A Polynomial Quantum Algorithm for Approximating the Jones Polynomial

Dorit Aharonov * Vaughan Jones † Zeph Landau ‡

November 10, 2005

Abstract

The Jones polynmial, discovered in 1984 [17], is an important knot invariant in topology. Among its many connections to various mathematical and physical areas, it is known (due to results by Witten [33]) to be intimately connected to Topological Quantum Field Theory (TQFT). The works of Freedman, Kitaev, Larsen and Wang [10, 11] provide an efficient simulation of TQFT by a quantum computer, and vice versa. These results implicitly imply the existence of an efficient quantum algorithm that provides a certain additive approximation of the Jones polynomial at the fifth root of unity, $e^{2\pi i/5}$, and moreover, that this problem is BQP-complete. Unfortunately, this important algorithm was never explicitly formulated. Moreover, the results in [10, 11] are heavily based on deep knowledge of TQFT, which makes the algorithm essentially inaccessible for computer scientists.

We provide an explicit and simple polynomial algorithm to approximate the Jones polynomial of an n strands braid with m crossings at any primitive root of unity $e^{2\pi i/k}$, where the running time of the algorithm is polynomial in m, n and k. It is likely that analogies can be drawn between our algorithm for the case of $k = 5$, and the results of [10], but we were unable to do that. Our algorithm is based, rather than on physical results from TQFT, on known mathematical results (specifically, the path model representation of the braid group and the uniqueness of the Markov trace for the Temperly Lieb algebra). By the results of [11], our algorithm solves a BQP complete problem.

The algorithm we provide exhibits a structure which we hope is generalizable to other quantum algorithmic problems. Candidates of particular interest are the approximations of other downwards self-reducible #P-hard problems, most notably, the important open problem of efficient approximation of the partition function of the Potts model, a model which is known to be tightly connected to the Jones polynomial [34].

1 Introduction

Most known quantum algorithms that achieve possible exponential speed-ups [28, 3, 29, 6, 31, 14, 22] are based on the quantum Fourier transform, and solve problems which are algebraic and number theoretic in nature. Another technique used in quantum algorithms is based on random walks. Arguably, the greatest challenge in the field of quantum computation (together with the physical realization of large scale quantum computers), is the design of quantum algorithms that are based on substantially different techniques.

*School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel. doria@cs.huji.ac.il.
†Department of Mathematics, U.C.Berkeley
‡Department of Mathematics, City College, NY

In this paper we describe an efficient quantum algorithm that approximates the #P-hard problem of evaluating the Jones polynomial - which we will define shortly. The best classical algorithm for this problem is exponential. Our algorithm is significantly different from all previously known quantum algorithms. First, it solves a problem which is combinatorial rather than algebraic or number theoretic in nature. Secondly, it does so without using the Fourier transform, but instead, by exploiting a certain structure of the problem. In some sense, it encodes this structure into unitary matrices. This abstract idea might be generalizable to other problems. Importantly, the problem our algorithm solves is known to be hard for BQP. This means that in a well defined sense, this is the hardest problem quantum computers can attack.

To describe the problem our algorithm solves, we start with some background on topology. A central issue in low dimensional topology is that of *knot invariants*. A knot invariant is a function on knots (or links – namely circles embedded in R^3) which is invariant under isotopy of the link, i.e., it does not change under stretching, moving, etc., but no cutting. An important knot invariant, the Alexander polynomial, was defined in 1928 [1]. In 1984, Jones [17] discovered a new knot invariant, now called the Jones polynomial $V_L(t)$, which is a Laurent polynomial in $\sqrt{t}$ with integer coefficients, and which is an invariant of the link L. Since its discovery, the Jones polynomial has found applications in numerous fields, from DNA recombination [25], to statistical physics [34].

It is easiest to define the Jones polynomial by what is called the *Kauffman bracket*, which is related to $V_L(t)$ by a factor which is easy to calculate. To do that, present a link L by a *link diagram*, namely a projection of the link onto two dimensions. This is a picture which involves crossings. The Kauffman bracket of the link, denoted by $< L >$, is a Laurant polynomial in A for $A = t^{-4}$. It can be defined in a downward self reducible way, namely, in terms of the Kauffman brackets of links with one less crossing: $< L >= A < L_1 > +A^{-1} < L_2 >$, where L_1 is the link achieved from L by replacing one of its crossings by $\cup\!\!\cap$ (we will refer to this picture as *capcup*) and L_2 is the same link except that the crossing is replaced by $\;|\;|\;$. Also, for L a knot which contains a loop, $< L >$ is defined to be $d = -A^2 - A^{-2}$ times the Kauffman bracket of the same knot except that the loop is removed. The parameter d is also called the *loop value*. Finally, the Kauffman bracket of the unknot, namely of one loop, is equal to 1.

The Jones polynomial can also be defined via braids. A braid of n strands and m crossings is described pictorially by n strands hanging alongside each other, with m crossings, each of two adjacent strands. A braid may be "closed" to form a link by tying its ends together. In this paper we will be interested in two ways to perform such closures, namely, the *trace closure* and the *plat closure* of the braid (to be defined later on). We will thus consider the Jones polynomial of links that are trace or plat closures of braids.

