

REPORTS IN INFORMATICS

ISSN 0333-3590

**Fast polynomial-space algorithms using Möbius
inversion: Improving on Steiner Tree and
related problems**

Jesper Nederlof

REPORT NO 386

February 2009



Department of Informatics
UNIVERSITY OF BERGEN
Bergen, Norway

This report has URL

<http://www.ii.uib.no/publikasjoner/texrap/pdf/2009-386.pdf>

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at

<http://www.ii.uib.no/publikasjoner/texrap/>.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Fast polynomial-space algorithms using Möbius inversion: Improving on Steiner Tree and related problems

Jesper Nederlof *

Department of Informatics
University of Bergen
N-5020 Bergen
Norway

February 18, 2009

Abstract

We give an $\mathcal{O}^*(2^k)$ -time algorithm for STEINER TREE using polynomial space, where k is the number of terminals. To obtain this result, we apply Möbius inversion, also known as Inclusion-Exclusion, to the famous Dreyfus-Wagner recurrence. Among our results are also polynomial-space $\mathcal{O}^*(2^n)$ algorithms for several \mathcal{NP} -complete spanning tree and partition problems. With these results, we solve open problems from three papers, and improve results from a STOC 07 paper.

The previous best known algorithms for these problems use the technique of dynamic programming among subsets, and they require exponential space. We use Möbius inversion, also known as Inclusion-Exclusion, as an algebraic tool to get improved algorithms. More specifically, we apply the zeta-transform to the recurrence associated with the dynamic programming approach and show that the number of states becomes polynomially bounded.

1 Introduction

One of the most widely used techniques for achieving efficient exponential time algorithms for \mathcal{NP} -hard problems is dynamic programming among subsets, but unfortunately an exponential storage requirement seems to be inherent to this technique. As mentioned by Woeginger [Woe04] this requirement makes them useless in practice. Therefore polynomial-space exact algorithms have already been studied for several \mathcal{NP} -hard problems [Woe04, Kar82, KGK77, FGK08, BK06, BH06b]. Hence, from both a theoretical and a practical perspective it is crucial to identify those dynamic programming algorithms that can be improved to require polynomial space, preferably maintaining the best known upper bound on the running time. In this paper we improve several algorithms in this way.

In 2006, Björklund et al. [BH06b] and independently Koivisto [Koi06] drew new attention to the technique of Inclusion-Exclusion: they both gave $\mathcal{O}^*(2^n)$ -time algorithms¹ for several set partition problems, the most prominent one being k -COLORING. Björklund et al. [BH06b] also

*Email: Jesper.Nederlof@ii.uib.no, WWW: <http://folk.uib.no/jne061/>

¹We use the notation $\mathcal{O}^*(c^n)$ for $c^n n^{\mathcal{O}(1)}$, and n denotes the number of nodes of the graph

	Problem	References
$\mathcal{O}^*(2^k)$	STEINER TREE ^{*2}	[BHKK07, FGK08]
$\mathcal{O}^*(2^n)$	DEGREE CONSTRAINED SPANNING TREE*	[GSS08]
	MAX INTERNAL SPANNING TREE*	[FRGS08, Jvw90]
	#c-SPANNING FORESTS*	[BH06b]
	COVER POLYNOMIAL*	[BHKK08a]
	MIN CONNECTED SPANNING SUBHYPERGRAPH ^{*3}	[BHKK07]
$\mathcal{O}^*(2.24^n)$	MIN SPANNING TREE IN HYPERGRAPH ^{*3}	[BHKK07]
	#HAMILTONIAN PATH	[Kar82, KGK77]
	#PERFECT MATCHING	[BH06a]
	k -COLORING	[BH06b, Koi06]

Table 1: An overview of problems that can be solved efficiently in polynomial space with the given time bound using Inclusion-Exclusion. For the problems indicated with a *, we give the first polynomial space algorithm with the given running time.

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 of Karp [Kar82] and (implicitly) Kohn et al. [KGK77] and for #PERFECT MATCHING of Björklund et al. [BH06a].

In this paper we show that some dynamic programming algorithms can be improved to obtain polynomial-space algorithms with the same worst-case running time. The paper heavily relies on the work of Björklund et al. [BHKK07, BH06a, BHKK08a, BH06b, Koi06]. Our results can be read from Table 1: we add many problems to the list of polynomial space Inclusion-Exclusion algorithms (note that this list is not extensive). The contributions of this paper are marked with an asterisk, and we also give some references to previous work on the problems.

STEINER TREE is one of the most well-studied NP-complete problems. The classical Dreyfus-Wagner [DW72] dynamic programming algorithm has been the fastest algorithm for over 30 years, but has recently been improved by Björklund et al. in [BHKK07]. In [FGK08] Fomin et al. initiated the study of polynomial space algorithms for this problem. They give polynomial space algorithms with running time bounded by $\mathcal{O}(5.96^k n^{\mathcal{O}(\log k)})$ and $\mathcal{O}^*(1.60)^n$ where k and n are respectively the number of terminals and nodes of the graph. They pose the open question whether STEINER TREE is fixed parameter tractable with respect to k . We answer this question affirmatively by providing a polynomial-space algorithm that requires $\mathcal{O}^*(2^k)$ time. Using techniques of [FGK08], this also gives a polynomial space $\mathcal{O}^*(1.3533^n)$ algorithm.

