Algorithmica (2013) 65:868–884
DOI 10.1007/s00453-012-9630-x

Fast Polynomial-Space Algorithms Using Inclusion-Exclusion

Improving on Steiner Tree and Related Problems

Jesper Nederlof

Received: 26 October 2010 / Accepted: 21 February 2012 / Published online: 10 March 2012

Abstract Given a graph with n vertices, k terminals and positive integer weights not larger than c, we compute a minimum STEINER TREE in $\mathcal{O}^\star(2^k c)$ time and $\mathcal{O}^\star(c)$ space, where the $\mathcal{O}^\star$ notation omits terms bounded by a polynomial in the input-size. We obtain the result by defining a generalization of walks, called branching walks, and combining it with the Inclusion-Exclusion technique. Using this combination we also give $\mathcal{O}^\star(2^n)$-time polynomial space algorithms for DEGREE CONSTRAINED SPANNING TREE, MAXIMUM INTERNAL SPANNING TREE and #SPANNING FOREST with a given number of components. Furthermore, using related techniques, we also present new polynomial space algorithms for computing the COVER POLYNOMIAL of a graph, CONVEX TREE COLORING and counting the number of perfect matchings of a graph.

Keywords Inclusion-Exclusion · Fixed parameter tractability · Exact algorithms · Polynomial space · Steiner tree

1 Introduction

Since $\mathcal{P}$ is believed to be not equal to $\mathcal{NP}$, it is accepted that algorithms solving $\mathcal{NP}$-hard problems use super-polynomial, or even exponential, time. This justifies the study of *moderately exponential time algorithms*: algorithms that solve $\mathcal{NP}$-hard problems exactly, but run in exponential time in the worst case. Naturally the challenge still is to get the worst-case complexity as low as possible, and this question

A preliminary version of parts this work appeared in the Proceedings of ICALP 2009.

This work is supported by the Research Council of Norway.

J. Nederlof (✉)
Department of Informatics, University of Bergen, PBOX 7803, 5020 Bergen, Norway
e-mail: jesper.nederlof@ii.uib.no

Table 1 The results of this work compared to the relevant previous results. The number of vertices on the input graph is denoted by n. The running times of the algorithms of this work are given in the *last column*. These new algorithms use polynomial space, except the ones indicated with the *, which use $\mathcal{O}^\star(c)$ space

Problem	Poly-space	By	Exp-space	By	Results
STEINER TREE with unit weights	$5.96^k n^{\mathcal{O}(\log k)}$	[11]	$\mathcal{O}^\star(2^k)$	[4]	$\mathcal{O}^\star(2^k)$
	$\mathcal{O}^\star(1.60^n)$	[11]	$\mathcal{O}^\star(1.36^n)$	[11]	$\mathcal{O}^\star(1.36^n)$
STEINER TREE	$5.96^k n^{\mathcal{O}(\log k)}$	[11]	$\mathcal{O}^\star((2+\epsilon)^k)$	[12]	$\mathcal{O}^\star(2^k c)^*$
			$\mathcal{O}^\star(2^k c)$	[4]	
DEGREE CONSTRAINED SPANNING TREE			$\mathcal{O}^\star(5.92^n)$	[1]	$\mathcal{O}^\star(2^n)$
MAXIMUM INTERNAL SPANNING TREE			$\mathcal{O}^\star(3^n)$	[10]	$\mathcal{O}^\star(2^n)$
#SPANNING FORESTS			$\mathcal{O}^\star(2^n)$	[5]	$\mathcal{O}^\star(2^n)$
COVER POLYNOMIAL	$\mathcal{O}^\star(3^n)$	[5]	$\mathcal{O}^\star(2^n)$	[5]	$\mathcal{O}^\star(2^n)$
CONVEX TREE COLORING			$\mathcal{O}^\star(2^k c)$	[19]	$\mathcal{O}^\star(2^k c)^*$
#PERFECT MATCHING	$\mathcal{O}^\star(2^n)$	[2]	$\mathcal{O}^\star(1.62^n)$	[16]	$\mathcal{O}^\star(1.95^n)$

already has given rise to many non-trivial algorithms. See for example the survey by Woeginger [20].

Many of these algorithms also use exponential space. In fact, the time/space-usage ratio is polynomial in the input (i.e. the running time is bounded by a polynomial function of the space usage). It is expected that these algorithms are not practical even for small problem-instances, since they tend to run out of memory, require lot of (slow) disk access, and hardly allow parallel execution. Therefore polynomial-space exact algorithms have already been studied for several $\mathcal{NP}$-hard problems [3, 7, 11, 14, 15, 20]. In this paper we focus on space usage and give improved algorithms for several problems.

In 2009, Björklund et al. [3] drew new attention to the well-known principle of Inclusion-Exclusion: They gave $\mathcal{O}^*(2^n)$-time algorithms for several set partition problems, the most prominent one being k-COLORING. They also mention a simple adjustment to their algorithm to achieve an $\mathcal{O}^*(2.24^n)$-time algorithm with polynomial space for k-COLORING. Also related to this are the $\mathcal{O}^*(2^n)$-time polynomial-space algorithms for #HAMILTONIAN PATH by Karp [14] and (implicitly) Kohn et al. [15], and for #PERFECT MATCHING by Björklund and Husfeldt [2].

Our algorithms heavily rely on the work of Björklund et al. [2–6]. The results can be read from Table 1.

STEINER TREE is one of the most well-studied $\mathcal{NP}$-complete problems. For this problem, the Dreyfus-Wagner [9] dynamic programming algorithm has been the fastest exact algorithm for over 30 years. However, recently Björklund et al. [4] gave an $\mathcal{O}^\star(2^k)$-time algorithm for the variant with bounded integer weights with k terminals, and Fuchs et al. [12] gave an $\mathcal{O}^\star(c^k)$-time algorithm for the general case, for any $c > 2$. Both algorithms use $\Omega(2^k)$ space. In 2008, Fomin et al. [11] initiated the study of polynomial space algorithms for STEINER TREE. They gave polynomial space algorithms with running times bounded by $5.96^k n^{\mathcal{O}(\log k)}$ and $\mathcal{O}^\star(1.60^n)$ where n is the number of vertices in the graph. They pose the question whether STEINER TREE is fixed parameter tractable with respect to k when there is a polynomial space restriction. We answer this question affirmatively by providing an algorithm that runs

in $\mathcal{O}^*(2^k)$ time and meets the restriction. Using the techniques of [11], this also leads to a polynomial-space $\mathcal{O}^*(1.36^n)$-time algorithm.

The MAX INTERNAL SPANNING TREE (MIST) and DEGREE CONSTRAINED SPANNING TREE (DCST; also called MIN-MAX DEGREE SPANNING TREE) problems are natural generalizations of HAMILTONIAN PATH. In [10], Fernau et al. ask if there exists an $\mathcal{O}^\star(2^n)$-time algorithm to solve MIST. In [13], Gaspers et al. ask if there exists an $\mathcal{O}^\star(2^n)$-time algorithm solving DCST. We answer both questions by giving polynomial-space algorithms with this running time.

