Combinatorics, Probability and Computing (2005) 14, 737–754. © 2005 Cambridge University Press
doi:10.1017/S0963548305007005 Printed in the United Kingdom

Approximate Counting and Quantum Computation

M. BORDEWICH,[1†] M. FREEDMAN,[2] L. LOVÁSZ[2] and D. WELSH[1‡]

[1]Mathematical Institute, University of Oxford, 24-29 St. Giles', Oxford, OX1
(e-mail: magnusb@comp.leeds.ac.uk, dwelsh@maths.ox.ac.uk)

[2]Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
(e-mail: michaelf@microsoft.com, lovasz@microsoft.com)

Received 15 October 2003; revised 7 February 2005

For Béla Bollobás on his 60th birthday

Motivated by the result that an 'approximate' evaluation of the Jones polynomial of a braid at a 5th root of unity can be used to simulate the quantum part of any algorithm in the quantum complexity class BQP, and results relating BQP to the counting class GapP, we introduce a form of additive approximation which can be used to simulate a function in BQP. We show that all functions in the classes #P and GapP have such an approximation scheme under certain natural normalizations. However, we are unable to determine whether the particular functions we are motivated by, such as the above evaluation of the Jones polynomial, can be approximated in this way. We close with some open problems motivated by this work.

1. Introduction

The quantum complexity class BQP consists of those decision problems that can be computed with bounded error, using quantum resources, in polynomial time. Relative to the polynomial hierarchy of classical computation, it is known that

$$\mathrm{BPP} \subseteq \mathrm{BQP} \subseteq \mathrm{PP} \subseteq \mathrm{PSPACE},$$

and at the moment none of these inclusions is known to be proper [1]. Recent work by Freedman, Kitaev, Larson and Wang [5] has shown that the 'quantum part' of any quantum computation can be replaced by an approximate evaluation of the Jones polynomial of a related braid. A classical polynomial time algorithm can convert a quantum circuit for an instance of such a problem, into a braid, such that the probability that the output of the quantum computation is zero is a simple (polynomial time) function

† Research funded by the EPSRC and Vodafone, and supported in part by ESPRIT Project RAND-APX.
‡ Research supported in part by ESPRIT Project RAND-APX.

of the Jones polynomial of the braid at a 5th root of unity. For an exact statement of this see Freedman, Kitaev, Larsen and Wang [5], or the more detailed papers by Freedman, Kitaev and Wang [6], and Freedman, Larsen and Wang [7, 8].

It therefore follows that if we take $A(L, x)$ to be an oracle that returns the evaluation of the Jones polynomial of a braid L at a point x, any BQP computation can be replicated by a classical polynomial time algorithm with one call to A, *i.e.*, BQP $\subseteq \mathrm{P}^A$. Since computing the Jones polynomial is in general a #P-hard problem, this does not help. However, it is not an exact evaluation of the Jones polynomial that is required, but an approximate evaluation at a specific point for braids of a specific class. Hence we may look for a weaker oracle A' such that BQP $\subseteq \mathrm{P}^{A'}$.

In a different approach Fortnow and Rogers [4] link quantum complexity to the classical complexity class GapP. In particular they show that for any quantum Turing machine M running in time $t(n)$ there is a GapP function f such that for all inputs x

$$\mathbf{Pr}(M(x) \text{ accepts}) = \frac{f(x)}{5^{2t(|x|)}}.$$

Again evaluating a general GapP function exactly is #P-hard; however, one can simulate M using a polynomial algorithm with access to an oracle A'', where A'' is an oracle giving an approximation to the GapP function f.

With this motivation we examine the type of approximation needed in order to simulate a quantum computation, and then consider the complexity of such approximations. It turns out that an *additive approximation* is sufficiently powerful. We should emphasize that a polynomial time additive approximation scheme is weaker than the familiar and much studied *fully polynomial randomized approximation scheme* (FPRAS). However, it is well known that any function which counts objects for which the corresponding decision problem is NP-complete cannot have an FPRAS (unless NP = RP). We show below that *all* #P functions do have polynomial time additive approximation schemes under natural normalizations. We also show that in two senses this is the best sort of approximation we can hope to achieve in polynomial time (see Theorems 4.1, 4.3 and 4.4).

2. Quantum computing

A *link* L is a smooth submanifold of S^3, consisting of $c(L)$ disjoint simple closed curves. A *braid* on m strings is constructed as follows. Take m distinct points in a horizontal line $(p_1, p_2, \dots, p_m)$ and link them to m distinct points $(q_1, q_2, \dots, q_m)$ lying on a parallel line, by m disjoint simple arcs f_i in $\mathbb{R}^3$, so that f_i starts at p_i and ends at $q_{\pi(i)}$ where π is a permutation. A braid can be closed in numerous ways, by identifying the points p_i and q_j in some way, creating a link. Similarly any link can be represented as a braid. In particular, the *plat closure* of a braid on $2m$ strings is obtained by identifying the points p_{2i-1} and p_{2i}, and q_{2i-1} and q_{2i} for $1 \leqslant i \leqslant m$.

2.1. Topological computing and the Jones polynomial

One of the major difficulties in building a quantum computer has been the sensitivity of the system to outside interference. Freedman, Kitaev, Larson and Wang [5] introduced the notion of *topological quantum computing*, in an attempt to make the computations less sensitive to small disturbances. The basic idea is as follows. One can create pairs of special quasi-particles, called anyons, in a 2-dimensional plane sandwiched between two blocks of a superconductor. The anyons have a certain probability of annihilating each other (leaving a vacuum) when brought together. However, this probability changes, according to the laws of quantum mechanics when one anyon is moved around the other before they are brought together. Even if it is moved in a complete circle around the other, on reaching its original position the probability of annihilation is changed. Thus a system of a large number of these particles can be used as a quantum computer for decision problems; pairs of anyons are created, moved around relative to each other, and then a predefined pair of the anyons is brought together. If this pair annihilate each other leaving a vacuum, this is taken to be an output of 0 (or rejection); if they do not it is taken to be an output of 1 (or acceptance). The paths in the 2-dimensional surface, combined with a time dimension, give rise to a 3-dimensional representation of the 'computation' as a braid. There remain major difficulties in constructing such a quantum computer, and controlling the movement of anyons. However, one of the important results of Freedman, Kitaev, Larsen and Wang is that small changes in the paths of the anyons do not affect the outcome of the computation; indeed it is determined by the isotopy class of the braid, and therefore stable under perturbations of the paths that do not change the braiding itself.

The way in which the probability changes is sufficiently subtle that such a quantum computer is universal in the following sense. The Kitaev–Solovay theorem [12, 14] together with the density theorem of Freedman, Larsen and Wang [7, 8] yields an algorithm which given any quantum circuit on $m/2$ qubits and error parameter ϵ, outputs a braid on m strings using a polynomial number of crossings (polynomial in m and $\log \epsilon^{-1}$). The topological quantum computation using this braid efficiently simulates the quantum circuit, (the probability of acceptance is within ϵ of the correct value). Since an algorithm for a BQP problem can be used to generate a quantum circuit for a given instance, the above result gives an explicit method for finding an equivalent topological quantum computation, and so the class BQP is the same under either model.