The MAX INTERNAL SPANNING TREE (MIST) and DEGREE CONSTRAINED SPANNING TREE (DCST) problems are natural generalizations of HAMILTONIAN PATH. In [FRGS08], Fernau et al. ask if there exists an $\mathcal{O}^*(2^n)$ -time algorithm to solve MIST. In [GSS08], Gaspers et al. ask if there exists an $\mathcal{O}^*(2^n)$ -time algorithm solving DCST. We answer both questions by giving polynomial-space algorithms with $\mathcal{O}^*(2^n)$ running time.

The COVER POLYNOMIAL of a directed graph introduced by Graham and Chung [CG95] generalizes all problems that can be solved using the two operations named deletion and contraction, and is designed to be the directed analogue of the Tutte polynomial. We improve the $\mathcal{O}^*(2^n)$ -time exponential-space algorithm of Björklund et al. [BHKK08a] to a polynomial-space $\mathcal{O}^*(2^n)$ -time algorithm. We also give the same improvement for counting the number of spanning forests, which is one particular case of the Tutte polynomial. For more information about the Tutte polynomial we refer to [BHKK08a].

Following [BHKK07], we also give the first $\mathcal{O}^*(2^n)$ -time polynomial-space algorithms for two

² k stands for the number of terminals.

³We assume the hypergraph is sparse and the range of the weight function is bounded.

spanning subhypergraph problems in sparse graphs with bounded weight functions. For clearness we mention that if the hypergraph is dense, the algorithm of [BHKK07] is faster.

The paper is organised as follows: in Section 2 we recall the principle of Inclusion-Exclusion and explain the well-known Hamiltonian path algorithm. After this we provide a natural extension by introducing the concept of *branching walks* and give the resulting improved algorithms. In the next section we show the Inclusion-Exclusion algorithms can be obtained from dynamic programming algorithms by taking the *zeta transform* of the associated recurrences. After this, we give a more structural approach to the subset products introduced in [BHKK07] and provide more applications.

We use the following definitions: for any set S we denote 2^S for the *power set* of S , i.e. the set consisting of all subsets of S . Given a (directed) graph $G = (V, E)$ the graph *induced by* X , where $X \subseteq V$, is the graph $G[X]$ with nodeset X and all edges in E only adjacent to nodes in X . A *walk of length* k in a graph $G = (V, E)$ is a tuple $W = (v_1, \dots, v_{k+1}) \in V^{k+1}$ such that $(v_i, v_{i+1}) \in E$ for each $0 \leq i \leq k$. W is *cyclic* if $v_1 = v_{k+1}$. We denote by $[boolean]$ 1 if *boolean* is true, and 0 otherwise. We also use the well-known concept of *generating functions*: with a sequence a_n , we associate the power series $\mathcal{A} = \sum_{i=0}^{\infty} a_i z^i$, where z is implicit. We denote by $\{z^i\}\mathcal{A}$ the coefficient of z^i . The motivation of this notation is the exploitation of the properties of multiplication of power series; for more information we refer to any textbook on discrete mathematics, for example [GKP94].

2 Inclusion-Exclusion formulations

Let us start this section by stating the principle of Inclusion-Exclusion. The following theorem can be found in many textbooks on discrete mathematics; we give the standard proof for completeness. For a more combinatorial proof we also refer to Bax [Bax96].

Theorem 1 (Folklore). *Let U be a set and $A_1, \dots, A_n \subseteq U$. With the convention $\bigcap_{i \in \emptyset} A_i = U$, the following holds:*

$$\left| \bigcap_{i=1}^n A_i \right| = \sum_{X \subseteq \{1, \dots, n\}} (-1)^{|X|} \left| \bigcap_{i \in X} \overline{A_i} \right| \quad (1)$$

Proof. We look at the contribution of each $e \in U$ to the right-hand side of the equation. Suppose $e \in \bigcap_{i=1}^n A_i$, then e is only counted in $|\bigcap_{i \in X} \overline{A_i}|$ if $X = \emptyset$, hence its contribution is 1. Now let $X \neq \{1, \dots, n\}$ be the inclusion-wise maximal set such that $e \in \bigcap_{i \in X} \overline{A_i}$. Note that for each $X' \subseteq X$, e is also contained in $\bigcap_{i \in X'} \overline{A_i}$. Hence, the contribution of e is $\sum_{i=0}^{|X|} (-1)^i \binom{|X|}{i}$, which is 0 due to the binomial theorem. \square

In this paper, we call any application of the above Theorem an *IE-formulation*. In this context we will refer to the set U as the *universe*, and to A_1, \dots, A_n as the *requirements*. Moreover, we call the task of computing $|\bigcap_{i \in X} \overline{A_i}|$ for an arbitrary $X \subseteq \{1, \dots, n\}$ the *simplified problem*. Note that if the simplified problem can be computed in polynomial time, there exists an $\mathcal{O}^*(2^n)$ -time polynomial-space algorithm that evaluates Equation 1.

