Exponential Time Complexity of the Permanent and the Tutte Polynomial
(Extended Abstract)

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

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

S. Abramsky et al. (Eds.): ICALP 2010, Part I, LNCS 6198, pp. 426–437, 2010.

famous formula [19] computes the permanent with only $\exp(O(n))$ arithmetic operations, and more recently, an algorithm with running time $\exp(O(n))$ for n-vertex graphs has also been found [4] for the Tutte polynomial. Curiously, both of these algorithms are based on the inclusion–exclusion principle. 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 $\exp(O(\sqrt{n}))$ [20]. 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 $\{1, \dots, n\}$. 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 [23] 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 [13] 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 [18] 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 [2]. One of the easy consequences of the present results 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 [1]; 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 result is 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".[1]

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 [12], 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 $(1, 1), (-1, -1), (0, -1), (-1, 0)$.

We give an $\exp(\Omega(n))$ lower bound that matches the $\exp(O(n))$ algorithm from [4] and 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 [20]. 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.

[1] 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 [11].

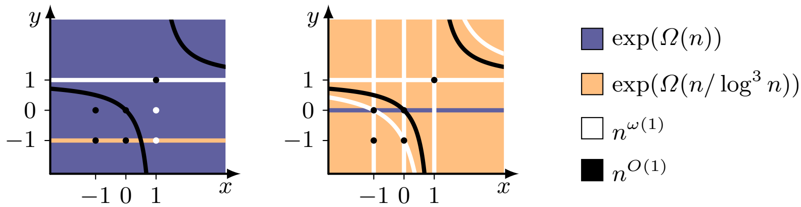


Fig. 1. Exponential time complexity under #ETH of the Tutte plane for multigraphs (left) and simple graphs (right) in terms of n, the number of vertices. White areas on the map correspond to uncharted territory. The black hyperbola $(x-1)(y-1) = 1$ and the four points close to the origin are in P. Everywhere 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 [16] 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 [10]. 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 [7] 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 [9], 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 [4].

2 Results

The exponential time hypothesis (ETH) as defined in [10] 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 $O(1.6423^n)$ [15]. 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, [10] 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 full paper, 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 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 $\text{per}(a_{ij})$ 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 is in §3. For (i), we follow a standard reduction by Valiant [23,17] but use a simple equality gadget derived from [5] 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 $\text{Tutte}(x, y)$ is defined for each pair (x, y) of rationals.

Name. $\text{Tutte}(x, y)$.
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 $\text{Tutte}^{01}(x, y)$ for the special case where the input graph is simple.

Our main result is that under #ETH, 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 [4] 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) Tutte(x, y) *requires time* $\exp(\Omega(n))$ *if* $(x-1)(y-1) \neq 1$ *and* $y \notin \{0, \pm 1\}$.
(ii) 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) Tutte$^{01}(x, 0)$ *requires time* $\exp(\Omega(m))$ *if* $x \in \{-2, -3, \ldots\}$.
(iv) 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 order to establish our exponential hardness results, 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 omitted, though we sketch the construction involved in the proof of Thm. 3 (ii, which may be of independent interest. 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 [3], the current best bound is $O(1.619^n)$ [14].

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 $[0, n] = \{0, 1, \dots, n\}$ we establish the reduction chain #3-Sat $\preccurlyeq$ $\text{Perm}^{-1,0,1}$ $\preccurlyeq$ $\text{Perm}^{[0,n]}$ $\preccurlyeq$ Perm^{01} while taking care of the instance sizes.

Proof (of Thm. 2). First, to prove (i), we reduce #3-Sat in polynomial time to $\text{Perm}^{-1,0,1}$ 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 #Sat(φ) can be derived from 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 [17] and [5]. 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 $\operatorname{per} G = (-2)^i \cdot \#\text{Sat}(\varphi)$.

To prove (ii), we reduce $\text{Perm}^{-1,0,1}$ in polynomial time to $\text{Perm}^{[0,n]}$ by interpolation: On input G, we conceptually replace all occurrences of the weight -1

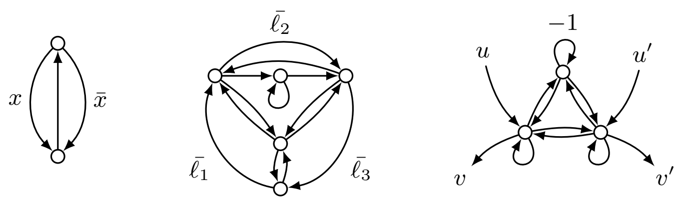


Fig. 2. Left: A selector gadget for variable x. Depending on which of the two cycles is chosen, we assume x to be set to true or false. Middle: A clause gadget for the clause $\ell_1 \vee \ell_2 \vee \ell_3$. The gadget allows all possible configurations for the outer edges, except for the case that 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.

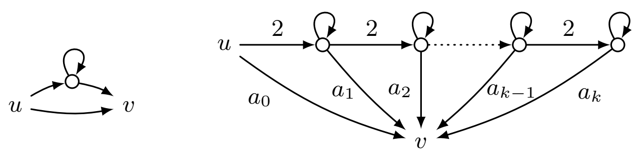


Fig. 3. Left: This gadget simulates in unweighted graphs edges uv of weight 2. Right: This gadget simulates edges uv of weight $a = \sum_{i=0}^{k} a_i 2^i$ with $a_i \in \{0, 1\}$.