Hence a quantum computation on $m/2$ qubits is approximately represented by a braid b. In showing that the topological quantum computation depends on the isotopy class of b alone [5], the following link L is considered. L is the plat closure of the composition of b^{-1}, b and a small loop γ inserted (between b^{-1} and b) around the leftmost two strings (see Figures 1 and 2). Both b and b^{-1} are needed as any quantum computation must be reversible; the loop γ effects a measurement of the qubit represented by the leftmost pair of strings. The conclusions of [5] may then be summarized as the following theorem: refer to [5] for full details.

Theorem 2.1. *Let π be a problem in BQP, with a polynomial time quantum algorithm $\mathscr{A}$, and let $\mathscr{I}$ be an instance of π. For any $\epsilon > 0$, a link L may be determined in time polynomial in $|\mathscr{I}|$ and $\log \epsilon$ such that*

$$\left| \mathbf{Pr}(\mathscr{A}(\mathscr{I}) = 0) - \frac{1}{1 + [2]_5^2} \left(1 + \frac{(-1)^{c(L)+w(L)}(-a)^{3w(L)} V_L(e^{2\pi i/5})}{[2]_5^{m(L)-2}} \right) \right| < \epsilon, \qquad (2.1)$$

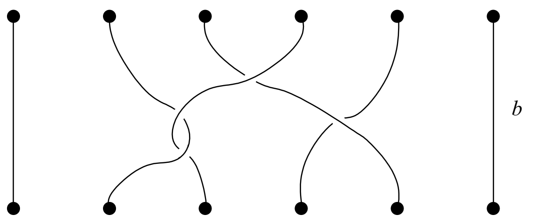

Figure 1. The braid b.

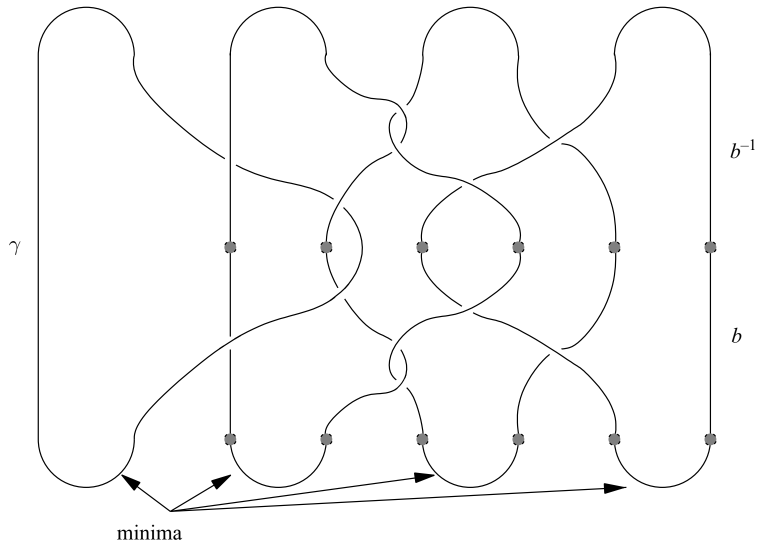

Figure 2. The link L.

where $a = e^{i\pi/10}$ *and* $[2]_5 = 2\cos\pi/5$ *and* $c(L), w(L)$ *and* $m(L)$ *are the number of components, writhe and number of minima of the link L respectively.*

The minima of [5] are the individual joins in the plat closure at the bottom of the braid, hence $m(L)$ is half the number of strings in b plus one. By construction, the number of strings in b is twice the number of (qu)bits in the input, hence $m(L) = |I| + 1$. The writhe is defined for an oriented link, and is the number of 'positively oriented' crossings minus the number of 'negatively oriented' crossings, with respect to the given orientation of L. It is easily computable. The Jones polynomial is also defined for oriented links; however, the formula above is independent of the orientation chosen for L. Since every crossing in b appears reversed in b^{-1}, these do not contribute to the writhe of L, hence the writhe is determined by the four crossings involving γ, and can only be $-4, 0$, or 4 (depending on the orientation of the two strands passing through γ). If L^* denotes L with the orientation of one component reversed, then $V_{L^*}(t) = t^{-3\lambda/2}V_L(t)$, where λ is the contribution to the writhe of L from crossings of the reversed component over (or under) the rest of L. Hence only reversing γ or one of the leftmost two strings can affect the Jones evaluation, and it is easily checked that this is compensated by the change in the terms involving the writhe. To be consistent, we will retain the notation $[2]_5$ for $2\cos\pi/5$ from [5] throughout.

It is Theorem 2.1 that gives rise to our interest in approximating $V_L(t)$. Further explanation of the derivation of this equation is given in [18] where the special but sufficient case $w(L) = 0$ is considered (we could restrict attention to braids with writhe zero without affecting the main results). Although this formula involves an evaluation at $e^{2\pi i/5}$, similar results can be obtained for the nth root of unity for any $n \geqslant 5, n \neq 6$ but these involve multiple Ls.

The preceding paragraphs explain how an evaluation of a Jones polynomial can yield the answer to a (general) quantum computation. There is a weak converse to this. Suppose we have a quantum computer at our disposal with which to learn something about a Jones evaluation of a link L. We may assume without loss of generality (Freedman, Larsen and Wang [7]) that our quantum computer is of the topological kind and thus nicely adapted to braids. We can (easily) write L as the plat closure of a braid b by starting with the link diagram and pulling the overcrossings up and the undercrossings down. Let m be the number of strands of this braid b. If we wish to evaluate $V_L(\alpha)$, $\alpha = e^{2\pi i/r}$, we encounter an important constant $d = 2\cos\pi/r$. The norm $|V_L(\alpha)|$ is bounded from above by $d^{m/2}$ with $|V_L(\alpha)| = d^{m/2}$ achieved only when L is the unlink on $m/2$ components, a case which occurs when b is the identity braid. Our quantum computer will be able to provide an additive approximation (see below) of $|V_L(\alpha)|$ as a variable with range $[0, d^{m/2}]$.

Given m marked points in the horizontal plane and the number α, there is a finite dimensional Hilbert space H on which m-strand braids act through a Jones representation p. The $m/2$ maxima (in the plat closure) determine a vector c in this space and the $m/2$ minima (in the plat closure) determine a vector in the dual H^*, which when identified with H by the Hermitian inner product, is the same c.

We have

$$\frac{V_L(\alpha)}{d^{m/2}} = \langle c|p(b)|c\rangle.$$

Furthermore $\mathrm{Prob}(|0\rangle) = |\langle c|p(b)|c\rangle|^2$, where $\mathrm{Prob}(|0\rangle)$ refers to the physical probability that below the cups, after all the 'particles' have been fused in pairs, the vacuum $|0\rangle$ is observed, that is, no nontrivial particles result from these fusions. The last formula reflects the quantum mechanical rule that the probability of observing an outcome, in this case $|0\rangle$, is proportional to the square of the component of the state vector in the $|0\rangle$-direction.

