going from $Q_{[v]}(J')$ to $P_{(v)}(J)$ drops the fixed behavior of v (as given by $J' \subseteq \text{tail}^+[v]$) and brings into scope the edges incident with v that were not previously considered, i.e., the set $E(v, \text{tail}(v))$. In the recurrence, the contributions of these edges, along with the contribution for v and $v+n$, have to be accounted for. Intuitively, factors such as $(1 + z \llbracket J_v = \emptyset \rrbracket)$ correspond to the option of adding, e.g., vertex v to a tuple: If not added, then this leaves the corresponding monomials unchanged (factor 1). If added (which requires some condition in $\llbracket \cdot \rrbracket$ to hold, in this case $J_v = \emptyset$) then this also creates modified monomials corresponding to this (factor z in this case). Similar factors correspond to the edges $e \in E(v, \text{tail}(v))$.

► **Lemma 6.**

$$\begin{aligned} P_{(v)}(J) &= \sum_{\substack{J' = J \cup J_v \\ J_v \subseteq \{v, v+n\}}} (-1)^{|J_v|} \cdot Q_{[v]}(J') \cdot (1 + z \llbracket J_v = \emptyset \rrbracket) \\ &\quad \cdot \prod_{\substack{e = \{u, v\} \\ e \in E(v, \text{tail}(v))}} 1 + x\omega^{\mathbf{w}(e)} \llbracket \{u, v\} \cap J' = \emptyset \rrbracket + y\omega^{\mathbf{w}(e)} \llbracket \{u+n, v+n\} \cap J' = \emptyset \rrbracket \end{aligned}$$

Getting $|\mathcal{M}_{w, \ell}|$ from $P_{(r)}(\emptyset)$. We now show how to get $|\mathcal{M}_{w, \ell}|$ from $P_{(r)}(J)$, specifically from $P_{(r)}(\emptyset)$ since that is the only possible choice for $J \subseteq \text{tail}^+(r) = \emptyset$. Essentially, we just need a specific coefficient of $P_{(r)}(\emptyset)$ depending on w and ℓ .

► **Lemma 7.** Let $P_{(r)}(\emptyset) = \sum_{a,b,c,d} \alpha_{a,b,c,d} \cdot x^a y^b z^c \omega^d$. Then $|\mathcal{M}_{w,\ell}| = \alpha_{\ell/2,\ell/2,n-\ell,w}$.

Proof. We recall the definition of $P_{(v)}(J)$ for $v \in V$ and $J \subseteq \text{tail}^+(v)$ as

$$P_{(v)}(J) = \sum_{\substack{(E_1, E_2, L) \in \\ \text{tpl}(\text{tree}[v])}} \text{mon}(E_1, E_2, L) \sum_{\substack{I=J \cup K \\ K \subseteq \text{tree}^+[v]}} (-1)^{|K|} \llbracket (E_1, E_2, L) \in \text{dsj}(I) \rrbracket.$$

Generally for $P_{(v)}(J)$, each $(E_1, E_2, L) \in \text{tpl}(\text{tree}[v])$ contributes

$$\alpha_{(E_1, E_2, L)} \cdot \text{mon}(E_1, E_2, L) = \alpha_{(E_1, E_2, L)} \cdot x^{|E_1|} y^{|E_2|} z^{|L|} \omega^{\mathbf{w}(E_1 \cup E_2)}$$

where

$$\alpha_{(E_1, E_2, L)} = \sum_{\substack{I=J \cup K \\ K \subseteq \text{tree}^+[v]}} (-1)^{|K|} \llbracket (E_1, E_2, L) \in \text{dsj}(I) \rrbracket.$$

Thus, any specific coefficient $\alpha_{a,b,c,d}$ is equal to the sum of $\alpha_{(E_1, E_2, L)}$ over all $(E_1, E_2, L) \in \text{tpl}(\text{tree}[v])$ with $\text{mon}(E_1, E_2, L) = x^a y^b z^c \omega^d$.

We now consider $P_{(r)}(\emptyset)$, i.e., $v = r$ and $\emptyset = J \subseteq \text{tail}^+(r) = \emptyset$, and apply this for $\alpha_{\ell/2,\ell/2,n-\ell,w}$:

$$\begin{aligned} \alpha_{\ell/2,\ell/2,n-\ell,w} &= \sum_{\substack{(E_1, E_2, L) \in \text{tpl}(\text{tree}[r]) \\ \text{mon}(E_1, E_2, L) = x^{\ell/2} y^{\ell/2} z^{n-\ell} \omega^w}} \sum_{\substack{I=J \cup K \\ K \subseteq \text{tree}^+[v]}} (-1)^{|K|} \llbracket (E_1, E_2, L) \in \text{dsj}(I) \rrbracket \\ &= \sum_{\substack{(E_1, E_2, L) \in \text{tpl}(V) \\ \text{mon}(E_1, E_2, L) = x^{\ell/2} y^{\ell/2} z^{n-\ell} \omega^w}} \sum_{I \subseteq [2n]} (-1)^{|I|} \llbracket (E_1, E_2, L) \in \text{dsj}(I) \rrbracket \end{aligned}$$

Note that $\text{tree}[r] = V$, $\text{tree}^+[r] = V^+ = [2n]$, and $I = K$ in the inner sum as $J = \emptyset$.

We observe that the outer sum is in fact over $(E_1, E_2, L) \in U_{w,\ell}$:

■ Recall that

$$U_{w,\ell} = \left\{ (E_1, E_2, L) \mid E_1, E_2 \in \binom{E}{\ell/2}, E_1 \cap E_2 = \emptyset, \mathbf{w}(E_1 \cup E_2) = w, L \in \binom{V}{n-\ell} \right\}.$$

■ Recall also the definition of $\text{tpl}(W)$, which can be simplified using $W = V$,

$$\begin{aligned} \text{tpl}(V) &= \{(E_1, E_2, L) \mid E_1 \subseteq \delta[V], E_2 \subseteq \delta[V], L \subseteq V, E_1 \cap E_2 = \emptyset\} \\ &= \{(E_1, E_2, L) \mid E_1 \subseteq E, E_2 \subseteq E, L \subseteq V, E_1 \cap E_2 = \emptyset\}. \end{aligned}$$

■ Finally, by definition of $\text{mon}(E_1, E_2, L)$, the second restriction of the sum is equivalent to $|E_1| = |E_2| = \frac{\ell}{2}$, $|L| = n - \ell$, and $\mathbf{w}(E_1 \cup E_2) = w$. Hence

$$\alpha_{\ell/2,\ell/2,n-\ell,w} = \sum_{(E_1, E_2, L) \in U_{w,\ell}} \sum_{I \subseteq [2n]} (-1)^{|I|} \llbracket (E_1, E_2, L) \in \text{dsj}(I) \rrbracket$$

We now observe that $(E_1, E_2, L) \in \text{dsj}(I)$ holds for $(E_1, E_2, L) \in U_{w,\ell}$ if and only if $(E_1, E_2, L) \in \bigcap_{i \in I} \bar{A}_{w,\ell,i}$:

■ Recall that $A_{w,\ell,i} \subseteq U_{w,\ell}$ for $i \in [2n]$ with

$$A_{w,\ell,i} := \begin{cases} \{(E_1, E_2, L) \in U_{w,\ell} \mid i \in V(E_1) \cup L\} & \text{if } i \in [n] \\ \{(E_1, E_2, L) \in U_{w,\ell} \mid i - n \in V(E_2) \cup L\} & \text{if } i \in [n+1, 2n]. \end{cases}$$

■ **Algorithm 1** `ComputeP` (v, J).

