

A tree-decomposed transfer matrix for computing exact Potts model partition functions for arbitrary graphs, with applications to planar graph colourings

Andrea Bedini^{1,2‡} and Jesper Lykke Jacobsen³

¹ Dipartimento di Fisica, Università degli Studi di Milano, I-20133, Milano, Italy

² INFN, Sezione di Milano, I-20133, Milano, Italy

³ LPTENS, École Normale Supérieure, 24 rue Lhomond, 75231 Paris, France

E-mail: andrea.bedini@mi.infn.it, jesper.jacobsen@ens.fr

Abstract.

Combining tree decomposition and transfer matrix techniques provides a very general algorithm for computing exact partition functions of statistical models defined on arbitrary graphs. The algorithm is particularly efficient in the case of planar graphs. We illustrate it by computing the Potts model partition functions and chromatic polynomials (the number of proper vertex colourings using Q colours) for large samples of random planar graphs with up to $N = 100$ vertices. In the latter case, our algorithm yields a sub-exponential average running time of $\sim \exp(1.516\sqrt{N})$, a substantial improvement over the exponential running time $\sim \exp(0.245N)$ provided by the hitherto best known algorithm. We study the statistics of chromatic roots of random planar graphs in some detail, comparing the findings with results for finite pieces of a regular lattice.

PACS numbers: 02.70.-c, 05.50+q, 89.70.Eg

Keywords: Tree decomposition, transfer matrix, chromatic polynomial, random planar graphs

1. Introduction

A typical problem in statistical physics is to compute the partition function Z of some model with short-ranged interactions between discrete degrees of freedom defined on the vertices of some graph. The partition function is a weighted sum over the states of these degrees of freedom. If the graph is a regular lattice (i.e., consists of a large number M of identical layers) one usually rewrites Z in terms of matrix elements of T^M , where T is a transfer matrix (or imaginary-time evolution operator in an equivalent path integral formulation) that corresponds to the addition of a single layer to the

‡ Present address: MASCOS, Department of Mathematics and Statistics, The University of Melbourne, VIC, 3010, Australia

lattice. This suggests solving the problem by diagonalising T , so that, in some sense, one has substituted algebraic complexity for combinatorial complexity. When the lattice is planar, the exact diagonalisation of T in the limit of an infinitely large system (or thermodynamic limit) can in many cases be achieved by using the powerful tools of integrability [1]. Alternatively, there exists many efficient means of diagonalising T numerically for rather large systems.

In many applications one is however interested in models defined on graphs that are not regular, but incorporate some element of randomness. One must then distinguish whether this randomness is of the annealed or the quenched type. For the annealed disorder, Z is a double sum over the graphs and the vertex degrees of freedom. In the case of planar graphs, this double sum can in many cases be evaluated analytically by random matrix techniques [2]. But arguably, the physically most relevant scenario is that of quenched disorder, where one would typically need to average the free energy $\log Z$, not just Z , over the disorder realisations.

Clearly, a maximal way to deal with quenched randomness would be to evaluate Z independently for each graph in the sample. This is generally not possible analytically, even for planar graphs. The purpose of this paper is to exhibit an efficient algorithm that evaluates Z exactly for an arbitrarily given graph. This algorithm applies to a rather large class of models of statistical physics, as outlined above, and it does not require the graph to be planar. Rather than delving into this generality—we shall briefly come back to the issue in the discussion—we have here chosen to focus on one significant specific application, namely the evaluation of the Potts model [3] partition function $Z_G(Q, v)$ (which, after a change of variables, equals the Tutte polynomial [4] known in graph theory) for a given planar graph G . We are particularly interested in the special case $v = -1$, where $Z_G(Q, v) \equiv \chi_G(Q)$ equals the number of Q -colourings (chromatic polynomial) of G . We now outline the physical motivation for studying this problem, before ending the introduction by giving a brief account on the working principle of our algorithm and its performance (computational complexity).

The Q -colouring problem consists in assigning to each of the vertices of a graph G any one of Q different colours, in such a way that adjacent vertices carry different colours. Each such assignment is known as a proper vertex colouring. The Q -colouring problem arises within physics in studies of frustrated antiferromagnetic spin models, and in spin glass theory in particular [5]. The number of possible colourings (possibly zero) can be shown [6, 7] to be a polynomial in Q , known as the chromatic polynomial $\chi_G(Q)$ of the graph. In particular it makes sense—and is useful—to generalise the original counting problem and to study the properties of $\chi_G(Q)$, considered as a polynomial of a formal variable Q .

The history of the Q -colouring problem is long and interesting, and we refer the reader to [8] for an extensive list of references. The case where G is a *planar* graph has attracted particular interest. A well-known result is then the four-colouring theorem [9] which states that $\chi_G(4) > 0$ for any planar G . Other results exploit the extension

of $\chi_G(Q)$ to non-integer values of Q . One interesting question is whether there exists, in the planar case, some Q_c so that $\chi_G(Q) > 0$ for all $Q \in [Q_c, \infty)$. This statement has been established as a theorem [10] for $Q_c = 5$, and the so-called Birkhoff-Lewis conjecture [10] that this extends to $Q_c = 4$ is widely believed to be true.

The case where G is a *regular* planar lattice has also been intensively studied. In particular, the location and properties of chromatic roots, $\chi_G(Q) = 0$, in the complex Q -plane has been studied for a variety of lattices and boundary conditions (see [11, 8, 12, 13, 14, 15] and references therein). Some of the mechanisms responsible for the generation of real chromatic roots in the region $Q \in [0, 4]$, close to or exactly at the so-called Beraha numbers $B_k = (2 \cos(\pi/k))^2$ with $k \geq 2$ integer, have even been understood analytically [16, 17]. One should also mention that there exists a family of planar graphs—planar triangulations actually—with real chromatic roots converging to $Q = 4$ from below [18], meaning that the value of Q_c in the Birkhoff-Lewis conjecture cannot be lowered further.

One issue which has been left largely unanswered by these studies is the distribution of the location of (real and complex) chromatic roots for ensembles of randomly chosen planar graphs. This situation appears to be the most interesting from the point of view of the physics of disordered frustrated systems. From the mathematical side, it has been proven [19] that chromatic roots are *dense* in the complex plane (except maybe in the disc $|Q-1| < 1$) for a special class of planar graphs (generalised theta graphs). However, this might not have much to do with the physical question of the chromatic roots of a *typical* planar graph. One would also want to know whether typical roots accumulate at the Beraha numbers, and which graph characteristics (connectivity, average coordination number, . . .) might be responsible for the location of roots.

In order to elucidate the amazingly intricate behaviour of chromatic roots in the limit of large graphs, it is clearly desirable to have efficient means of computing $\chi_G(Q)$. Let us recall that in theoretical computer science, an important outstanding question is whether the class P of decision problems that can be *answered* in polynomial time coincides with the class NP of problems for which a proposed answer can be *verified* in polynomial time. NP-complete problems are those to which any problem in NP can be reduced in polynomial time. At present, no polynomial-time algorithm has been found for any of the thousands of known NP-complete problems and it is hence widely believed that $P \neq NP$. Likewise, one can define a counting analogue of NP, denoted by #P, as the class of enumeration problems in which the objects being counted are the possible answers accepted by an NP machine. If an NP problem is often of the form “Are there any solutions that satisfy certain constraints?”, the corresponding #P problems ask “how many” rather than “are there any”. The class #P-complete of the hardest problems in #P is defined likewise. Clearly, #P-complete problems are at least as hard as NP-complete problems. It is known [20] that the counting of proper vertex colourings is a #P-complete problem for $Q = 3$, and the same is then true *a fortiori* for the computation of $\chi_G(Q)$, let alone $Z_G(Q, v)$ [21], for generic values of Q and v .