From the moment of the discovery of the Jones polynomial, the issue of how hard it is to compute was important. There is a very simple inductive algorithm (essentially due to Conway in [5]) to compute it by changing crossings in a link diagram, but, naively applied, this takes exponential time in the number of crossings. On the other hand the Alexander polynomial [1], can be computed by almost exactly the same exponential algorithm but can also be computed in polynomial time using a different definition of the Alexander polynomial. Moreover if the exponential algorithm for $V_L(t)$ is improved a little using some topology, it works much faster in practice on most links [7]. It was a significant step when it was shown [15] that the computation of $V_L(t)$ is #P-hard for all but a few values of t where $V_L(t)$ has an elementary interpretation (e.g., $V_L(-1)$ is the same as the

Alexander polynomial at -1). Thus a polynomial time algorithm for computing $V_L(t)$ for any value of t other than those elementary ones is unlikely. Of course, the #P-hardness of the problem does not rule out the possibility of good approximations; see [16]. Still, the best classical algorithms to approximate the Jones polynomial at all but at the trivial points are exponential.

Yet another special feature of $V_L(t)$ is its relationship with quantum mechanics. A deep connection was discovered by Witten 20 years ago [33] between the Jones polynomial evaluated at particular roots of unity, and Topological Quantum Field Theory (based on Chern-Simons theory). In [8, 9] a model of quantum computation was defined, which is based on TQFT and the Chern-Simons theory. Kitaev, Larsen, Freedman and Wang showed that this model is polynomially equivalent in computational power to the standard quantum computation model [10, 11]. This draws a strong connection between quantum computation and the Jones polynomial. In fact, the result in [10] implicitly implies an efficient quantum algorithm for the approximation of the Jones polynomial at $t = e^{2\pi i/5}$. Unfortunately, this improtant algorithm was never explicitly formulated. This is particularly unfortunate since it is known that the approximation problem is BQP-hard, by [11], and a quantum algorithm for this problem is thus of particular importance.

In this paper we provide an efficient, explicit and simple quantum algorithm to approximate the Jones polynomial at all points of the form $e^{2\pi i/k}$. We do this for both the trace and the plat closures of a braid B, which are denoted B^{tr} and B^{pl} respectively.

Theorem 1.1 *For a given braid B in B_n with m crossings, and a given integer k, there is a quantum algorithm which is polynomial in n, m, k which with all but exponentially (in n, m, k) small probability, outputs a result $\in [\mathcal{R}eV_{B^{tr}}(e^{2\pi i/k}) - \varepsilon d^{n-1}, \mathcal{R}eV_{B^{tr}}(e^{2\pi i/k}) + \varepsilon d^{n-1}]$ where $d = -A^2 - A^{-2}$, and ε is inverse polynomial in n, k, m. The same is true with the imaginary part.*

Theorem 1.2 *For a given braid B in B_n with m crossings, and a given integer k, there is a quantum algorithm which is polynomial in n, m, k which with all but exponentially (in n, m, k) small probability, outputs a result $\in [\mathcal{R}eV_{B^{pl}}(e^{2\pi i/k}) - \varepsilon d^{3n/2}/N, \mathcal{R}eV_{B^{pl}}(e^{2\pi i/k}) + \varepsilon d^{3n/2}/N]$ where $d = -A^2 - A^{-2}$ and ε is inverse polynomial in n, k, m (N is an exponentially large factor to be defined later). The same is true with the imaginary part.*

By this we make the connections implied between quantum computation and the Jones polynomial drawn in [10] explicit, and moreover, extend them beyond the fifth root of unity, to all roots of unity of the form $e^{2\pi i/k}$. Most importantly, our algorithm uses different methods than those used in [10] to simulate TQFT by quantum computation, and in particular, our algorithm and the proof of its correctness are fairly simple. They are based purely on algebraic results rather than on high energy physics, Chern-Simons theory and TQFT.

The following theorem stresses the importance of the problem of approximating the Jones polynomial to quantum computational complexity:

Theorem 1.3 Adapted from Freedman, Larsen and Wang [11] *The problem of approximating the Jones polynomial of the plat closure of the braid at $e^{2\pi i/k}$ for constant k, to within the accuracy given in Theorem 1.2, is BQP-hard.*

Curiously, the hardness result of Theorem 1.3 is not known to hold (regardless of k) for the approximation of the trace closure for which we give an algorithm as well. We remark about the difference between the two problems in the open questions section. We finish the introduction with an overview of the algorithm and the proof of its correctness, and with some conclusions and open questions.

1.1 Description of the Algorithm

To describe our algorithm, we first need to understand the definition of the Jones polynomial slightly better. The pictures that one gets by replacing each crossing in the braid of n strands by one of the two pictures $\{$  ,  $\}$, can be assigned the structure of an algebra, called the Temperley Lieb algebra, and denoted by TL_n. In fact, the map from the crossing to the appropriate linear combination of the above two pictures defines a *representation* of the group B_n of braids of n strands, inside the TL_n algebra. The Jones polynomial can be defined as a certain trace function (i.e., a linear function satisfying $tr(AB) = tr(BA)$) on the image of the braid in the TL_n algebra.

Our goal is thus to design an algorithm that approximates this trace. To this end we use a very useful fact about this trace: It satisfies some property, called the Markov property; moreover, any trace function on the TL_n algebra (or a representation of it) that satisfies this property is equal to the above trace! This leads us to the key idea of the algorithm: Suppose we can define a representation of the TL_n algebra by matrices operating on qubits. Suppose moreover that we can define a trace on this representation which satisifes the Markov property. By the uniqueness of the Markov trace on the algebra (and on any representation of it), if we could estimate the trace of the matrices of the representation, we would have estimated the Jones polynomial!

But what is the representation that should be used? If we are set to design a quantum algorithm, it is best if the representation induced on the braid group be unitary, so that we can hope to approximate its trace by a quantum computer. The Jones polynomial was in fact discovered [17] via representations of the braid group. The original representation in [17] was defined on the tensor product of n copies of the Hilbert space $\mathbb{C}^2$, one for each string in the braid, with the operators representing the crossings in the braid acting locally in the tensor product of the corresponding two copies of $\mathbb{C}^2$. This has an obvious structural resemblance to the quantum computation model which uses n qubits, but unfortunately, this representation is not unitary.

