Electronic Colloquium on Computational Complexity, Report No. 78 (2010)

Exponential Time Complexity of the Permanent and the Tutte Polynomial

Holger Dell[1]*, Thore Husfeldt[2], and Martin Wahlén[3]

[1] Humboldt University of Berlin, Germany
[2] IT University of Copenhagen, Denmark and Lund University, Sweden
[3] Lund University, Sweden and Uppsala University, Sweden

Abstract. The Exponential Time Hypothesis (ETH) says that deciding the satisfiability of n-variable 3-CNF formulas requires time $\exp(\Omega(n))$. We relax this hypothesis by introducing its counting version #ETH, namely that every algorithm that counts the satisfying assignments requires time $\exp(\Omega(n))$. We transfer the sparsification lemma for d-CNF formulas to the counting setting, which makes #ETH robust.

Under this hypothesis, we show lower bounds for well-studied #P-hard problems: Computing the permanent of an $n \times n$ matrix with m nonzero entries requires time $\exp(\Omega(m))$. Restricted to 01-matrices, the bound is $\exp(\Omega(m/\log m))$. Computing the Tutte polynomial of a multigraph with n vertices and m edges requires time $\exp(\Omega(n))$ at points (x, y) with $(x-1)(y-1) \neq 1$ and $y \notin \{0, \pm 1\}$. At points $(x, 0)$ with $x \notin \{0, \pm 1\}$ it requires time $\exp(\Omega(n))$, and if $x = -2, -3, \ldots$, it requires time $\exp(\Omega(m))$. For simple graphs, the bound is $\exp(\Omega(m/\log^3 m))$.

1 Introduction

The permanent of a matrix and the Tutte polynomial of a graph are central topics in the study of counting algorithms. Originally defined in the combinatorics literature, they unify and abstract many enumeration problems, including immediate questions about graphs such as computing the number of perfect matchings, spanning trees, forests, colourings, certain flows and orientations, but also less obvious connections to other fields, such as link polynomials from knot theory, reliability polynomials from network theory, and (maybe most importantly) the Ising and Potts models from statistical physics.

From its definition (repeated in (1) below), the permanent of an $n \times n$-matrix can be computed in $O(n!n)$ time, and the Tutte polynomial (2) can be evaluated in time exponential in the number of edges. Both problems are famously #P-hard, which rules out the existence of polynomial-time algorithms under standard complexity-theoretic assumptions, but that does not mean that we have to resign ourselves to brute-force evaluation of the definition. In fact, Ryser's famous formula [Rys63] computes the permanent with only $\exp(O(n))$ arithmetic

* Supported by the Deutsche Forschungsgemeinschaft within the research training group "Methods for Discrete Structures" (GRK 1408).

ISSN 1433-8092

operations, and more recently, an algorithm with running time $\exp(O(n))$ for n-vertex graphs has also been found [BHKK08] for the Tutte polynomial. Curiously, both of these algorithms are based on the inclusion–exclusion principle. In the present paper we show that these algorithms cannot be significantly improved, by providing conditional lower bounds of $\exp(\Omega(n))$ for both problems.

It is clear that #P-hardness is not the right conceptual framework for such claims, as it is unable to distinguish between different types of super-polynomial time complexities. For example, the Tutte polynomial for planar graphs remains #P-hard, but can be computed in time $\exp(O(\sqrt{n}))$ [SIT95]. Therefore, we work under the *Exponential Time Hypothesis* (ETH), viz. the complexity theoretic assumption that *some* hard problem (namely, Satisfiability of 3-CNF formulas in n variables) requires time $\exp(\Omega(n))$. More specifically, we introduce #ETH, a counting analogue of ETH which models the hypothesis that *counting* the satisfying assignments requires time $\exp(\Omega(n))$.

Computing the permanent. The permanent of an $n \times n$ matrix A is defined as

$$\operatorname{per} A = \sum_{\pi \in S_n} \prod_{1 \le i \le n} A_{i\pi(i)} , \tag{1}$$

where S_n is the set of permutations of $\{1, \ldots, n\}$. This is redolent of the determinant from linear algebra, $\det A = \sum_\pi \operatorname{sign}(\pi) \prod_i A_{i\pi(i)}$, the only difference is an easily computable sign for every summand. Both definitions involve a summation with $n!$ terms, but admit much faster algorithms that are textbook material: The determinant can be computed in polynomial time using Gaussian elimination and the permanent can be computed in $O(2^n n)$ operations using Ryser's formula.

Valiant's celebrated #P-hardness result [Val79] for the permanent shows that no polynomial-time algorithm à la "Gaussian elimination for the permanent" can exist unless P = NP, and indeed unless $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$. Several unconditional lower bounds for the permanent in restricted models of computation are also known. Jerrum and Snir [JS82] have shown that monotone arithmetic circuits need $n(2^{n-1} - 1)$ multiplications to compute the permanent, a bound they can match with a variant of Laplace's determinant expansion. Raz [Raz09] has shown that multi-linear arithmetic formulas for the permanent require size $\exp(\Omega(\log^2 n))$. Ryser's formula belongs to this class of formulas, but is much larger than the lower bound; no smaller construction is known. Intriguingly, the same lower bound holds for the determinant, where it is matched by a formula of size $\exp(O(\log^2 n))$ due to Berkowitz [Ber84]. One of the easy consequences of the present paper is that Ryser's formula is in some sense optimal under #ETH. In particular, no uniformly constructible, subexponential size formula such as Berkowitz's can exist for the permanent unless #ETH fails.

A related topic is the expression of $\operatorname{per} A$ in terms of $\det f(A)$, where $f(A)$ is a matrix of constants and entries from A and is typically much larger than A. This question has fascinated many mathematicians for a long time, see Agrawal's survey [Agr06]; the best known bound on the dimension of $f(A)$ is $\exp(O(n))$

and it is conjectured that all such constructions require exponential size. In particular, it is an important open problem if a permanent of size n can be expressed as a determinant of size $\exp(O(\log^2 n))$. Our paper shows that under #ETH, if such a matrix $f(A)$ exists, computing f must take time $\exp(\Omega(n))$.

Computing the Tutte polynomial. The Tutte polynomial, a bivariate polynomial associated with a given graph $G = (V, E)$ with n vertices and m edges, is defined as

$$T(G; x, y) = \sum_{A \subseteq E} (x-1)^{k(A)-k(E)} (y-1)^{k(A)+|A|-|V|}, \tag{2}$$

where $k(A)$ denotes the number of connected components of the subgraph (V, A).

Despite their unified definition (2), the various computational problems given by $T(G; x, y)$ for different points (x, y) differ widely in computational complexity, as well as in the methods used to find algorithms and lower bounds. For example, $T(G; 1, 1)$ equals the number of spanning trees in G, which happens to admit a polynomial time algorithm, curiously again based on Gaussian elimination. On the other hand, the best known algorithm for computing $T(G; 2, 1)$, the number of forests, runs in $\exp(O(n))$ time.

Computation of the Tutte polynomial has fascinated researchers in computer science and other fields for many decades. For example, the algorithms of Onsager and Fischer from the 1940s and 1960s for computing the so-called partition function for the planar Ising model are viewed as major successes of statistical physics and theoretical chemistry; this corresponds to computing $T(G; x, y)$ along the hyperbola $(x-1)(y-1) = 2$ for planar G. Many serious attempts were made to extend these results to other hyperbolas or graph classes, but "after a quarter of a century and absolutely no progress", Feynman in 1972 observed that "the exact solution for three dimensions has not yet been found". [4]

As for the permanent, the failure of theoretical physics to "solve the Potts model" and sundry other questions implicit in the computational complexity of the Tutte polynomial were explained only with Valiant's #P-hardness programme. After a number of papers, culminating in [JVW90], the polynomial-time complexity of exactly computing the Tutte polynomial at points (x, y) is now completely understood: it is #P-hard everywhere except at those points (x, y) where a polynomial-time algorithm is known; these points consist of the hyperbola $(x-1)(y-1) = 1$ as well as the four points $(1, 1), (-1, -1), (0, -1), (-1, 0)$.

In the present paper we show an $\exp(\Omega(n))$ lower bound to match the $\exp(O(n))$ algorithm from [BHKK08], which holds under #ETH everywhere except for $|y| = 1$. In particular, this establishes a gap to the planar case, which admits an $\exp(O(\sqrt{n}))$ algorithm [SIT95]. Our hardness results apply (though not everywhere, and sometimes with a weaker bound) even if the graphs are sparse and simple. These classes are of particular interest because most of the graphs arising from applications in statistical mechanics arise from bond structures, which are sparse and simple.

[4] The Feynman quote and many other quotes describing the frustration and puzzlement of physicists around that time can be found in the copious footnotes of [Ist00].

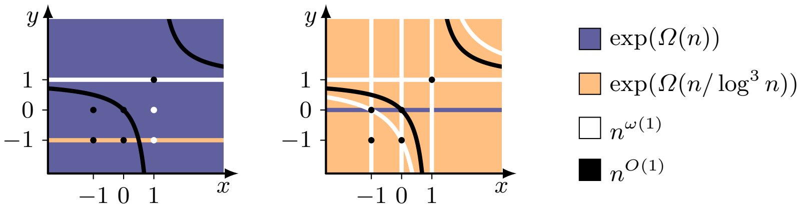


Fig. 1. Exponential time complexity under #ETH of the Tutte plane for multigraphs (left) and simple graphs (right) in terms of n, the number of vertices. White areas on the map correspond to uncharted territory. The black hyperbola $(x-1)(y-1) = 1$ and the four points close to the origin are in P. Everywhere else, in the shaded regions, we prove a lower bound exponential in n, or within a polylogarithmic factor of it.