Because the range of $|V_L(\alpha)|$ depends exponentially on the number of strings in the braid, $m(b)$, our quantum computer will give much better information (sooner) if we succeed in displaying L with, or nearly with, the minimal $m(b)$, called the *braid index* of L.

Turning to the computational question, it is a theorem of Thistlethwaite [15] that when L is an alternating link, with associated plane graph G, then

$$V_L(t) = \alpha T(G; -t, -t^{-1})$$

where α is an easily computable function, and T is the Tutte polynomial of the planar graph. It is known [17] that even for planar graphs, computing $T(G; x, y)$ is #P-hard, except when (x, y) is one of a few special points, or lies on a hyperbola satisfying $(x-1)(y-1) = q \in \{1, 2\}$.

Since $(-e^{2\pi i/n}, -e^{-2\pi i/n}), n \geqslant 5$, is not one of these 'easy' points, exact computation in polynomial time is not feasible (unless #P=P). It also seems unlikely that an FPRAS exists for these points. However, the notion of an FPRAS seems to be much stronger than the kind of approximation that is needed in the current context. For any BQP language L there is a quantum Turing machine M such that for all $x \in L$, M accepts with probability at least 3/4, and for all $x \notin L$, M accepts with probability at most 1/4. Therefore all that we require is to determine which quartile of its range $V_L(e^{2\pi i/5})$ lies in. We return to this topic in Section 5.

When considering algorithms on braids or links, the size of the input is taken to be the number of crossings. A quantum gate on $m/2$ qubits is converted into a braid on m strings of length polylog $(1/\epsilon)$, therefore the number of crossings in the braid associated with a BQP circuit is polynomially related to the number of gates in the BQP circuit. This in turn is bounded by a polynomial in the input size, hence an algorithm will either be polynomial with respect to both the number of crossings in the braid and the number of input qubits to the circuit, or neither.

2.2. GapP functions

The class of counting functions which constitute #P is the set of functions that count certificates of membership of a language belonging to NP, hence #P functions are constrained to evaluate to non-negative integers. The class of functions GapP can be regarded as the closure of #P under subtraction, that is to say a function $f : \mathscr{I} \mapsto \mathbb{Z}$ is in GapP if and only if there exist functions $g, h \in \#P$ such that $f(I) = g(I) - h(I)$ for all $I \in \mathscr{I}$. The class AWPP can be defined as follows [3]. A language L is in AWPP if and only if there exist a polynomial p and a GapP function g such that for all $I \in \mathscr{I}$,

$$\begin{aligned} I \in L &\Rightarrow \frac{3}{4} \leqslant \frac{g(I)}{2^{p(|I|)}} \leqslant 1, \\ I \notin L &\Rightarrow 0 \leqslant \frac{g(I)}{2^{p(|I|)}} \leqslant \frac{1}{4}. \end{aligned} \tag{2.2}$$

The increase in power of quantum computation over classical computation is that in a quantum computer there is an ability to cancel out computations paths. Fortnow and Rogers [4] show that this power is captured by the class GapP, in which a similar effect is seen. In particular they show that BQP $\subseteq$ AWPP. It therefore follows that for a BQP language L, polynomial p and GapP function g satisfying (2.2), determining which quartile of the range $[0, 2^{p(|I|)}]$ contains $g(I)$ would be enough to determine membership of L.

To summarize, our foremost problems can be interpreted as finding a suitable approximation for the Jones polynomial of a link, $V_L(t)$, the Tutte polynomial of an associated planar graph, $T(G; x, y)$, at a particular point, or for the GapP functions arising from BQP languages.

3. Approximation

Given a function $\psi : \mathscr{I} \mapsto \mathbb{R}$ for which no efficient exact evaluation algorithm is known, one may be interested in an 'approximate' answer instead. A standard approach is to look for a fully polynomial randomized approximation scheme (FPRAS) for the problem. If ψ is such a function and $I \in \mathscr{I}$ is an input, then an FPRAS for ψ is a randomized algorithm that given any $I \in \mathscr{I}, \epsilon > 0$ will output $\hat{\psi}(I, \epsilon)$, such that

$$\mathbf{Pr}[|\hat{\psi}(I,\epsilon) - \psi(I)| > \epsilon\psi(I)] < 1/4,$$

and the running time is polynomial in $|I|$ and ϵ^{-1}.

Here one might be prompted to consider the following sort of approximation. Suppose we know a range in which the answer lies. Can we say where in that range the answer lies? Is it in the top or bottom half of the range, or in which quartile? We shall see that this approach is unlikely to be feasible, and in Section 3.1 we present an alternative. Clearly this type of approximation depends on the nature of the range. For the moment let us restrict our attention to the class of functions in #P. We will make the standard assumption that for a given NDTM M there exists a fixed polynomial p such that for any input x, all certificates have size $p(|x|)$ (so the total number of possible certificates of M is $2^{p(|x|)}$). We would like to answer the following problem, denoted by π_r: Given r, for which k is the number of accepting certificates for x between $\frac{(k-1)}{r}2^{p(|x|)}$ and $\frac{k}{r}2^{p(|x|)}$?

The problem π_2 is simply to determine which inputs have more than half of all certificates as accepting certificates. The set of languages in this class is exactly the set PP of probabilistic polynomial time languages. Furthermore, π_2 is clearly Turing reducible to π_{2s}, for any positive integer s, since if $\pi_{2s}(x) \leqslant s$ then $\pi_2(x) = 1$, otherwise $\pi_2(x) = 2$. Hence it is no surprise that this approach to approximation is NP-hard for #SAT, indeed the following lemma shows that any attempt to approximate #SAT in this way, or any problem with a parsimonious reduction to SAT, is unlikely to work. The proof is straightforward and we omit it; details may be found in [2].

Lemma 3.1. *For $k \in \mathbb{Z}$, deciding whether a CNF formula in n literals has more than 2^{n-k} solutions is NP-hard.*

When $k = 1$ the same decision for disjunctive normal form (DNF) formulae is equivalent to that for SAT, since the negation of a SAT formula is in DNF, and hence for an instance F, we have $\#\mathrm{SAT}(F) = 2^n - \#\mathrm{DNF}(\overline{F})$, where n is the number of literals. This observation leads to the following related lemma. Again, the proof is omitted and details may be found in [2].

Lemma 3.2. *For $k \in \mathbb{Z}$, deciding whether a DNF formula in n literals has at least 2^{n-k} solutions is NP-hard.*

This may seem more counterintuitive since not only is DNF in P, but also #DNF has an FPRAS [11]. On the other hand, the next lemma shows that the number of stable (independent) sets of vertices in a graph (#SS) can be approximated in this way, even though it is #P-complete and does not admit an FPRAS unless NP $=$ RP. Essentially this is because the 'natural' upper bound on the number of stable sets, 2^n, is far too big unless the graph has very few edges. For details of the proof see [2].

