A Fine-Grained Classification of the Complexity of Evaluating the Tutte Polynomial on Integer Points Parameterized by Treewidth and Cutwidth

Isja Mannens
Utrecht University, The Netherlands

Jesper Nederlof
Utrecht University, The Netherlands

Abstract

We give a fine-grained classification of evaluating the Tutte polynomial $T(G;x,y)$ on all integer points on graphs with small treewidth and cutwidth. Specifically, we show for any point $(x,y) \in \mathbb{Z}^2$ that either

- $T(G;x,y)$ can be computed in polynomial time,
- $T(G;x,y)$ can be computed in $2^{O(tw)}n^{O(1)}$ time, but not in $2^{o(ctw)}n^{O(1)}$ time assuming the Exponential Time Hypothesis (ETH),
- $T(G;x,y)$ can be computed in $2^{O(tw \log tw)}n^{O(1)}$ time, but not in $2^{o(ctw \log ctw)}n^{O(1)}$ time assuming the ETH,

where we assume tree decompositions of treewidth tw and cutwidth decompositions of cutwidth ctw are given as input along with the input graph on n vertices and point (x,y).

To obtain these results, we refine the existing reductions that were instrumental for the seminal dichotomy by Jaeger, Welsh and Vertigan [Math. Proc. Cambridge Philos. Soc'90]. One of our technical contributions is a new rank bound of a matrix that indicates whether the union of two forests is a forest itself, which we use to show that the number of forests of a graph can be counted in $2^{O(tw)}n^{O(1)}$ time.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Width Parameters, Parameterized Complexity, Tutte Polynomial

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.82

Related Version *Full Version*: https://arxiv.org/abs/2307.01046

Supplementary Material *Software (MATLAB Code)*: https://github.com/isja-m/ForestRank4-5 archived at swh:1:dir:2e6936582c19e5fd2f127b3d1e601ecb9a1136f1

Funding Both authors are supported by the project CRACKNP that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 853234).

1 Introduction

We study the parameterized complexity of computing the Tutte Polynomial. The Tutte polynomial is a graph invariant that generalizes any graph invariant that satisfies a linear deletion-contraction recursion. Such invariants include the chromatic, flow and Jones polynomials, as well as invariants that count structures such as the number of forests or the number of spanning subgraphs. Due to its generality the Tutte polynomial is of great interest to a variety of fields, including knot theory, statistical physics and combinatorics.

For a number of these fields it is important to understand how difficult it is to compute the Tutte polynomial. A series of papers, culminating in the work by Jaeger, Vertigan, and Welsh [15] has given a complete dichotomy showing that the problem of evaluating the Tutte polynmial is #P-hard on all points except on the following *special points* on which it is known to be computable in polynomial time:

$$(1,1),(-1,-1),(0,-1),(-1,0),(i,-i),(-i,i),(j,j^2),(j^2,j),\ H_1 \tag{1}$$

where $j = e^{2\pi i/3}$ and $i = \sqrt{-1}$, and H_α denotes the hyperbola $\{(x,y) : (x-1)(y-1) = \alpha\}$. These hyperbolic curves turn out to be of great importance to understanding the complexity of the Tutte Polynomial, as the problem is generally equally hard on all points of the same curve, except for the *special points* listed in (1).

Further refinements of the result by [15] have since been made: Among others, a more fine-grained examination of the complexity was done by Brand et al. [5] (building on earlier work by Dell et. al. [12]): they showed that for almost all points the Tutte polynomial cannot be evaluated in $2^{o(n)}$ time on n-vertex graphs, assuming (a weaker counting version of) the Exponential Time Hypothesis. This is tight because, on the positive side, Björklund et al. [2] showed that the Tutte polynomial can be evaluated on any point in $2^n n^{O(1)}$ time.

Another perspective worth examining is that of the parameterized complexity of the problem, when parameterized by *width measures*. This is a rapidly evolving field within parameterized complexity.[1] Intuitively, it is concerned with the effects of structural properties of the given input graph on its complexity. This often generates results that have greater practical value and give a deeper understanding of the problem, in comparison with classical worst-case analysis. It is therefore natural to ask what a complexity classification for the Tutte Polynomial would look like in this parameterized context.

For the specific subject of evaluating the Tutte polynomial parameterized by width measures, research has already been done in this area over twenty years ago: Noble [21] has given a polynomial time algorithm for evaluation the Tutte Polynomial on bounded treewidth graphs. Noble mostly focused on the dependence on the number of vertices and edges, and showed each point of the Tutte polynomial can be evaluated in linear time, assuming the treewidth of the graph is constant. See also an independently discovered (but slower) algorithm by Andrzejak [1]. However, this glances over the *exponential* part of the runtime, i.e. the dependence on the treewidth. Since this is typically the bottleneck, recent work aims to refine our understanding of this exponential dependence with upper and lower bounds on complexity of the problem in terms of this parameter that match in a fine-grained sense.

In this work, we extend this research line and determine the fine-grained complexity for each integer point (x,y) of the problem of evaluating the Tutte polynomial (x,y). As was done in previous works, we base our lower bounds on the Exponential Time Hypothesis (ETH) and the Strong Exponential Time Hypothesis (SETH) formulated by Impagliazzo and Paturi [14]. For a given width parameter k, the former will be used to exclude run times of the form $k^{o(k)}n^{O(1)}$, while the latter will be used to exclude run times of the form $(c-\epsilon)^k n^{O(1)}$ for some constant c and any $\epsilon > 0$.

Specifically we consider the *treewidth*, *pathwidth* and *cutwidth* of the graph. The first two, in some sense, measure how close the graph is to looking like a tree or path respectively. The cutwidth measures how many edges are layered on top of each other when the vertices are placed in any linear order. We will more precisely define these parameters in the preliminaries.

[1] For example, the biennial Workshop on Graph Classes, Optimization, and Width Parameters (GROW) already had its 10'th edition recently `https://conferences.famnit.upr.si/event/22/`.

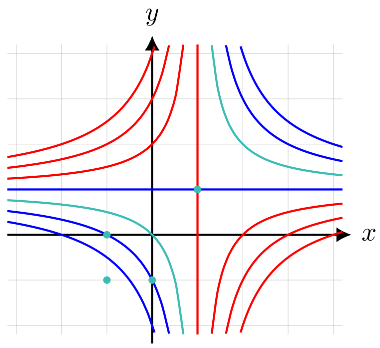

Figure 1 The red points have time complexity of the form $k^{O(k)}$, the blue points have time complexity of the form $O(c^k)$ for some constant c and the green points have polynomial time complexity.

Width measures in particular are interesting because instances where such structural parameters are small come up a lot in practice. For example, the curve H_2 corresponds to the partition function of the Ising model, which is widely studied in statistical physics, on graphs with particular topology such as lattice graphs or open/closed Cayley trees ([18]). In all such graphs with n vertices, even the cutwidth (the largest parameter we study) is at most $O(\sqrt{n})$.

