Parameterized counting of trees, forests and matroid bases

Cornelius Brand, Marc Roth

Saarland University and Cluster of Excellence (MMCI)
{cbrand,mroth}@mmci.uni-saarland.de

Abstract. We prove #W[1]-hardness of (1) counting trees with k edges in a given graph (2) the number of forests with k edges in a given graph and (3) counting bases of a given matroid of rank (or nullity) k representable over an arbitrary field of characteristic two, where k is the parameter.

1 Introduction

Parameterized counting complexity has produced results on the hardness of counting the number of paths, cliques and cycles with k edges in a given graph. One important step was the proof of #W[1]-hardness for computing the number of k-matchings in a simple graph [3]. This line of research culminated in a classification theorem of Curticapean and Marx [4] for the following problem: Given a graph H from a class of graphs $\mathcal{H}$ and an arbitrary graph G, compute the number of all subgraphs of G that are isomorphic to H, parameterized by $|V(H)|$. They proved that this problem is fixed-parameter tractable if the vertex cover number of all graphs in $\mathcal{H}$ is bounded by a constant, and #W[1]-hard otherwise.

This theorem does not cover the problem of counting all occurrences of *all* subgraphs of a certain size that are contained in a fixed class $\mathcal{H}$ of graphs. This leaves open the cases we consider, where $\mathcal{H}$ is the class of forests or trees.

Another problem that has yet escaped a parameterized analysis is the problem of counting bases in matroids. Matroids have been studied over decades and play a central role in numerous combinatorial applications (see e.g. [13]). Although they were treated in the parameterized world (see e.g. [5, Chap. 12] for an overview), the problem of computing the number of bases was only addressed from the classical point of view so far [15,14,12]. It should be noted that this problem comprises also a generalization of counting forests in a graph, which gives the connection to the previously mentioned problems. Building on our results on counting forests, this gap in knowledge is one we address in the subsequent sections.

1.1 Related work

It is known that computing the number of all (labeled) trees and computing the number of all forests are #P-hard problems, even on planar graphs [9,16,8]. A

general theorem of Eppstein implies their being fixed-parameter tractable on planar graphs [6].

The problem of counting k-independent sets in a binary matroid is #P-hard. This follows from the well-known fact that the k-forests of a graph G correspond one-to-one to the k-independent sets of the binary matroid represented by the incidence matrix of G over $\mathbb{F}_2$ (see e.g. [14]). Also, counting the bases of a binary matroid is #P-hard (see e.g. [15]). On the other hand, the number of bases of a regular matroid can be computed in polynomial time [12].

2 Preliminaries

The integers are denoted by $\mathbf{Z}$. The polynomial ring over some ring R in the variables $x_1, \ldots, x_n$ is written $R[x_1, \ldots, x_n]$. The set of matrices with m rows, n columns and entries from a set X is $\mathrm{Mat}(n \times m, X)$.

2.1 Parameterized counting complexity

We begin with basic definitions of parameterized counting complexity, following closely Chapt. 14 of the textbook [7], which we recommend to the interested reader for a more comprehensive overview of the topic. Our fundamental object of study is the following. A *parameterized counting problem* (F, k) consists of a function $F : \{0,1\}^* \to \mathbb{N}$ and a polynomial-time computable function $k : \{0,1\}^* \to \mathbb{N}$, called the *parameterization*.

A parameterized counting problem (F, k) is called *fixed-parameter tractable* if there is an algorithm A for computing F, a constant $c > 0$ and a computable function $f : \mathbb{N} \to \mathbb{N}$ such that A is running in time $f(k(x)) \cdot |x|^c$ for all $x \in \{0,1\}^*$. We say that such an algorithm runs in *fpt-time*. Let (F, k) and (F', k') be two parameterized counting problems. Then, a function $R : \{0,1\}^* \to \{0,1\}^*$ is called an *fpt parsimonious reduction from* (F, k) *to* (F', k') if

1. For all $x \in \{0,1\}^*$, $F(x) = F'(R(x))$.
2. R runs in fpt-time.
3. There is some computable $g : \mathbb{N} \to \mathbb{N}$ such that $k'(R(x)) \leq g(k(x))$ for all $x \in \{0,1\}^*$.

An algorithm A with oracle access to F' is called an *fpt Turing reduction from* (F, k) *to* (F', k') if

1. A computes F.
2. A runs in fpt-time.
3. There is some computable $g : \mathbb{N} \to \mathbb{N}$ such that for all $x \in \{0,1\}^*$ and for all instances y for which the oracle is queried during the execution of $A(x)$, $k'(y) \leq g(k(x))$.

The parameterized counting problem #k-Clique is defined as follows, and is parameterized by k: Given a graph G and an integer k, compute the number of

cliques of size k in G. The class $\#\mathsf{W}[1]$ is defined as the set of all parameterized counting problems (F, k) such that there is an fpt parsimonious reduction from (F, k) to $\#k$-Clique. A parameterized counting problem (F, k) is called $\#\mathsf{W}[1]$-*hard* if there is an fpt Turing reduction from $\#k$-Clique to (F, k).

2.2 Matroids

A *matroid* is a pair $M = (E, \mathcal{I})$ consisting of a finite ground set E and a family $\mathcal{I} \neq \emptyset$ of subsets of E that satisfies the following axioms:

1. $\mathcal{I}$ is downward closed, i.e. if $I \in \mathcal{I}$ and $I' \subset I$, then $I' \in \mathcal{I}$.
2. $\mathcal{I}$ has the exchange property, i.e. if $I_1, I_2 \in \mathcal{I}$ and $|I_1| < |I_2|$, then there is some $e \in I_2 - I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Note that this entails $\emptyset \in \mathcal{I}$. The elements of $\mathcal{I}$ are called *independent sets*, and an inclusion-wise maximal element of $\mathcal{I}$ is called a *basis* of M. The exchange property warrants that all bases have the same cardinality, and we call this cardinality the *rank* of M, written as $\operatorname{rk} M$. Furthermore we define $(|E| - \operatorname{rk} M)$ as the *nullity* of M. The pair $M_k = (E, \mathcal{I}_k)$ with $\mathcal{I}_k = \{I \in \mathcal{I} \mid |I| \leq k\}$ is again a matroid, called the *k-truncation* M_k of M.