It has been known since the 1970s [Law76] that graph 3-colouring can be solved in time $\exp(O(n))$, and this is matched by an $\exp(\Omega(n))$ lower bound under ETH [IPZ01]. Since graph 3-colouring corresponds to evaluating T at $(-2, 0)$, the exponential time complexity for $T(G; -2, 0)$ was thereby already understood. In particular, computing $T(G; x, y)$ for input G *and* (x, y) requires vertex-exponential time, an observation that is already made in [GHN06] without explicit reference to ETH.

The literature for computing the Tutte polynomial is very rich, and we make no attempt to survey it here. A recent paper of Goldberg and Jerrum [GJ08], which shows that the Tutte polynomial is even hard to approximate for large parts of the Tutte plane, contains an overview. A list of graph classes for which subexponential time algorithms are known can be found in [BHKK08].

2 Results

The exponential time hypothesis (ETH) as defined in [IPZ01] is that satisfiability of 3-CNF formulas cannot be computed substantially faster than by trying all possible assignments, i.e., it requires time $\exp(\Omega(n))$. We define the counting exponential time hypothesis via the counting version of 3-Sat.

Name #3-Sat
Input 3-CNF formula φ with n variables and m clauses.
Output The number of satisfying assignments to φ.

The best known algorithm for this problem runs in time $O(1.6423^n)$ [Kut07]. Our hardness results are based on the following hypothesis.

> (#ETH) There is a constant $c > 0$ such that no deterministic algorithm can compute #3-Sat in time $\exp(c \cdot n)$.

At the expense of making the hypothesis more unlikely, the term "deterministic" may be replaced by "randomized", but we ignore such issues here. Note that ETH trivially implies #ETH whereas the other direction is not known. By introducing the sparsification lemma, [IPZ01] show that ETH is a robust notion in the sense that the clause width 3 and the parameter n in its definition can be replaced by $d \geq 3$ and m, respectively, to get an equivalent hypothesis, albeit the constant c may change in doing so. We transfer the sparsification lemma to #d-SAT and get a similar kind of robustness for #ETH:

Theorem 1. *For all $d \geq 3$, #ETH holds if and only if #d-*SAT *requires time* $\exp(\Omega(m))$.

In the appendix, we go into more depth about computational complexity aspects of #ETH, including a proof of Thm. 1.

The Permanent. For a set S of rationals we define the following problems:

Name PERM^S
Input Square matrix A with entries from S.
Output The value of $\operatorname{per} A$.

We write PERM for $\text{PERM}^{\mathbb{N}}$. If B is a bipartite graph with a_{ij} edges from the ith vertex in the left half to the jth vertex in the right half $(1 \leq i, j \leq n)$, then $\operatorname{per}(a_{ij})$ equals the number of perfect matchings of B. Thus $\text{PERM}^{0,1}$ and PERM and can be viewed as counting the perfect matchings in bipartite graphs and multigraphs, respectively.

We express our lower bounds in terms of m, the number of non-zero entries of A. Without loss of generality, $n \leq m$, so the same bounds hold for the parameter n as well. Note that these bounds imply that the hardest instances have roughly linear density.

Theorem 2. *Under #ETH,*

(i) $\text{PERM}^{-1,0,1}$ *requires time* $\exp(\Omega(m))$.
(ii) PERM *requires time* $\exp(\Omega(m))$.
(iii) PERM^{01} *requires time* $\exp(\Omega(m/\log n))$.

The proof of this theorem is in §3. For (i), we follow a standard reduction by Valiant [Val79,Pap94] but use a simple equality gadget derived from [BD07] instead of Valiant's XOR-gadget. For (ii) we use interpolation to get rid of the negative weights. Finally, to establish (iii) we replace large positive weights by gadgets of logarithmic size, which increases the number of vertices and edges by a logarithmic factor.

The Tutte Polynomial. The computational problem TUTTE(x, y) is defined for each pair (x, y) of rationals.

Name TUTTE(x, y).
Input Undirected multigraph G with n vertices.
Output The value of $T(G; x, y)$.

In general, parallel edges and loops are allowed; we write $\textsc{Tutte}^{01}(x,y)$ for the special case where the input graph is simple.

Our main result is that under #ETH, $\textsc{Tutte}(x,y)$ requires time $\exp(\Omega(n))$ for specific points (x,y), but the size of the bound, and the graph classes for which it holds, varies. We summarise our results in the theorem below, see also Fig. 1. Our strongest reductions give edge-exponential lower bounds, i.e., bounds in terms of the parameter m, which implies the same bound in terms of n because $m \geq n$ in connected graphs. Moreover, a lower bound of $\exp(\Omega(m))$ together with the algorithm in time $\exp(O(n))$ from [BHKK08] implies that worst-case instances are *sparse*, in the sense that $m = O(n)$. At other points we have to settle for a vertex-exponential lower bound $\exp(\Omega(n))$. While this matches the best upper bound, it does not rule out a vertex-subexponential algorithm for sparse graphs.

Theorem 3. *Let $(x,y) \in \mathbb{Q}^2$. Under #ETH,*

(i) $\textsc{Tutte}(x,y)$ *requires time* $\exp(\Omega(n))$ *if* $(x-1)(y-1) \neq 1$ *and* $y \notin \{0, \pm 1\}$.
(ii) $\textsc{Tutte}^{01}(x,y)$ *requires time* $\exp(\Omega(m/\log^3 m))$ *if* $(x-1)(y-1) \notin \{0,1,2\}$ *and* $x \notin \{-1, 0\}$.
(iii) $\textsc{Tutte}^{01}(x,0)$ *requires time* $\exp(\Omega(m))$ *if* $x \in \{-2,-3,\ldots\}$.
(iv) $\textsc{Tutte}^{01}(x,0)$ *requires time* $\exp(\Omega(n))$ *if* $x \notin \{0,\pm 1\}$.

In an attempt to prove these results, we may first turn to the literature, which contains a cornucopia of constructions for proving hardness of the Tutte polynomial in various models. In these arguments, a central role is played by graph transformations called thickenings and stretches. A k-thickening replaces every edge by a bundle of k edges, and a k-stretch replaces every edge by a path of k edges. This is used do 'move' an evaluation from one point to another. For example, if H is the 2-stretch of G then $T(H;2,2) \sim T(G;4,\frac{4}{3})$. Thus, every algorithm for $(2,2)$ works also at $(4,\frac{4}{3})$, connecting the hardness of the two points. These reductions are very well-developed in the literature, and are used in models that are immune to polynomial-size changes in the input parameters, such as #P-hardness and approximation complexity. However, in the present paper we cannot always afford such constructions, otherwise our bounds would be of the form $\exp(\Omega(n^{1/r}))$ for some constant r depending on the blowup in the proof. In particular, the parameter n is destroyed already by a 2-stretch in a nonsparse graph.

The proofs are in §4–§6. Where we can, we sample from established methods, carefully avoiding or modifying those that are not parameter-preserving. At other times we require completely new ideas; the constructions in §5, which use Theta graph products instead of thickenings and stretches, may be of independent interest. Like many recent papers, we use Sokal's multivariate version of the Tutte polynomial, which vastly simplifies many of the technical details.

Consequences. The permanent and Tutte polynomial are equivalent to, or generalisations of, various other graph problems, so our lower bounds hold for these problems as well. In particular, it takes time $\exp(\Omega(m))$ to compute the following graph polynomials (for example, as a list of their coefficients) for a given simple graph: the Ising partition function, the q-state Potts partition function ($q \neq 0, 1, 2$), the reliability polynomial, the chromatic polynomial, and the flow polynomial. Moreover, we have $\exp(\Omega(n))$ lower bounds for the following counting problems on multigraphs: # perfect matchings, # cycle covers in digraphs, # connected spanning subgraphs, all-terminal graph reliability with given edge failure probability $p > 0$, # nowhere-zero k-flows ($k \neq 0, \pm 1$), and # acyclic orientations.

The lower bound for counting the number of perfect matchings holds even in bipartite graphs, where an $O(1.414^n)$ algorithm is given by Ryser's formula. Such algorithms are also known for general graphs [BH08], the current best bound is $O(1.619^n)$ [Koi09].

For simple graphs, we have $\exp(\Omega(m/\log m))$ lower bounds for # perfect matchings and # cycle covers in digraphs.

3 Hardness of the Permanent

This section contains the proof of Thm. 2. With $[0, n] = \{0, 1, \ldots, n\}$ we establish the reduction chain $\#3\text{-}\textsc{Sat} \preccurlyeq \textsc{Perm}^{-1,0,1} \preccurlyeq \textsc{Perm}^{[0,n]} \preccurlyeq \textsc{Perm}^{01}$ while taking care of the instance sizes.