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 the full paper by a reduction from the permanent). 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 [22] as follows. Let $G = (V, E)$ be an undirected graph whose edge weights are given by a function $\mathbf{w}: E \to \mathbb{Q}$. 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 [22, 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 the full paper. This connection has been used already in [8,9], 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 $|T| = 3$.

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, which we lay out in the full paper.

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 [9], 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 [22, eq. (4.6)] that $Z(G; q, \mathbf{w}) = Z(G - e; q, \mathbf{w}) + \mathbf{w}(e) Z(G/e; q, \mathbf{w})$.

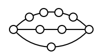
Lemma 2. *Computing the coefficients of the polynomial $v \mapsto Z(G; q, v)$ for a given simple graph G requires time* $\exp(\Omega(m))$ *under #ETH, for all $q \notin \{0, 1, 2\}$.*

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.

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 [6] as follows: given $G=(V,E)$, replace every edge $xy \in E$ by (a fresh copy of) Θ_S. 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, as studied by Sokal. The necessary formulas for $Z(G\otimes\Theta_S)$ can be derived from [21, prop 2.2, prop 2.3]. We present them here for the special case where all edge weights are the same.

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,*

$$Z(G\otimes\Theta_S;q,w) = q^{|E|-|S|}\cdot\prod_{s\in S}\big((q+w)^s - w^s\big)^{|E|}\cdot Z(G;q,w_S)\,, \tag{8}$$

where

$$w_S = -1 + \prod_{s\in S}\left(1+\frac{q}{(1+q/w)^s-1}\right)\,. \tag{9}$$

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.

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.

This lemma together with interpolation establishes Thm. 3 (ii).

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

References

1. Agrawal, M.: Determinant versus permanent. In: Proceedings of the 25th International Congress of Mathematicians, ICM, vol. 3, pp. 985–997 (2006)
2. Berkowitz, S.J.: On computing the determinant in small parallel time using a small number of processors. Information Processing Letters 18(3), 147–150 (1984)
3. Björklund, A., Husfeldt, T.: Exact algorithms for exact satisfiability and number of perfect matchings. Algorithmica 52(2), 226–249 (2008)
4. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Computing the Tutte polynomial in vertex-exponential time. In: FOCS, pp. 677–686 (2008)
5. Bläser, M., Dell, H.: Complexity of the cover polynomial. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 801–812. Springer, Heidelberg (2007)
6. Brylawski, T.: The Tutte polynomial, Matroid theory and its applications. In: Centro Internazionale Matematico Estivo, pp. 125–275 (1982)
7. Giménez, O., Hliněný, P., Noy, M.: Computing the Tutte polynomial on graphs of bounded clique-width. SIAM J. on Discrete Mathematics 20, 932–946 (2006)
8. Goldberg, L.A., Jerrum, M.: The complexity of ferromagnetic Ising with local fields. Combinatorics, Probability and Computing 16(1), 43–61 (2007)
9. Goldberg, L.A., Jerrum, M.: Inapproximability of the Tutte polynomial. Information and Computation 206(7), 908–929 (2008)
10. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? Journal of Computer and System Sciences 63(4), 512–530 (2001)
11. Istrail, S.: Statistical mechanics, three-dimensionality and NP-completeness. I. universality of intractability for the partition function of the Ising model across non-planar lattices. In: STOC, pp. 87–96 (2000)
12. Jaeger, F., Vertigan, D.L., Welsh, D.J.: On the computational complexity of the Jones and Tutte polynomials. Math. Proc. Cambridge 108(1), 35–53 (1990)
13. Jerrum, M., Snir, M.: Some exact complexity results for straight-line computations over semirings. Journal of the ACM 29(3), 874–897 (1982)
14. Koivisto, M.: Partitioning into sets of bounded cardinality. In: IWPEC, pp. 258–263 (2009)
15. Kutzkov, K.: New upper bound for the #3-SAT problem. Information Processing Letters 105(1), 1–5 (2007)
16. Lawler, E.L.: A note on the complexity of the chromatic number problem. Information Processing Letters 5, 66–67 (1976)
17. Papadimitriou, C.M.: Computational Complexity. Addison-Wesley, Reading (1994)
18. Raz, R.: Multi-linear formulas for permanent and determinant are of super-polynomial size. Journal of the ACM 56(2), 1–17 (2009)
19. Ryser, H.J.: Combinatorial mathematics. Carus Math. Monographs, vol. 14. Mathematical Association of America (1963)
20. Sekine, K., Imai, H., Tani, S.: Computing the Tutte polynomial of a graph of moderate size. In: Staples, J., Katoh, N., Eades, P., Moffat, A. (eds.) ISAAC 1995. LNCS, vol. 1004, pp. 224–233. Springer, Heidelberg (1995)
21. Sokal, A.D.: Chromatic roots are dense in the whole complex plane. Combinatorics, Probability and Computing 13(02), 221–261 (2004)
22. Sokal, A.D.: The multivariate Tutte polynomial (alias Potts model) for graphs and matroids. In: Surveys in Combinatorics. London Mathematical Society Lecture Note Series, vol. 327, pp. 173–226. Cambridge University Press, Cambridge (2005)
23. Valiant, L.G.: The complexity of computing the permanent. Theoretical Computer Science 8(2), 189–201 (1979)