Lemma 3.3. *Let G be a graph on n vertices. For $r \in \mathbb{Z}$, determining for which k, $\#SS(G) \in [\frac{(k-1)}{r}2^n, \frac{k}{r}2^n)$ is computable in time polynomial in n and r.*

Lemma 3.1 suggests that we cannot hope to fix a partition of the range and then determine in polynomial time in which section the answer lies; the difficulty associated with an NP-complete decision problem can be shifted to exactly the boundary between two parts of our partition. We therefore consider an alternative method of approximation which will meet our needs.

3.1. Additive approximation

Our approach to approximation consists of determining a small section of the range depending on the input, and in which we can say the answer lies with high probability. This gives rise to an additive approximation.

Definition 1 (Additive Approximation (AA)). Given any function $f : \mathscr{I} \mapsto \mathbb{C}$ and a normalization $u : \mathbb{Z}^+ \mapsto \mathbb{R}^+$, an additive approximation for (f, u) is a probabilistic algorithm which, given any $I \in \mathscr{I}, \epsilon > 0$, produces an output $\hat{f}(I)$ such that

$$\mathbf{Pr}[|f(I) - \hat{f}(I)| > \epsilon u(|I|)] < 1/4,$$

in time polynomial in $|I|$ and ϵ^{-1}.

Note that the $1/4$ in the definition could be replaced by any $\delta \in (0, 1/2)$, since we could reduce this error probability in polynomial time by taking several runs of the algorithm. Note also that most of the time we shall be considering the case where f is real. In contrast to the set of functions admitting an FPRAS, which is closed under addition but not under subtraction (*e.g.*, #DNF(f) has an FPRAS, but #SAT(f) $= 2^n -$ #DNF($\bar{f}$) does not), we have the following result, whose proof we leave to the reader.

Proposition 3.4. *Suppose (f, u) and (g, v) admit AA algorithms, then there exists AA algorithms for $(-f, u)$, $(f + g, u + v)$ and $(f - g, u + v)$. If, in addition, $|f(I)| \leqslant u(|I|)$ and $|g(I)| \leqslant v(|I|)$ for all I, then there is an AA algorithm for (fg, uv).*

The normalization is crucial. Since we are most interested in determining where in the range of possible values the answer lies, we shall usually be taking u to be an upper bound on $|f|$ depending only on input size. An additive approximation allows errors up to an absolute value of $\epsilon u(|I|)$, whereas an FPRAS allows only errors up to an absolute value of $\epsilon f(I)$. It is therefore a weaker notion of approximation, and it is easy to check that any function that admits an FPRAS also admits an AA algorithm under any upper bound.

Lemma 3.5. *Let $f : \mathscr{I} \mapsto \mathbb{R}$ be a function that admits an FPRAS, and let $u : \mathbb{Z}^+ \mapsto \mathbb{R}$ satisfy $|f(I)| \leqslant u(|I|)$ for all inputs $I \in \mathscr{I}$. Then (f, u) has an AA algorithm.*

Note also that a given function will have an AA with respect to some normalizations but not others. For example, we show later that for the number of proper 3-colourings of a connected graph G on n vertices, $P_G(3)$ where P_G is the chromatic polynomial of G, we have an AA for $(P_G(3), 2^n)$. However, for any constant $\delta > 0$, $(P_G(3), (2-\delta)^n)$ does not have an AA unless NP = RP (Theorem 4.4). In other words we can determine $P_G(3)$ to within an additive error $\epsilon 2^n$ in polynomial time, but we cannot approximate to within an additive error $\epsilon(2-\delta)^n$. Note that if $(f(I), u(|I|))$ has an AA, then for any fixed polynomial p, $(f(I), u(|I|)/p(|I|))$ also does, since we can absorb the polynomial factor in the normalization into ϵ at only a polynomial slowing of the algorithm.

It is the determination of the 'best' normalization for a given function that causes the greatest difficulties, particularly in relation to approximating $V_L(t)$. Nevertheless our first positive result shows that any function belonging to #P does have an AA algorithm under very natural normalizations.

4. Additive approximations for #P functions

The class of functions which constitute #P can be regarded as the set of functions that count certificates of membership of a language belonging to NP. For a given NP-language L there will be infinitely many NDTMs which check membership of L, and the certificates for a given input I will depend on the machine used in verification.

The main result of this section is that all such counting functions have additive approximation schemes under the 'natural normalization' associated with the corresponding NDTM. For example, if we take $f(G)$ to be the number of Hamiltonian circuits in a graph G, then two possible NDTMs for checking membership of L are M_1, which takes as certificates subsets of the edges, and checks that these form a cycle of length $|V|$, and M_2, which takes as certificates an ordering of the vertices $v_1, v_2, \ldots, v_n$ and checks that the edges between any two adjacent vertices in the ordering, and between v_n and v_1, do appear in the graph (to avoid double counting we must insist that relative to some fixed ordering of the vertices $\pi : V \mapsto 1, \ldots, n$, we have $\pi(v_1) = 1$ and $\pi(v_2) < \pi(v_n)$). In each case the number of good certificates for a given graph G is exactly the number of Hamiltonian circuits of G; however, M_1 has $2^{|E(G)|}$ possible certificates, while M_2 has $(|V|-1)!/2$ possible certificates. In either case the number of possible certificates is a natural upper bound on the number of Hamiltonian circuits. We show below that there is an additive approximation algorithm under the normalization associated with any such bound.

Theorem 4.1. *Let f be a function in the class #P, with an associated NDTM M, so that, for a given instance I, M has $f(I)$ accepting certificates, each of length $p(|I|)$. Then there exists an additive approximation algorithm for $(f, 2^{p(|I|)})$ that runs in time polynomial in $|I|$ and ϵ^{-1}.*

Proof. Given an instance I of f, we will select t computation paths, or certificates, uniformly at random from the $2^{p(|I|)}$ possible. We then run M using these inputs, and let $X_i, i = 1 \dots t$ be indicator functions which take value 1 if and only if the ith computation path accepts I. The estimator for $f(I)$ is then $X = \frac{2^{p(|I|)}}{t} \sum_{i=1}^{t} X_i$.

Clearly $\mathbf{E}[X] = f(I)$. It remains to show that we can select t only polynomially large, such that the error bounds given in Definition 1 are satisfied. First note from Chebyshev's inequality that

$$\begin{aligned} \mathbf{Pr}\left[|X - f(I)| \geqslant \epsilon 2^{p(|I|)}\right] &\leqslant \frac{\mathbf{Var}(X)}{\epsilon^2 2^{2p(|I|)}} \\ &\leqslant \frac{1}{t} \frac{\frac{f(I)}{2^{p(|I|)}}\left(1 - \frac{f(I)}{2^{p(|I|)}}\right)}{\epsilon^2} \\ &\leqslant \frac{1}{t\epsilon^2}. \end{aligned}$$

Now if $t = 4\epsilon^{-2} + 1$, we have

$$\mathbf{Pr}\left[|X - f(I)| \geqslant \epsilon 2^{p(|I|)}\right] < 1/4. \qquad \square$$

Turning briefly to GapP functions, we get the following immediate corollary.