1.1 Our contributions

Our classification handles points (x, y) differently based on whether $(x-1)(y-1)$ is negative, zero or positive, and reads as follows:

▶ **Theorem 1.1.** *Let G be a graph with given tree, path and cut decompositions of width* tw, pw *and* ctw *respectively. Let $(x, y) \in \mathbb{Z}^2$ be a non-special point, then up to some polynomial factor in $|V(G)|$, the following holds.*

1. *If $(x-1)(y-1) < 0$ or $x = 1$, then $T(G; x, y)$ can be computed in time $\mathrm{tw}^{O(\mathrm{tw})}$ and cannot be computed in time $\mathrm{ctw}^{o(\mathrm{ctw})}$ under ETH.*
2. *If $y = 1$, then $T(G; x, y)$ can be computed in time $O(4^{\mathrm{pw}})$ or $O(64^{\mathrm{tw}})$ and cannot be computed in time $2^{o(\mathrm{ctw})}$ under ETH.*
3. *If $(x-1)(y-1) = q > 1$, then $T(G; x, y)$ can be computed in time $O(q^{\mathrm{tw}})$. Furthermore,*
   a. *if $x \neq 0$, then $T(G; x, y)$ cannot be computed in time $O((q-\epsilon)^{\mathrm{ctw}})$ under SETH.*
   b. *if $x = 0$, then $T(G; x, y)$ cannot be computed in time $O((q-\epsilon)^{\mathrm{pw}})$ and $O(q-\epsilon)^{\mathrm{ctw}/2})$ under SETH.*

This is a fine-grained classification for evaluating the Tutte polynomial at any given integer point, simultaneously for all the parameters treewidth, pathwidth and cutwidth. This is because if a graph has cutwidth ctw, pathwidth pw and treewidth tw, then $\mathrm{tw} \leq \mathrm{pw} \leq \mathrm{ctw}$. Our result implies that, for evaluating the Tutte polynomial at a given integer point, it does not give a substantial advantage to have small cutwidth instead of small treewidth. This is somewhat surprising since, for example, for computing the closely related chromatic number of a graph there exists a $2^{\mathrm{ctw}} n^{O(1)}$ time algorithm, but any $\mathrm{pw}^{o(\mathrm{pw})} n^{O(1)}$ time algorithm would contradict the ETH [19].

Of particular interest are the upper bounds in Case 2. for the points $\{(x, y) : y = 1\}$, which are closely related to the problem of computing the number of forests in the input graph. One reason why this results stands out in particular is that it indicates an inherent asymmetry between the x- and y-axes, in this parameterized setting. In the general setting, problems related to the Tutte Polynomial often have a natural dual problem, which one can obtain by interchanging the x- and y-coordinates. For example the chromatic polynomial can be found (up to some computable term f) as $\chi_G(\lambda) = f(\lambda)T(1-\lambda, 0)$, while the flow polynomial can be found as $C_G(\lambda) = g(\lambda)T(0, 1-\lambda)$. These two problems are equivalent on planar graphs, in the sense that the chromatic number of a planar graph is equal to the flow number of its dual graph.

We note that for this curve we have an ETH bound, while for the other results of the form $c^{\text{tw}}n^{O(1)}$ we have a stronger SETH bound. We suspect that a $(4-\epsilon)^{\text{ctw}}n^{O(1)}$ lower bound for any $\epsilon > 0$, based on SETH, also holds for evaluating $T(G; 2, 1)$, but that it will take significant additional technical effort.

Techniques. In order to get the classification, our first step follows the method of [15] to reduce the evaluation of $T(G; x, y)$ for all points in hyperbola $H_\alpha = \{(x, y) : (x-1)(y-1) = \alpha\}$ to the evaluation to a *single* point in H_α. This is achieved in [15] by some graph operations (*stretch* and *thickening*), but these may increase the involved width parameters. We refine these operations in Section 3 to avoid this.

With this step being made, several cases of Theorem 1.1 then follow from a combination of new short separate and non-trivial arguments and previous work (including some very recent work such as [8, 13]).

However, for the upper bound in Case 2. of Theorem 1.1, our proof is more involved. To get our upper bound, we introduced the *forest compatibility matrix.* Its rows and columns are indexed with forests (encoded as partitions indicating their connected components). An entry in this matrix indicates whether the union of the two forests forms a forest itself. This matrix is closely related to matrices playing a crucial role in the Cut and Count method [11] and rank based method [4] to quickly solve connectivity problems on graphs with small tree-width. However, the previous rank upper bounds do not work for bounding the rank of the forest compatibility matrix over the reals since we check for *acyclicity* instead of *connectivity.* We nevertheless show that this the rank of this matrix is 4^n; in fact the set of non-crossing partitions forms a basis of this matrix. We prove this via an inductive argument that is somewhat similar to the rank bound of $2^{n/2-1}$ of the matchings connectivity matrix over $GF(2)$ from [10]. Subsequently, we show how to use this insight to get a $2^{O(\text{tw})}$ algorithm to evaluate $T(G; 1, 2)$ (i.e. counting the number of spanning forests).

1.2 Organization

The remainder of this paper supports Theorem 1.1, although some slightly less interesting cases (being the upper bound in Case 1.) are deferred to the full version of the paper [20]. Proofs of Lemmas and Theorems indicated with † are also deferred to the full version [20].

In Section 2 we describe some preliminaries. In Section 3 we show how to reduce the task of computing all points along a hyperbola curve to a single point. We now describe where each part of Theorem 1.1 can be found in the paper. The lower bound in Case 1. is given in Theorem 5.7 and 5.3. The lower bound in Case 2. is by Dell et al. [12]. The upper bound in Case 2. is given in Section 4 (specifically, Theorems 4.17 and 4.18). The lower bound in Case 3. is given in Theorem 5.1 (for $q = 2$) and Theorem 5.4 (for $q > 2$). The upper bound bound in 3. is given in Theorem 5.2 (for $q = 2$) and Theorem 5.5 (for $q > 2$).

2 Preliminaries

Computational Model. In this paper we frequently have real (and some inteormediate lemma's are even stated for even complex) numbers as intermediate results of computations. However, as is common in this area we work in the word RAM model in which all basic arithmetic operations with such numbers can be done in constant time, and therefore this does not influence our running time bounds.

Interpolation. Throughout this paper we will use interpolation to derive a polynomial, given a finite set of evaluations of said polynomial. For our purposes it suffices to note that this can be done in polynomial time, for example by solving the system of linear equations given by the Vandermonde matrix and the evaluations (see e.g. [7, Section 30.1]).

▶ **Lemma 2.1.** *Given pairs $(x_0, y_0), \ldots, (x_d, y_d)$, there exists an algorithm which computes the unique degree d polynomial p such that $p(x_i) = y_i$ for $i = 0, \ldots, d$ and runs in time $O(d^3)$.*

2.1 The Tutte polynomial

There are multiple ways of defining the Tutte polynomial. In this paper we will only need the following definition

$$T(G; x, y) = \sum_{A \subseteq E} (x-1)^{k(A)-k(E)}(y-1)^{k(A)+|A|-|V|},$$

where $k(A)$ denotes the number of connected components of the graph (V, A). We will often use the following notation

$$H_\alpha = \{(x, y) : (x-1)(y-1) = \alpha\}.$$

Note that these curves form hyperbolas and that for $\alpha = 0$ the hyperbola collapses into two orthogonal, straight lines. We refer to these two lines as separate curves

$$H_0^x = \{(x, y) : x = 1\},$$
$$H_0^y = \{(x, y) : y = 1\}.$$

Throughout the paper we will refer to the problem of finding the value of $T(G; a, b)$ for an individual point as *computing the Tutte polynomial on* (a, b). We will often restrict the Tutte polynomial to a one-dimensional curve H_α. Note that in this case the polynomial can be expressed as a univariate polynomial[2]

$$T_\alpha(G; t) := T\left(G; \frac{\alpha}{t} + 1, t + 1\right).$$

We refer to the problem of finding the coefficients of T_α as *computing the Tutte polynomial along* H_α.