Fortunately, it is in fact possible to give representations of the Temperley Lieb algebra which induce *unitary* representations of the braid group. Such representations were constructed in [19, 17] and are called the *path model* representations. For an integer k, the kth path model representation of the braid group B_n is defined using a gadget graph. This graph, denoted G_k, is simply the line of $k-2$ segments. The representation of B_n is then defined on the vector space spanned by *paths* of length n on this graph G_k. Each braid is mapped by this representation to a unitary matrix on this vector space. Some adaptation is required here to make this representation work on the space of n qubits.

The algorithm now goes as follows. If we want to evaluate the Jones polynomial $V_L(t)$ for L a closure of a braid in B_n, and $t = e^{2\pi i/k}$, we use the kth path model representation of B_n and consider the vector space of paths of length n on G_k, embedded into the Hilbert space of n qubits. It is not difficult to show that according to our embedding, each generator of the braid group, namely each crossing, is mapped by the path model representation to a unitary matrix which can be implemented efficiently by a quantum circuit. In other words, we can apply each crossing by a quantum circuit using polynomially many elementary quantum gates. The resulting unitary matrix corresponding to each crossing is no longer local (namely, it operates non trivially on more than just the two qubits corresponding to the strings in the crossing), but all that matters is that it can be applied efficiently. We find that the image (by the path model representation) of an entire braid can be applied efficiently by a quantum computer, where the number of elementary gates involved would be polynomial in n and in the number of crossings m. Let us call the unitary matrix

corresponding to a braid $Q(B)$.

To approximate the Jones polynomial of a trace closure of the braid, it suffices to approximate the Markov trace of $Q(B)$. This is done using standard quantum algorithmic techniques. The algorithm for the plat closure builds on similar ideas, though it is not directly stated in terms of traces. Thus we obtain a polynomial quantum algorithm for the BQP-complete problem of approximating the Jones polynomial of a plat closure of a braid.

We remark that after we have completed this work, we have learned about a previous independent attempt to prove similar results [27] using unitary representations of the braid group. Unfortunately, the work of [27] is greatly flawed, and in particular claims to provide an exact solution to the $\#P$-hard problem.

1.2 Conclusions and Further Directions

We have provided a simple algorithm for a BQP complete problem, which is different in its methods than previous quantum algorithms. In essence, what it does is to isolate a certain local structure of the problem, and assign gates which somehow exhibit the same local structure. Our hope is that this more combinatorial direction in quantum algorithms will lead to further progress in the area.

In particular, one very interesting question which seems related to the techniques presented here, is that of the approximation of the partition function of the Potts model [32]. The Potts model is tightly connected to the Jones polynomial [34], and the exact evaluation of its partition function is once again #P-hard [32]. A very important open question in mathematical physics is the complexity of the approximation of the partition function of the Potts model. We hope that the results of this paper will lead to progress in this question, or in other questions related to approximating $\#P$-complete problems.

Another open question is that of the computational complexity of the approximation of the Jones polynomial of the plat closure of a braid, at points $e^{2\pi i/k}$ for k which grows polyhnomially in the number of strands. We believe that these problems are also BQP-complete, but we were unable to prove this so far. The issue here is that in order to approximate any quantum circuit by the braid generators, one needs to apply the well known Solovay-Kitaev theorem [24], but this does not apply to non-constant generators as we get for non-constant k.

Finally, we briefly discuss the relation between the plat and the trace closures problems. It is known that any plat closure of a braid can be transformed efficiently into a trace closure of some other braid [30]. The reader might therefore find it curious that one of these problems is BQP-complete, while the other one is not known to be so. The explanation lies in the fact that the quality of the approximation in both algorithms depends *exponentially* on the number of strands in the braid. The transformation from plat to trace closures requires, in the worst case, a significant increase in the number of strands. This, unfortunately, degrades the quality of the approximation exponentially. The computational complexity of the trace closure problem is left as an open question.

Organization of paper: Section 2 provides the background. Section 3 introduces the unitary representation of the braid group B_n by operators on n qubits. Finally, in Section 4 we provide the quantum algorithms to approximate the Jones polynomial of the trace and plat closure of a braid, and prove their efficiency and correctness. Appendix A contains background on the model of quantum computation. Appendix B provides the necessary background on algebras and representations. Finally, appendix C contains proofs that were too technical to include in the paper.

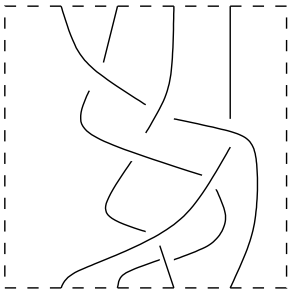
2 Background

2.1 The Braid Group

Consider two horizontal bars, one on top of the other, with n pegs on each. By an *n strand braid* we shall mean a set of n strands such that:

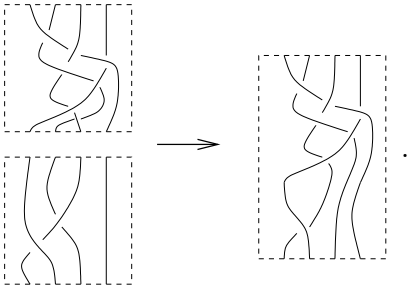
- Each strand is tied to one peg on the top bar and one peg on the bottom bar,
- Every peg has exactly one end attached to it,
- The strands may pass over and under each other,
- The tangent vector of every strand at any point along the path from top bar to bottom bar always has a non-zero component in the downward direction.