Theorem 4.2. *Let f be a GapP function such that $f = g - h$ where $g, h \in \#P$.*

(i) *Suppose that there are additive approximations for (g, u) and (h, v), then there is an additive approximation scheme for $(f, \max\{u, v\})$;*

(ii) *Suppose that $g(I)$ and $h(I)$ have certificates of length $p(|I|)$ for all I, then there is an additive approximation scheme for $(f, 2^p)$.*

Proof. From Proposition 3.4 we have that there is an additive approximation for $(f, u + v)$. From Definition 1 we can halve the permitted error for only a polynomial increase in running time, hence there is an AA for $(f, \frac{u+v}{2})$ and therefore also $(f, \max\{u, v\})$, which gives (i). When g and h have certificates of length p, by Theorem 4.1 there are AA schemes for $(g, 2^p)$ and $(h, 2^p)$, (ii) now follows from (i). $\square$

We have seen that all functions contained in #P have an AA algorithm relative to normalization by the size of the certificate space, and it is reasonable to ask if we could do better. However, we give two results that suggest this is already the best we can do in general. First we will see that sharpening our approximation to a logarithmic scale for the number of proper 5-colourings of a graph is NP-hard. Secondly, we show that the normalization by the number of possible certificates cannot be improved significantly in the case of the number of k-colourings of a graph.

Theorem 4.3. *Let $P_G(5)$ be the number of proper 5-colourings of a graph. Then for a general graph G on n vertices, there cannot be an additive approximation algorithm for $(\log(P_G(5) + 1), 3n)$ that runs in time polynomial in n and ϵ^{-1} unless $NP = RP$.*

Proof. Consider an NDTM for $P_G(5)$ that takes as certificates any 5-colouring of the graph, hence the certificates are of length $n\lceil\log 5\rceil = 3n$, where n is the number of vertices in G. We show that an AA algorithm for $(\log(P_G(5)+1), 3n)$ would be able to solve the NP-complete problem of determining whether a graph is 5-colourable.

Let G be a graph on n vertices and consider the following polynomial time transformation. We form G^+ by adding n isolated vertices to G. If G is not 5-colourable, then nor is G^+. However, if G is 5-colourable, each 5-colouring can be extended to 5^n 5-colourings of G^+. Therefore, if G is not 5-colourable $\log(P_{G^+}(5)+1) = 0$, whereas if G is 5-colourable,

$$\begin{aligned}\log(P_{G^+}(5)+1) &\geqslant \log(5^n.5!+1)\\ &> 2n.\end{aligned}$$

Hence an additive approximation algorithm for $(\log(P_{G^+}(5)+1), 6n)$ could determine whether or not G is 5-colourable in random polynomial time. □

Theorem 4.1 shows that for connected graphs there exists an AA algorithm for $(P_G(k), (k-1)^n)$ as follows. We can take an arbitrary spanning tree on G, and take the set of certificates to define colourings relative to this spanning tree, giving $k(k-1)^{n-1}$ possible certificates. We can then adjust the normalization by a constant factor $(k-1)/k$. We show that this cannot be improved, in the sense that the normalization (and therefore the error) cannot be reduced by any exponential factor. We have already noted that the normalization can be improved by any fixed polynomial factor.

Theorem 4.4. *If $NP \neq RP$ then for any fixed $k \geqslant 3, \delta > 0$ there cannot be a polynomial time AA algorithm for $(P_G(k), \phi(n))$ for connected graphs G on n vertices, for any function $\phi(n)$ of order $O((k-1-\delta)^n)$.*

Proof. Let $\phi(n) \leqslant c(k-1-\delta)^n$ for sufficiently large n. Take r such that $(k-1-\delta) \leqslant (k-1)^{1-1/r}$. Given any graph G, we form a graph H by attaching a path of length $n(r-1)$ to a vertex of G. Now

$$P_G(k)(k-1)^{n(r-1)} = P_H(k), \tag{4.1}$$

$$P_G(k) = \frac{P_H(k)}{\left((k-1)^{1-1/r}\right)^{nr}}, \tag{4.2}$$

$$P_G(k) = \frac{cP_H(k)}{\phi(nr)}\frac{\phi(nr)}{c\left((k-1)^{1-1/r}\right)^{nr}}. \tag{4.3}$$

Now suppose that there is an AA algorithm for $(P_H(k), \phi(|H|))$, then we can get an approximation $\hat{P_H}(k)$ within an additive error of $\frac{1}{2c}\phi(nr)$. Using $\hat{P_H}(k)$ and equation (4.2), we obtain an approximation $\hat{P_G}(k)$. By equation (4.3), $\hat{P_G}(k)$ is within an additive error of $1/2$, since

$$\frac{\phi(nr)}{c\left((k-1)^{1-1/r}\right)^{nr}} \leqslant \frac{\phi(nr)}{c(k-1-\delta)^{nr}} \leqslant 1.$$

Since $P_G(k)$ is integral, we can therefore determine it exactly. □

5. Approximating $V_L(t)$ and related quantities

We have seen in Section 2.1 that our primary problem is to decide whether or not there exists an additive approximation scheme for $(V_L(e^{2\pi i/5}), [2]_5^{m/2})$ where L is the plat closure of a braid on m strings, indeed an additive approximation for the absolute value of the Jones polynomial suffices. We make this precise in the following theorem.

Theorem 5.1. *Let A be a oracle which takes as input a braid b on m strings and $\epsilon > 0$, and returns an additive approximation for $(|V_L(e^{2\pi i/5})|, [2]_5^{m/2})$, where L is the plat closure of b. Then* $\mathrm{BQP} = \mathrm{P}^A$.

Proof. Recall from Section 2.1 that given a braid on m strings, a topological quantum computer can be constructed such that the probability of output zero is $\frac{|V_L(e^{2\pi i/5})|^2}{[2]_5^m}$ and the computer runs in time polynomial in m. Given $\epsilon > 0$, using independent runs of this computer and a standard sampling approach, the probability of zero can be estimated to within an error of ϵ^2, where the number of runs is polynomial in ϵ^{-1}. Hence we may estimate $|V_L(e^{2\pi i/5})|$ to within an absolute error of $\epsilon[2]_5^{m/2}$ in polynomial time. Therefore $\mathrm{P}^A \subseteq \mathrm{BQP}$.