The COVER POLYNOMIAL of a directed graph, introduced by Graham and Chung [8], generalizes all problems that can be solved using two operations named deletion and contraction of edges, and is designed to be the directed analogue of the Tutte polynomial. We improve the $\mathcal{O}^\star(3^n)$-time polynomial-space algorithm of Björklund et al. [5] to an $\mathcal{O}^\star(2^n)$-time polynomial-space algorithm. We also give the same improvement for #SPANNING FORESTS, which is one particular case of the Tutte polynomial. For more information about the Tutte polynomial we refer to [5].

Finally, we give two new algorithms for CONVEX TREE COLORING and #PERFECT MATCHING (counting the number of perfect matchings of a graph). (A special case of) the CONVEX TREE COLORING problem was studied in [19], where a $\mathcal{O}^\star(2^k c)$ time and $\mathcal{O}^\star(c)$ space algorithm was given. We will continue this study by emphasizing the space usage aspect. Finally, we show that # PERFECT MATCHING can be solved in $\mathcal{O}^\star(1.95^n)$ time and polynomial space, improving over the previous $\mathcal{O}^\star(2^n)$ time polynomial space algorithm of Björklund and Husfeldt [2]. It is worth to mention that with exponential space, one can count perfect matchings of general graphs even in $\mathcal{O}^\star(1.62^n)$ time due to Koivisto [16].

The paper is organized as follows: After the preliminaries in Sect. 2, we revisit the principle of Inclusion-Exclusion and the well-known Hamiltonian path algorithm in Sect. 3. After this we provide in Sect. 4 a natural extension by introducing the concept of *branching walks* and give the resulting algorithms. In the remaining sections we prove the remaining results that are not primarily based on branching walks.

2 Preliminaries and Notation

We will use the $\mathcal{O}^\star$ notation that suppresses any factor that is polynomial in the input size. For an integer i, we use $[i]$ to denote $\{1, \ldots, i\}$. We also use Iverson's bracket notation: for a proposition p, $[p]$ denotes 1 if p is `true` and 0 otherwise. Let $G = (V, E)$ be a graph. We will use $V(G)$ and $E(G)$ for the vertices V of G and the edges E of G, respectively. The number of vertices of the input graphs is denoted by n. The graph $G[X]$ *induced by* X is (i) $(X, (X \times X) \cap E)$ where $X \subseteq V$, and (ii) $(\bigcup_{e \in X} e, X)$ if $X \subseteq E$. For $v \in V(G)$, $N_G(v)$ are all vertices adjacent to v in G and $d_G(v) = |N_G(v)|$; The subscript G is omitted if the graph is clear from the context. A *walk of length* k in G is a tuple $W = (v_1, \ldots, v_{k+1}) \in V^{k+1}$ such that $(v_i, v_{i+1}) \in E$ for each $0 < i \leq k$. We say W is *from* v if $v_1 = v$, and W is *cyclic* if $v_1 = v_{k+1}$. Furthermore, $v \in V$ is said to be *visited* by W if $v_i = v$ for some $1 \leq i \leq k$. For a rooted tree T we will use $\mathtt{rt}(T)$ to denote the root of T. For a vertex $v \in V(T)$ we will also use $\mathtt{pa}(v)$ to denote the parent of v.

Also given a graph $G_1 = (V_1, E_1)$, a *homomorphism from* G_1 *to* G is a function $\varphi : V_1 \to V$ such that $(u, v) \in E_1$ implies $(\varphi(u), \varphi(v)) \in E$. Note that φ directly corresponds to a walk in the case that G_1 is a path. We will use the notation $\varphi(X) = \{\varphi(u) \mid u \in X\}$ and $\varphi(Y) = \{(\varphi(u), \varphi(v)) \mid (u, v) \in Y\}$ for $X \subseteq V_1$ and $Y \subseteq E_1$. If $G_2 = (V_2, E_2)$ is a third graph with $V_1 \cap V_2 = \emptyset$ and $\varphi_1 : V_1 \to V$ and $\varphi_2 : V_2 \to V$ are homomorphisms, we will use $\varphi_1 \cup \varphi_2$ to denote the homomorphism $V_1 \cup V_2 \to V$ defined by $\varphi(v) = \varphi_1(v)$ and $\varphi(w) = \varphi_2(w)$ with $v \in V_1$ and $w \in V_2$.

2.1 Model

We assume integers in the input are given in binary, and we will prove our results for the RAM computation model where any arithmetic operation and storing any integer is assumed to take constant time and space. However, all our results also hold in the (more realistic) RAM computation model where arithmetic operations and storing integers only take constant time and space if they are constant-sized. To see the significance, note for example that in the first model two n-bits integers can be added in constant time, while in the second it takes linear time.

3 Inclusion-Exclusion

Theorem 1 (Folklore) *Let U and R be sets and for every $v \in R$, let P_v be a subset of U. Use $\overline{P_v}$ to denote $U \setminus P_v$. With the convention $\bigcap_{i \in \emptyset} \overline{P_i} = U$, the following holds*:

$$\left| \bigcap_{v \in R} P_v \right| = \sum_{F \subseteq R} (-1)^{|F|} \left| \bigcap_{v \in F} \overline{P_v} \right|. \tag{1}$$

In this work, we call any application of the above theorem an *Inclusion-Exclusion-formulation* (IE-formulation). In the context of this paper it is convenient to use the following terminology: We refer to the set U as the *universe*, to R as the *requirement space* and to P_v as a *property*. Moreover, given a set $F \subseteq R$, we call the task of computing $|\bigcap_{v \in F} \overline{P_v}|$, the *simplified problem*. Note that if the simplified problem can be solved in $\mathcal{O}^\star(t(n))$ time and $\mathcal{O}^\star(s(n))$ space, the left-hand side of (1) can be determined in $\mathcal{O}^\star(2^n t(n))$-time and $\mathcal{O}^\star(s(n))$ space in the straightforward manner. All algorithms in this paper will exploit this observation.

3.1 Hamiltonian Paths

To illustrate Theorem 1, we will first recall the following IE-formulation due to Karp [14] for counting Hamiltonian paths. Given a (possibly directed) graph $G = (V, E)$, a Hamiltonian path is a walk that contains each vertex of G exactly once.[1]

Theorem 2 ([14]) *Hamiltonian paths of a graph of n vertices can be counted in $\mathcal{O}^\star(2^n)$ time and polynomial space.*

[1] Note that this is slightly different from the usual definition, since a path corresponds to two walks in both directions. So we will actually obtain a number that is exactly twice the number of Hamiltonian paths.

Proof Let $G = (V, E)$. In the context of Theorem 1, define the universe U as all walks of length $n-1$ in G, the requirement space $R = V$, and P_v as all walks of length $n-1$ that visit vertex v, for every $v \in V$. With these definitions, the left-hand side of (1), $|\bigcap_{v\in V} P_v|$, is the number of Hamiltonian paths in G. Now it remains to show how to solve the simplified problem. Given $F \subseteq V$ and $x \in V \setminus F$, let $w_F(x,k)$ be the number of walks from x of length k in $G[V \setminus F]$. Then $w_F(x,k)$ admits the following recurrence:

$$w_F(x,k) = \begin{cases} 1 & \text{if } k = 0, \\ \sum_{t\in N(x)\setminus F} w_F(t,k-1) & \text{otherwise.} \end{cases} \tag{2}$$

Also, notice that

$$\left|\bigcap_{v\in F} \overline{P_v}\right| = \sum_{s\in V\setminus F} w_F(s, n-1),$$

and hence the simplified problem can be solved in polynomial time using dynamic programming on (2) (the parameter F is fixed but is added for clarity). Thus it takes $\mathcal{O}^\star(2^n)$ time and polynomial space to evaluate (1), and the theorem follows. □

4 Branching Walks

Definition 1 A *branching walk* B in $G = (V, E)$ is a pair (T, φ) where T is an ordered rooted tree and $\varphi : V(T) \to V$ is a homomorphism from T to G. For a vertex $x \in V$, B *is from* x if $\varphi(\mathtt{rt}(T)) = x$. B *visits* a vertex $v \in V$ if $v \in \varphi(V(T))$. The *length* of B is $|E(T)|$.