In practice this means that any algorithm computing $\chi_G(Q)$ can be expected to have a running time that increases exponentially with the number of vertices N . However, lowering the coefficient of the exponent can still make a huge difference for studying the issues outlined above. One central goal of this paper is to make a substantial step in that direction, by lowering the average running time $\sim \exp(0.245N)$ of the previously best known algorithm [22] to $\sim \exp(1.516\sqrt{N})$, as illustrated in Fig. 1 below. The improvement from exponential to sub-exponential asymptotics is not only important from a theoretical point of view. To get a rough idea of what this means in practice, in ten seconds the algorithm [22] will compute $\chi_G(Q)$ for a typical planar graph of $N = 40$ vertices, whereas our algorithm can deal with $N = 100$ in the same time.

Algorithmic progress on #P-complete problems related to graph theory and network design has been made by several, usually widely separated, communities.

On one hand, statistical physicists have shown that the relevant partition functions can be constructed in analogy with the path integral formulation of quantum mechanics. To this end, the configuration of a partially elaborated graph are encoded as suitable quantum states, and the constant-time surface is swept over the graph by means of a time evolution operator known as the transfer matrix. Although rarely stated, this approach is valid not only for regular lattices but also for arbitrary graphs.

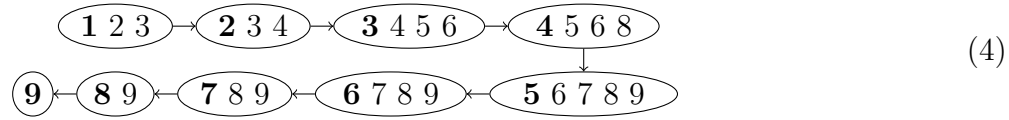
On the other hand, graph theorists have used that graphs can be divided into “weakly interacting” subgraphs through a so-called tree decomposition [23, 24], and solutions obtained for the subgraphs can be recursively combined into a complete solution.

The principle underlying the algorithmic progress to be described in this paper is that tree decomposition and transfer matrix methods can be combined in a very natural way. The main idea is that the tree decomposition is compatible with a recursive generalisation of the time evolution concept. Borrowing ideas from quantum field theory, the combination of partial solutions is obtained by the fusion of suitable state spaces. The resulting algorithm works on any graph, and can readily be adapted to many other problems of statistical physics, by suitable modifications of the state spaces and the fusion procedure.

In particular, we will apply this technique to the problem of computing the chromatic polynomial on planar graphs, obtaining exact solutions, for graphs with $N \simeq 100$ vertices, in only a few seconds.

The layout of the paper is as follows. In section 2 we briefly recall the relation between the Potts model partition function $Z_G(Q, v)$ and the chromatic polynomial $\chi_G(Q)$. In section 3 we present our algorithm and discuss its performance. The results for chromatic roots of planar graphs are given and discussed in section 4. Finally, we give our conclusions in section 5.

one of its neighbours is processed and it stays active until it is processed itself. Taking the vertices in lexicographic order we obtain the following decomposition:



where we wrote in bold face the vertex being processed at each time step.

Each bag has its own set of basis states consisting of the *partitions*§ of the currently active vertices. For instance, in the first time step, the basis states are the five partitions of the three-element set $\{1, 2, 3\}$:

$$\left| \overset{\circ}{1} \ \overset{\circ}{2} \ \overset{\circ}{3} \right\rangle, \quad \left| \overset{\circ}{1} \ \overset{\circ}{2} \ \overset{\circ}{3} \right\rangle, \quad \left| \overset{\circ}{1} \ \overset{\circ}{2} \ \overset{\circ}{3} \right\rangle, \quad \left| \overset{\circ}{1} \ \overset{\circ}{2} \ \overset{\circ}{3} \right\rangle, \quad \left| \overset{\circ}{1} \ \overset{\circ}{2} \ \overset{\circ}{3} \right\rangle.$$

These partitions describe how the active vertices are interconnected through $A \cap E_t$, where $E_t \subseteq E$ is the subset of edges having been processed at time t . A *state* is a linear superposition of basis states.

Processing a vertex consists in processing edges connecting it to unprocessed vertices and then deleting it. Since each edge $e \in E$ may or may not be present in A we process an edge (i, j) by acting on the state with an operator of the form $1 + vJ_{ij}$ where 1 is the identity operator and J_{ij} a *join operator*. A join operator acts on a basis state by amalgamating the blocks containing vertices i and j .

$$J_{ij} \left| \overset{\circ}{i} \ \overset{\circ}{j} \right\rangle = \left| \overset{\circ}{i} \ \overset{\circ}{j} \right\rangle, \quad J_{ij}^2 = J_{ij}. \tag{5}$$

Vertex deletion is defined in terms of a *deletion operator* D_i that removes i from the partition and applies a factor Q (resp. 1) if i was (resp. was not) a singleton.

$$D_i \left| \overset{\circ}{i} \ \overset{\circ}{j} \ \dots \right\rangle = Q \left| \overset{\circ}{j} \ \dots \right\rangle, \tag{6}$$

$$D_i \left| \overset{\circ}{i} \ \overset{\circ}{j} \ \dots \right\rangle = \left| \overset{\circ}{j} \ \dots \right\rangle. \tag{7}$$

For example, in (4), processing the first bag means computing the following composition

$$D_1 (1 + vJ_{12}) (1 + vJ_{13}) \left| \overset{\circ}{1} \ \overset{\circ}{2} \ \overset{\circ}{3} \right\rangle,$$

which gives

$$(Q + 2v) \left| \overset{\circ}{2} \ \overset{\circ}{3} \right\rangle + v^2 \left| \overset{\circ}{2} \ \overset{\circ}{3} \right\rangle,$$

concluding the first time step.

When a new active vertex is encountered it is inserted, as a singleton, in each partition composing the current state. After processing the last bag, the complete partition function (2) is obtained as the coefficient of the empty partition resulting from the deletion of the last active vertex.

At each step, the time and memory requirements are determined by the *bag size* n which is the number of vertices simultaneously active. If the graph is planar, the number

§ Recall that a partition of a finite set S is a collection of nonempty, pairwise disjoint subsets of S whose union is S . The subsets are called *blocks*.

of partitions to be considered is at most the Catalan number C_n , i.e., the number of non-crossing partitions. The corresponding generating function is

$$C(z) = \sum_{n=0}^{\infty} C_n z^n = \frac{1 - \sqrt{1 - 4z}}{2z}. \quad (8)$$

If the graph is not planar, the number of partitions is at most the Bell number B_n , with generating function

$$B(z) = \sum_{n=0}^{\infty} B_n \frac{z^n}{n!} = \exp(e^z - 1). \quad (9)$$

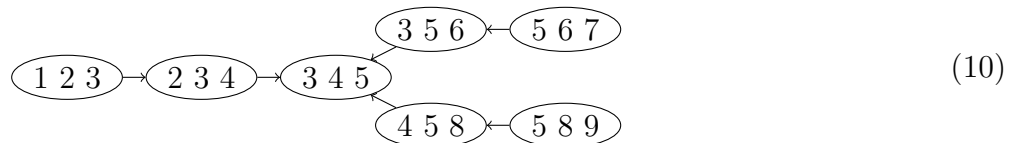
We have $C_n = 4^n n^{-3/2} \pi^{-1/2} [1 + O(1/n)]$, whereas B_n grows super-exponentially.