Input: $v \in V$ and $J \subseteq \text{tail}^+(v)$
Output: $P_{(v)}(J)$
 $P = 0$;
for $J_v \subseteq \{v, v+n\}$ **do** // outer sum in Lemma 6
 $J' = J \cup J_v$;
 $\text{summand} = (-1)^{|J_v|}$;
 if $J_v = \emptyset$ **then** $\text{summand} = \text{summand} \cdot (1+z)$;
 for $e = \{u, v\} \in E(v, \text{tail}(v))$ **do** // summand for inner product
 $\text{factor} = 1$;
 if $\{u, v\} \cap J' = \emptyset$ **then** $\text{factor} = \text{factor} + x\omega^{\mathbf{w}(e)}$;
 if $\{u+n, v+n\} \cap J' = \emptyset$ **then** $\text{factor} = \text{factor} + y\omega^{\mathbf{w}(e)}$;
 $\text{summand} = \text{summand} \cdot \text{factor}$;
 // multiplication with $Q_{[v]}(J')$, step by step using Lemma 5,
 implicit multiplication by $Q_{[v]}(J') = 1$ if v is a leaf
 for $u \in \text{children}(v)$ **do**
 $\text{summand} = \text{summand} \cdot \text{ComputeP}(u, J')$;
 $P = P + \text{summand}$;
return P ;

- Hence, $(E_1, E_2, L) \in \bar{A}_{w, \ell, i}$ is equivalent to $i \notin V(E_1) \cup L$ if $i \in [n]$ and to $i-n \notin V(E_2) \cup L$ if $i \in [2n]$. Then $(E_1, E_2, L) \in \bigcap_{i \in I} \bar{A}_{w, \ell, i}$ if and only if this holds for all $i \in I$. This in turn is equivalent to $(E_1, E_2, L) \in \text{dsj}(I)$, which was defined as

$$\begin{aligned} \text{dsj}(I) := \{ (E_1, E_2, L) \mid & \forall \{s, t\} \in E_1 : \{s, t\} \cap I = \emptyset, \\ & \forall \{s, t\} \in E_2 : \{s+n, t+n\} \cap I = \emptyset, \\ & L^+ \cap I = \emptyset \}. \end{aligned}$$

This allows us to further transform our equation, starting with changing the order of summation:

$$\begin{aligned} \alpha_{\ell/2, \ell/2, n-\ell, w} &= \sum_{I \subseteq [2n]} (-1)^{|I|} \sum_{(E_1, E_2, L) \in U_{w, \ell}} \mathbb{I}[(E_1, E_2, L) \in \text{dsj}(I)] \\ &= \sum_{I \subseteq [2n]} (-1)^{|I|} \sum_{(E_1, E_2, L) \in U_{w, \ell}} \mathbb{I}[(E_1, E_2, L) \in \bigcap_{i \in I} \bar{A}_{w, \ell, i}] \\ &= \sum_{I \subseteq [2n]} (-1)^{|I|} \left| \bigcap_{i \in I} \bar{A}_{w, \ell, i} \right| \\ &= |\mathcal{M}_{w, \ell}| \end{aligned}$$

This completes the proof. ◀

The algorithm. While it was convenient for proving the recurrences to define $P_{(v)}(J)$ and $Q_{[v]}(J')$ as sums over suitable triples (of essentially partial solutions), our algorithm unsurprisingly uses their representation via coefficients as polynomials in $\mathbb{Z}[x, y, z, \omega]$. (Also we just saw that the coefficients carry the information that we are after, e.g., $|\mathcal{M}_{w, \ell}|$.) Similar

to previous work (e.g., [15]), our algorithm proceeds by recursively computing the two types of polynomials using the established recurrences. We nevertheless spell out the pseudocode to more easily establish time and space complexity (see Algorithm 1). To this end, we also roll the computation of $Q_{[v]}(J')$ into the computation of $P_{(v)}(J)$ rather than making this a separate recursive call.

Time and space complexity. All polynomials are represented via their coefficients as polynomials in $\mathbb{Z}[x, y, z, \omega]$. It can be checked that the degrees of x and y are bounded by $m = |E|$, the degree of z is bounded by n , and the degree of ω is bounded by the total weight $\mathbf{w}(E)$ of all edges, which is at most $m \cdot N$. Using the definitions of $P_{(v)}(J)$ and $Q_{[v]}(J')$ it can also be checked that the absolute value of each coefficient is at most $2^m \cdot 2^m \cdot 2^n \cdot 4^n$ (by using trivial upper bounds for the number of terms in the summation), which gives polynomial-size binary encoding per coefficient. Overall, for N polynomial in the input size, we use polynomial-size per polynomial. The recursion depth is bounded by τ and each call internally uses at most three polynomials (namely P , *summand*, and *factor*). The space complexity is therefore bounded by $\mathcal{O}(\tau \cdot n^{\mathcal{O}(1)})$ when N is polynomially bounded.

Clearly, each single call to **ComputeP** can be implemented to run in polynomial time (optionally with fast polynomial multiplication, especially when using a large weight bound N) when not counting the recursion. Observe that for each node v there are $2^{\text{tail}^+(v)} \leq 4^\tau$ possible calls involving v and it can be verified that there are no repeated calls. Thus, the algorithm runs in time $4^\tau n^{\mathcal{O}(1)}$.

► **Theorem 8.** *Given a graph $G = (V, E)$ with n vertices, edge weights $\mathbf{w}: E \rightarrow \{1, \dots, N\}$ with $N = n^{\mathcal{O}(1)}$, values $w, \ell \in \mathbb{N}$, and an elimination forest $\mathcal{T}$ for G of depth τ , the value $|\mathcal{M}_{w, \ell}|$ can be computed in time $4^\tau \cdot n^{\mathcal{O}(1)}$ and polynomial space.*

Proof. If $\ell > n$, or ℓ not even, or $w > n \cdot N$ we can return 0 as the correct answer. If G is disconnected then a simple DP lets us compute for all i , w' , and ℓ' the value $|\mathcal{M}_{w', \ell'}|$ for the graph induced by the first i connected components of G by using the algorithm for the connected case on each connected component (and suitable values $\hat{w}, \hat{\ell}$). This causes only a polynomial in n overhead in the running time. The algorithm for the connected case and even $\ell \leq n$ was explained above. ◀

5 An algorithm for partial cycle cover

Using our algorithm for computing $|\mathcal{M}_{w, \ell}|$, given graph $G = (V, E)$, $w, \ell \in \mathbb{N}$, and an elimination forest $\mathcal{T}$ of depth τ of G , only little work remains for solving **PARTIAL CYCLE COVER**. Similar to cut and count we also rely on symmetries of (non-)solutions having too many cycles plus the isolation lemma to (probabilistically) guarantee a unique solution (if one exists). Differently (and closer to Nederlof et al. [23] who also use matchings) we use that the edges of an even cycle have two ways of being partitioned into an ordered pair of disjoint matchings covering the same vertex sets (i.e., ordered pairs of consistent matchings). Thus, non-solutions with too many even cycles can be filtered out by counting modulo a suitable power of 2. Since partial cycle covers may also contain odd cycles, however, we need to first use a simple trick to reduce to the bipartite case, where the above idea is sufficient.