A branching walk is a natural generalization of a walk: Notice that a branching walk (T, φ) is a walk in the special case that T is a path rooted at an endpoint. Since we will count distinct branching walks, we emphasize that two branching walks (T_1, φ_1) and (T_2, φ_2) are distinct if there is no isomorphism $\psi : T_1 \to T_2$ such that $\varphi_1(v) = \varphi_2(\psi(v))$ for every $v \in V(T_1)$.

4.1 Steiner Tree

In this section we will give an extension of the technique in the previous section to obtain a new IE-formulation for the STEINER TREE problem, which is defined as follows:

STEINER TREE

Input $G = (V, E)$, $c \in \mathbb{Z}^+$, weight function $w : E \to [c] \setminus 0$ and terminals $K \subseteq V$.
Question Is there a subtree (V', E') of G such that $K \subseteq V'$ and $\sum_{e\in E'} w(e) \leq c$?

Given a branching walk (T, φ), the quantity $\sum_{e\in E(T)} w(\varphi(e))$ is said to be its *weight*.

Lemma 1 *Let $s_0 \in K$. There exists a subtree $S = (V', E')$ of G such that $K \subseteq V'$ and $w(E') \leq c$ if and only if there exists a branching walk $B = (T, \varphi)$ from s_0 of weight at most c such that $K \subseteq \varphi(V(T))$.*

Proof For the forward direction, assume S to be ordered by fixing an arbitrary order. Then define $B = (S, \varphi)$, with $\varphi : V' \to V'$ the identity function, and let $\mathtt{rt}(S) = s_0$ (this is possible since $s_0 \in K \subseteq V(S)$). Then, clearly $w(\varphi(E(S))) \leq c$. For the backward direction, notice that $(\varphi(V(T)), \varphi(E(T)))$ is connected and if we let S be a spanning tree of $(\varphi(V(T)), \varphi(E(T)))$, it has the required properties. □

Consider the following IE-formulation: Let $s_0 \in K$ be an arbitrarily chosen terminal, and define the universe U to be the set of all branching walks (T, φ) from s_0 of weight at most c. Let the requirement space R be K. For every $v \in K$, define a property $P_v \subseteq U$ that consists of all branching walks in U that visit v. It follows that the left-hand side of (1), $|\bigcap_{v \in K} P_v|$, is the number of branching walks of weight at most c that contain all terminals. By Lemma 1, this quantity is larger than 0 if and only if the instance of Steiner Tree is a `yes`-instance. Hence we can restrict our goal to determining $|\bigcap_{v \in K} P_v|$. For this we use Theorem 1.

Before we proceed, first let us recall a basic combinatorial problem: Let $\mathcal{T}_n$ be the set of all distinct ordered rooted trees on n edges (also called the *Catalan numbers*). We will give a recurrence for $|\mathcal{T}_n|$. Let

$$\gamma : \bigcup_{i,j} (\mathcal{T}_i \times \mathcal{T}_j) \leftrightarrow \mathcal{T}_{i+j+1} \tag{3}$$

be the gluing operation such that $T = \gamma(T_1, T_2)$ is obtained by connecting $\mathtt{rt}(T_1)$ and $\mathtt{rt}(T_2)$, setting $\mathtt{rt}(T) = \mathtt{rt}(T_2)$, letting the first child of $\mathtt{rt}(T)$ be $\mathtt{rt}(T_1)$, being followed by the children of $\mathtt{rt}(T)$ in T_2. Clearly γ is a bijection so $|\mathcal{T}_n| = \sum_{i=0}^{n-i-1} |\mathcal{T}_i||\mathcal{T}_{n-i-1}|$.

We now continue applying Theorem 1 by showing how the simplified problem can be solved in $\mathcal{O}^\star(c)$ time and space. For $F \subseteq K$, define $\mathcal{B}_F(x, W)$ to be all branching walks (T, φ) of weight at most W from x in $G[V \setminus F]$, where $x \in V \setminus F$. Hence $U = \mathcal{B}_\emptyset(x, c)$. Let $b_F(x, W) = |\mathcal{B}_F(x, W)|$. First, note that the simplified problem is to compute

$$\left| \bigcap_{v \in F} \overline{P_v} \right| = b_F(s_0, W)$$

for a given set $F \subseteq K$ of terminals. Now $b_F(s_0, W)$ can be computed in polynomial time using the following lemma in combination with dynamic programming:

Lemma 2 *Let $F \subseteq K$ and $x \in V \setminus F$, then*

$$b_F(x, W) = \begin{cases} 1 \quad \textit{if } W = 0, & \text{(4a)} \\ \sum_{t \in N(x) \setminus F} \sum_{W_1 + W_2 = W - w(x,t)} b_F(t, W_1) b_F(x, W_2) & \\ \quad \textit{otherwise}. & \text{(4b)} \end{cases}$$

Proof There is one branching walk B of weight 0, $B = (T, \varphi)$, from x with T being a single vertex and φ mapping this single vertex to x, hence Case 4a. For the second case, define a function

$$\gamma' : \bigcup_{t \in N(x) \setminus F} \bigcup_{W_1, W_2} \big(\mathcal{B}_F(t, W_1) \times \mathcal{B}_F(x, W_2)\big) \leftrightarrow \mathcal{B}_F\big(x, W_1 + W_2 + w(x, t)\big)$$

by $\gamma'((T_1, \varphi_1), (T_2, \varphi_2)) = (\gamma(T_1, T_2), \varphi_1 \cup \varphi_2)$. Now γ' is a bijection since γ is a bijection as stated in (3) and for the branching walk $(\gamma(T_1, T_2), \varphi)$ it must hold that $\varphi(\mathtt{rt}(T_1)) \in N(\varphi(\mathtt{rt}(T_2)))$. Hence Case 4b follows. □

Theorem 3 *The* STEINER TREE *problem can be solved in* $\mathcal{O}^\star(2^k c)$ *time and* $\mathcal{O}^\star(c)$ *space.*

Proof Due to Lemma 1 the considered IE-formulation solves STEINER TREE, and we can use dynamic programming on (4b) to compute the simplified problem in $\mathcal{O}^\star(c^2)$ time. This can be further reduced to $\mathcal{O}^\star(c)$ time in a standard manner with the Fast Fourier Transform. Then the theorem follows from Theorem 1. □

The following result is a consequence of Theorem 3 and the considerations of Sect. 4.2 in [11]:

Theorem 4 *The* STEINER TREE *problem with unit weights can be solved in* $\mathcal{O}^\star(1.36^n)$ *time using polynomial space.*

Proof Modify Algorithm `steiner` as described in Sect. 4.2 in [11], except that we replace Step 4 of `steiner` with the algorithm due to Theorem 3. This clearly does not change the worst-case running time as claimed in Theorem 5 of [11], and it is easy to see that the new algorithm uses polynomial space. □

4.2 Degree Constrained Spanning Tree

DEGREE CONSTRAINED SPANNING TREE (DCST)

Input $G = (V, E)$, $1 \leq c \leq n$.
Question Is there a spanning tree of G with maximum degree at most c?