An example of a 4-strand braid:



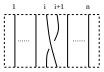
The set of n-strand braids, B_n, has a group structure with multiplication as follows. Given two n-strand braids b_1, b_2, place braid b_1 above b_2, remove the bottom b_1 bar and the top b_2 bar and fuse the bottom of the b_1 strands to the top of the b_2 strands.

The product of the above 4-strand braid with the 4-strand braid  is:



An algebraic presentation of the braid group due to Artin is as follows [2]: Let B_n be the group with generators $\{1, \sigma_1, \ldots \sigma_{n-1}\}$ and relations

1. $\sigma_i \sigma_j = \sigma_j \sigma_i$ for $|i - j| \geq 2$,
2. $\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}$.

This algebraic description corresponds to the pictorial picture of braids: σ_i corresponds to the pictorial braid  , and concatenating such pictures gives a general braid in B_n.

2.2 The Temperley-Lieb Algebras

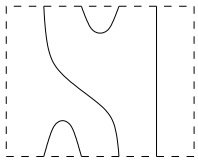
An algebra is a vector space A with a multiplication. The multiplication must be associative and distributive. We proceed to define a sequence of algebras:

Definition 2.1 *Given n an integer and d a complex number we define the Temperley-Lieb algebra $TL_n(d)$ to be the algebra generated by $\{1, E_1, \ldots, E_{n-1}\}$ with relations*

1. $E_i E_j = E_j E_i$, $|i-j| \geq 2$,
2. $E_i E_{i\pm 1} E_i = E_i$,
3. $E_i^2 = dE_i$.

When d is real, as it will be throughout the paper, we define the involution on $TL_n(d)$, denoted by $*$, by $(E_{i_1} E_{i_2} \cdots E_{i_r})^* = E_{i_r} E_{i_{r-1}} \cdots E_1$, and extend by linearity.

There is a well known geometric description of $TL_n(d)$ due to Kauffman [20]. It uses the notion of Kauffman n-diagrams, which is best explained by an example, e.g., a Kauffman 4-diagram:



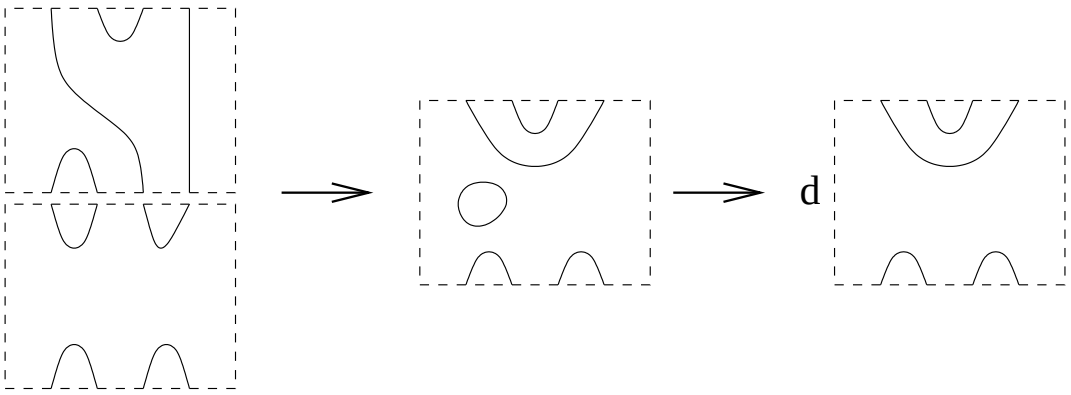
In general, a Kauffman n-diagram is a diagram as above, with n top pegs and end bottom pegs, and no crossings and no loops. More formally:

Definition 2.2 *Let D_n be a rectangle with n marked points on the top of the boundary and n marked points on the bottom. We define a Kauffman n-diagram to be a picture sitting inside D_n consisting of n non-intersecting curves that begin and end at distinct marked boundary points. We will consider two such diagrams equal if they are isotopically equivalent (keeping the boundary fixed).*

We define a vector space over these diagrams:

Definition 2.3 *Let $\mathcal{K}_n$ be the vector space of complex linear combinations of Kauffman n-diagrams.*

Multiplication of these diagrams is done just like in the case of braids. To multiply a diagram k_1 with a diagram k_2 we stack k_1 on top of k_2, and fuse the matching ends of the strands. However, the resulting diagram may contain loops, which means it is not in $\mathcal{K}_n$. Hence, the loops are removed, but the resulting diagram is multiplied by d^m (where m is the number of loops removed). For example, a multiplication of the above Kauffman 4-diagram with the Kauffman 4-diagram  results in:



This gives us a multiplication defined on the vector space $\mathcal{K}_n$, and thus, we have an algebra. This algebra is denoted $gTL_n(d)$. More formally:

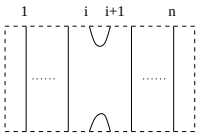
Definition 2.4 *Let $gTL_n(d)$ denote the algebra consisting of the vector space $\mathcal{K}_n$ with multiplication of two Kauffman diagrams k_1 and k_2 as follows:*

Stack k_1 on top of k_2 so that the bottom n marked points points of k_1 align with the top n marked points of k_2. Connect the strands of k_1 ending at the bottom of k_1 to the corresponding strands of k_2 starting at the top of k_2. The resulting picture may have some closed loops; let m be the number of such loops. Erasing the closed loops yields a Kauffman diagram k_3 with boundary the top of k_1 and the bottom of k_2. Define the product $k_1 \cdot k_2 = d^m k_3$.

For a Kauffman n-diagram k, define k^* to be the Kauffman n-diagram that is the reflection of k along a horizontal line. Extend the * map conjugate linearly to all of $gTL_n(d)$.