► **Lemma 9.** *Let G be a graph, let $k, \ell \in \mathbb{N}$, and let $\mathcal{T}$ be an elimination forest of depth τ of G . Let G' be obtained from G by subdividing all its edges and let $\ell' = 2\ell$. Then G has the same number of partial cycle covers with at most k cycles covering exactly ℓ vertices as G' has partial cycle covers with at most k cycles covering exactly ℓ' vertices. Moreover, an elimination tree $\mathcal{T}'$ of depth at most $\tau + 1$ of G' can be efficiently constructed from $\mathcal{T}$.*

Proof. Formally, let $G = (V, E)$ and let $G' = (V', E')$ with $V' = V \cup \{p_e \mid e \in E\}$ and $E' = \{\{p_e, v\} \mid e \in E, v \in e\}$. Let $\mathcal{C}_{k,\ell}$ denote the family of edge sets of partial cycle covers with at most k cycles and exactly ℓ vertices in G . Similarly, let $\mathcal{C}'_{k,\ell'}$ denote the same for G' , with at most k cycles and exactly $\ell' = 2\ell$ vertices. There is a one-to-one correspondence between elements of $\mathcal{C}_{k,\ell}$ and those of $\mathcal{C}'_{k,\ell'}$: Simply do respectively undo the edge subdivisions on the edges present in the respective partial cycle cover. (E.g., a cycle (u, v, w, u) in G corresponds to $(u, p_{\{u,v\}}, v, p_{\{v,w\}}, w, p_{\{w,u\}}, u)$ in G' with exactly twice the length, and vice versa.) It follows that $|\mathcal{C}_{k,\ell}| = |\mathcal{C}'_{k,\ell'}|$, as claimed.

To see the moreover part, take $\mathcal{T}$ and for each $e = \{u, v\} \in E$ attach the subdividing vertex p_e as a new child node to whoever of u and v is lower in $\mathcal{T}$; crucially, u and v must be in ancestor-descendant-relation in $\mathcal{T}$ due to the edge $\{u, v\}$ in G . In this way, for each edge $\{s, p_{\{s,t\}}\} \in E'$ node s must be an ancestor of $p_{\{s,t\}}$. Call this tree $\mathcal{T}'$. Clearly, the depth of $\mathcal{T}'$ is at most $\tau + 1$. $\blacktriangleleft$

Keeping the lemma in mind, the following theorem gives an algorithm for PARTIAL CYCLE COVER on bipartite graphs. The case of general graphs will be an immediate corollary.

► **Theorem 10.** *There is a randomized algorithm that, given a bipartite graph G , numbers $k, \ell \in \mathbb{N}$, and an elimination forest $\mathcal{T}$ of depth τ of G solves PARTIAL CYCLE COVER in $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space. The algorithm makes no false positives and may make a false negative with probability at most $\frac{1}{2}$.*

Proof. We again use $\mathcal{C}_{k,\ell}$ for the family of edge sets of partial cycles covers in G with at most k cycles and on exactly ℓ vertices. The algorithm needs to discover (up to the error chance) whether $\mathcal{C}_{k,\ell} \neq \emptyset$. Since G is bipartite, all occurring cycles are of even length.

Let $n = |V|$ and $m = |E|$. Choose $\mathbf{w}: E \rightarrow \{1, \dots, N\}$ with $N = 2|E| = \mathcal{O}(n^2)$ uniformly and independently at random. For each $w \in \{1, \dots, \ell N\}$, compute $|\mathcal{M}_{w,\ell}|$ in $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space. Output **yes** if $|\mathcal{M}_{w,\ell}| \not\equiv 0 \pmod{2^{k+1}}$ for any w ; otherwise output **no**. This completes the algorithm. Time and space complexity should be clear, let us argue absence of false positives and bound the probability of false negatives.

We begin with some observations about the family $\mathcal{M}_{w,\ell}$; let us recall its definition as

$$\mathcal{M}_{w,\ell} := \{(M_1, M_2) \mid M_1 \text{ and } M_2 \text{ are consistent matchings in } G \text{ of cardinality } \frac{\ell}{2} \text{ each, } V(M_1) = V(M_2), \mathbf{w}(M_1 \cup M_2) = w\}.$$

For each $(M_1, M_2) \in \mathcal{M}_{w,\ell}$, the two matchings M_1 and M_2 are consistent. Hence their union is a partial cycle cover (with each vertex in $V(M_1) = V(M_2)$ having degree two and all other vertices having degree zero). Obviously $|V(M_1)| = |V(M_2)| = \ell = 2|M_1| = 2|M_2|$, i.e., these partial cycle covers visit exactly ℓ vertices each. Similar to previous work [23] we use the symmetry in partial cycles covers with more than k cycles for cancellation. Presently, a partial cycle cover with exactly p even cycles on exactly ℓ vertices corresponds to 2^p ordered pairs of consistent matchings on (the same) ℓ vertices: Let $C_1, \dots, C_p$ denote the edge sets of the p even cycles. For each C_i there are two choices for putting half of its edges into the first matching and the other half into the second. Overall, this yields 2^p ordered pairs of consistent matchings per edge set of any partial cycle cover with p (even) cycles. Let us add a few observations about these pairs:

- They all consist of two consistent matchings of cardinality $\frac{\ell}{2}$ each.
- The weight of each pair is equal to the weight of the corresponding partial cycle cover.
- Thus, if any pair (M_1, M_2) is in $\mathcal{M}_{w,\ell}$ then all 2^p pairs corresponding to the partial cycle cover $M_1 \cup M_2$ are contained in $\mathcal{M}_{w,\ell}$.

It follows that the contribution of partial cycle covers on ℓ vertices but with more than k cycles is congruent 0 modulo 2^{k+1} , i.e., it does not affect the outcome of the algorithm.

Let us assume first that G has no partial cycle cover of at most k cycles on exactly ℓ vertices. Then $|\mathcal{C}_{k,\ell}| = 0$. Since every pair $(M_1, M_2) \in \mathcal{M}_{w,\ell}$ corresponds to a partial cycle cover on exactly ℓ vertices in G but not contained in $\mathcal{C}_{k,\ell} = \emptyset$, all those partial cycle covers have more than k cycles. But then $|\mathcal{M}_{w,\ell}| \equiv 0 \pmod{2^{k+1}}$ so the algorithm will return **no** irrespective of the (random) choice of $\mathbf{w}$. In other words, there are no false positives.

