

# Exponential Time Complexity of the Permanent and the Tutte Polynomial

HOLGER DELL

University of Wisconsin–Madison, USA

and

THORE HUSFELDT

IT University of Copenhagen, Denmark and Lund University, Sweden

and

DÁNIEL MARX

Humboldt University of Berlin, Germany

and

NINA TASLAMAM

IT University of Copenhagen, Denmark

and

MARTIN WAHLÉN

Lund University, Sweden and Uppsala University, Sweden

---

We show conditional lower bounds for well-studied #P-hard problems:

- The number of satisfying assignments of a 2-CNF formula with  $n$  variables cannot be counted in time  $\exp(o(n))$ , and the same is true for computing the number of all independent sets in an  $n$ -vertex graph.
- The permanent of an  $n \times n$  matrix with entries 0 and 1 cannot be computed in time  $\exp(o(n))$ .
- The Tutte polynomial of a multigraph with  $n$  vertices and  $m$  edges cannot be computed in time  $\exp(o(n))$  at most evaluation points  $(x, y)$  in the case of multigraphs, and it cannot be computed in time  $\exp(o(n/\text{poly log } n))$  in the case of simple graphs.

Our lower bounds are relative to (variants of) the Exponential Time Hypothesis (ETH), which says that the satisfiability of  $n$ -variable 3-CNF formulas cannot be decided in time  $\exp(o(n))$ . We relax this hypothesis by introducing its counting version #ETH, namely that the satisfying assignments cannot be counted in time  $\exp(o(n))$ . In order to use #ETH for lower bounds, we make it more robust by transferring the sparsification lemma for  $d$ -CNF formulas to the counting setting.

Categories and Subject Descriptors: F.2.1 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Numerical Algorithms and Problems*; G.2.1 [**Mathematics of Computing**]: Discrete Mathematics—*Combinatorics*

General Terms: Theory, Algorithms

---

Preliminary versions of this paper appeared in the proceedings of ICALP 2010 [Dell et al. 2010] and IPEC 2010 [Husfeldt and Taslamam 2010].

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 20 ACM 0000-0000/20/0000-0001 \$5.00

Additional Key Words and Phrases: computational complexity, counting problems, Tutte polynomial, permanent, exponential time hypothesis

---

## 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 [Ryser 1963] 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 [Björklund et al. 2008] 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  $\exp(O(\sqrt{n}))$  [Sekine et al. 1995]. 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) cannot be solved in time  $\exp(o(n))$ . More specifically, we introduce #ETH, a counting analogue of ETH which models the hypothesis that *counting* the satisfying assignments cannot be done in time  $\exp(o(n))$ .

### Computing the permanent

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

$$\text{per } A = \sum_{\pi \in S_n} \prod_{1 \leq i \leq n} A_{i\pi(i)}, \quad (1)$$

where  $S_n$  is the set of permutations of  $\{1, \dots, n\}$ . This is redolent of the determinant from linear algebra,  $\det A = \sum_{\pi} \text{sign}(\pi) \prod_i A_{i\pi(i)}$ , 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 [Valiant 1979] for the permanent shows that no polynomial-time algorithm à la “Gaussian elimination for the permanent” can exist unless  $P = NP$ , and indeed unless  $P = P^{\#P}$ . Several unconditional lower bounds for the permanent in restricted models of computation are also known. Jerum and Snir [1982] 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 [2009] 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 [1984]. One of the 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  $\text{per } A$  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 [Agrawal 2006]; 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  $\exp(O(\log^2 n))$ . We show 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

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

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.”<sup>1</sup>

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 [Jaeger et al. 1990], 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 [Björklund et al. 2008], 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 [Sekine et al. 1995]. 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.

It has been known since the 1970s [Lawler 1976] 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 [Impagliazzo et al. 2001]. 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 [Giménez et al. 2006] 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 [2008], which shows that the Tutte polynomial is hard to even 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 [Björklund et al. 2008].

### Complexity assumptions

The standard complexity assumption  $P \neq NP$  is clearly not sufficient for our purposes: according to our current knowledge,  $P \neq NP$  can be compatible with having fast subexponential-time algorithms for NP-hard problems such as 3-SAT. What we need is a complexity assumption stating that certain problems can be solved only in exponential time.

The exponential time hypothesis (ETH) as defined in [Impagliazzo et al. 2001] is that satisfiability of 3-CNF formulas cannot be computed substantially faster than by trying all possible assignments, i.e., it cannot be done in time  $\exp(o(n))$ . Formally, this reads as follows:

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

A different way of formulating ETH is to say that there is no algorithm deciding

<sup>1</sup>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 [Istail 2000].

3-SAT in time  $\exp(o(n))$ . The latter statement is clearly implied by the above statement, and it will be more convenient for discussion to use this form and state results this way.

In two of our lower bounds (Theorem 2 and Theorem 3(iii)), we need a slightly stronger assumption that rules out the possibility of randomized algorithms as well:

(rETH) There is a constant  $c > 0$  such that no *randomized* algorithm can decide 3-SAT in time  $\exp(c \cdot n)$  with error probability at most  $\frac{1}{3}$ .

The reason why we need rETH in these two proofs is that we are reducing from a variant of 3-SAT called UNIQUE 3-SAT, where we assume that the given 3-CNF formula has at most one satisfying assignment. Calabro et al. [2003] established a lower bound on UNIQUE 3-SAT assuming rETH, thus our results are also relative to this complexity assumption. Reducing from UNIQUE 3-SAT allows us to avoid the use of interpolation, which typically weakens the lower bound by polylogarithmic factors (compare the proofs of (ii) and (iii) in Theorem 3).

Intuitively, counting problems are much harder than deciding the existence of a solution: the latter problem needs to find only a single solution, while the former problem has to somehow reason about the set of all possible solutions. A formal evidence is that many natural counting problems are #P-hard and therefore not only as hard as all problems in NP but as hard as all the problems in the polynomial hierarchy [Toda 1989]. If counting problems seem to be so much harder, then it is natural to ask if their hardness can be demonstrated by a weaker complexity assumption than what is needed for the decision problems. We show that our lower bounds (with the exception of Theorem 2 and Theorem 3(ii)) can be obtained using the weaker complexity assumption stating that counting the number of solutions to a 3-CNF formula requires exponential time (i.e., a counting variant of ETH).

*Name.* #3-SAT

*Input.* 3-CNF formula  $\varphi$  with  $n$  variables and  $m$  clauses.

*Output.* The number of satisfying assignments to  $\varphi$ .

The best known algorithm for this problem runs in time  $O(1.6423^n)$  [Kutzkov 2007].

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

ETH trivially implies #ETH whereas the other direction is not known.

By introducing the sparsification lemma, Impagliazzo et al. [2001] show that ETH is a robust notion in the sense that the clause width 3 and the parameter  $n$  (number of variables) in its definition can be replaced by  $d \geq 3$  and  $m$  (number of clauses), respectively, to get an equivalent hypothesis, albeit the constant  $c$  may change in doing so. As most of the reductions are sensitive to the number of clauses, this stronger form of ETH is essential for proving tight lower bounds for concrete problems. In order to be able to use #ETH in such reductions, we transfer the sparsification lemma to # $d$ -SAT and get a similar kind of robustness for #ETH.

**Theorem 1.** *Let  $d \geq 3$  be an integer. Then #ETH holds if and only if there is a constant  $c > 0$  such that no deterministic algorithm can solve # $d$ -SAT time  $\exp(c \cdot m)$ .*

The proof of this theorem is spelled out in Appendix A.

**Results: Counting Independent Sets**

In light of Theorem 1, it is natural to consider the exponential time complexity of #2-SAT. Restricted to antimonotone 2-CNF formulas, this corresponds to counting *all* independent sets in a given graph, which cannot be done in time  $\exp(o(n/\log^3 n))$  under #ETH [Hoffmann 2010]. The loss of the poly log-factor in the exponent is due to the interpolation inherent in the hardness reduction. We avoid interpolation using the isolation lemma for  $d$ -CNF formulas [Calabro et al. 2003] and get an asymptotically tight lower bound, with the drawback that our lower bound only holds under the randomized version of ETH instead of #ETH.

**Theorem 2.** *Assuming rETH, there is no algorithm for counting independent sets and for #2-SAT running in time  $\exp(o(m))$ , where  $m$  is the number of edges and clauses, respectively.*

We discuss the isolation technique and prove this theorem in §2.

**Results: The Permanent**

For a set  $S$  of rationals we define the following problems:

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

We write  $\text{PERM}$  for  $\text{PERM}^{\mathbb{N}}$ . If  $B$  is a bipartite graph with  $A_{ij}$  edges from the  $i$ th vertex in the left half to the  $j$ th vertex in the right half ( $1 \leq i, j \leq n$ ), then  $\text{per}(A)$  equals the number of perfect matchings of  $B$ . Thus  $\text{PERM}$  and  $\text{PERM}^{0,1}$  can be viewed as counting the perfect matchings in bipartite multigraphs and bipartite simple graphs, 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 3.**

- (i)  $\text{PERM}^{-1,0,1}$  and  $\text{PERM}$  cannot be computed in time  $\exp(o(m))$  under #ETH.
- (ii)  $\text{PERM}^{0,1}$  cannot be computed in time  $\exp(o(m/\log n))$  under #ETH.
- (iii)  $\text{PERM}^{0,1}$  cannot be computed in time  $\exp(o(m))$  under rETH.