Proof (of Thm. 2). First, to prove (i), we reduce $\#3\text{-}\textsc{Sat}$ in polynomial time to $\textsc{Perm}^{-1,0,1}$ such that 3-CNF formulas φ with m clauses are mapped to graphs G with $O(m)$ edges. For technical reasons, we preprocess φ such that every variable x occurs equally often as a positive literal and as a negative literal $\bar{x}$ (e.g., by adding trivial clauses of the form $(x \vee \bar{x} \vee \bar{x})$ to φ). We construct G with $O(m)$ edges and weights $w : E \to \{\pm 1\}$ such that $\#\textsc{Sat}(\varphi)$ can be derived from $\operatorname{per} G$ in polynomial time. For weighted graphs, the permanent is

$$\operatorname{per} G = \sum_{C \subseteq E} w(C)\,, \qquad \text{where } w(C) = \prod_{e \in C} w(e)\,.$$

The sum above is over all cycle covers C of G, that is, subgraphs (V, C) with an in- and outdegree of 1 at every vertex.

In Fig. 2, the gadgets of the construction are depicted. For every variable x that occurs in φ, we add a *selector gadget* to G. For every clause $c = \ell_1 \vee \ell_2 \vee \ell_3$ of φ, we add a *clause gadget* to G. Finally, we connect the edge labelled by a literal ℓ in the selector gadget with all occurrences of ℓ in the clause gadgets, using *equality gadgets*. This concludes the construction of G.

The number of edges of the resulting graph G is linear in the number of clauses. The correctness of the reduction follows along the lines of [Pap94] and [BD07]. The satisfying assignments stand in bijection to cycle covers of weight $(-1)^i 2^j$ where i (resp. j) is the number of occurrences of literals set to false (resp. true) by the assignment, and all other cycle covers sum up to 0. Since we preprocessed φ such that $i = j$, we obtain $\operatorname{per} G = (-2)^i \cdot \#\textsc{Sat}(\varphi)$.

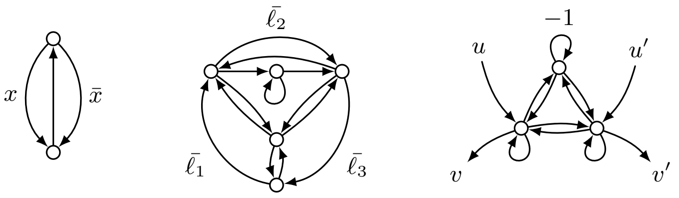


Fig. 2. Left: A selector gadget for variable x. Depending on which of the two cycles is chosen, we assume x to be set to true or false. Middle: A clause gadget for the clause $\ell_1 \vee \ell_2 \vee \ell_3$. The gadget allows all possible configurations for the outer edges, except for the case that all three are chosen (which would correspond to $\ell_1 = \ell_2 = \ell_3 = 0$). Right: An equality gadget that replaces two edges uv and $u'v'$. The top loop carries a weight of -1. It can be checked that the gadget contributes a weight of -1 if all four outer edges are taken, $+2$ if none of them is taken, and 0 otherwise.

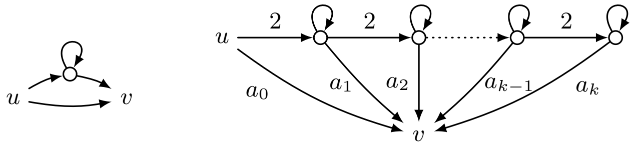


Fig. 3. Left: This gadget simulates in unweighted graphs edges uv of weight 2. Right: This gadget simulates edges uv of weight $a = \sum_{i=0}^{k} a_i 2^i$ with $a_i \in \{0, 1\}$.

To prove (ii), we reduce $\textsc{Perm}^{-1,0,1}$ in polynomial time to $\textsc{Perm}^{[0,n]}$ by interpolation: On input G, we conceptually replace all occurrences of the weight -1 by a variable x and call this new graph G_x. We can assume that only loops have weight x in G_x because the output graph G from the previous reduction has weight -1 only on loops. Then $p(x) = \operatorname{per} G_x$ is a polynomial of degree $d \leq n$.

If we replace x by a value $a \in [0, n]$, then G_a is a weighted graph with as many edges as G. As a consequence, we can use the oracle to compute $\operatorname{per} G_a$ for $a = 0, \ldots, d$ and then interpolate, to get the coefficients of the polynomial $p(x)$. At last, we return the value $p(-1) = \operatorname{per} G$. This completes the reduction, which queries the oracle $d + 1$ graphs that have at most m edges each.

For part (iii), we have to get rid of positive weights. Let G_a be one query of the last reduction. Again we assume that $a \leq n$ and that weights $\neq 1$ are only allowed at loop edges. We replace every edge of weight a by the gadget that is drawn in Fig. 3, and call this new unweighted graph G'. It can be checked easily that the gadget indeed simulates a weight of a (parallel paths correspond to addition, serial edges to multiplication), i.e., $\operatorname{per} G' = \operatorname{per} G_a$. Unfortunately, the reduction blows up the number of edges by a superconstant factor: The number of edges of G' is $m(G') \leq (m + n \log a) \leq O(m + n \log n)$. But since $m(G')/\log m(G') \leq O(m)$, the reduction shows that (iii) follows from (ii). ∎

These results immediately transfer to counting the number of perfect matchings in a graph even if the graph is restricted to be bipartite.

4 Hyperbolas in the Tutte plane

Our first goal will be to show that the Tutte polynomial is hard "for all hyperbolas" $(x-1)(y-1)=q$, except for $q=0$ (which we understand only partially), $q=1$ (which is in P), and for $q=2$ (which he handle separately in §6.3). From the hyperbolas, we will specialise the hardness result to individual points in the following sections.

4.1 The Multivariate Tutte Polynomial

We need Sokal's multivariate version of the Tutte polynomial, defined in [Sok05] as follows. Let $G=(V,E)$ be an undirected graph whose edge weights are given by a function $\mathbf{w}\colon E\to\mathbb{Q}$. Then

$$Z(G;q,\mathbf{w})=\sum_{A\subseteq E} q^{k(A)}\prod_{e\in A}\mathbf{w}(e)\,, \tag{3}$$

where $k(A)$ is the number of connected components in the subgraph (V,A). If $\mathbf{w}$ is single-valued, in the sense that $\mathbf{w}(e)=w$ for all $e\in E$, we slightly abuse notation and write $Z(G;q,w)$. With a single-valued weight function, the multivariate Tutte polynomial essentially equals the Tutte polynomial,

$$\begin{gathered}T(G;x,y)=(x-1)^{-k(E)}(y-1)^{-|V|}Z(G;q,w)\,,\\ \text{where } q=(x-1)(y-1) \text{ and } w=y-1\,,\end{gathered} \tag{4}$$

see [Sok05, eq. (2.26)]. The conceptual strength of the multivariate perspective is that it turns the Tutte polynomial's second variable y, suitably transformed, into an edge weight of the input graph. In particular, the multivariate formulation allows the graph to have different weights on different edges, which turns out to be a dramatic technical simplification even when, as in the present work, we are ultimately interested in the single-valued case.

Sokal's polynomial vanishes at $q=0$, so we will sometimes work with the polynomial

$$Z_0(G;q,\mathbf{w})=\sum_{A\subseteq E} q^{k(A)-k(E)}\prod_{e\in A}\mathbf{w}(e)\,,$$

which gives something non-trivial for $q=0$ and is otherwise a proxy for Z:

$$Z(G;q,\mathbf{w})=q^{k(E)}Z_0(G;q,\mathbf{w})\,. \tag{5}$$

4.2 Three-terminal minimum cut

We first establish that with two different edge weights, one of them negative, the multivariate Tutte polynomial computes the size of a 3-terminal minimum cut, for which we observe hardness under #ETH in Appendix B. This connection has been used already in [GJ07,GJ08], with different reductions, to prove hardness of approximation.

The graphs we will look at are connected and have rather simple weight functions. The edges are partitioned into two sets $E\dot{\cup}T$, and for fixed rational w the weight function is given by

$$\mathbf{w}(e) = \begin{cases} -1, & \text{if } e \in T, \\ w, & \text{if } e \in E. \end{cases} \tag{6}$$

For such a graph, we have

$$Z_0(G;q,\mathbf{w}) = \sum_{A\subseteq E\cup T} q^{k(A)-1} w^{|A\cap E|}(-1)^{|A\cap T|}. \tag{7}$$

For fixed G and q, this is a polynomial in w of degree at most m.

Lemma 1. *Let q be a rational number with $q \notin \{1,2\}$. Computing the coefficients of the polynomial $w \mapsto Z_0(G;q,\mathbf{w})$, with $\mathbf{w}$ as in* (6)*, for a given simple graph G requires time* $\exp(\Omega(m))$ *under #ETH.*

Moreover, this is true even if $|T| = 3$.

Proof. Suppose $G' = (V, E, t_1, t_2, t_3)$ is an instance of #3-Terminal MinCut with $n = |V|$ and $m = |E|$. As observed in App. B we can assume that G' is simple and connected. Modify G' by adding a triangle between the terminals, obtaining the graph $G = (V, E \cup T)$ where $T = \{t_1t_2, t_2t_3, t_1t_3\}$; note that $n(G) = n$, $m(G) = m + 3$, and $|T| = 3$.