Let us now assume that G has at least one partial cycle cover of at most k cycles on exactly ℓ vertices. Then $|\mathcal{C}_{k,\ell}| \geq 1$. By the isolation lemma it follows that with probability at least $\frac{1}{2}$ there is a unique element of $\mathcal{C}_{k,\ell}$ of minimum weight. Assuming that this succeeds, we show that the algorithm will return **yes**, so a false negative has probability at most $\frac{1}{2}$. Let $C^* \in \mathcal{C}_{k,\ell}$ be of unique minimum weight $w = \mathbf{w}(C^*)$. Thus, in $\mathcal{M}_{w,\ell}$ we have two types of pairs (M_1, M_2) of consistent matchings:

- Those pairs that correspond to C^* , i.e., with $M_1 \cup M_2 = C^*$. There are exactly 2^p of them, where $p \leq k$ is the number of cycles in C^* . Their number is not congruent to 0 modulo 2^{k+1} .
- All other pairs (M_1, M_2) . As observed above, their union is a partial cycle cover on exactly ℓ vertices (and weight w). They do not correspond to C^* , which is unique among these partial cycle covers with weight w and at most k cycles, so their corresponding partial cycle covers have more than k cycles. We have already discussed that their number is congruent 0 modulo 2^{k+1} .

Thus, assuming successful isolation (which has probability at least $\frac{1}{2}$) the algorithm discovers $|\mathcal{M}_{w,\ell}| \not\equiv 0 \pmod{2^{k+1}}$ and returns **yes**. Thus, returning **no**, a false negative, has probability at most $\frac{1}{2}$. This completes the proof. ◀

Using Lemma 9, the extension to general graphs is now an easy corollary.

► **Corollary 11.** *The algorithmic result of Theorem 10 also holds for general graphs G , all other assumptions being the same.*

Proof. Let $G = (V, E)$ be a (general) graph, $k, \ell \in \mathbb{N}$, and $\mathcal{T}$ an elimination forest of depth τ of G . Construct the graph G' as in Lemma 9 so that the number of partial cycle covers with at most k cycles on exactly ℓ vertices in G is equal to the number of partial cycle covers with at most k cycles on exactly $\ell' = 2\ell$ vertices in G' . Let $\mathcal{T}'$ be the obtained elimination forest of depth at most $\tau + 1$ for G' ; in short $|\mathcal{C}_{k,\ell}| = |\mathcal{C}'_{k,\ell'}|$. Since G' is bipartite, we can use the algorithm of Theorem 10 to determine whether $\mathcal{C}'_{k,\ell'} \neq \emptyset$ (up to error bounds), and return the same answer for G and $k, \ell \in \mathbb{N}$. Since G' is only polynomially larger than G , and since $\mathcal{T}'$ has depth at most $\tau + 1$, we get the same asymptotic bounds of $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space (and same one-sided error bound). ◀

Altogether, this constitutes a proof of Theorem 1 in the introduction. The transfer to the problems listed in Corollary 2 follows as explained by Nederlof et al. [23].

6 Conclusion

We have given a $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space algorithm for solving the PARTIAL CYCLE COVER problem on graphs given with an elimination forest of depth at most τ , which directly improves the time bounds for several related problems obtained in previous work [23]. It is worth noting that the dependence on the treedepth is now the same as that relative to treewidth, but the treewidth-based DP algorithm relies on an instance of cut-and-count that does not seem to transfer easily to a branching algorithm relative to treedepth (cf. [15, 23]). Intuitively, the DP computes path packings with each path being red or blue (a known

perspective of cut-and-count), using the symmetries in possible colorings of disconnected solutions, but the role of vertices changes as we update partial solutions. This does not seem to work in the branching, polynomial-space setting used (so far) relative to treedepth.

With the $4^\tau n^{\mathcal{O}(1)}$ time and polynomial space, e.g., for HAMILTONIAN CYCLE, it is natural to ask about further improvements. Notably, the conditional lower bound of $(2 + \sqrt{2} - \varepsilon)^{\text{pw}} n^{\mathcal{O}(1)}$ (i.e., relative to pathwidth) [9] has no implications for the treedepth parameterization. Nevertheless, it is a natural and interesting question whether $(2 + \sqrt{2})^\tau n^{\mathcal{O}(1)}$ time and polynomial space are possible when given an elimination forest of depth τ . If even possible, this should be quite challenging as the DP algorithm relative to pathwidth is a much more involved version of that relative to treewidth. (To note, this DP could be adapted to work on tree decompositions as well, but the lack of a fast operation at join nodes makes it worse than $4^{\text{tw}} n^{\mathcal{O}(1)}$ time.) Generally, it is very interesting what (conditionally optimal) algorithmic results can be transferred from treewidth and pathwidth to work in the same time but polynomial space relative to treedepth. Similarly, much (if not more) remains to do regarding (conditional) lower bounds relative to treedepth.

References

- 1 Mahdi Belbasi and Martin Fürer. Saving space by dynamic algebraization based on tree decomposition: Minimum dominating set. *CoRR*, abs/1711.10088, 2017. [arXiv:1711.10088](#).
- 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, Vera Chekan, Robert Ganian, Mamadou Moustapha Kanté, Matthias Mnich, Sang-il Oum, Michal Pilipczuk, and Erik Jan van Leeuwen. Space-efficient parameterized algorithms on graphs of low shrubdepth. *ACM Trans. Comput. Theory*, 17(3):18:1–18:42, 2025. [doi:10.1145/3723880](#).
- 4 Benjamin Bergougnoux, Vera Chekan, and Giannos Stamoulis. A logic-based algorithmic meta-theorem for treedepth: Single exponential FPT time and polynomial space. In Kasper Green Larsen and Barna Saha, editors, *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, BC, Canada, January 11-14, 2026*, pages 2069–2098. SIAM, 2026. [doi:10.1137/1.9781611978971.75](#).
- 5 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](#).
- 6 Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. [doi:10.1016/0890-5401\(90\)90043-H](#).
- 7 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016. [doi:10.1145/2925416](#).
- 8 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](#).
- 9 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, 2018. [doi:10.1145/3148227](#).
- 10 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. *ACM Trans. Algorithms*, 18(2):17:1–17:31, 2022. [doi:10.1145/3506707](#).