The proof of this theorem is in §3. For (i), we follow a standard reduction by Valiant [Valiant 1979; Papadimitriou 1994] but use a simple equality gadget derived from [Bläser and Dell 2007] instead of Valiant’s XOR-gadget, and we use interpolation to get rid of the negative weights. To establish (ii) we simulate edge weights  $w > 1$  by gadgets of size logarithmic in  $w$ , which increases the number of vertices and edges by a logarithmic factor. For (iii) we use the isolation lemma and the reduction from part (i), and we simulate the edge weights  $-1$  without interpolation by replacing them with 2 and doing computation modulo 3. Observe that (ii) is a stronger statement than (i), but uses a stronger complexity assumption (rETH instead of #ETH).

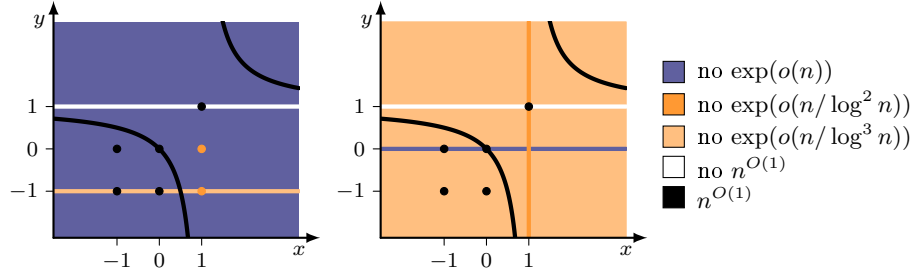


Fig. 1. Exponential time complexity under #ETH of the Tutte plane for multigraphs (left) and simple graphs (right) in terms of  $n$ , the number of vertices. The white line  $y = 1$  on the map is uncharted territory, and we only have the #P-hardness. The black hyperbola  $(x - 1)(y - 1) = 1$  and the 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.

#### Results: 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,  $\text{TUTTE}(x, y)$  cannot be computed in time  $\exp(o(n))$  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 Figure 1. For quick reference, we state the propositions in which the individual results are proved and the techniques used in each case.

**Theorem 4.** *Let  $(x, y) \in \mathbb{Q}^2$ . Under #ETH,*

- (i) ■  $\text{TUTTE}(x, y)$  cannot be computed in time  $\exp(o(n))$  if  $(x - 1)(y - 1) \neq 1$  and  $y \notin \{0, \pm 1\}$ ,  
( Proposition 3 in §5; stretching and thickening )
- (ii) ■  $\text{TUTTE}^{01}(x, y)$  cannot be computed in time  $\exp(o(n))$  if  $y = 0$  and  $x \notin \{0, \pm 1\}$ ,  
( Proposition 6 in Appendix B; Linial's reduction )
- (iii) ■  $\text{TUTTE}^{01}(x, y)$  cannot be computed in time  $\exp(o(m/\log^2 m))$  if  $x = 1$  and  $y \neq 1$ ,  
( Proposition 5 in §6; inflation with Bounce graphs )
- (iv) ■  $\text{TUTTE}^{01}(x, y)$  cannot be computed in time  $\exp(o(m/\log^3 m))$  if  $(x - 1)(y - 1) \notin \{0, 1\}$  and  $(x, y) \notin \{(-1, -1), (-1, 0), (0, -1)\}$ .  
( Proposition 4 in §6; inflation with Theta graphs )

Our reductions often give edge-exponential lower bounds, i.e., bounds in terms of the parameter  $m$ , which implies the same bounds in terms of  $n$  because  $m \geq n$  in connected graphs. Moreover, the lower bound stating that there is no algorithm running in time  $\exp(o(m/\text{poly log } m))$  together with the algorithm running in time  $\exp(O(n))$  from [Björklund et al. 2008] implies that worst-case instances are *sparse*, in the sense that  $m = O(n \cdot \text{poly log } n)$ .

In an attempt to prove Theorem 4, 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 to “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 complexity 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, we cannot always afford such constructions in our setting, 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 more subtle techniques, e.g., the constructions in §6, which use graph products with graphs of polylogarithmic size instead of thickenings and stretches. 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 under rETH and #ETH hold for these problems as well. In particular, the following graph polynomials (for example, as a list of their coefficients) cannot be computed in time  $\exp(o(m))$  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, our results show that the following counting problems on multigraphs cannot be solved in time  $\exp(o(n))$ : # 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 [Björklund and Husfeldt 2008], the current best bound is  $O(1.619^n)$  [Koivisto 2009].

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

## 2. COUNTING INDEPENDENT SETS

In this section, we establish Theorem 2, the hardness of counting independent sets and of #2-SAT. For the proof, we make use of the randomized ETH-hardness of the following problem.

*Name.* UNIQUE 3-SAT.

*Input.* 3-CNF formula  $\varphi$  with  $m$  clauses and at most one satisfying assignment.

*Decide.* Is  $\varphi$  satisfiable?

Calabro et al. [2003] use an isolation lemma for  $d$ -CNF formulas to show that solving this problem in subexponential time implies that the (randomized) exponential time hypothesis fails.

**Theorem 5 (Corollary 2 of Calabro et al. [2003]).**

rETH implies that UNIQUE 3-SAT cannot be computed in time  $\exp(o(m))$ .

We are now in the position to prove Theorem 2.

**Theorem 2 (restated).** *Assuming rETH, there is no algorithm for counting independent sets and for #2-SAT running in time  $\exp(o(m))$ , where  $m$  is the number of edges and clauses, respectively.*

*Proof.* Let  $\varphi$  be an instance of UNIQUE 3-SAT with  $m$  clauses. We construct a graph  $G$  with  $O(m)$  edges that has an odd number of independent sets if and only if  $\varphi$  is satisfiable. For each variable  $x$ , we introduce vertices  $x$  and  $\bar{x}$ , and the edge  $(x\bar{x})$ . This makes sure that any independent set of  $G$  chooses at most one of  $\{x, \bar{x}\}$ , so we can interpret the independent set as a partial assignment to the variables of  $\varphi$ . For each clause  $c = (\ell_1 \vee \ell_2 \vee \ell_3)$  of  $\varphi$ , we introduce a clique in  $G$  that consists of seven vertices  $c_1, \dots, c_7$ . These vertices correspond to the seven partial assignments that assign truth values to the literals  $\ell_1, \ell_2$ , and  $\ell_3$  in such a way that  $c$  is satisfied. Any independent set of  $G$  contains at most one  $c_i$  for each clause  $c$ . To ensure that the independent set chooses the variables and partial assignments of the clauses consistently, we add an edge for every  $c_i$  and every variable  $x$  occurring in the clause  $c$ : If the partial assignment that corresponds to  $c_i$  sets  $x$  to true, we add  $(c_i\bar{x})$  to  $G$ ; otherwise, we add  $(c_ix)$  to  $G$ . To finalize the construction, we introduce guard vertices  $g_x$  and  $g_c$  for every variable  $x$  and every clause  $c$ , along with the edges  $(g_x x)$ ,  $(g_x \bar{x})$ , and  $(g_c c_i)$  for  $i = 1, \dots, 7$ .

We now prove that  $G$  has the required properties. First, any independent set contains at most  $n$  literal vertices and at most  $m$  clause vertices. *Good* independent sets are those that contain exactly  $n$  literal and  $m$  clause vertices (and no guard vertex). Good independent sets correspond to the satisfying assignments of  $\varphi$  in a natural way. We now show that the number of bad independent sets is even. For this, let  $S$  be a bad independent set, that is,  $S$  is disjoint from  $\{x, \bar{x}\}$  for some  $x$  or it is disjoint from  $\{c_1, \dots, c_7\}$  for some clause  $c$ . By construction, the neighborhood of either  $g_x$  or  $g_c$  is disjoint from  $S$ . Let  $g$  be the lexicographically first guard vertex whose neighborhood is disjoint from  $S$ . Both the sets  $S \setminus \{g\}$  and  $S \cup \{g\}$  are bad independent sets and  $S$  is one of these sets. Formally, we can therefore define a function that maps these sets onto each other. This function is a well-defined involution on the set of bad independent sets, and it does not have any fixed points. Therefore, the number of bad independent sets is even, and the

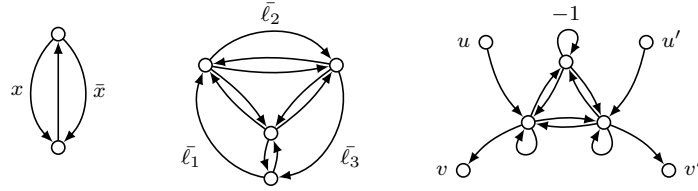


Fig. 2. Left: A selector gadget for variable  $x$ . Depending on which of the two cycles is chosen, we assume  $x$  to be set to true or false. Middle: A clause gadget for the clause  $\ell_1 \vee \ell_2 \vee \ell_3$ . The gadget allows all possible configurations for the outer edges, except for the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

parity of the number of independent sets of  $G$  is equal to the parity of the number of satisfying assignments of  $\varphi$ .

The above reduction shows that an  $\exp(o(m))$ -time algorithm for counting independent sets modulo 2 implies an  $\exp(o(m))$ -time algorithm for UNIQUE 3-SAT. By Theorem 5, this implies that rETH fails.

To establish the hardness of #2-SAT, we reduce from counting independent sets. Let  $G$  be a graph. For each vertex  $v$ , we introduce a variable  $v$ , and each edge  $(uv)$  becomes a clause  $(\bar{u} \vee \bar{v})$ . The satisfying assignments of the so constructed 2-CNF formula are in one-to-one correspondence with the independent sets of  $G$ . ■

### 3. THE PERMANENT

This section contains the proof of Theorem 3. With  $[0, n] = \{0, 1, \dots, n\}$  we establish the reduction chain  $\#3\text{-SAT} \preceq \text{PERM}^{-1,0,1} \preceq \text{PERM}^{[0,n]} \preceq \text{PERM}^{01}$  while taking care of the instance sizes.