For a field F, a *representation of M over F* is a mapping $\rho : E \to V$, where V is a vector space over F, such that for all $A \subseteq E$, A is independent if and only if $\rho(A)$ is linearly independent. M is called *representable* if it is representable over some field. If there is such a representation, we call M *representable over F*, or *F-linear*, and it holds that $\operatorname{rk}(\rho) = \operatorname{rk}(M)$. Conversely, every matrix over a field F induces an F-linear matroid on its columns, where sets of columns are independent if and only if they are linearly independent. A *k-truncation* of a matrix is the matrix of a representation of the k-truncation of the corresponding linear matroid, possibly over a different field. Recently, Lokshtanov et al. proved that a k-truncation of a matrix can be computed in deterministic polynomial time (see [10], Theorem 3.23).

In the following we will write $\mathbb{F}_q$ for the field with q elements.

Given a matroid $M = (E, \mathcal{I})$, the *dual matroid* M^* of M is a matroid on the same ground set as M, and $B \subseteq E$ is a basis of M^* if and only if $E \setminus B$ is a basis of M. Given a representation of a matroid M, a representation of M^* in the same field can be found in polynomial time (see e.g. [11]).

In the following, *all* matroids will be assumed to be representable, and encoded using a representing matrix ρ and a suitable encoding for the ground field. Furthermore, we can, without loss of generality, always assume that ρ has $\operatorname{rk}(M)$ rows, because row operations (multiplying a row by a non-zero scalar, and adding such multiples to other rows) do not affect linear independence of the columns of ρ. Hence, any ρ' obtained from ρ through row operations is a representation of M. In particular, by Gaussian elimination, we may assume all but the first $\operatorname{rk}(M)$ rows of ρ to be zero.

2.3 Graphs and matrices

We consider simple graphs without self-loops unless stated otherwise. Given a graph G we will write n for the number of vertices of G and m for the number of edges. A *k-forest* is an acyclic graph consisting of k edges and a *k-tree* is a connected k-forest. We say that two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there is a bijection $\varphi : V_1 \to V_2$ such that for all $u, v \in V_1$, $\{u, v\} \in E_1$ if and only if $\{\varphi(u), \varphi(v)\} \in E_2$. A *$k$-matching* of a graph $G = (V, E)$ is a subset of k edges such that no pair of edges has a common vertex. The following was established by Curticapean.

Theorem 1 ([3]). *Given a graph G and a parameter k, it is $\#\mathsf{W}[1]$-hard to compute the number of matchings of size k in G.*

Given a graph G with vertices $v_1, \ldots, v_n$ and edges $e_1, \ldots, e_m$ we define the (unoriented) *incidence matrix* $M[G] \in \mathrm{Mat}(n \times m, \mathbb{F}_2)$ of G by $M[G](i, j) := 1$ if $v_i \in e_j$ and 0 otherwise. A subset of columns of $M[G]$ is linearly independent (over $\mathbb{F}_2$) if and only if the corresponding edges form a k-forest in G.

3 Counting trees

Definition 1. *Given a graph G and a natural number k, we denote the problem of counting all acyclic, connected subsets of edges of G of size k as $\#k$-*Trees*, parameterized by k.*

In this section we will prove the hardness of counting trees. For the proof, we show hardness of an intermediate problem.

Definition 2 (Weighted k-trees). *Given a graph $G = (V, E)$ with edge weights $\{w_e\}_{e \in E}$ and $k \in \mathbb{N}$, $\#k$-*WTrees *is defined as the problem of computing*

$$\mathrm{WT}_k(G) := \sum_{t \in T_k(G)} \prod_{e \in t} w_e \,,$$

where $T_k(G)$ is the set of all acyclic, connected edge sets of size k in G.

Note that this polynomial bears some similarity with the multivariate forest polynomial, to be defined in the next section. In fact, both polynomials have in common that they can be viewed as the multivariate generating functions of the respective structures in a graph.

Borrowing an idea from [1] in the context of counting forests, we first consider the above polynomial on a modified graph, namely after adding an apex, and show that this reveals information on the number of k-matchings in the original graph. More precisely, edges incident to the apex will be assigned a weight z which leads to the following intermediate problem.

Definition 3. *Let k be a natural number, $G = (V, E)$ be a graph with an apex $a \in V$, that is, a vertex that is adjacent to every other vertex, and edge weights $\{w_e\}_{e \in E}$ such that $w_e = 1$ for all edges e that are not adjacent to a and $w_e = z$ for a fixed $z \leq k$ otherwise. Then we denote the problem of computing $\mathrm{WT}_k(G)$ parameterized by k as $\#k$-*WApexTrees.