- 11 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- 12 Andrew Drucker, Jesper Nederlof, and Rahul Santhanam. Exponential time paradigms through the polynomial time lens. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, Aarhus, Denmark, August 22-24, 2016*, volume 57 of *LIPIcs*, pages 36:1–36:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.36.
- 13 Baris Can Esmer, Jacob Focke, Dániel Marx, and Pawel Rżazewski. Fundamental problems on bounded-treewidth graphs: The real source of hardness. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, Tallinn, Estonia, July 8-12, 2024*, volume 297 of *LIPIcs*, pages 34:1–34:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICALP.2024.34.
- 14 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.
- 15 Falko Hegerfeld and Stefan Kratsch. Solving connectivity problems parameterized by treedepth in single-exponential time and polynomial space. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, Montpellier, France, March 10-13, 2020*, volume 154 of *LIPIcs*, pages 29:1–29:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.29.
- 16 Falko Hegerfeld and Stefan Kratsch. Towards exact structural thresholds for parameterized complexity. In Holger Dell and Jesper Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation, IPEC 2022, Potsdam, Germany, September 7-9, 2022*, volume 249 of *LIPIcs*, pages 17:1–17:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.IPEC.2022.17.
- 17 Lars Jaffke and Bart M. P. Jansen. Fine-grained parameterized complexity analysis of graph coloring problems. *Discret. Appl. Math.*, 327:33–46, 2023. doi:10.1016/J.DAM.2022.11.011.
- 18 Ioannis Koutis, Michal Włodarczyk, and Meirav Zehavi. Sidestepping barriers for dominating set in parameterized complexity. In Neeldhara Misra and Magnus Wahlström, editors, *18th International Symposium on Parameterized and Exact Computation, IPEC 2023, Amsterdam, The Netherlands, September 6-8, 2023*, volume 285 of *LIPIcs*, pages 31:1–31:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.IPEC.2023.31.
- 19 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.
- 20 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.
- 21 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987. doi:10.1007/BF02579206.
- 22 Wojciech Nadara, Michal Pilipczuk, and Marcin Smulewicz. Computing treedepth in polynomial space and linear FPT time. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, Berlin/Potsdam, Germany, September 5-9, 2022*, volume 244 of *LIPIcs*, pages 79:1–79:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.79.
- 23 Jesper Nederlof, Michal Pilipczuk, Céline M. F. Swennenhuis, and Karol Węgrzycki. Hamiltonian cycle parameterized by treedepth in single exponential time and polynomial space. *SIAM J. Discret. Math.*, 37(3):1566–1586, 2023. doi:10.1137/22M1518943.
- 24 Michal Pilipczuk and Marcin Wrochna. On space efficiency of algorithms working on structural decompositions of graphs. *ACM Trans. Comput. Theory*, 9(4):18:1–18:36, 2018. doi:10.1145/3154856.

- 25 Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 931–942. Springer, 2014. doi:10.1007/978-3-662-43948-7_77.
- 26 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*, Lecture Notes in Computer Science, pages 281–290. Springer, 2004. doi:10.1007/978-3-540-28639-4_25.

A Proof of Lemma 6

Proof. We first show how to rewrite part of the right-hand side using `mon` and `dsj`.

▷ **Claim 12.** For $J_v \subseteq \{v, v+n\}$ and $J' = J \cup J_v$ it holds that

$$\begin{aligned}
 & (1 + z \llbracket J_v = \emptyset \rrbracket) \prod_{\substack{e=\{u,v\} \\ e \in E(v, \text{tail}(v))}} 1 + x\omega^{\mathbf{w}(e)} \llbracket \{u, v\} \cap J' = \emptyset \rrbracket \\
 & \quad + y\omega^{\mathbf{w}(e)} \llbracket \{u+n, v+n\} \cap J' = \emptyset \rrbracket \\
 &= \sum_{\substack{\hat{E}_1 \subseteq E(v, \text{tail}(v)) \\ \hat{E}_2 \subseteq E(v, \text{tail}(v)) \\ \hat{L} \subseteq \{v\} \\ \hat{E}_1 \cap \hat{E}_2 = \emptyset}} \text{mon}(\hat{E}_1, \hat{E}_2, \hat{L}) \cdot \llbracket (\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J') \rrbracket.
 \end{aligned}$$

Proof. Intuitively, the RHS of the claimed equation is obtained by expanding out the product on the LHS but without gathering multiple copies of the same monomial $x^a y^b z^c \omega^d$ that may arise from $\text{mon}(\hat{E}_1, \hat{E}_2, \hat{L})$ over all choices of $\hat{E}_1$, $\hat{E}_2$, and $\hat{L}$ that fulfill $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$. Instead, there are separate summands for all combinations of choices.

LHS $\leq$ RHS: Consider a single occurrence of a monomial $x^a y^b z^c \omega^d$ in the expansion of the product on the LHS after dropping summands whose condition, e.g., $\{u, v\} \cap J' = \emptyset$, is not fulfilled (so they get factor $\llbracket \{u, v\} \cap J' = \emptyset \rrbracket = 0$). Clearly $c \in \{0, 1\}$. Let $\hat{L} = \{v\}$ if $c = 1$ and $\hat{L} = \emptyset$ otherwise (if $c = 0$); either way, $\hat{L} \subseteq \{v\}$. Observe that we can only have $c = 1$ if $J_v = \emptyset$ holds, or else $\llbracket J_v = \emptyset \rrbracket = 0$. Let $\hat{E}_1 \subseteq E(v, \text{tail}(v))$ contain those edges $e \in E(v, \text{tail}(v))$ where $x\omega^{\mathbf{w}(e)}$ was contributed to the monomial, which requires that $\{u, v\} \cap J' = \emptyset$ for these edges. Similarly, let $\hat{E}_2 \subseteq E(v, \text{tail}(v))$ contain those edges $e \in E(v, \text{tail}(v))$ where $y\omega^{\mathbf{w}(e)}$ was contributed to the monomial, which requires that $\{u+n, v+n\} \cap J' = \emptyset$ for these edges. Clearly, $\hat{E}_1 \cap \hat{E}_2 = \emptyset$ since for each $e \in E(v, \text{tail}(v))$ only one choice out of 1, $x\omega^{\mathbf{w}(e)}$, and $y\omega^{\mathbf{w}(e)}$ contributes to the monomial in the expansion of the product. Furthermore, since these are all possible contributions, we then have $x^a y^b z^c \omega^d = x^{|\hat{E}_1|} y^{|\hat{E}_2|} z^{|\hat{L}|} \omega^{\mathbf{w}(\hat{E}_1 \cup \hat{E}_2)} = \text{mon}(\hat{E}_1, \hat{E}_2, \hat{L})$. It remains to verify that $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$, so that $\llbracket (\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J') \rrbracket = 1$ and we get the same contribution of $\text{mon}(\hat{E}_1, \hat{E}_2, \hat{L})$ on the RHS (for this specific choice of $\hat{E}_1$, $\hat{E}_2$, and $\hat{L}$). The following conditions are required (and showed to hold) in order for $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$; recall that $J' = J \cup J_v$ and $J \subseteq \text{tail}^+(v)$:

- $\forall \{s, t\} \in \hat{E}_1 : \{s, t\} \cap J' = \emptyset$: Since $\hat{E}_1 \subseteq E(v, \text{tail}(v))$, let $\{s, t\} = \{u, v\}$. For all these edges we already observed that $\{u, v\} \cap J' = \emptyset$ for all edges in $\hat{E}_1$.
- $\forall \{s, t\} \in \hat{E}_2 : \{s+n, t+n\} \cap J' = \emptyset$: Similarly, let $\{s, t\} = \{u, v\}$. We similarly already observed that $\{u+n, v+n\} \cap J' = \emptyset$ for all edges in $\hat{E}_2$.

- $\hat{L}^+ \cap J' = \emptyset$: Since $\hat{L} \subseteq \{v\}$ we have $\hat{L}^+ \subseteq \{v, v+n\}$. Hence, $\hat{L}^+ \cap J' = \hat{L}^+ \cap J_v$. Recall that we can only have $c = 1$ and $\hat{L} = \{v\}$ when $J_v = \emptyset$; in this case $\hat{L}^+ \cap J_v = \emptyset$. Otherwise, we have $\hat{L} = \emptyset$ and, hence, $\hat{L}^+ = \emptyset$, so again $\hat{L}^+ \cap J_v = \emptyset$.