#### Theorem 3 (restated).

- (i)  $\text{PERM}^{-1,0,1}$  and  $\text{PERM}$  cannot be computed in time  $\exp(o(m))$  under #ETH.
- (ii)  $\text{PERM}^{01}$  cannot be computed in time  $\exp(o(m/\log n))$  under #ETH.
- (iii)  $\text{PERM}^{01}$  cannot be computed in time  $\exp(o(m))$  under rETH.

*Proof.* To establish (i), we reduce #3-SAT in polynomial time to  $\text{PERM}^{-1,0,1}$  such that 3-CNF formulas  $\varphi$  with  $m$  clauses are mapped to graphs  $G$  with  $O(m)$  edges. For technical reasons, we preprocess  $\varphi$  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  $\varphi$ ). We construct  $G$  with  $O(m)$  edges and weights  $w : E \rightarrow \{\pm 1\}$  such that  $\#\text{SAT}(\varphi)$  can be derived from  $\text{per } G$  in polynomial time. For weighted graphs, the permanent is

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

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.

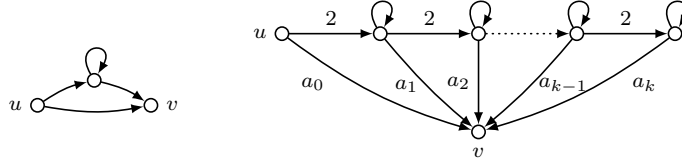


Fig. 3. Left: This gadget simulates in unweighted graphs edges  $uv$  of weight 2. Right: This gadget simulates edges  $uv$  of weight  $a = \sum_{i=0}^k a_i 2^i$  with  $a_i \in \{0, 1\}$ .

In Figure 2, the gadgets of the construction are depicted. For every variable  $x$  that occurs in  $\varphi$ , we add a *selector gadget* to  $G$ . For every clause  $c = \ell_1 \vee \ell_2 \vee \ell_3$  of  $\varphi$ , we add a *clause gadget* to  $G$ . Finally, we connect the edge labelled by a literal  $\ell$  in the selector gadget with all occurrences of  $\ell$  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 [Papadimitriou 1994] and [Bläser and Dell 2007]. 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  $\varphi$  such that  $i = j$  holds and  $i$  is constant over all assignments, we obtain  $\text{per } G = (-2)^i \cdot \#\text{SAT}(\varphi)$ .

For the second part of (i), 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  $-1$  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  $p(x) = \text{per } G_x$  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  $\text{per } G_a$  for  $a = 0, \dots, d$  and then interpolate, to get the coefficients of the polynomial  $p(x)$ . At last, we return the value  $p(-1) = \text{per } G$ . This completes the reduction, which queries the oracle  $d + 1$  graphs that have at most  $m$  edges each.

For (ii), we have to get rid of weights larger than 1. 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 Figure 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.,  $\text{per } G' = \text{per } G_a$ . Unfortunately, the reduction increases 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 implies that (ii).

For (iii), we assume that  $\text{rETH}$  holds. Theorem 5 gives that  $\text{UNIQUE 3-SAT}$  cannot be computed in time  $\exp(o(m))$ . Now we apply the first reduction of (i) to a formula  $\varphi$  which is promised to have at most one satisfying assignment. Then the number  $\text{per } G = (-2)^i \cdot \#\text{SAT}(\varphi)$  is either 0 or  $(-2)^i$ . In  $G$ , we replace each edge of weight  $-1$  by a gadget of weight  $2 \equiv -1 \pmod{3}$  and similarly get that  $(\text{per } G \pmod{3})$  is  $(0 \pmod{3})$  or  $(4^i \pmod{3})$ . Since  $(4^i \pmod{3}) \neq 0$ , we can distinguish

the case in which  $\varphi$  is unsatisfiable from the case in which  $\varphi$  has exactly one satisfying assignment. ■

#### 4. HYPERBOLAS IN THE TUTTE PLANE

Our first goal is to show that computing the coefficients of the Tutte polynomial is hard along any single hyperbola. It is useful to view the Tutte polynomial in the Fortuin–Kasteleyn formulation:

$$Z(G; q, w) = \sum_{A \subseteq E} q^{k(A)} w^{|A|}.$$

where  $k(A)$  is the number of connected components in the subgraph  $(V, A)$ . The connection to the Tutte polynomial is given by

$$T(G; x, y) = (x - 1)^{-k(E)} (y - 1)^{-|V|} Z(G; q, w), \quad (3)$$

where  $q = (x - 1)(y - 1)$  and  $w = y - 1$ ,

see [Sokal 2005, eq. (2.26)].

##### The Ising Hyperbola

We begin with the case  $q = 2$ .

**Proposition 1.** *If #ETH holds, the coefficients of the polynomial  $w \mapsto Z(G; 2, w)$  for a given simple graph  $G$  cannot be computed in time  $\exp(o(m))$ .*

*Proof.* The reduction is from #MAXCUT and well-known, see, e.g., [Jerrum and Sinclair 1993, Theorem 15].

*Name.* #MAXCUT

*Input.* Simple undirected graph  $G$ .

*Output.* The number of maximum cuts.

A maximum cut is a set  $C \subseteq V(G)$  that maximizes the number  $|E(C, \overline{C})|$  of edges of  $G$  that cross the cut. By the Fortuin–Kasteleyn identity [Sokal 2005, Theorem 2.3], one can express  $Z(G; 2, w)$  for  $G = (V, E)$  as

$$\sum_{\sigma: V \mapsto \pm 1} \prod_{uv \in E} (1 + w \cdot [\sigma(u) = \sigma(v)]).$$

Here the Iverson bracket  $[P]$  is 1 if  $P$  is true and is 0 if  $P$  is false. The sets  $\sigma^{-1}(1)$  and  $\sigma^{-1}(-1)$  define a cut in  $G$ , so we can write the above expression as

$$\sum_{U \subseteq V} \prod_{\substack{uv \in E \\ [u \in U] = [v \in U]}} (1 + w) = \sum_{C \subseteq V(G)} (1 + w)^{m - |E(C, \overline{C})|},$$

Now, the coefficient of  $(1 + w)^{m-c}$  in  $Z(G; 2, w)$  is the number of cuts in  $G$  of size  $c$ . In particular, after some interpolation, we can compute the number of maximum cuts in  $G$  from the coefficients of  $w \mapsto Z(G; 2, w)$ . But as we observe in Appendix B, #MAXCUT cannot be computed in time  $\exp(o(m))$  under #ETH. ■

### The Multivariate Tutte Polynomial

For other  $q$ , in particular nonintegers, it is simpler to work with a *multivariate* formulation of the Tutte polynomial due to Fortuin and Kasteleyn [Fortuin and Kasteleyn 1972]. We use Sokal's definition [Sokal 2005]: Let  $G = (V, E)$  be an undirected graph whose edge weights are given by a function  $\mathbf{w}: E \rightarrow \mathbb{Q}$ . Then

$$Z(G; q, \mathbf{w}) = \sum_{A \subseteq E} q^{k(A)} \prod_{e \in A} \mathbf{w}(e). \quad (4)$$

If  $\mathbf{w}$  is single-valued, in the sense that  $\mathbf{w}(e) = w$  for all  $e \in E$ , we recover  $Z(G; q, w)$ .

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 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 sometimes use 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  $Z$ :

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

### Three-terminal minimum cut

For  $q \notin \{1, 2\}$ , we first establish that with two different edge weights, one of them negative, the multivariate Tutte polynomial computes the number of 3-terminal minimum cuts, for which we observe hardness under #ETH in Appendix B. This connection has been used already in [Goldberg and Jerrum 2007; 2008], with different reductions, to prove hardness of approximation.

The graphs we consider here 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

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

For such a graph, we have

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

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  $q \notin \{1, 2\}$ . The coefficients of the polynomial  $w \mapsto Z_0(G; q, \mathbf{w})$ , with  $\mathbf{w}$  as in (6), for a given simple graph  $G$  cannot be computed in time  $\exp(o(m))$  under #ETH. Moreover, this is true even if  $|T| = 3$ .*

*Proof.* In Appendix B, we argue that a standard reduction from #MAXCUT already implies that the problem #3-TERMINAL MINCUT cannot be computed in time  $\exp(o(m))$  under #ETH.

*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 edge subsets  $A \subseteq E$  of minimal size that separate  $t_1$  from  $t_2$ ,  $t_2$  from  $t_3$ , and  $t_3$  from  $t_1$ .

We reduce this problem to the problem of evaluating the coefficients of  $Z_0$  at  $q \notin \{1, 2\}$ . Suppose  $G' = (V, E, t_1, t_2, t_3)$  is an instance of #3-TERMINAL MINCUT with  $n = |V|$  and  $m = |E|$ . We can assume that  $G'$  is simple and connected. We modify  $G'$  by adding a triangle between the terminals, obtaining the graph  $G = (V, E \cup T)$  where  $T = \{t_1 t_2, t_2 t_3, t_1 t_3\}$ ; note that  $n(G) = n$ ,  $m(G) = m + 3$ , and  $|T| = 3$ .

