The complexity of counting problems

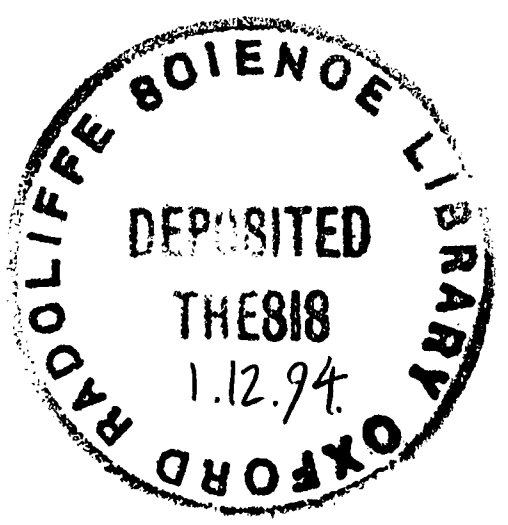
J. D. Annan

University College

Oxford

Submitted for the degree of Doctor of Philosophy

Hilary Term, 1994



The complexity of counting problems

J. D. Annan
University College
Submitted for D. Phil
Hilary Term, 1994

This thesis is concerned with the computational complexity of several counting problems occurring in graph theory, all of which are related directly to the Tutte polynomial. The Tutte polynomial contains as specialisations many fundamental quantities. Among these are the partition function of the Ising and Potts models on the graph and the reliability probability as well as basic combinatorial functions such as the chromatic and flow polynomials.

Polynomial-time computable recurrence relations are given for the Tutte polynomial of the complete graphs and complete bipartite graphs, which can be solved at certain points. The flow polynomials of a particular family of cubic graphs are shown to have arbitrarily large zeros, in contrast to the conjectured behaviour of the zeros of the chromatic and reliability polynomials.

The main results of the thesis are the following:

Theorem *For any positive (constant) integers i, j and k, computing the coefficient of $x^i y^j$ in the Tutte polynomial of an arbitrary graph is* $\#\mathbf{P}$*-complete. However, it can be checked in polynomial time whether the coefficient is less than k.*

The complexity of evaluating the Tutte polynomial modulo a prime is also considered, and some partial results are given.

Turning to approximation, a polynomial-time uniform generator for forests of dense graphs is given. Using this, a fully polynomial randomised approximation scheme (fpras) for the number of forests of a dense graph is created. This leads to the following generalisation:

Theorem *For any rational $x \geq 1$, there exists a fully polynomial randomised approximation scheme for evaluating the Tutte polynomial of dense graphs at the point $(x, 1)$.*

Contents

Acknowledgements

It is with great pleasure that I thank my supervisor, Prof Dominic Welsh. Without his inexhaustible supply of encouragement and inspiration, and a large slice of his time, this thesis would not exist. I am also grateful for Dr Colin McDiarmid's help during my first year of research. Part of Chapter 2 was prompted by computational work of Kyoko Sekine, and I am grateful to her for this.

Thanks are also due to Steve Noble, for helping to catch errors that I had overlooked, and to Pete Cowling and the rest of the Combinatorics group for many interesting tea-time conversations.

My thanks also go to Julia Hargreaves, for being there (when she wasn't somewhere else), and for dragging me up mountains.

Introduction

The aim of this thesis is to examine the complexity of several counting problems of a combinatorial nature. The problems that we consider are linked through their relationship with the Tutte polynomial (a two variable polynomial function of a graph that can be thought of as a generalisation of the chromatic polynomial). When evaluated at particular points or along certain lines, the Tutte polynomial gives many functions of interest. For example, the chromatic and reliability polynomials of a graph can be expressed as evaluations of its Tutte polynomial along certain lines. The partition function of the Ising model is another evaluation of the Tutte polynomial. More examples of these evaluations will be given in Chapter 1.

The thesis is organised along the following lines.

In Chapter 1, we begin by providing a brief introduction to the areas of graph theory and complexity theory that form the background to and the motivation for this thesis. The Tutte polynomial is defined, and its position and significance in both of these fields is briefly explained. The chapter concludes with two apparently new results concerning parsimonious reductions between **NP**-complete languages. These are, firstly, that there is a parsimonious reduction from SAT to 3-COL, and secondly, that there is a weakly parsimonious reduction from SAT to Planar 3-COL. Although, by virtue of their **NP**-completeness, polynomial-time reductions were known, we need the stronger result that the transformations are parsimonious in order to be

able to apply them to the problems that we discuss later.

The second chapter considers some families of graphs for which the Tutte polynomial can be efficiently calculated. We find polynomial-time computable recurrence relations for calculating the Tutte polynomials of the complete graphs $\{K_n\}$, and also for the complete bipartite graphs $\{K_{m,n}\}$. Although we are unable to solve the expressions that we derive in closed form, they can be solved at some particular points of interest to give several Tutte invariants of these graphs. We then look in more detail at the flow polynomial of graphs. By considering the Möbius Ladders, we show that cubic graphs can be found whose flow polynomials have zeros of arbitrarily large modulus. This is in contrast to two conjectures concerning the location of the zeros of the chromatic and reliability polynomials. This work was prompted by Kyoko Sekine's experimental results on the flow polynomial [Sek93].

In Chapter 3, we consider two distinct problems. Firstly, the complexity of calculating individual coefficients of the Tutte polynomial is considered. We find that the coefficients at the high degree end of the polynomial can be calculated in polynomial time, but determining any of the coefficients of the low degree terms is **#P**-complete. However, for any fixed integers i, j and k, it can be determined in polynomial time if the coefficient of the term in $x^i y^j$ is less than k. These results, contained in the first five sections of the chapter, are to be published in [Annb].

Secondly, the complexity of evaluating the Tutte polynomial at particular points modulo a prime p is considered. We conjecture that a similar result to that shown in [JVW90] holds: namely, for any fixed prime $p \geq 3$, evaluation of the Tutte polynomial of a graph modulo p at any point in the integer plane cannot be performed in polynomial time, except at a few known "easy" points. However, we have not been able to prove this. Partial results are given, proving that the conjecture holds for $p = 3$, and that for any other prime $p > 3$, there are many hard points.

Finally, in Chapter 4, we consider the problem of whether it is possible to

efficiently approximately compute functions that we cannot hope to evaluate exactly. The field of approximation algorithms has risen to prominence in the last few years with interest following from the work of Jerrum and Sinclair on approximating both the permanent of dense matrices, and the partition function of the Ising model. We explain the notion of a randomised Turing machine and what we mean by an efficient randomised approximation algorithm. We consider two counting problems again that arise naturally from the Tutte polynomial.

Firstly, we show that it is possible to approximate the number of forests of dense graphs efficiently. This is generalised to show that the Tutte polynomial of such graphs can be approximated at the point $(x, 1)$ for any rational $x \geq 1$. These results (contained in Sections 4.1-4.6) have been accepted for publication in [Anna].

Secondly, we consider the complexity of counting the number of unicyclic subgraphs of a graph, which was posed by Colbourn [Col93]. We show that this number can be efficiently approximated. Our method generalises to give approximation algorithms for the number of forests with a constant number of components. The complexity of exact counting for these problems remains open.

Chapter 1

Complexity Theory and the Tutte polynomial

We begin by providing a brief introduction to some of the areas of graph theory and complexity theory that will be investigated later in this thesis. We will then define the Tutte polynomial, and explain its position with respect to both of the above fields. The final section of the chapter contains two apparently new results on parsimonious reductions, which are much used in later chapters.

1.1 Basic ideas in graph theory

We will assume some familiarity with the concept of a graph $G = (V, E)$. The interested reader is referred to Wilson's book [Wil85] for introductory details.

We give here some definitions that will be used frequently throughout this thesis.

Definition 1.1.1 *A* tree *is a connected, acyclic subgraph, and a* spanning tree *is a tree containing all the vertices of* G.

Definition 1.1.2 *A* forest *is a (not necessarily connected) acyclic subgraph, i.e. a subgraph whose connected components are themselves trees.*

We will only be considering undirected graphs, but the edges of G can, if we wish, be oriented.

Definition 1.1.3 *An orientation is* acyclic *if it contains no oriented cycle.*

Definition 1.1.4 *$k(G)$ will denote the number of connected components of G, and we shall use $k(A)$ to denote the number of components of $G : A = (V, A)$, for any $A \subseteq E(G)$.*

Note that this definition counts even isolated vertices as components.

There are three single variable functions of a graph that we shall frequently encounter:

Definition 1.1.5 *For any $\lambda \in \mathbb{N}$, a (proper) λ-colouring of G is a function $f : V \to \{0, \ldots, \lambda - 1\}$ such that, for any adjacent vertices u and v, $f(u) \neq f(v)$.*

Definition 1.1.6 *$P(G; \lambda)$ is defined as the function that maps $\lambda \in \mathbb{N}$ to the number of λ-colourings of G.*

Definition 1.1.7 *Let H be an Abelian group, and w be an orientation of the edges of a graph G. A (nowhere-zero) H-flow of G is a mapping $\phi : E \to H \backslash \{0\}$ such that, at each vertex of G,*

$$\sum_{e \in \delta^+(v)} \phi(e) - \sum_{e \in \delta^-(v)} \phi(e) = 0$$

where $\delta^+(v)$ ($\delta^-(v)$) denote the edges oriented into (out of) v respectively.

It will be shown later that the number of H-flows of any graph G with respect to a particular fixed orientation w is not only independent of w, but also depends only on the order of the group, and not its structure. Thus we can say a graph G has a nowhere-zero t-flow if, for any Abelian group H of order t, G has a nowhere-zero H-flow.

Definition 1.1.8 *$F(G;\lambda)$ is defined as the function that maps $\lambda \in \mathbb{N}$ to the number of nowhere-zero λ-flows of G over an Abelian group of order λ.*

Definition 1.1.9 *$R(G;p)$ is the function that gives the reliability of a connected graph G: that is, the probability (as a function of p) that a graph G remains connected, if each edge of G is removed from the edge-set E with probability $q = 1 - p$, where $0 < p < 1$, independently of all the other edges.*

We will show in Section 1.3 that all three of these functions are in fact polynomials with integer coefficients, and so we call them the chromatic, flow and reliability polynomials respectively.

1.2 A brief introduction to complexity theory

Complexity theory can be succinctly described as the study of the tractability of problems, and a good introduction to the field can be found in Garey and Johnson's book [GJ79]. We will give a brief introduction to the ideas in complexity theory that we will explore later in this thesis.

We call an algorithm *polynomial time*, if the running time of the algorithm on any input of size n is bounded by a polynomial in n.

A *language* L is a set of finite strings over some finite alphabet Σ. We let $|x|$ denote the length of a string x, and Σ^n is the set of all strings of symbols of

Σ having length n. The "easy" languages are those which can be recognised in polynomial time:

Definition 1.2.1 P *is the set of languages that can be recognised in polynomial time by a deterministic Turing machine.*

Sometimes we do not wish to just make a simple yes/no decision, but instead compute a function. We can use a Turing machine to do this, if we take the contents of its work tape as its output whenever it halts (if it does so). The function is undefined on any input for which the Turing machine fails to halt. We therefore define the class of "easy" functions, which we call **FP**, as follows.

Definition 1.2.2 FP *is the class of all functions f that are computable in polynomial time.*

We will use the "witness" definition of nondeterministic polynomial time (**NP**):

Definition 1.2.3 NP *is the set of languages L for which there exists a relation $R \in \mathbf{P}$, and polynomial p for which the following two conditions hold:*

i. If $x \in L$, then $(x, w) \in R$ for at least one string $w \in \Sigma^{p(|x|)}$.

ii. If $x \notin L$, then $(x, w) \notin R$ for all $w \in \Sigma^{p(|x|)}$.

In the case $x \in L$, we call a string $w \in \Sigma^{p(|x|)}$ for which $(x, w) \in R$ a *witness* for x.

Definition 1.2.4 *The class* **RP** *(random polynomial time) is defined by replacing "at least one string" by "at least half of all strings" in condition (i) of the definition of* **NP**.

The intuition behind this definition is that, if we wish to show that x is a member of L for some $L \in \mathbf{RP}$, then a string chosen uniformly at random amongst all strings of the correct length will serve as a witness with probability at least $1/2$.

There are two commonly occurring notions of reduction betweens problems, the *polynomial (many-one) reduction* and the *Turing reduction.*

Definition 1.2.5 *A language L_1 is polynomially reducible to language L_2 if there exists a polynomially computable function f such that,*

$$\forall x \in \Sigma^*,\ f(x) \in L_2 \Leftrightarrow x \in L_1.$$

We write this as $L_1\ \alpha\ L_2$.

Such a function f is called a polynomial reduction from L_1 to L_2.

Definition 1.2.6 *A problem or language P_1 is Turing reducible to problem or language P_2 if there is a polynomial-time Turing machine that solves P_1, which makes a polynomially-bounded number of calls to a subroutine (or oracle) which solves P_2. We write this as $P_1\ \alpha_T\ P_2$.*

We can now define the concept of "hardness" and "easiness" for **NP**, which normally is taken to mean hardness and easiness with respect to Turing reductions.

Definition 1.2.7 *A problem or language P_1 is* **NP***-hard if, for all $L \in \mathbf{NP}$, L is Turing reducible to P_1. A problem or language P_2 is* **NP***-easy if, for some $L \in \mathbf{NP}$, P_2 is Turing reducible to L.*

It is usual to define the **NP**-complete problems using the more restrictive polynomial reductions:

Definition 1.2.8 *A language* $L_1 \in \mathbf{NP}$ *is* **NP**-complete *if, for all* $L_2 \in \mathbf{NP}$, L_2 *is polynomially reducible to* L_1.

Two well-known **NP**-complete problems that will be of interest to us are:

SAT The set of all finite sets of finite clauses of boolean variables or their negations, for which there is an assignment of values to the variables that simultaneously sets at least one variable in each clause to be true, and

3-COL The set of all finite graphs whose vertices can be properly coloured with three colours.

The definition of the class **NP** suggests the following "hard" function class, which contains the hardest functions that we will consider in this thesis. This function class was first suggested by Valiant in 1979 [Val79].

Definition 1.2.9 **#P** *is the class of functions that compute the size of the witness set of some language* $L \in \mathbf{NP}$ *with respect to a particular binary relation* R *for* L.

It should be emphasised that, for any $L \in \mathbf{NP}$, there are many different relations R for L, and the size of the witness set will depend on which R we choose. Usually, there is a "natural" R which we will assume is used. For instance, for an instance of SAT, we will use the relation that takes as witness a truth assignment for the variables of the instance, given in some canonical order, and checks that this assignment indeed satisfies the set of clauses. The number of witnesses here is therefore equal to the number of satisfying truth assignments.

In contrast to the situation regarding **NP**, it is common practice to use Turing reductions in the definition of completeness for **#P**:

Definition 1.2.10 *A function f is* **#P**-complete *if $f \in$ **#P**, and for any $g \in$ **#P**, $g \ \alpha_T \ f$.*

It is straightforward to show that **#P** has complete members, such as #SAT, the function that counts the number of satisfying truth assignments for an instance of SAT. Valiant's original paper showed that the counting functions associated with some of **NP**-complete languages were **#P**-complete.

If a polynomial reduction can be found from SAT to another language $L \in$ **NP** that preserves the size of the witness set, then this will immediately show the **#P**-completeness of counting the number of witnesses for L. We call such a reduction a *parsimonious reduction*, and define it formally as follows:

Definition 1.2.11 *A polynomial reduction f from L_1 to L_2 is parsimonious if, for all $x \in \Sigma^*$ the following holds:*

$$\left|\{w \in \Sigma^{p_1(|x|)} : \ (x, w) \in R_1\}\right| = \left|\{w' \in \Sigma^{p_2(|f(x)|)} : \ (f(x), w') \in R_2\}\right|$$

where R_1 and R_2 are the polynomial-time binary relations of L_1 and L_2 respectively, and p_1 and p_2 are polynomials that give the sizes of the witnesses.

A large subset of the known **NP**-complete languages, including SAT, are known to have parsimonious reductions amongst themselves (and therefore their associated counting problems are all **#P**-complete), and numerous examples of these appear in [Sim77]. However, some **NP**-complete languages are not thought to be be parsimoniously reducible to SAT, at least using the "natural" relation and witness, as we shall soon see.

One reason for our interest in the class of languages $L \in$ **NP** that are parsimoniously reducible from SAT is because of a result of Valiant and Vazirani, who show that, for any such language L, it is probably hard to determine if

a string x is contained in the language, even if it is known that there is at most one witness for x.

The main result of their paper is the following:

Theorem 1.2.12 (Valiant, Vazirani [VV86]) *Let A be any* **NP***-complete problem to which SAT is parsimoniously reducible. Let $\#A(x)$ denote the number of solutions [witnesses] to instance x. For each Boolean predicate Q, define the problem UA_Q by*

$$UA_Q = \begin{cases} 0 & \text{if } \#A(x) = 0 \\ 1 & \text{if } \#A(x) = 1 \\ Q(x) & \text{if } \#A(x) > 1. \end{cases}$$

If $UA_Q \in \mathbf{RP}$ for any Q, then $\mathbf{NP} = \mathbf{RP}$.

The significance of this theorem becomes clearer when it is paraphrased in the following way.

Let $A \in \mathbf{NP}$ be a language to which SAT is parsimoniously reducible. If there exists a randomised polynomial-time algorithm which gives the correct number of witnesses for all instances of A having 0 or 1 witnesses, and is arbitrary on all other inputs, then $\mathbf{NP} = \mathbf{RP}$.

It is generally thought that $\mathbf{NP} \neq \mathbf{RP}$, so this result means that we cannot hope to find such an algorithm. Conversely, if it is easy to decide whether a string has a single witness of membership in a language $L \in \mathbf{NP}$, then this theorem provides evidence that SAT is not parsimoniously reducible to L.

As an example, let us consider the language Edge k-COL, which is the set of all k-regular graphs whose edges can be coloured using k colours so that no incident edges are the same colour. This problem was shown to be **NP**-complete for $k = 3$ by Holyer [Hol81], and Leven and Galil extended this to the general case $k \geq 4$ [LG83], but Thomason proved that, for $k \geq 4$, the

only graphs with unique edge k-colourings are the star graphs $K_{1,k}$ [Tho78], and these graphs can of course be identified in polynomial time. Hence SAT is unlikely to be parsimoniously reducible to Edge k-COL, for any $k \geq 4$.

1.3 The Tutte polynomial

The Tutte polynomial of a graph, introduced by Tutte in 1947 [Tut47], is a two variable generating function which contains a great deal of information about the graph. It is defined in the following manner.

Let G be a graph with edge set E. For any subset A of E we define the *rank* $r(A)$ by

Definition 1.3.1 *The* rank *of a set of edges $A \subseteq E$ is given by*

$$r(A) = |V(G)| - k(A),$$

where $|V(G)|$ is the number of vertices of G, and using $k(A)$ as defined earlier.

The *Tutte polynomial of* G, which we write as $T(G; x, y)$, is defined in the following manner.

Definition 1.3.2 *If G is edgeless then $T(G; x, y) = 1$, otherwise*

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

where the sum is over all subsets of E.

We will often abbreviate $T(G; x, y)$ to $T(G)$. It is clear that the degree of this polynomial is $|V| - k(G)$ in x , and $|E| - |V| + k(G)$ in y.

If our graph G is planar, then we can form the planar dual, G^*. It is straightforward to show that

$$T(G;x,y) = T(G^*;y,x). \tag{1.3.3}$$

The Tutte polynomial can also be calculated recursively in the following manner:

If we write G'_e (G''_e) for the graphs obtained by deleting (contracting) an edge e then it is easy to see that, for any edge e that is neither an isthmus (edge e such that G'_e is disconnected) nor loop,

$$T(G) = T(G'_e) + T(G''_e). \tag{1.3.4}$$

When e is an isthmus or loop we use

$$T(G) = \begin{cases} xT(G''_e) & \text{if } e \text{ is an isthmus} \\ yT(G'_e) & \text{if } e \text{ is a loop} \end{cases} \tag{1.3.5}$$

respectively.

Interest in calculating the Tutte polynomial, or evaluating it at certain points, lies in the fact that, at different points, it evaluates to many other graph-theoretic functions. For instance, it is immediate from the formula given in Definition 1.3.2 that, at the point (1,1), the only terms which count towards the sum are those sets A for which $|A| = r(A) = r(E)$. For a connected graph G, these are the sets of edges that are spanning trees of G. If we denote the number of spanning trees of G by $\kappa(G)$, then we see that

$$T(G;1,1) = \kappa(G). \tag{1.3.6}$$

There are many other such specialisations of the Tutte polynomial, as can be shown by use of the following theorem.

Theorem 1.3.7 (Oxley and Welsh [OW79]) *For any function f that is well-defined on all graphs, and that satisfies*

$$f(G) = af(G'_e) + bf(G''_e)$$

whenever e is neither a loop nor isthmus, and

$$f(G) = \begin{cases} x_0 f(G''_e) & \text{if } e \text{ is an isthmus} \\ y_0 f(G'_e) & \text{if } e \text{ is a loop} \end{cases},$$

then f is given by

$$f(G) = a^{|E|-r(E)} b^{r(E)} T(G; \frac{x_0}{b}, \frac{y_0}{a})$$

Such a function f is known as a *Tutte-Gröthendieck invariant.*

This result can be used in a straightforward manner to give the following interpretations of the Tutte polynomial. For any connected graph G:

The number of forests of G, is given by $T(G; 2, 1)$. (1.3.8)

The number of connected spanning sets of G is given by $T(G; 1, 2)$. (1.3.9)

The number of acyclic orientations of G is given by $T(G; 2, 0)$. (1.3.10)

The chromatic polynomial $P(G; \lambda)$ is given by

$$P(G; \lambda) = (-1)^{r(E)} \lambda^{k(G)} T(G; 1 - \lambda, 0). \quad (1.3.11)$$

The flow polynomial $F(G, \lambda)$ is given by

$$F(G; \lambda) = (-1)^{|E|-r(E)} T(G; 0, 1 - \lambda). \quad (1.3.12)$$

The reliability polynomial $R(G; p)$ of a connected graph G is given by

$$R(G; p) = q^{|E|-r(E)} p^{r(E)} T(G; 1, q^{-1}) \quad (1.3.13)$$

where $q = 1 - p$.