The algebras $TL_n(d)$ and $gTL_n(d)$ are isomorphic:

Theorem 2.1 *The map $\psi : TL_n(d) \to gTL_n(d)$ given by the homomorphic extension of $\psi(E_i) =$*



is an isomorphism.

Proof: It is a simple exercise to check that the image of the relations given in definition 2.1 are relations in $gTL_n(d)$. All that remains then is to show that ψ is onto and that the dimensions of $TL_n(d)$ and $gTL_n(d)$ are equal. (The details can be found in []). □

It is clear that the subalgebra of $gTL_n(d)$ consisiting of linear combinations of only those Kauffman n diagrams with a vertical rightmost strand (i.e. diagrams where the top rightmarked point is connected to the bottom right marked point by a vertical time) is isomorphic to $gTl_{n-1}(d)$. We see then that there is a natural nesting of the algebra inclusions $gTl_1(d) \subset gTL_2(d) \subset \cdots \subset gTl_n(d)$. These are exactly the image by the map ψ of the natural algebra inclusions $Tl_1(d) \subset TL_2(d) \subset \cdots \subset Tl_n(d)$.

2.3 Representing the Braid Group Inside the Temperley Lieb Algebra

We define a mapping from the braid group to the Temperley-Lieb algebra:

Definition 2.5 *$\rho_A : B_n \mapsto TL_n(d)$ is defined by*

$$\rho_A(\sigma_i) = AE_i + A^{-1}I.$$

Claim 2.1 *For a complex number A which satisfies $d = -A^2 - A^{-2}$, the mapping ρ_A is a representation of the braid group B_n inside $TL_n(d)$.*

Proof: We need to check that the relations of the braid group are satisfied by this mapping.

1. For $|i - j| > 1$, $\rho_A(\sigma_i)$ commutes with $\rho_A(\sigma_j)$ since E_i commutes with E_j.
2. We need to show $\rho_A(\sigma_i)\rho_A(\sigma_{i+1})\rho_A(\sigma_i) = \rho_A(\sigma_{i+1})\rho_A(\sigma_i)\rho_A(\sigma_{i+1})$. Opening up the first expression we get $A^3E_iE_{i+1}E_i + AE_{i+1}E_i + AE_i^2 + A^{-1}E_i + AE_iE_{i+1} + A^{-1}E_{i+1} + A^{-1}E_i + A^{-3}$. The second expression gives $A^3E_{i+1}E_iE_{i+1} + AE_iE_{i+1} + AE_{i+1}^2 + A^{-1}E_{i+1} + AE_{i+1}E_i + A^{-1}E_i + A^{-1}E_{i+1} + A^{-3}$. We remove similar terms, and using the relations of the $TL_n(d)$ it remains to show that $(A^{-1} + Ad + A^3)E_i = (A^{-1} + Ad + A^3)E_{i+1}$. This holds because the constants are 0 due to the relation between d and A.

□

From now on we set $d = -A^2 - A^{-2}$.

2.4 Tangles

For this paper, we define a tangle to be a braid in which some of its crossings have been replaced by a picture of the form { ∪ / ∩ }. Both braids and Kauffman diagrams are tangles.

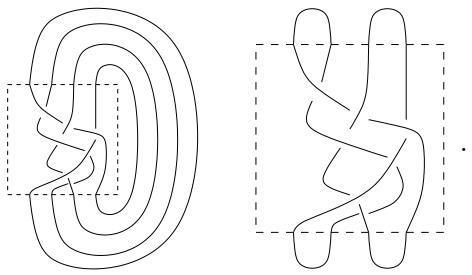
2.5 From Braids to Links

We can connect up the endpoints of a braid in a variety of ways to get links. We single out two such way:

Definition 2.6 *The trace closure of a braid B shall be the link achieved by connecting the tops of the braids around to the right to the bottom of the braid. We denote it by B^{tr}.*

Definition 2.7 *The plat closure of a $2n$-strand braid shall be the link formed by connecting on the top (respectively on the bottom) the strands beginning at the odd numbered pegs (respectively the strands ending at the odd numbered pegs) to the neighboring peg immediately to the right.*

Examples of the trace closure and the plat closure of the same 4-strand braid are:



We note that the above defined closures are also well defined for tangles.

To define the Jones polynomial, one considers an oriented version of the above links. An oriented link is a link with one arrow on each connected component of the link. By convention, the orientation of the link B^{tr} is given by assigning the downwards direction to every strand. One can easily convince oneself that this gives a consistent orientation to each loop in the resulting link, and so this results in an oriented link. There is no natural way to assign an orientation to the plat closure of a link. For the definition of the Jones polynomial of the plat closure we thus consider an arbitrary orientation. In fact, the Jones polynomial turns out to be almost independent of the orientation (up to a factor which is easy to calculate).

2.6 The Jones Polynomial

A definition of the Jones polynomial $V_L(t)$ due to Kauffman [20] is as follows. We start by defining the Kauffman bracket $< L >$ as a polynomial in A for A such that $A^{-4} = t$ (this is an equivalent definition to the one given in the introduction).