We mention that it is also possible to break Theorem 1 down in smaller steps, i.e. we choose a subset of $\{1, \dots, n\}$ and apply the theorem. This has recently been used for a faster exact algorithm for DOMINATING SET in van Rooij et al. [vRNvD]. We refer the more interested reader also to [Bax96]. Other methods to speed up IE-formulations are given by Björklund et al. [BH06a, BHKK08b] and is surveyed in Nederlof [Ned08].

We continue this section by giving some IE-formulations. The first one is well-known, but very illustrative and it will be extended in the next subsections.

2.1 Hamiltonian Paths

Given a graph $G = (V, E)$, a Hamiltonian path is a walk that contains each node exactly once. The $\#$ HAMILTONIAN PATH problem is to count the number of Hamiltonian paths. The following IE-formulation is due to Karp [Kar82]: define the universe U as all walks of length $n - 1$ in G , and define A_i as all of walks of length $n - 1$ that contain node i , for each $i \in V$. With these definitions, the left-hand side of Equation 1, $|\bigcap_{i \in V} A_i|$, is the number of Hamiltonian paths from s . Now it remains to show how to solve the simplified problem: given $s \in V \setminus X$, let $w_k(s)$ be the number of walks from s of length k in $G[V \setminus X]$. $w_k(s)$ admits the following recurrence:

$$w_k(s) = \begin{cases} 1 & \text{if } k=0 \\ \sum_{(s,t) \in E'} w_{k-1}(t) & \text{otherwise} \end{cases} \quad (2)$$

where E' is the edge set of the graph $G[V \setminus X]$. We mention that $w_k(s)$ is $\mathcal{O}((n - 1)^{n-1})$. To see this, note this bound is tight in a complete graph. Because of this the number of bits needed to represent $w_k(s)$ is polynomially bounded. Furthermore, we have:

$$|\bigcap_{i \in X} \overline{A_i}| = \sum_{s \in X} \frac{w_{n-1}(s)}{2}$$

Hence, the simplified problem can be solved in polynomial time using dynamic programming on Equation 2. Thus it takes $\mathcal{O}^*(2^n)$ time and polynomial space to evaluate the right-hand side of Equation 1 and to solve $\#$ HAMILTONIAN PATH in polynomial space within the same timebound.

2.2 Steiner trees

Assume a graph $G = (V, E)$ is given. The STEINER TREE problem is the following: given some set of *terminals* $R \subseteq V$ and an integer c , does there exist a connected subtree $T = (V_T, E_T)$ of G such that $R \subseteq V_T$ and $|E_T| \leq c$?

In this subsection we will give an extension of the previous subsection to obtain an IE-formulation for STEINER TREE. We introduce the following definition:

Definition 2. For any $s \in V$, a branching walk in G from s is a tuple $B = (s, \langle B_1, \dots, B_l \rangle)$ where $\langle B_1, \dots, B_l \rangle$ is a sequence of branching walks from neighbors of s in G . The length of B , denoted with $|B|$, is defined as:

$$|B| = \begin{cases} 0 & \text{if } l = 0 \\ \sum_{i=1}^l 1 + |B_i| & \text{otherwise} \end{cases}$$

The highest degree $\Delta(B)$ of B is the maximum of $l + 1$ and $\max_{1 \leq i \leq l} \Delta(B_i)$. The number of internal nodes of B $\iota(B)$ is $[l \neq 0] + \sum_{i=1}^l \iota(B_i)$. We also define $G[B] = (V[B], E[B])$ where $G[B]$ is the minimal subgraph such that B is a branching walk in $G[B]$, and call B an internal node if $l > 0$.

This definition is particularly useful in combination with the following lemma:

Lemma 3. Let $s \in R$. There exists a tree $T = (V_T, E_T)$ of G such that $|E_T| \leq c$ and $R \subseteq V_T$ if and only if there exists a branching walk B from s such that $R \subseteq V[B]$ and $|B| \leq c$.

Proof. Note that the first part follows from the observation that each T directly corresponds to a branching walk such that $G[B] = T$ and $|B| = |E_T|$. For the second part, notice we can choose an arbitrary spanning tree of $G[B]$ for T . \square

Consider the following IE-formulation: let $s \in T$ be an arbitrarily chosen terminal, and define the universe U as all branching walks from s of length c . For each $i \in T$, define a requirement A_i that consists of all elements of U that contain terminal i . It follows that the left-hand side of Equation 1 is the number of branching walks that contain all terminals. Due to Lemma 3, this is larger than 0 if and only if the instance of STEINER TREE is a yes-instance.