As mentioned in the introduction, the Tutte polynomial is known to be computable in polynomial time on the points

$$(1, 1), (-1, -1), (0, -1), (-1, 0), (i, -i), (-i, i), (j, j^2), (j^2, j) \tag{2}$$

and along the curve H_1 and it is #P to evaluate it on any other point. We call the points listed in (2), along with the points on the curve H_1 *special points.* See [15] for more details.

---

[2] Note that one can get rid of the negative powers of t in the following expression, by multiplying the whole polynomial by some power of t.

2.2 Width measures

We consider the width measures *treewidth*, *pathwidth* and *cutwidth* of a graph G (denoted respectively with $tw(G)$, $pw(G)$ and $ctw(G)$). We use standard notation (such as introduced in [9]); see the appendix

2.3 Brylawski's tensor product formula

In Section 3 we will make use of Brylawski's tensor product formula [6] to reduce the computation of $T(G; x', y')$ to that of $T(G'; x, y)$ for some other point (x', y') and some other graph G'. The original formula is formulated in terms of pointed matroids, however we will only need the formulation for (multi)graphs. Before we can state the formula, we first need to introduce some notation.

Given graphs G and H, where an edge $e \in E(H)$ is labeled as a special edge, we define the *pointed tensor product*[3] $G \otimes_e H$ of G and H as the graph given by the following procedure. For every edge $f \in E(G)$ we first create a copy H_f of H, then identify f with the copy of the edge e in H_f and finally remove the edge f (and thus also the edge e) from the graph.

Intuitively it might be easier to think of this product as replacing every edge of G with a copy $H \setminus e$, where two of the vertices in H are designated as gluing points. For example one could replace every edge with a path of length k by taking as H the cycle C_{k+1} on $k+1$ vertices, as seen in figure 2.

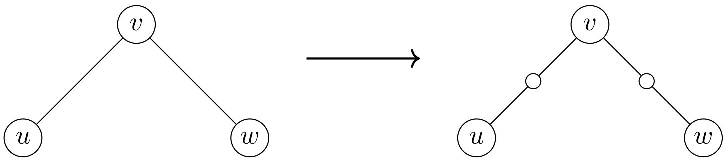

Figure 2 The pointed tensor product of the left-hand graph with a 3-cycle is given by the right-hand graph.

Note that this is not always well-defined, as one can choose which endpoint is identified with which. It turns out that this choice does not affect the graphic matroid of $G \otimes_e H$ and thus it does not affect the resulting Tutte polynomial. In this paper we will only consider graphs H that are symmetric over e and thus the product is actually well-defined.

We are now ready to state Brylawski's tensor product formula. Let T_C and T_L be the unique polynomials that satisfy the following system of equations

$$(x-1)T_C(H; x, y) + T_L(H; x, y) = T(H \setminus e; x, y)$$
$$T_C(H; x, y) + (y-1)T_L(H; x, y) = T(H/e; x, y).$$

We define

$$x' = \frac{T(H \backslash e; x, y)}{T_L(H; x, y)} \qquad y' = \frac{T(H/e; x, y)}{T_C(H; x, y)}.$$

Let $n = |V(H)|$, $m = |E(H)|$ and $k = k(E(H))$. Brylawski's tensor product formula states that

$$T(G \otimes_e H; x, y) = T_C(H; x, y)^{m-n+k} T_L(H; x, y)^{n-k} T(G; x', y').$$

[3] Note that this is different from the standard tensor product for graphs.

3 Reducing along the curve H_α

In this section we describe how we can lift hardness results from a single point $(a, b) \in H_\alpha$ to the whole curve H_α. We summarize the results from this section in the following theorem.

▶ **Theorem 3.1.** *Let $(a, b) \in \mathbb{C}^2$. Also let $T(G; x, y)$ be the Tutte polynomial of G and $\alpha := (a-1)(b-1)$. There exists a polynomial time reduction from computing T on (a, b) for graphs of given tree-, path- or cutwidth, to computing T along H_α for graphs width the following with parameters.*

- *If $|a| \notin \{0, 1\}$ or if $|b| \notin \{0, 1\}$ and $a \neq 0$, then the treewidth remains $\mathrm{tw}(G)$. The cutwidth and pathwidth become at most $\mathrm{ctw}(G) + 2$ and $\mathrm{pw}(G) + 2$ respectively.*
- *If $|b| \notin \{0, 1\}$ and $a = 0$, then the treewidth remains $\mathrm{tw}(G)$. The pathwidth becomes at most $\mathrm{pw}(G) + 2$ and the cutwidth becomes at most $2\,\mathrm{ctw}(G)$.*
- *If $|a|, |b| \in \{0, 1\}$, then the treewidth remains $\mathrm{tw}(G)$. The pathwidth becomes at most $\mathrm{pw}(G) + 2$ and the cutwidth becomes at most $12\,\mathrm{ctw}(G)$.*

Theorem 3.1 lets us lift both algorithms and lower bounds from a point (a, b) to the whole curve H_α. While our main theorem only requires Theorem 3.1 to be stated for integer valued points, we will state it as the most general version we can prove. We note that for Case 1. of Theorem 1.1, we do not care too much about constant multiplicative factors in the cutwidth, since we have an ETH bound of the form $\mathrm{ctw}(G)^{o(\mathrm{ctw}(G))}$. For Case 2. we only need the bounds on the treewidth and pathwidth. Thus the blowup in the cutwidth is only relevant for Case 3.. In this case the only integer valued points that fall under the third item of Theorem 3.1 are $(-1, 0), (0, -1)$ and $(-1, -1)$. These are all special points, which means that this item is not relevant for Case 3..

In our proofs we will make use of the following transformations.

▶ **Definition 3.2** ([15])**.** *Let G be a simple graph. We define the k-*stretch *${}^kG$ of G as the graph obtained by replacing every edge by a path of length k. We define the k-*thickening *${}_kG$ of G as the graph obtained by replacing every edge by k parallel edges.*

A new variant we introduce to keep the cutwidth low is defined as follows:

▶ **Definition 3.3.** *We define the* insulated k-thickening *${}_{(k)}G$ as the graph obtained by replacing every edge by a path of length 3 and then replacing the middle edge in each of these paths by k parallel edges.*

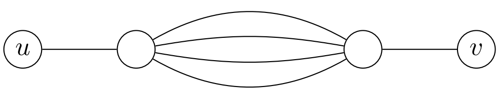

Figure 3 The result of applying the insulated 4-thickening to an edge between u and v.

3.1 Effect on width parameters

We give three lemmas that show how these transformations effect the parameters we use.