3.2. Tree decomposition

It turns out that the decomposition (4) of G is a special case of a more general construction. By definition, a *tree decomposition* [23, 24] of a graph $G = (V, E)$ is a collection of bags, organised as a tree (a connected graph with no cycles), and satisfying the following requirements:

- (i) For each $i \in V$, there exists a bag containing i ;
- (ii) For each $(ij) \in E$, there exists a bag containing both i and j ;
- (iii) For any $i \in V$, the set of bags containing i is connected in the tree.

The previous decomposition (4) is just a special case of a tree decomposition (a *path decomposition*). As an example of the general construction, applied to (3), consider



where the arrows form the unique path that connects each bag to the central one (the *root* of the tree).

The advantage of working with tree instead of path decompositions relies on the fact that in the former case a decomposition with smaller bags can be obtained (the latter being just a special case). Therefore, the number of states one has to keep track of is exponentially smaller, and the gain is significant.

The transfer matrix approach can be adapted naturally to this new general setting: Properties (i)–(ii) guarantee that each edge and vertex are processed within a definite bag. Property (iii) implies that each vertex has a definite life time in the recursion, its insertion and deletion being separated by the processing of all edges incident on it.

| Note that the restriction to non-crossing partitions holds true even if the ordering of the vertices, such as in (4), does not respect the planar embedding (3) of the whole graph (which will in general be unknown). Because we choose to store partitions independently of their crossing property, it has not been relevant to be able to explicitly keep track of the planar embedding. All that matters is that the number of partitions does not exceed C_n , and this is true since such a non-crossing representation could be obtained “by hand” by inspecting the planar embedding at each stage of the transfer process.

In this new version the algorithm starts from the root of the tree, which can be chosen arbitrarily, and runs through the tree recursively. Alternatively, this can be thought of as starting from the leaves and building up the tree inductively.

Consider first the case of a parent bag P with only one daughter bag D . Going up from D to P in the recursive step involves deleting vertices $D \setminus P$, inserting vertices $P \setminus D$ and finally processing edges in P . A tree decomposition does not specify when an edge $e = (ij)$ must be processed. A simple recipe would be to process e as soon as one encounters a bag containing both i and j . However we note that this freedom of choice can be exploited to optimise the algorithm (see below).

3.3. Fusion

We now discuss the case when a parent bag P has several daughters D_ℓ with $\ell = 1, 2, \dots, d$. In this case, vertex deletions and insertions are followed by a special *fusion* procedure. Suppose first $d = 2$, and let \mathcal{P}_1 be a partition of $D_1 \cap P$ with weight w_1 , and \mathcal{P}_2 a partition of $D_2 \cap P$ with weight w_2 . The fused state is then

$$\mathcal{P}_1 \otimes \mathcal{P}_2 = \mathcal{P}_1 \vee \mathcal{P}_2,$$

and it occurs with weight $w_1 w_2$. Here \vee denotes the join operation in the partition lattice.¶ For definiteness, let us explain how this can be computed in practice. First choose some set E_1 so that

$$\mathcal{P}_1 = \left(\prod_{e \in E_1} J_e \right) \mathcal{S}_1, \quad (11)$$

where \mathcal{S}_1 is the all-singleton partition of $D_1 \cap P$. Note that it is not necessary to choose E_1 as a subset of E , although this can always be done. Since $J_e J_{e'} = J_{e'} J_e$ the order in the above product is irrelevant. Similarly choose E_2 for \mathcal{P}_2 . The fused state can then be constructed explicitly as

$$\mathcal{P}_1 \otimes \mathcal{P}_2 = \left(\prod_{e \in E_1 \cup E_2} J_e \right) \mathcal{S}_{12},$$

where \mathcal{S}_{12} is the all-singleton partition of $(D_1 \cup D_2) \cap P$. For $d > 2$ daughters, the complete fusion can be accomplished by fusing D_1 with D_2 , then fusing the result with D_3 , and so on.

¶ We recall that a partial order of the partitions follows from defining a partition \mathcal{P}_a to be a *refinement* of \mathcal{P}_b , and we write $\mathcal{P}_a \preceq \mathcal{P}_b$, provided that each block in \mathcal{P}_a is a subset of some block in \mathcal{P}_b . With this partial order, the set of partitions form a *lattice*, in the sense that any two partitions \mathcal{P}_1 and \mathcal{P}_2 possess a largest lower bound called *meet* and denoted $\mathcal{P}_1 \wedge \mathcal{P}_2$ and a smallest upper bound called *join* and denoted $\mathcal{P}_1 \vee \mathcal{P}_2$. The meet $\mathcal{P}_1 \wedge \mathcal{P}_2$ has as blocks all nonempty intersections of a block from \mathcal{P}_1 with a block from \mathcal{P}_2 . The blocks of the join $\mathcal{P}_1 \vee \mathcal{P}_2$ are the smallest subsets which are exactly a union of blocks from both \mathcal{P}_1 and \mathcal{P}_2 . In the partition lattice, the bottom (or finest) element is the unique partition in which each block has size 1 (the all-singleton partition), and the top (or coarsest) element is the unique partition in which there is a single block (the all-connected partition).

Let us illustrate the fusion procedure for G with the tree decomposition (10). After processing the two left-most bags and deleting vertex 2, the propagating state is

$$\omega_1 | \begin{smallmatrix} \circ & \circ \\ 3 & 4 \end{smallmatrix} \rangle + \omega_2 | \begin{smallmatrix} \circ & \circ \\ 3 & 4 \end{smallmatrix} \rangle = (\omega_1 + \omega_2 \mathbf{J}_{34}) | \begin{smallmatrix} \circ & \circ \\ 3 & 4 \end{smallmatrix} \rangle, \quad (12)$$

where $\omega_1 = Q^2 + 3v(Q + v)$ and $\omega_2 = Q^2v + 3Qv^2 + 4v^3 + v^4$. By symmetry, the same result is obtained for the two top right bags (replacing 4 by 5) and for the two bottom right bags (replacing 3 by 5).⁺ The fused state arriving in the central bag is then

$$\begin{aligned} & (\omega_1 + \omega_2 \mathbf{J}_{34})(\omega_1 + \omega_2 \mathbf{J}_{35})(\omega_1 + \omega_2 \mathbf{J}_{45}) | \begin{smallmatrix} \circ & \circ & \circ \\ 3 & 4 & 5 \end{smallmatrix} \rangle \\ & = \omega_1^3 | \begin{smallmatrix} \circ & \circ & \circ \\ 3 & 4 & 5 \end{smallmatrix} \rangle + (3\omega_1\omega_2^2 + \omega_2^3) | \begin{smallmatrix} \circ & \circ & \circ \\ 3 & 4 & 5 \end{smallmatrix} \rangle \\ & + \omega_1^2\omega_2 (| \begin{smallmatrix} \circ & \circ & \circ \\ 3 & 4 & 5 \end{smallmatrix} \rangle + | \begin{smallmatrix} \circ & \circ & \circ \\ 3 & 4 & 5 \end{smallmatrix} \rangle + | \begin{smallmatrix} \circ & \circ & \circ \\ 3 & 4 & 5 \end{smallmatrix} \rangle), \end{aligned} \quad (13)$$

from which the result $Z_G(Q, v)$ follows upon deleting vertices 3,4,5.