Analogous to Lemma 1, we will use the following lemma to reduce our problem to computing some quantity involving branching walks:

Lemma 3 *There exists a spanning tree of* G *of maximum degree at most* c *if and only if there exists a branching walk* $B = (T, \varphi)$ *of length* $n - 1$ *such that* $\varphi(V(T)) = V$ *and the maximum degree of* T *is at most* c.

Proof For the forward direction, assume S to be a spanning tree of G of maximum degree at most c and order it by fixing an arbitrary order. Then define $B = (S, \varphi)$, with $\varphi : V(S) \to V(S)$ the identity function, and choose $\mathtt{rt}(S)$ arbitrarily. For the backward direction, notice that $(\varphi(V(T)), \varphi(E(T)))$ is a tree of maximum degree at most c. □

Now we can use the following IE-formulation: Define the universe U as all branching walks (T, φ) of length $n - 1$ such that the maximum degree of T is at most c. Define the requirement space $R = V$ and P_v to be all branching walks in U visiting v, for every $v \in V$. That is, let $P_v = \{(T, \varphi) \in U : v \in \varphi(V(T))\}$. For $F \subseteq V$,

define $\mathcal{D}_F(x, j, g)$ as all branching walks $(T, \varphi) \in U$ from x of length j in $G[V \setminus F]$ such that the degree of the root of T is at most g and the degree of every other vertex is at most c. Hence $U = \bigcup_{x \in V \setminus F} \mathcal{D}_\emptyset(x, n-1, c)$. Let $d_F(x, j, g) = |\mathcal{D}_F(x, j, g)|$. We apply Theorem 1, and conclude that the simplified problem is to compute the number of branching walks in the universe that avoid F, and hence

$$\left| \bigcap_{v \in F} \overline{P_v} \right| = \sum_{x \in V \setminus F} d_F(x, n-1, c).$$

This can be done in polynomial time with dynamic programming using the following lemma:

Lemma 4

$$d_F(x, j, g) = \begin{cases} [g \geq 0] \quad \textit{if } j = 0, & \text{(5a)} \\ \sum_{t \in N(x) \setminus F} \sum_{j_1 + j_2 = j-1} d_F(t, j_1, c-1)\, d_F(x, j_2, g-1) & \\ \quad \textit{otherwise}. & \text{(5b)} \end{cases}$$

Proof To see that Case 5a holds, notice that $d_F(x, 0, g) = 0$ if g is negative since the root has a non-negative degree, and otherwise there is one branching walk (T, φ) of length zero obtained by letting T be the tree on one vertex and letting φ be the function mapping this vertex to x.

For Case 5b, define $\gamma'((T_1, \varphi_1), (T_2, \varphi_2))$ to be the branching walk $(\gamma(T_1, T_2), \varphi_1 \cup \varphi_2)$, where γ is the gluing operation from (3). Then we claim that γ' is a bijection

$$\gamma' : \bigcup_{t \in N(x) \setminus F} \bigcup_{j_1, j_2} \big(\mathcal{D}_F(t, j_1, c-1) \times \mathcal{D}_F(x, j_2, g) \big) \leftrightarrow \mathcal{D}_F(x, j_1 + j_2 + 1, g+1).$$

To see this, define a (c, g)-tree as a tree of maximum degree c with the degree of its root at most g, and notice that $\gamma(T_1, T_2)$ is a (c, g)-tree if and only if T_1 is a $(c, c-1)$-tree and T_2 is a $(c, g-1)$-tree. Then Case 5b follows by combining with the arguments in the proof of Lemma 2. □

Theorem 5 *The* Degree Constrained Spanning Tree *problem can be solved in* $\mathcal{O}^\star(2^n)$ *time and polynomial space.*

Proof Due to Lemma 3 the considered IE-formulation solves DCST, and we can use dynamic programming on (4) to compute the simplified problem in polynomial time. Then the theorem follows from Theorem 1. □

4.3 Maximum Internal Spanning Tree

A vertex of a tree is called *internal* if its degree is at least 2.

Maximum internal Spanning Tree

Input $G = (V, E)$, $1 \leq c \leq n$.
Question Is there a spanning tree of G with at least c internal vertices?

Lemma 5 *There exists a spanning tree of G with at least c internal vertices if and only if there exists a branching walk $B = (T, \varphi)$ of length $n-1$ such that $\varphi(V(T)) = V$ and T has at least c internal vertices.*

Proof For the forward direction, assume S to be a spanning tree of G with at least c internal vertices and order it by fixing an arbitrary order. Define $B = (S, \varphi)$, with $\varphi : V(S) \to V(S)$ the identity function. It clearly has the required properties. For the backward direction, notice that $(V, \varphi(E(T)))$ is a spanning tree with at least c internal vertices since $\phi(E(T)) = n-1$. □

Consider the following IE-formulation: Define the universe U to be all branching walks (T, φ) of length $n-1$ such that T has at least c internal vertices. For $v \in V$, let $P_v = \{(T, \varphi) \in U : v \in \varphi(V(T))\}$. For $F \subseteq V$ and $\delta \in \mathbb{Z}_- \cup \{0, 1, 2\}$, define $\mathcal{M}_F^\delta(x, j, g)$ as all branching walks (T, φ) in $G[V \setminus F]$ of length j from x such that $|\{v \in V(T) \setminus \mathtt{rt}(T) : v \text{ is internal}\}| + [d(\mathtt{rt}(T)) \geq \delta] = g$. Let $m_F^\delta(x, j, g) = |\mathcal{M}_F^\delta(x, j, g)|$, then

Lemma 6

$$m_F^\delta(x, j, g) = \begin{cases} \big[g = [\delta \leq 0]\big] & \text{if } j = 0, \quad (6a) \\ \sum_{t \in N(x) \setminus F} \sum_{g_1+g_2=g} \sum_{j_1+j_2=j-1} m_F^1(t, j_1, g_1) & \\ \quad \times m_F^{\delta-1}(x, j_2, g_2) & \text{otherwise}. \quad (6b) \end{cases}$$

Proof To see that Case 6a holds, notice that there is only one branching walk of length 0 from x and its tree-part is a single vertex, so g must be equal to 1 if $\delta \leq 0$ and equal to 0 otherwise by definition. For Case 6b, first define $\gamma'((T_1, \varphi_1), (T_2, \varphi_2))$ as the branching walk $(\gamma(T_1, T_2), \varphi_1 \cup \varphi_2)$, where γ is the gluing operation from (3). Then we claim that γ' is a bijection

$$\gamma' : \bigcup_{t \in N(s) \setminus F} \bigcup_{g_1, g_2} \bigcup_{j_1, j_2} \big(\mathcal{M}_F^1(t, j_1, g_1) \times \mathcal{M}_F^{\delta-1}(s, j_2, g_2)\big) \leftrightarrow \mathcal{M}_F^\delta(s, j_1 + j_2 + 1, g_1 + g_2).$$

