The BQP-hardness of approximating the Jones polynomial

Dorit Aharonov and Itai Arad
Department of Computer Science and Engineering, Hebrew University, Jerusalem, Israel
E-mail: itaia@cs.huji.ac.il

New Journal of Physics 13 (2011) 035019 (32pp)
Received 13 December 2010
Published 22 March 2011
Online at http://www.njp.org/
doi:10.1088/1367-2630/13/3/035019

Abstract. A celebrated important result due to Freedman et al (2002 Commun. Math. Phys. 227 605–22) states that providing additive approximations of the Jones polynomial at the kth root of unity, for constant \( k = 5 \) and \( k \geq 7 \), is BQP-hard. Together with the algorithmic results of Aharonov et al (2005) and Freedman et al (2002 Commun. Math. Phys. 227 587–603), this gives perhaps the most natural BQP-complete problem known today and motivates further study of the topic. In this paper, we focus on the universality proof; we extend the result of Freedman et al (2002) to \( ks \) that grow polynomially with the number of strands and crossings in the link, thus extending the BQP-hardness of Jones polynomial approximations to all values to which the AJL algorithm applies (Aharonov et al 2005), proving that for all those values, the problems are BQP-complete. As a side benefit, we derive a fairly elementary proof of the Freedman et al density result, without referring to advanced results from Lie algebra representation theory, making this important result accessible to a wider audience in the computer science research community. We make use of two general lemmas we prove, the bridge lemma and the decoupling lemma, which provide tools for establishing the density of subgroups in SU(\( n \)). Those tools seem to be of independent interest in more general contexts of proving the quantum universality. Our result also implies a completely classical statement, that the multiplicative approximations of the Jones polynomial, at exactly the same values, are \#P-hard, via a recent result due to Kuperberg (2009 arXiv:0908.0512). Since the first publication of those results in their preliminary form (Aharonov and Arad 2006 arXiv:quant-ph/0605181), the methods we present here have been used in several other contexts (Aharonov and Arad 2007 arXiv:quant-ph/0702008;

\(^1\) Author to whom any correspondence should be addressed.
1. Introduction

What is the computational power of quantum computers? This question is fundamental from both a computer scientist and a physicist point of view. This paper attempts to improve our understanding of this question, by studying perhaps the most natural BQP-complete problem known to us today: the problem of approximating the Jones polynomial. Here we try to clarify the reasons for its BQP-hardness, as well as extend its applicability, and, in this way, gain a better understanding and new tools for proving quantum universality in general.
1.1. Background

The Jones polynomial, discovered in 1985 [26], is a very important knot invariant in topology; it assigns a one-variable Laurent polynomial $V_L(t)$ to a link $L$, in such a way that isotopic links are assigned the same polynomial. It is an extremely difficult object to compute—evaluating it at any point except for a few trivial ones is #P-hard [24]. The importance of the Jones polynomial was manifest in connections with numerous areas in mathematics, from the statistical physics model known as the Potts model to the study of DNA folding. Among its many connections, an extremely important one was drawn by Witten in 1989 to quantum mechanics, and, specifically, to topological quantum field theory (TQFT) [36]. Witten showed how the Jones polynomial naturally appears in the Wilson lines of the SU(2) Chern–Simons TQFT.

About a decade later, TQFT entered the scene of quantum computation when Freedman suggested a computational model based on this theory [17]. The works of Freedman et al [18]–[20] showed an equivalence between the TQFT model and the standard model of quantum computation. On the one hand, they gave an efficient simulation of TQFT by a quantum computer [18]. On the other hand, they showed that quantum computation can simulate TQFT efficiently [19]. These results draw interesting connections between quantum computation and the Jones polynomial. The simulation of TQFT by quantum computers implicitly implies (via the results of Witten) the existence of a quantum algorithm for approximating the Jones polynomial evaluated at the fifth root of unity $t = \exp(2\pi i/5)$, to within a certain additive approximation window. In the other direction, the simulation of quantum computers by TQFT implicitly implies that the same Jones polynomial approximation problem (with the same additive approximation window) is BQP-hard; the proof uses Lie algebras extensively. This draws an important equivalence between the two seemingly completely different problems of quantum computation and the approximation of the Jones polynomial of links.

The above-mentioned important results were stated in the language of TQFT, and relied on advanced results from Lie algebra theory; this made the results inaccessible for much of the computer science community for a while. In [15], clear statements of the results were provided using a computational language, but without proofs; an explicit algorithm was thus still missing, as well as a proof from first principles of universality.

A few years ago, Aharonov et al [8] provided an explicit and efficient quantum algorithm for the problem of approximating the Jones polynomial of a given link, at roots of unity of the form $\exp(2\pi i/k)$, using the standard quantum circuit model. The algorithm uses a combination of simple to state combinatorial and algebraic results of over 20 years ago due to Jones. The main ingredient is a certain matrix representation, called the path-model representation, which maps elements from an algebra of braid-like objects (called the Temperley–Lieb algebra $\text{TL}_n(d)$) to operators acting on paths of $n$ steps on a certain graph $G_d$. In the cases where this representation is unitary, this gives a simple-to-state quantum algorithm for the approximation of the Jones polynomial: the matrices are applied by the quantum computer, and the approximation of the Jones polynomial is derived by approximating a certain trace of the resulting unitary matrix. This bypasses the TQFT language altogether.

The universality proof due to Freedman et al [19], stated first in terms that were also closer to the TQFT language, can also be made explicit in the standard quantum model language, without referring to TQFT. This can be done using a mapping suggested by Kitaev [30] and independently Wocjan and Yard [39], in which the basis states of one qubit are encoded by one of two possible paths of length four in the space of the path-model representation.
The results described above imply an explicit proof in the standard quantum computation model that the problem of approximating the Jones polynomial at the fifth root of unity, and in fact, for any primitive root of unity \(\exp(2\pi i/k)\), for constant \(k > 4, k \neq 6\) is BQP-complete and thus equivalent in a well-defined sense to standard quantum computation. This is arguably the most natural BQP-complete problem known to us today (but see [37, 38, 40]). The fact that the problem is BQP-complete highlights the importance of this problem in the context of quantum computational complexity and motivates deeper investigations of the intriguing connections and insights revealed by those results.

We remark that, as is usually done in the literature, we slightly abuse notation and when we say a problem is BQP-complete, we in fact mean this in the context of promise problems; just like in the case of BPP, there are no known BQP-complete problems in the strict sense of the term, and so we actually mean that the problem is PromiseBQP-complete. For a detailed discussion of this point, see [5, 6] and references therein.

One natural direction to pursue is to try to generalize the algorithm in various directions. Several results extended the Jones polynomial approximation algorithms to other knot invariants and to more general braid closures (see e.g. [21, 39]) to the evaluation of the Potts model partition function and the Tutte polynomial [3], and to approximations of tensor networks [10], as well as to the Turaev–Viro invariant [9]. In this paper, we take the other direction: we attempt to study and further clarify the reasons for the BQP-hardness of those problems, and expand its range of applicability, with the hope of clarifying the source of the computational power of quantum computation.

1.2. Results and implications

We ask here the following natural question. It turns out that the algorithms given in the work of Aharonov et al work not only for constant \(k\), but also for asymptotically growing \(k\)s. To be more precise, Aharonov et al [8] give an efficient quantum algorithm to approximate the Jones polynomial of a certain closure (called the plat closure) of an \(n\)-strand braid with \(m\) crossings, evaluated at a primitive root of unity \(\exp(2\pi i/k)\). The running time of the algorithm is polynomial in \(m, n\) and \(k\). The algorithm is therefore efficient even if \(k\) grows polynomially with \(n\). On the other hand, the proof of BQP-hardness is only known to hold for constant \(k\). Therefore, in [8], the following natural question was raised: What is the complexity of approximating the Jones polynomial for polynomially bounded \(k\)? It was left open whether it is BQP-hard, doable in BPP or maybe somewhere in between.

In this paper, we resolve this question, and show that for any polynomially bounded \(k\), the problem is BQP-hard. The following is a rough statement of the result; the exact statement is given in theorem 13 in section 5.

**Theorem 1.1.** The problem of approximating the Jones polynomial of the plat closure of a given braid \(b\), with \(m\) crossings, at \(\exp(2\pi i/k)\), where both \(m\) and \(k\) are polynomially bounded in \(n\), to within the same accuracy as is done in [8], is BQP-complete.

We thus show that in all cases where the AJL algorithm [8] is known to be efficient, we derive that the problem it solves is BQP-complete. The proof is not a mere extension of the previous constant \(k\) case, and there are severe problems to overcome.

As a side benefit, our proof also simplifies the original proofs for the constant \(k\) case [19], and reproves it almost from first principles, without using advanced results from Lie algebra,
thus making it more accessible to the computer science audience. Indeed, since the preliminary 
publication in [2] of the results presented here, the methods developed here were applied in 
several other contexts (see section 1.7).

We will soon outline the general approach towards the proof of the universality of the 
constant $k$ case, the difficulties in extending the proof to non-constant $k$ and our methods 
for overcoming them. Before that, let us mention some interesting connections and further 
implications on the complexity of multiplicative approximations of the Jones polynomial.

1.3. Implication on hardness of the multiplicative approximation problem

A significant ‘drawback’ of the AJL algorithm is the fact that it provides an additive approximation to the Jones polynomial. It can approximate the Jones polynomial up to an additive error of $\Delta / \text{poly}(n)$, with $\Delta$ being some scale (which is easy to calculate). The problem is that the exact value of the polynomial might be exponentially smaller than $\Delta$, making this kind of additive approximation useless. A partial answer to this ‘drawback’ is found in its complementing result, the BQP-hardness theorem, which we reprove in this paper; it shows that despite the seeming weakness of the approximation, it is as hard as the hardest problems that a quantum computer can solve. Thus, there exist links for which the additive approximation of the AJL is nontrivial. Nevertheless, one can rightfully argue that the situation is still not satisfactory; additive approximations are far less interesting from an algorithmic point of view, and we would have liked to focus on a much better and more natural approximation notion, namely a multiplicative one.

Goldberg and Jerrum [22] studied the complexity of multiplicative approximations of the Tutte polynomial. The Jones polynomial (of alternating links) is a special case of this important polynomial. Their results imply that the multiplicative approximation (to within a constant arbitrarily close to 1) of the Jones polynomial of alternating links, at certain real values, is NP hard (relative to RP). Those values, however, do not intersect the values for which BQP additive approximations exist due to the AJL algorithm, as they apply only to real points, while the AJL works at the complex roots of unity. And so one might still hope that the AJL algorithm as well as the universality proofs for those values can be improved and stated using the multiplicative approximation notion.

A beautiful recent result obtained by Kuperberg [32] helps to shed light on this matter. Kuperberg observed that BQP hardness of additive approximations seems to go hand in hand with the #P-hardness of the multiplicative approximation at the same values, via the result of Aaronson that PostBQP $= \text{PP}$ [1], as well as on the exponential efficiency of the Solovay–Kitaev algorithm. Using these ideas, Kuperberg proved that the multiplicative approximations of the Jones polynomial of a plat closure of a braid, evaluated at the $k$th root of unity, for constant $k = 5$ and $k \geq 7$, are #P-hard. Essentially, the argument is that by Aaronson’s result, in order to solve #P-hard problems, it suffices to be able to compute, or even provide multiplicative approximations of, conditional probabilities for the outputs of a given quantum circuit. However, for $k$, for which BQP-hardness of additive approximations of the Jones polynomial holds, we can use the same mapping from circuits to links used for the BQP-hardness to derive exponentially good approximations of those conditional probabilities in terms of the Jones polynomial of some link, where the link need only be polynomial in the number of gates in

---

² They have also shown that in some special cases, multiplicative approximations of the Tutte polynomial are #P-hard, but these cases do not correspond to the Jones polynomial.
the circuit, due to the exponential efficiency of the Solovay–Kitaev theorem. Note that the final result is a purely classical result that is derived using quantum complexity tools. It turns out that the same argument works for the universality proofs in this paper, and, hence, we get the following corollary:

**Corollary 1.1.** The problems of the approximation of the Jones polynomial at the same points and parameters for which theorem 1.1 implies BQP-hardness, with the approximation replaced by multiplicative approximation to within a constant arbitrarily close to 1, are #P-hard.

We now proceed to outline the proof of theorem 1.1. Let us start with explaining the constant \( k \) case first.

1.4. Proof outline of the constant \( k \) case

Given an algorithm that calculates the Jones polynomial of any link at \( \exp(-2\pi i/k) \) (for some integer \( k > 4 \) and \( k \neq 6 \)) in polynomial time in the number of crossings in the link, and a classical Turing machine—we can simulate a quantum computer efficiently. How is that possible? The key idea, which is used also in the algorithmic result of [8], is the existence of an intimate connection between two, seemingly distinct, worlds: links and unitary matrices. This connection is the so-called ‘path-model representation’, which is defined for every integer \( k \). The \( k \)th path-model representation maps every \( n \)-strand braid \( b \) in the braid group (e.g. figure 1) into a unitary matrix \( \rho(b) \).

\( \rho(b) \) acts on a Hilbert space spanned by *paths* of length \( n \) on a certain graph, \( G_k \), which is simply the line graph of \( k - 2 \) segments (see figure 2).