▶ **Lemma 3.4** (†)**.** *Let G be a graph. Then we have that $\mathrm{tw}({}^kG) \leq \mathrm{tw}(G)$, $\mathrm{tw}({}_kG) \leq \mathrm{tw}(G)$ and $\mathrm{tw}({}_{(k)}G) \leq \mathrm{tw}(G)$.*

▶ **Lemma 3.5** (†)**.** *Let G be a graph. Then we have that $\mathrm{pw}({}^kG) \leq \mathrm{pw}(G) + 2$, $\mathrm{pw}({}_kG) \leq \mathrm{pw}(G)$ and $\mathrm{pw}({}_{(k)}G) \leq \mathrm{pw}(G) + 2$.*

▶ **Lemma 3.6** (†). *Let G be a graph. Then we have that* $\mathrm{ctw}({}^kG) \leq \mathrm{ctw}(G)$, $\mathrm{ctw}({}_kG) \leq k\,\mathrm{ctw}(G)$ *and* $\mathrm{ctw}({}_{(k)}G) \leq \mathrm{ctw}(G) + k - 1$.

We remark that the only significant blowup is that of the cutwidth, when applying the k-thickening. We will therefore limit our use of this transformation as much as possible.

3.2 Reductions

We now discuss the proof of Theorem 3.1. In the full version in the appendix we split the theorem into multiple separate cases. Here, we only give one case as a representative sample:

▶ **Lemma 3.7.** *Let $(a, b) \in \mathbb{C}^2$ be a point with $|a| \notin \{0, 1\}$. Also let $T(G; x, y)$ be the Tutte polynomial of G and $\alpha := (a-1)(b-1)$. There exists a polynomial time reduction from computing T on (a, b) for graphs of given tree-, path- or cutwidth, to computing T along H_α for graphs with the following with parameters. The treewidth and cutwidth remain* $\mathrm{tw}(G)$ *and* $\mathrm{ctw}(G)$ *respectively. The pathwidth becomes at most* $\mathrm{pw}(G) + 2$.

We prove this lemma using essentially the same proof as given in [15]. Note that in our setting we use Lemmas 3.4, 3.5 and 3.6 to ensure that relevant parameters are not increased by the operations we perform.

Proof. By Brylawski's tensor product formula [6], we find the following expression for the k-stretch of the graph G

$$(1 + a + \cdots + a^{k-1})^{k(E)} T\left(G; a^k, \frac{b + a + \cdots + a^{k-1}}{1 + a + \cdots + a^{k-1}}\right) = T({}^kG; a, b). \tag{3}$$

Note that

$$a^k - 1 = (1 + a + \cdots + a^{k-1})(a - 1)$$

and

$$\frac{b + a + \cdots + a^{k-1}}{1 + a + \cdots + a^{k-1}} - 1 = \frac{b - 1}{1 + a + \cdots + a^{k-1}}.$$

We find that the point on which we evaluate $T(G)$ in (3) also lies on H_α.

By examining the formula for the Tutte polynomial, we find that for $n = |V(G)|$ the degree of the Tutte polynomial is at most $n^2 + n$. By choosing $k = 0, \ldots, n^2 + n$, since $|a| \notin \{0, 1\}$, we can find $T(G; x, y)$, for $n^2 + n + 1$ different values of $(x, y) \in H_\alpha$. By lemma 2.1, we can now interpolate the univariate restriction

$$T_\alpha(G; t) = T\left(G; \frac{\alpha}{t} + 1, t + 1\right).$$

of $T(G)$ along H_α.

Note that by Lemmas 3.4 and 3.6 the k-stretch preserves both the cutwidth and the treewidth of the graph and by Lemma 3.5 the pathwidth increases by a constant additive factor. We find that any fine-grained parameterized lower bound for H_α extends to points (a, b). ◀

4 Counting forests

In this section we consider the problem of counting the number of forests in a graph. This problem corresponds to the point $(2,1)$ and thus by Theorem 3.1 any bounds found for this problem can be lifted to the whole curve H_0^y.

We trivially get the following lower bound from existing bounds on the non-parameterized version of the problem [12].

▶ **Theorem 4.1.** *Computing the Tutte polynomial along the curve H_0^y cannot be done in time $2^{o(\mathrm{ctw}(G))}n^{O(1)}$, unless #ETH fails.*

To complement this lower bound, we give an algorithm to count the number of forests in a graph G in $c^{\mathrm{tw}(G)}$ time. The algorithm uses a rank based approach, the runtime of which depends on the rank of the so called *forest compatibility matrix.* We start by introducing this matrix and examining its rank.

4.1 Notation

We will use the notation $[n] = \{1, \ldots, n\}$. Unless stated otherwise, we will assume the set $[n]$ to be ordered. We write $\pi \vdash S$ to indicate that π is a partition of S.

We will consider matrices indexed by partitions. We will write $M[\pi, \rho]$ for the element in the row corresponding to π and the column corresponding to ρ. We will write $M[\pi]$ for the vector containing all elements in the row corresponding to π.

4.2 Rank bound

In this section we prove the following theorem, for the so called *forest compatibility matrix* F_n.

▶ **Theorem 4.2.** *The rank of F_n is at most C_n, the n^{th} Catalan number. In particular* $\mathrm{rank}(F_n) = O(4^n n^{-3/2})$

Before we can define the forest compatibility matrix, we first need the following definitions.

▶ **Definition 4.3.** *We say that a boundaried graph $G = ([n] \cup V, E)$, with boundary $[n]$, is a* representative forest *for a partition $\pi \vdash [n]$, if for every $S \in \pi$ there is some connected component $C \subseteq V(G)$ such that $C \cap [n] = S$.*

Given two boundaried graphs G and H, both with boundary B, we define the glue *$G \oplus H$ of G and H as follows. First take the disjoint union of G and H. Then identify each $v \in B$ in G with its analogue in H.*

This definition shows how one can relate forests and partitions. Throughout the section we will mostly consider partitions as they capture only the information we need. The following definition elaborates on this by lifting the concept of cycles in a clue of two trees to a cycle inducet by two partitions.

▶ **Definition 4.4.** *Let $\pi, \rho \vdash [n]$ and let G_π and G_ρ be representative forests of π and ρ respectively. We say that π and ρ* induce a cycle *if $G_\pi \oplus G_\rho$ contains a cycle.*

It is not hard to see that it does not matter which representatives G_π and G_ρ we choose, since one only needs to know the connected components on $[n]$. This means that this definition is indeed well-defined. For this same reason, in the following definition, we only need a row and column for each partition of the separator.

▶ **Definition 4.5.** *We define the* forest compatibility matrix F_S *of a set* S *by*

$$F_S[\pi, \rho] := \begin{cases} 0 & \text{if } \pi \text{ and } \rho \text{ induce a cycle} \\ 1 & \text{otherwise} \end{cases}$$

for any $\pi, \rho \vdash S$. *We will write* $F_n := F_{[n]}$.

Finally we will need the following definition to bound the rank of the forest compatibility matrix.

▶ **Definition 4.6.** *We say that two sets* $A, B \in \pi$ *are* crossing *on an ordering* $<$, *if there are* $a_1, a_2 \in A$ *and* $b_1, b_2 \in B$ *such that* $a_1 < b_1 < a_2 < b_2$ *or* $b_1 < a_1 < b_2 < a_2$. *If a partition contains two crossing sets, we refer to it as a* crossing partition.