To see this, note that $\gamma(T_1, T_2)$ has g' internal vertices not equal to the root if and only if T_1 has g_1' and T_2 has g_2' internal vertices not equal to the root with $g_1' + g_2' = g'$ and

$$d_{\gamma(T_1,T_2)}\big(\mathtt{rt}\big(\gamma(T_1, T_2)\big)\big) = 1 + d_{T_2}\big(\mathtt{rt}(T_2)\big).$$

Hence the δ and g accumulators are modified correctly. Then Case 6b follows by combining with the arguments in the proof of Lemma 2. □

Theorem 6 *The* Maximum Internal Spanning Tree *problem can be solved in $\mathcal{O}^\star(2^n)$ time and polynomial space.*

Proof Use the IE-formulation as described above. The simplified problem is to compute $|\bigcap_{v \in F} \overline{P_v}|$, which is equal to $\sum_{s \in V \setminus F} m_F^2(s, n-1, c)$ since the root needs at least two neighbors in order to be an internal vertex. It follows from Lemma 5 that there exists $B \in \bigcap_{v \in V} P_v$ if and only if there exists a spanning tree with at least c leaves. □

4.4 Counting Spanning Forests

Define a *c-spanning forest* to be an acyclic spanning subgraph with exactly c connected components. In this section we will address the following problem:

#SPANNING FORESTS

Input $G = (V, E)$, $1 \leq c \leq n$.
Question The number of acyclic spanning subgraphs of G with exactly c connected components.

Assume that a total ordering $\prec$ on the vertex set V is given. Given a subset $X \subseteq V$, let $\min X$ be the minimum element of X with respect to $\prec$.

Definition 2 A *sorted branching walk* is a branching walk (T, φ) such that for every $v \in V(T)$ with children $c_1, \ldots, c_l$, numbered with respect to the order of T, $\varphi(c_i) \prec \varphi(c_j)$ for every $i < j$ and $\varphi(\mathtt{rt}(v)) = \min \varphi(V(T))$.

It can be noted that the definition implies no two children of vertex $v \in V(T)$ can be mapped to same vertex in G.

Lemma 7 *There is a bijection between the set of all subtrees S of G, and the set of all sorted branching walks $B = (T, \varphi)$ of length $|\varphi(E(T))|$ such that $\varphi(E(T))$ induces a subtree.*

Proof Note that it is sufficient to show that for every subtree S of G, there is exactly one a sorted branching walk $B = (T, \varphi)$ of length $|\varphi(E(T))|$ such that $\varphi(E(T)) = E(S)$.

Since $\varphi(V(T)) = V(S)$ and B is sorted, we know that $\varphi(\mathtt{rt}(T)) = \min V(T)$. Since $|E(S)| = |E(T)|$, φ is a bijection and hence S and T are isomorphic. Moreover $\varphi(c_i) \prec \varphi(c_j)$ whenever c_i occurs before c_j as a child of a vertex in T since B is sorted. This implies that T and φ are uniquely determined by S. □

Define $\mathcal{B}_F(l)$ to be all sorted branching walks (T, φ) of length l in G such that $F \cap \varphi(V(T)) = \emptyset$. Also, define $b_F(l) = |\mathcal{B}_F(l)|$.

Lemma 8 *The number of c-spanning forests is equal to*

$$\sum_{F \subseteq V} (-1)^{|F|} \frac{1}{c!} \sum_{l_1 + \cdots + l_c = n - c} \prod_{j=1}^{c} b_F(l_j). \tag{7}$$

Proof We apply Theorem 1. Define U to be the set of all families f of sorted branching walks of total length $n - c$ (that is, the sum of the length of all members of f is $n - c$). Let the requirement space be V, and for every $v \in V$ define P_v to be all elements $f \in U$ such that there is $(T, \varphi) \in f$ with $v \in \varphi(V(T))$. It is not hard to see that a term of the summation of (7) (ignoring $(-1)^{|F|}$) is equal to $|\bigcap_{v \in F} \overline{P_v}|$.

Applying Theorem 1, it remains to show that the number of c-spanning forests is $|\bigcap_{v \in V} P_v|$. To see this, notice that for a family of sorted branching walks $f =$

$\{(T_1, \varphi_1), \ldots, (T_c, \varphi_c)\}$, $\bigcup_{i=1}^{c} \varphi_i(V_i) = V$ if and only if $\bigcup_{i=1}^{c} \varphi_i(E_i)$ is a c-spanning forest. Then every c-spanning forest corresponds to exactly one set of branching walks of total length $n - c$ due to Lemma 7. The lemma follows. □

Define $\mathcal{B}_F(x, j, g)$ as all sorted branching walks (T, φ) from x of length j such that $\varphi(V(T)) \cap F = \emptyset$ and no child of the root of T is mapped to one of the first $g-1$ neighbors of x in $G[V \setminus F]$ with respect to the ordering $\prec$. Define $b_F(x, j, g) = |\mathcal{B}_F(x, j, g)|$. Use $N_F^q(x)$ to denote the qth element of the set $N(x) \setminus F$ with respect to the ordering $\prec$.

Lemma 9

$$b_F(x, j, g) = \begin{cases} 1 \quad \textit{if } j = 0, & \text{(8a)} \\ 0 \quad \textit{else if } g > |N(x) \cap F|, & \text{(8b)} \\ b_F(x, j, g+1) + \sum_{j_1+j_2=j-1} b_F\big(N_F^g(x), j_1, 1\big) & \\ \quad \times b_F(x, j_2, g+1) \quad \textit{otherwise}. & \text{(8c)} \end{cases}$$

Proof For Case 8a, notice there is exactly one branching walk (T, φ) with T being a single vertex and with φ mapping this vertex to x.

To see Case 8b, notice that since $g > |N(x) \cap F|$, in any branching walk in $\mathcal{B}_F(x, j, g)$ the root can not have any children but since $j > 0$ this must be the case, and hence such a branching walk does not exist.