As was shown by Jones [26, 27], the unitary matrix \( \rho(b) \) can be related to the Jones polynomial of the link \( b^{pl} \) derived from the braid \( b \) by closing its strands in a certain way called the *plat closure* (see the example in figure 3).

The connection is that the expectation value \( \langle \alpha | \rho(b) | \alpha \rangle \) (where \( | \alpha \rangle \) is some special state) is proportional to the Jones polynomial \( V_{b^{pl}} \) of the plat closure of the braid \( b \), evaluated at the \( k \)th root of unity (with an easy to calculate proportionality constant). To prove BQP-hardness of the approximation of the Jones polynomial, it thus suffices to prove the BQP-hardness of the approximation of \( \langle \alpha | \rho(b) | \alpha \rangle \) for a given braid \( b \).
The strategy to do this is by showing that any given quantum circuit, namely a sequence of gates $U_L \cdot U_{L-1} \cdots U_1$, can be mapped to a braid $b$, such that the value $\langle 0^n | U_L \cdot U_{L-1} \cdots U_1 | 0^n \rangle$ is proportional to $\langle \alpha | \rho(b) | \alpha \rangle$.

The BQP-hardness proof thus boils down to showing that a general quantum gate can be approximated efficiently using the unitary images of braids by the path-model representation. More precisely, one considers some subset of the generators of the braid group (each generator is simply a crossing of two adjacent strands). Each such generator is mapped to a certain ($k$-dependent) unitary operator on the space of paths. The main difficulty of the proof lies in showing that the group generated by these unitaries is dense in a large enough subgroup of the unitary group that contains all unitary gates. Once this is shown, it is standard to apply the famous Solovay–Kitaev theorem [31] to show that density implies efficiency. In other words, once the subgroup is dense, Solovay–Kitaev gives a method to approximate every gate in the quantum circuit by a polynomially bounded in length sequence of generators, and universality follows.

But how does one prove the density? The starting point is the Kitaev–Wocjan–Yard four-step encoding [30, 39], which encodes the state of one qubit into four steps paths. For two qubits, these paths correspond to eight-strand braids, four for each encoded qubit. In fact, the four-dimensional (4D) Hilbert space of the two qubits is encoded into a space spanned by four paths, which is embedded into an ‘invariant’ space spanned by all 14 paths on the eight strands;
for details, see section 4.1 and figure 8. Density thus means that we can approximate any matrix in SU(14) (and thus also any matrix in SU(4) embedded in it) using our \( k \)-dependent generators. In order to prove density, the idea is to first restrict our attention to some 2D subspace, and show density in SU(2). This was essentially done by Jones [25]. We then gradually increase the dimensionality of the space on which we have density, to SU(14), by adding one or more dimensions at a time; to this end, we introduce two lemmas that are useful tools for proving universality in general: the *bridge lemma* 4.1 and the *decoupling lemma* 4.2. We explain these lemmas later in the introduction, in section 1.6, since they are of independent interest. Using these lemmas, we can build up our way from density on SU(2) to the desired density on SU(14); this completes the density proof of the constant \( k \) case. We get an almost self-contained, fairly elementary proof.

1.5. Proof outline of the polynomially growing \( k \) case

We would now like to move to the asymptotically growing \( k \) case. Here, however, there is a subtle point in the above line of arguments. Indeed, density still holds. But the step of *density implies that efficiency* fails. The starting point of the Solovay–Kitaev theorem is the construction, using the set of generators, of an \( \epsilon \)-net in the unitary group, where \( \epsilon \) is some small enough constant. Such an \( \epsilon \)-net is easy to construct, given a *fixed* set of generators that span a dense subgroup—essentially, brute force would do the trick. More precisely, one considers an arbitrary \( \delta \)-net in the unitary group SU(14), for \( \delta \) being \( \epsilon /3 \); such a net contains a finite number of points. Due to the density, by brute force we can find \( \delta \) approximations of all those finitely many points by finite products of our generators, and those products constitute the \( \epsilon \)-net. The complexity of this initial step might be horrible, but it depends only on \( k \) and \( \epsilon \), and not on \( n \); for a fixed \( k \), it is thus constant. However, if \( k \) is asymptotically growing in \( n \), then so are the generators. The brute force procedure might depend in an uncontrollable way on \( k \), and thus on \( n \). It is therefore no longer clear whether the very first step of the Solovay–Kitaev theorem, that of creating the \( \epsilon \)-net, can be performed efficiently.

We give here a very rough sketch of how we overcome this difficulty. Looking at the \( k \) dependence of the generators, we see that as \( k \to \infty \), their eigenvectors converge to a fixed limit, while their corresponding eigenvalues behave as \( \exp(-2\pi i/k) \). The idea then is to fix a \( k_0 \) and to consider special auxiliary generators: generators whose eigenvectors coincide with the \( k \to \infty \) limit, but their eigenvalues are the fixed \( k_0 \) eigenvalues, \( \exp(-2\pi i/k_0) \). This set of auxiliary generators is independent of \( n \), and we show that it too spans a dense subgroup in SU(14); thus we can construct an \( \epsilon \)-net from it using a straightforward brute-force search. For every sufficiently large \( k \), the eigenvectors of the \( k \)-dependent generators would be close enough to those of the limit \( k \to \infty \), and thus to the eigenvectors of the auxiliary generators; by taking the \( k/k_0 \) power of the of the \( k \)-dependent generators, we obtain the \( k/k_0 \) power of their eigenvalues \( \exp(-2\pi i/k) \) and thus we approximate the \( \exp(-2\pi i/k_0) \) eigenvalues of the auxiliary generators. For large enough \( k \), the eigenvectors of \( k \) would be close enough to the \( k \to \infty \) eigenvectors, and the truncation error when approximating \( k/k_0 \) by an integer would be negligible, and so we obtain an approximation of the auxiliary generators by \( k \)-dependent generators. We can now substitute these approximations in the \( \epsilon \)-net made of the auxiliary generators, to get an efficient construction of an \( \epsilon \)-net consisting of the \( k \)-generators. We can now apply the Solovay–Kitaev theorem using this net.
1.6. Tools for universality: the bridge lemma and the decoupling lemma

We provide here rough statements of the two lemmas we use here for proving density, since they seem to be useful for proving universality in a variety of other contexts.

The bridge lemma roughly says that if we have density in the unitary groups acting on two orthogonal subspaces, $A$ and $B$, with $\dim(B) > \dim(A)$ and an additional unitary that mixes the two subspaces (in some well-defined sense), we also have density on the direct sum of the spaces. This general lemma is very reminiscent of a lemma that appeared in an early version of [4, 7]. Its proof is based on simple linear algebra, and is iterative; it uses a combination of ideas presented by Aharonov and Ben-Or [4, 7] and by Kitaev et al [31].

The decoupling lemma deals with the following scenario: a certain subgroup of the unitary matrices can be shown to be dense when restricted to one subspace and also to another subspace orthogonal to it. When we wish to combine the two spaces, we encounter a problem because there may be correlations between how the matrices act on the two subspaces. The lemma states that if the dimensions of the spaces are different, it is possible to ‘decouple’ those correlations and approach any unitary on one space while approaching the identity on another, and vice versa. The proof of the decoupling lemma uses a simple analysis.

1.7. Related work and discussion

Since the first publication of the results presented here (in preliminary form) [2], they have been used in several contexts: Shor and Jordan [33] built on the methods we develop here in order to prove the universality of a variant of the Jones polynomial approximation problem, in the model of quantum computation with one clean qubit. In their extension of the AJL algorithm [8] to the Potts model [3], Aharonov et al built on those methods to prove the universality of approximating the Jones polynomial in many other values, and even in values that correspond to non-unitary representations. We hope that the method we present here will be useful in future in other contexts as well.

Finally, we mention that the results of this paper should be viewed in a somewhat wider context of the notion of quantum ‘encoded universality’. By that we mean the following: rather than showing that a set of gates on $n$ qubits generates a dense subgroup in the unitary group on those $n$ qubits, as is done in the standard notion of quantum universality, one proves that the set of gates in fact generates a dense set in the unitary group on a space of dimension less than $2^n$, which is *embedded* or *encoded* in the bigger $2^n$-dimensional Hilbert space. If the encoding can be computed efficiently and the encoded Hilbert space is of large enough dimension, this suffices for an efficient simulation of universal quantum computation.

In fact, although not explicitly stated, encoded universality is exactly what was proved by Freedman et al in their original universality proof of the TQFT simulation [19], and of course in the universality proofs based on them [30, 39], including the present paper. The first time encoded universality was used can probably be tracked to the proof that real quantum computation suffices to simulate all of the quantum computation by Bernstein and Vazirani [14]. This notion was also used in various other contexts, e.g. in the context of fault tolerance and decoherence free subspaces [12], as well as in the encoded universality proof of the Heisenberg interaction [13, 29]. In this paper, we in fact provide general tools for proving density for such encoded universality scenarios.

This paper is organized as follows. In section 2, we provide the required mathematical background on links, braids, Temperley–Lieb algebra and the path-model representation that
is needed for the proof. In section 3, we state and prove the constant-\(k\) universality theorem by using the density and efficiency theorem. This theorem, which is the heart of the proof, is proved separately in section 4. In section 5, we state and prove the main result of this paper, the BQP-hardness for the \(k = \text{poly}(n)\) case. Finally, in section 6, we prove the bridge and decoupling lemmas that are used in the density proof in section 4.

2. Background: braid groups, the Temperley–Lieb algebra and path-model representations

In this section, we give a brief overview of the algebraic and topological definitions and tools that we need in order to prove theorem 1.1. We define the braid group, its embedding in the Temperley–Lieb algebra, and the path-model representation and its relation to the Jones polynomial. A more detailed description of these subjects based on first principles can be found in [3, 8].

2.1. The braid group \(B_n\)

Loosely speaking, a braid is a set of \(n\) strands that connect two horizontal bars, such that each strand is tied exactly to one peg on the top bar and one peg on the bottom bar. When drawing the braid schematically on a paper, the strands may pass over and under each other, but at any point they must not be completely horizontal. Braids that can be deformed into each other without tearing any of the strands are considered identical. An illustration of a four-strand braid is given in figure 1.

The set of all braids with \(n\) strands forms an infinite and discrete group, which is called the braid group \(B_n\). The product rule for \(b_1b_2\) is defined by placing the braid \(b_1\) above the braid \(b_2\) and fusing the bottom of the \(b_1\) strands with the top of the \(b_2\) strands. The identity element is the braid with \(n\) straight lines that connect each peg at the bottom bar to its corresponding peg at the upper bar.

In 1925, Artin proved that \(B_n\) admits a finite presentation (the Artin presentation) [11], with \(n - 1\) generators \(\{\sigma_i\}\) that satisfy the following constraints,

\[
\sigma_i\sigma_j = \sigma_j\sigma_i \quad \text{for} \quad |i - j| \geq 2, \quad (1)
\]

\[
\sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1}. \quad (2)
\]

Pictorially, \(\sigma_i\) is a braid that is identical to the unity braid in all strands except for the \(i\) and \(i + 1\) strands that cross each other once (the \(i + 1 \rightarrow i\) strand goes over the \(i \rightarrow i + 1\) strand), connecting the lower \(i\)th peg to the upper \(i + 1\) peg and vice versa. The diagram of \(\sigma_2\) in \(B_4\) is given in figure 4. It is an easy exercise to verify graphically that the braid generators indeed satisfy (1) and (2).

2.2. From braids to links

A link is an embedding of one or more closed loops in \(\mathbb{R}^3\). We first notice that a braid can be transformed into a link by connecting its open endpoints. Such an operation is called a closure, and here we focus on one particular closure: the plat closure. This closure is defined only for braids with an even number of strands. It is the link that is formed by connecting the top pegs with odd numbers to the peg on their right, and doing the same with the bottom pegs. The plat
Figure 4. The generator $\sigma_2$ in the braid group $B_4$.

Figure 5. Multiplying tangles in the $\text{TL}_n(d)$ algebra. The first diagram is put on top of the second, and the pegs are connected. In the resulting tangle, every loop is removed and replaced with an overall $d$ factor.

The closure of a braid $b \in B_n$ is denoted by $b^\text{pl}$. Figure 3 shows the plat closure of the four-strand braid from figure 1.

2.3. The Temperley–Lieb algebra $\text{TL}_n(d)$

We are interested in defining certain useful representations of the braid group $B_n$, which we will later relate to the Jones polynomial. To this end, we first consider the Temperley–Lieb algebra $\text{TL}_n(d)$ [34]. This is because the generators $\sigma_i$ of the braid group $B_n$, and therefore all of $B_n$, can be embedded in that algebra. Hence, any representation of $\text{TL}_n(d)$ yields a representation of $B_n$.

For any scalar $d$, the $\text{TL}_n(d)$ algebra is an algebra of tangle diagrams that, much like braid diagrams, connect $n$ lower pegs to $n$ upper pegs. However, unlike the case of braid diagrams, here we do not allow crossings, but we do allow horizontal lines, including local minima and maxima. Finally, closed loops are not allowed. To multiply two tangles, we put one on top of the other, connecting lower pegs with upper pegs. Any closed loop that is created in this process is then taken out of the diagram and replaced with an overall factor of $d$, called the loop value. See figure 5 for an example.