We focus our attention on the family $\mathscr{A}$ of edge subsets $A \subseteq E$ for which t_1, t_2, and t_3 each belong to a distinct component in the graph (V, A). In other words, A belongs to $\mathscr{A}$ if and only if $E - A$ is a 3-terminal cut in G'. Then we can split the sum in (7) into

$$Z_0(G;q,\mathbf{w}) = \sum_{B\subseteq T}\left(\sum_{A\in\mathscr{A}} q^{k(A\cup B)-1}w^{|A|}(-1)^{|B|} + \sum_{A\notin\mathscr{A}} q^{k(A\cup B)-1}w^{|A|}(-1)^{|B|}\right). \tag{8}$$

We first show that the second term of (8) vanishes. Consider an edge subset $A \notin \mathscr{A}$ and assume without loss of generality that it connects the terminals t_1 and t_2. Consider $B \subseteq T$, and let $B' = B \oplus \{t_1t_2\}$, so that B' is the same as B except for t_1t_2. Then the contributions of $A \cup B$ and $A \cup B'$ cancel: First, $k(A \cup B)$ equals $k(A \cup B')$ because t_1 and t_2 are connected through A already, so the presence or absence of the edge t_1t_2 makes no difference. Second, $(-1)^{|B|}$ equals $-(-1)^{|B'|}$.

We proceed to simplify the first term of (8). The edges in B only ever connect vertices in T, and for $A \in \mathscr{A}$, each of these lies in a separate component of (V, A), so

$$k(A \cup B) = \begin{cases} k(A) - |B|\,, & \text{if } |B| = 0, 1, 2, \\ k(A) - 2\,, & \text{if } |B| = 3. \end{cases}$$

Calculating the contribution of B for each size $|B|$, we arrive at

$$\sum_{B \subseteq T} \sum_{A \in \mathscr{A}} q^{k(A\cup B)-1} w^{|A|} (-1)^{|B|} = \sum_{A \in \mathscr{A}} q^{k(A)-1} (q^0 - 3q^{-1} + 3q^{-2} - q^{-2}) w^{|A|}\,,$$

and after some simplification we can give (8) as

$$Z_0(G; q, \mathbf{w}) = Q \cdot \sum_{A \in \mathscr{A}} q^{k(A)-3} w^{|A|}\,, \quad \text{where } Q = (q-1)(q-2)\,. \tag{9}$$

Note that, by assumption on q, we have $Q \neq 0$.

Let us write $\sum_{i=0}^{m} d_i w^i = Q^{-1} Z_0(G; q, \mathbf{w})$, that is, d_i is the coefficient of the monomial w^i in the sum above. More specifically,

$$Q \cdot d_i = \sum_{A \in \mathscr{A}\colon\, |A| = i} q^{k(A)-3}\,.$$

The edge subsets $A \in \mathscr{A}$ are exactly the complements of the 3-terminal cuts in G'. Now consider the family $\mathscr{C}$ of *minimal* 3-terminal cuts, all of size c. The sets $E - A$ in $\mathscr{C}$ are exactly the sets A of size $m - c$ in $\mathscr{A}$, and by minimality, $k(A) = 3$. Thus,

$$Q \cdot d_{m-c} = \sum_{A \in \mathscr{A}\colon\, |A| = m-c} q^{3-3} = |\mathscr{C}|.$$

Thus, if we could compute the coefficients $d_0, \ldots, d_m$ of $w \mapsto Z_0(G; q, \mathbf{w})$, then we could determine the smallest c so that $d_{m-c} \neq 0$ and return $d_{m-c} = |\mathscr{C}|/Q$, the number of 3-terminal mincuts. ■

From this result, the argument continues in two directions. For simple graphs and certain parts of the Tutte plane, we proceed in §4.3 and §5. For *nonsimple* graphs and certain (other) parts of the Tutte plane, we can use just thickening and interpolation; see §6.

4.3 The Tutte polynomial along a hyperbola

To apply Lemma 1 to the Tutte polynomial, we need to get rid of the negative edges, so that the weight function is single-valued. In [GJ08], this is done by thickenings and stretches, which we need to avoid. However, since the number of negative edges is small (in fact, 3), we can use another tool, deletion–contraction. We will omit the case $q = 0$ from this analysis, because we won't need it later, so we can work with Z instead of Z_0.

A *deletion–contraction* identity expresses a function of the graph G in terms of two graphs $G - e$ and G/e, where

$G - e$ arises from G by *deleting* the edge e and
G/e arises from G by *contracting* the edge e, that is, deleting it and identifying its endpoints so that remaining edges between these two endpoints become loops.

It is known [Sok05, eq. (4.6)] that

$$Z(G;q,\mathbf{w}) = Z(G-e;q,\mathbf{w}) + \mathbf{w}(e)Z(G/e;q,\mathbf{w}). \tag{10}$$

Lemma 2. *Computing the coefficients of the polynomial $v \mapsto Z(G;q,v)$ for a given simple graph G requires time $\exp(\Omega(m))$ under #ETH, for all $q \notin \{0,1,2\}$.*

Proof. Let $G = (V,E)$ be a graph as in the previous lemma. Using (5) and 3 applications of deletion–contraction (10) we can write

$$Z_0(G;q,\mathbf{w})q^{k(E)} = Z(G;q,\mathbf{w}) = \sum_{C\subseteq\{1,2,3\}} (-1)^{|C|} Z(G_C;q,\mathbf{w}), \tag{11}$$

where for each $C \subseteq \{1,2,3\}$, the graph G_C is constructed from G by removing the three edges e_1, e_2, e_3 with weight -1 as follows: if $i \in C$ then e_i is contracted, otherwise it is deleted. In any case, the edges of T have disappeared and remaining edges of G_C are in one-to-one correspondence with the edges in E; especially, they all have the same weight w, so $Z(G_C;q,\mathbf{w}) = Z(G_C;q,w)$.

The resulting G_C are not necessarily simple, because the contracted edges from T may have been part of a triangle. (In fact, investigating the details of the reduction we can see that this is indeed the case.) Thus we construct the simple graph G'_C from G_C by subdividing every edge into a 2-path. This operation, known as a 2-stretch, is known to largely preserve the value of Z, in particular,

$$Z(G_C;q,w) = (q+2w)^m Z(G'_C;q,w'), \quad \text{where } 1+\frac{q}{w} = \Big(1+\frac{q}{w'}\Big)^2.$$

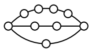
In summary, to compute $Z(G;q,\mathbf{w})$, we need to compute the 8 polynomials $v \mapsto Z(G_C;q,v)$, one for each G_C, and evaluate each at w'. We note that every G'_C is simple and has at most $n+m$ vertices and at most $2m$ edges. ■

5 Generalised Theta graphs

We now prove Thm. 3 (ii) by showing that most points (x,y) of the Tutte plane, are as hard as the entire hyperbola on which they lie, even for sparse, simple graphs. The drawback of our method is that we loose a polylogarithmic factor in the exponent of the lower bound and we do not get any results if $q := (x-1)(y-1) \in \{0,1,2\}$ or if $x \in \{-1,0\}$. However, the results are particularly interesting for the points on the line $y=-1$, for which we know no other good exponential lower bounds under #ETH, even in more general graph classes. We remark that the points $(-1,-1)$, $(0,-1)$, and $(\frac{1}{2},-1)$ on this line are known to admit a polynomial-time algorithm, and indeed our hardness result does not apply here. Also, since our technique does not work in the case $q=0$, the point $(1,-1)$ remains mysterious.

5.1 Generalised Theta graphs

For a set $S = \{s_1, \ldots, s_k\}$ of positive integers, the *generalised Theta graph Θ_S* consists of two vertices x and y joined by k internally disjoint paths of $s_1, \ldots, s_k$ edges, respectively. For example,

$\Theta_{\{2,3,5\}}$ is x  y .

For such a graph Θ_S, we will study the behaviour of the tensor product $G \otimes \Theta_S$ defined by Brylawski [Bry82] as follows: given $G = (V, E)$, replace every edge $xy \in E$ by (a fresh copy of) Θ_S. [5] What makes the $\otimes$-operation so useful in the study of Tutte polynomials is that the Tutte polynomial of $G \otimes H$ can be expressed in terms of the Tutte polynomials of G and H.

The Tutte polynomial of Theta graphs has been already studied by Sokal in the context of complex roots of the chromatic polynomial. The necessary formulas for $Z(G \otimes \Theta_S)$ can be derived from [Sok04, prop 2.2, prop 2.3]. We present them here for the special case where all edge weights are the same.[6]

Lemma 3 (Sokal). *Let q and w be rational numbers with $w \neq 0$ and $q \notin \{0, -2w\}$. Then, for all graphs G and finite sets S of positive integers,*

$$Z(G \otimes \Theta_S; q, w) = q^{|E|-|S|} \cdot \prod_{s \in S} \big((q+w)^s - w^s\big)^{|E|} \cdot Z(G; q, w_S)\,, \tag{12}$$

where

$$w_S = -1 + \prod_{s \in S} \left(1 + \frac{q}{(1+q/w)^s - 1}\right) . \tag{13}$$

5.2 Interpolation

Our plan is to compute the coefficients of the monovariate polynomial $w \mapsto Z(G; q, w)$ for given G and q by interpolation from sufficiently many evaluations of $Z(G; q, w_S) \sim Z(G \otimes \Theta_S; q, w)$. For this, we need that the number of different w_S is at least $|E| + 1$, one more than the degree of the polynomial.