Lemma 1. *#k-*WApexTrees *is* #W[1]*-hard.*

Proof. First, we will outline the proof: The problem we are reducing from is the problem of counting k-matchings. That is, given a graph G we want to compute the number of k matchings by using an oracle for #k-WApexTrees. To do so, we are going to add x isolated vertices to G and after that, add an apex a (that is adjacent to every vertex in G and to every isolated vertex). We call the resulting graph G_x. The first step in the reduction is to count the number of trees containing $2k$ edges such that exactly k edges are incident to a. This number can be computed by assigning weight z to every edge incident to a (yielding a graph we call $G_{x,z}$), computing the value of WT_{2k}, which is a polynomial in z, for different values of z and then interpolating the coefficient of z^k. Note that we can compute this number *for different values of* x. After that we show that those numbers induce a system of linear equations with a unique solution. Finally we prove that one entry of the solution vector is the number of $2k$-trees in G_0 such that (1) exactly k edges are incident to a and (2) for every of those edges $\{v_1, a\}, \ldots, \{v_k, a\}$ there exist $u_1, \ldots, u_k$ such that $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$ are also contained in the tree. As trees do not contain cycles, we have that the u_i are pairwise different which implies that trees satisfying (1) and (2) correspond (up to a factor of 2^k) to the k-matchings in G.

As stated before, we reduce from the problem of counting k-matchings. Let $G = (V, E)$ be a graph with $n = |V|$. For x and z we construct the graph $G_{x,z}$ as follows:

- Add x isolated vertices to G.
- Add a vertex a to G and connect a to all other vertices, including the isolated ones (i.e., a is an apex).
- Assign weights to the edges as follows: If $a \notin e$ then set $w_e = 1$. Otherwise set $w_e = z$.

The unweighted version of $G_{x,z}$ is just denoted by G_x. Furthermore we denote the set of edges adjacent to a as E_x^a. Now fix x and compute for all $i \in \{0, \ldots, 2k\}$ the value $P_i(x) = \mathrm{WT}_{2k}(G_{x,i})$ with an oracle for #k-WApexTrees. This corresponds to evaluating the following polynomial in points $0, \ldots, 2k$:

$$Q_x(z) = \sum_{t \in T_{2k}(G_x)} \prod_{e \in t} w_e = \sum_{t \in T_{2k}(G_x)} \prod_{\substack{e \in t \\ a \notin e}} 1 \cdot \prod_{\substack{e \in t \\ a \in e}} z = \sum_{t \in T_{2k}(G_x)} z^{|E_x^a \cap t|}$$

Therefore, we can interpolate all coefficients. In particular, we are interested in the coefficient of z^k, which is the number of trees of size $2k$ in G_x, such that exactly k edges of the tree are adjacent to the apex a. We call these trees *apex-fair*, and denote their set as F_x. Now, consider an edge $e = \{v, a\}$ of such an apex-fair tree t that is adjacent to a. We call e *apex-isolated* if v is not incident to any other edge of t. Otherwise we call e *apex-connected*, and we call the subtree t^c of t without apex-isolated edges *apex-connected* as well.[1] Note that t^c is indeed a

[1] This terminology stems from the fact that the removal of a leaves v isolated

tree, not only a forest. Furthermore, t^c is a tree in G_0 as edges that are connected to isolated vertices cannot be apex-connected.

We observe that for an apex-fair tree t in G_x, the subtree t^c in G_0 has exactly $s-k$ apex-connected edges, where s is the number of edges in t^c: Since t is apex-fair, we know that exactly k of the edges in t are not incident to a, and thus by definition, such an edge cannot be apex-isolated. Hence, these k edges are also in t^c, meaning that the remaining $s-k$ edges in t^c have to be incident to the apex, and indeed, they have to be apex-connected, since all apex-isolated edges were removed from t when constructing t^c.

We can partition the set F_x of apex-fair trees in G_x by the number s of edges that the corresponding apex-connected tree contains, say

$$F_x = \bigcup_{s=0}^{2k} B_x(s)$$

where $B_x(s)$ is the set of apex-fair trees t in G_x such that t^c is of size s. Clearly, this union is disjoint, and $B_x(i) = \emptyset$ for $0 \leq i \leq k$, since we consider apex-fair trees, meaning that $|F_x| = \sum_{s=k}^{2k} |B_x(s)|$. Now, consider a single apex-fair tree $t \in B_x(s)$ for some fixed s, which is hence of size $2k$. Then, as argued, t^c has exactly $s-k$ apex-connected edges, and k edges not incident to the apex. This leaves $2k-s$ edges of t to be chosen, and since there are already k edges present in t^c that are not incident to the apex, and t is apex-fair, all these $2k-s$ edges have be incident to the apex, and also have to be apex-isolated, since otherwise they would be in t^c. By the construction of G_x, there are $n+x$ possible choices for the apex-isolated edges in total, but s of these are already occupied by t^c, leaving $n+x-s$ choices for the $2k-s$ remaining edges.Hence, for every apex-connected tree t^c, there are exactly $\binom{n+x-s}{2k-s}$ fair trees t' such that $t'^c = t^c$. Letting β_s be the number of apex-connected trees of size s, this amounts to

$$|B_x(s)| = \beta_s \cdot \binom{n+x-s}{2k-s}$$

and thus

$$|F_x| = \sum_{s=k+1}^{2k} \beta_s \cdot \binom{n+x-s}{2k-s}.$$

For convenience, we perform an index shift on β, and find

$$|F_x| = \sum_{s=1}^{k} \beta_{s+k} \cdot \binom{n+x-(k+s)}{k-s}.$$

We can then evaluate $|F_x|$ for $x \in \{1, \ldots, k\}$ and solve for the β_s. To do so, we show that the corresponding matrix $A \in \mathbb{N}^{k \times k}$, defined by

$$A_{i,j} = \binom{i+n-k-j}{k-j}$$

has full rank.

Its j-th column is an evaluation vector of the polynomial

$$R_j(x) = \binom{x+n-k-j}{k-j}.$$

The degrees of the polynomials R_j are pairwise distinct and hence, the polynomials and the evaluation vectors are as well. It follows that A has full rank, i.e., we can indeed compute the coefficients β_s.