It remains to show how the simplified problem can be solved. Use $b_j(s)$ for the number of branching walks from s of length j in $G[V \setminus X]$. Note that the simplified problem is to compute $b_c(s) = |\bigcap_{i \in X} \overline{A_i}|$, for a given X . Define $\mathcal{B}(s)$ as the generating function of $b_j(s)$, i.e.:

$$\mathcal{B}(s) = \sum_{j=0}^{\infty} b_j(s) z^j$$

Now $b_c(s)$ can be directly deduced from $\mathcal{B}(s)$ (i.e. $\{z^c\}\mathcal{B}(s)$), and $\mathcal{B}(s)$ can be computed in polynomial time using the following two lemmas:

Lemma 4. *Let E' be the edge set of $G[V \setminus X]$, then*

$$\mathcal{B}(s) = 1 + \sum_{(s,t) \in E'} \mathcal{B}(t) \mathcal{B}(s) z \quad (3)$$

Proof. There is one branching walk of length 0 from s and this is also the coefficient of z^0 on the right-hand side of Equation 3. If the length j is larger than 0, there exists a first branching walk from a neighbor t of s , B_1 , in the sequence of Definition 2 and each branching walk from s of length j is a combination of a branching walk from some neighbor t and a branching walk from s such that their sizes sum up to $j - 1$, and all of such combinations contribute to the coefficients of $\{z^j\}$ of the left-hand and the right-hand side. \square

Lemma 5. *The number of bits needed to represent $\mathcal{B}(s)$ is $\mathcal{O}(c^2 \lg n)$*

Proof. We prove the lemma by giving an upper bound of $\mathcal{O}(c \lg n)$ on the number of bits needed to represent $\{z^c\}\mathcal{B}(s)$. Suppose $G[V \setminus X] = K_n$, i.e. a complete graph of n nodes. Because every node in G is equivalent, the parameter s is not important anymore so we drop it. Now we obtain $\mathcal{B} = 1 + n\mathcal{B}^2 z$, and one can prove an upper bound of $\mathcal{O}((nc)^c)$ on $\{z^c\}\mathcal{B}(s)$ with induction on c . \square

Now we combine the considerations mentioned above to obtain the following theorem:

Theorem 6. *There exists an algorithm that uses polynomial space for STEINER TREE with $2^k n^{\mathcal{O}(1)}$ running time, where k is the number of terminals.*

Proof. Due to Lemma 3, the mentioned IE-formulation indeed solves STEINER TREE. It remains to show that the right-hand side of Equation 1 can be evaluated in the given timebound: to see this, recall that $b_c(s) = |\bigcap_{i \in X} \overline{A_i}|$. Now use dynamic programming on Equation 3 to compute $b_c(s) = \{z^c\}\mathcal{B}(s)$. We get states for each $i \leq c$ and $s \in V \setminus X$. Moreover, the number of bits needed to represent $\mathcal{B}(s)$ is polynomially bounded, so we can use Equation 3 to compute a new state in linear time. \square

We mention that it is also possible to extend the algorithm for the weighted version of STEINER TREE. Furthermore, the following result is a direct consequence of Theorem 6 and the methods of Fomin et al. [FGK08]:

Corollary 7. *The STEINER TREE problem can be solved in $\mathcal{O}^*(1.3533^n)$ time using polynomial space.*

2.3 Further IE-formulations

In this section we will give IE-formulations for two problems that lead to algorithms running in $\mathcal{O}^*(2^n)$ -time with polynomial space. In the following assume we are given a graph $G = (V, E)$ and an integer c . In both IE-formulations we define A_i for each $i \in V$ as all elements of U that contain node i . The universe U itself will be tailor-made for each problem.

Degree constrained spanning tree

The DEGREE CONSTRAINED SPANNING TREE problem, also called MIN-MAX DEGREE SPANNING TREE, asks whether G has a spanning tree with highest degree at most c . We assume $c > 1$. Define U as all branching walks of length $n - 1$ with highest degree at most c from s , where s can be any $s \in V$. We use the following lemma:

Lemma 8. *There exists a branching walk $B \in U$ if and only if there exists a spanning tree with highest degree at most c .*

Proof. The first part follows from the observation that each spanning tree T directly corresponds to a branching walk B with $G[B] = T$ from a leaf of T , such that the maximum degree of T is $\Delta(B)$ and $|B| = n - 1$, using the assumption $c > 1$. For the second part, we can choose $G[B]$ for T . \square

Define $d_j(g, s)$ as the number of branching walks $B = (s, \langle B_1, \dots, B_l \rangle)$ such that $l \leq g$, $|B| = j$ and the maximum of the highest degrees of B_1, \dots, B_l is at most $c - 1$. Let $\mathcal{D}(g, s)$ be the generating function of $d_j(g, s)$. Note that the simplified problem is to compute $\sum_{s \in V \setminus X} \{z^{n-1}\} \mathcal{D}(c - 1, s)$, and the following holds:

$$\mathcal{D}(g, s) = [g \geq 0] + \sum_{(s,t) \in E'} \mathcal{D}(c - 1, t) \mathcal{D}(g - 1, s) z \quad (4)$$

where E' is defined as the edge set of $G[V \setminus X]$. To see that this equation holds, notice that $d_0(g, s) = [g \geq 0]$ by definition and if $j > 0$, $d_j(g, s)$ is a combination of two branching walks: in the branching walk leaving s we are allowed to choose $c - 1$ other neighbors, and in the other one we are allowed to use one neighbor less than before.