We focus our attention on the family  $\mathcal{A}$  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  $\mathcal{A}$  if and only if  $E - A$  is a 3-terminal cut in  $G'$ . Then we can split the sum in (7) into

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

We first show that the second term of (8) vanishes. Consider an edge subset  $A \notin \mathcal{A}$  and assume without loss of generality that it connects the terminals  $t_1$  and  $t_2$ . Consider  $B \subseteq T$ , and let  $B' = B \oplus \{t_1 t_2\}$ , so that  $B'$  is the same as  $B$  except for  $t_1 t_2$ . 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  $t_1 t_2$  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  $A \in \mathcal{A}$ , 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

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

and after some simplification we can write (8) as

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

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

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

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

The edge subsets  $A \in \mathcal{A}$  are exactly the complements of the 3-terminal cuts in  $G'$ . Now consider the family  $\mathcal{C}$  of *minimal* 3-terminal cuts, all of size  $c$ . The sets

$E - A$  in  $\mathcal{C}$  are exactly the sets  $A$  of size  $m - c$  in  $\mathcal{A}$ , and by minimality,  $k(A) = 3$ . Thus,

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

Thus, if we could compute the coefficients  $d_0, \dots, d_m$  of  $w \mapsto Q^{-1}Z_0(G; q, \mathbf{w})$ , then we could determine the smallest  $c$  so that  $d_{m-c} \neq 0$  and return  $d_{m-c} = |\mathcal{C}|/Q$ , the number of 3-terminal mincuts. ■

### General Hyperbolas

We want to use Lemma 1 to show that computing the coefficients of the univariate Tutte polynomial at any fixed  $q \notin \{1, 2\}$  is hard. For this, we need to get rid of negative weights and reduce to a single-valued weight function. In [Goldberg and Jerrum 2008], this is done by thickenings and stretches, which we have to avoid. Since the number of edges with a negative weight is small (in fact, 3), we can use another tool: deletion–contraction.

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  $e$  ( $\Delta \mapsto \mathcal{L}$ ) and  $G/e$  arises from  $G$  by *contracting* the edge  $e$  ( $\Delta \mapsto \infty$ ) that is, deleting it and identifying its endpoints (so any remaining edges between these two endpoints become loops).

It is known [Sokal 2005, eq. (4.6)] that

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

An edge  $e$  is a *bridge* of  $G$  if deleting  $e$  from  $G$  increases the number of connected components. The above gives a deletion–contraction identity for  $Z_0$  as well:

$$Z_0(G; q, \mathbf{w}) = \begin{cases} qZ_0(G - e; q, \mathbf{w}) + \mathbf{w}(e)Z_0(G/e; q, \mathbf{w}) & \text{if } e \text{ is a bridge,} \\ Z_0(G - e; q, \mathbf{w}) + \mathbf{w}(e)Z_0(G/e; q, \mathbf{w}) & \text{otherwise.} \end{cases} \quad (10)$$

**Proposition 2.** *Let  $q$  be a rational number with  $q \notin \{1, 2\}$ . The coefficients of the polynomial  $v \mapsto Z_0(G; q, v)$  for a given simple graph  $G$  cannot be computed in time  $\exp(o(m))$  under #ETH.*

By (5), this proposition also holds for  $Z$  instead of  $Z_0$  when  $q \notin \{0, 1, 2\}$ .

*Proof.* Let  $G = (V, E)$  be a graph as in the previous lemma, with three edges  $T = \{e_1, e_2, e_3\}$  of weight  $-1$ . The given reduction actually uses the restriction that  $G' = (V, E \setminus T)$  is connected, so we can assume that this is the case. Thus, none of the  $T$ -edges is a bridge, so three applications of (10) to delete and contract these edges, gives

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

where for each  $C \subseteq \{1, 2, 3\}$ , the graph  $G_C$  is constructed from  $G$  by removing  $e_1, e_2, e_3$  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  $Z_0(G_C; q, \mathbf{w}) = Z_0(G_C; q, w)$ .

The resulting  $G_C$  are not necessarily simple, because the contracted edges from  $T$  may have been part of a triangle and may have produced a loop. (In fact, investigating the details of the previous lemma, 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 3-path. This operation, known as a 3-stretch, is known to largely preserve the value of  $Z$  and  $Z_0$  (see [Sokal 2004] for the former and [Goldberg and Jerrum 2008] for the latter). In particular,

$$Z_0(G_C; q, w) = f(q, w')^m \cdot Z_0(G'_C; q, w'),$$

where for  $q \neq 0$

$$1 + \frac{q}{w} = \left(1 + \frac{q}{w'}\right)^3 \quad \text{and} \quad f(q, w') = q^{-1} \cdot ((q + w')^3 - w'^3),$$

and for  $q = 0$

$$w = w'/3 \quad \text{and} \quad f(q, w') = 1/(3w'^2).$$

In summary, to compute the coefficients of the polynomial  $w \mapsto Z_0(G; q, \mathbf{w})$ , we need to compute the 8 polynomials  $v \mapsto Z_0(G_C; q, v)$ , one for each  $G_C$ . We use the above equation and the assumed oracle for simple graphs to do this. We note that every  $G'_C$  is simple and has at most  $n + m$  vertices and at most  $2m$  edges. ■

## 5. INDIVIDUAL POINTS FOR MULTIGRAPHS

If we allow graphs to have multiple edges, we can use thickening and interpolation, one of the original strategies of [Jaeger et al. 1990], for relocating the hardness result for hyperbolas from Proposition 1 and Proposition 2 to individual points in the Tutte plane. For most points, this gives us tight bounds in terms of  $n$ , the number of vertices, but not for points with  $y \in \{0, \pm 1\}$ , where thickening fails completely.

We recall the thickening identities for the Tutte polynomial. The  $k$ -thickening of  $G$  is the graph  $G_k$  in which all edges have been replaced by  $k$  parallel edges. One can show [Sokal 2005, (4.21)] that, with  $w_k = (1 + w)^k - 1$ ,

$$Z(G; q, w_k) = Z(G_k; q, w). \tag{12}$$

It is easy to transfer this result to the Tutte polynomial  $T$  using (3), yielding special cases of Brylawski's well-known graph transformation rules.

We use interpolation and obtain Theorem 4(i) for  $y \neq 0$  from the following.

**Proposition 3.** *Let  $(q, w) \in \mathbb{Q}^2$  with  $w \notin \{0, -1, -2\}$  and  $q \neq 1$ .*

*$Z(G; q, w)$  for a given graph  $G$  (not necessarily simple) cannot be computed in time  $\exp(o(n))$  under #ETH.*

*Proof.* We observe that the values  $w_k = (1 + w)^k - 1$  are all distinct for  $k = 0, 1, \dots, m$ . Thus, the  $k$ -thickenings  $G_k$  of  $G$  give rise to  $m + 1$  different weight shifts, the evaluations of which,  $Z(G; q, w_k)$ , can be obtained from  $Z(G_k; q, w)$  using (12). Thus, with oracle access to  $G' \mapsto Z(G'; q, w)$ , we can compute the coefficients of the polynomial  $v \mapsto Z(G; q, v)$  in polynomial time for any given  $G$ .

By Proposition 1 and Proposition 2, this cannot be done in time  $\exp(o(n))$  under #ETH. Since the number of vertices is  $n$  in each  $G_k$ , computing  $G' \mapsto Z(G'; q, w)$  cannot be done in time  $\exp(o(n))$  under #ETH. ■

The proof of Theorem 4(ii) uses Linial’s well-known reduction for the chromatic polynomial [Linial 1986], and is deferred to Proposition 6 in Appendix B.

## 6. INDIVIDUAL POINTS FOR SIMPLE GRAPHS

In this section we show 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 lose a polylogarithmic factor in the exponent of the lower bound. 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.

### Graph inflations

We use the graph theoretic version of Brylawski’s tensor product for matroids [Brylawski 1982]. We found the following terminology more intuitive in our setting.

**Definition 1 (Graph inflation).** *Let  $H$  be a 2-terminal undirected graph. For any undirected graph  $G = (V, E)$ , an  $H$ -inflation of  $G$ , denoted  $G \otimes H$ , is obtained by replacing every edge  $xy \in E$  by (a fresh copy of)  $H$ , identifying  $x$  with one of the terminals of  $H$  and  $y$  with the other.*

If  $H$  is not symmetric with respect to its two terminals, then the graph  $G \otimes H$  need not be unique since there are in general two non-isomorphic ways to replace an edge  $xy$  by  $H$ . For us this difference does not matter since the resulting Tutte polynomials turn out to be the same; 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*. Thus we choose  $G \otimes H$  arbitrarily among the graphs that satisfy the condition in the definition above. Graph inflation is not commutative and Sokal uses the notation  $\vec{G}^H$ .

If  $H$  is a simple path of  $k$  edges,  $G \otimes H$  gives the usual  $k$ -stretch of  $G$ , and a bundle of  $k$  parallel edges results in a  $k$ -thickening. What makes graph inflations 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  $H$ , so that  $Z(G \otimes H; q, w) \sim Z(G; q, w')$  for some “shifted” weight  $w'$ .