3.4. Pruning

Problem specific features can be exploited to reduce further the number of basis states to be considered. As an example of this, note that in the colouring case ($v = -1$) the operator $\mathbf{O}_{ij} = 1 + v\mathbf{J}_{ij}$ associated with an edge (ij) is a projector, $\mathbf{O}_{ij}^2 = \mathbf{O}_{ij}$, and it annihilates the subspace of partitions where i and j are connected. It follows that one can discard basis states in which two vertices are connected, as soon as one discovers that an edge between them is going to be processed later within the same bag or within the parent bag. Especially before fusions this simple trick reduces substantially the number of basis states and thus leads to a big speed up.

3.5. Performance

For a planar graph, the state of a bag of size n is spanned by C_n basis states. (For a non-planar graph, replace C_n by B_n .) The memory needed by the algorithm is therefore proportional to $C_{n_{\max}}$, where n_{\max} is the size of the largest bag. The time needed to process one edge in a bag of size n is proportional to C_n .

However, most of the time is spent fusing states. For a parent P with d daughters D_ℓ , the number of basis state pairs to be fused is

$$\sum_{\ell=1}^d C_{|\mathcal{D}_{\ell-1} \cap P|} C_{|D_\ell \cap P|}, \quad (14)$$

where we have set $\mathcal{D}_k = \cup_{\ell=1}^k D_\ell$. Each of these elementary fusions (i.e., the joins $\mathcal{P}_1 \vee \mathcal{P}_2$) can be done in time linear* in the number of participating vertices. Note that we can choose the order of successive fusions so as to minimise the quantity (14).

⁺ Needless to say, the fusion procedure will work also in cases where the graph possesses no such symmetries. In our example, we have chosen G as a rather symmetric graph in order to keep the actual computations as simple as possible.

* We represent partitions as lists of numbers linking each participating vertex to the block it belongs to. The set E_1 in (11) can be obtained in a single scan of this list where one keeps track of the last

It is therefore essential for the algorithm that one knows how to obtain a good tree decomposition. The minimum of $n_{\max} - 1$ over all tree decompositions is known as the tree width k , but obtaining this is another NP-hard problem. However, the simple algorithm **GreedyFillIn** [27] gives an upper bound k_0 on k and a tree decomposition of width k_0 in time *linear* in the number of vertices N . For uniformly generated planar graphs we find that for $N = 40$ —a value enabling comparison with algorithms that determine k exactly—that $k_0 = k$ nearly always ($k_0 = k + 1$ with probability $\simeq 10^{-3}$). When k_0 is small enough that all the partitions can fit into the computer’s memory, the algorithm has proven to be very fast with execution time in the order of seconds.

To summarise, suppose that we wish to compute $Z_G(Q, v)$ or $\chi_G(Q)$ for a planar graph of N vertices and M edges, and that we are given a tree decomposition of B bags, with n_{\max} being the size of the largest bag. The number of (binary) fusions to be performed is $F = \sum_i (d_i - 1)$, where d_i is the number of daughters of bag i . For simplicity we assume a computational model where the operations on the weights can be done in unit time.‡ For the version of the algorithm without pruning, we have therefore shown that the worst-case running time is asymptotically proportional to

$$(N + M + B)C_{n_{\max}} n_{\max} + FC_{n_{\max}}^2 n_{\max}. \quad (15)$$

Note that this is linear in N and M for a fixed n_{\max} (cf. [28]).

We choose to test our algorithm against the one presented by Haggard et al. in [22]. We first generated a uniform sample of 100 planar graphs for each size $N = |V|$ between 20 and 100 using Fusy’s algorithm [31]. We then ran four different algorithms over this sample: the algorithm of [22], our first path-based transfer matrix algorithm, the new tree-based version algorithm and a tree-based version using the above pruning optimisation. Path decompositions were obtained with a variant of the **GreedyFillIn** algorithm in which the resulting tree decomposition was forced not to branch. Average running times are presented in Fig. 1, in the form of effective exponential fits.

While these fits clearly confirm the efficiency of the tree decomposed approach, they are actually misleading and hide an important fact. Namely, the tree width of a planar graph is bounded by $\alpha\sqrt{N}$, as follows from the famous planar separator theorem [29]. The currently best upper bound on the constant is $\alpha < 3.182$ [30]. These facts—combined with (15) and (8), and the heuristic observation of the near-ideality of the tree decompositions obtained by the **GreedyFillIn** heuristics—immediately imply an upper bound on the worst-case running time on the tree-based algorithms of $16^{3.182\sqrt{N}} = \exp(8.822\sqrt{N})$. It follows that the mean running time of these algorithms must be of the form $\exp(\beta\sqrt{N})$, with the constants β satisfying $\beta_{\text{tree+pruning}} \leq \beta_{\text{tree}}$. The fits for β , using the same data as in Fig. 1, are shown in Fig. 2.

seen vertex for each block. In this representation, applying J_e is constant time, apart from a possible linear-time procedure to bring the resulting list in a canonical form. This last procedure can be safely postponed after all the join operations, making the whole elementary fusion linear in the number of participating vertices. [Note that in [28] the join operation is claimed to be *quadratic* in n .]

‡ The weights are numbers for an evaluation of $Z_G(Q, v)$ or of $\chi_G(Q)$, polynomials with integer coefficients for $\chi_G(Q)$, and polynomials in two variables with integer coefficients for $Z_G(Q, v)$.

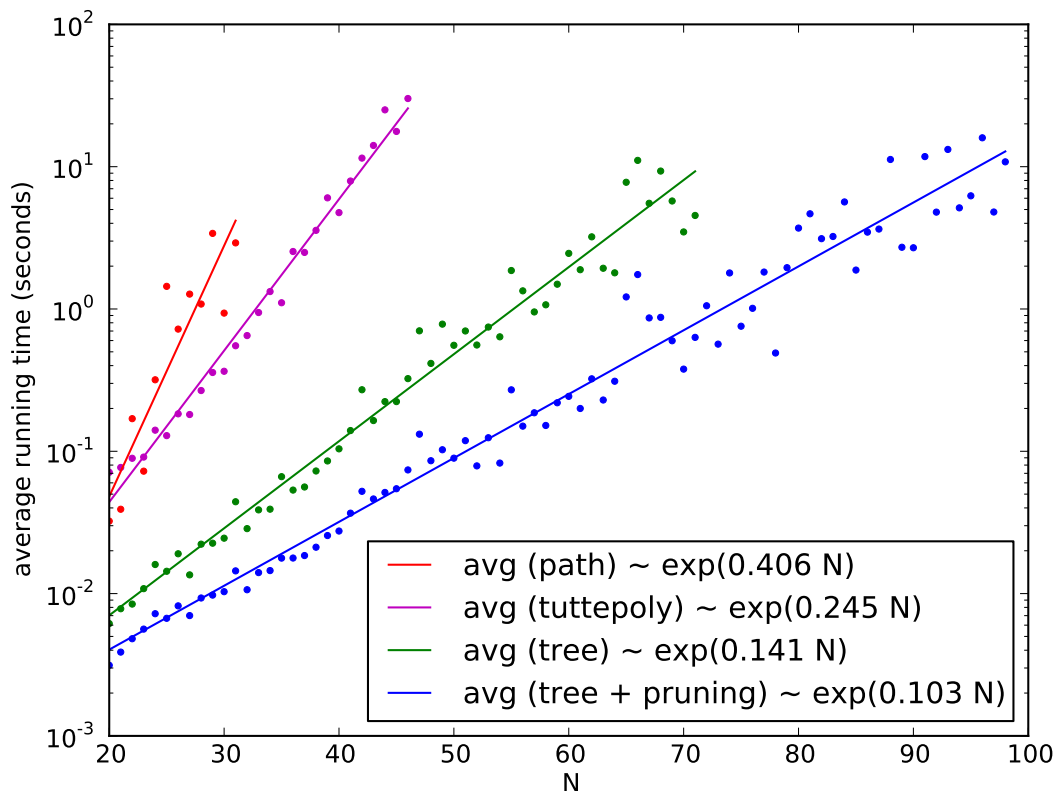


Figure 1. Average running time in seconds on a random planar graph of size N vertices. Each point is averaged over 100 graphs.