Max internal spanning tree

The MAX INTERNAL SPANNING TREE asks whether G has a spanning tree with at least c internal nodes. Now define the universe U as all branching walks $B = (s, \langle B_1, \dots, B_l \rangle)$ of length $n - 1$ with at least c internal nodes not equal to s , where s can be any $v \in V$.

Lemma 9. *There exists a branching walk $B \in U$ if and only if there exists a spanning tree of G with at least c internal nodes.*

Proof. The first part follows from the observation that each spanning tree T with at least c internal nodes directly corresponds to a branching walk $B \in U$ from a leaf of T such that $G[B] = T$. For the second part, we can choose $G[B]$ for T . \square

Define $m_{g,j}(s)$ as the number of branching walks of length j from s having g internal nodes. Note that $m_{g,0}(s) = [g = 0]$. Let $\mathcal{M}(s)$ be the generating function of $m_{g,j}(s)$, i.e. $\mathcal{M}(s) = \sum_{g,j=0}^{\infty} m_{g,j}(s) y^g z^j$. The following holds:

$$\mathcal{M}(s) = 1 + \sum_{(s,t) \in E'} \mathcal{M}(t) \mathcal{M}(s) y z \quad (5)$$

where E' is defined as the edge set of $G[V \setminus X]$. The equation holds because we count all possible distributions of the length and the number of internal nodes in the branching walks from s and the one from t , and s is automatically counted as an internal node. Now the simplified problem is to compute $\sum_{i=c}^n \sum_{s \in X} \{y^i z^{n-1}\} \mathcal{M}(s)$.

Theorem 10. DEGREE CONSTRAINED SPANNING TREE and MAX INTERNAL SPANNING TREE can be solved in $\mathcal{O}^*(2^n)$ time and polynomial space

Proof. Due to Lemmas 8 and 9, the IE-formulations decide whether the given problems are yes-instances. Hence, due to Theorem 1, it suffices to show that the right-hand side of Equation 1 can be computed in the given time and space bound. We can compute $\mathcal{D}(c, s)$ and $\sum_{v \in V \setminus X} \mathcal{M}(v)$ with Equations 4 and respectively 5: the number of bits needed to represent both \mathcal{D} and \mathcal{M} is polynomially bounded because they are smaller than \mathcal{B} and Lemma 5. Hence they can be computed in polynomial time and space with dynamic programming. Now the results follow. \square

3 Möbius inversion

In this section we study an algebraic equivalent of Inclusion-Exclusion, called Möbius inversion. Basically, it consists of the following two transforms:

Definition 11. Given a function $f : 2^V \rightarrow \mathbb{Z}$, the zeta-transform ζf and the Möbius-transform μf are defined as follows:

$$\zeta f(V) = \sum_{X \subseteq V} f(X) \quad \mu f(V) = \sum_{X \subseteq V} (-1)^{|V \setminus X|} f(X)$$

Now the principle of Möbius inversion can be formulated as the following theorem. Although it is already folklore, we make the relation to Inclusion-Exclusion more clear by giving a small proof.

Theorem 12 (Folklore). The Möbius-transform is the inverse of the zeta-transform.

Proof. Choose U and A_1, \dots, A_n such that for each $X \subseteq V$ we have:

$$f(X) = |\bigcap_{i \in X} A_i \setminus \bigcup_{i \in V \setminus X} A_i|$$

This can be done by making $f(X)$ elements for each $X \subseteq V$. Now $\zeta f(X) = |\bigcap_{i \in V \setminus X} \overline{A_i}|$, and hence $\mu \zeta f(V)$ is equal to the right-hand of Equation 1. Since $f(V)$ is also equal to the left-hand side of Equation 1, the result follows from Theorem 1. \square

Note that using the terminology of the previous section, any IE-formulation is equivalent to applying Möbius inversion and the simplified problem is to compute $\zeta f(X)$. Now we will reobtain the IE-formulation of Karp [Kar82] discussed in Subsection 2.1 by applying Möbius inversion to the Held-Karp dynamic programming approach of [HK62]. To our knowledge, we are the first that relate these classical algorithms to each other.