Definition 2.8 *Consider a link L, given by a link diagram. A state σ of L shall mean a choice, at each crossing*  *of L, from the set* {  ,  }. *To a state σ of a link L we associate the following expression $\sigma(L)$: Let σ^+ (σ^-) be the number of crossings for which σ chooses*  (). *Let $|\sigma|$ be the number of closed loops in the diagram gotten by replacing each crossing*  *by the choice indicated by the state σ. Define $\sigma(L) = A^{\sigma^+ - \sigma^-} d^{|\sigma|-1}$. The Kauffman bracket polynomial, also called the bracket state sum, for a link, is defined to be*

$$< L >= \sum_{all\ states\ \sigma} \sigma(L).$$

The connection between the Jones polynomial and the Kauffman bracket is given by a notion called the *writhe*:

Definition 2.9 *The writhe of an oriented link L is defined as follows. To each crossing that looks like this*  *we assign the value $+1$, whereas to each crossing that looks like this:*  *we assign the value -1. The writhe of an oriented link is the sum over all the crossings of these signs.*

Definition 2.10 *The Jones polynomial of an oriented link L is defined to be $V_L(t) = V_L(A^{-4}) = (-A)^{3w(L)} \cdot < L >$ where $w(L)$ is the writhe of the oriented link L, and $< L >$ is the bracket state sum of the link L, ignoring the orientation.*

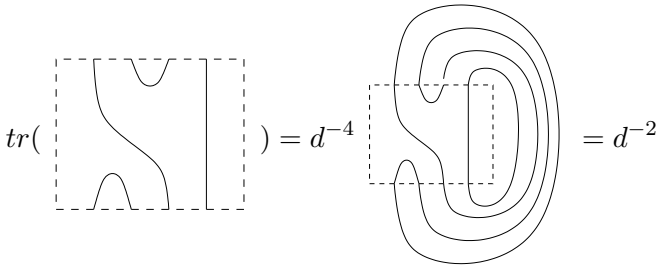
Thus, the Jones polynomial is a scaled version of the bracket polynomial. Moreover, the writhe of a link can be easily calculated from the link diagram, and hence the problem of calculating the bracket sum polynomial is equivalent in its complexity to that of calculating the Jones polynomial.

2.7 The Markov Trace

We define the following trace on $gTL_n(d)$.

Definition 2.11 *The Markov trace $tr : gTL_n(d) \to \mathbb{C}$ is defined on a Kauffman n-diagram K as follows. Connect the top n labelled points to the bottom n labelled points of K with non-intersecting curves that loop around to the right of K. In this way the rightmost labelled points on top and bottom get connected as do the second rightmost etc. (see picture 2). Let a be the number of loops of the resulting diagram. Define $tr(K) = d^{a-n}$. Extend tr to all of $gTL_n(d)$ by linearity.*

We include an example of the trace of the Kauffman 4-diagram given above:



Since $TL_n(d)$ and $gTL_n(d)$ are isomorphic, tr induces a trace on $TL_n(d)$; for simplicity we shall denote this map by tr as well.

Claim 2.2 *tr satisfies the following three properties:*

1. $tr(1) = 1$
2. $tr(XY) = tr(YX)$ *for any* $X, Y \in TL_n(d)$
3. *If* $X \in TL_{n-1}(d)$ *then* $tr(XE_{n-1}) = \frac{1}{d} tr(X)$.

Proof: It is straightforward to verify this by examining the appropriate pictures in $gTL_n(d)$. □

Of particular importance is the third property, which is referred to as the *Markov* property. These three properties uniquely determine a linear map on $TL_n(d)$:

Lemma 2.1 *[18] There is a unique linear function tr on $TL_n(d)$ (and on any representation of it) that satisfies properties* $1 - 3$.

Proof: By a reduced word $w \in TL_n(d)$ we shall mean a word in the set $\{1, E_1, \dots E_{n-1}\}$ that is not equal to cw' for any c a constant and w' a word of smaller length. We now show that a reduced word $w \in TL_n(d)$ contains at most one E_{n-1} term.

We induct on n. Clearly the only reduced words in $Tl_2(d)$ are 1 and E_1. Assume the statement is true for reduced words in $Tl_{n-1}(d)$. Suppose there exists a reduced word $w \in TL_n(d)$ containing more than one E_{n-1} term. Write $w = w_1 E_{n-1} w_2 E_{n-1} w_3$ with w_2 a word without E_{n-1}. Since w_2 must be reduced and is in $Tl_{n-1}(d)$, the induction hypothesis implies w_2 contains at most one E_{n-2} term. If w_2 does not contain a E_{n-2} term, $w_2 \in Tl_{n-2}(d)$ and it commutes with E_{n-1} so we have $w = w_1 w_2 E_{n-1} E_{n-1}$ which shows that w was not reduced. Otherwise we can write $w_2 = v E_{n-2} v'$ with v, v' both words in $TL_{n-2}(d)$. It follows therefore that v and v' commute with E_{n-1} and thus $w = w_1 v E_{n-1} E_{n-2} E_{n-1} v' w_3$ which again shows that w was not reduced. We conclude that any reduced word in $TL_n(d)$ contains at most one E_{n-1} term.

Given $w \in TL_n(d) \backslash Tl_{n-1}(d)$ a reduced word we write $w = w_1 E_{n-1} w_2$ with w_1, $w_2 \in Tl_{n-1}(d)$. Then $tr(w) = tr(w_2 w_1 E_{n-1}) = d tr(w_2 w_1)$, the first equality by property 2. The second by property 3. Thus for any word $w \in TL_n(d)$ we can reduce the trace computation to the trace of a word

$w_2 w_1 \in Tl_{n-1}(d)$. Iterating this process, (and using the fact that $tr(1) = 1$), we see that the trace of a word in $TL_n(d)$ is uniquely determined by the relations 1.-3. Since the trace is linear, the result follows. □

We have the following convenient description of the Jones polynomial in terms of the Markov trace.