These equations justify what we claimed at the end of Section 1.1 about the chromatic, flow, and reliability polynomials: they are indeed polynomials, with integer coefficients.

If we write H_α for the hyperbola

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

then we have the following evaluations.

Along the hyperbola H_1, the Tutte polynomial simplifies to

$$T(G; x, y) = x^{|E|}(x - 1)^{r(E)-|E|}.$$

The Ising model and Potts model are classical models of statistical physics. For introductions to them and details of the notation, see [Cip87] and [Wu82] respectively. Along the hyperbola H_2, we have

$$Z(G) = (2e^{-\beta J})^{|E|-r(E)}(4 \sinh \beta J)^{r(E)} T(G; \coth \beta J, e^{2\beta J}) \qquad (1.3.14)$$

where $Z(G)$ is the partition function of the Ising model for constant interaction energy J and no external field, and $\beta = 1/kT$ is a parameter determined by the absolute temperature T and Boltzmann's constant k . Along the hyperbola H_q, for any positive integer q, we get

$$Z_{\text{Potts}}(G) = q(e^K - 1)^{|V|-1} e^{-K|E|} T(G; \frac{e^K + q - 1}{e^K - 1}, e^K) \qquad (1.3.15)$$

where $Z_{\text{Potts}}(G)$ is the partition function of the q-state Potts model and $K = 2\beta J$.

Some more interpretations of the Tutte polynomial, and futher explanation of those given here, can be found in [Wel93a].

1.4 The complexity of Tutte invariants

It was shown by Linial [Lin86] that computing the number of 3-colourings of a graph is **#P**-complete, and so computing the entire Tutte polynomial must also be **#P**-complete. However, it is possible to evaluate the Tutte polynomial at certain points in the (x, y) plane. For instance, the well-known determinantal formula [Big74, page 34] gives a method for counting the number of spanning trees of a graph in polynomial-time, and the number of 2-colourings of a graph with k connected components is $2^{k(G)}$ if G is bipartite, and 0 otherwise. The following theorem of Jaeger *et al.* fully characterises the complexity of all points in the (x, y) plane.

Theorem 1.4.1 (Jaeger, Vertigan, Welsh [JVW90]) *It is* **#P***-hard to evaluate the Tutte polynomial of a graph at a particular point* (a, b) *of the complex plane except when either*

1. *the point lies on the hyperbola* H_1, *or*
2. *the point is one of the special points* $(1, 1)$, $(-1, 0)$, $(0, -1)$, $(-1, -1)$, $(i, -i)$, $(-i, i)$, (j, j^2), (j^2, j) *where* $j = e^{2\pi i/3}$.

At these special points, the evaluation can be carried out in polynomial time.

At each of the easy points listed above, there is a well-known combinatorial interpretation of the Tutte polynomial.

Vertigan [Ver91] strengthened the result above, showing that it is **#P**-complete to evaluate the Tutte polynomial even for planar graphs at any point (a, b) unless (a, b) is either one of the exceptions given above, or lies on H_2. The Tutte polynomial of a planar graph can be computed in polynomial time along the hyperbola H_2 using Kasteleyn's algorithm for the Ising partition function of a planar graph [Kas67]. Vertigan's result was further strengthened by Vertigan and Welsh [VW92], who showed that it holds even for the class of bipartite planar graphs.

1.5 k-COL is parsimonious with SAT

In [Lin86], Linial demonstrates the #P-completeness of counting the number of 3-colourings of a graph. However, he does not do this by establishing a parsimonious reduction, but instead via a Turing reduction. It is crucial for later results that we have a parsimonious reduction from SAT to k-COL, and so we will give such a reduction here.

For any graph that contains an edge, the number of possible 3-colourings (as we have defined) will always be a multiple of 6, since any permutation of the colours will result in a different 3-colouring. So if we take as our "witness" w a list of the colours assigned to the vertices by a 3-colouring of the graph, then we cannot hope for a parsimonious reduction from SAT as it is straightforward to construct an instance of SAT with only one satisfying truth assignment.

To avoid this problem, we wish to only count the 3-colourings that are "essentially distinct" from each other. By this, we mean that no two of these colorings induce the same partition of the vertices (into colour classes) and so do not just arise from each other by permuting the colours. In order to achieve this, we will use as our witness a partition of the vertices into three sets, and our binary relation R will accept only if the endpoints of each edge are in different sets of the partition, and the partition is lexicographically ordered (i.e. the vertices in each of the three sets are given in order, and the three sets are also given in the order of their first members). It is clear that the language defined by this relation is precisely the set of 3-colourable graphs, and the number of witnesses for a graph G will be the number of essentially distinct 3-colourings of G (counting a permutation of the colours as the same colouring).

3-SAT is the language consisting of all finite sets of clauses, each containing exactly 3 variables, for which there is a satisfying truth assignment. It is known that counting the number of satisfying truth assignments for an

instance of 3-SAT is #**P**-complete: a parsimomious reduction from SAT to 3-SAT is given in [Val79]. We will give a parsimonious reduction from 3-SAT to 3-COL: it is easy to check that the composition of parsimonious reductions is parsimonious, and so this will prove that SAT is parsimoniously reducible to 3-COL.

Proposition 1.5.1 *3-SAT is parsimoniously reducible to 3-COL.*

Proof: The reduction from 3-SAT to 3-COL given in [GJS76] is not parsimonious, but can be modified to make it so.

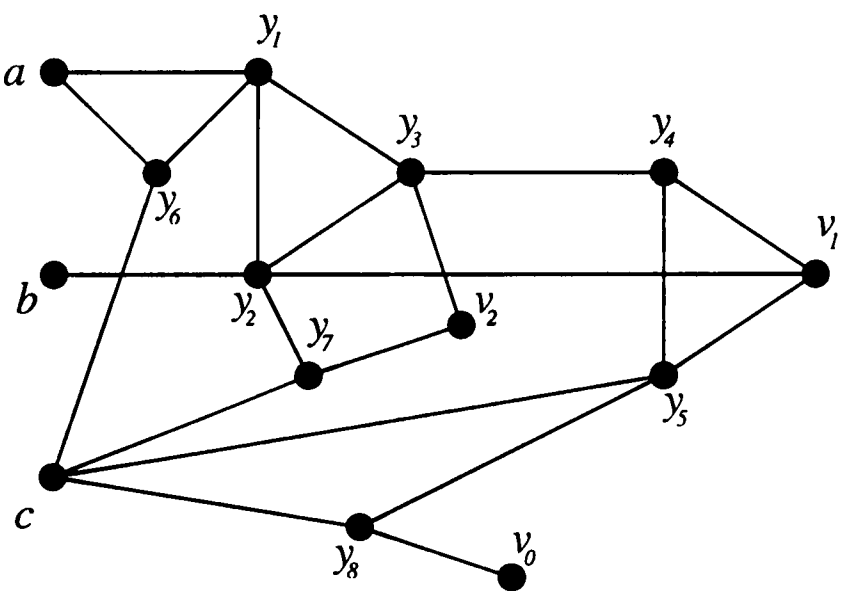


Figure 1.A: A "satisfaction-checking" graph, H.

The graph H shown in Figure 1.A is crucial in the proof. It is based upon a graph described in [GJS76]. It has two important properties.

1. If the nodes v_0, v_1, v_2 are coloured 0,1 and 2 respectively, then for any colouring of the nodes a, b and c such that at least one of these nodes is coloured 1 and the others (if any) are coloured 0, it is possible to extend this *uniquely* to a 3-colouring of H.

2. If a, b and c are all coloured 0, then there is no way of properly 3-colouring the remaining vertices of H so that v_1 is coloured 1.

These properties can easily be checked.

Let $S = \{S_i : 1 \leq i \leq r\}$ be a set of clauses on variables $\{u_j : 1 \leq j \leq s\}$, in which each clause contains exactly three literals (variables or negations of variables). We label the literals of clause S_i by a_i, b_i, and c_i.

We now describe the graph G as follows.

The set of vertices V of G is given by

$$V = \{v_0, v_1, v_2\} \bigcup \{x_j, \tilde{x}_j : 1 \leq j \leq s\} \bigcup \{y_{ij} : 1 \leq i \leq r,\ 1 \leq j \leq 8\}$$

The set E of edges is given by

$$E = E1 \bigcup E2 \bigcup E3$$

where $E1$, $E2$ and $E3$ are as described below.

We let $E1$ be the set of edges $\{(v_i, v_j) :\ i \neq j\}$.

$E2$ is the set of edges $\{(x_j, \tilde{x}_j),\ (x_j, v_2),\ (\tilde{x}_j, v_2) :\ 1 \leq j \leq s\}$.

$E3_i$ is the set of edges $\{(a_i, y_{i1}),\ (a_i, y_{i6}),\ (b_i, y_{i2}),\ (c_i, y_{i5}),\ (c_i, y_{i6}),\ (c_i, y_{i7}),$ $(c_i, y_{i8}),\ (y_{i1}, y_{i2}),\ (y_{i1}, y_{i3}),\ (y_{i1}, y_{i6}),\ (y_{i2}, y_{i3}),\ (y_{i3}, y_{i4}),\ (y_{i3}, v_2),\ (y_{i2}, v_1),$ $(y_{i2}, y_{i7}),\ (y_{i7}, v_2),\ (y_{i4}, y_{i5}),\ (y_{i4}, v_1),\ (y_{i5}, v_1),\ (y_{i5}, y_{i8}),\ (y_{i8}, v_0)\ \}$.

The set $E3$ is given by

$$E3 = \bigcup_{1 \leq i \leq r} E3_i.$$

Note that the subgraph of G induced by $E3_i$ is a copy of H.

We claim that G has exactly the same number of essentially distinct 3-colourings as S has satisfying truth assignments.

To establish this we observe that, firstly, the edges contained in $E1$ ensure that we colour each of the vertices v_0, v_1, v_2 differently, and so without loss of generality we will assume from now on that they are coloured 0,1,2 respectively. For each j, the edges in $E2$ ensure that exactly one of x_j and $\tilde{x}_j$ is

coloured 1, and the other is coloured 0. The conditions imposed by the edges in $E3_i$, as given above, together with the conditions imposed by $E1$ and $E2$, ensure that, in any 3-colouring of G, there is at least one literal from clause i that is coloured 1. There is a natural bijection between satisfying truth assignments for the variables of S, and 3-colourings of G, given by setting a variable u_j to be true in S if the vertex x_j is coloured 1 in a 3-colouring of G, and false if x_j is coloured 0. That this is indeed a bijection is clear from the properties of H.

Thus G has exactly the same number of distinct 3-colourings as S has satisfying truth assignments, and so we have proved that 3-SAT is parsimoniously reducible to 3-COL. □

Corollary 1.5.2 *3-SAT is parsimoniously reducible to k-COL, for any integer $k \geq 3$, where k can either be a constant, or be given as part of the input.*

Proof: We can parsimoniously reduce the problem of 3-colouring the graph G (constructed above) to k-colouring a graph G'. The graph G' is formed by taking G and a clique on $k-3$ vertices, and adding an edge from every vertex of G to every vertex of the clique. It is clear that in any k-colouring of G', the vertices of G are 3-coloured, and any 3-colouring of G can be extended to a k-colouring of G'. Moreover, the reduction is clearly parsimonious (up to permutations of the colours) as the $k-3$-clique can only be coloured essentially one way, and we can use as the witness an ordered partition of the vertices into colour classes as above. □

Note also that we know that the graph G' is not $k-1$-colourable, since G is not 2-colourable as it contains a triangle. This fact will be of use to us in Chapter 3.

Now we are able to use Valiant and Vazirani's result to show:

Proposition 1.5.3 *If there exists a polynomial-time algorithm that counts the number of essentially different 3-colourings of a graph modulo 3, then* $\mathbf{NP} = \mathbf{RP}$.

Proof: If we know that a graph has either zero or one 3-colouring, then counting the number of 3-colourings of a graph modulo 3 will tell us which of these two possibilities is in fact the case. As we have shown that SAT is parsimoniously reducible to 3-COL, Valiant and Vazirani's theorem tells us that if some polynomial-time algorithm can do this, then $\mathbf{NP} = \mathbf{RP}$. □

The planar case

Another result that we now strengthen is the result of Vertigan that counting the number of 3-colourings of a planar graph is #**P**-complete [Ver91]. His proof uses a Turing reduction, but we would like to be able to use Valiant and Vazirani's theorem again to strengthen Proposition 1.5.3 to the case of planar graphs.

A polynomial reduction is given by Garey, Johnson and Stockmeyer [GJS76] from 3-COL to Planar 3-COL (the set of planar graphs that are 3-colourable) that proves that deciding membership in Planar 3-COL is **NP**-complete. Unfortunately, their proof does not suffice to prove the #**P**-completeness of counting the number of 3-colourings, as their reduction is not a parsimonious one, and there does not appear to be any way of recovering the number of 3-colourings of the original graph from the number of 3-colourings of the planar graph to which it is reduced. Here we give a simple modification of their reduction. This modified reduction ensures that the size of the witness sets are related in an easily computable way. We call such a reduction *weakly parsimonious*, and define it formally as follows:

Definition 1.5.4 *A polynomial-time reduction f from L_1 to L_2 is* weakly parsimonious *if, for all $x \in \Sigma^*$, the following holds:*

$$|\{w \in \Sigma^* : (x, w) \in R_1\}| = g\left(|\{w' \in \Sigma^* : (f(x), w') \in R_2\}|\right)$$

where g can be calculated in polynomial time, and R_1 and R_2 are the polynomial-time binary relations of L_1 and L_2 respectively.

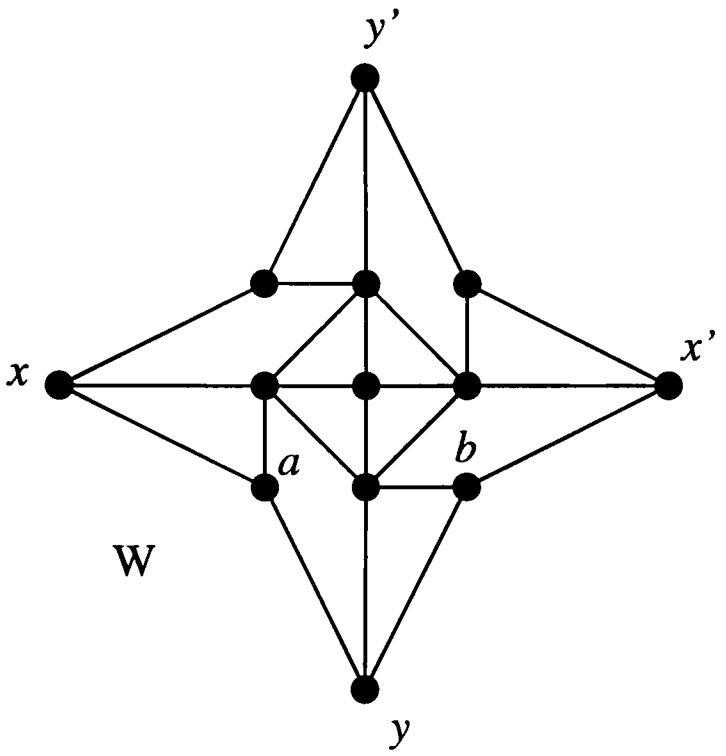
This means that it is possible to compute in polynomial time the number of witnesses for an instance of L_1 if we can count the number of witnesses for a single instance of L_2, so, if it is **#P**-complete to count the number of witnesses for L_1, it must also be **#P**-complete to count the number of witnesses for L_2.

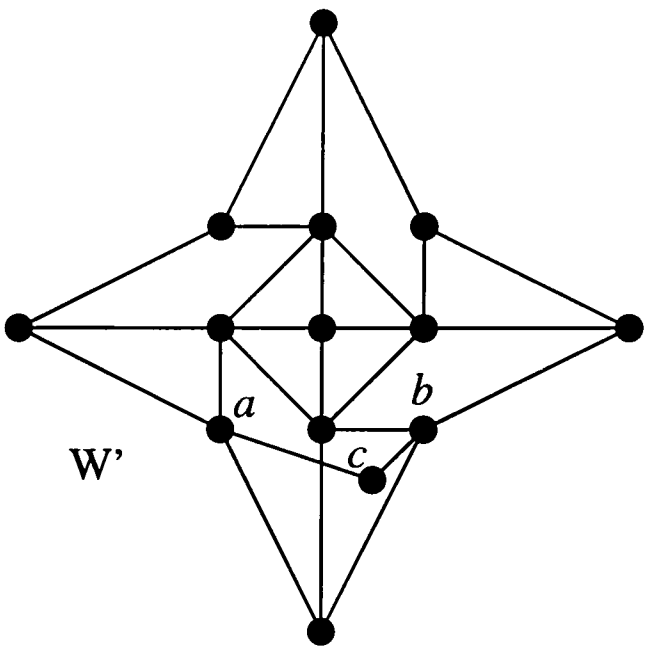
Proposition 1.5.5 *There exists a weakly parsimonious reduction from 3-COL to Planar 3-COL.*

Proof: We use a "gadget", which we can use to eliminate crossings in a representation of a non-planar graph G drawn in the plane. Thus we construct a planar graph H, and we ensure that the number of 3-colourings of G is related in a simple manner to the number of 3-colourings of H.

Firstly, we consider the gadget of Garey, Johnson and Stockmeyer, which is shown as W in Figure 1.B.

It is readily checked that, in any 3-colouring of the vertices of W, x and x' must be given the same colour (the x-colour), and y and y' must also be given the same colour (the y-colour). However, the x- and y-colours can either both be the same, or different. This graph can be used to eliminate edge crossings, as described in [GJS76], in such a way that the planar graph that is constructed is 3-colourable if and only if the original graph was 3-colourable.





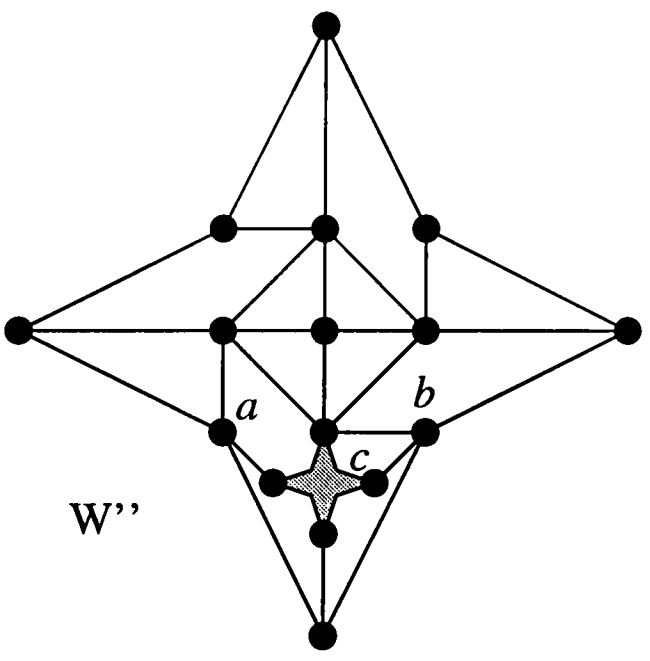


Figure 1.B: The planar crossover gadgets, showing the construction of W'' from W.

However, it can easily be checked that, if the x- and y-colours are the same, there are two ways of 3-colouring the remainder of the graph (given by permuting the colours not used for the corner vertices), whereas, if they are different, then there is a unique way of colouring the rest of W. Permuting the colours in the interior of W of course does not result in an essentially different colouring of W alone, but it will do so when W is inserted into a larger graph.

To form our new planar crossover graph, we firstly add an extra vertex c, with edges (a, c) and (c, b), to get a new graph W'. If the x- and y-colours are the same, then, for any 3-colouring of the vertices in W, there is only one possible colour for c (as a and b will have to take different colours), but if the x- and y-colours are different, then there will be two choices for c, as a and b will then be coloured the same. Therefore, whatever the x- and y-colours are, there are exactly two ways of colouring the remaining vertices in W'. However, we cannot use it as it stands to eliminate edge crossings, for the extra edges we have added will make the resulting graph non-planar. We can solve this problem by inserting another copy of W into W' to form W'', as shown in the diagram, noting that, however we try to colour W', the colours of the adjacent corners of the second copy of W will be different from each other.

This means that there are exactly 2 ways of 3-colouring our planar crossover W'', for any choice of x- and y-colours. So now we can use W'' to eliminate edge crossings, as in the original proof of the **NP**-completeness of deciding 3-colourability of planar graphs.

Noting that the number of essentially distinct 3-colourings of a graph is given by $\frac{1}{6}P(G;3)$ (and of course this number is always an integer), we see that

$$\frac{1}{6}P(G';3) = 2^c \cdot \frac{1}{6}P(G;3) \tag{1.5.6}$$

for any graph G, where c is the number of crossings in the original representation of G in the plane, and G' is a planar graph that can easily be

constructed from G. This completes the proof. □

Now, although our reduction to Planar 3-COL is not parsimonious, we can prove the same result as in the general case about counting modulo 3:

Proposition 1.5.7 *If there exists a polynomial-time algorithm that counts the number of essentially different 3-colourings of a planar graph modulo 3, then* $\mathbf{NP} = \mathbf{RP}$.