Hamiltonian path revisited

Now let us again consider #HAMILTONIAN PATH. Let $h^k(s, R)$ be the number of walks of k edges from s that at least contain everything in R , defined for $s \notin R$. Slightly adjusting the recurrence of [HK62], we have the following equation:

$$h_k(s, R) = \begin{cases} [R=\emptyset] & \text{if } k = 0 \\ \sum_{(s,t) \in E} h_{k-1}(t, R \setminus t) & \text{otherwise} \end{cases} \quad \begin{matrix} (6a) \\ (6b) \end{matrix}$$

Walks of 0 edges only contain the empty set, hence Case 6a equals $[R = \emptyset]$. In Case 6b we try all neighbors and continue with counting walks with $k - 1$ edges that contain $R \setminus t$. Now we take the zeta-transform on both sides of Equation 6, and obtain:

$$\zeta h_k(s, R) = \begin{cases} 1 & \text{if } k = 0 \\ \sum_{(s,t) \in E} \zeta h_{k-1}(t, R \setminus u) & \text{otherwise} \end{cases}$$

Notice that the parameter R is redundant and can be removed, and we reobtain Equation 2. Thus $\zeta h_{k-1}(s) = w_k(s)$, and we obtained Karp's IE-formulation for Hamiltonian path of Subsection 2.1.

3.1 Subset products

An application for which Möbius inversion is particularly suited is the computation of subset products, introduced by Björklund et al.. We will only use the following two products, but mention that the same method is applicable to the other ones:

Definition 13 ([BHKK07]). *Given two functions $f, g : 2^V \rightarrow \mathbb{Z}$, the cover product $(f *_c g)$ and the subset convolution $(f * g)$ are defined as follows:*

$$(f *_c g)(V) = \sum_{\substack{A, B \subseteq V \\ A \cup B = V}} f(A)g(B) \quad (f * g)(V) = \sum_{X \subseteq V} f(X)g(V \setminus X)$$

Defining $F(X) = z^{|X|}f(X)$ and $G(X) = z^{|X|}g(X)$, the cover product and subset convolution can be related to each other with:

$$(f * g)(V) = \{z^{|V|}\}(F *_c G)(V)$$

Assuming f and g can be evaluated in polynomial time, the obvious way to compute $(f *_c g)(V)$ would take $\mathcal{O}^*(3^n)$ time. In [BHKK07], Björklund et al. implicitly use the following theorem in order to obtain $\mathcal{O}^*(2^n)$ exponential space algorithms:

Theorem 14 ([BHKK07]). *Given two functions $f, g : 2^V \rightarrow \mathbb{Z}$, the following holds:*

$$\zeta(f *_c g) = (\zeta f)(\zeta g)$$

Proof. Consider the following rewriting:

$$\zeta(f *_c g) = \sum_{X \in V} \sum_{\substack{A, B \subseteq X \\ A \cup B = X}} f(A)g(B) = \left(\sum_{A \subseteq V} f(A) \right) \left(\sum_{B \subseteq V} f(B) \right)$$

The first equality follows from the definitions. For the second equality notice that for each $A, B \subseteq V$, there exists exactly one $X \subseteq V$ such that $A, B \subseteq X$ and $A \cup B = V$, hence we actually sum over each combination of two subsets A and B , which is expressed in the latter expression. \square

Steiner Tree revisited

Now let us again consider STEINER TREE. Let $s(v, R)$ be the size of the smallest Steiner tree of G with terminal set $R \cup v$. Our starting point is a slightly adjusted version of the famous Dreyfus-Wagner recurrence [DW72]:

$$s(v, R) = \begin{cases} 0 & \text{if } R = \emptyset & (7a) \\ \min_{(v,w) \in E} \min_{X \subseteq R \setminus w} 1 + s(w, X) + s(v, R \setminus w \setminus X) & \text{otherwise} & (7b) \end{cases}$$

That is, if $R = \emptyset$, the smallest Steiner tree contains no edges and hence has size 0. If $R \neq \emptyset$ we choose an edge (v, w) to be in the Steiner tree and continue with looking for two Steiner trees that together contain all terminals, such that their sizes are $c - 1$.

Now suppose we want to solve its decision variant: is there a Steiner tree with cost c for a given c ? Note that the answer is yes if and only if $s^c(c, R) > 0$, where we define $s^0(v, R) = [R = \emptyset]$, and for $c > 0$:

$$s_c(v, R) = \sum_{(v,w) \in E} \sum_{X \subseteq R \setminus w} \sum_{i=0}^{c-1} s_i(w, X) s_{c-1-i}(v, R \setminus w \setminus X)$$

The next step is to rewrite the equation in a more compact form using the generating function $\mathcal{S}(v, R) = \sum_{j=0}^{\infty} s_j(v, R) z^j$:

$$\mathcal{S}(v, R) = [R = \emptyset] + \sum_{(v,w) \in E} \sum_{X \subseteq R \setminus w} \mathcal{S}(w, X) \mathcal{S}(v, R \setminus w \setminus X) z$$

Now we recognize the convolution and using Equation 14 and Theorem 14 we obtain:

$$\zeta \mathcal{S}(v, R) = 1 + \sum_{(v,w) \in E} \zeta \mathcal{S}(w, R \setminus w) \zeta \mathcal{S}(v, R \setminus w) z$$

which is the same as Equation 3. Hence, we reobtained the IE-formulation of Subsection 2.2 without using the notion of branching walks and lemmas 4 and 3.

3.2 Further applications

In this subsection we give some applications of the methods considered in the previous subsection, continuing the work of Björklund et al. [BHKK08a, BHKK07].