Finally, consider β_{2k}: This is the number of apex-connected trees in G_0 with k edges in G_0 and k apex-connected edges. It follows that these trees correspond to k-matchings in G, more precisely, for every k-matching in G, there are 2^k such apex-connected trees. This concludes the proof. □

In the proof of Lemma 1 we exploited the fact that the weights of edges incident to the apex can be used to enforce some desired structure via interpolation. One might wonder why we explicitly enforced the edges with weight $\neq 1$ to be incident to the apex and the weights of those edges to be equal in the definition of #k-WApexTrees, instead of just defining a canonical edge-weighted variant of the problem of counting trees. The reason for this is the fact that we need this very structure in order to get rid of the weights and reduce to #k-Trees, at least in our reduction. Before we do this reduction, we prove another lemma.

Lemma 2. *Let $G = (V, E)$ be a simple graph and $A \subseteq E$ a subset of edges of size z. Then, the problem of counting the number of $(k+z)$ trees whose edges contain A can be solved in time $O(2^z) \cdot \text{poly}(|V|)$ if access to an oracle for #k-Trees is provided.*

Proof. Let S be a subset of edges of G. We define T_S as the set of all $(k+z)$ trees in G that do not contain any edge in S. Note that we can compute $|T_S|$ by deleting all edges in S and counting the number of $(k+z)$ trees in G by posing an oracle query. Furthermore, it holds that

$$T_{S_1} \cap T_{S_2} = T_{S_1 \cup S_2} \tag{1}$$

for any two subsets of edges S_1 and S_2. Let T be the set of all $(k+z)$ trees in G, i.e., $T := T_\emptyset$. Now we can express the number of $(k+z)$ trees whose edges contain A as

$$|T \setminus \bigcup_{e \in A} T_{\{e\}}|.$$

Using the inclusion-exclusion principle, we get

$$|T \setminus \bigcup_{e \in A} T_{\{e\}}| = |T| - \sum_{\emptyset \neq J \subseteq A} (-1)^{|J|-1} \, |\bigcap_{e \in J} T_{\{e\}}| = |T| - \sum_{\emptyset \neq J \subseteq A} (-1)^{|J|-1} \, |T_J|$$

where the second equality follows from (2). Finally, we observe that there are exactly 2^z summands, each of which can be computed by posing an oracle query for the corresponding T_J. □

This allows is to make the final step in the proof of Theorem 2:

Lemma 3. *#k-*WApexTrees *is fpt Turing reducible to #k-*Trees*.*

Proof. Let k be a natural number and $G = (V \cup \{a\}, E \cup V \times \{a\})$ be a graph with apex a and edge weights as in Definition 3. The goal is to compute

$$\mathrm{WT}_k(G) = \sum_{t \in T_k(G)} \prod_{e \in t} w_e \,,$$

where $w_e = z$ if $a \in e$ and $w_e = 1$ otherwise. First, we point out that $T_k(G)$ can be partitioned into trees (with k edges) that do not contain a and trees that do contain a. We denote the set of the latter as $T_k^a(G)$. Then we have that

$$\begin{aligned}\mathrm{WT}_k(G) = \sum_{t \in T_k(G)} \prod_{e \in t} w_e &= \sum_{t \in T_k(G-\{a\})} \prod_{e \in t} w_e + \sum_{t \in T_k^a(G)} \prod_{e \in t} w_e \\ &= |T_k(G - \{a\})| + \sum_{t \in T_k^a(G)} \prod_{e \in t} w_e \,.\end{aligned}$$

Now $|T_k(G - \{a\})|$ can be computed by querying the oracle. If $z = 0$ then $\sum_{t \in T_k^a(G)} \prod_{e \in t} w_e = 0$ as well, that is, we are done. Otherwise we need to realise the edges with weight z to compute $\sum_{t \in T_k^a(G)} \prod_{e \in t} w_e$. To do so, we construct the graph $G^z = (V^z, E^z)$ from G as follows:

- Delete the apex and the adjacent edges.
- Add apices $a_1, \ldots, a_z$ (including edges to all vertices in G).
- Add a vertex a and edges $\{a, a_i\}$ for all $i \in [z]$.

We first sketch the idea of the proof: Let $\hat{T}$ be the set of all trees with $k + z$ edges in G^z *such that all edges* $\{a, a_1\}, \ldots, \{a, a_z\}$ are met and let $t \in \hat{T}$. As t is connected, there is at least one vertex v in $G - \{a\}$ such that $\{v, a_i\}$ is contained in t for an $i \in [z]$. Furthermore, for every vertex v in $G - \{a\}$ *at most one* of the edges $\{v, a_1\}, \ldots, \{v, a_z\}$ can be contained in t, because otherwise t would have a cycle. This means that taking one of the edges $\{v, a_1\}, \ldots, \{v, a_z\}$ in G^z corresponds to taking edge $\{v, a\}$ of weight z in G. We will prove that

$$|\hat{T}| = \sum_{t \in T_k^a(G)} \prod_{e \in t} w_e \,.$$

Finally we will use Lemma 2, that is, an application of the inclusion-exclusion principle, to compute $|\hat{T}|$.
Now let $t \in \hat{T}$. As stated above we claim that for every vertex v in $G - \{a\}$ *at most one* of the edges $\{v, a_1\}, \ldots, \{v, a_z\}$ can be contained in t. Assuming not, there is a vertex v in $G - \{a\}$ and indices $i \neq j$ such that $\{v, a_i\}$ and $\{v, a_j\}$ are contained in t, but this induces the cycle (a, a_i, v, a_j, a) which contradicts the fact that t is a tree. Now we define a mapping $f : \hat{T} \to T_k^a(G)$ as follows: For every edge $e = \{u, v\}$ in $G - \{a\}$, e is contained in $f(t)$ if and only if e is