The braid group $B_n$ can be embedded in the $\text{TL}_n(d)$ algebra using the following map, shown schematically in figure 6,

$$\sigma_i \rightarrow AE_i + A^{-1}11.$$  \hspace{1cm} (3)
Figure 6. The embedding of the braid group $B_n$ in the Temperley–Lieb algebra $TL_n(d)$. The generator $\rho_i$ is mapped into a superposition of the tangles $E_i$ and $1$, with $A$ given by $-(A^2 + A^{-2}) = d$.

Here, $1$ is the identity tangle—the tangle that connects every lower $i$th peg to the corresponding upper $i$th peg. $E_i$ is the tangle that is formed by a ‘cap’ that connects the lower $i$ and $i+1$ pegs and a ‘cup’ that connects the corresponding upper pegs, and the reset of the pegs are connected by identity lines. Finally, $A$ is the scalar defined by

$$d = -(A^2 + A^{-2}). \quad (4)$$

It is an easy exercise to verify that the $\sigma_i$ defined by (3) indeed satisfy the Artin presentation (1) and (2).

It follows that any matrix representation of the $TL_n(d)$ algebra yields a matrix representation of the braid group $B_n$. We will next construct the representations that we will be using.

2.4. The path-model representation

The path-model representations are a family of representations for the Temperley–Lieb algebras [34] that induce representations for the braid group $B_n$ via (3). They were constructed in [26, 27], and form the basis of the AJL algorithm [8]. Here we will provide the just minimal details that are needed to understand the use of these representations when applied to the braid group. A broader presentation on this beautiful subject, together with its relation to the Temperley–Lieb algebras and the knot invariants, can be found in [3, 8].

We work with a sub-family of the path-model representations, which is characterized by an integer $k \geq 3$ and yields a representation for $TL_n(d)$ with

$$d = 2 \cos(\pi/k). \quad (5)$$

When applied to the braid group via (3), using

$$A = i e^{-i\pi/(2k)} \quad (6)$$

(which satisfies (4)), this representation becomes a unitary representation of $B_n$. The image of every tangle $T \in TL_n(d)$ (or $b \in B_n$) under this representation is denoted by $\rho(T)$ (or $\rho(b)$) and it acts on a finite Hilbert space. To understand the structure of this space, we introduce the graph $G_k$, which is made from a set of $k-1$ sites (vertices) and $k-2$ edges that connect them. The sites are ordered from bottom to top one above the other, as described in figure 2. To each site we assign a number according to its position, starting with 1 at the bottom.

We then consider all possible $n$-step walks (paths) over the graph $G_k$ that start at site 1 and never leaves $G_k$. We use these paths to define the Hilbert space $H_{n,k}$ of $n$-step paths over
Figure 7. An example of a tangle and two compatible paths. Here the lower path \( p = 1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 1 \) is shown to be compatible with the upper path \( p' = 1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 1 \). These paths define a unique labeling of every region in the tangle by a vertex of \( G_k \).

\[ G_k \]: every path \( p \) is mapped to a vector \( |p\rangle \in H_{n,k} \), and we define the set of all paths to be an orthonormal basis of \( H_{n,k} \).

To define \( \rho \), we will describe the action of \( \rho(T) \) on some \( |p\rangle \in H_{n,k} \), where \( T \in TL_n(d) \). \( \rho(T)|p\rangle \) is a linear combination of paths. A path \( p' \) with a non-vanishing weight in that combination is said to be compatible with \( p \) with respect to \( T \). To decide whether \( p' \) is compatible with \( p \), we first draw \( T \) in a box. The \( n \) lower pegs divide the lower boundary of the box into \( n+1 \) segments, which we call lower gaps, and similarly the upper pegs define \( n+1 \) upper gaps. We now associate every vertex of the path \( p \) with the lower gaps (starting from the leftmost gap, which must be 1), and the upper gaps with \( p' \). We notice that as \( T \) contains no loops, it partitions the box into non-overlapping regions, and each region must be connected to at least one gap (either lower or upper). Therefore every region in the box is associated with at least one vertex, either of \( p \) or of \( p' \) (or of both). Then \( p \) and \( p' \) are compatible iff every region is associated with exactly one vertex. When this happens, the paths define a ‘labeling’ or a ‘coloring’ of the regions. An example of two compatible paths and the coloring they define is shown in figure 7. There, the path \( p = 1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 1 \) is shown to be compatible with the path \( p' = 1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 1 \) with respect to the tangle \( T \).

To finish the definition of the path-model representation, we have to specify the weight of every compatible path. There is a beautiful derivation that yields such weights so that what we get is indeed a representation (see [3, 8] for a combinatorial exposition of this derivation); here we will not provide the details but only the resulting definition of the matrix representation. We define \( \theta \equiv \pi / k \) and then

\[ \lambda_j \equiv \sin(\pi j / k) = \sin(j \theta), \]  

and we have by equations (5) and (6)

\[ A = ie^{-i\theta/2}, \]  

\[ d = 2 \cos \theta. \]

We can now define the matrices \( \Phi_i = \rho(E_i) \) and through them \( \rho_i = \rho(\sigma_i) \) by (3)

\[ \rho(\sigma_i) = \rho(A E_i + A^{-1} \mathbb{1}) = A \Phi_i + A^{-1} \mathbb{1}. \]
We consider an arbitrary path $\rho = z_1 \rightarrow z_2 \rightarrow z_3 \rightarrow \ldots$, where $z_i$ is the position on the path before taking the $i$th step. For brevity, we will designate by $\ominus$ the path in which the $i$ and $i+1$ steps are descending (i.e. to $z_i - 1$ and then to $z_i - 2$), and similarly $\oslash$ for two ascending steps. Similarly, the paths $\oslash$ and $\ominus$ denote a combination of ascending and descending, and it is agreed that they coincide with each other at all but the $i$ and $i+1$ steps. Then the $\Phi_i$ matrices are given by

$$\Phi_i|\oslash\rangle = 0,$$  
(11)

$$\Phi_i|\oslash\rangle = \frac{\lambda_{z_{i+1}}}{\lambda_{z_i}}|\oslash\rangle + \frac{\sqrt{\lambda_{z_{i+1}} + 1}}{\lambda_{z_i}}|\ominus\rangle,$$  
(12)

$$\Phi_i|\ominus\rangle = \frac{\lambda_{z_{i-1}}}{\lambda_{z_i}}|\ominus\rangle + \frac{\sqrt{\lambda_{z_{i-1}} + 1}}{\lambda_{z_i}}|\oslash\rangle,$$  
(13)

$$\Phi_i|\ominus\rangle = 0.$$  
(14)

Note that by (10), the operators $\rho_i$ have the same invariant subspaces as the $\Phi_i$ operators. Specifically, in the paths basis, $\rho_i$ breaks into 1D and 2D blocks (but note that these are different blocks for different operators) that consist of $|\oslash\rangle, |\ominus\rangle$ paths and $|\oslash\rangle, |\ominus\rangle$ paths, respectively. We also see that $\Phi_i$, and hence $\rho_i$, does not change the end point of a path, because they only mix paths that coincide at all but the $i+1$ site. Therefore, the path representation breaks into representations over subspaces that correspond to paths that end at a particular $\ell$. We designate these subspaces by $H_{n,k,\ell}$, and note that $H_{n,k} = \sum_{\ell=1}^{k-1} \oplus H_{n,k,\ell}$.

2.5. From braids to links to the Jones polynomial

It turns out that there is a very strong connection between the path-model representation of a braid and the Jones polynomial of its plat closure. We will not define here the Jones polynomial, but only refer to it by the notation, $V_L(\cdot)$. The Jones polynomial of the plat closure of every $b \in B_n$ can be given by a ‘sandwich’ product of the operator $\rho(b)$ with a special vector $|\alpha\rangle \in H_{n,k}$. Specifically, let $b^{pl}$ denote the plat closure of the braid $b$ and $V_{b^{pl}}(\cdot)$ its Jones polynomial. Let $\alpha = 1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow \ldots = \oslash \ominus \oslash \ominus \ldots$ denote the ‘zigzag’ path and $|\alpha\rangle$ its corresponding vector. Then the following equality holds,

$$\langle \alpha | \rho(b) | \alpha \rangle = \frac{1}{\Delta} V_{b^{pl}}(A^{-1}),$$  
(15)

where $\Delta$ is given by

$$\Delta \equiv d^{n/2-1} (-A)^{3w(b^{pl})}.$$  
(16)

Here, $A = \text{ie}^{-in/(2k)}$ and $d = 2\cos(\pi/k)$ are given in (5) and (6), and $w(b^{pl})$ is the writhe of the link $b^{pl}$, which is a trivial function of a link—it is basically a sum over all its crossings. $V_{b^{pl}}(A^{-1})$ is the Jones polynomial of $b^{pl}$, evaluated at $A^{-1} = \exp(-2\pi i/k)$.

We note that both the writhe and the Jones polynomial are only defined for oriented links, and therefore we must choose some orientation for $b^{pl}$ to make the above well defined; it does not matter, however, which orientation we pick since the combination $(-A)^{-3w(b^{pl})}V_{b^{pl}}(A^{-1})$ is independent of the orientation (in agreement with the lhs of (15)). In fact, this combination is
precisely the Kauffman bracket \( \langle b^p \rangle \) [28], which is also a polynomial of the link, but we will not use this terminology here. We further note that \(|\Delta| = d^{n/2-1} = (2\cos(\pi/k))^{n/2-1}\). As we shall see in the following section and in section 5, this constant is the approximation scale of our additive approximation.

### 3. BQP-hardness for constant \( k \)

Equation (15) from the previous section establishes the connection between the Jones polynomial and a quantum-mechanical-like expectation value \( \langle \alpha | \rho(b) | \alpha \rangle \). It is this connection that enables, on the one hand, the approximation of the Jones polynomial by a quantum computer and, on the other hand, the simulation of a quantum computer by approximating the Jones polynomial.

In this section, we show the latter result. Specifically, we show that approximating the Jones polynomial at the \( k \)th root of unity \( \exp(2\pi i/k) \) for \( k > 4 \) and \( k \neq 6 \) is BQP-hard. This result has already been proved by Freedman et al [19]. Here and in the following section, we give our version of the proof, which uses somewhat more elementary machinery and enables us to prove the BQP-hardness of the \( k = \text{poly}(n) \) problem in section 5.

For a constant \( k \), the exact statement of the result is as follows:

**Theorem 3.1** (BQP-hardness for a fixed \( k \).) Let \( k > 4, k \neq 6 \) be an integer, and \( t = \exp(2\pi i/k) \) its corresponding root of unity. Let \( b \in B_n \) be a braid with \( m = \text{poly}(n) \) crossings, and \( b^pl \) its plat closure. Finally, let \( V_{b^pl}(t) \) be its Jones polynomial, and \( \Delta \) as defined in (16) so that \(|\Delta| = (2\cos(\pi/k))^{n/2-1}\). Then given a promise that either \(|V_{b^pl}(t)| \leq \frac{1}{10} |\Delta| \) or \(|V_{b^pl}(t)| \geq \frac{9}{10} |\Delta| \), it is BQP-hard to decide between the two.

The rest of this section is devoted to the proof of this theorem. The outline of the proof was given in the introduction, and we repeat it here for readability, and also in order to add a few missing details. Fix a \( k \), as in theorem 3.1. We assume we have access to a machine that for a given braid provides approximations of the Jones polynomial within the same accuracy as in theorem 3.1, in polynomial time. By (15) and the definition of the approximation window \( \Delta \) in (16), this means that we have access to a machine that, given a braid \( b \), can decide whether \(|\langle \alpha | \rho(b) | \alpha \rangle | \) is larger than 0.9 or smaller than 0.1. It therefore suffices to reduce a known BQP-hard problem to this latter approximation problem of \(|\langle \alpha | \rho(b) | \alpha \rangle | \).

We will do this by using the following problem, which is easily shown to be BQP-hard by standard arguments: given a quantum circuit of \( L \) gates, \( U = U_L \cdots U_2 \cdot U_1 \) on \( n \) qubits with \( L = \text{poly}(n) \), decide whether \(|\langle 0^\otimes n | U | 0^\otimes n \rangle | \leq \frac{1}{3} \) or \(|\langle 0^\otimes n | U | 0^\otimes n \rangle | \geq \frac{2}{3} \). This problem is easily seen to remain BQP-hard even if we assume that the qubits that the circuit acts on are set on a line, and each gate \( U_i \) is two-local, acting on adjacent qubits.

We will show how, given such a quantum circuit, one can efficiently find a braid \( b \) of polynomial number of strands and crossings such that \( \langle \alpha | \rho(b) | \alpha \rangle \) approximates \( \langle 0^\otimes n | U | 0^\otimes n \rangle \) (say, up to an additive error of 1/10). This will suffice to prove theorem (3.1).