Secondly, suppose we have a BQP language and an input x. By Theorem 2.1 we can determine a link L, of size polynomial in $|x|$, such that L satisfies equation (2.1), and the number of minima of L is $|x| + 1$. If x is in the language, $\mathbf{Pr}(0) < 1/4$, hence

$$\begin{aligned} 0 \leqslant \frac{1}{1+[2]_5^2}\left(1 + \frac{(-1)^{c(L)+w(L)}(-a)^{3w(L)}V_L\left(e^{2\pi i/5}\right)}{[2]_5^{|x|-1}}\right) &< 1/4, \\ -[2]_5^{|x|+1}[2]_5^{-2} \leqslant (-1)^{c(L)+w(L)}(-a)^{3w(L)}V_L\left(e^{2\pi i/5}\right) &< [2]_5^{|x|+1}[2]_5^{-2}\left(\frac{1+[2]_5^2}{4} - 1\right), \\ \left|V_L\left(e^{2\pi i/5}\right)\right| &< [2]_5^{|x|+1}0.39. \end{aligned} \tag{5.1}$$

Whereas, if x is not in the language, $\mathbf{Pr}(0) > 3/4$, hence

$$\begin{aligned} \frac{1}{1+[2]_5^2}\left(1 + \frac{(-1)^{c(L)+w(L)}(-a)^{3w(L)}V_L\left(e^{2\pi i/5}\right)}{[2]_5^{|x|-1}}\right) &> 3/4, \\ (-1)^{c(L)+w(L)}(-a)^{3w(L)}V_L\left(e^{2\pi i/5}\right) &> [2]_5^{|x|+1}[2]_5^{-2}\left(\frac{3(1+[2]_5^2)}{4} - 1\right), \\ \left|V_L\left(e^{2\pi i/5}\right)\right| &> [2]_5^{|x|+1}0.65. \end{aligned} \tag{5.2}$$

Clearly use of an oracle giving an additive approximation for $(|V_L(e^{2\pi i/5})|, [2]_5^{|x|+1})$ will enable us to distinguish these two cases with probability at least 3/4. Hence $\mathrm{BQP} \subseteq \mathrm{P}^A$. □

We saw in Section 2.1 the equivalence of the Jones polynomial and a specialization of the Tutte polynomial for alternating links, hence we would like an AA for a general planar graph G for

$$\left(T\left(G; -e^{2\pi i/5}, -e^{-2\pi i/5}\right), u\right),$$

where u is some reasonable upper bound.

Hyperbolae of the form $H_q := (x-1)(y-1) = q$ play a crucial role in the manipulation of the Tutte polynomial; loosely speaking, the process of performing a *tensor product* on an input graph G with some other fixed graph N enables us to 'move around' the Tutte plane. That is, the new graph $G \otimes N$ satisfies

$$T(G \otimes N; x, y) = f(N; x, y) T(G; X, Y), \tag{5.3}$$

where f and the arguments X, Y can be computed in time polynomial in $|x|, |y|$ and $|N|$. However, for any choice of N, the new points X, Y satisfy

$$(X-1)(Y-1) = (x-1)(y-1) = q. \tag{5.4}$$

Thus, such transformations restrict us to remain on the initial hyperbola H_q; see [10] for further details. The close relationship between points on the hyperbola enables us to use an additive approximation at one point to get an additive approximation at any other point (X, Y) on the same hyperbola for which there exists a suitable planar N which transforms (x, y) to (X, Y) by (5.3).

Proposition 5.2. *Let $x, y \in \mathbb{Q}$ and N a planar graph on k vertices be fixed. Suppose there is an AA scheme for $(T(G; x, y), u(n))$ for any planar G on n vertices and m edges. Then there is also an AA for*

$$(T(G; X, Y), u(n + m(k-2))),$$

where X and Y are the points determined by the transformation in (5.3) (*depending only on x, y and N*). □

Proof. Let X and Y be the points satisfying (5.3). Since $G \otimes N$ is planar (see the construction in [10]) we may use the AA scheme for

$$(T(G \otimes N; x, y), u(|V(G \otimes N)|))$$

to get an approximation to within an error $\epsilon u(|V(G \otimes N)|)$ with probability at least 3/4 in polynomial time. Note that $|V(G \otimes N)| = n + m(k-2)$. Since the running time of the AA scheme is polynomial in ϵ^{-1}, and $f(N; x, y)$ is a constant, we can approximate to within an error of $\epsilon |f(N; x, y)| u(n + m(k-2))$ and still run in polynomial time. By (5.3) we have

$$T(G; X, Y) = \frac{T(G \otimes N; x, y)}{f(N; x, y)}.$$

Hence the AA for $T(G \otimes N; x, y)$ yields an AA for $T(G; X, Y)$ with error at most $\epsilon u(n + m(k-2))$ in polynomial time. □

Because of the important role of these hyperbolae, it is natural to look at the hyperbolae containing the roots of unity $(-e^{\frac{2\pi i}{n}}, -e^{-\frac{2\pi i}{n}})$. These are H_{q_n}, where $q_n = 2 + 2\cos(2\pi/n)$, which cut the x-axis at

$$x = -1 - 2\cos(2\pi/n), \tag{5.5}$$

corresponding to an evaluation of the chromatic polynomial at one of the well-known *Beraha numbers* $B_n = 2 + 2\cos(2\pi/n)$. Since for real x and y and any graph N, the related points X and Y will also be real, we cannot find an N such that we can directly relate $T(G \otimes N; 1 - B_5, 0)$ and $T(G; -e^{\frac{2\pi i}{5}}, -e^{-\frac{2\pi i}{5}})$. Whether or not we can find a point within absolute value ϵ of $(1 - B_5, 0)$ that can be directly related to $(-e^{\frac{2\pi i}{5}}, -e^{-\frac{2\pi i}{5}})$ is an interesting ongoing question. We present some positive results below, and return to these difficulties in Section 7.

First note that by Theorems 4.1 and 4.2 we know that $T(G; x, y)$, $x, y \in \mathbb{Z}$ will have an AA scheme with respect to an appropriate normalization, since evaluations at these points are GapP functions. However, the drawback is that often the naive normalization will be too large. This will not always be the case, for example the point $(1 - \lambda, 0), \lambda \in \mathbb{Z}$, gives the number of proper λ colourings, and by Theorems 4.3 and 4.4 here we have a best possible normalization.

When we consider the non-integer points, the situation is more complicated. A straightforward sampling approach gives the following result.

Proposition 5.3. *For rational (x, y) and a connected graph G, there exists a AA algorithm for the following:*

(i) $(T(G; x, y), y^{|E|}(y-1)^{-|V|+1})$ *when* $\{x = 1, y > 1\}$,
(ii) $(T(G; x, y), x^{|E|}(x-1)^{|V|-|E|-1})$ *when* $\{x > 1, y = 1\}$,
(iii) $(T(G; x, y), y^{|E|}(x-1)^{|V|-1})$ *when* $\{x > 1, y > 1\}$.

In other regions, in particular where there are negative terms in the expansion of T, cancellation between terms means that there is no longer a natural upper bound by which to normalize. We return to this problem in Section 7.