Lemma 2.2 *Given a braid B, then*

$$V_{B^{tr}}(A^{-4}) = (-A)^{3w(B^{tr})} d^{n-1} tr(\rho_A(B)).$$

Proof: By Definition 2.10, we need to show that $< B^{tr} >= tr(\rho_A(B)) d^{n-1}$. We observe that there exists a one to one correspondence between states that appear in the bracket sum $< B^{tr} >$, and Kauffman n-diagrams that appear in $\rho_A(B)$. The weight of an element in the bracket state sum corresponding to the state σ is $A^{\sigma^+ - \sigma^-} d^{|\sigma|-1}$. We observe that the corresponding Kauffman n-diagram appears in $\rho_A(B)$ with the weight $A^{\sigma^+ - \sigma^-}$. Hence, by linearity of the trace, it remains to show that for each σ, the trace of the Kauffman diagram corresponding to σ, times d^{n-1}, equals to the the remaining factor in the contribution of σ to the bracket state sum, $d^{|\sigma|-1}$. This is true since by the definition of the trace of a Kauffman diagram, it is exactly $d^{|\sigma|-n}$. □

In fact, this lemma applies also if the braid B is replaced by a tangle C.

2.8 The Hadamard Test

Suppose a unitary matrix U can be applied efficiently by a quantum circuit Q. Consider a vector $|\alpha\rangle$ which we can generate efficiently. The following fact is standard in quantum computation. There exists an efficient quantum circuit whose output is a random variable $\in \{-1, 1\}$, and whose expectation is the real (imaginary) part of $\langle\alpha|U|\alpha\rangle$. This is very similar to the usual SWAP test often used in quantum computation.

Initialize the quantum computer with the two-register state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |\alpha\rangle$. Now, apply the gates in Q one by one, on the second register, conditioned that the first qubit is in the state $|1\rangle$. Since each gate is replaced by a gate on one more qubit, this operation is efficient. This gives the state $\frac{1}{\sqrt{2}}(|0\rangle \otimes |\alpha\rangle + |1\rangle \otimes Q_A(B)|\alpha\rangle$. Apply a Hadamard gate on the first qubit, measure, and output 1 if the measurement result is $|0\rangle$, -1 if the measurement result is $|1\rangle$. The expectation of the output is exactly $\mathcal{R}e\langle\alpha|U|\alpha\rangle$.

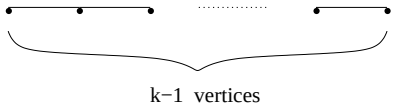
To get a random variable whose expectation is the imaginary part, start with the state $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \otimes |\alpha\rangle$, and apply instead of the Hadamard gate, the gate which takes $|0\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$, and $|1\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$.

3 Unitary Representation of B_n

Here we define a unitary representation of the braid group. To do this we start with a definition of a representation of the Temperley Lieb algebra $TL_n(d)$, and this induces a representation of the braid group B_n, via the mapping defined in Definition 2.5. The representations we use here are essentially the path model representations due to Jones [], except that these representations were not defined originally to apply to strings of n bits, which we need in order to apply them later on n qubits. The adaptation is fairly straight forward. We start by defining a representation of the Temperley Lieb algebra $TL_n(d)$, and proceed to derive the representation of the braid group.

3.1 The Representation of $TL_n(d)$

The representation of $TL_n(d)$ is defined only if $d = 2cos(\pi/k)$, for some integer $k > 2$. We consider the graph G_k which is a line of $k-2$ edges, namely, of $k-1$ sites:



A string of n bits (a computational basis state) represents a path of n steps on G_k, if when we read the bits from left to right, and interpret 0 to be "left" and 1 to be "right", then the resulting path starting at the left most site of G_k remains inside G_k all the time.

We shall interpret a string of n bits to be a sequence of instructions, where a 0 shall mean take one step to the left and a 1 shall mean take one step to the right. We shall restrict our attention to those n bit strings that describe a path that starts at the leftmost vertex of G_k and remains inside G_k at each step. From here on when we say "path" we actually mean the bit string that represents the path.

Definition 3.1 *We define $P_{n,k,\ell}$ to be the set of all paths p on G_k of n steps which start at the left most site and end at the ℓ's site. We define the subspace $\mathcal{H}_{n,k,\ell} \subset \mathcal{H}_n$ to be the span of $|i\rangle$ over all $i \in P_{n,k,\ell}$. In a similar way, we define $P_{n,k}$ to be all paths with no restriction on the final point, i.e., $P_{n,k} = \cup_{l=1}^{k} P_{n,k,l}$, and we define $\mathcal{H}_{n,k}$ to be the span of the corresponding computational basis states.*

We define a representation Φ as a homomorphism from $TL_n(d)$ to matrices operating on the subspace $H_{n,k}$.

To define Φ it suffices to specify the images of the E_i's, $\Phi(E_i) = \Phi_i$ (Φ is then extended to the entire algebra by the multiplication property of a representation, and by linearity). To uniquely define Φ_i on $H_{n,k}$, it suffices to define what it does to each basis element, namely, to $|p\rangle$ for $p \in P_{n,k}$. We need some notation. Define the following tridiagonal $k-1$ by $k-1$ matrix:

$$M_k = \begin{pmatrix} 0 & 1 & \cdots & & & 0 \\ 1 & 0 & 1 & & & \\ & 1 & & & & \\ \vdots & & & \ddots & & \\ & & & & 0 & 1 \\ 0 & & \cdots & & 1 & 0 \end{pmatrix} \tag{1}$$

Claim 3.1 *The $(k-1)$-dimensional vector λ defined by $\lambda_\ell = sin(\pi\ell/k)$ for $\ell \in \{1, ..., k-1\}$ is an eigenvector of M_k, with eigenvalue $d = 2cos(\pi/k)$.*

Proof: See Appendix C. □

We set $\lambda_\ell = sin(\pi\ell/k))$ for $\ell \in \{1, ..., k-1\}$. We also need the following notation:

Definition 3.2 *Let $p|_i$ denote the restriction of a path p to its first $i-1$ coordinates. Given a path z on G_k, we denote by $\ell(z) \in \{1, ..., k-1\}$ the location in G_k that the path z reached in its final site. Denote $z_i = \ell(p|_i)$.*