Proof: Given any graph G, we can construct a planar graph G' that has exactly 2^c times as many different 3-colourings as G, using our construction above. Thus we have

$$\left[\frac{1}{6}P(G';3)\right] = 2^c \left[\frac{1}{6}P(G;3)\right]$$

or, reducing both sides modulo 3,

$$\left[\frac{1}{6}P(G';3)\right] \equiv (-1)^c \left[\frac{1}{6}P(G;3)\right] \pmod 3.$$

As c is easily computed, it is obvious that if we can count the number of 3-colourings of G' modulo 3, we can obtain the number of 3-colourings of G modulo 3, but we have already shown that if we can do this in polynomial time, then $\mathbf{NP} = \mathbf{RP}$. □

We have been unable to find a parsimonious reduction from 3-COL to Planar 3-COL. It is easy to see that any attempt to construct a gadget that can be used to eliminate edge-crossings in a similar manner to the weakly parsimonious reduction that we have just given, must be doomed to failure: if the four "corner" vertices of the gadget are given the same colour, then there must be at least 2 ways of extending the colouring to the whole gadget (given by swapping the other two colours). So if any parsimonious reduction exists, it must be more complex. This suggests the following problem:

Open Problem 1.5.8 *How hard is it to determine whether a planar graph is uniquely 3-colourable?*

If a parsimonious reduction from SAT to Planar 3-COL exists, then it is unlikely that polynomial-time algorithm exists for the above problem, since the existence of such an algorithm would again imply that **RP** = **NP**.

We know, from the 4-Colour Theorem, that any planar graph must be 4-colourable. However, it is **#P**-complete to determine the exact number of 4-colourings of planar graphs [Ver91]. The following problem is also interesting:

Open Problem 1.5.9 *How hard is it to determine whether a planar graph is uniquely 4-colourable?*

Elementary considerations show that, in order for a planar graph to be uniquely 4-colourable, it must contain at least $3|V| - 6$ edges, and so it must be a triangulation. The dual problem is to determine whether a planar cubic graph has a unique edge 3-colouring (to see this, use the group $\mathbb{Z}_2 \times \mathbb{Z}_2$ for the nowhere-zero 4-flow problem on the planar dual).

Fiorini [Fio75] conjectures that every uniquely 3-edge colourable planar graph contains a triangle, other than $K_{1,3}$. If this conjecture is true, then deciding unique 4-colourability of planar graphs is easy, since such a graph must contain a trivalent vertex, and this vertex (and its incident edges) can then be deleted without altering the property of being uniquely 4-colourable. This operation can be repeated until we are left with a K_4. For more on the subject of unique edge-colourings, see [Tho78].

Chapter 2

The Tutte polynomial of families of graphs

We have seen that, unless $\#\mathbf{P} = \mathbf{FP}$, there is no efficient way of calculating the Tutte polynomial of an arbitrary graph. However, for some families of graphs, there are simple methods of determining the Tutte polynomial of a member of that family. We derive simple recursive expressions for the Tutte polynomials of the complete graphs, and the complete bipartite graphs, from which formulae for some Tutte invariants can be derived. In the final section, we will also answer a question concerning the location of the zeros of the flow polynomial.

2.1 Recursive families of graphs

Although we cannot hope to find a way of calculating the Tutte polynomial of an arbitrary graph in polynomial time, there is no reason why we might not be able to do so for some greatly restricted class of graphs. As a trivial example, it is immediate from (1.3.5) that the Tutte polynomial of a tree with n vertices is equal to x^{n-1}.

Attempts to find methods for calculating the Tutte polynomials of particular families of graphs is motivated in part by problems occurring on the square lattice in statistical physics. For instance, we can consider the following question.

Open Problem 2.1.1 *How does the partition function of the q-state Potts model on the $n \times n$ square lattice L_n behave as n grows?*

Just about the only non-trivial exact result concerning the square lattice is the following theorem of Lieb:

Theorem 2.1.2 (Lieb [Lie67]) *The asymptotic limit of the number of 3-colourings of the square lattice satisfies*

$$\lim_{n\to\infty} P(L_n; 3)^{1/n^2} = \left(\frac{4}{3}\right)^{3/2}.$$

Other than this, little is known about the asymptotic behaviour of the Tutte polynomial of the square lattice, but there are many conjectures and open problems in this area.

These questions seem hard to answer, and yet it is highly unlikely that they are even **NP**-hard, still less **#P**-complete, as the following argument shows.

We say a set of strings S (of symbols from a finite alphabet Σ) is *sparse* if, for all n, the number of elements in S having length n is bounded by a polynomial in n. We make use of the following result:

Theorem 2.1.3 (Mahaney [Mah82]) *If there is a sparse* **NP***-hard set, then* $\mathbf{P} = \mathbf{NP}$.

Lemma 2.1.4 *If there exists an* **NP***-hard function f mapping integers to integers (encoded as binary strings), with a sparse domain set and such that, for some polynomial p, $|f(x)| \leq p(|x|)$ for all x, then* $\mathbf{P} = \mathbf{NP}$.

Proof: Given any such function f, we can define a language that contains precisely the set of strings given by $\{(x,k) :$ the kth bit of $f(x)$ is a $1\}$. This language is clearly as hard as f, and will also be sparse if both the domain of f is sparse and f only produces polynomially many bits. □

Corollary 2.1.5 *If it is* **NP***-hard to calculate the Tutte polynomial of the square lattice* L_n*, then* $\mathbf{P} = \mathbf{NP}$.

The name **#P1** was proposed by Valiant [Val79] for the set of functions that count the number of witnesses for a Turing machine that has a unary input language. He also gave an example of a language that is **#P1**-complete. It is not known if any of the "natural" problems on the square lattice are **#P1**-complete, but they clearly belong to **#P1**.

Biggs, Damerell and Sands [BDS72] define a family of graphs $\{G_n\}$ to be a *recursive family* if there is a homogeneous recurrence relation

$$T_{n+d} + p_1 T_{n+d-1} + \cdots + p_d T_n = 0 \qquad (2.1.6)$$

where T_i is the Tutte polynomial of G_i, and p_j is some polynomial in x and y with integer coefficients, independent of n.

It is not hard to show that, in these cases, the Tutte polynomials of the members of the family can be calculated in polynomial time. They derive homogeneous recurrence relations for the Tutte polynomials of several recursive families of graphs, including the wheels, ladders and Möbius ladders, and using these, are able to find closed-form fomulae for the chromatic polynomials of members of these families. This allows them to check a conjecture about the location of the zeros of the chromatic polynomials of graphs of bounded degree.

However, as they make clear in their paper, this method can only work for a very limited choice of families. Let $e_n = |E(G_n)|$ and $v_n = |V(G_n)|$. We

know that the degree of $T(G_n)$ in x is $v_n - 1$, and maximum degree in y is $e_n - v_n + 1$. As the recurrence relation (2.1.6) is linear, both these degrees must be linear in n. This implies that the average vertex degree, $2e_n/v_v$, cannot tend to infinity with n unless v_n is constant.

So, as they point out, this method cannot work for families with increasing average degree and an increasing number of vertices, such as the complete graphs K_n and the complete bipartite graphs $K_{m,n}$.

Our aim over the next four sections is to extend their method so that we can find polynomial-time computable expressions for the Tutte polynomial of some families of graphs with a high degree of symmetry. In particular, we are able to derive recursive formulae for the Tutte polynomial of the complete graphs K_n, and the complete bipartite graphs $K_{m,n}$.

2.2 Calculating the Tutte polynomial of K_n

A generating function for the Tutte polynomial of the complete graphs was given by Tutte in 1967.

Theorem 2.2.1 (Tutte [Tut67]) *The Tutte polynomial of the complete graph has as generating function*

$$\sum_{n=0}^{\infty} \frac{(y-1)^n u^n}{n!} T(K_n; x, y) = \left\{ \sum_{q=0}^{\infty} \frac{u^q}{q!} y^{\frac{1}{2}q(q-1)} \right\}^{(x-1)(y-1)}.$$

It can be shown (although Tutte did not explicitly point this out) that this formula can be used to give a polynomial-time algorithm for evaluating the Tutte polynomial of the complete graphs at any particular point (x, y), as long as $y \neq 1$. So, if it is desired, the whole polynomial can be calculated by Lagrangian interpolation, also in polynomial time (details of this are contained in [JVW90]). However, performing this direct computation is tedious

and unlikely to give much insight into the behaviour of the Tutte polynomial of K_n as n increases.

We give a direct method of computing the Tutte polynomial of the complete graphs recursively, which can be used to give simple formulae for evaluations of the Tutte polynomial at a few points, as follows.

Consider the following family of graphs $\{G_{r,s}\}$, as illustrated in Figure 2.A. Each graph $G_{r,s}$ has $s+1$ vertices labelled $v_1, \ldots, v_{s+1}$. The subgraph induced by $v_1, \ldots, v_s$ is a clique, and there are $r > 0$ edges in parallel connecting v_{s+1} to each other vertex, which we call a *bundle*, making a total of $s(s+2r-1)/2$ edges.

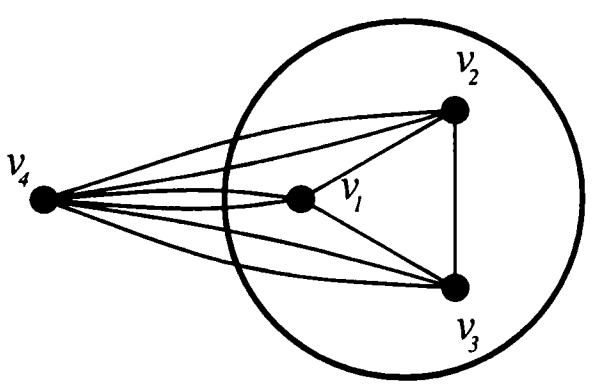


Figure 2.A: $G_{2,3}$

We now apply the deletion/contraction formula (1.3.4) to all of the edges adjoining v_{s+1}. We have to consider every possible combination of deletions and contractions of the rs edges in the bundles, making sure we do not contract a loop or delete an isthmus. From each bundle, we can delete all the edges, or delete some and then contract one, which causes all the remaining edges in that bundle to become loops. We must contract at least one of all of the edges, for if we were to delete all of them, we would disconnect the graph and so must have deleted an isthmus, which is not allowed. If we contract a total of i edges, for $i \geq 2$, we can choose which bundles to pick them from in $\binom{s}{i}$ ways, and from these bundles, we can firstly delete any number from 0 up to $r-1$ edges, and then contract a single edge, creating any number

between 0 and $r-1$ loops from the remaining edges. When this process is complete, we are left with a graph on $s-i+1$ vertices.

If we delete all the loops from this graph, we see that the remaining graph is isomorphic to $G_{i,s-i}$. There will be $\binom{i}{2}$ loops corresponding to edges that connected the vertices within the original s-clique (as all the vertices spanned by the set of contracted edges are now identified), and also a number of loops formed by the edges that were neither deleted or contracted in the bundles.

We need to take a bit more care over the case $i = 1$, for if we choose to delete every edge up to the last edge in the last bundle, then by the time we get to it, it will have become an isthmus. This would mean that we have to contract it, and multiply the Tutte polynomial of the resulting graph (which is isomorphic to $G_{1,s-1}$) by x. When we sum over all possibilities, and use the rule for removing loops given in Equation 1.3.5, we reach the following equations.

Proposition 2.2.2 *The Tutte polynomial of $G_{r,s}$ is given by*

$$\begin{aligned} T(G_{r,s}) &= \sum_{i=1}^{s} \binom{s}{i} y^{\binom{i}{2}} (1+y+\cdots+y^{r-1})^i T(G_{i,s-i}) \\ &\quad +(x-1)T(G_{1,s-1}) \end{aligned}$$

for $s > 0$, and

$$T(G_{r,0}) = 1.$$

Noting that K_n and $G_{1,n-1}$ are isomorphic, we can now calculate the Tutte polynomial of K_n recursively using these equations.

Using the formulae above to calculate the Tutte polynomial of K_n, we must firstly calculate $T(G_{j,k})$ for all $G_{j,k}$ such that $j+k \leq n-1$. There are $O(n^2)$ of these graphs, and the Tutte polynomial of each of these graphs will have $O(n^3)$ terms (1.3.2), and the size of each coefficient is certainly no more than n^2 bits, so the Tutte polynomial of K_n can be found in polynomial time.

If we consider the deletions and contractions in a different way, we can arrive at a second, slightly different, recursive formula. Choosing just one edge from each bundle, each can be either deleted or contracted independently of the others, as long as r is greater than 1. If we choose to contract i of them and delete the other $s-i$, we can pick these i in $\binom{s}{i}$ ways, and we are left with a graph isomorphic to $G_{r+i-1,s-i}$ with some loops. Similar reasoning to that above shows that there are in fact $\binom{i}{2}+i(r-1)$ extra loops. If $r=1$, then we cannot delete the edge from every bundle, as this would entail the deletion of an isthmus, and similar reasoning to that used earlier gives as our equations:

$$T(G_{r,s}) = \begin{cases} \sum_{i=0}^{s} \binom{s}{i} y^{\binom{i}{2}} y^{i(r-1)} T(G_{r+i-1,s-i}) & r>1 \text{ and } s>0 \\ \sum_{i=1}^{s} \binom{s}{i} y^{\binom{i}{2}} T(G_{i,s-i}) + (x-1)T(G_{1,s-1}) & r=1 \text{ and } s>0 \\ 1 & s=0 \end{cases} \tag{2.2.3}$$

and now these formulae can be used as above to calculate the Tutte polynomial of K_n.

2.3 Tutte invariants of the complete graphs

We can attempt to solve these recurrence relations at particular points of interest, or along particular lines.

For instance, the relationship between the Tutte and chromatic polynomials of a graph is given by Proposition 1.3.11. If we use our recursive formulae, then we easily reach the well-known equation for the chromatic polynomial of the complete graph.

Settting $y = 0$ in the formulae given in Proposition 2.2.2 gives us

$$T(G_{r,s}; x, 0) = \begin{cases} (x+s-1)T(G_{1,s-1}; x, 0) & \text{if } s > 0 \\ 1 & \text{if } s = 0 \end{cases}$$

which solves to give

$$T(G_{r,s}; x, 0) = x(x+1)\cdots(x+s-1)$$

and then (1.3.11) gives the familiar result

$$P(K_n; \lambda) = \lambda(\lambda-1)\ldots(\lambda-n+1).$$

We can also find a simple formula for the number of spanning trees of $G_{r,s}$. If we substitute $x = y = 1$ in Proposition 2.2.2, our recurrence relation simplifies to

$$\kappa(G_{r,s}) = \begin{cases} \sum_{i=1}^{s} \binom{s}{i} r^i \kappa(G_{i,s-i}) & \text{if } s > 0 \\ 1 & \text{if } s = 0 \end{cases}$$

We now show by induction on s that

$$\kappa(G_{r,s}) = r(r+s)^{s-1}$$

It is easy to verify that this expression is correct for $s = 0$, and, if we assume it holds for all $s \leq k$ and all r then

$$\begin{aligned} \kappa(G_{r,k+1}) &= \sum_{i=1}^{k+1} \binom{k+1}{i} r^i \cdot i \cdot (k+1)^{k-i} \\ &= (k+1)^k \sum \binom{k+1}{i} \left(\frac{r}{k+1}\right)^i \cdot i \\ &= (k+1)^k \frac{r}{k+1} \sum \binom{k+1}{i} \cdot i \cdot \left(\frac{r}{k+1}\right)^{i-1} \end{aligned}$$

and, observing that

$$\begin{aligned}\sum_{j=1}^{n}\binom{n}{j}jx^{j-1} &= \frac{d}{dx}\left(\sum_{j=1}^{n}\binom{n}{j}x^{j}\right)\\ &= \frac{d}{dx}\left((x+1)^n-1\right)\\ &= n(x+1)^{n-1} \qquad (2.3.1)\end{aligned}$$

we arrive at

$$\begin{aligned}\kappa(G_{r,k+1}) &= (k+1)^k\cdot\frac{r}{k+1}\cdot(k+1)\cdot(\frac{r}{k+1}+1)^k\\ &= r(r+k+1)^k.\end{aligned}$$

By setting $r=1$ and $s=n-1$, we get Cayley's formula that the number of labelled spanning trees on n vertices is equal to $\kappa(K_n)=\kappa(S_{1,n-1})=n^{n-2}$.

We will consider the flow polynomial in Section 2.6.

2.4 Complete bipartite graphs

We now consider the Tutte polynomial of complete bipartite graphs $K_{m,n}$. The graph $K_{m,n}$ consists of $m+n$ vertices $\{u_1,\ldots,u_m,v_1,\ldots,v_n\}$ with edge set $\{(u_i,v_j):\ 1\le i\le m, 1\le j\le n\}$.

It cannot be possible to find a solution in the manner of Biggs *et al.* for general complete bipartite graphs $K_{m,n}$ as the average vertex degree is unbounded when both n and m increase. But we can still show that the Tutte polynomial of these graphs can be computed in polynomial time.

We define $G_{r,s,t}=(V,E)$ as follows:

$$V=\{v_1,\ldots,v_r\}\bigcup\{u_1,\ldots,u_s\}\bigcup\{w\}$$

$$E=\{(v_i,u_j):\ 1\le i\le r,\ 1\le j\le s\}\bigcup\{t \text{ copies of } (w,u_j):\ 1\le j\le s\}$$

so $G_{r,s,t}$ contains a $K_{r,s}$ and an extra vertex w connected to each of the u_j by t edges in parallel. Note that $G_{r,s,1}$ is isomorphic to $K_{r+1,s}$.

As an example, Figure 2.B is a picture of the graph $G_{3,4,2}$.

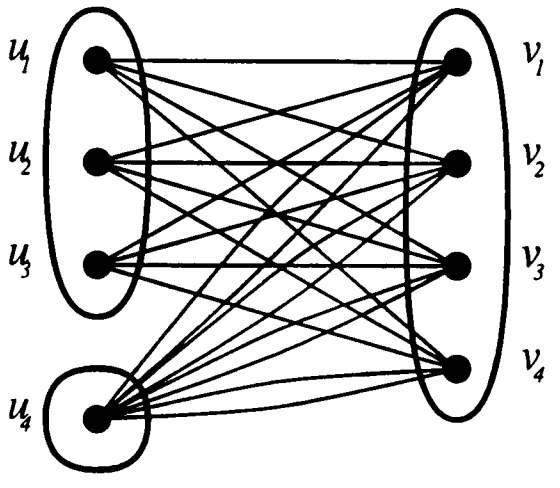


Figure 2.B: $G_{3,4,2}$

We can apply the deletion/contraction algorithm to the edges incident with w, in a similar manner to in the previous section, and deduce the following formula:

Proposition 2.4.1 *The Tutte polynomial of $G_{r,s,t}$ is given by*

$$\begin{aligned} T(G_{r,s,t}) &= \sum_{i=1}^{s} \binom{s}{i} (1 + y + \cdots + y^{t-1})^i T(G_{s-i,r,i}) \\ &+ (x-1)T(G_{s-1,r,1}) \end{aligned}$$

for $r,\ s > 0$,

$$T(G_{0,s,t}) = (x + y + \cdots + y^{t-1})^s,$$

and

$$T(G_{r,0,t}) = 1.$$

The complete bipartite graph $K_{m,n}$ is isomorphic to $G_{m-1,n,1}$, and so we can use these equations to calculate $T(K_{m,n})$. In order to do this, we have to

firstly calculate $T(G_{r,s,t})$ for all $G_{r,s,t}$ such that $r+s+t < m+n$, and there are $O((m+n)^3)$ of these graphs. Each polynomial contains at most $O((n+m)nm)$ terms, and each coefficient is at most nm bits long, so the entire calculation can be performed in polynomial time.

2.5 Some Tutte invariants of complete bipartite graphs

We can again use our recurrence relation to calculate formulae for Tutte invariants.

Firstly, the number of spanning trees of $K_{m,n}$ is fairly easy to deduce. If we set $x = 1$, $y = 1$ then the equations simplify to give

$$\begin{aligned} \kappa(G_{r,s,t}) &= \sum_{i=1}^{s} \binom{s}{i} t^i \kappa(G_{s-i,r,i}) \\ \kappa(G_{0,s,t}) &= t^s. \end{aligned} \tag{2.5.1}$$

We now show

Lemma 2.5.2 *The number of spanning trees of $G_{r,s,t}$ is equal to $t \cdot s^r (t+r)^{s-1}$ (for $s > 0$).*

Proof: We induct on $r+s$. The base case, $r+s = 1$, is given by the second equation from (2.5.1), as we can only have $r = 0$, $s = 1$.

If we assume that our lemma is true for all r, s such that $r+s \leq k$, then for any r, s such that $r+s = k+1$ we have

$$\kappa(G_{r,s,t}) = \sum_{i=1}^{s} \binom{s}{i} t^i \kappa(G_{s-i,r,i})$$

$$= \sum_{i=1}^{s} \binom{s}{i} t^i \cdot i \cdot r^{s-i} s^{r-1} \text{ (by the inductive hypothesis)}$$
$$= r^s s^{r-1} \frac{t}{r} \sum_{i=1}^{s} \binom{s}{i} \cdot i \cdot \left(\frac{t}{r}\right)^{i-1}.$$

Using (2.3.1), we get

$$\begin{aligned} \kappa(G_{r,s,t}) &= r^s s^{r-1} \frac{t}{r} s \left(\frac{t}{r} + 1\right)^{s-1} \\ &= ts^r (t+r)^{s-1}. \end{aligned}$$

□

By taking $t = 1$, we reach the formula given by Onaderra in [Ona73] for the number of spanning trees of complete bipartite graphs:

$$\kappa(K_{r,s}) = \kappa(G_{r-1,s,1}) = s^{r-1} r^{s-1}.$$

Next we consider the chromatic polynomial. For convenience, we will write $T'(G)$ for $T(G; x, 0)$.

When we set $y = 0$ in Proposition 2.4.1, the equations simplify to give

$$T'(G_{r,s,t}) = \sum_{j=1}^{s} \binom{s}{j} T'(G_{s-j,r,j}) + (x-1) T'(G_{s-1,r,1})$$

for $r > 0$, and

$$T'(G_{0,s,t}) = x^s.$$

We observe from the above that $T'(G_{r,s,t}) = T'(G_{r,s,1})$, since the variable t does not appear on the right hand side of either formula. This is unsurprising, as multiple edges cannot affect the chromatic polynomial of any graph.