contained in t and for every edge $e_a = \{v, a\}$, e is contained in $f(t)$ if and only if there is an $i \in [z]$ such that $\{v, a_i\}$ is contained in t. Note that $f(t)$ contains a as there is at least one vertex v in $G - \{a\}$ such that $\{v, a_i\}$ is contained in t for an $i \in [z]$. For a tree $t_G \in T_k^a(G)$ we define $g(t_G) := \{t \in \hat{T} \mid f(t) = t_G\}$. Now let $T_k^a(G)[\ell]$ be the set of all trees $t_G \in T_k^a(G)$ such that there are exactly ℓ vertices v in $G - \{a\}$ such that $\{v, a\}$ is contained in t_G. For every such vertex, there are z possibilities to construct a tree $t \in \hat{T}$ such that $f(t) = t_G$ (by taking one of the edges $\{v, a_1\}, \ldots, \{v, a_z\}$). Hence $|g(t_G)| = z^\ell$ if $t_G \in T_k^a(G)[\ell]$. Furthermore we claim that

$$\hat{T} = \dot{\bigcup}_{t_G \in T_k^a(G)} g(t_G)\,.$$

This follows from the observation that the sets $g(t_G)$ are the classes of the equivalence relation $t \sim t' \Leftrightarrow f(t) = f(t')$. Putting everything together we have

$$\begin{aligned}\sum_{t \in T_k^a(G)} \prod_{e \in t} w_e &= \sum_{\ell=1}^{k} \sum_{t \in T_k^a(G)[\ell]} \prod_{e \in t} w_e = \sum_{\ell=1}^{k} \sum_{t \in T_k^a(G)[\ell]} z^\ell \\ &= \sum_{\ell=1}^{k} \sum_{t \in T_k^a(G)[\ell]} |g(t)| = \sum_{t \in T_k^a(G)} |g(t)| = \left|\dot{\bigcup}_{t_G \in T_k^a(G)} g(t_G)\right| = |\hat{T}|\end{aligned}$$

It remains to show how to compute $|\hat{T}|$ with an oracle for #k-Trees. This can be done by applying Lemma 2 with $A = \{\{a, a_1\}, \ldots, \{a, a_z\}\}$. □

We can thus state:

Theorem 2. *#k-Trees is #W[1]-hard when parameterized by k.*

Proof. In Lemma 1 it was shown that #k-WApexTrees is #W[1]-hard and in Lemma 3 we proved that #k-WApexTrees is fpt Turing reducible to #k-Trees. It follows that #k-Trees is #W[1]-hard as well.

4 Counting forests

In this section, we will prove that counting k-forests is #W[1]-hard. This result will be used to show hardness of counting matroid bases in fields of fixed characteristic.

Definition 4. *Let $G = (V, E)$ be a multigraph with edges labeled with formal variables $\{w_e\}_{e \in E}$. Then the* multivariate forest polynomial *of G is defined as*

$$F(G; \{w_e\}_{e \in E}) = \sum_{A \subseteq E \text{ acyclic}} \prod_{e \in A} w_e\,.$$

The polynomial obtained by replacing all weights w_e with a fresh variable x, i.e., setting $w_e = x$ for all $e \in E$, is called the univariate forest polynomial *of the graph and is simply denoted $F(G; x)$.*

For a forest A in G, let $c(A)$ be the family of all sets $T \subseteq V(G)$ such that $T \neq \emptyset$ and T is a maximal connected component in A. Adding an apex, that is, a new vertex that is connected to all other vertices, to a graph $G = (V, E)$ and labeling each of the new edges with a new variable z makes the univariate forest polynomial into a bivariate one, namely $F(G'; x, z)$, where G' is the described graph with an added apex. In the following, G will always be the original graph, and G' will be the graph obtained in this way. Note that $F(G'; x, z) \in \mathbf{Z}[x, z] \cong (\mathbf{Z}[z])[x]$. In particular, the coefficient of x^k in $F(G'; x, z)$ is an element of $\mathbf{Z}[z]$. To make this very clear in the following, we shall refer to this element of $\mathbf{Z}[z]$ as the *coefficient polynomial* of x^k in $F(G'; x, z)$.

Lemma 4. *There is a polynomial-time Turing reduction from counting k-matchings in a graph G to computing the coefficient polynomial of x^k of the bivariate forest polynomial $F(G'; x, z)$ of the graph G', parameterized by k in both problems. In particular, this reduction retains the parameter k and is thus even an fpt Turing reduction.*

Proof. The coefficient polynomial $C_k(z) \in \mathbf{Z}[z]$ of x^k in $F(G'; x, z)$ can be expressed in terms of G through

$$C_k(z) = \sum_{A \in \binom{E}{k} \text{ acyclic in } G} \prod_{T \in c(A)} (1 + |T|z),$$

as follows immediately from Lemma 7 in [1] after specializing to x and z. Since a forest in G with k edges can cover at most $2k$ nodes of G, at least $n - 2k$ nodes of G are left uncovered by T, and are thus present in $c(A)$ as components T with $|T| = 1$. This shows that the product $\prod_{T \in c(A)}(1 + |T|z)$ of each summand (and hence also $C_k(z)$) is a multiple of $(1 + z)^{n-2k}$. Thus, the polynomial quotient

$$Q_k(z) := C_k(z)/(1 + z)^{n-2k}$$

is a well-defined element of $\mathbf{Z}[z]$.