We see that for each single occurrence of any monomial in the expansion of the product on the LHS there is a unique (and private) choice of $\hat{E}_1$, $\hat{E}_2$, and $\hat{L}$ that contributes the same monomial as a summand to the RHS. Since there are no negative terms on the RHS it follows that $LHS \leq RHS$, as claimed.

$RHS \leq LHS$: Consider a nonzero summand on the RHS and let $\hat{E}_1, \hat{E}_2 \subseteq E(v, \text{tail}(v))$ with $\hat{E}_1 \cap \hat{E}_2$ and $\hat{L} \subseteq \{v\}$ be the corresponding choices. We have $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$ (as required for a nonzero summand). The summand hence is

$$\text{mon}(\hat{E}_1, \hat{E}_2, \hat{L}) = x^{|\hat{E}_1|} y^{|\hat{E}_2|} z^{|\hat{L}|} \omega^{\mathbf{w}(\hat{E}_1 \cup \hat{E}_2)}.$$

Using $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$ we show that there is the same monomial occurring in the expansion of the product of the LHS for a combination of choices that is unique to $(\hat{E}_1, \hat{E}_2, \hat{L})$:

- From $(1 + z\llbracket J_v = \emptyset \rrbracket)$ choose 1 if $\hat{L} = \emptyset$ and choose z if $\hat{L} = \{v\}$. In the latter case, we indeed have $\llbracket J_v = \emptyset \rrbracket = 1$: From $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$ we get $\emptyset = \hat{L}^+ \cap J' \supseteq \{v, v+n\} \cap J_v$, which is only possible when $J_v = \emptyset$ (as $J_v \subseteq \{v, v+n\}$). In both cases, this matches the contribution of $\hat{L}$ to $\text{mon}(\hat{E}_1, \hat{E}_2, \hat{L}) = x^{|\hat{E}_1|} y^{|\hat{E}_2|} z^{|\hat{L}|} \omega^{\mathbf{w}(\hat{E}_1 \cup \hat{E}_2)}$, namely $z^{|\hat{L}|}$.
- Now consider the factor $1 + x\omega^{\mathbf{w}(e)}\llbracket \{u, v\} \cap J' = \emptyset \rrbracket + y\omega^{\mathbf{w}(e)}\llbracket \{u+n, v+n\} \cap J' = \emptyset \rrbracket$ for some edge $e = \{u, v\} \in E(v, \text{tail}(v))$. We choose depending on which, if any, of $\hat{E}_1$ and $\hat{E}_2$ contains e ; recall that $\hat{E}_1 \cap \hat{E}_2 = \emptyset$:
 - If $e \notin \hat{E}_1$ and $e \notin \hat{E}_2$ then choose 1. This matches the (trivial) contribution of e to $\text{mon}(\hat{E}_1, \hat{E}_2, \hat{L}) = x^{|\hat{E}_1|} y^{|\hat{E}_2|} z^{|\hat{L}|} \omega^{\mathbf{w}(\hat{E}_1 \cup \hat{E}_2)}$.
 - If $e \in \hat{E}_1$ and $e \notin \hat{E}_2$ then choose $x\omega^{\mathbf{w}(e)}$. Let us check that $\llbracket \{u, v\} \cap J' = \emptyset \rrbracket = 1$: From $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$ we get $\forall \{s, t\} \in \hat{E}_1 : \{s, t\} \cap J' = \emptyset$, so indeed $\{u, v\} \cap J' = \emptyset$. Again, $x\omega^{\mathbf{w}(e)}$ matches the contribution of e to $\text{mon}(\hat{E}_1, \hat{E}_2, \hat{L})$.
 - If $e \notin \hat{E}_1$ and $e \in \hat{E}_2$ then choose $y\omega^{\mathbf{w}(e)}$. Let us check that $\llbracket \{u+n, v+n\} \cap J' = \emptyset \rrbracket = 1$: From $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$ we get $\forall \{s, t\} \in \hat{E}_2 : \{s+n, t+n\} \cap J' = \emptyset$, so indeed $\{u+n, v+n\} \cap J' = \emptyset$. Also here, $y\omega^{\mathbf{w}(e)}$ matches the contribution of e to $\text{mon}(\hat{E}_1, \hat{E}_2, \hat{L})$.
 - It is not possible that $e \in \hat{E}_1$ and $e \in \hat{E}_2$.

Clearly, if we consider two tuples $(\hat{E}_1, \hat{E}_2, \hat{L}) \neq (\hat{E}'_1, \hat{E}'_2, \hat{L}')$ corresponding to summands on the RHS then above we make a different choice from $(1 + z\llbracket J_v = \emptyset \rrbracket)$, if $\hat{L} \neq \hat{L}'$, or from $1 + x\omega^{\mathbf{w}(e)}\llbracket \{u, v\} \cap J' = \emptyset \rrbracket + y\omega^{\mathbf{w}(e)}\llbracket \{u+n, v+n\} \cap J' = \emptyset \rrbracket$, for all e with $e \in \hat{E}_1 \triangle \hat{E}'_1$ and/or $e \in \hat{E}_2 \triangle \hat{E}'_2$. (This does not exclude getting the same monomial, but ensures that each summand corresponds to a unique term in the expansion.)

Overall, for each nonzero summand on the RHS we find the same monomial as a unique (and private) term in the expansion of the product on the LHS. Since there are no negative terms, we get that $RHS \leq LHS$.

Thus, $LHS = RHS$, as claimed. $\triangleleft$

Using the claim, the right-hand side of the equation in the lemma statement can be written more concisely:

$$\sum_{\substack{J' = J \cup J_v \\ J_v \subseteq \{v, v+n\}}} (-1)^{|J_v|} \cdot Q_{[v]}(J') \sum_{\substack{\hat{E}_1 \subseteq E(v, \text{tail}(v)) \\ \hat{E}_2 \subseteq E(v, \text{tail}(v)) \\ \hat{L} \subseteq \{v\} \\ \hat{E}_1 \cap \hat{E}_2 = \emptyset}} \text{mon}(\hat{E}_1, \hat{E}_2, \hat{L}) \cdot \llbracket (\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J') \rrbracket$$

We plug in the definition of $Q_{[v]}(J')$:

$$\begin{aligned}
&= \sum_{\substack{J' = J \cup J_v \\ J_v \subseteq \{v, v+n\}}} (-1)^{|J_v|} \sum_{\substack{(E'_1, E'_2, L') \in \\ \text{tpl}(\text{tree}(v))}} \text{mon}(E'_1, E'_2, L') \\
&\cdot \sum_{\substack{I' = J' \cup K' \\ K' \subseteq \text{tree}^+(v)}} (-1)^{|K'|} \cdot \llbracket (E'_1, E'_2, L') \in \text{dsj}(I') \rrbracket \\
&\cdot \sum_{\substack{\hat{E}_1 \subseteq E(v, \text{tail}(v)) \\ \hat{E}_2 \subseteq E(v, \text{tail}(v)) \\ \hat{L} \subseteq \{v\} \\ \hat{E}_1 \cap \hat{E}_2 = \emptyset}} \text{mon}(\hat{E}_1, \hat{E}_2, \hat{L}) \cdot \llbracket (\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J') \rrbracket
\end{aligned}$$