[5] In the interest of formality we note there are two ways of doing this, because G is undirected. However, by virtue of our choice of second operand, the resulting graphs will be isomorphic no matter which orientation we choose, so we can dispense with some of the formalities normally needed at this point of definition. For general second operands, the resulting graphs need no be isomorphic, but the resulting Tutte polynomials turn out to be the same anyway; in fact, in any graph one can remove a maximal biconnected component and reinsert it in the other direction without changing the Tutte polynomial, an operation that is called the *Whitney twist.* The operation is more naturally defined for matroids, in which context it is sometimes called a tensor product. Note that the operation is not commutative; Sokal uses the notation $\boldsymbol{G^H}$.

[6] An alternative is to derive them more directly from Sokal's parallel and edge reduction rules, which is not difficult. Since all edges weights are the same, the result could also be established from the classical Tutte polynomial and the constructions in [JVW90], but this would probably be very laborious.

Lemma 4. *Let q and w be rational numbers with $w \neq 0$ and $q \notin \{0, -w, -2w\}$. For all integers $m \geq 1$, there exist sets $S_0, \ldots, S_m$ of positive integers such that*

(i) $\sum_{s \in S_i} s \leq O(\log^3 m)$ *for all i, and*
(ii) $w_{S_i} \neq w_{S_j}$ *for all $i \neq j$.*

Furthermore, the sets S_i can be computed in time polynomial in m.

Proof. Let $b = |1 + q/w|$ and $f(s) = 1 + q/(b^s - 1)$ for $s > 0$. Our choice of parameters ensures that $b > 0$ and $b \neq 1$, so f is a well-defined, continuous, and strictly monotone function from $\mathbb{R}^+ \to \mathbb{R}$. Furthermore, $w_S = -1 + \prod_{s \in S} f(s)$ for all positive even integers s. Now let $s_0 \geq 2$ be an even integer such that $f(s)$ is nonzero and has the same sign as $f(s_0)$ for all $s \geq s_0$. For $i = 0, \ldots, m$, let $b_\ell \cdots b_0$ denote the binary expansion of i where $\ell = \lfloor \log m \rfloor$. For a large constant gap parameter $\Delta > 6$ only depending on q and chosen later, define

$$S_i = \{\, s_0 + \Delta \lceil \log m \rceil \cdot (2j + b_j) \colon 0 \leq j \leq \ell \,\} .$$

The salient feature of this construction is that all sets S_i are different, of equal small cardinality, contain only positive even integers, and are from a range where f does not change sign. Most important for our analysis is that the elements of the S_i are spaced apart significantly, i.e.,

$$\text{for } i, j \text{ and any } s \in S_i \text{ and } t \in S_j, \text{ either } s = t \text{ or } |s - t| \geq \Delta \log m. \tag{P}$$

From $|S_i| = \lfloor \log m \rfloor + 1$ and the fact that all numbers in the sets are bounded by $O(\log^2 m)$, we immediately get (i).

To establish (ii), let $0 \leq i < j \leq m$. We want to show that $w_{S_i} \neq w_{S_j}$. Let us define $S = S_i \setminus S_j$ and $T = S_j \setminus S_i$. From (13), we see by multiplying with $(w_{S_i \cap S_j} + 1)$ on both sides that $w_S + 1 = w_T + 1$ is equivalent to $w_{S_i} = w_{S_j}$ since $w_{S_i \cap S_j} \neq -1$.

It remains to show that $\prod_{s \in S} f(s) \neq \prod_{t \in T} f(t)$. Equivalently,

$$\prod_{s \in S} \left(b^s + q - 1\right) \prod_{t \in T} \left(b^t - 1\right) - \prod_{t \in T} \left(b^t + q - 1\right) \prod_{s \in S} \left(b^s - 1\right) \neq 0 \tag{14}$$

We will factor out the products in (14). Using the notation $\|X\| = \sum_{x \in X} x$, we rewrite

$$\prod_{s \in S} \left(b^s + q - 1\right) \prod_{t \in T} \left(b^t - 1\right) = \sum_{X \subseteq S \cup T} (-1)^{|T \setminus X|} (q-1)^{|S \setminus X|} b^{\|X\|} .$$

Here we use the convention that for $X \subseteq S \cup T$, the term b^s is taken in the first factor if $s \in X \cap S$, and b^t is taken in the second factor if $t \in X \cap T$. Doing this for both terms of (14) and collecting terms we arrive at the equivalent claim

$$\sum_{X \subseteq S \cup T} g(X) \neq 0 \,, \tag{15}$$

where

$$g(X) = \Big((-1)^{|T\setminus X|}(q-1)^{|S\setminus X|} - (-1)^{|S\setminus X|}(q-1)^{|T\setminus X|}\Big) \cdot b^{\|X\|}\,. \tag{16}$$

Let s_1 be the smallest element of $S \cup T$ and without loss of generality assume that $s_1 \in S$ (otherwise exchange S and T). Now from (16) and $|S| = |T|$, it follows that

$$\begin{aligned} g\big(S \cup T\big) &= g(\emptyset) = 0 \\ g\big((S \cup T) \setminus \{s_1\}\big) &= q \cdot b^{\|S\cup T\| - s_1} \\ g\big(\{s_1\}\big) &= q(1-q)^{|S|-1} \cdot b^{s_1}\,. \end{aligned}$$

The largest exponent of b with nonzero coefficient in (16) is $\|S \cup T\| - s_1$ and all other exponents are at least $\Delta \log m$ smaller than that. Similarly, the smallest exponent of b with nonzero coefficient is s_1 and all other exponents are at least $\Delta \log m$ larger. We will let X_0 denote the term with the largest contribution in (15); so we set $X_0 = S \cup T \setminus \{s_1\}$ for $b > 1$ and $X_0 = \{s_1\}$ for $b < 1$.

The total contribution of the remaining terms is $h = \sum_{X \neq X_0} g(X)$. We will prove (15) by showing $|h| < |g(X_0)|$. From the triangle inequality and the fact that $S \cup T$ has at most $4m^2$ subsets X, we get

$$|h| \le 4m^2 \cdot \max_{X \neq X_0} |g(X)| \le 4m^2 \cdot 2|q-1|^{1+\log m} \cdot b^{\|X_0\| \pm \Delta \log m}$$

where the sign in $\pm\Delta \log m$ depends on whether b is larger or smaller than 1. If $b > 1$, the sign is negative. In this case, notice that $\Delta = \Delta(q)$ can be chosen so that $4m^2 \cdot 2|q-1|^{1+\log m} < |q| \cdot b^{\Delta \log m}$ for all $m \ge 2$. If $b < 1$, we can similarly choose Δ as to satisfy $4m^2 \cdot 2|q-1|^{1+\log m} < |q| \cdot |1-q|^{|S|-1} \cdot b^{-\Delta \log m}$. Thus, in both cases we have $|h| < |g(X_0)|$, establishing (ii). ■

We are ready to formalise the interpolation idea and prove Thm. 3 (ii).

Proof (of Thm. 3 (ii)). Let $(x,y) \in \mathbb{Q}^2$ with $q := (x-1)(y-1) \notin \{0,1,2\}$ and $x \notin \{-1,0\}$. We prove that $\text{Tutte}^{01}(x,y)$ requires time $\exp(\Omega(m/\log^3 m))$. Let G be a simple graph with n vertices and m edges. We will compute the coefficients of the polynomial $v \mapsto Z(G;q,v)$, which requires time $\exp(\Omega(m))$ by Lem. 2.

Set $w = y - 1$ and note $w \neq 0$ and $q \notin \{0, -w, -2w\}$. Construct the sets $S_0, \ldots, S_m$ as given by Lem. 4. For each $i = 0, \ldots, m$, construct the graph $G \otimes \Theta_{S_i}$ and compute $Z(G;q,w_{S_i})$ using (12). By Lem. 4 the points $w_{S_0}, \ldots, w_{S_m}$ are different. Moreover, as is clear from (3), the function $v \mapsto Z(G;q,v)$ is a polynomial of degree at most m, so we can use interpolation to recover its coefficients. The evaluations defined by (12) only ever evaluate simple graphs with $O(n \log^3 n)$ vertices and $O(m \log^3 m)$ edges at the point (q,w). ■

6 Interpolation and thickening

If we allow graphs to have multiple edges, we can use thickening and interpolation, one of the original strategies of [JVW90], for relocating the hardness result for hyperbolas from Lem. 2 to individual points in the Tutte plane. This gives us slightly better bounds in some points (but none at all on the line $y = -1$).