Observe that it is precisely the k-matchings of G that will have $2k$ covered and $n - 2k$ uncovered nodes, and such a k-matching has k components with $|T| = 2$, and $n - 2k$ components with $|T| = 1$. Therefore, the summand corresponding to some A in $C_k(z)$ is of the form $(1 + z)^{n-2k}(1 + 2z)^k$ if and only if A is a k-matching. Likewise, the number of k-matchings is precisely the number of such monomials in $C_k(z)$. In all other monomials, the factor $(1 + z)$ is hence present with degree at least $n - 2k + 1$. Denote with M_k the number of k-matchings in G. After substituting $z \mapsto y - 1$ (hence, $z = y + 1$), this can be stated as follows:

$$Q_k(y) = C_k(y)/y^{n-2k} = M_k \cdot (2y - 1)^k + y \cdot R(y)$$

for some polynomial $R(y)$. We see that for $y = 0$, $y \cdot R(y) = 0$, and hence, keeping in mind that $z = y + 1$, it follows

$$Q_k(y = 0) = Q_k(z = -1) = M_k \cdot (-1)^k .$$

We now argue why this is an fpt Turing reduction. Note that in the coordinates y^i, the polynomial division is just a shift of coefficients. Therefore, an oracle to the k-th coefficient of $F(G'; x, z)$, as provided in a Turing reduction, yields the polynomial $C_k(z)$. After a change of basis from z to $y - 1$ and a corresponding shift of coefficients to perform the division by y^{n-2k}, we can evaluate the resulting polynomial $Q_k(y)$ at $y = 0$ and obtain $M_k \cdot (-1)^k$ and thus M_k. This can clearly be done in polynomial time in the size of G (and k, for that matter) once $C_k(z)$ was obtained, and the only oracle query involved does not alter the parameter and is hence valid for an fpt Turing reduction. □

This proves that the coefficient polynomial of x^k in the bivariate polynomial $F(G'; x, z)$ is hard to compute. We now want to show that this implies that the k-th coefficient (which is a natural number, not a polynomial) of the univariate polynomial is hard to compute. We do this by reducing the compution the coefficient polynomial of x^k in $F(G'; x, z)$ to computing the k-th coefficient in a suitable univariate forest polynomial.

We first show that, although the degree of the coefficient polynomial $C_k(z)$ (in the bivariate case) is not bounded by $f(k)$, but $\Omega(n)$, it suffices to know $O(k)$ coefficients of the coefficient polynomial $C_k(z)$ in order to reconstruct the whole coefficient polynomial. This is essentially an application of the Chinese Remainder Theorem for polynomials, and is given in detail in the full version [2].

Lemma 5. *There is an fpt Turing reduction from computing the coefficient polynomial $C_k(z)$ of x^k in $F(G'; x, z)$ to computing the first k coefficients of univariate forest polynomials on multigraphs.*

Combining the above proves:

Theorem 3. *Given a graph G and a number k, it is $\#\mathsf{W}[1]$-hard to compute the number of acyclic subsets of edges of size k in G, parameterized by k.*

Proof. Combining Theorem 1 with Lemma 4 and Lemma 5 yields that computing the first k coefficients of the univariate forest polynomial of multigraphs is $\#\mathsf{W}[1]$-hard. Using Lemma 9 from [1] allows to express the forest polynomial of the multigraph graph G as $F(G; x) = p(x) \cdot F(G'; g(x))$, where G' is a simple graph, and p, g are functions such that g is invertible. Now, observe that computing the mapping $G \mapsto F_k(G; x)$ is $\#\mathsf{W}[1]$-hard for each fixed x, where $F_k(G; x)$ is the forest polynomial $F(G; x)$ evaluated at x only over the first k coefficients: It is easy to see that $F_k(G; ax) = F_k(G(a); x)$, where G is the graph obtained from G by replacing each edge with a copies of weight x. By using k different values for a, this would allow polynomial interpolation of the first k coefficients of $F(G; x)$. Employing now the equation $F_k(G; x) = p(x) \cdot F_k(G'; g(x))$ and the properties of g, this shows that computing the mapping $G' \mapsto F_k(G'; x)$ on *simple graphs* G' is $\#\mathsf{W}[1]$-hard for all x, and hence also for $x = 1$, where it coincides with counting forests.

□

5 Counting bases in matroids

Definition 5. *The problem of counting the number of bases of a matroid parameterized by its rank (nullity) is denoted as* #Rank-Bases *(*#Nullity-Bases*).*

Lemma 6. *The problem of counting k-forests in a simple graph is fpt Turing reducible to the problem* #Rank-Bases*, even when the matroid is restricted to be representable over a field of characteristic* 2*.*

Proof. Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ and a natural number k, we want to count the k-forests of G. Therefore we first construct the incidence matrix $M[G] \in \mathrm{Mat}(n \times m, \mathbb{F}_2)$ of G. Recall that the linearly independent k-subsets of columns of $M[G]$ correspond one-to-one to k-forests in G. In the next step, we compute the reduced row echelon form of $M[G]$ by applying elementary row operations. As stated in the beginning, these operations do not change the linear dependency of the column vectors. Then, we delete the zero rows which also does not change the linear dependency of the columns. We denote the resulting matrix as $M^{\mathrm{red}}[G]$. Now, let r be the rank of $M^{\mathrm{red}}[G]$, which equals the rank of $M[G]$. Note that $M^{\mathrm{red}}[G] \in \mathrm{Mat}(r \times m, \mathbb{F}_2)$. If $r < k$, then we output 0, as G does not have any k-forests in this case. Otherwise, we k-truncate $M^{\mathrm{red}}[G]$ in polynomial time by the deterministic algorithm of Lokshtanov et al. [10] and end up with the matrix $M^{(k)}[G] \in \mathrm{Mat}(k \times m, \mathbb{F}_{2^{rk}})$. Observe that the linear dependency of the column vectors is preserved, i.e., whenever columns $c_1, \ldots, c_k$ are linearly independent in $M[G]$, they are also linearly independent in $M^{(k)}[G]$ and vice versa. Therefore, the rank of $M^{(k)}[G]$ is at least k, since $M[G]$ has rank greater or equal k. As $M^{(k)}[G]$ has only k rows, it follows that the rank is *exactly* k, i.e., $M^{(k)}[G]$ has full rank. Furthermore, the number of linearly independent k-subsets of columns of $M^{(k)}[G]$ equals the number of k-forests in G. As the rank of $M^{(k)}[G]$ is full, we conclude that the number of bases of the matroid that is represented by $M^{(k)}[G]$ equals the number of k-forests in G. Finally, this matroid is representable over $\mathbb{F}_{2^{rk}}$—a field of characteristic 2—by construction. □