We reorder summation:

$$\begin{aligned}
&= \sum_{\substack{(E'_1, E'_2, L') \in \hat{E}_1 \subseteq E(v, \text{tail}(v)) \\ \text{tpl}(\text{tree}(v)) \hat{E}_2 \subseteq E(v, \text{tail}(v)) \\ \hat{L} \subseteq \{v\} \\ \hat{E}_1 \cap \hat{E}_2 = \emptyset}} \sum_{\substack{J' = J \cup J_v \\ J_v \subseteq \{v, v+n\}}} \sum_{\substack{I' = J' \cup K' \\ K' \subseteq \text{tree}^+(v)}} \text{mon}(E'_1, E'_2, L') \cdot \text{mon}(\hat{E}_1, \hat{E}_2, \hat{L}) \\
&\cdot \sum_{\substack{J' = J \cup J_v \\ J_v \subseteq \{v, v+n\}}} \sum_{\substack{I' = J' \cup K' \\ K' \subseteq \text{tree}^+(v)}} (-1)^{|J_v|} \cdot (-1)^{|K'|} \cdot \llbracket (E'_1, E'_2, L') \in \text{dsj}(I') \rrbracket \\
&\cdot \llbracket (\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J') \rrbracket
\end{aligned}$$

We merge the inner two summations and consolidate the products in the summands: First, replace $J_v \subseteq \{v, v+n\}$ and $K' \subseteq \text{tree}^+(v)$ by $K = J_v \cup K' \subseteq \text{tree}^+[v]$, using that $\text{tree}^+[v] = \text{tree}^+(v) \cup \{v, v+n\}$, and replace $(-1)^{|J_v|} \cdot (-1)^{|K'|}$ by $(-1)^{|K|}$. We use $I := J \cup K$ to reflect the change, but note that $I' = J' \cup K' = J \cup J_v \cup K' = J \cup K = I$. Finally, replace $\llbracket (E'_1, E'_2, L') \in \text{dsj}(I') \rrbracket \cdot \llbracket (\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J') \rrbracket$ by $\llbracket (E'_1 \cup \hat{E}_1, E'_2 \cup \hat{E}_2, L' \cup \hat{L}) \in \text{dsj}(I) \rrbracket$, which we justify now:

- Since $I' = I$ and $J' \subseteq I$, we have that $(E'_1 \cup \hat{E}_1, E'_2 \cup \hat{E}_2, L' \cup \hat{L}) \in \text{dsj}(I)$ implies $(E'_1, E'_2, L') \in \text{dsj}(I')$ and $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$.
- Conversely, if $(E'_1, E'_2, L') \in \text{dsj}(I')$ and $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$ then (with a bit more work) we also get that $(E'_1 \cup \hat{E}_1, E'_2 \cup \hat{E}_2, L' \cup \hat{L}) \in \text{dsj}(I)$:
 - $\forall \{s, t\} \in E'_1 \cup \hat{E}_1 : \{s, t\} \cap I = \emptyset$: We have this for all edges in E'_1 from $(E'_1, E'_2, L') \in \text{dsj}(I')$ since $I' = I$. Let $\{s, t\} \in \hat{E}_1 \subseteq E(v, \text{tail}(v))$; then $s, t \in \text{tail}[v]$ and, hence, $\{s, t\} \cap \text{tree}^+(v) = \emptyset$. Hence, $\{s, t\} \cap I = \{s, t\} \cap (J' \cup K') \subseteq \{s, t\} \cap (J' \cup \text{tree}^+(v)) = \{s, t\} \cap J' = \emptyset$, where the final equality holds as $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$.
 - $\forall \{s, t\} \in E'_2 \cup \hat{E}_2 : \{s+n, t+n\} \cap I = \emptyset$: Fully analogous to previous item.
 - $(L' \cup \hat{L})^+ \cap I = \emptyset$: Clearly $(L' \cup \hat{L})^+ = L'^+ \cup \hat{L}^+$. We have $L'^+ \cap I = \emptyset$ since $I = I'$ and $(E'_1, E'_2, L') \in \text{dsj}(I')$. Furthermore, since $\hat{L} \subseteq \{v\}$ and $K' \subseteq \text{tree}^+(v)$, we have $\hat{L}^+ \cap I = \hat{L}^+ \cap (J' \cup K') = \hat{L}^+ \cap J' = \emptyset$ as $(\hat{E}_1, \hat{E}_2, \hat{L}) \in \text{dsj}(J')$.

With this replacement we arrive at the following:

$$\begin{aligned}
&= \sum_{\substack{(E'_1, E'_2, L') \in \hat{E}_1 \subseteq E(v, \text{tail}(v)) \\ \text{tpl}(\text{tree}(v)) \hat{E}_2 \subseteq E(v, \text{tail}(v)) \\ \hat{L} \subseteq \{v\} \\ \hat{E}_1 \cap \hat{E}_2 = \emptyset}} \sum_{\substack{I = J \cup K \\ K \subseteq \text{tree}^+[v]}} \text{mon}(E'_1, E'_2, L') \cdot \text{mon}(\hat{E}_1, \hat{E}_2, \hat{L}) \\
&\cdot \sum_{\substack{I = J \cup K \\ K \subseteq \text{tree}^+[v]}} (-1)^{|K|} \cdot \llbracket (E'_1 \cup \hat{E}_1, E'_2 \cup \hat{E}_2, L' \cup \hat{L}) \in \text{dsj}(I) \rrbracket
\end{aligned}$$

Finally, we merge the outer two summations and consolidate the product preceding the inner summation. Intuitively, we want to merge $E_1 = E'_1 \cup \hat{E}_1$, $E_2 = E'_2 \cup \hat{E}_2$, and $L = L' \cup \hat{L}$, and let the (new) outer summation go over $(E_1, E_2, L) \in \mathbf{tpl}(\mathbf{tree}[v])$. We show that this yields the same summands, hence the same result:

- Let $(E'_1, E'_2, L') \in \mathbf{tpl}(\mathbf{tree}(v))$, which entails $E'_1, E'_2 \subseteq \delta[\mathbf{tree}(v)]$, $L' \subseteq \mathbf{tree}(v)$, and $E'_1 \cap E'_2 = \emptyset$. Further, let $\hat{E}_1 \subseteq E(v, \mathbf{tail}(v))$, $\hat{E}_2 \subseteq E(v, \mathbf{tail}(v))$, and $\hat{L} \subseteq \{v\}$ with $\hat{E}_1 \cap \hat{E}_2 = \emptyset$. Using these, let $E_1 = E'_1 \cup \hat{E}_1$, $E_2 = E'_2 \cup \hat{E}_2$, and $L = L' \cup \hat{L}$. We show that (E_1, E_2, L) is in $\mathbf{tpl}(\mathbf{tree}[v])$:
 - $E_1 \subseteq \delta[\mathbf{tree}[v]]$: Since $E'_1 \subseteq \delta[\mathbf{tree}(v)]$ and $\hat{E}_1 \subseteq E(v, \mathbf{tail}(v))$, it follows that $E_1 = E'_1 \cup \hat{E}_1 \subseteq \delta[\mathbf{tree}[v]]$.
 - $E_2 \subseteq \delta[\mathbf{tree}[v]]$: Analogous.
 - $L \subseteq \mathbf{tree}[v]$: Since $L' \subseteq \mathbf{tree}(v)$ and $\hat{L} \subseteq \{v\}$, it follows that $L = L' \cup \hat{L} \subseteq \mathbf{tree}[v]$.
 - $E_1 \cap E_2 = \emptyset$: Since $\mathbf{tree}(v) \cap \mathbf{tail}[v] = \mathbf{tree}(v) \cap (\{v\} \cup \mathbf{tail}(v)) = \emptyset$, we have $\delta[\mathbf{tree}(v)] \cap E(v, \mathbf{tail}(v)) = \emptyset$. Thus, $E'_i \cap \hat{E}_j = \emptyset$ for $i, j \in \{1, 2\}$. We also have that $E'_1 \cap E'_2 = \emptyset$ and $\hat{E}_1 \cap \hat{E}_2 = \emptyset$. Hence, $E_1 \cap E_2 = (E'_1 \cup \hat{E}_1) \cap (E'_2 \cup \hat{E}_2) = \emptyset$.
- Conversely, let $(E_1, E_2, L) \in \mathbf{tpl}(\mathbf{tree}[v])$. We show that there are partitions $E_1 = E'_1 \cup \hat{E}_1$, $E_2 = E'_2 \cup \hat{E}_2$, and $L = L' \cup \hat{L}$ such that $(E'_1, E'_2, L') \in \mathbf{tpl}(\mathbf{tree}(v))$, $\hat{E}_1 \subseteq E(v, \mathbf{tail}(v))$, $\hat{E}_2 \subseteq E(v, \mathbf{tail}(v))$, $\hat{L} \subseteq \{v\}$, and $\hat{E}_1 \cap \hat{E}_2 = \emptyset$. Concretely, let $E'_1 = E_1 \cap \delta[\mathbf{tree}(v)]$ and $\hat{E}_1 = E_1 \setminus E'_1$. Similarly, let $E'_2 = E_2 \cap \delta[\mathbf{tree}(v)]$ and $\hat{E}_2 = E_2 \setminus E'_2$. Finally, let $L' = L \cap \mathbf{tree}(v)$ and $\hat{L} = L \setminus L'$.

We check that $(E'_1, E'_2, L') \in \mathbf{tpl}(\mathbf{tree}(v))$:

- By definition we have $E'_1, E'_2 \subseteq \delta[\mathbf{tree}(v)]$ and $L' \subseteq \mathbf{tree}(v)$.
- Since $E_1 \cap E_2 = \emptyset$ and $E'_1 \subseteq E_1$ and $E'_2 \subseteq E_2$, we have $E'_1 \cap E'_2 = \emptyset$.

We check that $\hat{E}_1 \subseteq E(v, \mathbf{tail}(v))$, $\hat{E}_2 \subseteq E(v, \mathbf{tail}(v))$, $\hat{L} \subseteq \{v\}$, and $\hat{E}_1 \cap \hat{E}_2 = \emptyset$:

- $\hat{E}_1 \subseteq E(v, \mathbf{tail}(v))$: We have $\hat{E}_1 = E_1 \setminus E'_1 = E_1 \setminus \delta[\mathbf{tree}(v)]$ and $E_1 \subseteq \delta[\mathbf{tree}[v]]$. Thus, $\hat{E}_1$ only contains edges with at least one endpoint in $\mathbf{tree}[v]$ but no endpoint in $\mathbf{tree}(v)$. Hence all edges therein have v as an endpoint and (still) no endpoint in $\mathbf{tree}(v)$. By the definition of an elimination tree, the other endpoint of such edges must be an ancestor of v , i.e., must be in $\mathbf{tail}(v)$. Thus, $\hat{E}_1 \subseteq E(v, \mathbf{tail}(v))$.
- $\hat{E}_2 \subseteq E(v, \mathbf{tail}(v))$: Analogous.
- $\hat{L} \subseteq \{v\}$: Since $L \subseteq \mathbf{tree}[v] = \mathbf{tree}(v) \cup \{v\}$ and $L' = L \cap \mathbf{tree}(v)$, we have $\hat{L} = L \setminus L' = L \setminus \mathbf{tree}(v) \subseteq \mathbf{tree}[v] \setminus \mathbf{tree}(v) = \{v\}$.
- Since $E_1 \cap E_2 = \emptyset$ and $\hat{E}_1 \subseteq E_1$ and $\hat{E}_2 \subseteq E_2$, we have $\hat{E}_1 \cap \hat{E}_2 = \emptyset$.
- It is easy to see that the above gives a bijection between $(E_1, E_2, L) \in \mathbf{tpl}(\mathbf{tree}[v])$ and the combinations of choices in the ranges of the outer two sums. Forward we have a projection of each component into two disjoint sets, e.g., E_1 into $E'_1 = E_1 \cap \delta[\mathbf{tree}(v)] \subseteq \delta[\mathbf{tree}(v)]$ and $\hat{E}_1 = E_1 \setminus E'_1 \subseteq E(v, \mathbf{tail}(v))$. Backward we take unions of sets that are subsets of disjoint sets, so different pairs of sets cannot have the same union.
- It remains to check that along this bijection we get the same summands. Clearly the inner sums yield the same because they only depend on, e.g., $E_1 = E'_1 \cup \hat{E}_1$ not on the separate sets. It remains to consider $\mathbf{mon}(E'_1, E'_2, L') \cdot \mathbf{mon}(\hat{E}_1, \hat{E}_2, \hat{L})$, using the disjointness observed above, e.g., $E'_i \cap \hat{E}_j = \emptyset$ for $i, j \in \{1, 2\}$:

$$\begin{aligned}
 \mathbf{mon}(E'_1, E'_2, L') \cdot \mathbf{mon}(\hat{E}_1, \hat{E}_2, \hat{L}) &= x^{|E'_1|} y^{|E'_2|} z^{|L'|} \omega^{\mathbf{w}(E'_1 \cup E'_2)} \cdot x^{|E_1|} y^{|E_2|} z^{|L|} \omega^{\mathbf{w}(\hat{E}_1 \cup \hat{E}_2)} \\
 &= x^{|E'_1 \cup \hat{E}_1|} y^{|E'_2 \cup \hat{E}_2|} z^{|L' \cup \hat{L}|} \omega^{\mathbf{w}((E'_1 \cup \hat{E}_1) \cup (E'_2 \cup \hat{E}_2))} \\
 &= \mathbf{mon}(E'_1 \cup \hat{E}_1, E'_2 \cup \hat{E}_2, L' \cup \hat{L}) \\
 &= \mathbf{mon}(E_1, E_2, L)
 \end{aligned}$$

103:22 A Faster Polynomial-Space Algorithm for Hamiltonian Cycle

This completes the argument, and we arrive at the following after making the replacements:

$$\begin{aligned}
 &= \sum_{\substack{(E_1, E_2, L) \in \\ \text{tpl}(\text{tree}[v])}} \text{mon}(E_1, E_2, L) \sum_{\substack{I=J \cup K \\ K \subseteq \text{tree}^+[v]}} (-1)^{|K|} \llbracket (E_1, E_2, L) \in \text{dsj}(I) \rrbracket \\
 &= P_{(v)}(J)
 \end{aligned}$$

This completes the proof. ◀