Throughout this section it will sometimes be convenient to think of the ordering as a permutation.

The general idea behind the proof of Theorem 4.2 is to show that any partition can be "uncrossed", i.e. its row in F_n can be written as a linear combination of rows, corresponding to non-crossing partitions.

4.2.1 Manipulating partitions

For the proof of Theorem 4.2 we will need the following operations, which will allow us to manipulate partitions by contracting an expanding intervals and projecting down to subsets of the ground set.

▶ **Definition 4.7.** *An* interval *is a subset* $I \subseteq [n]$ *of consecutive numbers, i.e. there is no* $b \notin I$ *such that* $a_1 < b < a_2$ *for some* $a_1, a_2 \in I$. *Given an interval* I *and a partition* π *of* $[n]$, *we define the* contraction $\pi -_i I$ *of* π *by* I *as the partition of the set* $[n] -_i I := ([n] \cup \{i\}) \setminus I$ *given by we merging all sets that intersect* I *and replacing* I *by a single element* i, *i.e.*

$$\pi -_i I := \{S \in \pi : S \cap I = \emptyset\} \cup \left\{ \left(\bigcup \{S \in \pi : S \cap I \neq \emptyset\} \cup \{i\} \right) \setminus I \right\}.$$

If we have an ordering on $[n]$, *we place* i *in the same place in the ordering as* I, *that is for any* $a \in [n] \setminus I$ *and* $b \in I$, *we have* $a < b$ *if and only if* $a < i$.

We define the blowup $\pi +_i I$ *of* π *by* I *as the partition of the set* $[n] +_i I := ([n] \cup I) \setminus \{i\}$, *given by adding all elements of* I *to the set that contains* i *and then removing* i, *i.e.*

$$\pi +_i I := \{S \in \pi : i \notin S\} \cup \{(S \setminus \{i\}) \cup I : i \in S\}.$$

Again we place I *in the same place in the ordering as* i.

We will sometimes abuse notation and refer to $[n] -_i I$ as simply $[n']$ for $n' = n - |I| + 1$.

We now turn our attention to a number of useful lemmas. The first lemma intuitively says that if we contract an interval contained in some partition, then any decomposition of the resulting smaller partition gives the same decomposition of the larger partition.

▶ **Lemma 4.8.** *Let* π *be a partition of* $[n]$ *and let* I *be an interval such that* $I \subseteq S \in \pi$. *We set* $n' = n - |I| + 1$. *Suppose that for some set of partitions* $\mathcal{R}$ *of* $[n']$, *we have* $F_{n'}[\pi -_i I] = \sum_{\rho \in \mathcal{R}} a_\rho F_{n'}[\rho]$. *Then* $F_n[\pi] = \sum_{\rho \in \mathcal{R}} a_\rho F_n[\rho +_i I]$.

Proof. Let χ be some partition of $[n]$. Note that if $|S' \cap I| \geq 2$ for some $S' \in \chi$, we have that $F_n[\pi, \chi] = F_n[\rho +_i I, \chi] = 0$. Thus we may assume that χ contains no such sets. Also note that if there is some cycle that only requires I and not the rest of S, then again we have that $F_n[\pi, \chi] = F_n[\rho +_i I, \chi] = 0$. Thus we may assume that any cycle induced by χ and π that has a set that intersects I, also requires a set that intersects $S \setminus I$, but not I.

We now claim that for χ with the above assumptions we have $F_n[\rho +_i I, \chi] = F_n[\rho, \chi -_i I]$ for any ρ. This would immediately imply that for such χ

$$F_n[\pi, \chi] = F_n[\pi -_i I, \chi -_i I] = \sum_{\rho \in R} a_\rho F_n[\rho, \chi -_i I] = \sum_{\rho \in R} a_\rho F_n[\rho +_i I, \chi],$$

which proves the lemma.

First note that if ρ and $\chi -_i I$ induce a cycle, that does not involve i, then $\rho +_i I$ and χ also induce that same cycle and vice versa.

Now suppose that $\rho +_i I$ and χ induce a cycle involving I, then there is some S' in the cycle that intersects I. By assumption there is also some set $S'' \in \chi$ in the cycle, that intersects $S \setminus I$, but not I. W.l.o.g. the cycle does not loop back on itself and thus these sets are the only two in the cycle that intersect S. Note that S' gets merged into the set containing i, but S'' does not. Since the rest of the cycle does not involve I, it is unaffected and thus the cycle remains intact after contraction.

In the reverse direction we assume that ρ and $\chi -_i I$ induce a cycle involving i, then it is clear to see that this cycle survives after blowing up i, using one of the sets in χ that intersect I. This proves the claim and thus the lemma. ◀

This next lemma intuitively says that if we project our partition to a subset of the ground set, then any decomposition of the resulting smaller partition gives the same decomposition of the larger partition.

▶ **Lemma 4.9.** *Let π be a partition of $[n]$ and let $n' < n$. Suppose that for some set of partitions $\mathcal{R}$ of $[n']$, we have $F_{n'}[\pi|_{[n']}] = \sum_{\rho \in \mathcal{R}} a_\rho F_{n'}[\rho]$, then $F_n[\pi] = \sum_{\rho \in \mathcal{R}} a_\rho F_n[\rho \sqcup \pi|_{[n]\setminus[n']}]$.*

Proof. Let χ be some partition of $[n]$. If χ and $\pi|_{[n]\setminus[n']}$ induce a cycle, then the statement trivially holds. In the rest of the proof we will therefore assume that any cycle induced by χ and π requires the use of $\pi|_{[n']}$.

We first define an equivalence relation $\sim$ on $[n]$ by defining two elements to be equivalent if they are either in the same set of χ or in the same set of $\pi|_{[n]\setminus[n']}$. We then complete this to a full equivalence relation. We now define the partition χ' of $[n']$ as the set of equivalence classes of $\sim$, restricted to $[n']$.

We claim that $F_n[\rho \sqcup \pi|_{[n]\setminus[n']}, \chi] = F_{n'}[\rho, \chi']$ for any ρ, which would immediately imply that

$$F_n[\pi, \chi] = F_{n'}[\pi|_{[n']}, \chi'] = \sum_{\rho \in \mathcal{R}} a_\rho F_{n'}[\rho, \chi'] = \sum_{\rho \in \mathcal{R}} a_\rho F_n[\rho \sqcup \pi|_{[n]\setminus[n']}, \chi]$$

which proves the lemma.

Suppose that $\rho \sqcup \pi|_{[n]\setminus[n']}$ and χ induce some cycle. Since the cycle must pass through $[n']$, there must be some path from one element of $[n']$ to another, induced by $\rho \sqcup \pi|_{[n]\setminus[n']}$ and χ. Since all elements in this path are equivalent, this path must lie entirely inside of a set $S' \in \chi'$ and thus replacing such a path with S' results in a cycle induced by ρ and χ'. Note that if a cycle only requires sets from $\pi|_{[n']}$, this operation results in a single set S' from χ' in the new cycle. However, since any set involved in the old cycle must contain at least two elements in the path, that set together with S' induces a cycle.