We now prove:

Lemma 2.5.3 *The Tutte polynomial of $G_{r,s,1}$ evaluated along the line $y = 0$ is given by*

$$T'(G_{r,s,1}) = \sum_{k=0}^{r} (-1)^{r+k} S(r+1, k+1)(x+k)^s \langle x \rangle_k$$

where $\langle x \rangle_k$ is the kth rising factorial polynomial *given by*

$$\langle x \rangle_k = \begin{cases} 1 & k = 0 \\ (x+k-1)\langle x \rangle_{k-1} & k > 0 \end{cases}$$

and the $S(n,k)$ are the Second Stirling numbers.

The Second Stirling number $S(n,k)$ is defined as the number of different ways of partitioning n different objects into k unlabelled sets such that no set is empty. For convenience, $S(0,0)$ is given the value 1 [Com74, page 200].

Proof: We use induction on r. The base case $r = 0$ is given directly by the formula. We assume that the lemma has been shown true for all $r \leq r'$, and write

$$T'(G_{r'+1,s,1}) = \sum_{j=1}^{s} \binom{s}{j} T'(G_{s-j,r'+1,1}) + (x-1)T'(G_{s-1,r'+1,1}).$$

Noting that $G_{r,s+1,1}$ and $G_{s,r+1,1}$ are isomorphic for all r and s we can rewrite this as

$$T'(G_{r'+1,s,1}) = \sum_{j=1}^{s} \binom{s}{j} T'(G_{r',s+1-j,1}) + (x-1)T'(G_{r',s,1}).$$

By the inductive hypothesis, the right hand side of this equation is equal to

$$\sum_{j=1}^{s} \binom{s}{j} \left(\sum_{k=0}^{r'} (-1)^{r'+k} S(r'+1, k+1)(x+k)^{s+1-j} \langle x \rangle_k \right)$$

$$+ \quad (x-1) \sum_{k=0}^{r'} (-1)^{r'+k} S(r'+1, k+1)(x+k)^s \langle x \rangle_k. \qquad (2.5.4)$$

We now note the simple identity that

$$\sum_{i=1}^{s} \binom{s}{i} x^{-i} = \left(\frac{x+1}{x}\right)^{s} - 1 \tag{2.5.5}$$

which we make use of to solve the recurrence relation in the following manner.

We reverse the order of summation in (2.5.4) to get

$$\sum_{k=0}^{r'} (-1)^{r'+k} S(r'+1, k+1)(x+k)^{s+1} \langle x \rangle_k \times \left[\left(\sum_{j=1}^{s} \binom{s}{j} (x+k)^{-j} \right) + \frac{x-1}{x+k} \right]$$

and, using identity (2.5.5), this becomes

$$\begin{aligned} & \sum_{k=0}^{r'} (-1)^{r'+k} S(r'+1, k+1)(x+k)^{s+1} \\ & \quad \times \langle x \rangle_k \left[\left(\frac{x+k+1}{x+k} \right)^{s} - 1 + \frac{x-1}{x+k} \right] \\ = & \sum_{k=0}^{r'} (-1)^{r'+k} S(r'+1, k+1) \\ & \quad \times \left[(x+k+1)^{s}(x+k)\langle x \rangle_k + \left((x-1)(x+k)^{s} - (x+k)^{s+1} \right) \langle x \rangle_k \right] \\ = & \sum_{k=0}^{r'} (-1)^{r'+k} S(r'+1, k+1) \\ & \quad \times \left[(x+k+1)^{s} \langle x \rangle_{k+1} - (k+1)(x+k)^{s} \langle x \rangle_k \right]. \end{aligned}$$

This can be rearranged to give

$$\sum_{k=0}^{r'+1} (-1)^{r'+1+k} S(r'+2, k+1)(x+k)^{s} \langle x \rangle_k$$

where

$$S(r,k) = S(r-1, k-1) + kS(r-1, k) \quad \text{for } 1 \leq r, k$$

and

$$S(r,0) = S(0,k) = 0, \text{ except } S(0,0) = 1,$$

which is exactly the recurrence relation for the second Stirling numbers given as Theorem A in Comtet [Com74, page 208] □

We can now write down the chromatic polynomial using the equation given earlier linking the Tutte and chromatic polynomials of graphs.

Proposition 2.5.6 *The chromatic polynomial of the complete bipartite graph, $K_{m,n}$, is given by*

$$P(K_{m,n};\lambda) = \sum_{k=1}^{m} S(m,k)(\lambda - k)^n(\lambda)_k$$

where $(\lambda)_k$ is the kth falling factorial polynomial, *given by $(\lambda)_0 = 1$, and $(\lambda)_k$ is equal to $(\lambda - k + 1)(\lambda)_{k-1}$ for $k > 0$.*

Proof: This is obtained by the substitution

$$\begin{aligned}
P(K_{m,n};\lambda) &= (-1)^{m+n-1}\lambda T(G_{m-1,n,1}; 1-\lambda, 0) \\
&= (-1)^n\lambda \sum_{j=0}^{m-1}(-1)^j S(m,j+1)(j+1-\lambda)^n(-1)^j(\lambda)_{j+1}\lambda^{-1} \\
&= \sum_{j=0}^{m-1} S(m,j+1)(\lambda-(j+1))^n(\lambda)_{j+1} \\
&= \sum_{k=1}^{m} S(m,k)(\lambda-k)^n(\lambda)_k.
\end{aligned}$$

□

As an interesting corollary of this, we consider the Tutte polynomial of the star graph $K_{n,1}$. We can think of this graph in two ways, either as $K_{n,1}$ or as $K_{1,n}$, and, equating the two formulae for the chromatic polynomials given by Proposition 2.5.6, we see that

$$\lambda(\lambda-1)^n = \sum_{k=1}^{n} S(n,k)(\lambda-k)(\lambda)_k.$$

This, when we divide both sides by λ and substitute $z = \lambda - 1$, gives us, for any $n > 0$,

Corollary 2.5.7

$$z^n = \sum_{k=1}^{n} S(n,k)(z)_k.$$

This is equivalent to the generating function given in [Com74, page 207, Theorem B], with the irrelevant term $S(n,0)$ omitted.

2.6 Zeros of graph polynomials

Much work has been done on locating the zeros of the chromatic and reliability polynomials. There is a long-standing conjecture about the size of the zeros of the chromatic polynomial of graphs of bounded degree, based on a generalisation of a theorem of Brooks [Bro41]:

Open Problem 2.6.1 *Does there exist a function $B : \mathbb{N} \to \mathbb{N}$ such that for any regular graph G, all zeros $z \in \mathbb{C}$ of the chromatic polynomial of G satisfy*

$$|z| \leq B(\delta_G),$$

where δ_G is the degree of a vertex of G.

This was posed in [BDS72]. As a special case of this, they suggested that $B(3) = 3$ may suffice, since this bound holds for all cubic graphs of at most 10 vertices, as well as two infinite families of cubic graphs that they consider in their paper. This bound has since been shown to hold for all cubic graphs of up to 16 vertices and some larger graphs [RR91], but no general proof has been found. Many interesting results have however been obtained, see for example [BRW, Woo92].

The reliability of a graph has also been studied extensively, and the following conjecture concerning the zeros of the reliability polynomial of any graph appears in [BC92]:

Conjecture 2.6.2 *The zeros of the reliability polynomial of a connected graph lie in the region* $|z-1| \leq 1$ *in the complex plane.*

They are able to prove that all the real zeros of the reliability polynomial lie in $\{0\} \bigcup (1,2]$, and provide some evidence for their conjecture. They show that their conjecture holds for $K_{2,m}$ for all m, but are not able to prove that their conjecture holds even for the complete graphs.

As the flow polynomial is the dual of the chromatic polynomial, it might be expected that its zeros would also lie in some bounded region of the plane. However, we will show that this cannot be the case, by demonstrating that for a particular family of cubic graphs (the Möbius ladders), the flow polynomial has a zero of arbitrarily large modulus.

We can attempt to find the zeros of the flow polynomial of the complete and complete bipartite graphs, using our recurrence relation. Unfortunately, for general m and n, we cannot find a closed-form solution to the recurrence relation, so we have to be content with restricted cases.

If we fix m to be a constant, then we can obtain the Tutte polynomial of $K_{m,n}$ in closed form, by solving the recurrence relation (2.4.1) directly. For example, the following is straightforward to check.

Proposition 2.6.3 *The Tutte polynomial of* $K_{3,n}$ *is given by*

$$\begin{aligned} T(K_{3,n};x,y) &= (x+2)^n(x-1)^2 \\ &\quad +(y-1)^{-2}\left[(x+y+y^2)^n-(x+2)^n\right] \\ &\quad +(y-1)^{-2}\left[(3xy-3y-3x)\left((x+y+1)^n-(x+2)^n\right)\right] \end{aligned}$$

for $y \neq 1$.

It is straightforward to obtain the reliability polynomial from this, and check that (2.6.2) holds for this family.

We can also use this equation to obtain the flow polynomial, via (1.3.12):

Corollary 2.6.4 *The flow polynomial of $K_{3,n}$ is given by*

$$F(K_{3,n};\lambda) = (\lambda-1)(\lambda-2)\cdot\frac{1}{\lambda^2}\left[(\lambda-2)^{n-1}\left\{(\lambda-1)^{n-1}-3\cdot(-1)^{n-1}\right\}+2^n\right].$$

It can be shown that the only real zeros of this function are at $\lambda = 1$ and $\lambda = 2$ [Sek93], and, moreover, they are both of multiplicity 1. The remaining complex zeros all lie in a bounded region of the complex plane, as the following makes clear.

Proposition 2.6.5 *For any fixed $\epsilon > 0$, all but a finite number of the zeros of all the flow polynomials $\{F(K_{3,n})\}$ lie between the two Cassinian Ovals given by*

$$2-\epsilon = |(z-1)(z-2)|$$

and

$$2+\epsilon = |(z-1)(z-2)|.$$

Proof: Any complex zero z must satisfy

$$[(z-1)(z-2)]^{n-1} + 2^n + (-1)^n\cdot 3(z-2)^{n-1} = 0.$$

If $|(z-1)(z-2)| > 2+\epsilon$, then we also must have $|z-1| > 1+\epsilon/4$ and so $|(z-1)(z-2)| > (1+\epsilon/4)\,|z-2|$. Hence, there exists a n_ϵ such that

$$|(z-1)(z-2)|^{n-1} > 2^n + 3\,|z-2|^{n-1}$$

for all $n \geq n_\epsilon$.

Similarly, if $|(z-1)(z-2)| < 2-\epsilon$, then $|z-2| < 2-\epsilon/4$ and so there exists a n'_ϵ such that

$$2^n > |(z-1)(z-2)|^{n-1} + 3\,|z-2|^{n-1}$$

for all $n > n'_\epsilon$.

So, for any $\epsilon > 0$, there are no complex zeros of any of the flow polynomials of $K_{3,n}$ lying outside the specified annulus for all $n > \max\{n_\epsilon, n'_\epsilon\}$. □

A similar result can be shown for the zeros of the flow polynomial of $K_{4,n}$: namely, that there are only a finite number of zeros outside the region given by

$$\left\{z : \left|z^2 - 3z + 3\right| \leq 3 + \epsilon\right\}$$

for any positive $\epsilon > 0$. However, we have not managed to obtain any general results about the location of the zeros of the flow polynomial of the general complete bipartite graph $K_{m,n}$.

A natural question to ask is:

Open Problem 2.6.6 *How does the largest zero of the flow polynomial of the complete bipartite graph $K_{m,n}$ grow with m and n?*

We can also use a program such as Mathematica to work out the flow polynomial of the complete graph K_n for small values of n and see how the zeros behave. A diagram of the location of the complex zeros of the flow polynomial of the complete graphs $K_5, \ldots, K_9$ is given in [Sek93]: it appears that they lie close to the circle $|z - 1| = 3/2$ in the complex plane, but we have not been able to prove this for general n, and so this leads us to conjecture the following:

Conjecture 2.6.7 *The zeros of the flow polynomials of the complete graphs K_n lie in a bounded region of the complex plane.*

2.7 Zeros of the flow polynomial

Tutte's 5-flow conjecture is still open:

Conjecture 2.7.1 (Tutte [Tut54]) *Every bridgeless graph has a nowhere-zero 5 flow.*

Seymour has proved that every bridgeless graph has a nowhere-zero 6 flow. It might be hoped that there would be some bound on the size of the complex zeros of the flow polynomial, analogous to either of the conjectures given earlier concerning the zeros of the chromatic and reliability polynomials. However, we will now show that no such conjecture could be true, by demonstrating that, for a particular family of cubic graphs, there is no bound on the size of the zeros of their flow polynomials.

Proposition 2.7.2 *For every constant c, there is a k such that the flow polynomial of M_k, the Möbius Ladder on $2k$ vertices, has a zero z satisfying $|z| > c$.*

Proof: The Möbius Ladder, M_k, consists of a cycle with $2k$ vertices, and extra edges between opposite vertices in the cycle. A recurrence relationship for the Tutte polynomial of the Möbius Ladder can be obtained, by considering possible sequences of deletions and contractions. The recurrence relation is given in [BDS72]. We substitute $x = 0$, $y = 1 - \lambda$ and solve to get

$$F(M_k; \lambda) = (\lambda - 1)(\lambda - 3)^k + (\lambda - 2)^k + (-1)^{k-1}.$$

For simplicity, we will insist on k being odd, so this equation simplifies to

$$F(M_k; \lambda) = (\lambda - 1)(\lambda - 3)^k + (\lambda - 2)^k + 1.$$

We will write this as

$$F(M_k; \lambda) = A_k(\lambda) + B_k(\lambda),$$

where

$$A_k(\lambda) = (\lambda - 1)(\lambda - 3)^k$$

and

$$B_k(\lambda) = (\lambda - 2)^k + 1.$$

Any zero z of this equation must satisfy the equations

$$|A_k(z)| = |B_k(z)| \tag{2.7.3}$$

$$\arg(A_k(z)) \equiv \arg(B_k(z)) - \pi \pmod{2\pi} \tag{2.7.4}$$

which we call the modulus equation and argument equation respectively.

We write $z = 3 + de^{i\theta}$, and consider the region of the complex plane in which $0 \leq \theta \leq \pi/2$ and $|d| > 1$.

It is immediate that, for $0 \leq \theta \leq \pi/2$,

$$\sqrt{d^2+1} \leq |z-2| \leq d+1$$

and

$$d \leq |z-1| \leq d+2.$$

Substituting these into the formula for the flow polynomial, we get

$$d^{k+1} \leq |A_k(z)| \leq d^k(d+2)$$
$$(d+1)^k + 1 \geq |B_k(z)| \geq (\sqrt{d^2+1})^k - 1.$$

So, if $4d^2 \ln(d+3) < k$,

$$\left(1 + \frac{1}{d^2}\right)^{k/2} > e^{\ln(d+3)} > d + 2 + d^{-k}$$

(since $(1 + \frac{1}{x})^{2x} > e$ for all $x > 1$) which implies that

$$\left(\frac{d^2+1}{d^2}\right)^{k/2} - d^{-k} > d+2$$

or

$$\left(\sqrt{d^2+1}\right)^k - 1 > d^k(d+2),$$

which shows that

$$|B_k(z)| > |A_k(z)|\,.$$

Conversely, if we take $d\ln(d-1) > k$, then we see that

$$\left(1+\frac{1}{d}\right)^k < e^{\ln(d-1)} = d-1$$

(since $(1+\frac{1}{x})^x < e$ for all $x > 1$), which implies that

$$\left(1+\frac{1}{d}\right)^k < d - d^{-k}$$

or

$$\left(\frac{d+1}{d}\right)^k + d^{-k} < d,$$

and this shows us that

$$|B_k(z)| < |A_k(z)|\,.$$

It is clear that both $|A_k|$ and $|B_k|$ are continuous real-valued functions in the quadrant $0 \le \theta \le \pi/2$. Fix the value of θ and consider the function $|A_k| - |B_k|$. We see that

$$\begin{aligned} |A_k| - |B_k| &= (4 + 4\cos\theta d + d^2)^{1/2} \cdot d^k - (1 + 2\cos\theta d + d^2)^{k/2} - 1 \\ &= d^k \left((4 + 4\cos\theta d + d^2)^{1/2} - \left(\frac{1}{d^2} + \frac{2\cos\theta}{d} + 1 \right)^{k/2} - \frac{1}{d^k} \right). \end{aligned}$$

This function clearly increases with d, and this, combined with the previous argument, shows that there is exactly one value of d for which $|A_k| = |B_k|$, for each value of θ in the range $0 \le \theta \le \pi/2$. The continuity of A_k and B_k ensures that the set of points given by $\{z : |A_k(z)| = |B_k(z)|\,,\ 0 \le \theta \le \pi/2\}$ form a continuous curve in the region $2d^2\ln(d+3) > k > d\ln(d-1)$ along which $|A_k| = |B_k|$. We call this curve U.

We now show that, at some point p on U,

$$\arg(A_k(p)) \equiv -\arg(B_k(p)) \pmod{2\pi}.$$

At $\theta = 0$, we see immediately that $\arg A_k = \arg B_k = 0$. At $\theta = \pi/2$, we can write $z = 2 + d'e^{i(\pi/2-\phi)} = 1 + d''e^{i(\pi/2-\psi)}$ for some angles $0 < \phi,\ \psi < \pi/2$ and see that

$$\arg A_k = k\pi/2 + (\pi/2 - \psi) = (k+1)\pi/2 - \psi$$

and

$$\arg B_k = k(\pi/2 - \phi) + \epsilon = k\pi/2 - k\phi + \epsilon$$

where ϵ is a small correction term (the difference between the arguments of $(z-2)^k$ and $(z-2)^k + 1$).

The difference between these two arguments is therefore

$$\pi/2 + k\phi - \psi - \epsilon.$$

It is immediate that ψ and ϵ are both bounded by constants, as they are angles between 0 and 2π. if we can show that the product $k\phi$ increases without bound when k increases (of course, the value of ϕ depends on k) then the final result will follow.

We have a lower bound for ϕ (in terms of k) since we know that, along the curve U, we must have $d\ln(d-1) \leq k$. As $\phi = \arctan(d^{-1}) > (2d)^{-1}$ (for d sufficiently large), we see that

$$k\phi \geq \frac{1}{2}\ln(d-1)$$

if d is sufficiently large.

Of course, $\arg(A_k)$ and $\arg(B_k)$ are both continuous along the curve U, and so at some point p on U we must have

$$\arg(B_k(p)) \equiv \arg(A_k(p)) + \pi \pmod{2\pi}.$$

This point is therefore a zero of the flow polynomial of M_k.

It remains only to point out that the modulus of this zero we have found for $F(M_k; \lambda)$ must increase with k, which is clear since we have shown that $d^2 \ln(d+3) \geq k$ for this zero. $\square$

Chapter 3

The complexity of the coefficients, and the Tutte plane modulo p

We now consider in more detail some aspects of the complexity of the Tutte polynomial. We attack two problems in this chapter. Firstly, the complexity of individual coefficients of the Tutte polynomial is considered. The coefficients at the high degree end of the polynomial can be calculated in polynomial time, but determining the low degree coefficients is #**P**-complete. However, checking to see if a particular coefficient is less than a fixed constant can be performed in polynomial time.

These results, contained in the first five sections, have been accepted for publication in [Annb].

Secondly, the complexity of evaluating the Tutte polynomial at particular points modulo a prime p is considered. We conjecture that a similar result to that shown in [JVW90] holds: namely, for any fixed $p \geq 3$, evaluation of the Tutte polynomial of a graph at any point in the integer plane is hard,

except at a few known "easy" points. However, we have not been able to prove this.

3.1 The coefficients of the Tutte polynomial

We can write the Tutte polynomial of a graph G in the following form:

$$T(G;x,y) = \sum_{i,j} b_{i,j} x^i y^j. \tag{3.1.1}$$

Some facts are known about the values of the coefficients $b_{i,j}$. In particular, for a connected graph G with n vertices and m edges:

1. $b_{i,j}(G) \geq 0$ for all i, j.
2. $b_{0,0}(G) = 0$, if G contains at least one edge.
3. $b_{1,0}(G) = b_{0,1}(G)$, if G contains at least 2 edges.
4. $b_{i,j}(G) = 0$ for all $i > n-1$ or $j > m-n+1$.
5. $b_{n-1,0}(G) = 1$ and $b_{n-2,0}(G) = m$ if G is loopless.
6. $b_{0,m-n+1}(G) = 1$, if G is isthmus-free.

As well as being **#P**-hard to compute, the Tutte polynomial is of course **#P**-easy, as the following lemma shows.

Lemma 3.1.2 *Calculating the Tutte polynomial of a graph is* **#P***-easy.*

Proof: We will show that the function $f(G,i,j) = b_{i,j}(G)$ is contained in **#P**, by describing a nondeterministic Turing machine that, on input (G,i,j), has exactly $b_{i,j}(G)$ witnesses that cause it to accept the input, and runs in polynomial-time.

We order the edge-set arbitrarily. For any spanning tree T of a graph, we define the *internal activity* to be the number of edges $e \in T$ such that e is the least element in the cutset of the graph defined by the partition induced by $T \backslash e$. The *external activity* is given by the number of edges $e \in E \backslash T$ such that e is the least element in the unique circuit contained in $T \cup e$. It was shown by Tutte [Tut54] that the value of $b_{i,j}(G)$ is equal to the number of spanning trees of G with internal activity i and external activity j, and that therefore the ordering of the edges in the graph is immaterial. The internal and external activities of a particular spanning tree can clearly be recognised in polynomial-time.