6. An alternative approach

Returning to our original motivation, at the moment we are unable to determine whether there is an AA algorithm for $(V_L(e^{2\pi i/5}), [2]_5^{m/2})$ where L is the plat closure of a braid on m strings. However, we now show that in order to simulate a quantum computation it would be sufficient to determine the sign of the real part of $V_L(e^{2\pi i/5})$. Particularly in the case that the writhe of L is zero, and hence $V_L(e^{2\pi i/5})$ is real, this seems an easier problem.

Theorem 6.1. *Let $A(L)$ be an oracle that returns the sign of the real part of the Jones polynomial of the link L evaluated at $e^{2\pi i/5}$. Then* $\mathrm{BQP} \subseteq \mathrm{P}^A$.

Proof. This proof follows that of Theorem 5.1. Suppose we have a BQP language and an input x. By Theorem 2.1 we can determine a link L, of size polynomial in $|x|$, such that L satisfies equation (2.1), and the number of minima of L is $|x| + 1$. We now assume that $w(L) = 0$; the proof in the cases $w(L) = 4$ and $w(L) = -4$ follow by a similar argument. Simplifying equation (2.1) in the case $w(L) = 0$, we have

$$\mathbf{Pr}(0) = \frac{1}{1 + [2]_5^2}\left(1 + \frac{(-1)^{c(L)} V_L\left(e^{2\pi i/5}\right)}{[2]_5^{m(L)-2}}\right). \tag{6.1}$$

If x is in the language, $\mathbf{Pr}(0) < 1/4$, hence

$$\begin{aligned}\frac{1}{1+[2]_5^2}\left(1+\frac{(-1)^{c(L)}V_L\left(e^{2\pi i/5}\right)}{[2]_5^{|x|-1}}\right) &< 1/4,\\ (-1)^{c(L)}V_L\left(e^{2\pi i/5}\right) &< [2]_5^{|x|-1}\left(\frac{1+[2]_5^2}{4}-1\right),\\ (-1)^{c(L)}V_L\left(e^{2\pi i/5}\right) &< -[2]_5^{|x|-1}0.09 < 0. \end{aligned} \tag{6.2}$$

Whereas, if x is not in the language, $\mathbf{Pr}(0) > 3/4$, hence

$$\begin{aligned}\frac{1}{1+[2]_5^2}\left(1+\frac{(-1)^{c(L)}V_L\left(e^{2\pi i/5}\right)}{[2]_5^{|x|-1}}\right) &> 3/4,\\ (-1)^{c(L)}V_L\left(e^{2\pi i/5}\right) &> [2]_5^{|x|-1}\left(\frac{3(1+[2]_5^2)}{4}-1\right),\\ (-1)^{c(L)}V_L\left(e^{2\pi i/5}\right) &> [2]_5^{|x|-1}1.71 > 0. \end{aligned} \tag{6.3}$$

Clearly use of an oracle giving an additive approximation for the sign of $V_L(e^{2\pi i/5})$ will enable us to distinguish these two cases with probability at least 3/4. Hence $\mathrm{BQP} \subseteq \mathrm{P}^A$. □

In the previous section we outlined the importance of the hyperbolae $H_q := (x-1)(y-1) = q$ to the Tutte polynomial. For x, y, X, Y and N related as in equation (5.3), we can determine the sign of $T(G; X, Y)$ if we can determine the sign of $T(G \otimes N; x, y)$.

This gives rise to the natural question of the complexity of determining whether a function is greater than or less than zero, in particular the Tutte polynomial, of which the Jones is a specialization. It is immediate from the definitions that the Tutte is non-negative in the region $x, y \geqslant 0$. At all other integer points on the axes the Tutte polynomial counts either colourings or flows, up to easy multiplicative factors. Since these factors may be positive or negative, we can always select one of either '$T(G; x, y)$ is non-negative' or '$T(G; x, y)$ is non-positive' that is true, in polynomial time. In the above situation we are not concerned with cases in which the value is exactly zero, hence this would suffice. We consider the situation at other points in the next section.

7. Some combinatorial and complexity questions

We close with the following questions which have been prompted by this work.

In Section 5 we noted that we are unable to find a suitable normalization for approximating the Tutte polynomial when the expansion included negative terms. We return to this here and examine the chromatic polynomial to highlight the difficulties. We have seen that for a connected graph G, we have an additive approximation for $(P_G(\lambda), (\lambda-1)^n))$ for all $\lambda \in \mathbb{Z}^+$. However, we are most interested in an additive approximation at the non-integral Beraha numbers. One might hope to achieve the above approximation for all $\lambda \in \mathbb{R}^{>1}$; however, this seems unlikely as $(\lambda-1)^n$ is not even close to being an upper bound for $P_G(\lambda)$. Indeed, consider the complete graphs: for small δ,

$$P_{K_n}(1+\delta) = (1+\delta)(\delta)(-1+\delta)\cdots(-n+\delta+1) \tag{7.1}$$
$$\approx (-1)^{n-2}\delta(n-2)!. \tag{7.2}$$

This prompts the first open question.

Question 1. *What is the best upper bound depending on λ, n and m for $|P_G(\lambda)|$ for all (planar) graphs G on n vertices and m edges?*

As far as we are aware the best upper bound known [19] is

$$|P_G(\lambda)| \leqslant |\lambda|^{n-m}(|\lambda|+1)^m \quad \lambda \in \mathbb{C}.$$

For general graphs we can make the following small improvement.

Proposition 7.1. *Let G be a graph and let $\lambda \in \mathbb{C}$, then*

$$|P_G(\lambda)| \leqslant \left(\frac{m}{n}-1\right)^{n-m}\left(\frac{m}{n}\right)^m \qquad \textit{for } \frac{m}{n} \geqslant |\lambda|+1,$$
$$|P_G(\lambda)| \leqslant (|\lambda|)^{n-m}(|\lambda|+1)^m \qquad \textit{for } \frac{m}{n} < |\lambda|+1.$$

If G is a connected graph then

$$|P_G(\lambda)| \leqslant \left(\frac{m}{n-1}-1\right)^{n-m-1}\left(\frac{m}{n-1}\right)^m|\lambda| \qquad \textit{for } \frac{m}{n-1} \geqslant |\lambda-1|,$$
$$|P_G(\lambda)| \leqslant (|\lambda-1|-1)^{n-m-1}(|\lambda-1|)^m|\lambda| \qquad \textit{for } \frac{m}{n-1} < |\lambda-1|.$$

These bounds hold for all $\lambda \in \mathbb{C}$; however, the Beraha numbers have special characteristics. The evaluations of the chromatic polynomial at these points have some beautiful, but not totally understood, properties [16]. The values begin $4, 0, 1, 2, 1+\tau, 3, \ldots$ and converge towards 4, where τ is the golden ratio $\frac{1+\sqrt{5}}{2}$. The integers in this series are clearly central to the theory of chromatic polynomials. Writing $B_5 = 1+\tau$, then for any plane triangulation T on n vertices:

$$|P_T(B_5)| \leqslant \tau^{5-n}. \tag{7.3}$$

For a connected graph G with average degree at least 3.24 (note that a planar triangulation has average degree $6-12/n$), the above proposition gives

$$|P_G(B_5)| \leqslant \left(\frac{m}{n-1}-1\right)^{n-m-1}\left(\frac{m}{n-1}\right)^m B_5.$$

Hence we ask the following.