Similarly, in the reverse direction we take a cycle induced by ρ and χ' and blow up any sets of χ' into a path in the corresponding connected component to find a cycle induced by $\rho \sqcup \pi|_{[n]\setminus[n']}$ and χ. ◀

The following two lemmas help ensure that our operations do not introduce new crossings. The first of the two lemmas shows us that we can safely contract an interval, so long as it is contained in a set of the partition.

▶ **Lemma 4.10** (†). *Let $I \subseteq [n]$ be an interval of $[n]$. Let π be a non-crossing partition of $[n] -_i I$. Then $\pi +_i I$ is also non-crossing.*

This next lemma shows us that, in our setting, projection is safe, as long as we do not forget any elements of sets that cross one another.

▶ **Lemma 4.11** (†). *Let $\pi \vdash [n]$ be a partition such that only $A, B \in \pi$ cross each other and all other pairs of sets in π are non-crossing. Then for a non-crossing partition ρ of $A \cup B$ we have that $\rho \cup \pi|_{[n]\setminus(A\cup B)}$ is non-crossing.*

4.2.2 Proof of the rank bound

With Lemmas 4.8, 4.9, 4.10 and 4.11 in hand, we are now ready to describe the main uncrossing operation.

▶ **Lemma 4.12.** *Let π be a non-crossing partition on an ordering p. In time $O(n)$ we can find constants c_ρ, such that $F_n[\pi] = \sum_{\rho\in\mathcal{N}} c_\rho F_n[\rho]$, where $\mathcal{N}$ is the set of partitions that are non crossing on $p \circ (i, i+1)$.*

Proof. Throughout the proof, we will consider the partition π on the ordering $p \circ (i, i+1)$. We first note that since π is non-crossing on p, any crossing of π must involve both i and $i+1$. Let $i \in A \in \pi$ and $i+1 \in B \in \pi$. If $A = B$, then π is non-crossing and thus we may assume that $A \neq B$. Note that $\pi|_{A\cup B}$, when viewed as a partition of $A \cup B$, consists of either 4 or 5 intervals which alternate between A and B. Define π' as the partition given by contracting these intervals. We find that π' is a partition on n' elements, where either $n' = 4$ or $n' = 5$ elements, with intervals of size 1 (see Figure 4).

We can explicitly construct the forest compatibility matrices for $n' \in \{4, 5\}$ and check that the non-crossing partitions give a basis. With this paper we provide a MATLAB script that verifies this. Thus we can write

$$F_{n'}[\pi'] = \sum_{\rho\in\mathcal{R}} c_\rho F_{n'}[\rho],$$

where $\mathcal{R}$ is the set of non-crossing partitions of $[n']$. By Lemma 4.8 we find that

$$F_{A\cup B}[\pi|_{A\cup B}] = \sum_{\rho\in\mathcal{R}} c_\rho F_{A\cup B}[\rho +_{i_1} I_1 + \cdots +_{i_{n'}} I_{n'}].$$

By Lemma 4.10 each $\rho +_{i_1} I_1 + \cdots +_{i_{n'}} I_{n'}$ is still non-crossing. By Lemma 4.9 we find

$$F_n[\pi] = \sum_{\rho\in\mathcal{R}} c_\rho F_{A\cup B}[(\rho +_{i_1} I_1 + \cdots +_{i_{n'}} I_{n'}) \cup \pi|_{[n]\setminus(A\cup B)}].$$

By Lemma 4.11 each $(\rho +_{i_1} I_1 + \cdots +_{i_{n'}} I_{n'}) \cup \pi|_{[n]\setminus(A\cup B)}$ is still non-crossing. We conclude that $F_n[\pi]$ can be written as a linear combination of rows corresponding to non-crossing partitions.

Note that we can construct π' in $O(n)$ time. We then find the c_ρ in $O(1)$ time and reconstruct the $(\rho +_{i_1} I_1 + \cdots +_{i_{n'}} I_{n'}) \cup \pi|_{[n]\setminus(A\cup B)}$ in $O(n)$ time. ◀

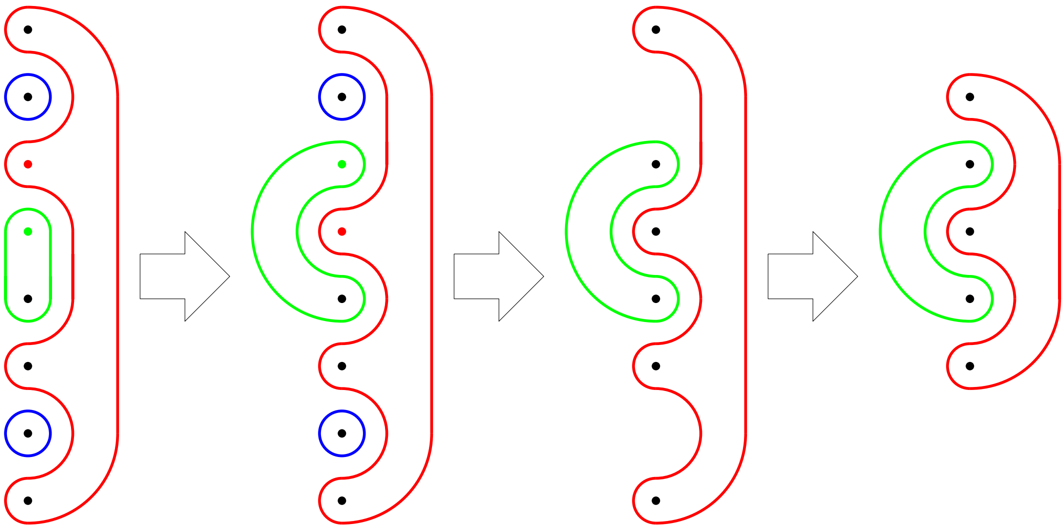

Figure 4 From left to right, these are examples of π before the swap, π after the swap, $\pi|_{A\cup B}$ and π'.

By repeatedly applying Lemma 4.12, we can prove the following theorem.

▶ **Theorem 4.13.** *The rows corresponding to non-crossing partitions span a row basis of the forest compatibility matrix F_n.*

Proof. Let π be a partition of $[n]$ such that we can turn it into a non-crossing partition by swapping two consecutive elements i and $i+1$ in the order of $[n]$. By Lemma 4.12 we can write the row $F_n[\pi]$ corresponding to π as a linear combination of rows corresponding to non-crossing partitions of $[n]$. This shows that, for B_p the set of rows corresponding to non-crossing partitions on p, we have $B_{p\circ(i,i+1)} \subseteq \text{span}(B_p)$. Since every partition is non-crossing for some permutation and every permutation can be decomposed into 2-cycles on consecutive elements, this implies that every row can be written as a linear combination of rows corresponding to non-crossing partitions on some fixed ordering p. ◀

From this we immediately find a proof for Theorem 4.2.

Proof of Theorem 4.2. By Theorem 4.13 the non-crossing partitions form a basis of F_n. Since there are C_n such partitions we find $\text{rank}(F_n) \leq C_n$. ◀

4.3 Algorithm

We will now describe the algorithm for counting forests. We first define the dynamic programming table and the notion of representation. For details on how to compute the table entries, see the full version [20].