3.6. Technical aspects

For completeness we give a few implementational details. Partitions relevant to the propagating state are kept in a hash table for fast (amortised constant time) access. The corresponding weights are polynomials in Q whose coefficients rapidly exceed the machine integer size $M = 2^{32}$ when running on big graphs. To solve this issue without requiring additional memory we used modular arithmetics: the algorithm was run many times with coefficients computed modulo primes $p < M$, and the original coefficients were reconstructed by the Chinese remainder theorem.

4. Results

As a simple application of our algorithm we obtained the distribution of the chromatic roots $\{Q \mid \chi_G(Q) = 0\}$ for ensembles of planar graphs of up to $N = 100$ vertices. This problem is interesting both in statistical mechanics and in graph theory [4, 16, 17, 8]. As known from Lee-Yang theory, the roots signal phase transitions.

Before turning to the actual results, it is useful to give some results for *regular* planar graphs of a few hundred vertices. To this end, we have used a particular

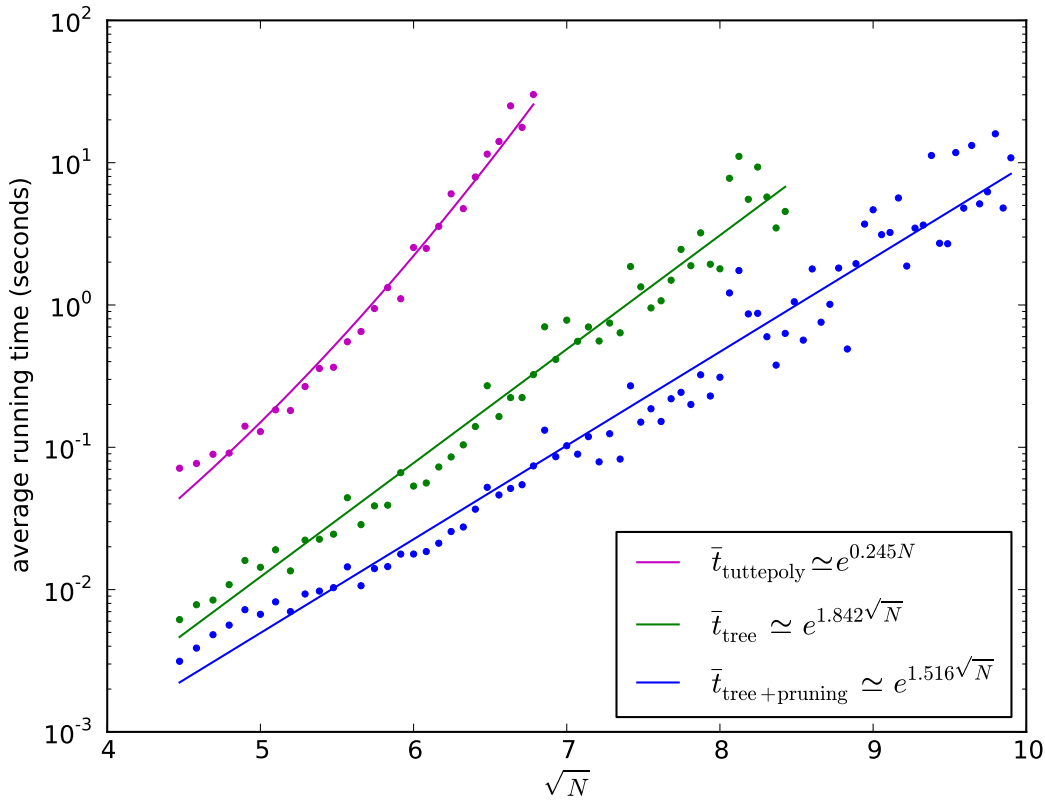


Figure 2. Average running time in seconds on a random planar graph of size N vertices, now fitted to the form $\exp(\beta\sqrt{N})$. The data points are identical to those of Fig. 1.

adaptation of the traditional (path-decomposed) transfer matrix algorithm that takes special advantage of the regular lattice structure [32] to compute $\chi_G(Q)$ for $L \times L$ sections of the square and triangular lattice (the latter being considered as a square lattice with added diagonals). The boundary conditions are free-free, in the terminology of [8]. The chromatic polynomials for $L = 10, 12$ have been checked against [12], while those for $L = 14, 16, 18$ are new.

The resulting locations of the chromatic roots in the complex Q -plane are shown in Fig. 3 for the square lattice, and in Fig. 4 for the triangular lattice. As in earlier work [8, 12, 13] we notice that most of the roots fall on connected curves which, very roughly speaking, tend to form a egg-shaped figure with a pair of prongs on the right side. Without going into details (which are numerous, see [33, 11, 16, 17, 8, 12, 13, 34] for more exact statements) in the limit $L \rightarrow \infty$ the egg-shaped part is expected to enclose a segment $Q \in [0, Q_0]$ of the real axis, with $Q_0 = 3$ for the square lattice, and $Q_0 \approx 3.81967$ for the triangular lattice. In the interior of the egg one observes additional isolated real zeros right at, or extremely close to, the so-called Beraha numbers

$$B_k = (2 \cos(\pi/k))^2 \quad (k = 2, 3, 4, \dots) \quad (16)$$

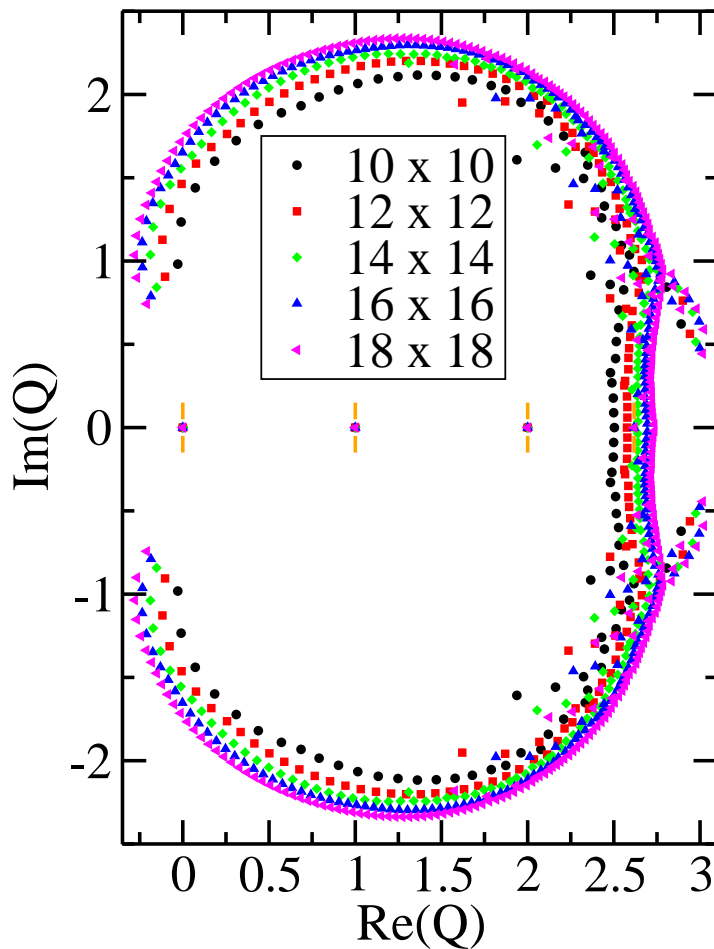


Figure 3. Chromatic roots for $L \times L$ sections of the square lattice with free boundary conditions, and $L = 10, 12, 14, 16, 18$. Beraha numbers B_k with $k = 2, 3, 4, 5$ are shown as small vertical line segments.