Question 2. *Is there a better bound for $|P_G(B_n)|$ than there is for an evaluation at a general point?*

Following the results of Section 6 we are also prompted to examine the complexity of determining whether the Tutte polynomial is greater than or equal to, or less than zero at a given point. Recall that this decision problem is trivial for $x, y \geqslant 0$, and for integer points on the axes. Again considering the specialization to the chromatic polynomial we ask the following.

Question 3. *For fixed $\lambda \in \mathbb{Q}$, is it NP-hard to decide whether $P_G(\lambda)$ is greater than or equal to, or less than zero?*

Note that this is trivial for $\lambda \in \mathbb{Z}$. It is also P-time decidable for $\lambda < 32/27$ by the following theorem of Woodall [19] and Jackson [9].

Theorem 7.2. *Let G be a graph without loops on n vertices, κ components and b blocks.*
(i) *If* $\lambda < 0$, *then $P_G(\lambda)$ is nonzero with the sign of* $(-1)^n$.
(ii) *If* $0 < \lambda < 1$, *then $P_G(\lambda)$ is nonzero with the sign of* $(-1)^{n-\kappa}$.
(iii) *If* $1 < \lambda < \frac{32}{27}$, *then $P_G(\lambda)$ is nonzero with the sign of* $(-1)^{n-\kappa-b}$.

Note that $P_G(\lambda) \neq 0$ for $\lambda \in \mathbb{Q}\backslash\mathbb{Z}$, since the chromatic polynomial has integer coefficients. It is easy to show the following.

- Let $\lambda \in \mathbb{Q}\backslash\mathbb{Z}$. If deciding whether $P_G(\lambda) > 0$ is NP-hard, then it is also NP-hard to decide whether $P_G(\lambda + 1) > 0$ for a general graph G.

However, since it is easy to decide for $\lambda < 32/27$, the converse cannot be true for all $\lambda \in \mathbb{Q}\backslash\mathbb{Z}$ unless these questions are all in P. It would be interesting to know the answer to the following questions.

Question 4. *Does there exist a critical $\alpha > 0$ such that deciding whether $P_G(\lambda)$ is greater than or less than zero is NP-hard for all rational $\lambda > \alpha$, $\lambda \notin \mathbb{Z}$?*

Question 5. *Is this critical α equal to 32/27?*

As before we are more interested in evaluating the chromatic polynomial at the Beraha points than at general non-integers, and the graphs we are most interested in are planar. Hence we ask the following specific question.

Question 6. *For planar graphs, is the problem of deciding whether $P_G(B_n)$ is greater or less than zero NP-hard?*

For any graph G, not necessarily planar, it is known that $P_G(B_n) \neq 0$ for $n \geqslant 5, n \neq 6, 10$, [13]. Also Tutte [16] has shown that for any planar triangulation the following equation holds, writing $B_{10} = \tau\sqrt{5}$,

$$P_T(B_{10}) = \sqrt{5}\tau^{3(n-3)}(P_T(B_5))^2. \tag{7.4}$$

So $P_T(B_{10}) > 0$ for all plane triangulations T, indeed a simple reverse induction shows that for any planar graph G, $P_G(B_{10}) > 0$ holds. Further, for any outerplanar graph G, we can form the planar graph G^+ by adding a new vertex adjacent to all original vertices. Since

$$P_{G^+}(\lambda + 1) = (\lambda + 1)P_G(\lambda)$$

holds for all positive integers, it holds for all $\lambda \in \mathbb{R}$. Noting that $B_{10} = B_5 + 1$, we conclude that $P_G(B_5) > 0$ for all outerplanar graphs G.

Acknowledgements

The authors would like to thank the referee for many helpful comments and suggestions, in particular for suggesting a simpler proof of Theorem 4.3. We also thank Graham Brightwell and Colin McDiarmid for comments on an earlier version of this work [2].

References

[1] Adleman, L., DeMarris, J. and Huang, M. (1997) Quantum computability. *SIAM J. Computing* **26** 1524–1540.
[2] Bordewich, M. (2003) The complexity of counting and randomised approximation. PhD thesis, New College, Oxford University.
[3] Fenner, S. (2003) PP-lowness and a simple definition of AWPP. *Theory Comput. Syst.* **36** 199–212.
[4] Fortnow, L. and Rogers, J. (1999) Complexity limitations on quantum computation. *J. Comput. Syst. Sci.* **59** 240–252.
[5] Freedman, M., Kitaev, A., Larsen, M. and Wang, Z. (2003) Topological quantum computation. *Bull. Amer. Math. Soc.* **40** 31–38.
[6] Freedman, M., Kitaev, A. and Wang, Z. (2002) Simulation of topological field theories by quantum computers. *Commun. Math. Phys.* **227** 587–603.
[7] Freedman, M., Larsen, M. and Wang, Z. (2002) Density representations of braid groups and distribution of values of Jones invariants. *Commun. Math. Phys.* **228** 177–199.
[8] Freedman, M., Larsen, M. and Wang, Z. (2002) A modular functor which is universal for quantum computation. *Commun. Math. Phys.* **227** 605–622.
[9] Jackson, B. (1993) A zero-free interval for chromatic polynomials of graphs. *Combin. Probab. Comput.* **2** 325–336.
[10] Jaeger, F., Vertigan, D. and Welsh, D. J. A. (1990) On the computational complexity of the Jones and Tutte polynomials. *Math. Proc. Camb. Phil. Soc.* **108** 35–53.
[11] Karp, R., Luby, M. and Madras, N. (1989) Monte Carlo approximation algorithms for enumeration problems. *J. Algorithms* **10** 429–448.
[12] Kitaev, A. (2002) Quantum computations: algorithms and error correction. *Russian Math. Survey* **52:61** 1191–1249.
[13] Salas, J. and Sokal, A. (2000) Transfer matrices and partition-function zeros for antiferromagnetic Potts models. *J. Statist. Phys.* **98** 551–588.
[14] Solovay, R. Private communications.
[15] Thistlethwaite, M. B. (1987) A spanning tree expansion of the Jones polynomial. *Topology* **26** 297–309.
[16] Tutte, W. T. (1970) On chromatic polynomials and the golden ratio. *J. Combin. Theory Ser. B* **9** 289–296.
[17] Vertigan, D. and Welsh, D. J. A. (1992) The computational complexity of the Tutte plane: the bipartite case. *Combin. Probab. Comput.* **1** 181–187.
[18] Wang, Z. (2003) Addendum: Derivation of the formula in [FKLW]. `www.tqc.iu.edu`.
[19] Woodall, D. R. (1978) Zeros of chromatic polynomials. In *Combinatorial Surveys: Proceedings of the Sixth British Combinatorial Conference* (P. J. Cameron, ed.), Academic Press, London, pp. 199–223.