It is now clear that the language $\{(G, i, j) : b_{i,j}(G) > 0\}$ is contained in **NP**. Our Turing machine accepts these inputs by checking that a spanning tree (supplied as the "witness") has the required internal and external activities (for an assumed canonical ordering of the edge set of G). If we require also that the spanning tree is given in lexicographical order, then there are precisely $b_{i,j}(G)$ witnesses, and so the function that counts the number of witnesses for this language is in **#P**.

Hence the whole polynomial is **#P**-easy to calculate. □

3.2 Easy coefficients

As we have seen above, some of the coefficients are easy to calculate. The following theorem extends this.

Theorem 3.2.1 *For any fixed integer constant k, on input G;*

i. $b_{n-1-k,j}(G)$ *can be calculated in polynomial-time, for any* j.

ii. $b_{i,m-n+1-k}$ *can be calculated in polynomial-time, for any* i. *v*

Proof: By repeatedly applying the deletion/contraction formulae (1.3.4) and (1.3.5) to a graph G with (arbitrarily) ordered edge-set E we can construct a rooted binary tree of depth m, where the nodes of the tree correspond to graphs. The root corresponds to G, and the sons of a node correspond to the graphs obtained by contracting and deleting the lexicographically first edge remaining in the edge-set of the graph corresponding to that node, if that edge is not an isthmus or loop. If the edge is an isthmus (loop), then the node only has one son, corresponding to the contraction (deletion) of that edge. The leaves of the tree correspond to single-vertex edgeless graphs.

Every leaf contributes to one of the coefficients of the Tutte polynomial of the graph. If, on the path from the root to the leaf, i isthmuses and j loops are removed, then the leaf contributes the term $x^i y^j$. Summing the contributions from all the leaves gives the Tutte polynomial.

A leaf will make a contribution to one of the coefficients $b_{n-1-k,j}$ if exactly $n-1-k$ isthmuses are encountered on the path from the root. Now, every time an edge is contracted, the number of vertices of the graph is reduced by one. Exactly $n-1$ edges (some of which are isthmuses) are contracted, as we end up with a single vertex. Therefore, for the leaf to contribute to $b_{n-1-k,j}$ for any j, then, at the nodes with two sons, the path to the leaf must pass from a node to the son corresponding to the contraction of an edge exactly k times, and to the other son each other time. The value of j in the index of the coefficient is equal to the number of loops removed on the path from the root to the leaf.

Of course, the tree is too big to construct in polynomial-time, but counting the coefficients of the required terms can still be easily carried out in polynomial-time by performing a depth-first search of the relevant nodes. We do this by testing each subset of k edges (which we take to be the edges that we will contract at the nodes with two sons) to see if this gives us a possible path in the tree, and if so, how many loops are encountered on that path. The number of such subsets is bounded by $\binom{m}{k} \leq m^k$ and so the

algorithm clearly runs in polynomial-time.

We can perform an analogous computation for the coefficients $b_{i,m-n+1-l}$, as in this case we must choose to delete exactly l edges at the nodes at which we have a choice, and must then sum the number of isthmuses removed along the path to a leaf to work out the value of i. So all these coefficients can also be calculated in polynomial-time. □

3.3 Hard coefficients

We now show that the coefficients of low degree are hard:

Theorem 3.3.1 *$b_{1,0}(G)$ is a #P-complete function.*

Proof: We have already shown that $b_{1,0}(G)$ is contained within #P. It remains to prove that it is #P-hard.

To prove hardness, given a graph G, we construct a family of graphs $\{G^{(k)}\}$ from it. Using an oracle for the coefficients $b_{1,0}(G^{(k)})$, we calculate the number of 3-colourings of G, which we have shown to be #P-hard in Chapter 1 (Proposition 1.5.1). First, we prove a lemma concerning the density of primes.

Lemma 3.3.2 *For $n > 80$, the product of the primes less than n^2 is greater than 3^n.*

Proof: We define $\vartheta(n) = \ln \prod_{p \leq n} p$, where p runs over all primes less than n, and $\psi(n) = \sum_{i=1}^{\ln n / \ln 2} \vartheta(n^{1/i})$. From Hardy and Wright [HW60, Chapter 22], we find that $\vartheta(n) < 2n \ln 2$ and $\psi(n) \geq \frac{n}{4} \ln 2$ for $n \geq 2$. So we have:

$$\begin{aligned}
\vartheta(n^2) &= \psi(n^2) - \sum_{i=2}^{2\ln n/\ln 2} \vartheta(n^{2/i}) \\
&\geq \frac{1}{4}n^2 \ln 2 - \frac{2\ln n}{\ln 2} \cdot 2n \ln 2 \\
&\geq n(n/4 - 4\ln n) \\
&> n \ln 3
\end{aligned}$$

for $n > 80$. □

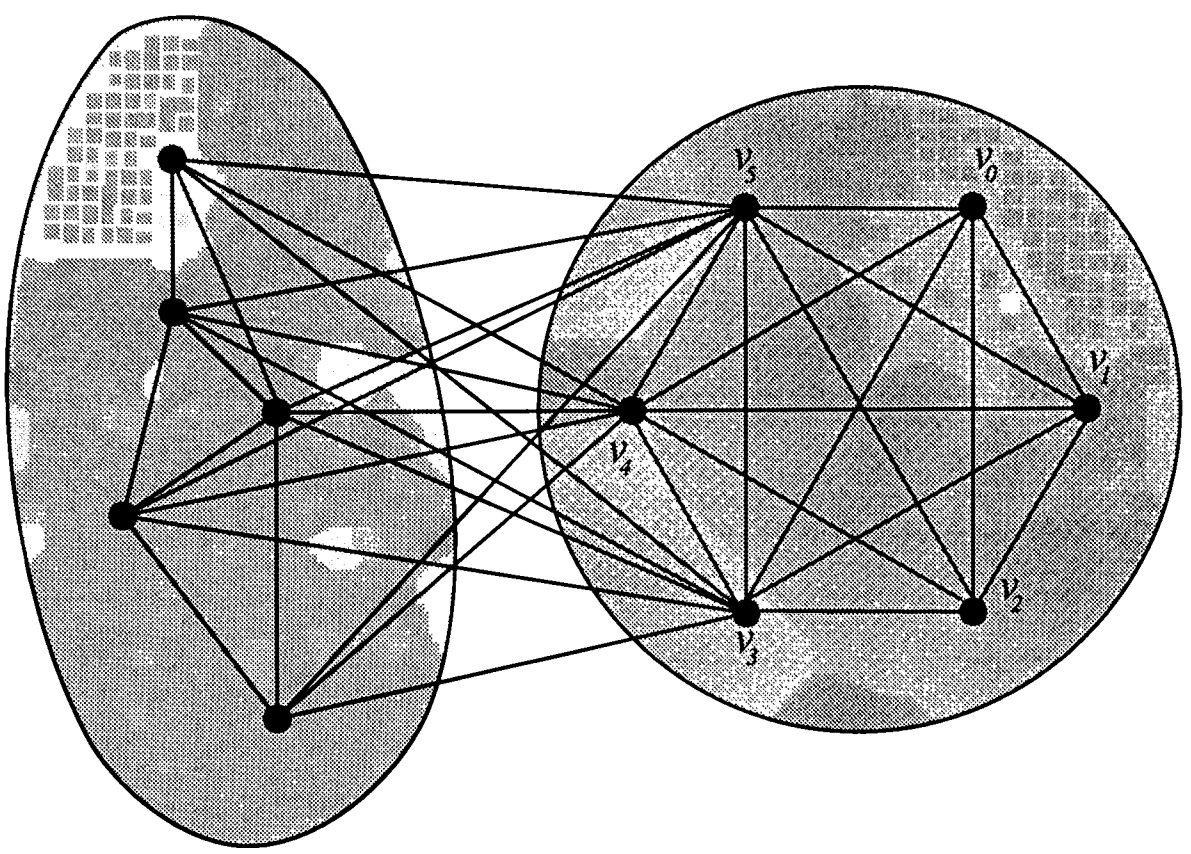
For the kth prime p_k, $3 \leq p_k < n^2$, we construct $G^{(k)}$ as follows. Form a clique on the vertices $\{v_1, \ldots, v_{p_k+1}\}$, and add edges (u_i, v_j) for all $u_i \in V(G)$ and $4 \leq j \leq p_k + 1$. As an example, Figure 3.A shows the construction of $G^{(3)}$ from G.

Consider a $(p_k + 1)$-colouring of $G^{(k)}$. Each vertex in the clique that we have added to G must be coloured differently, and we can do this in exactly $(p_k + 1)!$ ways. Moreover, for every colouring of the clique, the vertices of G can only be coloured with the three colours assigned to the vertices v_1, v_2 and v_3. Furthermore, for any fixed colouring of the clique, there is a 1-1 correspondence between 3-colourings of G and extensions of the $(p_k + 1)$-colouring of the clique to the whole graph $G^{(k)}$. So:

$$P(G^{(k)}; p_k + 1) = (p_k + 1)! \cdot P(G; 3),$$

and, using Equation 1.3.11 relating the Tutte and chromatic polynomials, we see that:

$$\begin{aligned}
(-1)^{n+p_k}(p_k + 1)T(G^{(k)}; -p_k, 0) &= (p_k + 1)! \cdot P(G; 3) \\
(-1)^{n+p_k} \sum_{i=0}^{n+p_k} b_{i,0}(G^{(k)})(-p_k)^i &= p_k! \cdot P(G; 3)
\end{aligned}$$



G $\mathbf{K}_6$

Figure 3.A: $G^{(3)}$.

$$(-1)^{n+p_k} \sum_{i=1}^{n+p_k} b_{i,0}(G^{(k)})(-p_k)^{i-1} \quad = \quad (p_k - 1)! \cdot P(G;3).$$

Reducing both sides of this last equation modulo p_k gives

$$(-1)^{n+p_k} b_{1,0}(G^{(k)}) \equiv (p_k - 1)! \cdot P(G;3) \pmod{p_k}$$

and using Wilson's Lemma ($(p-1)! \equiv -1 \pmod{p}$) we reach

$$b_{1,0}(G^{(k)}) \equiv (-1)^{n+p_k+1} \cdot P(G;3) \pmod{p_k}.$$

Thus, since p_k is certainly odd,

$$b_{1,0}(G^{(k)}) \equiv (-1)^{n} \cdot P(G;3) \pmod{p_k}.$$

The construction of all the $G^{(k)}$ can certainly be performed in polynomial-time, as the primes less than n^2 can be found by a process of trial division: the time taken by this will be a polynomial in n.

Therefore, using an oracle that returns the coefficient of x in the Tutte polynomial of a graph, we can find the number of 3-colourings of any graph modulo all primes between 3 and n^2. Clearly, the number of 3-colourings of any graph containing an edge is even (and the edgeless graph on n vertices has exactly 3^n 3-colourings), as permuting the colours generates 6 distinct colourings corresponding to any particular partition of the vertices into colour classes. Using the Chinese Remainder Theorem, we can therefore calculate the number of 3-colourings of G modulo the lowest common multiple of the primes less than n^2, which is of course their product. As this product is greater than the number of possible 3-colourings of G, we know that the unique solution in $\{0, \ldots, 3^n\}$ is the actual number of 3-colourings of G. This number is exactly six times the number of essentially distinct 3-colourings of G, and we have shown that counting the number of essentially distinct 3-colourings is **#P**-hard since k-COL is parsimonious with SAT.

All the required arithmetic steps can easily be carried out in polynomial time: this is made simpler by the fact that our problem size is $O(n^2)$ rather than the more usual $O(\log n)$ for numerical problems. □

Since, as noted above, $b_{0,1}$ is equal to $b_{1,0}$, the coefficient of y is also **#P**-complete to compute. This hardness result can can be extended to include many more coefficients:

Corollary 3.3.3 *For all non-negative integers i, j, the coefficients $b_{1+i,j}$ and $b_{i,1+j}$ are both* **#P***-complete.*

Proof: Given a graph G, we construct a new graph H by adjoining a path of length i to G, consisting of new vertices $\{v_1, \ldots, v_i\}$ and edges $v_k v_{k+1}$, $1 \leq k < i$, and an edge uv_1 where u is an arbitrary vertex in G. We then add j loops at vertex v_i. The coefficient of $x^{i+1}y^j$ in the Tutte polynomial of H is equal to the coefficient of x in the Tutte polynomial of G, and so calculating it is **#P**-hard. Similarly, the coefficient of $x^i y^{j+1}$ in the Tutte polynomial of H is equal to the coefficient of y in the Tutte polynomial of G, and so is also **#P**-hard. □

In fact, we can go even further than this:

Corollary 3.3.4 *For all constants α, c, $0 \leq \alpha < 1$, $0 < c \leq 1$, $b_{\lfloor \alpha n+1 \rfloor,0}$ and $b_{\lfloor n-n^c+1 \rfloor,0}$ are both* **#P***-complete.*

Proof: As above, given any graph G, we add a path to it to form H. If we add a path of length $\lfloor \frac{\alpha n}{1-\alpha} \rfloor$, $\lfloor n^{1/c} - n \rfloor$ respectively, then it is simple to check that the given coefficient of the new graph H is equal to $b_{1,0}(G)$, and the transformation is a polynomial one. □

Similar results for large values of the second index of the coefficients can be obtained by adding loops to the graph, rather than isthmuses, and if we add both loops and isthmuses, we can easily reach results like the following:

Corollary 3.3.5 *Computing* $b_{\lfloor \frac{n-1}{2} \rfloor, \lfloor \frac{m-n+1}{2} \rfloor}(G)$ *is* **#P***-complete.*

3.4 Other coefficients, and an open problem

Some coefficients are not amenable to either of the approaches given here. For any increasing function $f(n)$, calculating $b_{n-1-f(n),j}$ by brute force calculation as used in Theorem 3.2.1 above will take a superpolynomial $\left(O\left(\binom{n}{f(n)}\right)\right)$ amount of time.

On the other hand, if the function increases too slowly (slower than n^c for all $c > 0$), the technique, described above, of adding a path and loops to the graph to change the indices of the coefficients of the polynomial will cause a superpolynomial expansion of the graph and hence not be a polynomial-time reduction.

An analogous situation arises in many other problems. For instance, counting the number of k-cliques of a graph is **#P**-complete for an arbitrary k [GJ79] but can be done in polynomial-time for any fixed constant k by brute force (just look at each set of k vertices of G in turn, and check if they form a clique). However, for $k = \lfloor \log n \rfloor$, the complexity is not clear. This suggests the following as an open problem:

Open Problem 3.4.1 *How hard is it to calculate* $b_{\lfloor n-1-\log n \rfloor, j}$*, for arbitrary integer* j*?*

3.5 A polynomial-time predicate for the coefficients of the Tutte polynomial

Perhaps surprisingly, given Theorem 3.3.1, it is still possible to find out some information about the lowest coefficient of the Tutte polynomial. For

instance, we show below that, for a connected graph G with at least 2 edges, $b_{1,0}(G) = 0$ precisely when G contains a loop or an articulation vertex (Lemma 3.5.5), and this is easy to check in polynomial-time.

Furthermore, it is straightforward to show (by induction on the number of edges) that, for all graphs G with at least 2 edges, $b_{1,0}(G) = 1$ precisely when G is a *series-parallel* graph. A series-parallel graph is one which can be formed by taking the 2-cycle, and performing a sequence of series-parallel extensions: that is, either adding a new edge in parallel with an edge already present, or adding a new vertex in the middle of an edge that is already present.

It is easy to check in polynomial-time if a particular graph is series-parallel. All that has to be done is to pick an edge of the graph, and consider the graphs formed when we either delete it or contract it. If and only if the original graph is series-parallel, then one of the two graphs so formed will also be series-parallel, and the other one will have a loop or articulation vertex (since $b_{1,0} \geq 0$ and $b_{1,0}(G) = b_{1,0}(G') + b_{1,0}(G'')$). This process can be repeated on the candidate series-parallel graph (i.e. the graph that does not contain a loop or articulation vertex) until the 2-cycle is reached. This means that it can be checked in polynomial time if $b_{1,0}(G) = 1$. We now extend this result:

Theorem 3.5.1 *The predicate "$b_{1,0}(G)$ is less than k" is in* **P** *for any fixed integer k.*

In order to prove Theorem 3.5.1, we need some preliminary lemmata. We begin with some results for connected matroids that can be found in [Oxl92]. We will restate them in the terminology of graph theory. The equivalent concept in graph theory to a connected matroid is a *block*.

Definition 3.5.2 *A* block *of a graph is a subgraph that is either a single*

loop togther with its incident vertex, or a maximal 2-vertex-connected and loopless subgraph.

Lemma 3.5.3 *If G is a block, then for any edge e in G, at least one of G'_e, G''_e is a block.*

Proof: Suppose G'_e is not a block, and let C_1 be one of its blocks. For any vertices $x \in C_1$ and $y \in G'_e \backslash C_1$, there is a cycle C_{xy} in G containing both x and y. This cycle must contain e, as there is no such cycle in G'_e. So $C_{xy} \backslash e$ is a cycle of G''_e containing both x and y. This shows that all $x \in C_1$ and $y \in G'_e \backslash C_1$ are contained within the same block in G''_e, and so it follows that G''_e is a block. □

A *minor* of a graph G is a graph M that can be obtained from a subgraph of G by contracting edges. We denote this by $M \leq G$.

Lemma 3.5.4 *For any block G, and any minor M of G that is also a block, there is a sequence of deletions and contractions that transforms G into M in which no loops or isthmuses are deleted or contracted.*

Proof: This is proved as Corollary 4.3.7 in [Oxl92], stated in terms of connected matroids. □

Now we can prove some lemmata necessary to complete the proof of Theorem 3.5.1.

Lemma 3.5.5 *For any graph G that contains at least 2 edges, G is a block if and only if $b_{1,0}(G) > 0$.*

Proof: First, we prove the right to left implication. It was shown by Tutte [Tut54] that if a graph is the union of some subgraphs that all intersect at a single vertex, then its Tutte polynomial is equal to the product

of the Tutte polynomials of these subgraphs (this is clear from inspection of the formula for the Tutte polynomial (1.3.2)). Clearly, if this is the case, then $b_{1,0}(G) = 0$.

For the left to right implication, note that it is trivially true for all blocks of 2 edges (there is only one such graph, the 2-cycle) and assume for a contradiction that G is a counterexample with the fewest possible number of edges, that is, G is a block and $b_{1,0}(G) = 0$. Pick an arbitrary edge e in G. It is neither an isthmus nor a loop, for G has none. We know from Lemma 3.5.3 above that at least one of G'_e and G''_e is a block, and so by the inductive hypothesis at least one of $b_{1,0}(G'_e)$ and $b_{1,0}(G''_e)$ is non-zero (and of course, both are non-negative). So, using the deletion/contraction formula (1.3.4) we see that $b_{1,0}(G) = b_{1,0}(G'_e) + b_{1,0}(G''_e) > 0$, and the contradiction has been reached. □

So, in order to determine whether the lowest coefficient of the Tutte polynomial of a graph is less than k, we only need to consider blocks.

We say a set of graphs is *minor-closed* if, for any graph in the set, all minors of that graph are also contained in the set.

The following observation is trivial:

Lemma 3.5.6 *The set $\mathcal{F}^k$ of blocks G for which $b_{1,0}(G) < k$, together with all minors of these blocks, is minor-closed.*

Proof: Any minor of a minor of a graph, is itself a minor of the original graph. □

Lemma 3.5.7 *For every graph $G \in \mathcal{F}^k$, $b_{1,0}(G) < k$.*

Proof: If there is a graph $H \in \mathcal{F}^k$ such that $b_{1,0}(G) \geq k$, then it must be a block, and also be a minor of some block G with $b_{1,0}(G) < k$.

Using the deletion/contraction formula (1.3.4) repeatedly, we can calculate the Tutte polynomial of a graph as the sum of the Tutte polynomials of smaller graphs.

By Lemma 3.5.4, we know that we can get from any block G to any minor H that is also a block with a sequence of deletions and contractions that do not create any loops or isthmuses. We order the edges of G according to this sequence, with the edges in the sequence preceding those not in it. When we apply the deletion/contraction formulae to the edges of G in order, one of the graphs created along the way will be H. Moreover, it will have been created by a sequence of deletions and contractions not involving loops or isthmuses, and so its contribution to the Tutte polynomial of G will not have to be multiplied by a power of x or y. This shows that it is possible to express the Tutte polynomial in the following way:

$$T(G) = \sum_{H \in \mathcal{H}} T(H)$$

where $\mathcal{H}$ is a set of minors of G including H. All the coefficients of the Tutte polynomial are non-negative integers, so $b_{1,0}(G) \geq b_{1,0}(H)$. Hence if G is a block which has x-coefficient less than k, then for all minors $H \leq G$, $b_{1,0}(H) < k$. □

Finally, we are ready to prove Theorem 3.5.1.

Proof of Theorem 3.5.1: We use the positive resolution of Wagner's Conjecture, as proved by Robertson and Seymour in [RS93]: *Any set of finite graphs contains only a finite number of minor-minimal elements.*

So there are only a finite number of minor-minimal graphs in the complement of $\mathcal{F}^k$. Let us denote by $\mathcal{O}^k$ the set of these graphs. We call this the *obstruction set for* $\mathcal{F}^k$. Note that although the existence of this set has been proved, the proof is nonconstructive: that is, no means of finding the set is known.

Since the set $\mathcal{F}^k$ is minor-closed, in order to check for membership in $\mathcal{F}^k$, all we need to do is check whether or not a candidate graph has any element of the obstruction set as a minor.

For a fixed M, Robertson and Seymour have also proved that there is a polynomial-time algorithm that, on input G, will decide whether or not $M \leq G$ [RS93]. So there is an algorithm which will check for all the possible "forbidden minors" in polynomial-time, and hence determine whether or not $b_{1,0}(G) < k$. □

Corollary 3.5.8 *The predicate "$b_{1,0}(G)$ equals k" is polynomial-time computable, for arbitrary fixed k.*

Proof: This is immediate, as $b_{1,0}(G) = k$ if and only if $b_{1,0}(G) < k+1$ and $b_{1,0}(G) \not< k$. □

We can again extend these results to cover more coefficients, in the following way.