We begin by introducing the Kitaev–Wocjan–Yard four-step encoding that maps strings of bits to paths and would enable us to map any quantum gate \( U_j \) to an operator on the space of paths.
3.1. The four-step encoding

In the four-step encoding, we encode every bit by a four-step path that starts and ends at the first site,

\[ |0\rangle \overset{\text{def}}{=} |1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rangle = |\begin{array}{c} \uparrow \\ \downarrow \\ \uparrow \\ \downarrow \end{array}\rangle, \quad (17) \]

\[ |1\rangle \overset{\text{def}}{=} |1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rangle = |\begin{array}{c} \uparrow \\ \downarrow \\ \uparrow \\ \downarrow \end{array}\rangle. \quad (18) \]

Then a string of \( n \) encoded qubits \(|x\rangle\) is encoded as a \( 4n \)-step path in \( H_{4n,k} \), and is denoted by \(|x\rangle\). These paths are not arbitrary paths in \( H_{4n,k} \), as they return to the first site every four steps. We designate by \( S \) the subspace that is spanned by all these paths. We note that the zigzag path \(|\alpha\rangle \in H_{4n,k} \) is actually the encoded string \(|0\rangle^{\otimes n}\).

Next, just as we encode bit strings, we encode the computational gates: every gate \( U \) is encoded by

\[ U = \sum_{i,j} U_{ij} |i\rangle \langle j| + \mathbb{1}_{\text{over rest of space}}, \quad (19) \]

where \( i \) and \( j \) denote bit strings and \(|i\rangle \) and \(|j\rangle\) their encoding. Then the product \( U = U_L \cdots U_1 \) naturally translates to \( U = U_L \cdots U_1 \) and so by finding braids \( b_i \in B_{4n} \) such that \( \rho(b_i) \simeq U_i \) and then taking their product \( b = b_L \cdots b_1 \), we will obtain \( \rho(b) \simeq U \). Consequently,

\[ \langle 0^{\otimes n}|U|0^{\otimes n}\rangle = \langle 0^{\otimes n}|U_1|0^{\otimes n}\rangle \simeq \langle 0^{\otimes n} |\rho(b)|0^{\otimes n}\rangle = \langle \alpha |\rho(b)|\alpha\rangle. \quad (20) \]

In fact, we will not be so ambitious; we will only require that \( \rho(b_i) \simeq U_i \) on the subspace \( S \) and show that this suffices.

The advantage of using this particular encoding is that, together with the tensorial structure of the qubits, it allows us to concentrate on the ‘reduced’ braid group \( B_8 \) instead of the larger group \( B_{4n} \). Let us explain exactly what is meant by that. Suppose that we wish to perform an operation on the \( s, s+1 \) encoded qubits of some path \(|p\rangle \in S \). Then we must use a braid \( b \in B_{4n} \) that mixes the eight strands \( 4(s-1)+1 \rightarrow 4(s+1) \) while being trivial on the rest. However, since \(|p\rangle \in S \), its path reaches the first site before the steps \( 4(s-1)+1 \) and \( 4(s+1)+1 \). Therefore the three partial paths that are defined by the steps \( 1 \rightarrow 4(s-1), 4(s-1)+1 \rightarrow 4(s+1), 4(s+1)+1 \rightarrow 4n \) are all legitimate paths over the graph \( G_k \) (i.e. they start and end at the first site and never leave \( G_k \)). We designate these partial paths by \( p_0, \tilde{p} \) and \( p_1 \), respectively, and rewrite \(|p\rangle = |p_0\rangle \otimes |\tilde{p}\rangle \otimes |p_1\rangle\). Note also that \(|\tilde{p}\rangle \in H_{8,k,1} \). We will add a tilde to all vectors and operators that act on the \( H_{8,k,1} \) space. In particular, we define \( \tilde{b} \in B_8 \) to be the ‘reduced’ version of \( b \), created by the eight nontrivial strands of \( b \).

It is now easy to verify that

\[ \rho(b)|p\rangle = \rho(b)(|p_0\rangle \otimes |\tilde{p}\rangle \otimes |p_1\rangle) = |p_0\rangle \otimes (\rho(\tilde{b})|\tilde{p}\rangle) \otimes |p_1\rangle. \quad (21) \]

This follows from the definition of the generators \( p_i \) in (10) and (11)–(14), which only depend on \( z_i \)—the position of the path after \( i-1 \) steps, and not on the index \( i \) itself.

By linearity, we can extend (21) to all vectors in \( S \), which are simply superpositions of encoded paths. Therefore, as long as \(|p\rangle \in S \), it is enough to search for an appropriate braid in the much simpler group, \( B_8 \), instead of looking in the full \( B_{4n} \) group. What remains to be shown is that (i) we can approximate any operator on \( H_{8,k,1} \) using a \( \tilde{b} \in B_8 \) (and that this can be done efficiently) and (ii) that the state we work with is always sufficiently close to the subspace \( S \) where (21) is valid. The next theorem and its subsequent claim show exactly that.

New Journal of Physics 13 (2011) 035019 (http://www.njp.org/)
Theorem 3.2 (density and efficiency in $B_8$ for a constant $k$). Fix $k > 4$, $k \neq 6$, and let $\tilde{U}$ be an encoded two-qubit quantum gate, and $\delta > 0$. Then there exists a braid $b \in B_8$, consisting of poly$(1/\delta)$ generators of $B_8$, such that for every $|\tilde{p}\rangle \in H_{8,k,1}$,

$$\| (\rho(\tilde{b}) - \tilde{U}) |\tilde{p}\rangle \| \leq \delta,$$

which can be found in poly$(1/\delta)$ time.

The proof of this theorem is given in section 4. Let us now see how, together with (21), it can be used to construct the appropriate braid $b \in B_{8n}$ in (20).

Let $U = U_L \cdots U_1$ be our quantum circuit, with $U_i$ being local two-qubit gates, and let $\epsilon > 0$ be an arbitrary constant. For every $U_i$, we use the theorem to construct a braid $\tilde{b}_i \in B_8$, with $\delta = \epsilon / L$, and extend it into a braid $b_i \in B_n$ by adding identity strands around it. Finally, $b$ is taken to be the product of these $b_i$s. We have

**Claim 3.1.** $\| U_L \cdot U_{L-1} \cdots U_1 |\alpha\rangle - \rho(b_L) \cdot \rho(b_{L-1}) \cdots \rho(b_1)|\alpha\rangle \| \leq \epsilon$.

**Proof.** The claim is easily proved by induction. Indeed, assume that

$$\| U_{i-1} \cdot U_i |\alpha\rangle - \rho(b_{i-1}) \cdots \rho(b_1)|\alpha\rangle \| \leq \frac{i-1}{L} \epsilon,$$

and define $|\beta\rangle \overset{\text{def}}{=} U_{i-1} \cdots U_1 |\alpha\rangle$. It is easy to verify that any encoded gate $U$ sends the subspace $S$ into itself and therefore $|\beta\rangle \in S$. Consequently,

$$\| U_i |\beta\rangle - \rho(b_i) |\beta\rangle \| = \| \tilde{U}_i |\tilde{\beta}\rangle - \rho(\tilde{b}_i) |\tilde{\beta}\rangle \| \leq \frac{1}{L} \epsilon,$$

where the first equality follows from the reduction in (21) and the second inequality follows from the way in which we constructed $\tilde{b}_i \in B_8$. Then, using the induction assumption together with the triangle inequality, we obtain

$$\| U_L \cdots U_1 |\alpha\rangle - \rho(b_L) \cdots \rho(b_1)|\alpha\rangle \| = \| U_L |\beta\rangle - \rho(b_L) |\beta\rangle + \rho(b_L) U_{L-1} \cdots U_1 |\alpha\rangle - \rho(b_L) \cdots \rho(b_1)|\alpha\rangle \|$$

$$\leq \| U_L |\beta\rangle - \rho(b_L) |\beta\rangle \| + \| U_{L-1} \cdots U_1 |\alpha\rangle - \rho(b_L) \cdots \rho(b_1)|\alpha\rangle \|$$

$$\leq \frac{\epsilon}{L} + \left( i - 1 \right) \frac{\epsilon}{L} = \frac{i \epsilon}{L}.$$

This shows that the braid $b = b_L \cdots b_1$ satisfies

$$\| (\alpha \rho(b) |\alpha\rangle = 0^\otimes n |U 0^\otimes n \rangle \| \leq \epsilon.$$

Taking $\epsilon = 1/10$ and using (15) then enables us to decide whether $\|0^\otimes n |U 0^\otimes n \rangle \| \leq 1/3$ or $\|0^\otimes n |U 0^\otimes n \rangle \| > 2/3$ by deciding whether $|V_{\text{int}}(t)| \leq \frac{\Delta}{2}$ or $|V_{\text{int}}(t)| \geq \frac{\Delta}{2}$, as required. Moreover, this procedure is efficient since by theorem 3.2, the number of braid generators that are needed to approximate every gate is of the order of poly$(1/\delta)$, and they can be found in time poly$(1/\delta)$. Therefore, overall, as $L = \text{poly}(n)$ and $\delta = \epsilon / L$, $b$ is made of poly$(n, 1/\epsilon)$ gates and can be found in time $T = \text{poly}(n, 1/\epsilon)$. This concludes the proof of theorem 3.1. □

In the next section, we will prove theorem 3.2—the $B_8$ density and efficiency theorem for constant $k$. 

New Journal of Physics 13 (2011) 035019 (http://www.njp.org/)
4. Proving the $B_8$ density and efficiency theorem

Our strategy for proving theorem 3.2 is to use the famous Solovay–Kitaev theorem [31], which shows that density implies efficiency. Specifically, we will first prove that the operators $\rho_1, \ldots, \rho_7$ can approximate any unitary operator on $H_{8,k,1}$. In other words, we will show that they generate a dense subgroup in $SU(H_{8,k,1})$. After such density is proved, the Solovay–Kitaev theorem tells us that it is possible to find a $\delta$-approximation of any unitary $U$ that consists of no more than $\text{poly}(\log(\delta^{-1}))$ generators in $\text{poly}(\log(\delta^{-1}))$ steps, thereby proving theorem 3.2.

The rest of this section is therefore devoted to proving that $\rho_1, \ldots, \rho_7$ generate a dense subset of $SU(H_{8,k,1})$. We begin by analyzing the structure of the subspace $H_{8,k,1}$ and the generators $\rho_1, \ldots, \rho_7$ of the $B_8$ path representation that act on it.
4.1. The structure of the generators in $H_{8,k,1}$

We begin by noting that for $k > 5$, $H_{8,k,1}$ consists of exactly 14 paths\(^3\), and hence it is a 14D space. These paths are labeled by the numbers 1, ..., 14 and are shown graphically in figure 8.

Let us now describe the structure of the generators on this space. In section 2.4, we saw that the generators break into 2D and 1D blocks when represented in the standard basis. Let us look at these blocks in some more detail.

First, by (11) and (14), $\Phi_i$ nullifies paths of the form $|\not\leftrightarrow\rangle$ and $|\not\rightarrow\rangle$, and as a result they become eigenvectors of $\rho_i$ with an eigenvalue $A^{-1}$.

The $2 \times 2$ blocks of $\Phi_i$ mix $|\not\rightarrow\rangle$ and $|\not\leftrightarrow\rangle$. By (12) and (13), these blocks are

\[
[\Phi_i]_{2 \times 2} = \begin{pmatrix}
\frac{\lambda_{z_i+1}}{\lambda_{z_i}} & \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}} \\
\frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}} & \frac{\lambda_{z_i-1}}{\lambda_{z_i}}
\end{pmatrix}.
\]  

(26)

This matrix has two eigenvalues, 0 and $2 \cos \theta$, and consequently (by (10)) the eigenvalues of $\rho_i$ in these blocks are $\{A^{-1}, -A^{-1} e^{-2i\theta}\}$—independent of $z_i$. In fact, it is not hard to see that all of the $\rho_i$ operators are equivalent, namely equal under a unitary change of basis. We further note that when $z_i = 1$, the off-diagonal terms vanish (because $\lambda_{z_i-1} = \lambda_0 = 0$) and the blocks become diagonal.

The $2 \times 2$ matrix that diagonalizes $[\Phi_i]_{2 \times 2}$ (and consequently $[\rho_i]_{2 \times 2}$) is

\[
M(z_i) \overset{def}{=} \frac{1}{\sqrt{\lambda_{z_i+1} + \lambda_{z_i-1}}} \begin{pmatrix}
\sqrt{\lambda_{z_i+1}} & -\sqrt{\lambda_{z_i-1}} \\
\sqrt{\lambda_{z_i-1}} & \sqrt{\lambda_{z_i+1}}
\end{pmatrix}.
\]  

(27)

Inside that subspace we have

\[
[\rho_i]_{2 \times 2} = A^{-1} \cdot M(z_i) \begin{pmatrix}
-e^{-2i\theta} & 0 \\
0 & 1
\end{pmatrix} M^\dagger(z_i).
\]  

(28)

Using the labeling of figure 8, we write down the block structure of the seven generators $\rho_1, \ldots, \rho_7$ in table 1. For each operator, the table lists the nontrivial blocks where $\Phi_i$ does not vanish. The 1D blocks correspond to the $z_i = 1$ case, and the 2D blocks correspond to the $z_i > 1$ case.

\begin{table}
| $\rho_i$ | Block Structure |
|--------|----------------|
| $\rho_1$ | (1) (3) (5) (7) (9) |
| $\rho_2$ | (1, 2) (3, 4) (5, 6) (7, 8) (9, 12) |
| $\rho_3$ | (1) (3) (6, 10) (8, 11) (12, 13) |
| $\rho_4$ | (1, 5) (2, 6) (3, 7) (4, 8) (13, 14) |
| $\rho_5$ | (1) (2) (7, 9) (8, 12) (11, 13) |
| $\rho_6$ | (1, 3) (2, 4) (5, 7) (6, 8) (10, 11) |
| $\rho_7$ | (1) (2) (5) (6) (10) |
\end{table}

\(^3\) For $k = 5$, there are actually only 13 paths, as path 14 is illegal (it gets out of the graph). Nevertheless, it is easy to see that the density proof still holds in that border case. For $k = 4$, we cannot prove density (see theorem 4.1), while for $k < 4$ the $H_{8,k,1}$ is too small to encode two qubits.