For fixed rational points  $(q, w)$ , we want to use interpolation to prove the hardness of computing  $Z(G; q, w)$  for a given graph  $G$ . The basic idea is to find a suitable class of graphs  $\{H_i\}$ , such that we can compute the coefficients of the monovariate polynomial  $v \mapsto Z(G; q, v)$  for given  $G$  and  $q$  by interpolation from sufficiently many evaluations of  $Z(G; q, w_i) \sim Z(G \otimes H_i; q, w)$ . For this, we need that the number of different weight shifts  $\{w_i\}$  provided by the graph class  $\{H_i\}$  is at least  $|E(G)| + 1$ , one more than the degree of the polynomial.



does not change sign. Most important for our analysis is that the elements of the  $S_i$  are spaced apart significantly, i.e.,

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

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 (14), 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,

$$\prod_{s \in S} (b^s + q - 1) \prod_{t \in T} (b^t - 1) - \prod_{t \in T} (b^t + q - 1) \prod_{s \in S} (b^s - 1) \neq 0 \quad (15)$$

We will factor out the products in (15). Using the notation  $\|X\| = \sum_{x \in X} x$ , we rewrite

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

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 (15) and collecting terms we arrive at the equivalent claim

$$\sum_{X \subseteq S \cup T} g(X) \neq 0, \quad (16)$$

where

$$g(X) = \left( (-1)^{|T \setminus X|} (q - 1)^{|S \setminus X|} - (-1)^{|S \setminus X|} (q - 1)^{|T \setminus X|} \right) \cdot b^{\|X\|}. \quad (17)$$

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 (17) and  $|S| = |T|$ , it follows that

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

The largest exponent of  $b$  with nonzero coefficient in (17) 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 (16); 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 prove (16) 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

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

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  $\Delta = \Delta(q, w)$  can be chosen so that  $4m^2 \cdot 2|q - 1|^{1+\log m} < |q| \cdot b^{\Delta \log m}$  for all  $m \geq 2$ . If  $b < 1$ , we can similarly choose  $\Delta$  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. ■

### Points on the Hyperbolas

We show Theorem 4(iv), that evaluating  $Z$  at most points  $(q, w)$  with  $q \notin \{0, 1\}$  is hard.

**Proposition 4.** *Let  $(q, w) \in \mathbb{Q}^2 \setminus \{(4, -2), (2, -1), (2, -2)\}$  with  $q \notin \{0, 1\}$  and  $w \neq 0$ . If #ETH holds, then  $Z(G; q, w)$  for a given simple graph  $G$  cannot be computed in time  $\exp(o(m/\log^3 m))$ .*

By (3), the points  $(4, -2)$ ,  $(2, -1)$ , and  $(2, -2)$  in the  $(q, w)$ -plane correspond to the polynomial-time computable points  $(-1, -1)$ ,  $(-1, 0)$ , and  $(0, -1)$  in the  $(x, y)$ -plane.

*Proof.* We reduce from the problem of computing the coefficients of the polynomial  $v \mapsto Z(G; q, v)$ , which cannot be done in time  $\exp(o(m))$  for  $q \notin \{0, 1\}$  by Proposition 1 and Proposition 2 (assuming #ETH). We interpolate as in the proof of Proposition 3, but instead of thickenings we use Theta inflations to keep the number of edges relatively small.

First we consider the degenerate case in which  $q = -w$  or  $q = -2w$ . For a positive integer constant  $k$ , let  $G'$  be the  $k$ -thickening of  $G$ . This transformation shifts the weight to  $w'$  with

$$w' = (1 + w)^k - 1,$$

which allows us to compute  $Z(G; q, w')$  from  $Z(G'; q, w)$  using (12). In the case  $q = -w$ , we have  $1 + w = 1 - q$ , which cannot be 1 or 0, but which can also not be  $-1$  since then  $(q, w) = (2, -2)$ . Similarly, in the case  $q = -2w$ , we have  $1 + w = 1 - q/2$ , which cannot be 1. It can also not be 0 since then  $(q, w) = (2, -1)$ , neither can it be  $-1$  since then  $(q, w) = (4, -2)$ . Thus, in any case,  $(1 + w) \notin \{0, \pm 1\}$ . This means that we can choose  $k$  large enough so that  $q \notin \{-w', -2w'\}$ . This remains true if we let  $G''$  be the 2-stretch to  $G'$ , which shifts the weight to  $w''$  with

$$1 + \frac{q}{w''} = \left(1 + \frac{q}{w'}\right)^2,$$

so that  $Z(G; q, w'')$  can be computed from  $Z(G''; q, w)$  (see [Sokal 2004]). We choose  $k$  so that  $q \notin \{-w'', -2w''\}$ . The graph  $G''$  after this transformation is simple and the number of edges is only increased by a constant factor of  $2k$ .

By the above, we can assume w.l.o.g. that  $q \notin \{-w, -2w\}$ . We observe that the conditions  $w \neq 0$  and  $q \notin \{0, -w, -2w\}$  of Lemma 3 hold, and thus we can compute  $m + 1$  sets  $S_0, S_1, \dots, S_m$  with all distinct weight shifts  $w_0, \dots, w_m$  under Theta inflations.

For a given graph  $G$ , let  $G_i = G \otimes \Theta_{S_i}$ . Using Lemma 2, we can compute the values  $Z(G; q; w_i)$  from  $Z(G_i; q, w)$ . Moreover, as is clear from (4), 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. We remark that the  $G_i$  are simple graphs with at most  $O(m \log^3 m)$  edges, so the claim follows. ■

### Bounce Graphs

The *reliability line* of the Tutte plane, i.e., the line  $x = 1$ , is not covered by the above since here  $q = 0$  holds. On this line, the Tutte polynomial specializes to the *reliability polynomial*  $R(G; p)$  (with  $p = 1/y$ ), an object studied in algebraic graph theory [Godsil and Royle 2001, Section 15.8]. Given a connected graph  $G$  and a probability  $p$ ,  $R(G; p)$  is the probability that  $G$  stays connected if every edge independently fails with probability  $p$ . For example  $R(\mathfrak{C}_3; \frac{1}{3}) = Pr(\mathfrak{C}_3) + 5Pr(\mathfrak{C}_2) = (\frac{2}{3})^5 + 5 \cdot \frac{1}{3} \cdot (\frac{2}{3})^4 = \frac{112}{243}$ . Note that  $R(G; 1) = 0$  for all connected graphs, so  $p = 1$  is easy to evaluate – as it should be since it corresponds to the polynomial-time solvable point  $(1, 1)$  in the Tutte plane.

Along the reliability line, weight shift identities take a different form. Using deletion–contraction identities we obtain the following rules, which are simple multi-weighted generalizations of [Goldberg and Jerrum 2008, Section 4.3].

**Lemma 4.** *Let  $G$  be a graph with edge weights given by  $\mathbf{w} : E(G) \rightarrow \mathbb{Q}$ .*

*If  $\varphi(G)$  is obtained from  $G$  by replacing a single edge  $e \in E$  with a simple path of  $k$  edges  $P = \{e_1, \dots, e_k\}$  with  $\mathbf{w}(e_i) = w_i$ , then*

$$Z_0(\varphi(G); 0, \mathbf{w}) = C_P \cdot Z_0(G; 0, \mathbf{w}[e \mapsto w']),$$

where

$$\frac{1}{w'} = \frac{1}{w_1} + \dots + \frac{1}{w_k} \quad \text{and} \quad C_P = \frac{1}{w'} \prod_{i=1}^k w_i.$$

Here  $\mathbf{w}[e \mapsto w']$  denotes the function  $\mathbf{w}' : E(G) \rightarrow \mathbb{Q}$  that is identical to  $\mathbf{w}$  except at the point  $e$  where it is  $\mathbf{w}'(e) = w'$ .

**Lemma 5.** *If  $\varphi(G)$  is obtained from  $G$  by replacing a single edge  $e \in E$  with a bundle of parallel edges  $B = \{e_1, \dots, e_k\}$  with  $\mathbf{w}(e_i) = w_i$ , then*

$$Z_0(\varphi(G); 0, \mathbf{w}) = Z_0(G; 0, \mathbf{w}[e \mapsto w']),$$

where

$$w' = -1 + \prod_{i=1}^k (1 + w_i).$$

**Corollary 1.** *If  $\varphi(G)$  is obtained from  $G$  by replacing a single edge  $e \in E$  with a simple path of  $k$  edges of constant weight  $w$ , then*

$$Z_0(\varphi(G); 0, \mathbf{w}) = kw^{k-1} \cdot Z_0(G; 0, \mathbf{w}[e \mapsto w/k]), \quad (18)$$

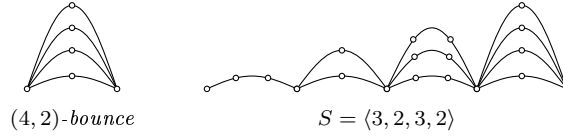
and if it is obtained from  $G$  by replacing  $e \in E$  with a bundle of  $k$  parallel edges of constant weight  $w$ , then

$$Z_0(\varphi(G); 0, \mathbf{w}) = Z_0(G; 0, \mathbf{w}[e \mapsto (1 + w)^k - 1]). \quad (19)$$

These rules are transitive [Goldberg and Jerrum 2008, Lemma 1], and so can be freely combined for more intricate weight shifts. We define a class of graph inflations, *Bounce inflations*, and use the above to show that they give rise to distinct

weight shifts along the reliability line of the Tutte polynomial. Bounce inflations are mildly inspired by  $l$ -byte numbers, in the sense that each has associated to it a sequence of length  $l$ , such that the lexicographic order of these sequences determines the size of the corresponding (shifted) weights.

**Definition 2 (Bounce graph).** For positive integers  $i$  (height) and  $s$  (width), an  $(i, s)$ -bounce is the graph obtained by identifying all the left and all the right endpoints of  $i$  simple paths of length  $s$  each. Given a sequence  $S = \langle s_1, s_2, \dots, s_l \rangle$  of  $l$  positive integers, the Bounce graph  $B_S$  is the graph obtained by concatenating  $l$  bounces at their endpoints, where the  $i$ -th bounce is an  $(i, s_i)$ -bounce, i.e., its height is  $i$  and its width is  $s_i$ .