We need to strengthen the result of Lemma 3.5.4 a little.

Lemma 3.5.9 *For any block G, and any connected minor M of G, there is a sequence of deletions and contractions that transforms G into M in which no loops or isthmuses are deleted or contracted.*

Proof: We use induction on the number of blocks of the minor M.

The base case, when the minor is itself a block, corresponds to Lemma 3.5.4.

Assume that it is true when the minor has at most i blocks. For a minor with $i+1$ blocks, we can find a sequence of deletions and contractions that realises M. Let us assume that this sequence minimises the number of loops and isthmuses removed. There is a first time in the sequence that we remove an edge e such that the resulting graph H has more than one block. If M

contains some edges from each of the blocks of H, then we can apply the induction hypothesis to each block of H. If H contains only two blocks, $H = H_1 \bigcup H_2$, and M is wholly contained within one of these blocks, H_1, then we can reorder the sequence of deletions and contractions so that we reach H_1 without having to remove a loop or isthmus (by Lemma 3.5.3), and then continue removing edges as in the original sequence to form M. This contradicts the assumption that our original sequence minimised the number of loops and isthmuses removed. The remaining case is that H contains more than two blocks, and the edge set of M does not intersect each of these blocks. In this case, then the last edge to have been removed to form H must have been contracted. Consider the previous graph in the sequence, H'. If all of the edges that will form the blocks that contain no edges needed to form M when e is contracted, are removed from H', then the graph spanned by the remaining edges will be a block, and so can be reached from G without removing any loops or isthmuses. M will still be a minor of this graph, and applying the remainder of the original sequence of deletions and contractions to this graph will result in a sequence with fewer loops and isthmuses removed than in the original sequence, which again gives us the required contradiction. □

Corollary 3.5.10 *The predicate "$b_{i,j}(G)$ is less than k" is computable in polynomial-time for arbitrary fixed i, j and k.*

Proof: Consider the set $\mathcal{F}^k_{i,j}$, consisting of all blocks G such that $b_{i,j}(G) < k$, plus all their minors. This set is of course minor-closed. As in the proof of Theorem 3.5.1, there can be no graph H contained in $\mathcal{F}^k_{i,j}$ such that $b_{i,j}(H) \geq k$, because if there was, it would have to be a minor of some block G with $b_{i,j}(G) < k$. This is impossible, as we can use Lemma 3.5.9 to show that we can express the Tutte polynomial of G as the sum of the Tutte polynomials of a set of graphs including H, as before.

So again we have an obstruction set $\mathcal{O}^k_{i,j}$, with a finite number of minor-minimal elements, and we can check in polynomial-time whether or not any of these forbidden minors is in fact a minor of the candidate graph. □

3.6 The Tutte polynomial modulo p.

We know that it is **#P**-complete to compute the Tutte polynomial of an arbitrary graph at any point except for a few easy points. Przytycka and Przytycki [PP93] show, using essentially the same counting argument as in Theorem 3.2.1, that various specialisations and generalisations of the Tutte polynomial arising in knot theory can be evaluated modulo some simple ideals, despite the fact that they are **#P**-hard to evaluate fully. This gives a class of polynomially-computable invariants for links. Here we consider the difficulty of computing the Tutte modulo a prime p. It must be **#P**-complete to compute the Tutte polynomial modulo a prime p if p is given as part of the input, except for at the easy points listed in [JVW90], since we can use the Chinese Remainder Theorem to work out the actual value of the Tutte polynomial if we are given the value modulo a large enough set of distinct primes.

However, for a particular fixed prime, there is no *a priori* reason why we should not be able to calculate the Tutte polynomial at a particular point modulo that prime, and indeed, Welsh [Wel93a] notes the following easy result:

Proposition 3.6.1 *The Tutte polynomial of any graph can be calculated modulo 2 at any point of the integer lattice.*

Proof: For any point (x, y), it is obvious that

$$T(G; x, y) \equiv T(G; x + 2k, y + 2l) \pmod 2$$

for any integers k and l, since $T(G;x,y)$ is a polynomial over the integers. So the parity of the Tutte polynomial evaluated at (x,y) is equal to the parity at one of the four points (0,0), (1,1), (-1,0), (0,-1) (depending on whether x and y are odd or even), and it is easy to evaluate the Tutte polynomial at these points. □

Our aim here is to examine how this result generalises for other primes.

For a fixed prime, the only tool that seems to be available to prove hardness is the result of Valiant and Vazirani that we gave in Chapter 1. In order to use this result, we need to have a parsimonious or at least a weakly parsimonious transformation from SAT to an evaluation of the Tutte. We have already given parsimonious reductions from SAT to all the points $(k,0)$, for all integers $k \leq -2$, via our parsimonious reduction from SAT to 3COL given in Chapter 1. We also note that there is a weakly parsimonious reduction from SAT to the point $(0,-2)$ since we have a weakly parsimonious reduction to the point $(-2,0)$ for planar graphs (Proposition 1.5.5), and we can then use duality to show that the point $(0,-2)$ is as hard as $(-2,0)$. However, we cannot use this method for other points along the y-axis since we cannot take the dual of a non-planar graph (at least, not without extending our domain to more general matroids).

We say (x,y) where x,y are integers is *congruent* to (x',y') modulo p if, for some integers k and l, $x = x' + pk$ and $y = y' + pl$.

Lemma 3.6.2 *For a fixed prime p, and any graph G, the Tutte polynomial of G can be evaluated modulo p, in polynomial time, at any point (x,y) of the integer lattice that is either congruent to one of the four points $(-1,0)$, $(0,-1)$, $(-1,-1)$, $(1,1)$ $\pmod p$ or that lies on the hyperbola $H_1^p = \{(x,y) : (x-1)(y-1) \equiv 1 \pmod p\}$.*

Proof: It is clear that

$$T(G;x,y) \equiv T(G;x',y') \pmod p$$

for any (x', y') congruent to (x, y) modulo p, and we know that the Tutte polynomial at any of the four listed "easy points" can be calculated in polynomial time, so the remainder modulo p can easily be found.

For any point on H_1^p and graph G on n vertices, we can write the Tutte polynomial in the following way (1.3.2):

$$\begin{aligned} T(G;x,y) &= \sum_{A\subseteq E} (x-1)^{r(E)-r(A)}(y-1)^{|A|-r(A)} \\ &\equiv \sum_{A\subseteq E} (x-1)^{r(E)-r(A)-(|A|-r(A))} \pmod{p} \\ &\equiv (x-1)^{r(E)} \sum_{A\subseteq E} (x-1)^{-|A|} \pmod{p} \\ &\equiv (x-1)^{r(E)} \left[\frac{x}{x-1}\right]^{|E|} \pmod{p} \\ &\equiv x^{|E|}(x-1)^{r(E)-|E|} \pmod{p} \end{aligned}$$

and so the Tutte polynomial can be evaluated modulo p at the point (x, y). □

This is exactly the same reasoning as that used to show that the Tutte polynomial can be evaluated on H_1, but with all the arithmetic carried out modulo p.

The following conjecture seems like a natural extension of the main result of [JVW90]:

Conjecture 3.6.3 *Under the assumption that* **NP** $\neq$ **RP**, *there is no randomised polynomial-time algorithm for evaluating the Tutte polynomial of an arbitrary graph modulo a fixed prime p at any point (x, y) of the integer lattice unless either of the following two conditions hold:*

1. *(x, y) is congruent to one of the points* $(0, -1)$, $(-1, 0)$, $(-1, -1)$, $(1, 1)$ $(\text{mod } p)$.

2. (x, y) *lies on the hyperbola* $H_1^p = \{(x, y) : (x-1)(y-1) \equiv 1 \pmod{p}\}$.

At any of the above points, the calculation can be carried out in polynomial time.

3.7 Partial results

We have shown that evaluating the Tutte polynomial modulo a fixed prime p is easy for the special points and the hyperbola H_1^p, but have been unable to prove that all the other points are hard. In this section we give partial results which give some support to our conjecture.

Proposition 3.7.1 *Unless* **NP** = **RP**, *there is no polynomial-time algorithm for evaluating the Tutte polynomial of an arbitrary graph G modulo a fixed prime p, at any point congruent modulo p to $(1, b)$ for any $b \not\equiv 1 \pmod{p}$.*

Proof: We use the ideas contained in [JVW90] to reduce the problem of evaluating the Tutte polynomial, modulo p, at a point at which this is known to be hard, to evaluating the Tutte polynomial, modulo p, at the point $(1, b)$.

Definition 3.7.2 *The k-stretch of a graph G, is the graph obtained by replacing each edge of G by a path of length k. We denote the graph thus obtained by ${}^kG$. The k-thickening of G is the graph obtained by replacing each edge of G by k edges in parallel, and is denoted by G^k.*

The following formula appears in [JVW90].

$$T({}^kG; a, b) = (1+a+\ldots+a^{k-1})^{n-r(G)} T\left(G; a^k, \frac{b + a + \ldots + a^{k-1}}{1 + a + \ldots + a^{k-1}}\right). \quad (3.7.3)$$

For $a = 1$ this simplifies to

$$T({}^kG; 1, b) = k^{n-r(G)} T\left(G; 1, \frac{b + (k-1)}{k}\right). \tag{3.7.4}$$

and so, if we choose $k \equiv 1 - b \pmod{p}$ then we get

$$T({}^kG; 1, b) \equiv (1-b)^{n-r(G)} T(G; 1-p, 0) \pmod{p}. \tag{3.7.5}$$

As $b \not\equiv 1 \pmod{p}$, we could therefore use an algorithm for the Tutte polynomial evaluated (modulo p) at the point $(1, b)$ to find out the Tutte polynomial of G evaluated (modulo p) at the point $(1-p, 0)$. Equation 1.3.11 gives us

$$p^{k(G)} T(G; 1-p, 0) = P(G; p).$$

For a graph G that is p-colourable, but not $p-1$-colourable, it is clear that the chromatic polynomial of G evaluated at p is equal to $p!$ times the number of essentially distinct p-colourings of G.

Using Wilson's Lemma again shows us that

$$T(G; 1-p, 0) \equiv -C_p(G) \pmod{p}$$

where $C_p(G)$ is the number of distinct p-colourings of G that use all p colours. The parsimonious reduction given in Chapter 1 (1.5.2) from SAT to k-COL ensures that the graph that is created is not $k-1$-colourable, so, in this case, $C_p(G)$ is equal to the number of essentially distinct p-colourings of G. Applying Valiant and Vazirani's theorem (1.2.12) shows us immediately that if we can evaluate $C_p(G)$ modulo p in polynomial time, then **NP** = **RP**. □

Unfortunately, the methods of Jaeger *et al.* will be of no use in showing that evaluation at the point $(a, 1)$ modulo p is hard, as their tranformations will only shift the point of evaluation along the line $y = 1$. We might expect that

evaluation at the point $(0,1)$ modulo p is hard, as it counts the number of p-flows modulo p, but we can only show this when $p = 3$, when it follows by duality from the hardness of counting the number of 3-colourings of planar graphs modulo 3. It is known that all isthmus-free graphs have a nowhere-zero 6-flow [Sey81]. Therefore, it cannot be hard to distinguish between graphs with none or exactly one 6-flow, and so we cannot use Valiant and Vazirani's result for large primes. The case $p = 5$ is uncertain: if Tutte's 5-flow conjecture is true, then again there is no immediate reason why the number of 5-flows should be hard to evaluate modulo 5.

We can however show that the following evaluations are hard:

Proposition 3.7.6 *Unless* **NP** = **RP**, *there is no polynomial-time algorithm to evaluate the Tutte polynomial of a graph, modulo p, for a fixed prime $p > 2$, at any point (a, y) or (x, a) of the integer lattice that lies on the hyperbola H^p_α for some $\alpha \notin \{0, 1\}$ and where a is a primitive root modulo p.*

Proof: Firstly we consider the point (a, y). As a is a primitive root, there exists a k such that $a^k \equiv 1 - \alpha \pmod{p}$. If we form the k-stretch of G as before, we have

$$T({}^kG; a, y) = (1+a+\ldots+a^{k-1})^{n-r(G)} T\left(G; a^k, \frac{y + a + \ldots + a^{k-1}}{1 + a + \ldots + a^{k-1}}\right). \quad (3.7.7)$$

As we know that the coefficients (a, y) and $(a^k, \frac{y+a+\ldots+a^{k-1}}{1+a+\ldots+a^{k-1}})$ lie on the same hyperbola H^p_α, and $a^k \equiv 1 - \alpha \pmod{p}$, it must be the case that

$$\frac{y + a + \ldots + a^{k-1}}{1 + a + \ldots + a^{k-1}} \equiv 0 \pmod{p}.$$

We know that the Tutte polynomial is hard to evaluate modulo p at the point $(1 - \alpha, 0)$ for α, $p > 2$ since the evaluation at that point is equal to the number of essentially distinct α-colourings of the graph modulo p. It is also

clear that the multiplicand $(1 + a + \ldots + a^{k-1})^{n-r(G)}$ in the above equation is non-zero modulo p, from the identity $1 - a^k = (1 - a)(1 + a + \ldots + a^{k-1})$ since we know that $a^k \not\equiv 1 \pmod{p}$. So if we could evaluate the Tutte polynomial of an arbitrary graph at the point (a, y) modulo p, then we could do the same at the point $(1 - \alpha, 0)$ which we know to be hard.

For a point of the form (x, a) on some hyperbola other than H_0^p or H_1^p, where a is a primitive root as before, we can extend the above method to show that these points are also hard.

Firstly, by constructing the k-thickening of G, we can shift the point of evaluation according to the following equation:

$$T(G^k; x, a) = (1+a+\ldots+a^{k-1})^{n-r(G)} T\left(G; \frac{x + a + \ldots + a^{k-1}}{1 + a + \ldots + a^{k-1}}, a^k\right). \quad (3.7.8)$$

Since we know that the value of a^k (as k varies) runs through all equivalence classes modulo p except for 0, and the point remains on the hyperbola H_α^p, it must be the case that $\frac{x+a+\ldots+a^{k-1}}{1+a+\ldots+a^{k-1}}$ runs through all the equivalence classes except 1. Hence, there is a k such that the value $\frac{x+a+\ldots+a^{k-1}}{1+a+\ldots+a^{k-1}}$ is a primitive root modulo p. This new point is a hard one, as we have just shown above. Also, the multiplicand in the equation above is non-zero, and so the point (x, a) is hard modulo the fixed prime p. □

3.8 Specific values of p

Using the results above, we can check that Conjecture 3.6.3 is true for $p = 3$, but even for $p = 5$ there are some points whose complexity is unclear. These points are the set $\{(x, 1) : x \neq 1\}$. For $p = 7$, we have 3 and 5 as primitive roots, and we can use *ad hoc* trial and error methods to find the required stretching and thickening operations to show that most points are

hard modulo 7, but the analogous set of points $\{(x,1) : x \neq 1\}$ causes the same problems as before.

If we could find suitable transformations, then we could reduce any point on H^p_α for $\alpha \neq 0$ to the point $(1-\alpha, 0)$, and this would prove that our conjecture above was true with the exception of the points on the line $y = 1$. Unfortunately, there does not appear to be a straightforward way of finding such transformations. The general function for such a transformation (a *tensor tranformation*) is given in [JVW90], but is unwieldy for complicated transformations.

Even if a method of finding such a transformation could be found, there would still remain the case of the points on the line $y = 1$. It is possible (though it does not seem likely) that for any fixed prime $p \geq 5$, we could evaluate the Tutte polynomial of any graph at any point $(x, 1)$ modulo p, if the time taken by algorithms for different values of p grew sufficiently rapidly for large values of p. This would be somewhat analogous with the situation described in Theorems 3.3.1 and 3.5.1 where the predicate "$b_{1,0}(G) < k$" was shown to be hard for variable k but easy for any fixed k. However, it seems less likely that this will be the case here, since we have already shown it to be hard to evaluate the Tutte polynomial modulo 3 at these points.

The following questions are therefore still to be resolved.

Open Problem 3.8.1 *What is the complexity of counting the number of forests (or number of nowhere-zero p-flows) of a graph modulo p, for a fixed prime $p \geq 5$?*

Chapter 4

Randomised approximation

We have met many counting functions that are probably hard to evaluate exactly. It is interesting to consider whether they can be approximately evaluated. We will explain what we mean by an "efficient" approximation algorithm, and then consider some counting problems that arise naturally from the Tutte polynomial.

We concern ourselves principally with the number of forests of a graph. Sections $4.1 - 4.6$ contain a description of an efficient approximation algorithm for the number of forests of dense graphs, and a generalisation to the approximation of the Tutte polynomial at any point $(x, 1)$ for rational $x \geq 1$. These results are to appear in [Anna].

The last section considers the problems of counting the number of forests with a particular number of components, and the number of connected spanning subgraphs with a particular number of edges. These problems are generalisations of a question posed by Colbourn [Col93]. A simple approximation algorithm is given for some cases.

4.1 Definitions and introduction

If we are going to attempt to calculate an approximate answer, it is reasonable to also introduce an element of randomness into our model of computation. This can be shown to add a significant amount of power to approximation algorithms (for instance approximating the volume of a convex body), in contrast to the fact that it is not known whether or not $\mathbf{RP} = \mathbf{P}$.

The standard model for randomised computation is the probabilistic Turing machine (PTM) introduced by Gill [Gil77]. A PTM is a Turing machine equipped with an output tape and with special *coin-tossing* states. Each coin-tossing state has two possible transitions, and when such a state is reached in a computation, the transition is chosen by the toss of a fair coin. The output of such a machine is just the contents of the output tape if the PTM reaches an accepting configuration. However, it is easily seen that any computation of the machine occurs with probability exactly $i2^{-j}$ for some integers i, j, and so it is not possible in this model to choose between exactly 3 possible outcomes equiprobably.

It is usually much more convenient to use a more general model, which Sinclair [Sin93] calls an oracle coin machine (OCM). In an OCM the unbiased coin is replaced with a biased coin, the bias of which is set by the Turing machine itself as the ratio of two integers written on a special *bias tape*: if r, s are integers written in binary on the bias tape and $r \leq s$, then one transition is picked with probability r/s and the other with probability $1 - r/s$.

The OCM can of course choose between 3 outcomes equiprobably, and so is more powerful in general. Fortunately, it does not matter whether we use the PTM or OCM for our algorithms: Sinclair also shows that if a fpras or an almost uniform generator exists using an OCM, it can be modified to run on a PTM. So, for convenience, we will use an OCM.

The following definition is the standard model of an efficient approximation algorithm.

Definition 4.1.1 ([KL83]) *A* fully polynomial randomised approximation scheme (fpras) *for a function f is a randomised algorithm $\mathcal{A}$ which, on input (x, ϵ), with probability greater than $3/4$, produces an output $\mathcal{A}(x, \epsilon)$ satisfying*

$$(1+\epsilon)^{-1} \leq \frac{\mathcal{A}(x,\epsilon)}{f(x)} \leq 1+\epsilon$$

and which runs in time polynomial in $|x|$ and ϵ^{-1}.

The probability of the algorithm failing to produce a sufficiently accurate approximation can be decreased from the value of $1/4$ above to any $\delta > 0$, in exchange for an increase in the running time of the algorithm by a factor of $O(\log(\delta^{-1}))$, by the simple method of running the algorithm $O(\log \delta)$ times, and taking the median output as the answer. If this process is incorporated into the algorithm, then it is often called an *epsilon-delta approximation scheme.*

It is clear that there is little hope of finding a fpras that will approximately evaluate the Tutte polynomial at the point (-2,0), since this would tell us whether or not a graph could be properly 3-coloured. This is of course an **NP**-complete problem and so is generally thought likely to be intractable, even using randomised algorithms. This suggests the following obvious result:

Proposition 4.1.2 *If it is* **NP***-hard to decide whether f is non-zero, then there cannot exist a fpras for f unless* $\mathbf{NP} = \mathbf{RP}$.

However, if the associated existence problem is easy, there is no *a priori* reason why we should not be able to approximately evaluate hard functions using randomised algorithms, and much attention has focussed in this direction over recent years.

Jerrum, Valiant and Vazirani [JVV86] show that, for a large class of problems (ones which are *self-reducible* [Sch76]), approximate counting of solutions

and almost uniform generation of solutions are polynomially equivalent. A *uniform generator* is a randomised algorithm that, given an instance of a problem in **NP**, generates uniformly at random one of the "witnesses" for that instance, and produces some output at least half the time. *Almost uniform generation* is a slightly weaker condition, only requiring that the output distribution is "close to" uniform (in a well-defined sense).

Using this idea, Jerrum and Sinclair [JS89] present a fpras for computing the permanent of dense matrices with all entries 0 or 1. The permanent of an $n \times n$ matrix is equal to the number of perfect matchings of a particular bipartite graph on $2n$ vertices. The vertices of the graph can be partitioned into two equal-sized sets of vertices, $\{u_1, \ldots, u_n\}$ and $\{v_1, \ldots, v_n\}$ such that each edge of the graph has its endpoints in different sets, and there is an edge between u_i and v_j if and only if the value in the ith row and jth column of the matrix is 1.

For a matrix to be dense (according to their definition), at least half the entries in each of its rows and columns must be ones. Exactly counting the number of perfect matchings in the graphs corresponding to these matrices was shown to be **#P**-hard by Broder [Bro86].

Some progress has also been made in approximating Tutte invariants of graphs. Jerrum and Sinclair again [JS90] present a fpras for the ferromagnetic Ising partition function of a graph, while showing that no fpras exists for the antiferromagnetic Ising partition function unless **NP** = **RP**. The ferromagnetic case corresponds to the branch of the hyperbola H_2 in which x and y are both greater than 1 and the antiferromagnetic case to the part of the other branch for which x is less than or equal to -1, and y therefore lies between 0 and 1. Attempts to generalise the positive result to the more general q-state Potts model have not so far succeeded, but it can be shown that there is no fpras for the antiferromagnetic Potts model (the section of the hyperbola for which $0 \leq y \leq 1$) unless **NP** = **RP**, for essentially the same reasons as for the Ising model.