---

New Journal of Physics 13 (2011) 035019 (http://www.njp.org/)
4.2. Proving the density

We will now prove the density part of theorem 3.2. We will show that the seven operators \( \rho_i \) can approximate any special unitary matrix on \( H_{8, k, 1} \), provided that \( k > 4 \) and \( k \neq 6 \). As it is a 14D space, we are interested in matrices \( U \in SU(14) \).

We begin by considering the action of \( \rho_1 \) and \( \rho_2 \) on this subspace. From table 1, we see that these operators act nontrivially on the five \( 2 \times 2 \) blocks \( \{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}, \{9, 12\} \), while applying the trivial \( A^{-1} \) phase on the rest. In these blocks, the \( \rho_1 \) operator is represented by \((i, j)\), whereas the \( \rho_2 \) operator is represented by \((i, j)\). Additionally, the operators on all five blocks are equivalent, namely equal under a unitary change of basis. The following theorem assures us that in each such block we may approximate any SU(2) matrix.

**Theorem 4.1** (Jones [25]). If \( k > 4 \), and \( k \neq 6 \), then in each \( 2 \times 2 \) block, the group that is generated by \( \rho_1 \) and \( \rho_2 \) is dense in SU(2).

**Proof.** Since \( \rho_1 \) and \( \rho_2 \) are not in SU(2), we will look at their images under the canonical homomorphism \( U(2) \to SU(2) \), which takes \( W \in U(2) \to (\det W)^{-1/2}W \), and prove that these images form a dense set in SU(2). Then using the fact that [SU(2), SU(2)] = SU(2), it will follow that also \( \rho_1 \) and \( \rho_2 \) generate a dense set in SU(2).

Let \( G = \langle \rho_1, \rho_2 \rangle \) be the group that is generated by \( \rho_1, \rho_2 \). We first use the fact that \( G \) is infinite as long as \( k > 2 \) and \( k \neq 4, 6 \). This fact was proved by Jones in 1983 and appears in theorem 5.1 on page 262 of [25]. The proof uses the canonical homomorphism between SU(2) and SO(3) and the well-known classification of all the finite subgroups of SO(3).

To approximate any element in SU(2) to within an \( \epsilon \), we pick two matrices in \( g_1, g_2 \in G \) such that \( \|g_1 - g_2\| < \epsilon/3 \) (we can do that since \( G \) has an infinite number of elements and SU(2) is compact), and set \( g = g_1 g_2^{-1} \). Then \( \|g - 1\| < \epsilon/3 \), and consequently, if \( e^{\pm i\lambda} \) are the eigenvalues of \( g \), then \( |e^{\pm i\lambda} - 1| < \epsilon/3 \). In addition, \( g \) must be non-commuting with at least one of the matrices \( \rho_1 \) or \( \rho_2 \), which we shall denote by \( T \).

Let \( U \) be the diagonalizing matrix of \( g \): \( g = U^{-1} \text{diag}(e^{i\lambda}, e^{-i\lambda})U \), and define the two continuous families of matrices

\[
R(\phi) \overset{\text{def}}{=} U^{-1} \text{diag}(e^{i\phi}, e^{-i\phi})U, \tag{29}
\]

\[
S(\phi) \overset{\text{def}}{=} \sigma^{-1} R(\phi) \sigma. \tag{30}
\]

Then it is easy to see that any matrix \( V \in SU(2) \) can be presented as the product \( R(\alpha) S(\beta) R(\gamma) \) for a suitable choice of \( \alpha, \beta, \gamma \in \mathbb{R} \) (see, for example, [31]). But since \( |e^{i\phi} - 1| < \epsilon/3 \), any member in the families \( R(\cdot), S(\cdot) \) can be approximated by multiplications of \( g \) and \( \sigma \) up to a distance of \( \epsilon/3 \), and therefore the multiple \( R(\alpha) S(\beta) R(\gamma) \) can be approximated to within \( \epsilon \). \( \square \)

Next, consider what happens when we are also allowed to act with \( \rho_3 \). Looking at table 1, we see that the resulting operators are block-diagonal with respect to the blocks \( \{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}, \{9, 12\}, \{11\} \). Obviously, we can still approximate any SU(2) matrix in the \( 2 \times 2 \) blocks. The next lemma provides a way to increase the dimensionality of the space on which we have density, in the following way: suppose we have a set of operators that is dense

\[ \text{For } k = 5, \text{ we look at } SU(13) \text{ and ignore vector 14.} \]

New Journal of Physics 13 (2011) 035019 (http://www.njp.org/)
on SU(A) and on SU(B), for two orthogonal subspaces A, B. Suppose, in addition, that we have a unitary operator W on A ⊕ B that mixes these two spaces. Specifically, we demand that there exists a vector |u⟩ ∈ A such that W|u⟩ has some nonzero projection on B. We call such a transformation a bridge between A and B. Then using this bridge, together with the density on A and B, we have density in SU(A ⊕ B).

**Lemma 4.1** (the bridge lemma). Consider a linear space C that is a direct sum of two orthogonal subspaces A and B, and assume that dim B > dim A ≥ 1. Let W be a bridge transformation between A and B in the sense that was defined above. Then any U ∈ SU(C) can be approximated to an arbitrary precision using a finite sequence of transformations from SU(A), SU(B) and W. Consequently, the group generated by SU(A), SU(B) and W is dense in SU(C).

**Proof.** Given in section 6.

The bridge lemma implies that it is also possible to approximate any SU(3) matrix in the 3 × 3 blocks. As an example, consider the {5, 6, 10} block. From theorem 4.1, we already know that we are able to approximate any SU(2) transformation on the {5, 6} block, and, by definition, we also have density on the block {10} because it is 1D. We may therefore take the transformation ρ3 as a bridge between these two subspaces since, for example, it takes the path 10 into a superposition of 10 and 6. Lemma 4.1 therefore guarantees that together they can approximate every transformation in SU(3).

In the above reasoning, there are two small cavities that are worth mentioning, since they will appear in the rest of the proof. Firstly, the mixing transformation ρ3 is in U(3) rather than in SU(3). This, however, is not a real problem, as we can always consider the transformation \( \tilde{\rho}_3 \) def = cρ3 with c some phase that fixes \( \tilde{\rho}_3 \) in SU(3). Then \( \langle \rho_1, \rho_2, \tilde{\rho}_3 \rangle \) is dense in SU(3), and since [SU(N), SU(N)] = SU(N) then also \( \{\langle \rho_1, \rho_2, \tilde{\rho}_3 \rangle, \langle \rho_1, \rho_2, \tilde{\rho}_3 \rangle\} \) is dense in SU(3). But the last group is equal to \( \{\langle \rho_1, \rho_2, \rho_3 \rangle, \langle \rho_1, \rho_2, \rho_3 \rangle\} \) since the group bracket cancels out the phase c and therefore also \( \langle \rho_1, \rho_2, \rho_3 \rangle \) is dense in SU(3).

Secondly, we know we can approximate any transformation in SU(2), while lemma 4.1 assumes that we can get any transformation in SU(2) precisely. But since the approximation is made of a finite product of operators, all of which can be approximated as accurately as desired by \( \rho_1, \rho_2, \rho_3 \), it follows that we can also approximate any transformation in SU(3) to any desired accuracy.

Naturally, the next step is to consider what happens when we are also allowed to act with \( \rho_4 \). From table 1, we see that the resulting transformations will be invariant under the subspaces \{1, 2, 5, 6, 10\}, \{3, 4, 7, 8, 11\}, \{9, 12, 13, 14\}, which together make up the entire 14D subspace. We can use lemma 4.1 again to learn that we can approximate any SU(4) transformation in the {9, 12, 13, 14} block. But what about the two other, 5D blocks? There we cannot use lemma 4.1 directly. To understand why this is so, consider, for example, the subspace \{1, 2, 5, 6, 10\}. We know that using \( \rho_1, \rho_2, \rho_3 \) we can approximate any SU(2) transformation on the {1, 2} block and any SU(3) transformation on the {5, 6, 10} block. We also know that \( \rho_4 \) bridges these two blocks. However, to use lemma 4.1, we must be able to approximate the SU(2) transformations independently of the SU(3) transformation. In other words, we must be able to approximate an SU(2) transformation on the subspace \{1, 2\} while leaving the subspace \{5, 6, 10\} invariant and vice versa. But this is not a priori true since the
transformations on \{1, 2\} are generated by some sequence of the operators \( \rho_1, \rho_2, \rho_3 \), which \textit{simultaneously} generates some transformation on \{5, 6, 10\}. Luckily, we can use the fact that the dimensionality of the two subspaces is different in order to prove that such decoupling is possible:

**Lemma 4.2** (the decoupling lemma). Let \( G \) be an infinite discrete group, and let \( A, B \) be two finite Linear spaces with different dimensionality. Let \( \tau_a \) and \( \tau_b \) be two homomorphisms of \( G \) into \( SU(A) \) and \( SU(B) \), respectively, and assume that \( \tau_a(G) \) is dense in \( SU(A) \) and \( \tau_b(G) \) is dense in \( SU(B) \). Then for any \( U \in SU(A) \) there exists a series \( \{ \sigma_n \} \) in \( G \) such that

\[
\tau_a(\sigma_n) \rightarrow U, \\
\tau_b(\sigma_n) \rightarrow 1,
\]

and vice versa.

**Proof.** Given in section 6.

It is therefore clear that we are able to approximate any SU(5) transformation on the \{1, 2, 5, 6, 10\} and \{3, 4, 6, 7, 8, 11\} blocks. Using \( \rho_5 \), we can now mix the \{3, 4, 7, 8, 11\} subspace with the \{9, 12, 13, 14\} subspace, and using the fact that their dimensionality is different, together with lemmas 4.1 and 4.2—we are assured that we can approximate any SU(9) transformation on the combined 9D subspace.

Finally, by using \( \rho_6 \), we mix the 5D block \{1, 2, 5, 6, 10\} with the 9D block from above—thereby approximating any transformation in SU(14). This completes the density proof.

5. BQP-hardness for \( k = \text{poly}(n) \)

In this section, we prove the central result of this paper. We prove a stronger version of theorem 3.1, in which \( k \) is allowed to depend polynomially on \( n \):

**Theorem 5.1** (BQP-hardness for a \( k = \text{poly}(n) \)). Let \( p(\cdot) \) be some polynomial, and let \( b \in B_n \) be a braid with \( m = \text{poly}(n) \) crossings, and \( b^{pl} \) its plat closure. Finally, let \( V_{b^{pl}}(t) \) be its Jones polynomial at \( t = \exp(2\pi i / k) \) with \( k = p(n) \), and define \( \Delta \) as (16), so that \( |\Delta| = 2(\cos(\pi / k))^{n/2-1} \). Then, given the promise that either \( |V_{b^{pl}}(t)| \leq \frac{1}{10} |\Delta| \) or \( |V_{b^{pl}}(t)| \geq \frac{9}{10} |\Delta| \), it is BQP-hard to decide between the two.

Looking at the proof of theorem 3.1, it is readily evident that the only obstacle that prevents it to prove also this case is the fact that we do not know how theorem 3.2 depends on \( k \). Specifically, we do not know the dependence of the running time as well as the length of the resultant braid on \( k \).

It is therefore easy to see that the following stronger version of theorem 3.1, in which both running time and braid length are polynomial in \( k \), would be enough to prove theorem 5.1.

**Theorem 5.2** (\( B_S \) for \( k = \text{poly}(n) \)). Let \( k > 4 \), \( k \neq 6 \), and let \( \tilde{U} \) be an encoded two-qubit quantum gate, and \( \delta > 0 \). Then there exists a braid \( \tilde{b} \in B_{8k} \), consisting of \( \text{poly}(1/\delta, k) \) generators of \( B_{8k} \), such that for every \( |\tilde{p}\rangle \in H_{8,k,1} \),

\[
\| (\rho(\tilde{b}) - \tilde{U})|\tilde{p}\rangle\| \leq \delta,
\]

which can be found in \( \text{poly}(1/\delta, k) \) time.
Indeed it is very easy to see that the very same proof of theorem 3.1, but with theorem 3.2 replaced by the above theorem with $k = \text{poly}(n)$, proves theorem 5.1. The rest of the section would therefore be devoted to proving theorem 5.2.