The reason why this is so is linked to particularities of the representation theory of the quantum algebra that underlies the two-dimensional Potts model [16, 17].

The existence of a finite-size equivalent of Q_0 is clearly visible from Figs. 3–4. One could propose defining $Q_0(L)$ as the largest real root, but one should be careful since in some cases the egg-shaped curve does not possess a root right on the real axis. Barring these (and other) difficulties, one could speculate that if the result for an arbitrary planar graph was qualitatively similar to that of these regular lattices, the probability distribution of complex roots would look like some broadened-out egg. In addition, the distribution of real roots would be a superposition of sharp peaks centered at the Beraha numbers, and a broad background distribution corresponding to the $Q_0(L)$.

This is indeed more-or-less what we observe. For a sample of 2500 uniformly drawn [31] planar graphs of $N = 100$ vertices we show the distributions of complex chromatic roots in Fig. 5, and the distribution of the subset of real roots in Fig. 6. Regarding the complex roots, we note that although chromatic roots of planar graphs have been

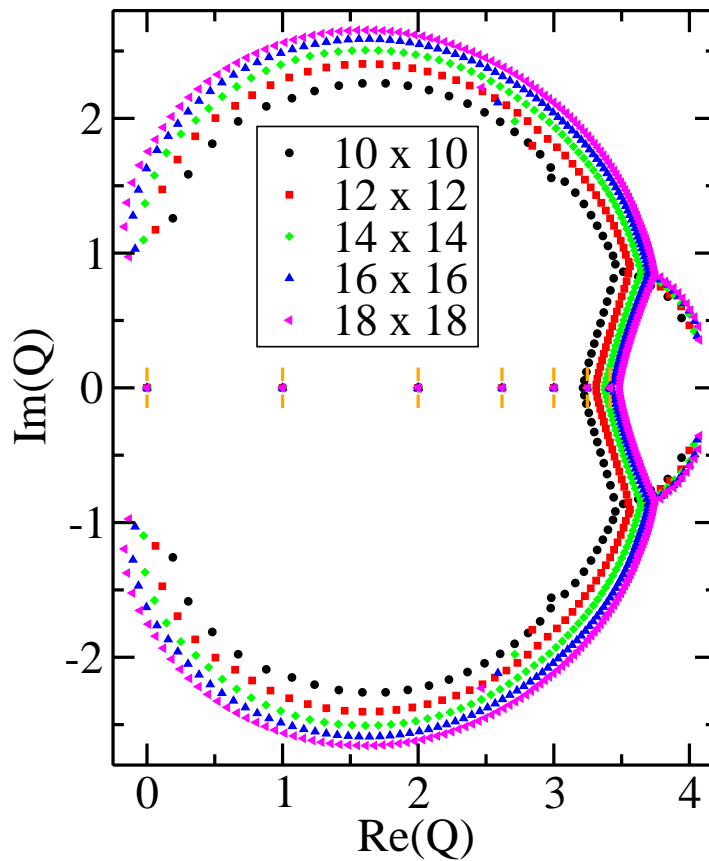


Figure 4. Chromatic roots for $L \times L$ sections of the triangular lattice with free boundary conditions, and $L = 10, 12, 14, 16, 18$. Beraha numbers B_k with $k = 2, 3, 4, 5, 6, 7, 8$ are shown as small vertical line segments.

shown to be dense in the complex plane [19] (except maybe in the disc $|Q - 1| < 1$), typical roots are clearly quite close to the origin.

As to the real chromatic roots, the absence of roots on the negative real axis, and the intervals $(0, 1)$ and $(1, 32/27]$ follow from a theorem [35] (see also [36]). The roots found here respect this theorem as well as the Birkhoff-Lewis conjecture [10]. Apart from that, Fig. 6 shows as expected a superposition of a broad background distribution and sharp peaks centered at $Q = B_k$ with $k = 2, 3, 4, 5, 6$ (we have $B_5 \simeq 2.61803$). It seems likely that for $N \rightarrow \infty$ this background will extend to the interval $(32/27, 4)$ and peaks will occur at all Beraha numbers. We also expect that the maximum of the background distribution may be shifted further to the right by requiring the graphs to be two- or three-connected, and so presumably the peaks at the first few B_k should stand out more clearly. We plan to investigate this and further issues in more detail elsewhere [37].