The number  $l$  is the length of the Bounce graph  $B_S$ .

Inflating a graph by a Bounce graph shifts the weights on the reliability line as follows.

**Lemma 6.** For any graph  $G$  with  $m$  edges, any sequence  $S = \langle s_1, s_2, \dots, s_l \rangle$  of positive integers, and any non-zero rational number  $w$ , we have

$$Z_0(G \otimes B_S; 0, w) = C_S^m \cdot Z_0(G; 0, w_S),$$


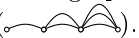
where

$$\frac{1}{w_S} = \sum_{i=1}^l \frac{1}{(1 + w/s_i)^i - 1} \quad \text{and} \quad C_S = \frac{1}{w_S} \cdot \prod_{i=1}^l w^{(s_i-1)i} ((w + s_i)^i - s_i^i).$$

*Proof.* We start with  $G \otimes B_S$  and consider the effect that replacing one of the  $m$  canonical copies of  $B_S$  with a single edge  $e$  has. We show that, with  $\varphi$  denoting this operation,

$$Z_0(G \otimes B_S; 0, w) = C_S \cdot Z_0(\varphi(G \otimes B_S); 0, \mathbf{w}[e \mapsto w_S]), \quad (20)$$

where  $w_S$  has the above form, and  $\mathbf{w}$  has the old value  $w$  on all unaffected edges. The lemma then follows by successively applying  $\varphi$  to each canonical copy of  $B_S$  in  $G \otimes B_S$ .

The first step towards transforming a Bounce graph (say, ) into a single edge, consists of contracting the paths of the bounces to a single edge each. For the  $i$ -th bounce, this is just the inverse of an  $s_i$ -stretching applied to each of the  $i$  paths. By (18) of Corollary 1, this “unstretching” gives a factor  $(s_i w^{s_i-1})^i$  to the polynomial, and each edge in the resulting  $(i, 1)$ -bounce receives a weight of  $w/s_i$  in the modified graph. Repeating this process for every bounce simplifies the Bounce graph into a Bounce graph of length  $l$  that is generated by a sequence of 1s (). Let  $\phi(G \otimes B_S)$  denote the graph in which one Bounce graphs

has been simplified. By transitivity, we have the weight shift

$$Z_0(G \otimes B_S; 0, w) = \left( \prod_{i=1}^l (s_i w^{s_i-1})^i \right) \cdot Z_0(\phi(G \otimes B_S); 0, \mathbf{w}'),$$

where  $\mathbf{w}'$  takes the value  $w/s_i$  on every edge of the  $i$ th bounce of the simplified Bounce graph, and the old value  $w$  outside the simplified Bounce graph. Next, we successively replace each of its  $(i, 1)$ -bounces by a single edge to get a simple path  $(\circ \text{---} \circ \text{---} \circ)$  of length  $l$ . This transformation is just an “unthickening” of each  $(i, 1)$ -bounce, and from (19) of Corollary 1 we know that it does not produce any new factors for the polynomial, but the weight of the  $i$ th edge in this path becomes

$$w_i = (1 + w/s_i)^i - 1.$$

Finally, we compress the path into a single edge  $e$ . Then the claim in (20) follows by a single application of Lemma 4. ■

We now show that Bounce inflations provide a rich enough class of weight shifts. The ranges of  $w$  for which we prove this is general enough to allow for interpolation on the whole reliability line, and we make no attempt at extending the ranges.

**Lemma 7.** *Let  $w$  be a rational number with  $w \in (-1, 0)$  or  $w \in (9, \infty)$ . For all integers  $m \geq 1$ , there exist sequences  $S_0, \dots, S_m$  of positive integers such that*

- (i)  $|E(B_{S_i})| \leq O(\log^2 m)$  for all  $i$ , and
- (ii)  $w_{S_i} \neq w_{S_j}$  for all  $i \neq j$ .

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

*Proof.* We consider the set of sequences  $S = \langle s_1, \dots, s_l \rangle$  of length  $l = r \log(m+1)$ , with  $s_i \in \{2, 3\}$  for all  $i$  which are positive integer multiples of  $r$ , and  $s_i = 2$  for all other  $i$ . Here  $r$  is a positive integer and will be chosen later, only depending on  $w$ . Since  $r$  is a constant, this construction satisfies i.

Now consider any two distinct sequences  $S = \langle s_i \rangle$  and  $T = \langle t_i \rangle$ . To show ii, we consider the difference

$$\Delta = \frac{1}{w_S} - \frac{1}{w_T},$$

and show that  $\Delta \neq 0$ .

Using Lemma 6 we get a sum expression for  $\Delta$ .

$$\begin{aligned} \Delta &= \sum_{i=1}^l \frac{1}{(1 + w/s_i)^i - 1} - \sum_{i=1}^l \frac{1}{(1 + w/t_i)^i - 1} \\ &= \sum_{i=1}^l g((1 + w/s_i)^i) - \sum_{i=1}^l g((1 + w/t_i)^i), \end{aligned} \tag{21}$$

where  $g$  is the function  $g(x) = \frac{1}{x-1}$ . This function is negative and strictly decreasing on  $(0, 1)$  and positive and strictly decreasing on  $(1, \infty)$ . It is convenient to set  $a, b \in \{(1 + w/3), (1 + w/2)\}$  so that  $a < b$ . By the monotonicity of  $g$ , we have  $g(a^i) > g(b^i)$  for all positive  $i$ .

**Case 1:**  $w > 9$ . Here we have  $a = (1 + w/3)$  and  $b = (1 + w/2)$ . We set  $r = 1$  and let  $k$  be the smallest index for which the sequences differ, i.e.,  $s_k \neq t_k$ . We assume w.l.o.g. that  $s_k = 3$  and  $t_k = 2$ , otherwise we exchange the roles of  $S$  and  $T$ . In (21), terms of the sum for  $i < k$  cancel. The terms corresponding to  $i = k$  are  $g(a^k) - g(b^k) > 0$ . We apply the monotonicity of  $g$  to the terms for  $i > k$ , which allows us to lower bound  $\Delta$  as follows.

$$\Delta \geq g(a^k) + \sum_{i=k+1}^l g(b^i) - g(b^k) - \sum_{i=k+1}^l g(a^i) = f(a) - f(b),$$

where

$$f(x) = g(x^k) - \sum_{i=k+1}^l g(x^i) = \frac{1}{x^k - 1} - \sum_{i=k+1}^l \frac{1}{x^i - 1}. \quad (22)$$

We now claim that  $f$  is strictly decreasing in  $(4, \infty)$ . This implies that  $\Delta > 0$  since  $w > 9$  guarantees  $a, b > 4$ , and we get  $\Delta \geq f(a) - f(b) > 0$ . To prove the claim, we show that the derivative of  $f$  is negative on  $(4, \infty)$ :

$$f'(x) = -\frac{kx^{k-1}}{(x^k - 1)^2} + \sum_{i=k+1}^l \frac{ix^{i-1}}{(x^i - 1)^2}. \quad (23)$$

The terms of the sum here, let us call them  $T_i(x)$ , satisfy

$$T_i(x) > 2 \cdot T_{i+1}(x)$$

for all  $i$  and all  $x > 4$ . To see this, note that the inequality is equivalent to

$$2 \left(1 + \frac{1}{i}\right) x < \left(x + \frac{x-1}{x^i - 1}\right)^2.$$

This statement is true for all reals  $x > 4$  and all positive integers  $i$  since then we have that  $\text{LHS} \leq 4x < x^2 \leq \text{RHS}$ . Thus, for  $x > 4$ , we have

$$f'(x) < \frac{kx^{k-1}}{(x^k - 1)^2} \left(-1 + \sum_{i=k+1}^l \frac{1}{2^{i-k}}\right) < 0,$$

which suffices to prove the claim.

**Case 2:**  $w \in (-1, 0)$ . Here we have  $a = (1 + w/2)$  and  $b = (1 + w/3)$ . We choose  $r$  to be a positive integer that satisfies  $b^r < \frac{1}{4}$ . Let  $rk$  be the smallest index for which the sequences differ, i.e.,  $s_{rk} \neq t_{rk}$ . We assume w.l.o.g. that  $s_{rk} = 3$  and  $t_{rk} = 2$ , otherwise we exchange the roles of  $S$  and  $T$ . In (21), terms of the sum for  $i < rk$  cancel, and so do terms for those  $i$ 's which are not integer multiples of  $r$ . The terms corresponding to  $i = rk$  are  $g(b^{rk}) - g(a^{rk}) < 0$ . We apply the monotonicity of  $g$  to the remaining terms for  $i > rk$ , which allows us to upper bound  $\Delta$  as follows.

$$\Delta \leq g(b^{rk}) + \sum_{i=k+1}^{l/r} g(a^{ri}) - g(a^{rk}) - \sum_{i=k+1}^{l/r} g(b^{ri})$$

For  $x \in (0, 1)$ , we can expand  $g(x)$  into the geometric series

$$g(x) = \frac{1}{x-1} = -\sum_{j=0}^{\infty} x^j.$$

Applying this representation to our estimate for  $\Delta$  and rearranging terms, we arrive at

$$\Delta \leq \sum_{j=0}^{\infty} \left( (a^{rj})^k - (b^{rj})^k + \sum_{i=k+1}^{l/r} ((b^{rj})^i - (a^{rj})^i) \right) = \sum_{j=0}^{\infty} (F(a^{rj}) - F(b^{rj})),$$

where  $F$  is the function

$$F(y) = y^k - \sum_{i=k+1}^{l/r} y^i.$$

We claim that  $F$  is strictly increasing on  $(0, \frac{1}{4})$ . This implies  $\Delta < 0$  since the choice of  $r$  makes sure that  $a^{rj}$  and  $b^{rj}$  are in the range  $(0, \frac{1}{4})$  for all positive integers  $j$ . Thus, since the term for  $j = 0$  is 0 and  $F(a^{rj}) - F(b^{rj}) < 0$  for  $j \geq 1$ , the claim indeed implies  $\Delta < 0$ .