For Case 8c, first notice that $\mathcal{B}_F(x, j, g+1)$ are exactly all branching walks $(T, \varphi) \in \mathcal{B}_F(x, j_1 + j_2 + 1, g)$ where no child of the root of T is mapped to $N_F^g(x)$. Define $\gamma'((T_1, \varphi_1), (T_2, \varphi_2))$ as the branching walk $(\gamma(T_1, T_2), \varphi_1 \cup \varphi_2)$, where γ is the gluing operation from (3). Then it is not too hard to see that γ' is a bijection

$$\gamma' : \bigcup_{j_1, j_2} \Big(\mathcal{B}_F\big(N_F^g(x), j_1, 1\big) \times \mathcal{B}_F(x, j_2, g+1)\Big) \leftrightarrow \mathcal{B}_F(x, j_1 + j_2 + 1, g) \setminus \mathcal{B}_F(x, j_1 + j_2 + 1, g+1)$$

since $\gamma'(B_1, B_2)$ and $\gamma'(B_1', B_2')$ are distinct whenever either B_1 and B_1' or B_2 and B_2' are distinct. Then Case 8c follows by combining with the arguments in the proof of Lemma 2. □

Theorem 7 *The #*SPANNING FORESTS *problem can be solved in $\mathcal{O}^\star(2^n)$ time and polynomial space.*

Proof Let $\overline{F} = V \setminus F$ and $b_F(l) = \sum_{x \in \overline{F}} b_{\overline{F}[x]}(x, l, 1)$, where $\overline{F}[x]$ stands for the set of all elements e in $\overline{F}$ such that $x \prec e$. Using dynamic programming in combination with Lemma 9, the values $b_F(l)$ can be computed for every $1 \le l \le n$ for a fixed F. Also using standard dynamic programming, the simplified problem (that is, the summand of (8a) ignoring the $(-1)^{|F|}$) can be computed in polynomial time. Hence (8a) can be evaluated within the claimed resource bounds and the theorem follows from Lemma 8. □

5 Cover Polynomial

We use $x^{\underline{i}}$ for the falling factorial $\frac{x!}{(x-i)!}$. The cover polynomial of a directed graph $D = (V, A)$ can be defined as (see also [5, 8]):

$$\sum_{i,j} c(i, j) x^{\underline{i}} y^{\underline{j}}$$

where $c(i, j)$ is defined as the number of ways to partition V into i directed paths and j directed cycles of D. In this section we will address the following problem:

COVER POLYNOMIAL

Input A graph G.
Question The coefficients $c(i, j)$ for every $0 \leq i, j \leq n$.

Since paths and cycles with l edges contain $l + 1$ and l vertices respectively, the sum of the lengths of the paths and cycles in such a partition will be $n - i$. Moreover, if V is covered by i paths and j cycles with lengths summing up to $n - 1$, the path and cycles are disjoint because of the size restriction. Recall from Sect. 3.1 that $w_F(s, j)$ is the number of walks starting from s of length j avoiding F. Similarly, define $w_F(j)$ as all walks of length j avoiding F and $\hat{w}_F(j)$ as the number of cyclic walks of length j avoiding F.

Lemma 10

$$c(i, j) = \sum_{F \subseteq V} (-1)^{|F|} \frac{1}{i!j!} \sum_{l_1 + \cdots + l_{i+j} = n-i} \left(\prod_{k=1}^{i} w_F(l_k) \right) \left(\prod_{k=i+1}^{i+j} \hat{w}_F(l_k) \right). \quad (9)$$

Proof We apply Theorem 1. Define U as all combinations of i walks and j cyclic walks with lengths summing up to exactly $n - i$. Define the requirement space to be V, and for every $v \in V$ let P_v be all combinations of walks and cyclic walks in U such that at least one of the (cyclic) walks visits v. It is easy to see that each summand (ignoring the $(-1)^{|F|}$) equals $|\bigcap_{v \in F} \overline{P_v}|$, and since $|\bigcap_{v \in F} P_v| = c(i, j)$ by the above discussion, the lemma follows from Theorem 1. □

Theorem 8 *The* COVER POLYNOMIAL *problem can be solved in* $\mathcal{O}^\star(2^n)$ *time and polynomial space.*

Proof By using minor modifications of the dynamic programming algorithm using (2), for every $F \subseteq V$ and $0 \leq j \leq n$, both $w_F(j)$ and $\hat{w}_F(j)$ can be computed in polynomial time. Using straightforward dynamic programming, the simplified problem can be obtained from the values $w_F(j)$ and $\hat{w}_F(j)$. Using this, (9) can be evaluated within the claimed resource bounds. □

6 Convex Tree Coloring

In this section we solve a generalization of the CONVEX RECOLORING problem studies in [19] and improve upon one of their results. Let $T = (V, E)$ be a tree and

$w : E(T) \times [k] \times [k] \to [c]$. A *k-coloring* is a function $\gamma : V \to [k]$. As before, for $X \subseteq V$ let $\varphi(X)$ be $\bigcup_{v \in X} \varphi(v)$. The coloring γ is *minimal* if $\gamma(V) = [k]$. Define $w(\gamma) = \sum_{(u,v) \in E(T)} w((u, v), \gamma(u), \gamma(v))$. The coloring γ is *convex* if for every $x, y, z \in V$ such that y is in the (unique) path from x to z it holds that $\gamma(x) = \gamma(z)$ implies $\gamma(x) = \gamma(y)$, or more informally: γ is convex if all its color classes induce a connected subtree. It is worth mentioning that in [19] the slightly different CONVEX TREE RECOLORING (CTR) was studied, but that CTR is a special case of CTC.

CONVEX TREE COLORING (CTC)

Input A tree $T = (V, E)$ with a weight function $w : E \times [k] \times [k] \to [c]$.
Question Is there a convex coloring $\gamma : V \to [k]$ with $w(\gamma) \leq c$?

Let us first note that we can safely assume that T is in fact binary, since if it is not, we can insert vertices and force them to have the same color by using an appropriate weight function. Also note we can restrict ourselves to minimal colorings by k vertices of degree one to arbitrary vertices and setting $w(\cdot, \cdot, \cdot =) = 0$. Hence the colors assigned to the added vertices does not matter at all, but if a color is not minimal, it can be extended to a minimal one using the new added vertices. Thus we can restrict ourselves to solving the following variant:

MINIMAL CONVEX BINARY TREE COLORING (MCBTC)

Input A binary tree $T = (V, E)$ with a weight function $w : E \times [k] \times [k] \to [c]$.
Question Is there a minimal convex coloring $\gamma : V \to [k]$ with $w(\gamma) \leq c$?

For a given coloring γ, use $\texttt{pce}(\gamma)$ to denote the number of poly-chromatic edges, i.e. the number of edges $(y, z) \in E(T)$ such that $\varphi(y) \neq \varphi(z)$.

Lemma 11 *The instance of* MCBTC *is a* yes*-instance if and only if there exists a k-coloring* γ *such that* $\texttt{pce}(\gamma) = k - 1$, *and* $w(\gamma) \leq c$.

Proof In a tree, the fact that γ gives k monochromatic connected components is equivalent with $\texttt{pce}(\gamma) = k - 1$ since after contracting all monochromatic edges, we again obtain a tree. □

Using Lemma 11, we can reduce our problem to determining the existence of a minimal k-coloring with $k - 1$ polychromatic edges. We will use Theorem 1: For notational convenience add a vertex to T, make it adjacent to an arbitrarily chosen other vertex of T and let the added vertex be the root of T. For the added edge, set all corresponding weights to 0. Define the universe $U = \{\gamma : V \to [k] \mid \texttt{pce}(\gamma) = k - 1 \wedge w(\gamma) \leq c\}$. Define the requirement space to be $[k]$ with for each $1 \leq v \leq k$ a requirement P_v being all elements γ of U with $v \in \gamma(V)$. Then $|\bigcap_{v \in [l]} P_v| > 0$ if and only if the current instance is a yes-instance by Lemma 11. Define $\mathcal{C}_F(s, W, g, p)$ as all k-colorings γ of $T[s]$ with $\texttt{pce}(\gamma) = g + [\gamma(\texttt{rt}(T)) \neq p]$, $\gamma(V(T[s])) \cap F = \emptyset$ and $w(\gamma) \leq W$ and let $c_F(s, W, g, p) = |\mathcal{C}_F(s, W, g, p)|$. Also note that $|\bigcap_{v \in F} \overline{P_v}| c_F(\texttt{rt}(T), c, k - 1, 1)$ (the color of the root is not relevant here so we just set it to 1). It remains to show how to compute $c_F(\texttt{rt}(T), c, k - 1, p)$:

Lemma 12

$$c_F(s, W, g, p) = \begin{cases} \left[w\big((s, \mathtt{pa}(s)\big), p, p\big) \leq W\right] & \text{if } s \text{ is a leaf} \wedge g = 0, \quad (10a) \\ \sum_{q \in [l] \setminus (F \cup \{p\})} [w((s, \mathtt{pa}(s)), q, p) \leq W] & \\ \quad \text{if } s \text{ is a leaf} \wedge g = 1, & (10b) \\ 0 \quad \text{if } s \text{ is a leaf} \wedge g > 1, & (10c) \\ \hat{c}_F(s, j, g, p) \quad \text{otherwise}, & (10d) \end{cases}$$

where

$$\hat{c}_F(s, W, g, p) = \sum_{\substack{g_1+g_2=l \\ W_1+W_2=W-w((s,\mathtt{pa}(s)),p,p)}} c_F(s_1, W_1, g_1, q) c_F(s_2, W_2, g_2, q) \quad (11)$$

$$+ \sum_{q \in [l] \setminus (F \cup \{p\})} \sum_{\substack{g_1+g_2=l-1 \\ W_1+W_2=W-w((s,\mathtt{pa}(s)),q,p)}} c_F(s_1, W_1, g_1, q) \times c_F(s_2, W_2, g_2, q). \quad (12)$$

Proof For Case 10a, the only possible valid coloring is the one where $\gamma(s) = \gamma(\mathtt{pa}(s))$. For Case 10b, any coloring γ with $\gamma(s) \neq \gamma(\mathtt{pa}(s))$ and low enough cost is counted. For Case 10c, no such a coloring exists since a leaf is only adjacent to one edge. For Case 10d, s is not a child and hence has two children s_1 and s_2. It is not too hard to see that the term in (11) is the number of colorings in $\mathcal{C}_F(s, W, g, p)$ where $\gamma(s) = \gamma(\mathtt{pa}(s))$ and that the term in (12) is the number of colorings in $\mathcal{C}_F(s, W, g, p)$ where $\gamma(s) \neq \gamma(\mathtt{pa}(s))$. □

Theorem 9 *The* Convex Tree Coloring *problem can be solved in* $\mathcal{O}^\star(2^k c)$ *time and* $\mathcal{O}^\star(c)$ *space.*

Proof By Lemma 11 and the discussion before it we can reduce Convex Tree Coloring to finding a minimal k-coloring γ with $k-1$ polychromatic edges and $w(\gamma) \leq c$. This can be solved using the IE-formulation as discussed above. The simplified problem can be solved in $\mathcal{O}^\star(c^2)$ time and $\mathcal{O}^\star(c)$ using Lemma 1 and using standard Fast Fourier Transform techniques this can be reduced to $\mathcal{O}^\star(c)$ time and space. The theorem then follows from Theorem 1. □

7 Counting Perfect Matchings

In this section we will count the number of perfect matchings in a graph $G = (V, E)$. We let $|V| = 2n$. First, we arbitrarily partition the vertex set V into A and B (thus, $A \cup B = V$, and $A \cap B = \emptyset$), with $|A| = |B| = n$. Let an *l-matching* of G be a perfect matching $M \subseteq E$ such that $|\{e \in M : |e \cap A| = 1\}| = l$, i.e. a perfect matching containing exactly l edges with exactly one endpoint in A.

We will need the following simple lemma:

Lemma 13 ([18], Lemma 4.10; [1], Theorem 4) *Given an independent set $S \subseteq V$, the number of perfect matching of G can be computed in $\mathcal{O}^\star(2^{2n-|S|})$.*

We will first give two algorithms that we combine later. Both algorithms solve the same problem, but with different running times. Afterwards we show how to combine the two to obtain the main result of this section.

Lemma 14 *l-matchings can be counted in $\mathcal{O}^*\left(\binom{n}{l}2^n\right)$ time and polynomial space.*

Proof Given a perfect matching M, define

$$L(M) = \left\{a \in A : \exists (a,b) \in M \wedge b \in B\right\}.$$

That is, $L(M)$ is the set of all vertices in A that are matched with a vertex in b in M. Then, if we define $f(X)$ as the number of perfect matchings M such that $L(M) = X$, then we can compute the number of l-matchings according to

$$\#l\text{-matchings} = \sum_{X \in \binom{A}{l}} f(X). \tag{13}$$

Define $g(X, B)$ as the number of perfect matchings in the graph obtained from $G[X \cup B]$ by making X into an independent set. Then $f(X) = \mathtt{pm}(A \setminus X)g(X, B)$ where $\mathtt{pm}(A \setminus X)$ is the number of perfect matching in $G[A \setminus X]$. Using Lemma 13 both $\mathtt{pm}(A \setminus X)$ and $g(X, B)$ can be computed in polynomial space and time $\mathcal{O}^\star(2^{n-l})$ and $\mathcal{O}^\star(2^n)$ respectively. Hence the lemma follows. □

We proceed to the next algorithm:

Lemma 15 *l-matchings can be counted in $\mathcal{O}^*\left(\binom{n}{\frac{n}{2}+\frac{l}{2}}2^n\right)$ time and polynomial space.*

Proof Arbitrarily choose a total ordering $\prec$ on A and use the shorthand $k = \frac{n+l}{2}$. We will use Theorem 1: Define the universe U to be

$$U = \left\{\left((u_i, v_i)\right)_{i \leq k} \in \left((A \times V) \cap E\right)^k \;\middle|\; \forall i < j : u_i \prec u_j\right\} \times E^{n-k}$$

and the requirement space to be V. Define P_v to be

$$\left\{\left((u_1, v_1), \ldots, (u_n, v_n)\right) \in U \;\middle|\; v \in \{u_1, \ldots, u_n\} \cup \{v_1, \ldots, v_n\}\right\}$$

for each $v \in V$. We claim that $|\bigcap_{v \in V} P_v|$ is $2^n(\frac{(n-l)}{2})!$ times the number of l-matchings. To see this, notice that for every l-matching there are exactly $2^n(\frac{n-l}{2})!$ elements in $|\bigcap_{v \in F} \overline{P_v}|$ obtained by all permutations of the $\frac{n-l}{2}$ edges contained in $G[B]$ and then flipping the order of the vertices of the edge. Moreover, every element in $|\bigcap_{v \in F} \overline{P_v}|$ can obtained in this way from exactly one l-matching.