Perhaps less well-known are the results of Edwards [Edw86] concerning 3-colourings of dense graphs. We use as our measure of denseness the minimum degree of a vertex of G, as a proportion of the total number of vertices:

Definition 4.1.3 *A graph G is* α-dense *(or* has density α*) if every vertex in G has at least $\alpha\,|V(G)|$ neighbours. We write this as $G \in \mathcal{G}_\alpha$. A family of graphs is* dense *if each member of the family is α-dense for some constant $\alpha > 0$.*

Edwards shows that if $\alpha > 1/2$, then deciding whether an α-dense graph is 3-colourable is in **P**, and also that the number of 3-colourings can be counted exactly in polynomial-time in this case. However, for $\alpha < 1/2$, the decision problem remains **NP**-complete, and so we cannot hope for an approximation algorithm (as explained above). The number of k-colourings of a graph can easily be calculated from the Tutte polynomial evaluated at the point $(1-k, 0)$ and the results for $k = 3$ generalise to any integer $k > 3$, with the threshold between easiness of counting and hardness of approximation being $\frac{k-2}{k-1}$ in general.

There are two general points that can be inferred from these results. Firstly, it appears that problems become easier when we restrict our attention to dense graphs. This was noted by Dahlaus, Hajnal and Karpinski [DHK93], who give a very fast parallel algorithm to construct a Hamiltonian circuit in a $1/2$-dense graph. Dyer, Frieze and Jerrum have recently created a fpras for approximating the number of Hamiltonian circuits in these graphs [DFJ93].

Secondly, these results suggest a possible frontier between the points in the Tutte plane at which approximation is possible, and those at which it is not. In [Wel93a], Welsh makes the following conjecture:

Conjecture 4.1.4 *There exists a fpras for computing the Tutte polynomial at each (rational) point (x, y) of the positive quadrant $x \geq 1$, $y \geq 1$.*

The number of forests is a point on the boundary of this region, which also includes the ferromagnetic case of the general q-state Potts model.

Recent work of Welsh himself lends further support to this conjecture: in [Wel93b], he shows that, for dense graphs, the Tutte polynomial can be approximated at any point (x, y) in the area defined by

$$\{(x, y) : \ x \geq 1, \ y > 1, \text{ and } (x-1)(y-1) \leq 1\}.$$

Further work of Welsh with Alan Frieze and Noga Alon has since led to the elimination of the constraint $(x-1)(y-1) \leq 1$ in the above. This result, in conjunction with the results of Sections 4.2-4.6, shows that Welsh's conjecture is indeed true when restricted to dense graphs.

However, it does not appear to be possible to extend the methods used to include all graphs.

There is no obvious reason why we should not be able to approximate the Tutte polynomial in the whole positive quadrant $x \geq 0$, $y \geq 0$, as there are no hard decision problems that could cause problems along the lines of Proposition 4.1.2. This region includes the point $(2,0)$ which counts the number of acyclic orientations of a graph, and also the points $(1,0)$ and $(0,1)$, which evaluate to the lowest coefficients of the chromatic and flow polynomials respectively. Of these, perhaps $(2,0)$ is the most interesting, and so we pose the following open problem:

Open Problem 4.1.5 *Does there exist a fpras for counting the number of acyclic orientations of a (dense) graph?*

It is possible that the restriction to dense graphs will make this easier to answer: however, the most obvious approaches do not appear to work.

4.2 Forests in dense graphs

We now consider the problem of approximately counting the number of forests in a dense graph.

We will show in Section 4.5 that we cannot *exactly* count the number of forests of a dense graph in polynomial-time, unless **NP** = **RP**.

Initially, we will restrict our attention to simple graphs: we will consider graphs containing multiple edges in section 4.6.

Given a graph $G \in \mathcal{G}_\alpha$ on vertices $u_1, \ldots, u_n$, we form a new graph G^+ by adding a new vertex v to the vertex set of G, and adding edges (v, u_i) connecting each vertex of G to v. We call this the *join* of G and v.

It is well known that there are fast (polynomial-time) algorithms for generating spanning trees of any graph uniformly at random. We are interested in the degree of the vertex v in the random spanning tree of G^+ (which we write as T_R^+). We need to prove the following lemma and corollary.

Lemma 4.2.1 *The probability that the edge (v, u_i) is contained in a randomly chosen spanning tree of G^+ is at most $2/(\alpha n + 2)$, for $1 \leq i \leq n$.*

Corollary 4.2.2 *The probability that the degree of v in T_R^+ is greater than $4/\alpha$ is less than $1/2$.*

Proof of Lemma 4.2.1: We use the theory of electrical networks to bound the probability that a particular edge (v, u_i) is in T_R^+. We form an electrical network from G^+ by replacing every edge by a resistor of resistance 1 Ω. The following two propositions are needed:

Proposition 4.2.3 *If the resistance of any resistor of a network is increased, the effective resistance between any two points cannot decrease. Similarly, if any resistance is decreased, the effective resistance cannot increase.*

Proposition 4.2.4 *The probability that any edge $e \in G$ is contained in a randomly seleted spanning tree of a graph is equal to the effective resistance between the endpoints of e in the associated electrical network.*

A proof of Proposition 4.2.3 is given in [DS84] as the monotonicity law, but it dates back a long time before this (as Rayleigh's Principle). Proposition 4.2.4, proved in [BSST40], is due in principle to Kirchhoff.

Now we find an upper bound for the resistance of G^+ between u_i and v. The vertex u_i has at least αn neighbours in G, each of which is connected to v. By Proposition 4.2.3, if we remove every resistor except for the ones connecting u_i to its neighbours, and the resistors connecting v to the neighbours of u_i in G, we can only increase the resistance between v and u_i. But we are left with at least αn disjoint paths of length 2 and a path of length 1 (the edge vu_i), giving a total resistance of $2/(\alpha n + 2)\ \Omega$, which, by Proposition 4.2.4, completes the proof of Lemma 4.2.1. □

Proof of Corollary 4.2.2: The expected degree of vertex v in T_R^+ is equal to the sum, over all edges e incident with v, of the probability that that edge is in the random tree T_R^+. By Lemma 4.2.1, we have shown that each of these probabilities is less than $2/\alpha n$. As there are only n edges incident with v, the expected degree of v is less than $2/\alpha$.

Letting d denote the degree of v in T_R^+, we can write:

$$E(d) = E(d \mid d > 4/\alpha) \cdot P(d > 4/\alpha) + E(d \mid d \leq 4/\alpha) \cdot P(d \leq 4/\alpha).$$

So, taking just the first term in the sum, we have:

$$2/\alpha > E(d) \geq 4/\alpha \cdot P(d > 4/\alpha)$$

or

$$P(d > 4/\alpha) < 1/2.$$

□

4.3 An algorithm for the uniform generation of forests

We now present an algorithm for the uniform generation of forests of a dense graph.

A spanning tree T of G^+ induces in the obvious way a forest F of G, given by $F = T \backslash v$. This mapping is clearly surjective, but not in general injective. The following lemma tells us how many spanning trees there are in G^+ that induce a particular forest F of G.

For any forest F with connected components $F_1, \ldots, F_c$ we say that the *size* of F_i is $|V(F_i)|$.

Lemma 4.3.1 *The number of spanning trees of G^+ that induce a given forest of G is equal to the product of the sizes of the connected components of the forest.*

Proof: Consider an arbitrary component of the forest, of size s say. Exactly one of its vertices must be directly connected to v in any tree that induces the forest. As there is one edge from each vertex in G to v, there are exactly s ways in which this can occur, independently of how the other components are connected to v. □

We define a binary relation as follows:

Definition 4.3.2 *For a spanning tree T of G^+ and forest F of G, we say T* is related to *F (denoted $\mathbf{R}\langle T, F\rangle$) if the following two conditions hold:*

1. $T \backslash v = F$
2. *In T, v is connected to the vertex of lowest index in each component of F.*

It is easy to see that, for any forest F of G, there is exactly one spanning tree T of G^+ such that $\mathbf{R}\langle T, F\rangle$.

The algorithm for the uniform generation of forests in G is given in Figure 4.A. We assume that the subroutine $\mathrm{SpT}(H)$ returns a uniformly generated random spanning tree of any graph H. This can certainly be achieved in polynomial time using an OCM. It is possible that the algorithm will produce more than one forest: in this case, we simply take the first forest that it outputs, and disregard the others.

```
(1)     G+ := join of G and v
(2)     for c := 1 to n^{4/α} do begin
(3)         T := SpT(G+)
(4)         F := T\v
(5)         if R⟨T, F⟩ then output F
(6)     end
```

Figure 4.A: The uniform forest generator, FGEN_α

Theorem 4.3.3 *The following statements are true:*

1. *$FGEN_\alpha$ generates each forest of G equiprobably.*
2. *For any $G \in \mathcal{G}_\alpha$, $FGEN_\alpha$ outputs a forest with probability greater than $1/2$.*
3. *$FGEN_\alpha$ runs in polynomial-time.*

Proof: Clearly, the algorithm runs in polynomial-time. To prove the first statement consider a single repetition of the **do** loop. For every forest F

of G, there is exactly one spanning tree T of G^+ such that $\mathbf{R}\langle T, F\rangle$. Each tree in G^+ is picked with equal probability, so each forest in G is output equiprobably.

For the second statement, from Corollary 4.2.2 we know that with probability greater than $1/2$, the degree of v in the randomly chosen tree T is no greater than $4/\alpha$. In this case, there are at most $(\alpha n/4)^{4/\alpha}$ spanning trees in G^+ that induce the forest $T\backslash v$. In one repetition of the loop, the probability that T is related to F, and so F is output in line (5), is therefore at least $1/2 \cdot 1/(\alpha n/4)^{4/\alpha} > n^{-4/\alpha}$. Hence the probability that the algorithm fails to produce a forest in every iteration of the loop is bounded above by $(1 - 1/n^{4/\alpha})^{n^{4/\alpha}} < e^{-1} < 1/2$. □

4.4 Approximating the number of forests

We now combine the above with the idea of self-reducibility, and a modified version of the algorithm of Jerrum, Valiant and Vazirani [JVV86] to construct a polynomial-time algorithm to approximate the number of forests of $G \in \mathcal{G}_\alpha$. We write $f(G)$ for the number of forests of any graph G.

Firstly, a reminder of an earlier result:

Lemma 4.4.1 *The number of forests of the complete graph K_n can be calculated in polynomial-time.*

Proof: We have the following recurrence relation for the Tutte polynomial of the graph $G_{r,m}$ (as defined in Chapter 2) evaluated at the point $(2, 1)$, by substitution into Proposition 2.2.2:

$$F_{r,m} = \sum_{i=1}^{m-1} \binom{m}{i} r^i F_{i,m-i} + F_{1,m-1} + 1, \ m \geq 2$$

$$F_{r,1} = r + 1$$

where $F_{r,m} = f(G_{r,m})$ for positive integers r and m, and so $F_{1,n-1} = f(K_n)$.

It is clear that to evaluate $F_{1,n-1}$ we only need to evaluate the $F_{r,m}$ for which $r+m \leq n-1$. So we must use the recurrence relation $O(n^2)$ times in all, each time summing $O(n)$ terms. This is certainly a polynomial-time calculation, the exact time taken depending on the model of computation used. □

We write $f(G)$ in the following way:

$$f(G) = f(K_n) \cdot \frac{f(K_n \backslash e_1)}{f(K_n)} \cdot \frac{f(K_n \backslash e_1, e_2)}{f(K_n \backslash e_1)} \cdots \frac{f(G)}{f(G \cup e_l)},$$

where the edges $e_1, \ldots, e_l$ that are successively deleted to create the sequence of graphs are exactly the edges in $K_n \backslash G$ (in any order).

For any dense graph $G \in \mathcal{G}_\alpha$ and edge $e \notin G$, we approximate the fraction $f(G)/f(G \cup e)$ by the simple method of generating forests in $G \cup e$ uniformly at random, and counting the proportion of these that are in fact contained in G. So we approximate independently each term in the product above, and if each approximation is accurate enough, then their product will approximate the number of forests in G within ratio $1+\epsilon$. The algorithm for approximating the number of forests of a dense graph is given in Figure 4.B.

Theorem 4.4.2 *The algorithm $APPROX_\alpha(G, \epsilon)$ is a fpras for the number of forests of any graph $G \in \mathcal{G}_\alpha$.*

Proof: We have to show that two things happen simultaneously with probability at least 3/4:

1. The algorithm produces an output.
2. This output approximates the number of forests of G within the required ratio $1 + \epsilon$.

(1) $H := K_n$
(2) $\Pi := f(H)$
(3) $t := 180n^6/\epsilon^2$
(4) **while** $H \neq G$ **do begin**
(5) Let e be any edge in H but not in G
(6) make $3t$ calls to $\mathrm{FGEN}_\alpha(H)$
(7) **if** at least t of the trials yield an output **then**
(8) let $S = \{y_1, \ldots, y_t\}$ be the first t outputs of $\mathrm{FGEN}_\alpha(H)$
(9) **else halt**
(10) $\beta := |\{y \in S : e \notin y\}| / t$
(11) $H := H \backslash e$
(12) $\Pi := \Pi \times \beta$
(13) **endwhile**
(14) **output** Π

Figure 4.B: The fpras for forests, $\mathrm{APPROX}_\alpha(G, \epsilon)$.

Clearly the algorithm runs in polynomial time: the main loop is repeated n^2 times at most.

We know that FGEN fails to produce a forest with probability at most $1/2$. Consider the $3t$ calls to FGEN in one loop of the algorithm, and let the random variable F be the number of times that no output is given by FGEN. Clearly $E(F) \leq 3t/2$ and $Var(F) \leq 3t/4$, and hence, using Chebyshev's inequality we have $P(F > 2t) < 3/t$. Since t is certainly bounded below by $180n^2$, the probability that some output is produced is greater than $(1 - 1/60n^2)^{n^2}$, which is easily shown to be greater than $59/60$.

Now we claim that on every iteration of the loop, with high probability, β will approximate $f(H \backslash e)/f(H)$ within ratio $1 + \epsilon/2n^2$. Consider the random variable $X = |\{y \in S : e \notin y\}|/t$. Let $\mu = E(X)$. Since X is the average of t independent $0,1$-valued random variables, $Var(X) \leq 1/t$. Using Chebyshev's inequality again we see

$$P\left(|X - \mu| \leq \epsilon/6n^2\right) > 1 - 36n^4/\epsilon^2 t \geq 1 - 1/5n^2.$$

Now μ is never less than $1/2$: to see this, consider the set of all forests of $\hat{G} = G \bigcup e$. These forests can be partitioned into two sets; the forests that include e, and the ones that do not. The forests that do not include e are precisely the forests of G. There is a natural isomorphism between the forests that contain e, and the forests of $\hat{G}''_e$, and of course $f(\hat{G}) = f(\hat{G}'_e) + f(\hat{G}''_e) \geq f(\hat{G}''_e)$. This allows us to bound the *relative* error of β, as we now have:

$$P\left(\left|\frac{\beta}{\mu} - 1\right| \leq \frac{\epsilon}{3n^2}\right) \geq 1 - \frac{1}{5n^2}.$$

Since $\epsilon/n^2 \leq 1$, it is easy to see that $(1 + \epsilon/2n^2)^{-1} \leq (1 - \epsilon/3n^2)$. So, with probability at least $1 - 1/5n^2$, β approximates μ within ratio $1 + \epsilon/2n^2$.

The probability of β being such an accurate approximation in every iteration is at least $(1 - 1/5n^2)^{n^2}$, which is greater than $4/5$. If β is this accurate

in each iteration, then the final output Π approximates $f(G)$ within ratio $(1+\epsilon/2n^2)^{n^2}$ which is no worse than $1+\epsilon$. So the probability of the algorithm producing an output *and* the output being sufficiently accurate is at least $\frac{59}{60} \cdot \frac{4}{5} > \frac{3}{4}$, as required. □

4.5 Exact counting is hard

The proof in [JVW90] that evaluating the Tutte polynomial of a graph is **#P**-hard at all but a few points does not demonstrate the hardness of counting the number of forests of dense graphs, as their proof relies on the hardness of the planar case, which follows by duality from the hardness of computing the reliability of a planar graph. However, by using a different approach, we now show that, unless **NP** = **RP**, we cannot exactly count the number of forests of dense graphs in polynomial time.

Proposition 4.5.1 *Counting the number of forests of dense graphs cannot be done in polynomial time, unless* **NP** = **RP**.

Proof: The proof consists of a series of reductions, which we give as a series of lemmas.

Lemma 4.5.2 *Calculating the constant coefficient of the flow polynomial of a dense graph is Turing reducible to counting the number of forests of dense graphs.*

Proof: It can easily be checked from (1.3.12) that the constant coefficient of the flow polynomial of a graph G is given (up to an easily computable sign) by the Tutte polynomial of G evaluated at the point $(0, 1)$. The number of forests is given by the Tutte polynomial of G evaluated at the point $(2, 1)$.

If we consider the Tutte polynomial evaluated along the line $y = 1$, it is clear that it will be a polynomial of degree $|V| - k(G)$. If a graph G is dense, then clearly the k-thickening of the graph (3.7.2) will also be dense. Therefore, if we have an algorithm that calculates the number of forests of a dense graph, then we can use it to obtain the number of forests of the k-thickening of G, for any $k \geq 1$. If we write G^k for the k-thickening of G, then (3.7.8) simplifies to:

$$T(G^k; 2, 1) = kT(G; 1 + \frac{1}{k}, 1).$$

Therefore, if we can count the number of forests of the k-thickening of G, for all values of k between 1 and $|V|$, we can calculate the Tutte polynomial of G along the line $y = 1$ by Lagrangian interpolation. This then enables us to compute the constant coefficient of the flow polynomial of G, by setting $x = 0$. □

Lemma 4.5.3 *Calculating the number of nowhere-zero 3-flows of a planar graph, modulo 3, is polynomially reducible to calculating the constant coefficient of the flow polynomial of a dense graph.*

Proof: Note that the number of nowhere-zero 3-flows of a graph G, modulo 3, is equal to the constant coefficient of the flow polynomial of that graph taken modulo 3.

Given an arbitrary graph G', we construct a new graph G'' from it by adding 3 edges in parallel between every pair of vertices that were not already adjacent in G'. G'' is certainly dense, as every vertex is connected to every other vertex in it.

We now show that

$$F(G'; 3) \equiv F(G''; 3) \pmod 3. \tag{4.5.4}$$

For simplicity, we orient the edges of G' and G'' in the direction of the ordering of the vertices. Any nowhere zero 3-flow of G'' induces in a natural way a 3-flow (*not* necessarily a nowhere-zero 3-flow) on the complete graph K_n, by deleting two of the edges from each bundle of three parallel edges in G'', and adding the flows that they carried to the single remaining edge (where the addition is of course carried out modulo 3). We call two nowhere-zero 3-flows on G'' *similar* if they induce the same 3-flow on the complete graph in this manner. It is clear that this property of similarity is in fact an equivalence relation, and so we can use this to partition the nowhere-zero 3-flows of G'' into equivalence classes.

The size of each of these equivalence classes depends on the net flow of each bundle in G''. If a particular bundle carries a net flow of 0 (modulo 3), then there are exactly 2 ways of assigning flows to the three edges in the bundle in any nowhere-zero 3-flow for G'': either all of the edges carry a flow of 1, or all of the edges carry a flow of 2. However, if a particular bundle carries a net flow of 1 (or 2), then there are exactly 3 ways of assigning flows to the edges of the bundle: the three edges must carry flows of 1, 1 and 2 between them (or 1, 2 and 2) and there are exactly 3 distinct ways in which these flows can be arranged in each case.

So the number of nowhere-zero 3-flows of G'' that are contained in a single equivalence class C is given by

$$|C| = 3^i 2^j$$

where i is the number of bundles with non-zero net flow, and j is the number of bundles with a net flow of zero.

The total number of nowhere-zero 3-flows of G'' is of course simply the sum of the cardinalities of all the equivalence classes. To evaluate this sum modulo 3, we can ignore any equivalence class whose cardinality is a multiple of 3.

For the cardinality of an equivalence class to not be a multiple of three, there must be no bundles in all the members of that class with a non-zero net flow. The flow induced on the complete graph by the members of such an equivalence class must therefore take the value 0 on each of the edges that replaced a bundle of G'', and take a non-zero value on each edge that did not replace a bundle. In other words, the support of the induced flow is the edge set of the original graph G', and so the induced flow corresponds to a nowhere-zero 3-flow of G'. This correspondence between these particular equivalence classes and the nowhere-zero 3-flows of G' is clearly 1-1. Furthermore, each of these equivalence classes has the same cardinality (2^j where j is the number of bundles in G''). It follows that

$$F(G''; 3) \equiv 2^j \cdot F(G'; 3) \pmod 3$$

and substituting -1 for 2 in this leads to the equation given above (4.5.4).

So, if we could calculate the constant coefficient of the flow polynomial of a dense graph then we could count the number of nowhere-zero 3-flows of *any* graph, modulo 3. □

Lemma 4.5.5 *If it is possible to evaluate the number of 3-flows (modulo 3) of a planar graph in polynomial time, then* **NP** = **RP**.

Proof: This follows immediately from Proposition 1.5.7 by duality. □

To prove Proposition 4.5.1, we simply combine Lemmas 4.5.2, 4.5.3 and 4.5.5. □

4.6 Further results

We now use the ideas of Jaeger *et al.* [JVW90] to generalise our results a little. We can approximate the Tutte polynomial at other points along the

line $y = 1$, and we can also attempt to apply our ideas to graphs that are not dense.

Firstly, however, we must explain how multiple edges can be handled.