It remains to argue the claim. We show that the derivative of  $F$  is positive.

$$F'(y) = ky^{k-1} - \sum_{i=k+1}^{l/r} iy^{i-1}.$$

We obtain  $F'(y) > 0$  from the following calculation, using the fact that  $y \in (0, \frac{1}{4})$ .

$$\begin{aligned} (ky^{k-1})^{-1} \cdot \sum_{i=k+1}^{l/r} iy^{i-1} &= \sum_{i=k+1}^{l/r} \frac{i}{k} y^{i-k} = \sum_{i=1}^{l/r-k} \left(1 + \frac{i}{k}\right) y^i \\ &\leq \sum_{i=1}^{l/r-k} (1+i) y^i \leq \sum_{i=1}^{\infty} y^i + \sum_{i=1}^{\infty} iy^i \\ &= \frac{1}{1-y} - 1 + \frac{y}{(1-y)^2} \leq \frac{4}{3} - 1 + \frac{4}{9} < 1. \quad \blacksquare \end{aligned}$$

#### Points on the Reliability Line

We prove Theorem 4 (iii).

**Proposition 5.** *Let  $w \neq 0$  be a rational number. If #ETH holds, then  $Z_0(G; 0, w)$  for a given simple graph  $G$  cannot be computed in time  $\exp(o(m/\log^2 m))$ .*

*Proof.* If  $w < 0$ , we can pick a positive integer  $k$  big enough such that

$$w' := w/k > -1.$$

This weight shift corresponds to the  $k$ -stretch of  $G$  (Corollary 1). On the other hand, if  $w > 0$ , we can pick a positive integer  $k$  such that

$$w' := (w/2 + 1)^k - 1 > 9.$$

This is the weight shift that corresponds to the 2-stretch of the  $k$ -thickening of  $G$  (Corollary 1). In any case we can compute  $Z(G; w', q)$  from  $Z(G'; w, q)$ . The graph remains simple after any of these transformations, and the number of edges is only increased by a constant factor of at most  $2k$ .

By the above, we can assume w.l.o.g. that  $w \in (-1, 0)$  or  $w > 9$ . Lemma 7 then allows us to construct  $m + 1$  bounce graphs  $B_S$  such that the corresponding weight shifts  $w_S$  are all distinct by property ii. By Lemma 6, we can compute the values  $Z_0(G; 0, w_S)$  from  $Z_0(G \otimes B_S; 0, w)$ , i.e., we get evaluations of  $v \mapsto Z_0(G; 0, v)$  at  $m + 1$  distinct points. Since the degree of this polynomial is  $m$ , we obtain its coefficients by interpolation. By Proposition 2, these coefficients cannot be computed in time  $\exp(o(m))$  under #ETH. By Lemma 7 i, each  $G \otimes B_S$  has at most  $O(m \log^2 m)$  edges, which implies that  $Z_0(G; 0, w)$  for given  $G$  cannot be computed in time  $\exp(o(m/\log^2 m))$  as claimed. ■

## 7. CONCLUSION AND FURTHER WORK

Our results for the Tutte polynomial leave open the line  $y = 1$  except for the point  $(1, 1)$ , even in the case of multigraphs. That line corresponds to counting the number of forest weighted by the number of edges, i.e.,  $T(G; 1 + 1/w, 1) \sim F(G; w) = \sum_{\text{forests } F} w^{|F|}$ . Thickening and Theta inflation with the analysis in the proof of Lemma 6 suffice to show that every point is as hard as computing the coefficients of  $F(G; w)$  without increasing the number of vertices for multigraphs and with an increase in the number of edges by a factor of  $O(\log^2 m)$  in the case of simple graphs. However, we do not know that computing those coefficients requires exponential time. And of course, it would be nice to improve our conditional lower bounds  $\exp(\Omega(n/\text{poly log } n))$  to match the corresponding upper bounds  $\exp(O(n))$ .

### Acknowledgements

The authors are grateful to Andreas Björklund, Leslie Ann Goldberg, and Dieter van Melkebeek for valuable comments.

### REFERENCES

- AGRAWAL, M. 2006. Determinant versus permanent. In *Proceedings of the 25th International Congress of Mathematicians, ICM 2006*. Vol. 3. 985–997.
- BERKOWITZ, S. J. 1984. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters* 18, 3, 147–150.
- BJÖRKLUND, A. AND HUSFELDT, T. 2008. Exact algorithms for exact satisfiability and number of perfect matchings. *Algorithmica* 52, 2, 226–249.
- BJÖRKLUND, A., HUSFELDT, T., KASKI, P., AND KOIVISTO, M. 2008. Computing the Tutte polynomial in vertex-exponential time. In *Proceedings of the 47th annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*. 677–686.
- BLÄSER, M. AND DELL, H. 2007. Complexity of the cover polynomial. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming, ICALP 2007*. Lecture Notes in Computer Science, vol. 4596. Springer, 801–812.
- BRYLAWSKI, T. 1982. The Tutte polynomial, Matroid theory and its applications. *Centro Internazionale Matematico Estivo*, 125–275.
- CALABRO, C., IMPAGLIAZZO, R., KABANETS, V., AND PATURI, R. 2003. The Complexity of Unique  $k$ -SAT: An Isolation Lemma for  $k$ -CNFs. In *Proceedings of the 18th IEEE Conference on Computational Complexity, CCC 2003*. IEEE Computer Society, 135.

- DAHLHAUS, E., JOHNSON, D. S., PAPADIMITRIOU, C. H., SEYMOUR, P. D., AND YANNAKAKIS, M. 1994. The complexity of multiterminal cuts. *SIAM Journal on Computing* 23, 4, 864–894.
- DELL, H., HUSFELDT, T., AND WAHLÉN, M. 2010. Exponential time complexity of the permanent and the Tutte polynomial. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming, ICALP 2010*. Lecture Notes in Computer Science, vol. 6198. Springer, 426–437.
- FLUM, J. AND GROHE, M. 2006. *Parameterized Complexity Theory*. Springer.
- FORTUIN, C. M. AND KASTELEYN, P. W. 1972. On the random-cluster model: I. Introduction and relation to other models. *Physica* 57, 4, 536–564.
- GIMÉNEZ, O., HLINĚNÝ, P., AND NOY, M. 2006. Computing the Tutte polynomial on graphs of bounded clique-width. *SIAM Journal on Discrete Mathematics* 20, 932–946.
- GODSIL, C. AND ROYLE, G. 2001. *Algebraic Graph Theory*. Graduate Texts in Mathematics. Springer.
- GOLDBERG, L. A. AND JERRUM, M. 2007. The complexity of ferromagnetic Ising with local fields. *Combinatorics, Probability and Computing* 16, 1, 43–61.
- GOLDBERG, L. A. AND JERRUM, M. 2008. Inapproximability of the Tutte polynomial. *Information and Computation* 206, 7, 908–929.
- HOFFMANN, C. 2010. Exponential time complexity of weighted counting of independent sets. In *Proceedings of the 5th International Symposium on Parameterized and Exact Complexity, IPEC 2010*. Lecture Notes in Computer Science, vol. 6478. Springer, 180–191.
- HUSFELDT, T. AND TASLAMAN, N. 2010. The exponential time complexity of computing the probability that a graph is connected. In *Proceedings of the 5th International Symposium on Parameterized and Exact Complexity, IPEC 2010*. Lecture Notes in Computer Science, vol. 6478. Springer, 192–203.
- IMPAGLIAZZO, R., PATURI, R., AND ZANE, F. 2001. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences* 63, 4, 512–530.
- ISTRAIL, S. 2000. 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*. ACM, 87–96.
- JAEGER, F., VERTIGAN, D. L., AND WELSH, D. J. 1990. On the computational complexity of the Jones and Tutte polynomials. *Mathematical proceedings of the Cambridge Philosophical Society* 108, 1, 35–53.
- JERRUM, M. AND SINCLAIR, A. 1993. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing* 22, 5, 1087–1116.
- JERRUM, M. AND SNIR, M. 1982. Some exact complexity results for straight-line computations over semirings. *Journal of the ACM* 29, 3, 874–897.
- KOIVISTO, M. 2009. Partitioning into sets of bounded cardinality. In *Proceedings of the 4th International Workshop on Parameterized and Exact Complexity, IWPEC 2009*. Lecture Notes in Computer Science, vol. 5917. Springer, 258–263.
- KUTZKOV, K. 2007. New upper bound for the #3-SAT problem. *Information Processing Letters* 105, 1, 1–5.
- LAWLER, E. L. 1976. A note on the complexity of the chromatic number problem. *Information Processing Letters* 5, 66–67.
- LINAL, N. 1986. Hard enumeration problems in geometry and combinatorics. *SIAM Journal on Algebraic and Discrete Methods* 7, 2, 331–335.
- PAPADIMITRIOU, C. M. 1994. *Computational Complexity*. Addison-Wesley, Reading, Massachusetts.
- RAZ, R. 2009. Multi-linear formulas for permanent and determinant are of super-polynomial size. *Journal of the ACM* 56, 2, 1–17.
- RYSER, H. J. 1963. Combinatorial mathematics. *Number 14 in Carus Math. Monographs*. Mathematical Association of America.