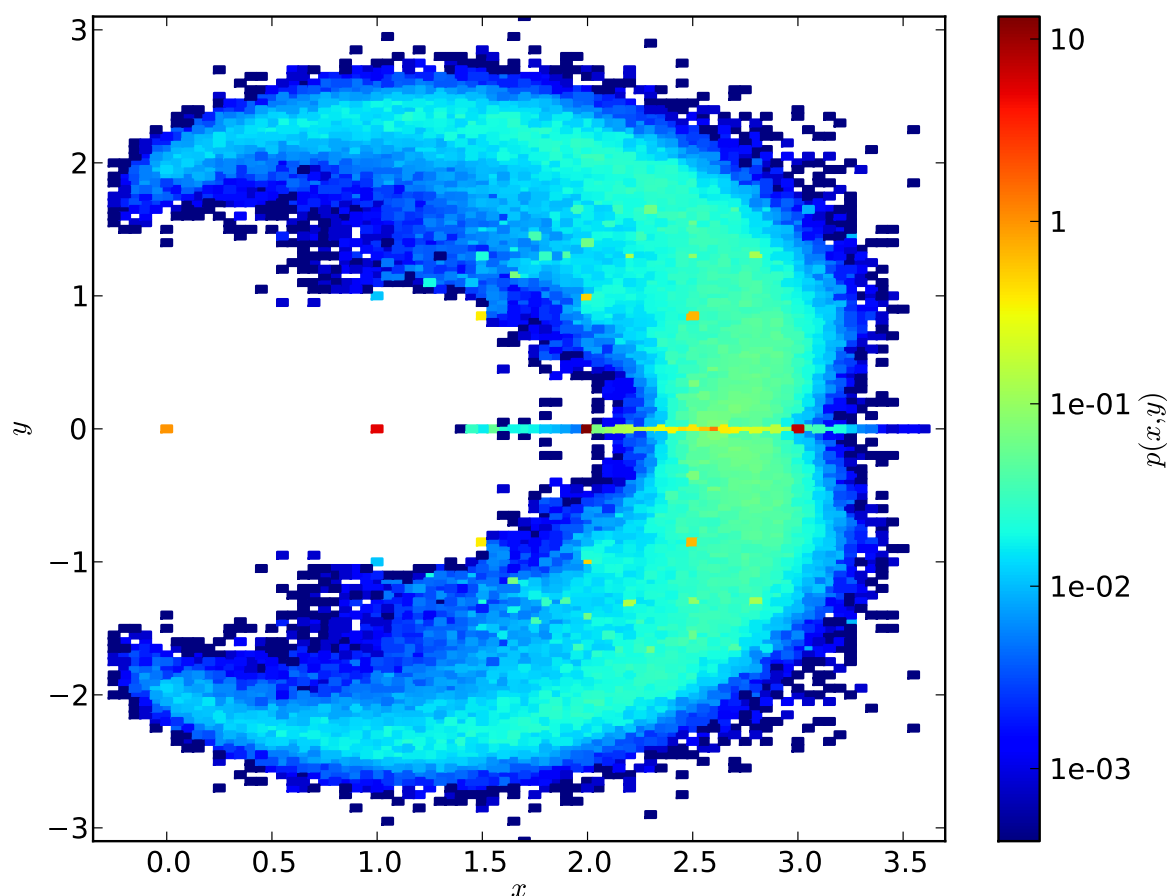


Figure 5. Distribution for complex chromatic roots $Q = x + iy$ obtained from a sample of 2500 planar graphs of size 100. The normalisation is such that $p(x, y)$ is the expected number of roots in $[x - 0.05, x + 0.05] \times [y - 0.05, y + 0.05]$ counted with their multiplicity.

5. Conclusion

In this paper we have shown how to obtain an efficient algorithm for solving instances of graph-related #P-complete problems. The case of the Potts partition function $Z_G(Q, v)$, and in particular its specialisation the chromatic polynomial $\chi_G(Q) = Z_G(Q, -1)$, were studied in detail, and some results on the distribution of real and complex chromatic roots for uniformly drawn random planar graphs were given.

Our algorithmic approach has some overlap with an algorithm described—but apparently not implemented—by Noble [28]. This author also combines partial evaluations of $Z_G(Q, v)$ —treated by him in the Tutte polynomial formulation [4]—in the basis of partitions with the notion of tree decomposition. There are however a number of important differences. First, our use of transfer matrices—and specifically of its factorisation within each bag, i.e., our choice to process a single edge at a time within each bag—leads to a better time complexity. In particular, treating a daughter bag with no sisters has worst-case running time $C_{n_{\max}}^2 n_{\max}^2 + C_{n_{\max}} 2^{n_{\max}(n_{\max}-1)/2}$ in Noble’s

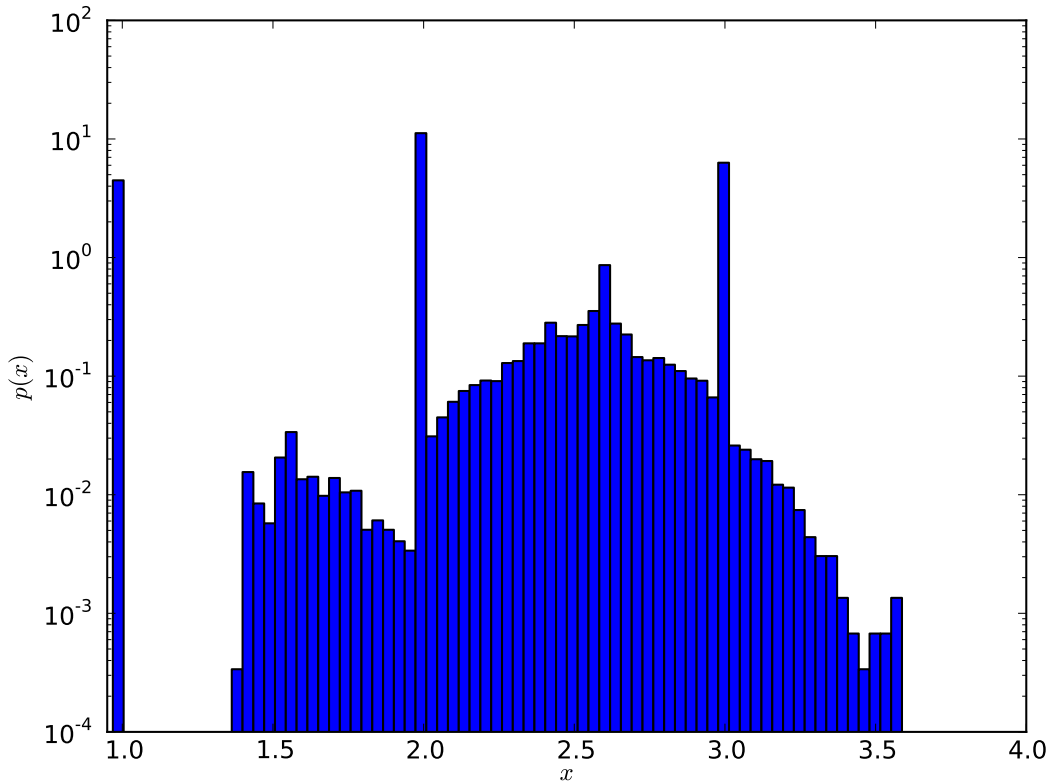


Figure 6. Probability distribution for the subset of chromatic roots in Fig. 5 that lie on the real axis. Normalised to unit total area.

approach versus $C_{n_{\max}} n_{\max}$ in ours. Second, we treat binary fusions more efficiently, exploiting the fact that a given pair $\mathcal{P}_1, \mathcal{P}_2$ of input partitions gives rise to a *unique* output partition $\mathcal{P}_1 \vee \mathcal{P}_2$: this reduces the time complexity for binary fusions from worst-case time $C_{n_{\max}}^3 n_{\max}^2$ in [28] to $C_{n_{\max}}^2 n_{\max}$ in our approach. Third, our algorithm appears easier to describe and implement—as we have indeed done—and it avoids having to deal with the $Q \rightarrow 0$ limit ($x = 1$ in the notations of [28]) as a particular case.

It is thus clear that what matters the most is to obtain a good tree decomposition, and in particular the running time is essentially determined by n_{\max} . In Fig. 7 we show the distribution of n_{\max} obtained from applying the algorithm `GreedyFillIn` to random planar graphs as a function of N .

It is plainly visible from Fig. 7 that both the mean and worst-case n_{\max} exhibit a slower than linear growth with N . Assuming that `GreedyFillIn` yields always an n_{\max} which is of the order of the true tree width, we would have $n_{\max} \leq \text{cst} \sqrt{N}$, and the data of Fig. 7 appear to be compatible with such a scaling. We defer the further investigation of the asymptotic behaviour of `GreedyFillIn` to future work [37].