**Proof.** As in the proof of theorem 3.2, our main mathematical tool is the Solovay–Kitaev theorem. We would like to use the fact that for $k > 6$, the generators $\rho_1, \ldots, \rho_7$ form a dense subset of $\text{SU}(14)$, and then use the Solovay–Kitaev algorithm to efficiently generate a $\delta$-approximation for any given gate.

There is a problem, however, with this simplistic approach. The Solovay–Kitaev algorithm contains an initial step, where an $\epsilon$-net is constructed; this is a finite set of operators that is generated by $\rho_1, \ldots, \rho_7$ and has the property that every operator in $\text{SU}(14)$ is closer than $\epsilon$ to at least one of the elements of the net. $\epsilon$ is a finite constant that is unrelated to the target accuracy $\delta$ and whose actual value is of the order of $10^{-2}$ (see, for example, [16]). The existence of such a net is guaranteed since we know that $\rho_1, \ldots, \rho_7$ generate a dense set in $\text{SU}(14)$; its construction time, however, depends on the generators. For a fixed set of generators, this is not a problem; the construction time becomes a constant.

However, the situation becomes more tricky when $k$ is no longer fixed. The operators $\rho_1, \ldots, \rho_7$ become $k$ dependent, and we can no longer treat the $\epsilon$-net construction as a constant step. Its complexity must be taken into account. The question is therefore whether we can still guarantee that the overall computational cost is polynomial in $k$ and in $\log(\delta^{-1})$? The answer is positive; this is what will be proved in this section. The main observation is that the generators $\rho_1, \ldots, \rho_7$ do not behave randomly, but rather converge nicely to a $k = \infty$ limit. The idea of how to make use of this fact was explained in the introduction; essentially, the idea is that as $k$ becomes larger and larger, the generators $\rho_1, \ldots, \rho_7$ do not behave randomly, but rather converge nicely to a $k = \infty$ limit. Their dependence on $k$ is simple enough so that we can easily approximate $k_0$ generators by products of $k$ generators for $k$ that are multplicities of $k_0$. Therefore, we can construct an $\epsilon$-net at some large enough yet constant $k_0$, and approximate each element in the net by products of high-$k$ generators, thereby efficiently obtaining an $\epsilon$-net for high $k$s. We now provide the details.

Let us therefore begin by considering the $k \rightarrow \infty$ limit of the $\rho_i$ operators. From section 4.1, we recall that in the standard basis these operators decompose into the $1 \times 1$ or $2 \times 2$ blocks. The diagonalizing matrix of the $2 \times 2$ blocks is $M_k(z)$, given by (27), and the eigenvalues are $z$ independent, given by $\{A^{-1}, -A^{-1} e^{-2\theta}\}$ (see (28)). Here and in what follows, we have explicitly added the subscript $k$ to $M(z)$ to indicate its dependence on $k$.

Note that up to an overall factor of $A^{-1}$, the eigenvalues of the generators are $\{1, -\exp(-2i\pi/k)\}$. So we can express the eigenvalues of low $k$s as products of eigenvalues of high $k$s. However, this is still not enough, as we want the low-$k$ generators themselves to be approximated by products of high-$k$ generators. Luckily, we note that the diagonalizing matrix $M_k(z)$ converges nicely to $M_\infty(z)$ as $k \rightarrow \infty$.

$$M_\infty(z) \equiv \lim_{k \rightarrow \infty} M_k(z) = \frac{1}{2z} \begin{pmatrix} \sqrt{z+1} & -\sqrt{z-1} \\ \sqrt{z-1} & \sqrt{z+1} \end{pmatrix}.$$  \hspace{1cm} (33)

We can therefore define auxiliary low-$k$ generators by taking the $k \rightarrow \infty$ diagonalizing matrix $M_\infty(z)$ together with the eigenvalues at some low $k_0$. Then for high enough $k$s, for which $M_k(z)$ is close enough to $M_\infty(z)$, we can approximate the auxiliary generators by powers of $\rho_i(k)$.
Let remains valid since it only relies on the eigenvalues of the generating operators and on the fact that for \( z > 1 \), \( M_k(z) \) mixes the two standard basis vectors. We will thus generate an \( \epsilon \)-net from \( \{ \hat{\rho}_i \} \) and use it to generate the \( \epsilon \)-net of \( \{ \rho_i \} \) for high \( k \). This is proved in the following lemma:

**Lemma 5.1.** Let \( \hat{E} \) be an \( \epsilon/2 \)-net, generated from \( \{ \hat{\rho}_i \} \), and assume without loss of generality that each element in \( \hat{E} \) is a group commutator (this is possible since SU(14) is a simple Lie group and therefore [SU(14), SU(14)] = SU(14)). Then for large enough \( k \), it is possible to generate an \( \epsilon \)-net \( E_k \) by replacing every occurrence of \( \hat{\rho}_i \) in \( \hat{E} \) by \( (\rho_i)^{2m} \), where \( \{ \rho_i \} \) are the generators at \( k \), and \( m = \mathcal{O}(k) \).

**Proof.** Let \( d \) be the maximal number of generators that are needed to construct an element in \( \hat{E} \). We wish to be able to approximate any \( \hat{\rho}_i \) up to at least \( \epsilon/2d \) using \( \rho_i \). The first thing we take care of is that \( M_k(z) \) will be close enough to \( M_\infty(z) \). We therefore pick an integer \( K_1 \) such that for any \( k > K_1 \), \( \| M_k(z) - M_\infty(z) \| < \epsilon/(6d) \).

Next, we must find a \( K_2 \) such that for any \( k > K_2 \), the eigenvalues of \( (\rho_i)^{2m} \) will be close enough to \( \hat{\rho}_i \), for some yet to be determined \( m \). This is more conveniently done by defining

\[
P_i \overset{\text{def}}{=} A(k_0) \hat{\rho}_i, \quad Q_i \overset{\text{def}}{=} [A(k) \rho_i]^2,
\]

and approximating the operators \( P_i \) with \( Q_i \). In the end, the factors \( A(k_0) \) and \( A(k) \) will cancel out when we plug these operators to the group commutator of each element in \( \hat{E} \). The logic behind these definitions is that these factors cause one of the eigenvalues of both \( P_i \) and \( Q_i \) to be exactly one (see (28)) and therefore we only have to match the remaining eigenvalues. Indeed, the nontrivial eigenvalue of \( P_i \) is \( \exp(-2\pi i/k_0) = \exp(-i\pi(2 + k_0)/k_0) \), whereas the nontrivial eigenvalue of \( Q_i \) is \( \exp(-4\pi i/k) \). We therefore define

\[
m \overset{\text{def}}{=} \left\lfloor \frac{(2 + k_0)/k_0}{4/k} \right\rfloor,
\]

and let \( K_2 \) be such that for every \( k > K_2 \)

\[
|e^{-i\pi(2+k_0)/k_0} - e^{-4m\pi i/k}| < \epsilon/(6d).
\]

It is easy to see that it is enough to choose \( K_2 \) (larger than \( k_0 \)) for which \( |\exp(-4\pi i/K_2) - 1| < \epsilon/(6d) \).

Assume then that \( k > \max(K_1, K_2) \) and let us estimate the distance between \( P_i \) and \( (Q_i)^m \). This is the maximal distance between the corresponding blocks in the standard basis. In the
$1 \times 1$ blocks both operators have an eigenvalue $1$ and therefore the distance is zero. In the $2 \times 2$ blocks, we have

$$[P]_{2 \times 2} = M_{\infty}^{-1}(z) \begin{pmatrix} e^{-i\pi(2k_0)/k_0} & 0 \\ 0 & 1 \end{pmatrix} M_{\infty}(z), \quad (38)$$

$$[Q^m]_{2 \times 2} = V^{-1}(z) \begin{pmatrix} e^{-4m\pi i/k} & 0 \\ 0 & 1 \end{pmatrix} V(z), \quad (39)$$

and consequently

$$\|P - Q^m\|_{2 \times 2} \leq \|M_{\infty}^{-1}(z) - M_{\infty}^{-1}(z)\| + \left| e^{-i\pi(2k_0)/k_0} - e^{-4m\pi i/k} \right| + \|M_{\infty}(z) - M(z)\| \leq \epsilon/(2d). \quad (40)$$

Let us now return to the $\hat{E}$-net and create the $E_k$-net. Any element in $\hat{E}$ is a commutator of products of $\hat{\rho}_i$ and therefore remains unchanged if we replace $\hat{\rho}_i \to P_i$, because the phase factors cancel out in the commutator. The distance of this product from a product in which we replace $P_i \to (Q_i)^m$ is smaller than $\epsilon/2$ since $\|P_i - (Q_i)^m\| < \epsilon/(2d)$ and we have at most $d$ terms in the product. The $(Q_i)^m$’s product is unchanged upon the replacement $(Q_i)^m \to (\rho_i)^{2m}$ (again, the phase factors cancel out), and this results in the $E_k$ net. Hence any element in $\hat{E}$ can be efficiently approximated up to a distance $\epsilon/2$ by an element of $E_k$. It follows that $E_k$ is an $\epsilon$-net.

It follows that we can create an $\epsilon$-net from the operators $\rho_i$, and because $m < k$, the number of steps that are needed to create this net is bounded by $\text{poly}(k)$. The next step would be the application of the Solovay–Kitaev algorithm to approximate any transformation $U \in \text{SU}(14)$ up to an error $\delta$—and so the overall computational cost is bounded by $\text{poly}(k, 1/\delta)$, as required.

6. General tools for proving universality: the bridge lemma and the decoupling lemma

We provide here the proofs of the bridge lemma and the decoupling lemma. For convenience, we restate the lemmas.

6.1. The bridge lemma

Let us start by redefining a bridge transformation:

**Definition 6.1.** Given two orthogonal subspaces $A$ and $B$, a unitary operator $W$ on $A \oplus B$ is said to be a bridge between $A$ and $B$ if there exists a vector $|u\rangle \in A$ such that $W|u\rangle$ has some nonzero projection on $B$. Note that this notion is symmetric, since the existence of such a vector implies the existence of $|u\rangle \in B$ such that $W|u\rangle$ has a nonzero projection on $A$, by unitarity. We sometimes say that such a transformation mixes the two subspaces.

We restate the bridge lemma:

**Lemma 4.1** (the bridge lemma). Consider a linear space $C$ that is a direct sum of two orthogonal subspaces $A$ and $B$, and assume that $\dim B > \dim A \geq 1$. Let $W$ be a bridge transformation between $A$ and $B$. Then any $U \in \text{SU}(C)$ can be approximated to an arbitrary precision using a finite sequence of transformations from $\text{SU}(A)$, $\text{SU}(B)$ and $W$. Consequently, the group generated by $\text{SU}(A)$, $\text{SU}(B)$ and $W$ is dense in $\text{SU}(C)$.

New Journal of Physics 13 (2011) 035019 (http://www.njp.org/)
To prove lemma 4.1, we first need to prove the following two lemmas:

**Lemma 6.1.** Consider a linear space $C$ that is a direct sum of two subspaces $A$ and $B$ such that $\dim B > \dim A \geq 1$, and let $W \in SU(C)$ be a bridge transformation that mixes the two subspaces. Then for every pair of normalized vectors $|\psi\rangle, |\phi\rangle \in C$, we can approximate a transformation $T_{\psi \rightarrow \phi} \in SU(C)$ such that $T_{\psi \rightarrow \phi}|\psi\rangle = |\phi\rangle$, to any desired accuracy using a finite product of transformations from $SU(A)$, $SU(B)$ and $W$.

**Proof.** Instead of approximating the transformation $T_{\psi \rightarrow \phi}$ for any two vectors $|\psi\rangle, |\phi\rangle$, we will approximate a transformation $T_{\phi}$ that transforms a particular vector $|v^*\rangle$ to an arbitrary vector $|\psi\rangle$. Then, $T_{\psi \rightarrow \phi} = T_{\phi}T_{\psi}^{-1}$.

We begin by finding a vector $|v^*\rangle \in B$ for which $W|v^*\rangle \in B$. Such a vector must exist since $\dim B > \dim A$. Indeed, let $|v_1\rangle, \ldots, |v_n\rangle$ be a basis of $B$. Then $W|v_i\rangle = \alpha_i|u_i^*\rangle + \beta_i|v_i'\rangle$, with $|u_i^*\rangle \in A$ and $|v_i'\rangle \in B$. Then since $\dim B > \dim A$, the $|u_i^*\rangle$ vectors are linearly dependent and we can find a nontrivial linear combination such that $\sum \alpha_i|u_i^*\rangle = 0$. Then the vector $|v^*\rangle \equiv \sum c_i|v_i\rangle$ is in $B$ and $W|v^*\rangle$ has no projection on $A$.

Next, we pick a vector $|u^*\rangle \in A$ for which $W|u^*\rangle$ has some projection on $B$. Such a vector must exist since $W$ is a bridge transformation between $A$ and $B$. We write $W|u^*\rangle = a|u\rangle + b|v\rangle$ with $|u\rangle \in A$, $|v\rangle \in B$ and $|a|^2 + |b|^2 = 1$. If $a \neq 0$, we find a transformation $U \in SU(A)$ that takes $|u\rangle$ to $|u^*\rangle$, and define $\tilde{W} = UW$; otherwise, we set $\tilde{W} = W$. We have thus constructed a transformation $\tilde{W}$ for which $\tilde{W}|v^*\rangle \in B$ and $\tilde{W}|u^*\rangle = a|u^*\rangle + b|v\rangle$ for some $0 \leq |a| < 1$.