We recall the thickening identities for the multivariate Tutte polynomial. Assuming that there are only two edge weights α_1 and α_2, the weight function will be $\mathbf{w} : E \to \{\alpha_1, \alpha_2\}$. We denote by G_{k_1,k_2} the k_1, k_2*-thickening of* $(G, \mathbf{w})$ if all edges e of G with $\mathbf{w}(e) = \alpha_i$ have been replaced by k_i parallel edges. Let $\mathbf{w}_{k_1,k_2}$ be the weight function that a k_1, k_2-thickening from $\mathbf{w}$ 'moves' to. One can show [Sok05, (4.21)] that, with $\mathbf{w}_{k_1,k_2}(e) = (1 + \alpha_i)^{k_i} - 1$ whenever $\mathbf{w}(e) = \alpha_i$, it holds that

$$Z(G; q, \mathbf{w}_{k_1,k_2}) = Z(G_{k_1,k_2}; q, \mathbf{w})\,. \tag{17}$$

It is easy to transfer this result to the Tutte polynomial T using (4), yielding special cases of Brylawski's well-known graph transformation rules.

6.1 The multivariate Tutte polynomial along a hyperbola

It will be convenient to introduce the computational problem Sokal(q) for fixed q, which is to compute the coefficients of $Z_0(G; q, \mathbf{w})$, seen as a bivariate polynomial in the two edge weights $\mathbf{w}(e) \in \{\alpha_1, \alpha_2\}$.

Name Sokal(q).
Input $G = (V, E)$ and $\mathbf{w} : E \to \{\alpha_1, \alpha_2\}$ for *variables* α_1 and α_2.
Output The coefficients of the bivariate polynomial $\{\alpha_1, \alpha_2\} \mapsto Z_0(G; q, \mathbf{w})$.

Now we can state the most general interpolation process that we will need.

Lemma 5. *Let* $(x, y) \in \mathbb{Q}$ *with* $y \notin \{0, \pm 1\}$ *and let* $q = (x - 1)(y - 1)$*. There is a polynomial-time reduction from* Sokal(q) *to* Tutte(x, y) *that, on input a graph with* n *vertices and* m *edges, queries only graphs with* n *vertices and at most* m^2 *edges.*

Proof. Let $G = (V, E)$ and $\mathbf{w} : E \to \{\alpha_1, \alpha_2\}$ be the instance to Sokal(q). Let $p'(\alpha_1, \alpha_2) = Z_0(G; q, \mathbf{w})$, a polynomial in α_1 and α_2. For convenience, define $p(\alpha_1, \alpha_2) = p'(\alpha_1 - 1, \alpha_2 - 1)$. The maximum degree of p is at most m.

By abuse of notation, let $y - 1$ also denote the constant function $E \to \{y - 1\}$. Clearly, $Z_0(G; q, y - 1) = p(y, y)$. By (4), it holds $Z_0(G'; q, y - 1) = (y - 1)^{|V| - k(E)} T(G'; x, y)$ for all graphs G'.

Using (17) for $(k_1, k_2) \in \mathbb{N}^2$, we can 'shift' the point (y, y) to (y^{k_1}, y^{k_2}), that is, we get identities $p(y^{k_1}, y^{k_2}) = Z_0(G_{k_1,k_2}; q, y - 1)$. We see these identities as a linear equation system

$$Y \cdot \begin{pmatrix} c_{0,0} \\ \vdots \\ c_{i,j} \\ \vdots \\ c_{m,m} \end{pmatrix} = \begin{pmatrix} p\big(y^0, y^0\big) \\ \vdots \\ p\big(y^{k_1}, y^{k_2}\big) \\ \vdots \\ p\big(y^m, y^m\big) \end{pmatrix} ,$$

where Y is the $(m+1)^2 \times (m+1)^2$ bivariate Vandermonde matrix with

$$Y_{(i,j),(k_1,k_2)} = y^{ik_1+jk_2}\,,$$

for all $(k_1, k_2) \in \{0, \ldots, m\}^2$ and $(i, j) \in \{0, \ldots, m\}^2$. The solution $c_{i,j}$ of the equation system are the coefficients of the polynomial $p(\alpha_1, \alpha_2) = \sum_{i,j} c_{i,j}\alpha_1^i \alpha_2^j$. Since $y \notin \{0, \pm 1\}$, the points (y^{k_1}, y^{k_2}) are all pairwise distinct. The bivariate Vandermonde matrix is non-singular since it is the Kronecker product $Y = Y_1 \otimes Y_1$ of two univariate non-singular Vandermonde matrices Y_1,

$$Y_1 = \begin{pmatrix} (y^0)^0 & \ldots & (y^0)^i & \ldots & (y^0)^m \\ \vdots & & \vdots & & \vdots \\ (y^k)^0 & \ldots & (y^k)^i & \ldots & (y^k)^m \\ \vdots & & \vdots & & \vdots \\ (y^m)^0 & \ldots & (y^m)^i & \ldots & (y^m)^m \end{pmatrix}.$$

We interpolate the polynomial p by solving for its coefficients in the above linear equation system. For this, we make $(m+1)^2$ queries to the oracle $\text{TUTTE}(x, y)$ and each query graph G_{k_1,k_2} has n vertices and at most m^2 edges. ∎

6.2 General component weights

Here we consider the case where the component weight q is neither 1 nor 2. This establishes Thm. 3 (i) except for $q = 2$, which is handled in §6.3.

Proof (of Thm. 3 (i) for $q \neq 2$). Let $(x, y) \in \mathbb{Q}^2$ with $y \notin \{0, \pm 1\}$ and $q := (x-1)(y-1) \notin \{1, 2\}$. We show that $\text{TUTTE}(x, y)$ requires time $\exp(\Omega(n))$ under #ETH.

We start from Lem. 1. Let $G = (V, E \cup T)$ be a graph with edge weights as given in (6). We compute $w \rightarrow Z_0(G; q, \mathbf{w})$ using an algorithm for $\text{SOKAL}(q)$ on the instance $G = (V, E \cup T)$ and weights $\mathbf{w}'$ given by $\mathbf{w}'(e) = \alpha_1 = w$ $(e \in E)$ and $\mathbf{w}'(e) = \alpha_2 = -1$ $(e \in T)$. Thus, $\text{SOKAL}(q)$ requires time $\exp(\Omega(m))$ under #ETH, and the result now follows from the interpolation Lemma 5. ∎

6.3 From the Permanent to the Ising hyperbola

Now we reduce the permanent to $\text{SOKAL}(2)$, the multivariate Tutte polynomial at $q = 2$, and from there to $\text{TUTTE}(x, y)$ with $(x-1)(y-1) = 2$.

Proof (of Thm. 3 (i) for $q = 2$). Let $(x, y) \in \mathbb{Q}^2$ with $q := (x-1)(y-1) = 2$. We show that $\text{TUTTE}(x, y)$ requires time $\exp(\Omega(n))$ under #ETH. For this, we establish a chain of reductions $\text{PERM} \preccurlyeq \text{SOKAL}(q) \preccurlyeq \text{TUTTE}(x, y)$ in which the number of vertices is increased by no more than a constant factor. Then the desired result follows from Thm. 2 (ii). The first reduction follows the argument sketched in [GJ08, Lemma 16], which we give in detail below for completeness.

The second reduction is Lemma 5, observing that the lemma's condition $y \notin \{-1, 0, 1\}$ holds because $q = 2$ and (x, y) is non-special.

We turn to the first reduction. Let A be an $n \times n$ matrix with nonnegative integer entries a_{ij}, of which m are nonzero. It is useful to view $\operatorname{per} A$ as the number of perfect matchings of the bipartite multigraph $G = (V, E)$ with vertex set $V = \{1, \ldots, 2n\}$ and a_{ij} edges from i to $n + j$.

Construct $G' = (V', E')$ by setting $V' = V \cup \{t\}$ and $E' = E \cup T$, where $T = \{\, tv \colon v \in V \,\}$. Clearly $n(G') = n(G) + 1$ and $m(G') = n(G) + m(G)$. Define the edge weight function $\mathbf{w} \colon E' \to \{\alpha, -2\}$ by $\mathbf{w}(e) = \alpha$ if $e \in E$ and $\mathbf{w}(e) = -2$ if $e \in T$. Now we have the following:

$$\begin{aligned} Z(G'; 2, \mathbf{w}) &= \sum_{A \subseteq E'} 2^{k(A)} \prod_{e \in A} \mathbf{w}(e) \\ &= \sum_{B \subseteq E} \sum_{C \subseteq T} 2^{k(B \cup C)} \prod_{e \in B \cup C} \mathbf{w}(e) = \sum_{B \subseteq E} h(B) \alpha^{|B|} \,, \end{aligned}$$

where

$$h(B) = \sum_{C \subseteq T} (-2)^{|C|} 2^{k(B \cup C)} \,.$$

A graph is *bridge connected* if it is connected and does not contain any bridge. In [GJ08, Proof of Lemma 16] it is shown that

1. Let $|B| = n$. If $(V', T \cup B)$ is bridge connected, then $h(B) = 2^{n+1}$, and $h(B) = 0$, otherwise.
2. The set $\{\, B \colon |B| = n$ and $(V', T \cup B)$ is bridge connected$\,\}$ is in 1–1 correspondence with the set of perfect matchings of G. Specifically, $(V', T \cup B)$ is bridge connected if and only if B is a perfect matching in G.