We should also mention that the use of tree decomposition in the context of *decision* problems has previously been advocated by a number of authors (see [38] and references

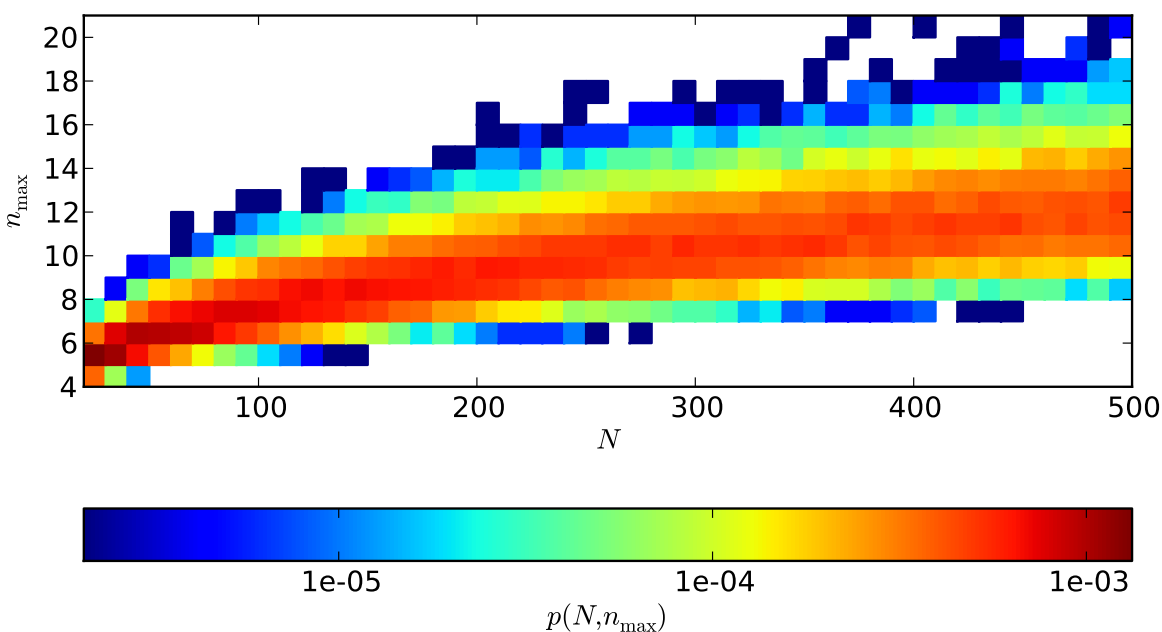


Figure 7. Probability distribution $p(N, n_{\max})$ of obtaining a tree decomposition of maximum bag size n_{\max} (i.e., width $n_{\max} - 1$) when applying the `GreedyFillIn` algorithm to random planar graphs, as a function of the graph size N . The data is computed over a sample of 100 planar graphs for each size between 20 and 500, binned in blocks of 10 on the x axis and normalised to unit total area.

therein). In these works, the solution is found by doing dynamic programming on the tree-decomposed graph.

Let us conclude by commenting on the generality of our algorithm. Going through the list of known NP-complete problems related to graphs and network design, one easily realises that most of them can be promoted to counting problems (which cannot be easier), and a counting polynomial analogous to $\chi_G(Q)$ can be defined. Adapting our algorithm to those cases usually requires just a problem-specific definition of the states, of the single-edge transfer process, and of the fusion procedure, whereas the bulk of the method can be taken over without changes.

It is worth underlining that our ability to solve efficiently an enumeration problem can be exploited to obtain a specific instance that solves the corresponding decision problem with only an additional linear time factor.

In particular, we have adapted (but not yet implemented) our algorithm to counting versions of the following problems: Hamiltonian walks and cycles, longest path, vertex cover, dominating set, feedback vertex set, and minimum maximal independent set. We plan to report on these problems elsewhere [37].

Acknowledgments

This work was supported by the European Community Network ENRAGE (grant MRTN-CT-2004-005616) and by the Agence Nationale de la Recherche (grant ANR-06-BLAN-0124-03). The authors thank A.D. Sokal for discussions and for commenting on an earlier version of the manuscript.

References

- [1] R.J. Baxter, *Exactly solved models in statistical mechanics* (Academic Press, London, 1982).
- [2] M.L. Mehta, *Random matrices*, 3rd edition (Elsevier, Amsterdam, 2004).
- [3] R.B. Potts, Proc. Camb. Phil. Soc. **48**, 106 (1952).
- [4] W.T. Tutte, Canad. J. Math. **6**, 80–91 (1954).
- [5] R. Mulet, A. Pagnani, M. Weight and R. Zecchina, Phys. Rev. Lett. **89**, 268701 (2002).
- [6] G.D. Birkhoff, Ann. Math. **14**, 42 (1912).
- [7] H. Whitney, Bull. Amer. Math. Soc. **38**, 572 (1932).
- [8] J. Salas and A.D. Sokal, J. Stat. Phys. **104**, 609–699 (2001).
- [9] K. Appel and W. Haken, Bull. Amer. Math. Soc. **82**, 711 (1976).
- [10] G.D. Birkhoff and D.C. Lewis, Trans. Amer. Math. Soc. **60**, 355 (1946).
- [11] R.J. Baxter, J. Phys. A **20**, 5241 (1987).
- [12] J.L. Jacobsen and J. Salas, J. Stat. Phys. **104**, 701 (2001).
- [13] J.L. Jacobsen, J. Salas and A.D. Sokal, J. Stat. Phys. **112**, 921 (2003).
- [14] J.L. Jacobsen and J. Salas, J. Stat. Phys. **122**, 705 (2006).
- [15] J.L. Jacobsen and J. Salas, Nucl. Phys. B **783**, 238 (2007).
- [16] H. Saleur, Comm. Math. Phys. **132**, 657 (1990).
- [17] H. Saleur, Nucl. Phys. B **360**, 219 (1991).
- [18] G.F. Royle, Annals Combin. **12**, 195–210 (2008).
- [19] A.D. Sokal, Combin. Probab. Comput. **13**, 221–261 (2004).
- [20] D.J.A. Welsh, *The computational complexity of some classical problems from statistical physics*, in G.R. Grimmett and D.J.A. Welsh (eds.), *Disorder in physical systems*, Clarendon Press, Oxford, 1990, pp. 307–321.
- [21] F. Jaeger, D. Vertigan and D. Welsh, Math. Proc. Camb. Phil. Soc. **108**, 35–53 (1990).
- [22] G. Haggard, D.J. Pearce and G. Royle, *Computing Tutte polynomials*. Technical report NI09024-CSM, Isaac Newton Institute for Mathematical Sciences, 2009.
- [23] N. Robertson and P.D. Seymour, J. Comb. Theory B **36**, 49–64 (1984).
- [24] H.L. Bodlaender, Theor. Comp. Science **209**, 1–45 (1998).
- [25] C.M. Fortuin and P.W. Kasteleyn, Physica **57**, 536–564 (1972).
- [26] H.W.J. Blöte and M.P. Nightingale, Physica A **112**, 405 (1982).
- [27] H.L. Bodlaender and A.M.C.A. Koster, Inform. Comput. **208**, 259–275 (2010).
- [28] S.D. Noble, Combin. Probab. Comput. **7**, 307–321 (1998).
- [29] N. Alon, P. Seymour and R. Thomas, J. Am. Math. Soc. **3**, 801–808 (1990).
- [30] F.V. Fomin and D.M. Thilikos, J. Graph Theory **51**, 53–81 (2006).
- [31] E. Fusy, Random Struct. Alg. **35**, 464–522 (2009).
- [32] J.L. Jacobsen, *Bulk, surface and corner free energy series for the chromatic polynomial on the square and triangular lattices*, arXiv:1005.3609.
- [33] R.J. Baxter, J. Phys. A **19**, 2821 (1986).
- [34] R.J. Baxter, Proc. Roy. Soc. London A **383**, 43 (1982).
- [35] B. Jackson, Combin. Probab. Comput. **2**, 325–336 (1993).
- [36] C. Thomassen, Combin. Probab. Comput. **6**, 497–506 (1997).
- [37] A. Bedini and J.L. Jacobsen, in preparation.

- [38] F.V. Fomin and D.M. Thilikos, *Lecture Notes in Comput. Sci.* **2996**, 56–67 (2004).