Cover Polynomial

We use $x^{\underline{i}}$ for the falling factorial $\frac{x!}{(x-i)!}$. A Hamiltonian cycle of a graph is a cyclic walk that contains all nodes exactly once. The cover polynomial of a given directed graph $D = (V, A)$ and integers i and j , is to compute [CG95, BHKK08a]

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

where $c_D(i, j)$ can be interpreted as the number of ways to partition D into i directed paths and j directed cycles, i.e.:

$$c_D(i, j)(V) = \frac{1}{i!j!} \sum_{X_1, \dots, X_{i+j}} \left(\prod_{p=1}^i h(X_p) \right) \left(\prod_{p=i+1}^{i+j} c(X_p) \right)$$

where the sum is over all partitions X_1, \dots, X_{i+j} of V , $h(X_p)$ is the number of Hamiltonian paths of $D[X_p]$ and $c(X_p)$ is the number of Hamiltonian cycles of $D[X_p]$.

Define $\mathcal{H}(X) = z^{|X|}h(X)$ and $\mathcal{C}(X) = z^{|X|}c(X)$, then we can get the following formula by iteratively applying Theorem 14:

$$\zeta_{c_D}(i, j)(V) = \frac{1}{i!j!} \{z^n\} \zeta \mathcal{H}^i(V) \zeta \mathcal{C}^j(V)$$

Note that $\mathcal{H}(V) = \sum_{s \in V} \sum_{g=0}^{\infty} w_{g-1}(s) z^g$ and $\mathcal{C}(V) = \sum_{s \in V} \sum_{g=0}^{\infty} c_g(s) z^g$, where $c_g(s)$ is the number of cyclic walks with length g from s . Since c_g can be computed in polynomial time using a variant of Equation 2, $\zeta_{c_D}(i, j)(V)$ can be computed in polynomial time.

#c-Spanning forests

A c -spanning forest of $G = (V, E)$ is an acyclic subgraph of G with exactly c connected components. Denote by $\tau(c, X)$ the number of c -spanning forests of $G[X]$. The # c -SPANNING FORESTS problem is to compute $\tau(c, V)$. Notice that:

$$\tau(c, X) = \frac{1}{c!} \sum_{X_1, \dots, X_c} \tau(1, X_i)$$

where the sum is over all partitions X_1, \dots, X_c of X . From Theorem 14 it follows that $\zeta \tau(c, X)$ is equal to $\{z^{|X|}\} (\zeta \mathcal{T}(1, X))^c$ where $\mathcal{T}(1, X) = z^{|X|} \tau(1, X)$. We mention that $\zeta \mathcal{T}(1, X)$ can be computed in polynomial time very similar to the algorithms of the previous section, we omit the details.

Spanning subhypergraphs

Let us start with recalling some hypergraph terminology: a *hypergraph* is a tuple $H = (V, \mathcal{E})$ where $\mathcal{E} \subseteq 2^V$. The *incidence graph* $I[H]$ is the bipartite graph with nodes for any $v \in V$ and edge $E \in \mathcal{E}$, and edges between v and E if $v \in E$.

Suppose we are given a hypergraph H and weight function $w : \mathcal{E} \rightarrow \{1, \dots, M\}$ and an integer c . The MINIMUM CONNECTED SPANNING SUBHYPERGRAPH (MCSH) problem asks whether there exists $\mathcal{F} \subseteq \mathcal{E}$ such that $I[(V, \mathcal{F})]$ is connected and $\sum_{E \in \mathcal{F}} w(E) = c$.

Define $q_c(s, R)$ for $s \notin R$ as the number of $\mathcal{F} \subseteq \mathcal{E}$ such that $I[(R \cup s, \mathcal{F})]$ is connected and $\sum_{E \in \mathcal{F}} w(E) = c$. Let $\mathcal{Q}(s, R)$ be the generating function of $q_c(s, R)$. Note that:

$$\mathcal{Q}(s, R) = [R = \emptyset] + \sum_{s \in E \in \mathcal{E}} (\mathcal{Q}(v_1) *_c \mathcal{Q}(v_2) *_c \dots *_c \mathcal{Q}(v_{|E|-1})) (R \setminus E) z^{w(E)}$$

where we assume without loss of generality that $E \setminus s$ is $v_1, \dots, v_{|E|-1}$. To see that this equation holds, note that $q_0(s, R) = [R = \emptyset]$ and if $c > 0$ we choose a hyperedge containing s and proceed with finding a set of subhypergraphs containing neighbors from s that together cover the nodes $R \setminus E$.

Now we take the zeta transform, and using Theorem 14 we obtain:

$$\zeta \mathcal{Q}(s, R) = 1 + \sum_{s \in E \in \mathcal{E}} \prod_{i=1}^{|E|-1} \zeta \mathcal{Q}(v_i, R \setminus s) z^{w(E)}$$

Assuming M the number of edges is bounded polynomially in $|V|$, $\zeta \mathcal{Q}(s, R)$ can be computed in polynomial time. Similar to MCSH, MINIMUM SPANNING TREE IN HYPERGRAPHS (MSTH)

asks whether there exists $\mathcal{F} \subseteq \mathcal{E}$ such that $I[(V, \mathcal{F})]$ is tree and $\sum_{E \in \mathcal{F}} w(E) = c$. For solving this problem, use the same method but replace the cover product with a subset convolution.