In particular, $|B|$ equals the number of perfect matchings of G, and hence the permanent of A. Thus, we can compute

$$\operatorname{per} A = 2^{-(n+1)} c_n = 2^{-(n+1)} \sum_{B \subseteq E \colon |B| = n} h(B)$$

if we know the coefficients of the polynomial $p(\alpha) = \sum_i c_i \alpha^i := Z(G'; 2, \mathbf{w})$. This gives the desired reduction from PERM to SOKAL(2). ■

7 Linial's Reduction Along the x-Axis

The chromatic polynomial $\chi(G; q)$ of G is the polynomial in q with the property that, for all $c \in \mathbb{N}$, the evaluation $\chi(G; c)$ is the number of proper c-colourings of the vertices of G. We write $\chi(q)$ for the function $G \mapsto \chi(G; q)$. It is known that the Tutte polynomial specializes to the chromatic polynomial for $y = 0$:

$$\chi(G; q) = (-1)^{n(G) - k(G)} q^{k(G)} T(G; 1 - q, 0) \,. \tag{18}$$

The case $y = 0$ cannot be handled by the proof in §6.2 because thickenings do not produce enough points for the interpolation. Instead, we use Linial's reduction [Lin86] for this line.

Proof (of Thm. 3 (iii) and (iv)). Let $x \in \mathbb{Q}$ and $q = 1 - x$. We use (18) to see that evaluating the chromatic polynomial $\chi(q)$ is equivalent to evaluating $\textsc{Tutte}(x, 0)$ if $q \neq 0$. Since $\chi(3)$ is the number of 3-colourings, the case $q = 3$ requires time $\exp(\Omega(m))$ under #ETH, even for simple graphs (cf. App. B). For $i \in \{1, 2, \ldots\}$ and all real r, Linial's identity is

$$\chi(G + K_i; r) = r(r-1)\ldots(r-i+1) \cdot \chi(G; r-i)\,, \tag{19}$$

where $G + K_i$ is the simple graph consisting of G and a clique K_i on i vertices, each of which is adjacent to every vertex of G.

For $q \in \{4, 5, \ldots\}$, we can set $i = q - 3$ and directly compute $\chi(G; 3) = \chi(G; q-i) = \chi(G+K_i; q)/[q(q-1)\cdots 4]$. Since $m(G+K_i) = m(G)+i\cdot n(G)+\binom{i}{2} \leq O(m(G))$, it follows that $\chi(q)$ requires time $\exp(\Omega(m))$ under #ETH, even for simple graphs. This establishes part (iii) of the theorem.

For part (iv), let $x \notin \{1, 0, -1, -2, -3, \ldots\}$. We show that $\textsc{Tutte}^{01}(x, 0)$ requires time $\exp(\Omega(n))$ under #ETH. Indeed, with access to $\chi(q)$, we can compute $\chi(G; q-i)$ for all $i = 0, \ldots, n$, noting that all prefactors in (19) nonzero. From these $n + 1$ values, we interpolate to get the coefficients of the polynomial $r \mapsto \chi(G; r)$, which in turn allows us evaluate $\chi(G; 3)$. In this case, the size of the oracle queries depends nonlinearly on the size of G, in particular $m(G+K_n) \sim n^2$. However, the number of vertices is $n(G+K_i) \leq 2n \leq O(m(G))$. Thus, since $\chi(3)$ requires time $\exp(\Omega(n))$ under #ETH, this also holds for $\chi(q)$, even for simple graphs. ■

The only points on the x-axis not covered here are $q \in \{0, 1, 2\}$, corresponding to $x \in \{1, 0, -1\}$. Two of these admit polynomial time algorithms, so we expect no lower bound to hold; the exponential complexity of $(1, 0)$ is left open under #ETH.

8 Conclusion

Our results leave open a number of cases. For the Tutte polynomial of multigraphs, we have left the line $y = 1$ and the points $(1, 0)$ and $(1, -1)$ on the plane unexplored. The line corresponds to forest counting $T(G; z+1, 1) \sim \sum_{\text{forests } F} z^{|V|-|F|}$ and the point $(1, 0)$ is the intersection of the reliability polynomial $x = 1$ and the chromatic polynomial $y = 0$. For simple graphs, the two lines $x = -1, 0$ and the two hyperbolas $(x-1)(y-1) = 0, 2$ are open, and for most of the plane our lower bound of $\exp(\Omega(n/\log^3 n))$ does not quite match the upper bound of $\exp(O(n))$. Furthermore, closing the gap between upper and lower bound for the permanent of $(n \times n)$-matrices with entries 0 and 1 seems to be particularly interesting.

Acknowledgements

The authors are grateful to Leslie Ann Goldberg and Andreas Björklund for valuable comments.

References

[Agr06] Manindra Agrawal. Determinant versus permanent. In *Proceedings of the 25th International Congress of Mathematicians, ICM 2006*, volume 3, pages 985–997, 2006.

[BD07] Markus Bläser and Holger Dell. Complexity of the cover polynomial. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming, ICALP 2007*, volume 4596 of *Lecture Notes in Computer Science*, pages 801–812. Springer, 2007.

[Ber84] Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, 1984.

[BH08] Andreas Björklund and Thore Husfeldt. Exact algorithms for exact satisfiability and number of perfect matchings. *Algorithmica*, 52(2):226–249, 2008.

[BHKK08] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Computing the Tutte polynomial in vertex-exponential time. In *Proceedings of the 47th annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pages 677–686, 2008.

[Bry82] Thomas Brylawski. The Tutte polynomial, Matroid theory and its applications. *Centro Internazionale Matematico Estivo*, pages 125–275, 1982.

[DJP$^+$94] Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Springer, 2006.

[GHN06] Omer Giménez, Petr Hliněný, and Marc Noy. Computing the Tutte polynomial on graphs of bounded clique-width. *SIAM Journal on Discrete Mathematics*, 20:932–946, 2006.

[GJ07] Leslie Ann Goldberg and Mark Jerrum. The complexity of ferromagnetic Ising with local fields. *Combinatorics, Probability and Computing*, 16(1):43–61, 2007.

[GJ08] Leslie Ann Goldberg and Mark Jerrum. Inapproximability of the Tutte polynomial. *Information and Computation*, 206(7):908–929, 2008.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[Ist00] Sorin Istrail. Statistical mechanics, three-dimensionality and NP-completeness. I. universality of intractability for the partition function of the Ising model across non-planar lattices. In *Proceedings of the 32nd annual ACM Symposium on Theory of Computing, STOC 2000*, pages 87–96. ACM, 2000.

[JS82] Mark Jerrum and Marc Snir. Some exact complexity results for straight-line computations over semirings. *Journal of the ACM*, 29(3):874–897, 1982.

[JVW90] François Jaeger, Dirk L. Vertigan, and Dominic J.A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Mathematical proceedings of the Cambridge Philosophical Society*, 108(1):35–53, 1990.

[Koi09] Mikko Koivisto. Partitioning into sets of bounded cardinality. In *Proceedings of the 7th International Workshop on Parameterized and Exact Complexity, IWPEC 2009*, pages 258–263, 2009.

[Kut07] Konstantin Kutzkov. New upper bound for the #3-SAT problem. *Information Processing Letters*, 105(1):1–5, 2007.

[Law76] Eugene L. Lawler. A note on the complexity of the chromatic number problem. *Information Processing Letters*, 5:66–67, 1976.

[Lin86] Nathan Linial. Hard enumeration problems in geometry and combinatorics. *SIAM Journal on Algebraic and Discrete Methods*, 7(2):331–335, 1986.

[Pap94] Christos M. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, Massachusetts, 1994.

[Raz09] Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *Journal of the ACM*, 56(2):1–17, 2009.

[Rys63] Herbert J. Ryser. Combinatorial mathematics. *Number 14 in Carus Math. Monographs. Mathematical Association of America*, 1963.

[SIT95] Kyoko Sekine, Hiroshi Imai, and Seiichiro Tani. Computing the Tutte polynomial of a graph of moderate size. In *Proceedings of the 6th International Symposium on Algorithms and Computation, ISAAC 1995*, number 1004 in Lecture Notes in Computer Science, pages 224–233. Springer, 1995.

[Sok04] Alan D. Sokal. Chromatic roots are dense in the whole complex plane. *Combinatorics, Probability and Computing*, 13(02):221–261, 2004.

[Sok05] Alan D. Sokal. The multivariate Tutte polynomial (alias Potts model) for graphs and matroids. In *Surveys in Combinatorics*, volume 327 of *London Mathematical Society Lecture Note Series*, pages 173–226. Cambridge University Press, 2005.

[Val79] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

A The Counting Exponential Time Hypothesis

Sparsification is the process of reducing the density of graphs, formulas, or other combinatorial objects, while some properties of the objects like the answer to a computational problem are be preserved.

The objective of sparsification is twofold. From an algorithmic perspective, efficient sparsification procedures can be used as a preprocessing step to make input instances sparse and thus possibly simpler and smaller, such that only the core information about the input remains. In the literature, such applications of sparsification procedures are called kernelization or (lossy) compression. From a complexity theoretic point of view, sparsification is a tool to identify those instances of a problem that are computationally the hardest. If an NP-hard problem admits efficient sparsification, the hardest instances are sparse.

In the context of the exponential time hypothesis, the sparsification lemma provides a way to show that the hardest instances of d-Sat are sparse and thus the parameter n can be replaced with m in the statement of the exponential time hypothesis. The following is the sparsification lemma as formulated in [FG06, Lemma 16.17].