Lemma 7. *The problem of counting k-forests in a simple graph is fpt Turing-reducible to the problem* #Nullity-Bases*, even when the matroid is restricted to be representable over a field of characteristic* 2*.*

Proof. We proceed as in the proof of Lemma 6. Having $M^{(k)}[G]$, we construct its dual matroid $M^*[G]$, which can be done in polynomial time (see e.g. [11]). It holds that the number of bases of $M^*[G]$ equals the number of bases of $M^{(k)}[G]$. Furthermore, the rank of $M^*[G]$ is $n - k$, i.e., its nullity is k, which concludes the proof. □

Theorem 4. #Rank-Bases *and* #Nullity-Bases *are* #W[1]*-hard, even when restricted to matroids representable over a field of characteristic* 2*.*

Proof. Follows from Lemma 6, Lemma 7 and Theorem 3. □

One might ask whether the same is true for matroids that are representable over finite fields. Due to Vertigan [15], it is known that the classical problem of counting bases in binary matroids is #P-hard. However, unless #W[1] coincides with the class of fixed-parameter tractable problems, this does not hold for the parameterized versions:

Theorem 5. *For every fixed finite field $\mathbb{F}$, the problems* #Rank-Bases *and* #Nullity-Bases *are fixed parameter tractable for matroids given in a linear representation over $\mathbb{F}$.*

Proof (of Theorem 5). We give an fpt algorithm for #Rank-Bases.
For #Nullity-Bases, an algorithm follows by computing the dual matroid before.

Let s be the size of the finite field, M be the representation of the given matroid and let k be its rank. We can assume that M only has k rows. For otherwise, we can compute the reduced row echelon form and delete zero rows, which does not change the linear dependencies of the column vectors. If M has only k rows, then there are at most s^k different column vectors. Therefore, we remember the muliplicity of every column vector and delete multiple occurences afterwards. We end up with a matrix with at most s^k columns. Then, we can check for every k-subset of columns whether they are linearly independent. If this is the case, we just multiply the multiplicities of the columns and in the end, we output the sum of all those terms. The running time of this procedure is bounded by $\binom{s^k}{k} \cdot \mathrm{poly}(n)$, where n is the number of columns of the matrix. □

6 Open Problems

The progress we made in this work is incremental in nature: We added three new ones to the growing list of #W[1]-hard problems. Remarkably, their complexity was unknown despite their naturalness and the far-reaching classification results that are available. Of course, this leaves open the question of parameterized complexity for infinitely many other problems of similar flavor, namely, counting combinations of subgraphs with a certain property. It would be very much desirable to extend known dichotomy theorems to cover also these cases, so as to ease the arduous task of classifying an infinite number of problems by hand.

Acknowledgements

The authors wish to thank Holger Dell, Radu Curticapean and Markus Bläser for helpful comments on this work.

References

1. Cornelius Brand, Holger Dell, and Marc Roth. Fine-grained dichotomies for the Tutte plane and Boolean #CSP. *CoRR*, abs/1606.06581, 2016.

2. Cornelius Brand and Marc Roth. Parameterized counting of trees, forests and matroid bases. *CoRR*, abs/1611.01823, 2016.

3. Radu Curticapean. *Counting Matchings of Size k Is #W[1]-Hard*, pages 352–363. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

4. Radu Curticapean and Dániel Marx. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 130–139, 2014.

5. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

6. David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Graph Algorithms and Applications*, page 283, 2002.

7. J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

8. Heidi Gebauer and Yoshio Okamoto. Fast exponential-time algorithms for the forest counting and the Tutte polynomial computation in graph classes. *Int. J. Found. Comput. Sci.*, 20(1):25–44, 2009.

9. Mark Jerrum. Counting trees in a graph is #P-complete. *Inf. Process. Lett.*, 51(3):111–116, 1994.

10. Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, and Saket Saurabh. *Deterministic Truncation of Linear Matroids*, pages 922–934. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

11. Dániel Marx. A parameterized view on matroid optimization problems. *Theoretical Computer Science*, 410(44):4471–4479, 2009.

12. Stephen B. Maurer. Matrix generalizations of some theorems on trees, cycles and cocycles in graphs. *SIAM Journal on Applied Mathematics*, 30(1):143–148, 1976.

13. James G. Oxley. *Matroid Theory*. Oxford University Press, 1992.

14. Michael Snook. Counting bases of representable matroids. *Electr. J. Comb.*, 19(4):P41, 2012.

15. Dirk Vertigan. Bicycle dimension and special points of the Tutte polynomial. *Journal of Combinatorial Theory, Series B*, 74(2):378–396, 1998.

16. Dirk Vertigan and Dominic J. A. Welsh. The compunational complexity of the Tutte plane: the bipartite case. *Combinatorics, Probability & Computing*, 1:181–187, 1992.