▶ **Definition 4.14.** *Let G be a graph and let $(\mathbb{T}, (B_x)_{x\in V(D)})$ be a tree/path decomposition of G. Recall that G_x is defined as the graph induced by the union of all bags, whose nodes are descendants of x in $\mathbb{T}$. We define the dynamic programming table τ by*

$$\tau_x[\pi] := |\{X \subseteq E(G_x) : (V,X) \text{ is acyclic },\\ \forall u,v \in B_x \text{ there is a path in } (V,X) \text{ from } u \text{ to } v \text{ iff } \exists S \in \pi \text{ s.t. } u,v \in S\}|$$

In other words, the table entry $\tau_x[\pi]$ counts the number of forests in G_x whose connected components agree with π. In the rest of this section, we will refer to the number of nonzero entries $\tau_x[\pi]$ in a 'row' τ_x of the dynamic programming table as the support of τ_x, written $\text{supp}(\tau_x)$. Our aim will be to ensure that the support of our rows remains contained in the entries corresponding to non-crossing partitions for some ordering on the bag B_x. This is captured in the following definition.

▶ **Definition 4.15.** *We say a vector a, indexed by partitions, is* reduced *on an ordering p, if $a_\pi = 0$ for any partition π that is crossing for p.*

In order to ensure that we do not lose any relevant information we will reduce our rows, while retaining the following property for the matrix F_{B_x}.

▶ **Definition 4.16.** *Given a matrix M, we say that a vector a M-*represents *a vector b if $Ma = Mb$*

In the full version of the paper[20], we describe a dynamic programming algorithm that works with reduced rows, rather than the whole table. It does so by alternating between reducing the current row in the table and computing the next row. This allows us to work with a table where the rows effectively have size $\text{rank}(F_{\text{tw}})$ (or $\text{rank}(F_{\text{pw}})$). Doing so, we establish the following:

▶ **Theorem 4.17** (†)**.** *There exists an algorithm that, given a graph G with a path decomposition of width $\text{pw}(G)$, computes the number of forests in the graph in time $4^{\text{pw}(G)}n^{O(1)}$.*

▶ **Theorem 4.18** (†)**.** *There exists an algorithm that, given a graph G with a tree decomposition of width $\text{tw}(G)$, computes the number of forests in the graph in time $64^{\text{tw}(G)}n^{O(1)}$.*

5 Other cases

In this section we handle the remaining cases mentioned in Theorem 1.1.

5.1 The curve H_2

The curve H_2 is equivalent to the partition function of the Ising model. Both our proofs for the upper and lower bound on the complexity will make use of this fact.

▶ **Theorem 5.1.** *Computing the Tutte polynomial along the curve H_2 cannot be done in time $(2-\epsilon)^{\text{ctw}(G)}n^{O(1)}$, unless SETH fails.*

Proof Sketch. Using known equivalences we first reduce #MaximumClosedSubgraphs to the problem of computing the Tutte polynomial along the curve H_2. We then apply a simple transformation, based on a similar argument by [17] to ensure the graph only has odd degree vertices. It then suffices to note that on graphs with only odd degree vertices, the complement of a perfect matching is a maximum closed subgraph and thus #MaximumClosedSubgraphs is equivalent to #PerfectMatchings on such graphs. This allows us to lift an existing lower bound from [8] on #PerfectMatchings to #MaximumClosedSubgraphs. ◀

We also show in the full version that this lower bound can be matched with a tight upper bound. The proof uses dynamic programming combined with subset convolution [3, 9].

▶ **Theorem 5.2.** *Let G be a graph with a given tree decomposition of width $\text{tw}(G)$. There exists an algorithm that computes $T(G; a, b)$, for $(a, b) \in H_2$, in time $2^{\text{tw}(G)}n^{O(1)}$.*

5.2 The curve H_0^x

The curve H_0^x contains the point $(1, 2)$, which counts the number of connected edgesets of a connected graph. Using existing results this gives an ETH lower bound which matches the running time of the algorithm mentioned in theorem 1.1.

▶ **Theorem 5.3.** *Let $0 < \alpha < 1$. Computing the Tutte polynomial along the curve H_0^x cannot be done in time $(\alpha \operatorname{ctw}(G) - \epsilon)^{(1-\alpha)\operatorname{ctw}(G)/2} n^{O(1)}$, unless SETH fails.*

Proof. In [13] a lower bound of $p^{\operatorname{ctw}(G)}$ is found for counting connected edgesets modulo p. In the reduction the authors reduce to counting essentially distinct q-coloring modulo p, with cutwidth $\operatorname{ctw}(G) + q^2$ and $p = q$. Thus we find a lower bound of $p^{\operatorname{ctw}(G)-p^2} = (\alpha \operatorname{ctw}(G))^{(1-\alpha)\operatorname{ctw}(G)/2}$ for $p = (\alpha \operatorname{ctw}(G))^{1/2}$. ◀

5.3 The curve H_q for $q \in \mathbb{Z}_{\geq 3}$

These curves contain the points $(1 - q, 0)$, which count the number of q-colorings. Using previous results and a folklore algorithm, we find matching upper and lower bounds for these points and thus for the whole curves.

▶ **Theorem 5.4.** *Let $q \in \mathbb{Z}_{\geq 3}$. Computing the Tutte polynomial along the curve H_q cannot be done in time $(q - \epsilon)^{\operatorname{ctw}(G)} n^{O(1)}$, unless SETH fails.*

Proof. Note that H_q contains the point $(1 - q, 0)$. Computing the Tutte polynomial on this point is equivalent to counting the number of q-colorings of the graph G.

By choosing a modulus $p > q$ we can apply the results from [13] to find a lower bound of $q^{\operatorname{ctw}(G)}$ on the time complexity of counting q-colorings modulo p. This lower bound clearly extends to general counting. ◀

▶ **Theorem 5.5.** *Let G be a graph with a given tree decomposition of width $\operatorname{tw}(G)$ and $q \in \mathbb{Z}_{\geq 3}$. There exists an algorithm that computes $T(G; a, b)$ for $(a, b) \in H_q$ in time $q^{\operatorname{tw}(G)} n^{O(1)}$.*

This theorem is a direct consequence of combining Theorem 3.1 with the following folklore result:

▶ **Theorem 5.6** (Folklore). *Let G be a graph with a given tree decomposition of width $\operatorname{tw}(G)$ and $q \in \mathbb{Z}_{\geq 3}$. There exists an algorithm that computes the number of q-colorings of G in time $q^{\operatorname{tw}(G)} n^{O(1)}$.*

5.4 The curve H_{-q} for $q \in \mathbb{Z}_{>0}$

These curves contain the points $(1 + q, 0)$. Using the same results we used to prove theorem 5.4 and exploiting the fact these results hold for modular counting, we find an ETH lower bound which matches the running time of the algorithm mentioned in theorem 1.1.