Theorem 15. COVER POLYNOMIAL, $\#c$ -SPANNING FORESTS, MCSH and MSTH can be solved in $\mathcal{O}^*(2^n)$ time and polynomial space.

Proof. Due to the considerations mentioned above, the discussed zeta transforms can be computed in polynomial time. Now use the Möbius inversion formula and notice that the discussed functions indeed decide the problems. \square

Conclusion

We studied applications where the zeta transform is computable in polynomial time. As mentioned in the introduction, our algorithms considerably improve on dynamic programming in practice: in addition to improving the space requirement, our algorithms will always be (significantly) faster when combined with techniques from [vRNvD, Bax96]. We want to mention that applying Möbius inversion to a problem is not straightforward: first one has to come up with a function $f(X)$ with the wanted properties. Although for some applications old definitions like the Dreyfus-Wagner recurrence already suffice, this is not always trivial.

To support finding more applications, it is also interesting whether more subset products with similar nice properties can be found. We also refer to [BHKK07].

Acknowledgements

The author wants to thank his advisor Pinar Heggernes and Daniel Lokshantov for their valuable support and insightful remarks on this paper.

References

- [Bax96] Eric T. Bax. Recurrence-Based Reductions for Inclusion and Exclusion Algorithms Applied to $\#P$ Problems. 1996. 2, 2, 3.2
- [BH06a] Andreas Björklund and Thore Husfeldt. Exact algorithms for exact satisfiability and number of perfect matchings. *Lecture Notes in Computer Science*, 4051:548–559, 2006. 1, 2
- [BH06b] Andreas Björklund and Thore Husfeldt. Inclusion–exclusion algorithms for counting set partitions. In *FOCS*, pages 575–582, 2006. 1
- [BHKK07] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In *STOC*, pages 67–74, 2007. 1, 1, 13, 3.1, 14, 3.2, 3.2
- [BHKK08a] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Computing the tutte polynomial in vertex-exponential time. In *FOCS*, pages 677–686, 2008. 1, 1, 3.2, 3.2
- [BHKK08b] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Trimmed moebius inversion and graphs of bounded degree. In *STACS*, pages 85–96, 2008. 2
- [BK06] Hans L. Bodlaender and Dieter Kratsch. An exact algorithm for graph coloring with polynomial memory. Technical Report UU-CS-2006-015, Department of Information and Computing Sciences, Utrecht University, 2006. 1
- [CG95] Fan R. K. Chung and Ronald L. Graham. On the cover polynomial of a digraph. *J. Comb. Theory, Ser. B*, 65(2):273–290, 1995. 1, 3.2
- [DW72] S.E. Dreyfus and R.A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972. 1, 3.1

- [FGK08] Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. Faster steiner tree computation in polynomial-space. In *ESA*, pages 430–441, 2008. 1, 1, 2.2
- [FRGS08] Henning Fernau, Daniel Raible, Serge Gaspers, and Alexey A. Stepanov. Exact exponential time algorithms for max internal spanning tree. *CoRR*, abs/0811.1875, 2008. 1, 1
- [GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994. 1
- [GSS08] Serge Gaspers, Saket Saurabh, and Alexey A. Stepanov. A moderately exponential time algorithm for full degree spanning tree. In *TAMC*, pages 479–489, 2008. 1, 1
- [HK62] Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962. 3, 3
- [JVW90] Francois Jaeger., Dirk L. Vertigan, and Dominic J. A. Welsh. On the computational complexity of the jones and tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society*, 108(01):35–53, 1990. 1
- [Kar82] Richard M. Karp. Dynamic programming meets the principle of inclusion and exclusion. *Oper. Res. Lett.*, 1:49–51, 1982. 1, 2.1, 3
- [KGK77] Samuel Kohn, Allan Gottlieb, and Meryle Kohn. A generating function approach to the traveling salesman problem. In *ACM '77: Proceedings of the 1977 annual conference*, pages 294–300, New York, NY, USA, 1977. ACM. 1
- [Koi06] Mikko Koivisto. An $\mathcal{O}^*(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion–exclusion. In *FOCS*, pages 583–590, 2006. 1
- [Ned08] Jesper Nederlof. Inclusion exclusion for hard problems. Master’s thesis, Utrecht University, August 2008. 2
- [vRNvD] Johan M. M. van Rooij, Jesper Nederlof, and Thomas C. van Dijk. Inclusion/exclusion meets measure and conquer: Exact algorithms for counting dominating sets. Technical Report UU-CS-2008-043, Utrecht, The Netherlands, 2008. 2, 3.2
- [Woe04] Gerhard J. Woeginger. Space and time complexity of exact algorithms: Some open problems (invited talk). In *IWPEC*, pages 281–290, 2004. 1