Appendix

Proof (of Lemma 2). Let S be a subset of edges of G. We define T_S as the set of all $(k+z)$ trees in G that do not contain any edge in S. Note that we can compute $|T_S|$ by deleting all edges in S and counting the number of $(k+z)$ trees in G by posing an oracle query. Furthermore, it holds that

$$T_{S_1} \cap T_{S_2} = T_{S_1 \cup S_2} \tag{2}$$

for any two subsets of edges S_1 and S_2. Let T be the set of all $(k+z)$ trees in G, i.e., $T := T_\emptyset$. Now we can express the number of $(k+z)$ trees whose edges contain A as

$$|T \setminus \bigcup_{e \in A} T_{\{e\}}| \,.$$

Using the inclusion-exclusion principle, we get

$$|T \setminus \bigcup_{e \in A} T_{\{e\}}| = |T| - \sum_{\emptyset \neq J \subseteq A} (-1)^{|J|-1} \, |\bigcap_{e \in J} T_{\{e\}}| = |T| - \sum_{\emptyset \neq J \subseteq A} (-1)^{|J|-1} \, |T_J|$$

where the second equality follows from (2). Finally, we observe that there are exactly 2^z summands, each of which can be computed by posing an oracle query for the corresponding T_J. □

We will use the Chinese Remainder Theorem as stated in

Theorem 6 (Chinese Remainder Theorem). *Let $R = \mathbf{Q}[z]$, $m_0, \ldots, m_{r-1} \in R$ pairwise coprime, and $m = m_0 \cdots m_{r-1}$, $d_i = \deg m_i \geq 1$ for $0 \leq i < r, n = \deg m = \sum_{i=0}^{r-1} d_i$ and $v_i \in R$ with $\deg v_i < d_i$. Then the unique solution $f \in R$ with $\deg f < n$ of the system*

$$f \equiv v_i \bmod m_i \text{ for } 0 \leq i < r$$

can be computed using a polynomial number of bit operations in the d_i and the size of the coefficients of m.

Proof (of Lemma 5). Define for $A \subseteq E$

$$M_A(z) := \prod_{T \in c(A)} (1 + |T| z) \,.$$

Note that these are precisely the products appearing as summands in $C_k(z)$. This is the *univariate* forest polynomial of a multigraph S_A. Namely, we let S_A be a star-graph with $|c(A)| + 1$ vertices, and for each vertex v_T corresponding to some component $T \in c(A)$, S_A has $|T|$ edges labeled with z between the center and v_T. The forests in S_A are precisely those edge sets that select at most one edge between each node v_T and the center. For each $T \in c(A)$, we therefore have the choice between any of the $|T|$ edges to the corresponding vertex v_T, or we

can choose no edge to v_T at all. In analogy to the Binomial Theorem, selecting at most one of these $|T|$ edges then corresponds to the sum

$$1 + \underbrace{z + \ldots + z}_{|T|} = (1 + |T|z)\,,$$

and combining the selections for all vertices corresponds to the product.

This amounts to

$$M_A(z) = F(S_A; z)\,.$$

Since we are considering the coefficient of x^k, we know that A will have between $n-k$ and $n-2k$ single-vertex components. In other words, $(1+z)^{n-2k}$ divides $F(S_A; z)$, or algebraically speaking,

$$F(S_A; z) + (1+z)^{n-2k} \cdot \mathbf{Z}[z] = 0 + (1+z)^{n-2k} \cdot \mathbf{Z}[z]\,. \tag{3}$$

Additionally, suppose we know the first $k+1$ coefficients of $F(S_A; z)$, i.e. we know a polynomial g of degree at most k such that

$$F(S_A; z) + z^{k+1} \cdot \mathbf{Z}[z] = g + z^{k+1} \cdot \mathbf{Z}[z]\,. \tag{4}$$

Observing that $(1+z)^{n-2k}$ and z^{k+1} are coprime, the Chinese Remainder Theorem guarantees a unique solution of degree at most $n-2k+k = n-k$ of the system (3), (4). Since $c(A)$ has at most $n-k$ components, $F(S_A; z)$ is in fact of degree at most $n-k$, and can be uniquely recovered.

This argument readily extends to a sum of forest polynomials: If we know the sum of the first k coefficients of $\{F(S_A; z)\}_{A \in \mathcal{A}}$ for some collection of forests $\mathcal{A}$ of equal size, then we can reconstruct the sum of all coefficients of this set of polynomials. For, all these polynomials have the required divisibility property, and hence their sum has it as well.

We now turn to the issue of reconstructing the coefficients of $x^i z^j$ for $i+j \leq 2k$ when given access to the first $2k$ coefficients of the univariate forest polynomial. Letting $G'(a,b)$ be the graph obtained from G' by replacing those edges labeled x with a parallel edges, and by replacing those edges labeled z with b parallel edges, and labeling all these edges with x, we have

$$F(G'; ax, bx) = F(G'(a,b); x)\,.$$

We write c_{ij} for the coefficients of the monomial $x^i z^j$ in $F(G'; x, z)$, and $d(a,b)_i$ for the coefficient of x^i in $F(G'(a,b); x)$. For $t \leq 2k$, this equality implies on the monomial level that

$$x^t \cdot \left(\sum_{i+j=t} a^i b^j c_{ij} \right) = x^t \cdot d(a,b)_t\,.$$

In particular, if $d(a,b)_t$ is known for $t \leq 2k$, and we are looking to reconstruct c_{ij} with $i+j \leq 2k$, then this identity provides us with $2k+1$ linear equations in the

$\binom{2k}{2}$ variables c_{ij}. Using $O(k)$ suitably chosen values for the pair (a, b), we can solve the system for the c_{ij} and obtain the coefficients of $x^i z^j$ for $i + j \leq 2k$. In particular this implies that we can compute the coefficients of $x^k, x^k z, \ldots, x^k z^k$, which is nothing else than the first $k + 1$ coefficients of the sum of the forest polynomials $F(S_A; z)$ for $|A| = k$. As argued above, this enables us to compute the remaining coefficients of this sum, i.e. the coefficient (an element of $\mathbf{Z}[z]$) of x^k in $F(G'; x, z)$. □