Thus, combining Theorem 1 with the above we obtain that the number of l-matchings is

$$\frac{1}{2^n(\frac{n-l}{2})!}\sum_{F\subseteq V}(-1)^{|F|}\left|\bigcap_{v\in F}\overline{P_v}\right|. \tag{14}$$

We proceed by *trimming* (see also [6]): Observe that $|\bigcap_{v\in F}\overline{P_v}| = 0$ if $|F\cap A| > (n-k)$ since $u_1,\ldots,u_k \in A$ are distinct for any $((u_1,v_1),\ldots,(u_n,v_n)) \in U$. Hence for evaluating (14), we can restrict ourselves to sum over $F\subseteq V$ such that $|F\cap A|\leq (n-k)$. The number of $F\subseteq V$ such that $|F\cap A|\leq (n-k)$ is $\mathcal{O}^\star(\binom{n}{n-k}2^n) = \mathcal{O}^\star(\binom{n}{k}2^n)$. Hence to prove the lemma, it suffices to show that $|\bigcap_{v\in F}\overline{P_v}|$ can be computed in polynomial time for a fixed F.

Let $\{a_1,\ldots,a_n\}$ be obtained by sorting A with respect to the total ordering $\prec$. Define

$$p_F(r,m) = \left\{\big((u_i,v_i)\big)_{i\leq m} \in \Big(\big((A\setminus F)\times(V\setminus F)\big)\cap E\Big)^m \;\Big|\; \forall i < j : u_i \prec u_j \prec a_r\right\}.$$

Then we have $|\bigcap_{v\in F}\overline{P_v}| = p_F(n,k)|E(G[B\setminus F])|^{n-k}$ and it is easy to see that $p_F(n,k)$ can be computed according to

$$p_F(r,m) = \begin{cases} [m=1]d_{G[V\setminus F]}(v_1) & \text{if } r=1, \quad (15a)\\ p_F(r-1,m) + d_{G[V\setminus F]}p_F(r-1,m-1) & \text{otherwise}, \quad (15b)\end{cases}$$

hence $p_F(n,k)$ and $|\bigcap_{v\in F}\overline{P_v}|$ can be computed in polynomial time and the lemma follows. □

Theorem 10 *The number of perfect matchings of a graph G can be computed in $\mathcal{O}^\star(1.95^{|V(G)|})$ time and polynomial space.*

Proof Sum over all $0\leq l\leq n$ over the number of l-matchings computed by the algorithm of minimum running time among the algorithms of Lemmas 14 and 15. The running time of this algorithm is

$$\max_{0\leq l\leq n}\min\left\{\binom{n}{l}2^n, \binom{n}{\frac{1}{2}n+\frac{1}{2}l}2^n\right\} = \max_{0\leq l\leq n}\min\left\{\binom{n}{l}, \binom{n}{\frac{1}{2}n+\frac{1}{2}l}\right\}2^n. \tag{16}$$

Since $0\leq p\leq p'\leq \frac{n}{2}$ implies $\binom{n}{p}\leq\binom{n}{p'}$, (16) is maximized if $\min\{l,\frac{n}{2}-\frac{l}{2}\}$ is maximized, which is at $l=\frac{n}{3}$. Hence (16) is equal to $\binom{n}{\frac{n}{3}}2^n = \mathcal{O}^\star(1.89^n 2^n)$, where the latter is due to standard approximations (see for example Lemma 4 of [7]). □

8 Further Remarks

It is worth mentioning that, as the proofs in this paper might suggest, the results of this paper admit a generalization. The study of such a generalization was initiated in the conference version of this work and finally given in [17]. In the latter Theorem 3 is also improved by giving a $\mathcal{O}^\star(2^k c)$ time and polynomial space algorithm.

References

1. Amini, O., Fomin, F.V., Saurabh, S.: Counting subgraphs via homomorphisms. In: Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP), Part I. Lecture Notes in Computer Science, vol. 5555, pp. 71–82. Springer, Berlin (2009)
2. Björklund, A., Husfeldt, T.: Exact algorithms for exact satisfiability and number of perfect matchings. Algorithmica **52**(2), 226–249 (2008)
3. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion-exclusion. SIAM J. Comput. **39**(2), 546–563 (2009)
4. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets Möbius: fast subset convolution. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC), pp. 67–74. ACM, New York (2007)
5. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Computing the Tutte polynomial in vertex-exponential time. In: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (STOC), pp. 677–686. IEEE Comput. Soc., Los Alamitos (2008)
6. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Trimmed Moebius inversion and graphs of bounded degree. Theory Comput. Syst. **47**(3), 637–654 (2010)
7. Bodlaender, H.L., Kratsch, D.: An exact algorithm for graph coloring with polynomial memory. Technical Report UU-CS-2006-015, Department of Information and Computing Sciences, Utrecht University (2006)
8. Chung, F.R.K., Graham, R.L.: On the cover polynomial of a digraph. J. Comb. Theory, Ser. B **65**(2), 273–290 (1995)
9. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks **1**, 195–207 (1972). doi:10.1002/net.3230010302
10. Fernau, H., Gaspers, S., Raible, D.: Exact and parameterized algorithms for max internal spanning tree. In: Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science (WG). Lecture Notes in Computer Science, vol. 5911, pp. 100–111. Springer, Berlin (2009)
11. Fomin, F.V., Grandoni, F., Kratsch, D.: Faster Steiner tree computation in polynomial-space. In: Proceedings of the 16th Annual European Symposium on Algorithms (ESA). Lecture Notes in Computer Science, vol. 5193, pp. 430–441. Springer, Berlin (2008)
12. Fuchs, B., Kern, W., Mölle, D., Richter, S., Rossmanith, P., Wang, X.: Dynamic programming for minimum Steiner trees. Theory Comput. Syst. **41**(3), 493–500 (2007)
13. Gaspers, S., Saurabh, S., Stepanov, A.A.: A moderately exponential time algorithm for full degree spanning tree. In: Proceedings of the 5th International Conference on Theory and Applications of Models of Computation (TAMC). Lecture Notes in Computer Science, vol. 4978, pp. 479–489. Springer, Berlin (2008)
14. Karp, R.M.: Dynamic programming meets the principle of inclusion and exclusion. Oper. Res. Lett. **1**, 49–51 (1982)
15. Kohn, S., Gottlieb, A., Kohn, M.: A generating function approach to the traveling salesman problem. In: Proceedings of the 1977 Annual Conference (ACM), pp. 294–300. ACM, New York (1977). doi:10.1145/800179.810218
16. Koivisto, M.: Partitioning into sets of bounded cardinality. In: Proceedings of the 4th International Workshop Parameterized and Exact Computation (IWPEC). Lecture Notes in Computer Science, vol. 5917, pp. 258–263. Springer, Berlin (2009)
17. Lokshtanov, D., Nederlof, J.: Saving space by algebraization. In: Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC), pp. 321–330. ACM, New York (2010)
18. Nederlof, J.: Inclusion exclusion for hard problems. Master's thesis, Utrecht University (August 2008)
19. Ponta, O., Hüffner, F., Niedermeier, R.: Speeding up dynamic programming for some $\mathcal{NP}$-hard graph recoloring problems. In: Proceedings of the 5th International Conference on Theory and Applications of Models of Computation (TAMC), pp. 490–501. Springer, Berlin (2008)
20. Woeginger, G.J.: Space and time complexity of exact algorithms: some open problems (invited talk). In: First International Workshop on Parameterized and Exact Computation (IWPEC). Lecture Notes in Computer Science, vol. 3162, pp. 281–290. Springer, Berlin (2004)