▶ **Theorem 5.7.** *Let $q \in \mathbb{Z}_{>0}$. Computing the Tutte polynomial along the curve H_{-q} cannot be done in $\operatorname{ctw}(G)^{o(\operatorname{ctw}(G))}$ time, unless ETH fails.*

Proof. Like mentioned earlier H_{-q} contains the point $(1 + q, 0)$. For a prime $p > q$ we have that $T(G; 1 + q, 0) \equiv_p T(G; 1 + q - p, 0)$. This means that computing the Tutte polynomial modulo p at the point $(1 + q, 0)$ is equivalent to counting the number of $p - q$-colorings of G modulo p. Since $q > 0$ and $p > q$ we find that $0 < p - q < p$ and thus as before, by [13], we find a lower bound of $(p - q)^{\operatorname{ctw}(G)}$. Since the cutwidth of the construction in [13] is $O(n + rp^{r+2})$ for some r dependant on $p - q$ and ϵ. We find that there is no algorithm running in time $O((p - q - \epsilon)^{\operatorname{ctw}(G) - rp^{r+2}}) = O((\alpha \operatorname{ctw}(G) - \epsilon)^{\operatorname{ctw}(G)(1-\alpha)/(r+2)})$, where $p - q = (\alpha \operatorname{ctw}(G))^{1/(r+2)}$. ◀

6 Conclusion

In this paper we gave a classification of the complexity, parameterized by treewidth/pathwidth/cutwidth, of evaluating the Tutte polynomial at integer points into either computable

- in polynomial time,
- in $\mathrm{tw}^{O(\mathrm{tw})} n^{O(1)}$ time but not in $\mathrm{ctw}^{o(\mathrm{ctw})} n^{O(1)}$ time,
- in $q^{\mathrm{tw}} n^{O(1)}$ time but not in $2^{o(\mathrm{ctw})}$ (and for many points not even in $r^{\mathrm{ctw}} n^{O(1)}$ time for some constants $q > r$),

assuming the (Strong) Exponential Time Hypothesis.

This classification turned out to be somewhat surprising, especially considering the asymmetry between $H_0^x = \{(x, y) : x = 1\}$ and $H_0^y = \{(x, y) : y = 1\}$ that does not show up in other classifications such as the ones from [5, 12, 15].

Our paper leaves ample opportunities for further research. First, we believe that our rank upper bound should have more applications for counting forests with different properties. For example, it seems plausible that it can be used to count all Feedback Vertex Sets in time $2^{O(\mathrm{tw})} n^{O(1)}$ or the number of spanning trees with k components in time $2^{O(\mathrm{tw})} n^{O(1)}$. The latter result would improve over a result by Peng and Fei Wan [22] that show how to count the number of spanning forests with k components (or equivalently, $n - k - 1$ edges) in $\mathrm{tw}^{O(\mathrm{tw})} n^{O(1)}$ time. We decided to not initiate this study in this paper to retain the focus on the Tutte polynomial.

Second, it would be interesting to see if our classification of the complexity of all points on $\mathbb{Z}^2$ can be extended to a classification of the complexity of all points on $\mathbb{R}^2$ (or even $\mathbb{C}^2$). Typically, evaluation at non-integer points can be reduced to integers points (leading to hardness for non-integer points), but we were not able to establish such a reduction without considerably increasing the width parameters.

Third, it would be interesting to see if a similar classification can be made when parameterized by the vertex cover number instead of treewidth/pathwidth/cutwidth. We already know that the runtime of $2^n n^{O(1)}$ by Björklund et al. [2] for evaluating the Tutte polynomial cannot be strengthened to a general $2^{O(k)} n^{O(1)}$ time algorithm where k is the minimum vertex cover size of the input graph due to a result by Jaffke and Jansen [16], but this still leaves the complexity of evaluating at many other points open.

References

1. Artur Andrzejak. An algorithm for the Tutte polynomials of graphs of bounded treewidth. *Discret. Math.*, 190(1-3):39–54, 1998. doi:10.1016/S0012-365X(98)00113-7.
2. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Computing the Tutte polynomial in vertex-exponential time. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 677–686. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.40.
3. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution, 2006. doi:10.48550/arXiv.CS/0611101.
4. Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015. doi:10.1016/j.ic.2014.12.008.
5. Cornelius Brand, Holger Dell, and Marc Roth. Fine-grained dichotomies for the Tutte plane and Boolean #csp. *Algorithmica*, 81(2):541–556, 2019. doi:10.1007/s00453-018-0472-z.
6. Thomas H Brylawski. The Tutte polynomial. In *Matroid Theory and Its Applications: Lectures Given at the Centro Internazionale Matematico Estivo*. Springer Berlin, Heidelberg, Varenna (como), Italy, 1980. doi:10.1007/978-3-642-11110-5.

7 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. URL: `http://mitpress.mit.edu/books/introduction-algorithms`.

8 Radu Curticapean and Dániel Marx. *Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus*, pages 1650–1669. Society for Industrial and Applied Mathematics, 2015. `doi:10.1137/1.9781611974331.ch113`.

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

10 Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Fast Hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, 2018. `doi:10.1145/3148227`.

11 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. *ACM Trans. Algorithms*, 18(2):17:1–17:31, 2022. `doi:10.1145/3506707`.

12 Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlén. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Transactions on Algorithms*, 10(4):1–32, August 2014. `doi:10.1145/2635812`.

13 Carla Groenland, Isja Mannens, Jesper Nederlof, and Krisztina Szilágyi. Tight Bounds for Counting Colorings and Connected Edge Sets Parameterized by Cutwidth. In *39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022)*, pages 36:1–36:20, 2022. `doi:10.48550/arXiv.2110.02730`.

14 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. `doi:10.1006/jcss.2000.1727`.

15 F. Jaeger, D. L. Vertigan, and D. J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society*, 108(1):35–53, 1990. `doi:10.1017/S0305004100068936`.

16 Lars Jaffke and Bart M. P. Jansen. Fine-grained parameterized complexity analysis of graph coloring problems. *Discret. Appl. Math.*, 327:33–46, 2023. `doi:10.1016/j.dam.2022.11.011`.

17 Mark Jerrum. Two-dimensional monomer-dimer systems are computationally intractable. *Journal of Statistical Physics*, 48(1-2):121–134, July 1987. `doi:10.1007/BF01010403`.

18 John E. Krizan, Peter F. Barth, and M.L. Glasser. Phase transitions for the Ising model on the closed Cayley tree. *Physica A: Statistical Mechanics and its Applications*, 119(1):230–242, 1983. `doi:10.1016/0378-4371(83)90157-7`.

19 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM J. Comput.*, 47(3):675–702, 2018. `doi:10.1137/16M1104834`.

20 Isja Mannens and Jesper Nederlof. A fine-grained classification of the complexity of evaluating the Tutte polynomial on integer points parameterized by treewidth and cutwidth. *CoRR*, To-appear, 2023. URL: `https://arxiv.org/abs/2307.01046`.

21 Steven D. Noble. Evaluating the Tutte polynomial for graphs of bounded tree-width. *Comb. Probab. Comput.*, 7(3):307–321, 1998. URL: `http://journals.cambridge.org/action/displayAbstract?aid=46641`.

22 Xin-Zhuang Chen Peng-Fei Wan. Computing the number of k-component spanning forests of a graph with bounded treewidth. *Journal of the Operations Research Society of China*, 7(2):385, 2019. `doi:10.1007/s40305-019-00241-4`.