Our original algorithm will not work, since a graph with multiple edges is not contained within K_n. However, it will be contained within the k-thickening of K_n, where k is the greatest multiplicity of any edge of the graph. We can simply modify our algorithm to begin with the k-thickening of K_n, and again remove edges one at a time until G is reached, each time approximating the ratio of the number of forests in successive graphs so formed.

We will also need to know the number of forests of the k-thickened complete graph. Equation 3.7.8 tells us how to do this: it simplifies to

$$T(K_n^k; 2, 1) = kT(K_n; 1 + \frac{1}{k}, 1),$$

and our recurrence relation for the Tutte polynomial of the complete graph (2.2.2) evaluated at the point $(x, 1)$ simplifies to

$$T(G_{r,m}; x, 1) = \sum_{i=1}^{m-1} \binom{m}{i} r^i T(G_{i,m-i}; x, 1) + (x-1)T(G_{1,m-1}; x, 1) + 1, \ m \geq 2 \tag{4.6.1}$$

$$T(G_{r,1}; x, 1) = x + r - 1.$$

Hence we can create a fpras for the number of forests of dense graphs with multiple edges. The running time will depend polynomially on the greatest multiplicity of an edge.

We can now approximate the Tutte polynomial at other points along the line $y = 1$:

Theorem 4.6.2 *For any rational $x \geq 1$, there exists a fpras for evaluating the Tutte polynomial of dense graphs at the point $(x, 1)$.*

Note that the running time of our algorithm may not be bounded by a polynomial function of x and the other inputs, and so x cannot be input as a variable.

Proof: Firstly, we prove Theorem 4.6.2 for positive integer values of x. It has already been demonstrated for $x = 1$ and $x = 2$. For $x \geq 2$, we take our input graph G and form a new graph by adding a new vertex v, and connecting it to every vertex in G with a bundle of $x - 1$ edges in parallel. We call the new graph thus obtained the $x - 1$-join of G and v, and denote it by $G^{(x-1)+}$. Note that $G^{(1)+}$ is just the same as G^+.

We now describe an algorithm $\mathrm{FGEN}_\alpha^{x-1}$, which outputs a randomly selected forest of G according to a biased distribution. This will enable us to approximate the Tutte polynomial of G evaluated at the point $(x, 1)$ in a similar manner to our approximation for the number of forests.

The new algorithm $\mathrm{FGEN}_\alpha^{x-1}$ is given in Figure 4.C. We now consider the probability of a particular forest F being output in a single iteration of the loop. Recall that we write $k(F)$ for the number of connected components of the forest *including isolated vertices.* There are exactly $(x-1)^{k(F)}$ spanning trees in $G^{(x-1)+}$ that are related to a forest F (using the binary relation $\mathbf{R}$ that we have already defined), as, for each component of F, there are $x-1$ possible edges in $G^{(x-1)+}$ connecting the vertex of smallest index in this component to v. So the probability of a particular forest being generated in a single iteration of the main loop is proportional to $(x-1)^{k(F)}$. It is straightforward to show, in a similar manner to the method used for Corollary 4.2.2, that for at least half of the spanning trees of $G^{(x-1)+}$, the degree of the distinguished vertex v is less than $2x/\alpha$. So, as for FGEN_α, with probability greater than $1/2$ this algorithm produces some output. The running time of this algorithm is where the exponential dependence on x arises: this is necessary in order to ensure that some output is produced (with a high enough probability).

We now look again at the equation for the Tutte polynomial of a graph that we gave in Chapter 1 (Definition 1.3.2):

(1) $G^{(x-1)+} := (x-1)$-join of G and v
(2) **for** $c := 1$ to $n^{2x/\alpha}$ **do begin**
(3) $T := \mathrm{SpT}(G^{(x-1)+})$
(4) $F := T \backslash v$
(5) **if** $\mathbf{R}\langle T, F\rangle$ **then output** F
(6) **end**

Figure 4.C: The forest generator, $\mathrm{FGEN}_{\alpha}^{x-1}$

$$T(G) = \sum_{A \subseteq E} (x-1)^{r(E)-r(A)} (y-1)^{|A|-r(A)}.$$

At the point $(x, 1)$, the summand is non-zero only when $|A| = r(A)$, that is, when A is the edge set of a forest of G. So we can rewrite this expression as:

$$T(G) = (x-1)^{r(E)} \sum_{A} (x-1)^{-r(A)} \tag{4.6.3}$$

where the summation is over the edge sets A of all forests of G. Substitution of our definition of rank given in the first chapter (1.3.1) into this equation gives

$$T(G; x, 1) = (x-1)^{-1} \sum_{A} (x-1)^{k(A)}. \tag{4.6.4}$$

So the Tutte polynomial of G evaluated at $(x, 1)$ is just $1/(x-1)$ times the weighted sum over all the forests of G, with the weight of a forest F equal to $(x-1)^{k(F)}$. This is precisely the weighting given to the forests using the random generation algorithm $\mathrm{FGEN}_{\alpha}^{x-1}$.

Writing $f^{x-1}(G)$ in place of $T(G;x,1)$ we have:

$$f^{x-1}(G) = f^{x-1}(K_n) \cdot \frac{f^{x-1}(K_n \backslash e_1)}{f^{x-1}(K_n)} \cdots \frac{f^{x-1}(G)}{f^{x-1}(G \cup e_l)}, \tag{4.6.5}$$

where the edges $e_1, \ldots, e_l$ are precisely the edges contained in $K_n \backslash G$.

The recurrence relation given above (4.6.1) can be used to calculate the Tutte polynomial of the complete graph at the point $(x,1)$ in polynomial time.

We can again approximate each fraction in 4.6.5 by generating forests in the larger graph according to the non-uniform distribution of $\mathrm{FGEN}_\alpha^{x-1}$ described above, and seeing how many of them are contained in the smaller graph. The algorithm for APPROX_α will do this, if we change all the references to FGEN_α in it to refer to $\mathrm{FGEN}_\alpha^{x-1}$ instead. The proof that this new algorithm works is identical to the proof for the previous counting algorithm.

Hence we have a fpras for the Tutte polynomial of dense graphs at the point $(x,1)$ for any positive integer $x \geq 1$.

We now consider rational $x = 1 + a/b$, where a and b are positive coprime integers.

It has just been shown that the Tutte polynomial of the b-thickening of G can be approximated at the point $(1+a,1)$ and Equation (3.7.8) tells us that

$$T(G^b; 1+a, 1) = bT(G; \frac{a+b}{b}, 1),$$

which completes the proof. □

We now turn our attention to see what we can achieve with graphs that are not dense. The ideas presented earlier for uniformly generating a random forest will not necessarily work in polynomial-time: the algorithm FGEN_α will not produce an output with high enough probability when run on graphs that are sparse.

The best we can achieve for a non-dense graph G, is by forming the n^2-thickening of G. To simplify the notation, we will denote the n^2-thickening of G by $\tilde{G}$.

$\tilde{G}$ is not dense according to our definition, but the algorithm FGEN_α can still be shown to work for a suitable choice of α. We know that FGEN_α produces forests uniformly: we just have to show that, for some constant α, it is more likely than not to produce some output. We calculate a bound on the resistance between v and any other vertex in the associated electrical network. We can assume that the graph is connected: if not, then we can treat each connected component separately and then multiply the results together. Given any $u_i \neq v$, we show that the resistance between v and u_i is small.

Instead of replacing each edge in $\tilde{G}$ with a resistor of 1 Ω, to give n^2 resistors between each connected pair of terminals, we use n resistors in parallel, each of $1/n$ Ω. This will not alter the resistance between any pair of terminals of the network. We can find n edge-disjoint paths from v to u_i, each going along one 1 Ω resistor and up to $(n-1)$ $1/n$ Ω resistors, by going from v to any of its neighbours, then following the shortest path to u_i without visiting v again. We will not need to use any resistor more than once, as there are at least n resistors between every pair of adjacent vertices not including v. So we get n edge-disjoint paths, each of less than 2 Ω resistance, giving a total resistance of less than $2/n$ Ω.

So, setting $\alpha = 1$, we see that the algorithm $\mathrm{FGEN}_\alpha(\tilde{G})$ will produce some output with probability at least $1/2$, when $\tilde{G}$ is the n^2-thickening of any graph G.

From (3.7.8) we know that an evaluation of the Tutte polynomial of a thickened graph can be expressed easily as a multiple of the Tutte polynomial of the original graph, evaluated at a different point. In particular:

$$T(\tilde{G}; 2, 1) = n^2 T(G; 1 + \frac{1}{n^2}, 1). \tag{4.6.6}$$

We have to make some minor modifications to our previous approximation algorithm APPROX_α. However, the same principles apply as before, relating uniform generation to approximate counting. As we can calculate the number of forests in the n^2-thickening of the complete graph (using equations 4.6.1 and 4.6.6), then it is not difficult to construct a fpras that approximates the number of forests of the thickened graph.

Therefore, we can approximate the number of forests of $\tilde{G}$ in polynomial-time. So, we have shown:

Corollary 4.6.7 *There exists a fpras to evaluate the Tutte polynomial of any simple graph G on n vertices at the point $(1 + 1/n^2, 1)$.*

Of course, exact evaluation at this point is #**P**-hard, but, for large n, it is close to the point $(1, 1)$ at which evaluation is easy. Thus the significance of evaluation at this point is limited: however, we have managed here to create a fpras that evaluates the Tutte polynomial of *any* graph at this point as opposed to merely the "almost all" graphs that the set of all dense graphs includes.

4.7 Forests of a fixed size

We now consider the closely related problem of counting the number of forests with a given number of edges. This is just the problem of counting the number of bases of the truncation of the graphic matroid, and of course exact counting (for an arbitrary number of edges) is #**P**-hard since it is #**P**-hard to exactly count the forests of a graph.

However, approximating the number of bases might be much easier, and the following open problem has provoked much interest:

Open Problem 4.7.1 *Does there exist a fpras for counting the number of bases of an arbitrary matroid?*

There is a natural algorithm for generating bases almost uniformly at random (the "natural" random walk on the bases of the matroid), and, in some special cases, it has indeed been shown to work in polynomial time [FM92]. However, the general case still lacks a proof.

Returning our attention to forests, we write $F_i(G)$ for the number of forests of G with i edges. Forests with $n-1$ edges (where n is the number of vertices of the graph) are of course just spanning trees, and so can be counted in polynomial time.

It is easy to show the following:

Lemma 4.7.2 *The number of forests consisting of exactly k edges can be counted in polynomial time, for any constant integer k.*

Proof: We wish to count the number of subsets of k edges that are acyclic. For any graph with $|E|$ edges, there are no more than $|E|^k$ subsets of k edges, and so trying all possibilities will only take a polynomial amount of time. □

The complexity of counting the number of forests with $n-k$ edges is however not so clear. The closely related problem of counting the number of spanning sets of $n-1+k$ edges is equally interesting, and the following restricted variant has recently been posed as an open problem:

Open Problem 4.7.3 (Colbourn [Col93]) *Can the number of spanning connected unicyclic subgraphs of a graph be computed efficiently?*

These subgraphs are "close" to being spanning trees, since they contain a spanning tree plus only one extra edge. It is not unreasonable to hope that

it might be possible to count them precisely, but it is not clear how this could be achieved. However, it has been shown that this number can be computed efficiently for planar graphs [LC83].

As well as being the bases of a particular matroid, the number of these subgraphs is equal to a particular coefficient in an expansion of the reliability polynomial of the graph. This also motivates interest in trying to count them, in order to (hopefully) learn something about the behaviour of the reliability polynomial.

We have not answered this question in a deterministic manner, but instead we consider approximations.

Proposition 4.7.4 *For any fixed integer k, there exists a fpras for the number of forests of a graph with $n-k$ edges, and for the number of spanning connected subgraphs of $n-1+k$ edges, where n is the number of vertices of the graph.*

Proof: We will only explain the method in detail for forests: the case of spanning sets is very similar.

It is clear that the number of forests containing $n-k$ edges satisfies the following self-reducibility relationship:

$$F_{n-k}(G) = F_{n-k}(G') + F_{(n-1)-k}(G'').$$

The important result of Jerrum, Valiant and Vazirani [JVV86] that we mentioned earlier tells us that if we can generate elememts of F_{n-k} uniformly at random in polynomial time, then we can approximately count them in polynomial time. So we just need to show that we can generate them uniformly in polynomial time. We can do this, using the following algorithm.

We order the edges arbitrarily. The algorithm performs the following sequence of operations:

Repeat $n^{2(k-1)}$ times:

1. Pick a spanning tree of G uniformly at random, and call it T.
2. Choose $k-1$ edges of T uniformly at random, and delete them to get a forest F.
3. If the edges that were deleted in step 2 are the lexicographically lowest edges in G (in the ordering) that would connect the components of F into a spanning tree, then output F and terminate.

We need to show two things:

1. The algorithm generates forests uniformly.
2. The algorithm outputs a forest with probability greater than $1/2$.

Uniformity is easy: in order to output a particular forest in F_{n-k}, there is exactly one spanning tree that must be picked in step 1, and exactly one set of $k-1$ edges that have to be deleted in step 2. Hence, in one iteration of the algorithm, each forest in F_{n-k} is output equiprobably. But the probability that the "correct" spanning tree and set of edges was chosen is at least $\frac{1}{n^{2(k-1)}}$, as there are at most $n^{2(k-1)}$ ways of choosing a set of $k-1$ edges to add to F in order to get a spanning tree of G. Hence the probability that no output is produced is bounded by $(1-1/d)^d$ (where $d = n^{2(k-1)}$), which is never greater than $1/2$.

For connected spanning subgraphs with $n-1+k$ edges, the method is similar: we pick a spanning tree of G at random, and add k extra edges uniformly at random. The resulting graph is output if the spanning tree that was originally chosen is the lexicographically first spanning tree contained in the resulting subgraph. Performing this $(n-1+k)^k$ times ensures that we will generate some output with probability greater than $1/2$. □

Bibliography

[Anna] J. D. Annan. An approximation algorithm for counting the number of forests in dense graphs. To appear, *Combinatorics, Probability & Computing.*

[Annb] J. D. Annan. The complexities of the coefficients of the Tutte polynomial. To appear, *Discrete Applied Maths.*

[BC92] J. I. Brown and C. J. Colbourn. Roots of the reliability polynomial. *SIAM J. Disc. Math.*, 5(4):571–585, 1992.

[BDS72] N. L. Biggs, R. M. Damerell, and D. A. Sands. Recursive families of graphs. *Journal of Combinatorial Theory*, 12:123–131, 1972.

[Big74] N. L. Biggs. *Algebraic Graph Theory.* Cambridge University Press, 1974.

[Bro41] R. L. Brooks. On colouring the nodes of a network. *Proc. Cam. Phil. Soc.*, 37:194–197, 1941.

[Bro86] A. Z. Broder. How hard is it to marry at random? (On the approximation of the permanent). *Proceedings of the 18th ACM Symposium on Theory of Computing*, pages 50–58, 1986.

[BRW] F. Brenti, G. F. Royle, and D. G. Wagner. Location of zeros of chromatic polynomials and related polynomials of graphs. To appear, *Canad. J. Math.*

[BSST40] R. L. Brooks, C. A. B. Smith, A. H. Stone, and W. H. Tutte. The dissection of rectangles into squares. *Duke Mathematics Journal*, 7:312–340, 1940.

[Cip87] B. A. Cipra. An introduction to the Ising model. *American Math. Monthly*, 94:937–959, 1987.

[Col93] C. J. Colbourn. Some open problems on reliability polynomials. Technical Report 93–28, DIMACS, 1993.

[Com74] L. Comtet. *Advanced Combinatorics.* D. Reidel, Holland, 1974.

[DFJ93] M. Dyer, A. Frieze, and M. Jerrum. Approximately counting Hamiltonian cycles in dense graphs. Technical Report ECS – LFCS – 93 – 259, Dept. of Computer Science, University of Edinburgh, 1993. Accepted for publication in ACM – SIAM Symposium on Discrete Algorithms.

[DHK93] E. Dahlaus, K. Hajnal, and M. Karpinski. On the parallel complexity of hamiltonian cycle and matching problem on dense graphs. *Journal of Algorithms*, 15:367–384, 1993.

[DS84] P. G. Doyle and J. L. Snell. Random walks and electrical networks. *Carus Mathematical Monographs*, 22, 1984.

[Edw86] K. Edwards. The complexity of colouring problems on dense graphs. *Theoretical Computer Science*, 43:337–343, 1986.

[Fio75] S. Fiorini. On the chromatic index of a graph, III. Uniquely edge-colourable graphs. *Quart. J. Math. Oxford Ser 26*, 3:129–140, 1975.

[FM92] T. Feder and M. Mihail. Balanced matroids. *Proceedings of 24th Annual ACM Syposium on the Theory of Computation*, pages 26–38, 1992.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal of Computing*, 6:675–695, 1977.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman, 1979.

[GJS76] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified **NP**-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[Hol81] I. Holyer. The **NP**-completeness of edge colouring. *SIAM Journal of Computing*, 10:718–720, 1981.

[HW60] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers.* Oxford University Press, 1960.

[JS89] M. R. Jerrum and A. J. Sinclair. Approximating the permanent. *SIAM Journal of Computing*, 18(6):1149–1178, 1989.

[JS90] M. R. Jerrum and A. J. Sinclair. Polynomial-time approximation algorithms for the Ising model. *Proc. 17th ICALP*, pages 462–475, 1990. Extended abstract.

[JVV86] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[JVW90] F. Jaeger, D. L. Vertigan, and D. J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Math. Proc. Camb. Phil. Soc.*, 108:35–53, 1990.

[Kas67] P. W. Kasteleyn. Graph theory and crystal physics. *Graph Theory and Theoretical Physics*, pages 43–110, 1967.

[KL83] R. M. Karp and M. Luby. Monte-Carlo algorithms for enumeration and reliability problems. *Proc. 24th annual IEEE Symposium on the Foundations of Computer Science*, pages 56–64, 1983.

[LC83] C. I. Liu and Y. Chow. Enumeration of connected spanning subgraphs. *Acta Mathematica Hungarica*, 41:27–36, 1983.

[LG83] D. Leven and Z. Galil. **NP**-completeness of finding the chromatic index of regular graphs. *J. Algorithms*, 4:35–44, 1983.

[Lie67] E. H. Lieb. Residual entropy of square ice. *Phys. Rev.*, 162:162–171, 1967.

[Lin86] N. Linial. Hard enumeration problems in geometry and combinatorics. *SIAM J. Alg. Disc. Math.*, 7:331–335, 1986.

[Mah82] S. R. Mahaney. Sparse complete sets for **NP**: solution of a conjecture by Berman and Hartmanis. *Journal of Computer and System Sciences*, 25:130–143, 1982.

[Ona73] R. Onadera. On the number of spanning trees in a complete n-partite graph. *Math. Tensor Quarterly*, 23:142–146, 1973.

[OW79] J. G. Oxley and D. J. A. Welsh. The Tutte polynomial and percolation. In J. A. Bondy and U. R. S. Murty, editors, *Graph Theory and Related Topics*, pages 329–339. Academic Press, London, 1979.

[Oxl92] J. G. Oxley. *Matroid Theory*. Oxford University Press, 1992.

[PP93] T. M. Przytycka and J. H. Przytycki. Subexponentially computable truncations of Jones-type polynomials. *Contemporary Mathematics*, 147:63–109, 1993.

[RR91] R. C. Read and G. F. Royle. Chromatic roots of families of graphs. In *Graph Theory, Combinatorics and Applications.* J. Wiley, New York, 1991.

[RS93] N. Robertson and P. D. Seymour. Graph minors I–XVII. *J. Combinatorial Theory (Series B)*, 1985–1993.

[Sch76] C. P. Schnorr. Optimal algorithms for self-reducible problems. *Proc. 3rd ICALP*, pages 322–337, 1976.

[Sek93] K. Sekine. The flow polynomial and its computation. M. Sc. Thesis, University of Oxford, 1993.

[Sey81] P. D. Seymour. Nowhere-zero 6-flows. *J. Comb. Theory B*, 30:130–135, 1981.

[Sim77] J. Simon. On the difference between one and many. In *Lecture Notes in Computer Science 52*, pages 480–192. Springer-Verlag, 1977.

[Sin93] A. J. Sinclair. *Algorithms for random generation and counting: a Markov chain approach.* Birkhäuser, Boston, 1993.

[Tho78] A. G. Thomason. Hamiltonian cycles and uniquely edge colourable graphs. *Ann. Disc. Math.*, 3:259–268, 1978.

[Tut47] W. T. Tutte. A ring in graph theory. *Proceedings of the Cambridge Philosophical Society*, 43:26–40, 1947.

[Tut54] W. T. Tutte. A contribution to the theory of chromatic polynomials. *Canad. J. Math.*, 6:80–91, 1954.

[Tut67] W. T. Tutte. On dichromatic polynomials. *Journal of Combinatorial Theory*, 2:301–320, 1967.

[Val79] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8:410–421, 1979.

[Ver91] D. L. Vertigan. The computational complexity of tutte invariants for planar graphs. *SIAM Journal of Computing*, 1991. submitted.

[VV86] L. G. Valiant and V. V. Vazirani. **NP** is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[VW92] D. L. Vertigan and D. J. A. Welsh. The computational complexity of the Tutte plane: the bipartite case. *Combinatorics, Probability & Computing*, 1:181–187, 1992.

[Wel93a] D. J. A. Welsh. *Complexity: Knots, Colourings and Counting*, volume 186 of *London Math. Soc. Lecture Notes*. Cambridge University Press, 1993.

[Wel93b] D. J. A. Welsh. Randomised approximation in the Tutte plane. *Combinatorics, Probability & Computing*, 1993. (to appear).

[Wil85] R. J. Wilson. *Introduction to Graph Theory*. Longman, 1985.

[Woo92] D. R. Woodall. A zero-free interval for chromatic polynomials. *Discrete Mathematics*, 101:333–341, 1992.

[Wu82] F. Y. Wu. The Potts model. *Rev. Mod. Phys.*, 54:235–268, 1982.