Now let $|\psi\rangle = \alpha|u_0\rangle + \beta|v_0\rangle$ be an arbitrary vector, with $|u_0\rangle \in A$ and $|v_0\rangle \in B$. We will now apply a series of unitary operations that will take $|\psi\rangle$ closer and closer to $|v^*\rangle$. We start by moving $|u_0\rangle$ to $|u^*\rangle$ and $|v_0\rangle$ to $|v^*\rangle$ using transformations from $SU(A)$ and $SU(B)$, respectively, yielding the vector $|\psi_1\rangle = \alpha|u^*\rangle + \beta|v^*\rangle$. Using $\tilde{W}$, we obtain

$$|\psi_1'\rangle = \tilde{W}|\psi_1\rangle = \alpha a|u^*\rangle + \alpha b|v\rangle + \tilde{W}|v^*\rangle.$$  

As $|v\rangle$, $\tilde{W}|v^*\rangle \in B$, we can now apply a transformation from $SU(B)$ that takes $\alpha b|v\rangle + \tilde{W}|v^*\rangle$ to $c_2|v^*\rangle$. Here, $c_2$ is the norm of $\alpha b|v\rangle + \tilde{W}|v^*\rangle$. We obtain

$$|\psi_2\rangle = \alpha a|u^*\rangle + c_2|v^*\rangle.$$  

Comparing $|\psi_2\rangle$ to $|\psi_1\rangle = \alpha|u^*\rangle + \beta|v^*\rangle$, we see that we managed to move some of the weight from $|u^*\rangle$ to $|v^*\rangle$ because $|a| < 1$ and both vectors are normalized (we only used unitary transformations).

We now iterate this process: we get $|\psi_2\rangle$ by applying the $W_2$ transformation on $|\psi_2\rangle$, and obtain $|\psi_3\rangle$ by moving the $B$ part of $|\psi_2\rangle$ to $|v^*\rangle$. After $n$ such iterations, we obtain

$$|\psi_n\rangle = \alpha a^{n-1}|u^*\rangle + c_n|v^*\rangle,$$

and since $|a| < 1$, it is obvious that we exponentially converge to $|v^*\rangle$, and in particular we can approximate $T_{\phi}$ (and hence $T_{\psi \rightarrow \phi}$) to any desired accuracy using a finite number of transformations.

To continue, we need to be able to move a vector from subspace $A$ to subspace $B$ without affecting the rest of the vectors in subspace $A$. The following lemma guarantees that this is possible.
Lemma 6.2. Under the same conditions of lemma 6.1, let \( \{|u_1\}, \ldots, |u_n\}\) be an orthonormal basis of \(A\) and \(\{|v_1\}, \ldots, |v_m\}\) be an orthonormal basis of \(B\). Then using a finite product of transformations from \(\text{SU}(A)\), \(\text{SU}(B)\) and the bridge \(W\) between \(A\) and \(B\), it is possible to approximate to any accuracy a transformation \(T\) that moves \(|u_1\) to \(|v_1\), while leaving the vectors \(|u_2\), \ldots, |u_n\) unchanged.

Proof. For \(\dim A = 1\), the problem is trivial since we can simply use lemma 6.1. Assume then that \(\dim A > 1\), and define the subspaces \(A' \overset{\text{def}}{=} \text{span}\{|u_1\} \ldots |u_{n-1}\}\) and \(B' \overset{\text{def}}{=} \text{span}\{|v_1\} \ldots |v_m\}\). By lemma 6.1, we can approximate a transformation \(\tilde{T}\) that takes \(|u_n\) to \(|v_1\). Consider now all of the operators of the form \(W = \tilde{T}^{-1}UV'\tilde{T}\) with \(U \in \text{SU}(A)\) and \(V' \in \text{SU}(B')\). Clearly, \(W\) takes \(|u_n\) to itself—and therefore leaves invariant the subspace \(A' \oplus B\). We claim that there is at least one such transformation, \(W^{(1)}\), that also mixes the subspaces \(A'\) and \(B\). If this is indeed the case, then we can repeat the argument for the subspaces \(A'\) and \(B\), which together with the particular transformation \(W^{(1)}\) satisfy the conditions of lemma 6.1. Consequently, we find a transformation \(W^{(2)}\) that takes \(|u_{n-1}\) to \(|v_1\) while leaving \(|u_n\) unchanged and mixing the space that is spanned by \(|u_1\), \ldots, |u_{n-2}\) with \(B\). Repeating this again and again, we are left in the end with a transformation \(W^{(n)}\) that transforms \(|u_1\) to \(|v_1\) and is the identity over \(|u_2\), \ldots, |u_n\). Since this recursion has only \(n\) steps, it is clear that at any step we can approximate the mixing transformation \(W^{(i)}\) to any desired accuracy using a finite product of operators from \(\text{SU}(A)\), \(\text{SU}(B)\) and \(W\).

Let us now see why \(W^{(1)}\) must exist. Indeed, if no such transformation exists, then for every two operators \(U \in \text{SU}(A)\) and \(V' \in \text{SU}(B')\) there is no mixing between the subspaces \(A'\) and \(B\), and therefore there must exist operators \(U' \in \text{SU}(A')\) and \(V \in \text{SU}(B)\) such that

\[
\tilde{T}^{-1}UV'\tilde{T} = U'V.
\] (44)

Then, for every \(|u\) \in \(A\) and \(|v\) \in \(B\),

\[
\langle u|\tilde{T}^{-1}UV'\tilde{T}v\rangle = \langle u|U'Vv\rangle = 0,
\] (45)

which implies that for every \(U \in \text{SU}(A)\) and \(V' \in \text{SU}(B')\),

\[
\langle u\tilde{T}U|V'\tilde{T}v\rangle = 0.
\] (46)

Notice that this equation holds also when we take one of the operators, \(U\) or \(V'\), to be the identity. We will use this to show that \(\tilde{T}A = A\)—in contradiction with the fact that \(\tilde{T}|u_1\) = \(|v_1\).

We first deduce that \(\tilde{T}A' \subset A\). To do this we show that for all \(|u\) \in \(A'\), \(\tilde{T}|u\) has no projection on \(B\). We already know \(\tilde{T}|u\) has zero projection on \(|v_1\) (since \(|u_1\) moves to \(|v_1\)), so it suffices to show that \(\tilde{T}|u\) has no projection on \(B'\).

Indeed, by (46) it would suffice to show that \(V'\tilde{T}v)\) can be made to be an arbitrary vector in \(B'\). This follows from the following reasoning. By the same argument as in the proof of lemma 6.1, there must be a vector \(|v^*\) \in \(B\) such that \(\tilde{T}|v^*\) \in \(B\). Moreover, \(\tilde{T}|v^*\) must be in \(B'\) since

\[
\langle v_1|\tilde{T}v^*\rangle = \langle v_1\tilde{T}^{-1}|v^*\rangle = \langle u_1|v^*\rangle = 0.
\] (47)

Since \(\tilde{T}|v^*\) \in \(B'\), using an arbitrary \(V' \in \text{SU}(B')\), \(V'\tilde{T}|v^*\) can be made to be any vector in \(B'\).

Now pick any \(|u^*\) \in \(A'\). Then \(\tilde{T}|u^*\) \in \(A\), and therefore with an arbitrary transformation \(U \in \text{SU}(A)\), \(UT|u^*\) can be made to be an arbitrary vector in \(A\). But since for every \(|v\) \in \(B\) we have \((u^*\tilde{T}U|V\tilde{T}v) = 0\) then \(\tilde{T}|v\) has no projection on \(A\) and, consequently, \(\tilde{T}B = B\). But then, since \(\tilde{T}\) is unitary, it follows that \(\tilde{T}A = A\)—which is the contradiction we were seeking. \(\square\)
Having proved the last two lemmas, we are now in a position to prove lemma 4.1:

**Proof.** Let \( \{|u_1\rangle, \ldots, |u_n\rangle\} \) be an orthonormal basis of \( A \) and similarly \( \{|v_1\rangle, \ldots, |v_n\rangle\} \) an orthonormal basis of \( B \). We define the following sequence of subspaces,

\[
B_i = B_{i-1} \oplus |u_i\rangle
\]

for \( i = 1 \cdots n \), where \( B_0 = B \).

Let us show how to approximate an arbitrary \( U \in \text{SU}(B_1) \), given \( \text{SU}(B) \), \( \text{SU}(A) \) and \( W \). From lemma 6.2, we can use \( \text{SU}(B) \), \( \text{SU}(A) \) and \( W \) to approximate a transformation \( T \) that takes \( |u_1\rangle \) to \( |v_1\rangle \) while leaving the rest of the vectors in \( A \) intact. Therefore \( T \in \text{SU}(B_1) \). Now pick an eigenvector \( |\psi\rangle \in B_1 \) of \( U \) with an eigenvalue \( e^{i\theta} \). Using lemma 6.1 with respect to the subspaces \( \text{span}|u_1\rangle, B \) and the mixing transformation \( T \), we can approximate a transformation \( W_\psi \in \text{SU}(B_1) \) that takes \( |\psi\rangle \) to \( |u_1\rangle \). We first show how to approximate the transformation \( U_1 = W_\psi U W_\psi^{-1} \).

We note that \( U_1 \) has \( |u_1\rangle \) as an eigenvector with an eigenvalue \( e^{i\theta} \). Consequently, \( U_1 \) leaves the subspace \( B \) invariant. Let \( V_1 \) be the transformation in \( \text{SU}(B) \) that satisfies \( V_1|v_1\rangle = e^{i\theta}|v_1\rangle \), \( V_1|v_2\rangle = e^{-i\theta}|v_2\rangle \), and leaves the rest of the basis vectors unchanged. Recalling that \( T \) takes \( |u_1\rangle \) to \( |v_1\rangle \), we see that \( T^{-1}V_1T \) has \( |u_1\rangle \) as an eigenvector with eigenvalue \( e^{i\theta} \) and leaves the subspace \( B \) invariant. So the only difference between \( U_1 \) and \( T^{-1}V_1T \) is some transformation \( V_2 \in \text{SU}(B) \), and therefore

\[
U_1 = V_2T^{-1}V_1T,
\]

and consequently,

\[
U = W_\psi^{-1}V_2T^{-1}V_1TW_\psi.
\]

Now that we have generated all transformations in \( \text{SU}(B_1) \), we can generate \( \text{SU}(B_2) \) using the very same procedure—except that now \( B_1 \) plays the role of \( B \) and \( |u_2\rangle \) plays the role of \( |u_1\rangle \). By the same method, we can work our way up to \( \text{SU}(B_n) = \text{SU}(C) \). \( \square \)

### 6.2. The decoupling lemma

**Lemma 4.2** (the decoupling lemma). *Let \( G \) be an infinite discrete group, and let \( A, B \) be two finite linear spaces with different dimensionality. Let \( \tau_a \) and \( \tau_b \) be two homomorphisms of \( G \) into \( \text{SU}(A) \) and \( \text{SU}(B) \), respectively, and assume that \( \tau_a(G) \) is dense in \( \text{SU}(A) \) and \( \tau_b(G) \) is dense in \( \text{SU}(B) \). Then for any \( U \in \text{SU}(A) \) there exist a series \( \{\sigma_n\} \) in \( G \) such that

\[
\tau_a(\sigma_n) \rightarrow U,
\]

\[
\tau_b(\sigma_n) \rightarrow \mathbb{1},
\]

and vice versa.*

**Proof.** We define two subgroups \( H_a \triangleleft \text{SU}(A) \) and \( H_b \triangleleft \text{SU}(B) \) by

\[
H_a \overset{\text{def}}{=} \left\{ U \in \text{SU}(A) \mid \exists \{\sigma_n\} \text{ s.t. } \tau_a(\sigma_n) \rightarrow U \right\},
\]

\[
H_b \overset{\text{def}}{=} \left\{ V \in \text{SU}(B) \mid \exists \{\sigma_n\} \text{ s.t. } \tau_a(\sigma_n) \rightarrow \mathbb{1} \right\}.
\]

The theorem will be proved once we show that \( H_a = \text{SU}(A) \) and \( H_b = \text{SU}(B) \).
For each coset \( M \) follows:

\[
\begin{align*}
\lim_{n \to \infty} \tau_b(\sigma_n^{(k)}) &= U_k, \\
\lim_{n \to \infty} \tau_b(\sigma_n^{(k)}) &= 1.
\end{align*}
\]

Without loss of generality, we may choose the series such that for every \( n, k \),

\[
\| \tau_b(\sigma_n^{(k)}) - U_k \| < 1/n, \quad \| \tau_b(\sigma_n^{(k)}) - 1 \| < 1/n.
\]

Then the series \( \tau_b(\sigma_n^{(k)}) \to U \), and we are assured that \( \tau_b(\sigma_n^{(k)}) \to 1 \).