Lemma 6 (Sparsification Lemma). *Let $d \geq 2$. There exists a computable function $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ such that for every $k \in \mathbb{N}$ and every d-CNF formula γ with n variables, we can find a formula*

$$\beta = \bigvee_{i \in [t]} \gamma_i$$

such that:

(1) β is equivalent to γ,
(2) $t \leq 2^{n/k}$ and
(3) each γ_i is a subformula of γ in which each variable occurs at most $f(d,k)$ times.

Furthermore, β can be computed from γ and k in time $t \cdot \mathrm{poly}(n)$.

We sketch below a small modification in the proof of the sparsification lemma that allows us to replace (1) with the condition

(1') $\mathrm{sat}(\gamma) = \dot{\bigcup}_i \mathrm{sat}(\gamma_i)\,,$

where $\mathrm{sat}(\varphi)$ is the set of assignments that satisfy the formula φ. In particular, (1') implies $\#\textsc{Sat}(\gamma) = \sum_i \#\textsc{Sat}(\gamma_i)$, which means that the sparsification lemma can be used for the counting version of 3-Sat.

Proof (sketch). We adapt the terminology of [FG06, Proof of Lemma 16.17] and we follow their construction precisely, except for a small change in the sparsification algorithm. When the algorithm decides to branch for a CNF-formula γ

and a flower $\alpha = \{\delta_1, \dots, \delta_p\}$, the original algorithm would branch on the two formulas

$$\begin{aligned}\gamma^\alpha_{\text{heart}} &= \gamma \setminus \{\delta_1, \dots, \delta_p\} \cup \{\delta\}\,,\\ \gamma^\alpha_{\text{petals}} &= \gamma \setminus \{\delta_1, \dots, \delta_p\} \cup \{\delta_1 \setminus \delta, \dots, \delta_p \setminus \delta\}\,.\end{aligned}$$

We modify the branching on the petals to read

$$\gamma^\alpha_{\text{petals}} = \gamma \setminus \{\delta_1, \dots, \delta_p\} \cup \{\delta_1 \setminus \delta, \dots, \delta_p \setminus \delta\} \cup \big\{\ \{\neg l\} : l \in \delta\ \big\}\,.$$

This way, the satisfying assignments become disjoint: In each branching step, we guess whether the heart contains a literal set to true, or whether all literals in the heart are set to false and each petal contains a literals set to true.

Now we have that, for all CNF-formulas γ, all assignments σ to the variables of γ, and all flowers α of γ,

(i) σ satisfies γ if and only if σ satisfies $\gamma^\alpha_{\text{heart}} \vee \gamma^\alpha_{\text{petals}}$, and
(ii) σ does not satisfy $\gamma^\alpha_{\text{heart}}$ or σ does not satisfy $\gamma^\alpha_{\text{petals}}$.

By induction, we see that at the end of the algorithm,

(i) σ satisfies γ if and only if σ satisfies some γ_i, and
(ii) σ satisfies at most one γ_i.

This implies that $\text{sat}(\gamma) = \dot{\bigcup}_{i \in [t]} \text{sat}(\gamma_i)$.

Notice that our new construction adds at most n clauses of size 1 to the formulas γ_i compared to the old one. Furthermore, our construction does not make t any larger because the REDUCE-step removes all clauses that properly contain $\{\neg l\}$ and thus these unit clauses never appear in a flower. ∎

Proof (of Thm. 1). For all integers $d \geq 3$ and $k \geq 1$, the sparsification lemma gives an oracle reduction from #d-SAT to #d-SAT that, on input a formula γ with n variables, only queries formulas with $m' = O(n)$ clauses, such that the reduction runs in time $\exp(O(n/k))$. Now, if for every $c > 0$ there is an algorithm for #d-SAT that runs in time $\exp(cm)$, we can combine this algorithm and the above oracle reduction to obtain an algorithm for #d-SAT that runs in time $\exp(O(n/k) + c \cdot m') = \exp(O(n/k) + c \cdot O(n))$. Since this holds for all small $c > 0$ and large k, we have for every $c' > 0$ an algorithm for #d-SAT running in time $\exp(c' \cdot n)$. This proves that for all $d \geq 3$, #d-SAT can be solved in variable-subexponential time if and only if it can be solved in clause-subexponential time.

To establish the equivalence between different d's, we transform an instance φ of #d-SAT into an instance φ' of #3-SAT that has the same number of satisfying assignments. The formula φ' is constructed as in the standard width-reduction for d-CNF formulas, i.e., by introducing a constant number of new variables for every clause of φ. Thus, since the number of clauses of φ' is $O(m)$, any clause-subexponential algorithm for #3-SAT implies a clause-subexponential algorithm for #d-SAT. The converse of this implication is trivial. ∎

B Hardness of 3-Colouring and 3-Terminal MinCut

The purpose of this part is to show that the standard reductions from 3-Sat to the computational problems 3-Colouring, NAE-Sat, MaxCut, and 3-Terminal MinCut already preserve the number of solutions and increase the number of clauses or edges of the instances by at most a constant factor. This then implies that the corresponding counting problems require time exponential in the number of clauses or edges unless #ETH fails.

Theorem 4. *Deterministically computing the problems* #NAE-Sat*,* #Max-Cut*, and* #3-Terminal MinCut *requires time* $\exp(\Omega(m))$ *unless #ETH fails.*

In the following, we formally define the problems, sketch the standard NP-hardness reductions, and provide their analyses as needed to proof Thm. 4.

Name #NAE-Sat
Input 3-CNF formula φ.
Output The number of truth assignments, so that no clause $\{a, b, c\} \in \varphi$ contains only literals with the same truth value.

Lemma 7. *There is a polynomial-time mapping reduction from* #3-Sat *to* #NAE-Sat *that maps formulas with* m *clauses to formulas with* $O(m)$ *clauses.*

Proof. Let φ be a 3-CNF formula with n variables and m clauses. To construct the instance φ' to NAE-Sat, we first replace every trivariate clause $(a \vee b \vee c)$ with the clauses

$$(x \vee \overline{a}) \wedge (x \vee \overline{b}) \wedge (\overline{x} \vee a \vee b) \wedge (x \vee c).$$

where x is a fresh variable. These clauses force x to have the value of $a \vee b$ in a satisfying assignment. It can be checked that these clauses are satisfied exactly if the original clause was satisfied and moreover that the trivariate clause is never all-false or all-true. In total, we increase the number of clauses four-fold.

Finally, introduce yet another fresh variable z and add this single variable (positively) to every mono- and bivariate clause. It is well-known that this modification turns the instance into an instance of NAE-Sat (see [Pap94, Theorem 3]). The resulting instance φ'' has $4m$ clauses and #NAE-Sat(φ'') = #3-Sat(φ). ∎

Name #3-Colouring
Input Simple undirected graph G.
Output The number of proper vertex-colourings with three colours.

Lemma 8. *There is a polynomial-time mapping reduction from* #NAE-Sat *to* #3-Colouring *that maps formulas with* m *clauses to graphs with* $O(m)$ *edges.*

Proof. The hardness of 3-Colouring under ETH is already observed in [IPZ01] but without mentioning that it even holds if we use m instead of n to measure the size of the instance.

The graph G that is constructed in [Pap94, Theorem 9.8] from an NAE-Sat-instance φ with n variables and m clauses has $n' = 1 + 2n + 3m$ vertices and $m' = 3n + 6m$ edges. Furthermore the number of proper 3-colourings is equal to #NAE-Sat(φ). ∎

Name #MaxCut
Input Simple undirected graph G.
Output The number of maximum cuts.

Lemma 9. *There is a polynomial-time mapping reduction from* #NAE-Sat *to* #MaxCut *that maps formulas with m clauses to graphs with $O(m)$ edges.*

Proof. We follow the reduction in [Pap94, Theorem 9.5]. Given an instance to NAE-Sat with n variables and m clauses, the reduction produces an instance to MaxCut, a graph with $2n$ vertices and at most $3m + 3m = 6m$ edges. Furthermore, the number of solutions is equal. ∎

Name #3-Terminal MinCut
Input Simple undirected graph $G = (V, E)$ with three distinguished vertices ("terminals") $t_1, t_2, t_3 \in V$.
Output The number of cuts of minimal size that separate t_1 from t_2, t_2 from t_3, and t_3 from t_1.

Lemma 10. *There is a polynomial-time mapping reduction from* #MaxCut *to* #3-Terminal MinCut *that maps graphs with m edges to graphs with $O(m)$ edges.*

Proof. We follow the reduction in [DJP$^+$94]. So let $G = (V, E)$ be a graph with n vertices and m edges. It is made explicit in [DJP$^+$94] that the construction builds a graph F with $n' = 3+n+4m = O(m)$ vertices. For the number of edges, every $uv \in E$ results in a gadget graph C with 18 edges, so the number of edges in F is $18m = O(m)$. The construction is such that the number of minimum 3-terminal cuts of F equals the number of maximum cuts of G. ∎

Proof (of Thm. 4). Assume one of the problems can be solved in time $\exp(cm)$ for every $c > 0$. Then 3-Sat can be solved by first applying the applicable reductions of the preceding lemmas and then invoking the assumed algorithm. This gives for every $c > 0$ an algorithm for 3-Sat that runs in time $\exp(O(cm))$, which implies that #ETH fails. ∎