- SEKINE, K., IMAI, H., AND TANI, S. 1995. 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. Springer, 224–233.
- SOKAL, A. D. 2004. Chromatic roots are dense in the whole complex plane. *Combinatorics, Probability and Computing* 13, 02, 221–261.
- SOKAL, A. D. 2005. The multivariate Tutte polynomial (alias Potts model) for graphs and matroids. In *Surveys in Combinatorics*. London Mathematical Society Lecture Note Series, vol. 327. Cambridge University Press, 173–226.
- TODA, S. 1989. On the computational power of PP and +P. In *Proceedings of the 30th annual IEEE Symposium on Foundations of Computer Science, FOCS 1989*. 514–519.
- VALIANT, L. G. 1979. The complexity of computing the permanent. *Theoretical Computer Science* 8, 2, 189–201.

## A. THE SPARSIFICATION LEMMA

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 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 kernelizations. 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 [Flum and Grohe 2006, Lemma 16.17].

**Lemma 8 (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  $\gamma$  with  $n$  variables, we can find a formula*

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

such that:

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

Furthermore,  $\beta$  can be computed from  $\gamma$  and  $k$  in time  $t \cdot \text{poly}(n)$ .

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

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

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

*Proof (sketch).* We adapt the terminology of [Flum and Grohe 2006, 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  $\gamma$  and a flower  $\alpha = \{\delta_1, \dots, \delta_p\}$ , the original algorithm would branch on the two formulas

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

We modify the branching on the petals to read

$$\gamma_{\text{petals}}^\alpha = \gamma \setminus \{\delta_1, \dots, \delta_p\} \cup \{\delta_1 \setminus \delta, \dots, \delta_p \setminus \delta\} \cup \{ \{-l\} : l \in \delta \}.$$

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  $\gamma$ , all assignments  $\sigma$  to the variables of  $\gamma$ , and all flowers  $\alpha$  of  $\gamma$ ,

- (i)  $\sigma$  satisfies  $\gamma$  if and only if  $\sigma$  satisfies  $\gamma_{\text{heart}}^\alpha \vee \gamma_{\text{petals}}^\alpha$ , and
- (ii)  $\sigma$  does not satisfy  $\gamma_{\text{heart}}^\alpha$  or  $\sigma$  does not satisfy  $\gamma_{\text{petals}}^\alpha$ .

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

- (i)  $\sigma$  satisfies  $\gamma$  if and only if  $\sigma$  satisfies some  $\gamma_i$ , and
- (ii)  $\sigma$  satisfies at most one  $\gamma_i$ .

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  $\gamma_i$  compared to the old one. Furthermore, our construction does not make  $t$  any larger because the REDUCE-step removes all clauses that properly contain  $\{-l\}$  and thus these unit clauses never appear in a flower. ■

*Proof (of Theorem 1).* For all integers  $d \geq 3$  and  $k \geq 1$ , the sparsification lemma gives an oracle reduction from  $\#d\text{-SAT}$  to  $\#d\text{-SAT}$  that, on input a formula  $\gamma$  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  $\#d\text{-SAT}$  that runs in time  $\exp(cm)$ , we can combine this algorithm and the above oracle reduction to obtain an algorithm for  $\#d\text{-SAT}$  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  $\#d\text{-SAT}$  running in time  $\exp(c' \cdot n)$ . This proves that for all  $d \geq 3$ ,  $\#d\text{-SAT}$  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  $\varphi$  of  $\#d\text{-SAT}$  into an instance  $\varphi'$  of  $\#3\text{-SAT}$  that has the same number of satisfying assignments. The formula  $\varphi'$  is constructed as in the standard width-reduction for  $d\text{-CNF}$  formulas, i.e., by introducing a constant number of new variables for every clause of  $\varphi$ . Thus, since the number of clauses of  $\varphi'$  is  $O(m)$ , any clause-subexponential algorithm for  $\#3\text{-SAT}$  implies a clause-subexponential algorithm for  $\#d\text{-SAT}$ . ■

## B. HARDNESS OF 3-COLOURING AND 3-TERMINAL MINCUT

The purpose of this section 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  $\#\text{ETH}$  fails.

**Theorem 6.** *Deterministically computing the problems #NAE-SAT, #MAXCUT, and #3-TERMINAL MINCUT cannot be done in time  $\exp(o(m))$  unless #ETH fails.*

In the following, we formally define the problems, sketch the standard NP-hardness reductions, and provide their analyses as needed to prove Theorem 6.

*Name.* #NAE-SAT

*Input.* 3-CNF formula  $\varphi$ .

*Output.* The number of truth assignments, so that no clause  $\{a, b, c\} \in \varphi$  contains only literals with the same truth value.

**Lemma 9.** *There is a polynomial-time reduction from #3-SAT to #NAE-SAT that maps formulas with  $m$  clauses to formulas with  $O(m)$  clauses.*

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

$$(x \vee \bar{a}) \wedge (x \vee \bar{b}) \wedge (\bar{x} \vee a \vee b) \wedge (x \vee c),$$

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 [Papadimitriou 1994, Theorem 3]). The resulting instance  $\varphi''$  has  $4m$  clauses and  $\#NAE-SAT(\varphi'') = \#3-SAT(\varphi)$ . ■

*Name.* #MAXCUT

*Input.* Simple undirected graph  $G$ .

*Output.* The number of maximum cuts.

A maximum cut is a set  $C \subseteq V(G)$  that maximizes the number  $|E(C, \bar{C})|$  of edges of  $G$  that cross the cut.

**Lemma 10.** *There is a polynomial-time reduction from #NAE-SAT to #MAXCUT that maps formulas with  $m$  clauses to graphs with  $O(m)$  edges.*

*Proof.* We follow the reduction in [Papadimitriou 1994, 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 11.** *There is a polynomial-time reduction from the #MAXCUT problem to #3-TERMINAL MINCUT that maps graphs with  $m$  edges to graphs with  $O(m)$  edges.*

*Proof.* We follow the reduction in [Dahlhaus et al. 1994]. So let  $G = (V, E)$  be a graph with  $n$  vertices and  $m$  edges. It is made explicit in [Dahlhaus et al. 1994] 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 Theorem 6).* 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. ■

### Hardness of Colouring and Other Individual Points on the Chromatic Line

Theorem 4(ii) cannot be handled by the proof of Proposition 3 because thickenings do not produce enough points for the interpolation. Instead, we use Linal's reduction [Linal 1986] along this line. For this, we need to observe that #3-COLOURING is hard under #ETH.

*Name.* #3-COLOURING

*Input.* Simple undirected graph  $G$ .

*Output.* The number of proper vertex-colourings with three colours.

**Lemma 12.** *There is a polynomial-time reduction from the #NAE-SAT problem 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 [Impagliazzo et al. 2001] but without mentioning that it even holds if we use  $m$  instead of  $n$  to measure the size of the instance. For the counting variant, observe that the graph  $G$  that is constructed in [Papadimitriou 1994, Theorem 9.8] from an NAE-SAT-instance  $\varphi$  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( $\varphi$ ). ■

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  $G \mapsto \chi(G; q)$ . The Tutte polynomial specializes to the chromatic polynomial for  $y = 0$ :

$$\chi(G; q) = (-1)^{n(G)-k(G)} q^{k(G)} T(G; 1 - q, 0). \quad (24)$$

We now prove Theorem 4(ii).

**Proposition 6.** *Let  $x \in \{-2, -3, \dots\}$ .*

*If #ETH holds, then  $\text{TUTTE}^{01}(x, 0)$  cannot be computed in time  $\exp(o(m))$ .*

*Proof.* Set  $q = 1 - x$ . We use (24) to see that evaluating the chromatic polynomial  $\chi(q)$  is equivalent to evaluating  $\text{TUTTE}(x, 0)$  if  $q \neq 0$ . Since  $\chi(3)$  is the number of 3-colourings, the case  $q = 3$  cannot be done in time  $\exp(o(m))$  under #ETH, even for simple graphs (cf. App. B). For  $i \in \{1, 2, \dots\}$  and all real  $r$ , Linal's identity is

$$\chi(G + K_i; r) = r(r - 1) \dots (r - i + 1) \cdot \chi(G; r - i), \quad (25)$$

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  $q \in \{4, 5, \dots\}$ , 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  $\chi(q)$  cannot be computed in time  $\exp(o(m))$  under #ETH, even for simple graphs. ■

**Proposition 7.** *Let  $x \notin \mathbb{Q} \setminus \{1, 0, -1, -2, -3, \dots\}$ .*

*If #ETH holds, then  $\text{TUTTE}^{01}(x, 0)$  cannot be computed in time  $\exp(o(n))$ .*

*Proof.* Set  $q = 1 - x$ . We show that  $\text{TUTTE}^{01}(x, 0)$  cannot be computed in time  $\exp(o(n))$  under #ETH. Indeed, with access to  $\chi(q)$ , we can compute  $\chi(G; q - i)$  for all  $i = 0, \dots, n$ , noting that all prefactors in (25) 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  $\chi(3)$  cannot be computed in time  $\exp(o(n))$  under #ETH, this also holds for  $\chi(q)$ , even for simple graphs. ■

The only points on the  $x$ -axis not covered here are  $x \in \{1, 0, -1\}$ . Two of these admit polynomial time algorithms, so we expect no hardness result. By Theorem 4(iii), the Tutte polynomial at the point  $(1, 0)$  cannot be evaluated in time  $\exp(o(m/\log^2 m))$  under #ETH.