Now, any nontrivial normal subgroup of \( SU(N) \) must be finite\(^5\). Therefore, if we show that \( H_a \) and \( H_b \) are infinite, it will follow that \( H_a = SU(A) \) and \( H_b = SU(B) \). To do that, we first show that there is an isomorphism of groups \( M \) (a 1–1 and onto mapping that preserves the action of the group) between the coset groups \( SU(A)/H_A \) and \( SU(B)/H_B \). We define \( M \) as follows: \( M(UH_A) = VH_B \) if there exists a series \( \{\sigma_n\} \) such that

\[
\tau_a(\sigma_n) \to U, \quad \tau_b(\sigma_n) \to V.
\]

We first need to show that this function is well defined for all cosets \( UH_A \). This follows from:

- For each coset \( UH_A \), there exists at least one series that satisfies the requirements of the definition of \( M \). Indeed, pick a series \( \{\sigma_n\} \) such that \( \tau_a(\sigma_n) \to U \). Then the series \( \tau_b(\sigma_n) \) in \( SU(B) \) must have a limiting point since \( SU(B) \) is compact. Therefore there exists a sub-series \( \{\sigma_n\} \) such that \( \tau_a(\sigma_n) \to U \) and \( \tau_b(\sigma_n) \to V \). We have \( M(UH_A) = VH_B \).

- \( M(UH_A) \) is defined uniquely. Indeed, assume there exist two series \( \{\sigma_n^{(1)}\} \) and \( \{\sigma_n^{(2)}\} \) such that

\[
\begin{align*}
\tau_a(\sigma_n^{(1)}) &\to U_1, \quad \tau_b(\sigma_n^{(1)}) \to V_1, \\
\tau_a(\sigma_n^{(2)}) &\to U_2, \quad \tau_b(\sigma_n^{(2)}) \to V_2,
\end{align*}
\]

with \( U_1 \) and \( U_2 \) in the coset \( UH_A \). We will show that \( V_1 \) and \( V_2 \) must be in the same coset of \( H_B \). Denote \( \Delta \equiv U_1U_2^{-1} \). Since \( H_a \) is normal, \( \Delta \) is in \( H_a \) and we may therefore find a series \( \{\sigma_n^{(3)}\} \) such that

\[
\begin{align*}
\tau_a(\sigma_n^{(3)}) &\to U_1U_2^{-1}, \quad \tau_b(\sigma_n^{(3)}) \to 1.
\end{align*}
\]

Then looking at the series \( \sigma_n^{(4)} \equiv (\sigma_n^{(1)})^{-1}\sigma_n^{(3)}\sigma_n^{(2)} \), we find that

\[
\begin{align*}
\tau_a(\sigma_n^{(4)}) &\to U_1^{-1}U_1U_2^{-1}U_2 = 1, \quad \tau_b(\sigma_n^{(4)}) \to V_1^{-1}V_2.
\end{align*}
\]

Therefore \( V_1^{-1}V_2 \in H_a \), and so \( V_1H_B = V_2H_B \).

It remains for us to show that \( M \) is 1–1, onto, and preserves the action of the group. The onto part follows if we start with a series that converges to some \( V \) in \( VH_B \) and apply the same reasoning as in the first item above. The 1–1 part follows if we apply the same reasoning as in the second point above, starting with the \( V_1, V_2 \) in the same coset instead of the opposite direction. The fact that \( M \) is a homomorphism follows from the fact that \( \tau \) is a representation.

---

\(^5\) This follows from the fact that the quotient group \( SU(N)/Z(SU(N)) \) is a simple group, and \( Z(SU(N)) \)—the center of \( SU(N) \)—is finite. See, for example, theorem 11.26 on page 108 of [23].

*New Journal of Physics* **13** (2011) 035019 (http://www.njp.org/)
Recall now that $H_a$ and $H_b$ can be either finite groups or equal to their ‘supergroup’. So there are four possibilities:

(i) $H_a$ is finite and $H_b = \text{SU}(B)$.
(ii) $H_b$ is finite and $H_a = \text{SU}(A)$.
(iii) Both $H_a$ and $H_b$ are finite.
(iv) $H_a = \text{SU}(A)$ and $H_b = \text{SU}(B)$.

The first and second cases are impossible since, for example, if $H_a$ is finite and $H_b = \text{SU}(B)$, then $\text{SU}(B)/H_b$ has only one coset, while $\text{SU}(A)/H_a$ has infinitely many—and thus they cannot be related by a 1–1 onto map.

Let us now see why the third case is also impossible. To do this, we will show that $M$ is a continuous map. Indeed, assume that $U_k H_a \to U H_a$ (here, convergence means that for some representatives of the cosets we have $U_k \to U$. It is easy to check that this is well defined). Then let $M(U_k H_a) = V_k H_b$, $M(U H_a) = V H_b$. We will show that $V_k H_b \to V H_b$. The proof is straightforward, similar to the proof that $H_a$ is closed. Pick a series of series $\{\sigma_{n}^{(k)}\}$ such that

$$\lim_{n \to \infty} \tau_a(\sigma_{n}^{(k)}) = U_k,$$

$$\lim_{n \to \infty} \tau_b(\sigma_{n}^{(k)}) = V_k,$$

and without any loss of generality we assume that

$$\|\tau_a(\sigma_{n}^{(k)}) - U_k\| < 1/n, \quad \|\tau_b(\sigma_{n}^{(k)}) - V_k\| < 1/n.$$

Then, since $U_k \to U$, we have $\tau_a(\sigma_{n}^{(k)}) \to U$, and since $M(U H_a) = V H_b$, we can find a sub-series $\tau_b(\sigma_{n}^{(k)})$ that converges to some $\tilde{V} \in V H_b$. In order not to overload the notation, let us re-define $k$ to be that sub-series. We now claim $V_k \to \tilde{V}$. Indeed, for each $\epsilon > 0$, we can choose $K$ such that for each $k > K$, $1/k < \epsilon/2$ and $\|\tau_b(\sigma_{n}^{(k)}) - \tilde{V}\| < \epsilon/2$. Then

$$\|V_k - \tilde{V}\| \leq \|V_k - \tau_b(\sigma_{n}^{(k)})\| + \|\tau_b(\sigma_{n}^{(k)}) - \tilde{V}\| \leq \epsilon.$$  

Now $H_a$ and $H_b$ are closed normal subgroups, and therefore $\text{SU}(A)/H_a$ and $\text{SU}(B)/H_b$ are Lie groups themselves (see, for example, theorem 3.64 on p 124 of [35]). Furthermore, since every continuous homomorphism between Lie groups is also smooth (see, for example, theorem 3.39 on p 109 of [35]), we have found a smooth diffeomorphism (1–1 homeomorphism) between two differentiable manifolds. However, since both $H_a$ and $H_b$ are finite then

$$\dim \text{SU}(A)/H_a = \dim \text{SU}(A) \neq \dim \text{SU}(B) = \dim \text{SU}(B)/H_b,$$

and it is therefore impossible to find a diffeomorphism between the two manifolds. □

**Acknowledgments**

The authors thank an anonymous referee for spotting a couple of errors in the parameters of the main theorems.
References

[1] Aaronson S 2005 Quantum computing, postselection and probabilistic polynomial-time Proc. R. Soc. A 347 333–82 (arXiv:quant-ph/0412187)

[2] Aharonov D and Arad I 2006 The BQP-hardness of approximating the Jones polynomial arXiv:quant-ph/0605181

[3] Aharonov D and Arad I 2007 Polynomial quantum algorithms for additive approximations of the Potts model and other points of the Tutte plane arXiv:quant-ph/0702008

[4] Aharonov D and Ben-Or M 1997 Fault-tolerant quantum computation with constant error Proc. 29th Annu. ACM Symp. on Theory of Computing pp 176–88

[5] Janzing D and Wocjan P 2007 A simple PromiseBQP-complete matrix problem Theory Comput. 3 61–79

[6] Goldreich O 2005 On promise problems Electronic Colloquium on Computational Complexity 18 Online at http://eccc.hpi-web.de/eccc-reports/2005/TR05-018/index.html

[7] Aharonov D and Ben-Or M 2008 Quantum computation with constant error rate SIAM J. Comput. 38 1207–82

[8] Aharonov D, Jones V F and Landau Z 2006 Proc. 38th ACM Symp. on Theory of Computing (STOC 2006) (Seattle, WA) (arXiv:quant-ph/0511096)

[9] Alagic G, Jordan S P, König R and Reichardt B W 2010 Estimating Turaev–Viro three-manifold invariants is universal for quantum computation Phys. Rev. A 82 040302 (arXiv:1003.0923)

[10] Arad I and Landau Z 2010 Quantum computation and the evaluation of tensor networks SIAM J. Comput. 39 3089–121 (arXiv:0805.0040)

[11] Artin E 1925 Theorie der Zopfe, Abh. math. sem. Hamburg 4 47–72

[12] Bacon D, Kempe J, Lidar D A and Whaley K B 2000 Universal fault-tolerant quantum computation on decoherence-free subspaces Phys. Rev. Lett. 85 1758–61

[13] Bacon D, Kempe J, DiVincenzo D P, Lidar D A and Whaley K B 2001 Encoded Universality in Physical Implementations of a Quantum Computer: Proc. 1st Int. Conf. on Experimental Implementations of Quantum Computation (Sydney, Australia 2001) ed R Clark (Princeton, NJ: Rinton) pp 257 (arXiv:quant-ph/0102140)

[14] Bernstein E and Vazirani U 1997 Quantum Complexity Theory: Proc. Symp. on the Theory of Computing (STOC 1993) (Special Issue on Quantum Computation of the SIAM J. Comput. October 1997)

[15] Bordewich M, Freedman M, Lovasz L and Welsh D 2005 Approximate counting and quantum computation Comb. Probab. Comput. 14 737–54

[16] Dawson C M and Nielsen M A 2005 The Solovay–Kitaev algorithm arXiv:quant-ph/0505030

[17] Freedman M H 1998 P/NP and the quantum field computer Proc. Natl Acad. Sci. USA 95 98–101

[18] Freedman M H, Kitaev A and Wang Z 2002 Simulation of topological field theories by quantum computers Commun. Math. Phys. 227 587–603

[19] Freedman M H, Larsen M and Wang Z 2002 A modular functor which is universal for quantum computation Commun. Math. Phys. 227 605–22

[20] Freedman M H, Kitaev A, Larsen M and Wang Z 2003 Topological Quantum Computation. Mathematical Challenges of the 21st Century (Los Angeles, CA, 2000) Bull. Am. Math. Soc. (N.S.) 40 31–8

[21] Garnerone S, Marzuoli A and Rasetti M 2006 Quantum automata, braid group and link polynomials arXiv:quant-ph/0601169

[22] Goldberg L A and Jerrum M 2008 Inapproximability of the Tutte polynomial Inf. Comput. 206 908–929

[23] Grove L C 2001 Classical groups and geometric algebra (Graduate Studies in Mathematics vol 39) (Providence, RI: American Mathematical Society)

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

[25] Jones V F R 1983 Braid groups, Hecke algebras and type II1 factors Geometric Methods in Operator Algebras: Proc. US–Japan Seminar (Kyoto, July 1983) (see theorem 3.1, p 262)
[26] Jones V F R 1985 A polynomial invariant for via Von Neumann algebras Bull. Am. Math. Soc. (N.S.) 12 103–11
[27] Jones V F R 1986 Braid groups, Hecke algebras and type II factors Geometric Methods in Operator Algebras (Pitman Research Notes in Mathematics 123) pp 242–73
[28] Kauffman L 1987 State models and the Jones polynomial Topology 26 395–407
[29] Kempe J, Bacon D, DiVincenzo D P and Whaley K B 2001 Encoded universality from a single physical interaction QIP 1 (special issue) 33–55 (arXiv:quant-ph/0112013)
[30] Kitaev A Yu 2005 private communication
[31] Kitaev A Yu, Shen A H and Vyalyi M N 2002 Classical and Quantum Computation (Graduate Studies in Mathematics vol 47) (Providence, RI: American Mathematical Society)
[32] Kuperberg G 2009 How hard is it to approximate the Jones polynomial? arXiv:0908.0512
[33] Peter W S and Stephen P J 2008 Estimating Jones polynomials is a complete problem for one clean qubit Quantum Inf. Comput. 8 681 (arXiv:0707.2831v3)
[34] Temperley H N V and Lieb E H 1971 Relations between the percolation and colouring problem and other graph-theoretical problems associated with regular planar lattices: some exact results for the ‘Precolation’ problem Proc. R. Soc. A 322 251–80
[35] Warner F W 1983 Foundations of Differentiable Manifolds and Lie Groups 2nd edn (Berlin: Springer)
[36] Witten E 1989 Quantum field theory and the Jones polynomial Commun. Math. Phys. 121 351–99
[37] Wocjan P and Janzing D 2006 Estimating diagonal entries of powers of sparse symmetric matrices is BQP-complete arXiv:quant-ph/0606229
[38] Wocjan P and Janzing D 2007 A Promise BQP-complete string rewriting problem arXiv:0705.1180
[39] Wocjan P and Yard J 2006 The Jones polynomial: quantum algorithms and applications in quantum complexity theory arXiv:quant-ph/0603069
[40] Wocjan P and Zhang S 2006 Several natural BQP-complete problems arXiv:quant-ph/0606179