We can now define the operation of Φ_i on a path p. Φ_i is defined as an operation on the first $i+1$ coordinates in p:

$$\begin{aligned}
\Phi_i|p|_i 00\rangle &= 0 \\
\Phi_i|p|_i 01\rangle &= \frac{\lambda_{z_i-1}}{\lambda_{z_i}}|p|_i 01\rangle + \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}}|p|_i 10\rangle \\
\Phi_i|p|_i 10\rangle &= \frac{\lambda_{z_i+1}}{\lambda_{z_i}}|p|_i 10\rangle + \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}}|p|_i 01\rangle \\
\Phi_i|p|_i 11\rangle &= 0
\end{aligned} \tag{2}$$

To apply Φ_i on the n-bit string $|p\rangle$ we tensor the above transformation with identity on the last $n-i-1$ qubits. For dealing with the edge cases, we use the convention $\lambda_j = 0$ for any $j \notin \{1, .., k-1\}$.

Claim 3.2 *Φ is a representation of $TL_n(d)$.*

Proof: The proof involves checking that all the relations of the algebra are satisfied. We use the fact that λ is an eigenvector of M_k, with eigenvalue d. The proof is fairly technical and is given in Appendix C. □

3.2 Unitary Representation of the Braid Group

We use the representation of the braid group inside the $TL_n(d)$ algebra (Definition 2.5) to derive a unitary representation of B_n.

Claim 3.3 *Let $A = ie^{-\pi/2k}$. Define the map φ by specifying its operation on the generators σ_i of B_n to be $\varphi(\sigma_i) = \varphi_i = \Phi(\rho_A(\sigma_i)) = A\Phi_i + A^{-1}I$. The map φ is a unitary representation of B_n.*

Proof: The fact that Φ is a representation of the braid group follows from 3.2, and from the fact that the braid group is represented inside the $TL_n(d)$ algebra, by Claim 2.1. To prove unitarity, we first prove:

Claim 3.4 *$\Phi_i = \Phi_i^\dagger$.*

The proof appears in Appendix C. The unitarity follows by simple verification, using the fact that Φ_i is Hermitian by Claim 3.4, and the fact that $|A| = 1$ and so $A^{-1} = A^*$. We have $\Phi(\rho_A(\sigma_i))\Phi(\rho_A(\sigma_i))^\dagger = (A^{-1}I + A\Phi_i)((A^{-1})^*I + A^*\Phi_i^\dagger) = I + A^{-2}\Phi_i + A^2\Phi_i + d\Phi_i = I$. □

The map φ can be extended to operate on tangles in the obvious way: Each crossing is mapped by ρ_A to a TL algebra element, and then Φ is applied.

4 The Quantum Algorithm

We are now ready to write down the quantum algorithm that provides an additive approximation of the Jones polynomial, for both the plat closure of a briad or the trace closure of a braid. For this we first show that the unitary representation of each crossing, namely, the unitary matrices φ_i, can be implemented efficiently. The matrices φ_i are defined so far only on $H_{n,k}$ which is a subspace of the Hilbert space of n qubits. We arbitrarily define their extension to the rest of the Hilbert space to be the identity.

Claim 4.1 *For all $i \in \{1, ..., n\}$, φ_i can be implemented on the Hilbert space of n qubits using* $\mathrm{poly(n, k)}$ *gates.*

Proof: A path is a sequence of 0's and 1's, each such bit corresponds to going right or left. Consider a path p in $P_{n,k}$, and an i between 1 to n. We can easily calculate $z_i = \ell(p|_{i-1}$ (which is a number between 1 and $k-1$), and write it down on $O(log(k))$ ancilla qubits. To do this we initialize a regiter of $log(2k)$ qubits to 0. We then apply a local gate so that the last qubit in the register is 1. This initilaize the register which counts the location on G_k to 1. Now, we apply the following unitary transformation on the first qubit in the path and the extra register:

$$|b\rangle|\ell\rangle \mapsto |b\rangle|\ell + (-1)^b \mathrm{mod} 2k\rangle.$$

Since this is a unitary operation on $log(2k) + 1$ qubits, it can be applied using polynomially in k many elementary quantum gates. We now move to the next qubit in the path and apply the same transformation; We do this on all the qubits in the path up to the $i-1$th qubit, from left to right. We end up with the extra register carrying $\ell(p|_{i-1})$.

Now, Φ_i, depends only on the location $\ell(p|_{i-1})$ and on the i and $i+1$ qubits. Hence, once again we have a unitary transformation which operates on logarithmically in k many qubits, and so we can implement it in polynomially in k many quantum gates.

After we apply Φ_i, we erase the calculation of $\ell(p|_{i-1})$ by applying the inverse of the first transformation which wrote the location down. □

As a corollary, we can deduce that

Corollary 4.1 *For every braid $B \in B_n$, with m crossings, there exists a quantum circuit $Q_A(B)$ that applies $\varphi(B)$ on n qubits, using* $\mathrm{poly(m, n, k)}$ *elementary gates.*

Proof: Order the crossings in the braid in topological order, and apply the corresponding unitary matrix of each crossing, φ_i, one by one, in that order. Each crossing takes $\mathrm{poly}(n, k)$ elementary gates by Claim 4.1, and there are m of them. □

We can now describe the two algorithms. The input for both is a braid of n strands and m crossings, and an integer k.

Algorithm Approximate-Jones-Plat-Closure

- Repeat for $j = 1$ to $poly(n, m, k)$:
  1. Generate the state $|\alpha\rangle = |1, 0, 1, 0, \ldots, 1, 0\rangle$
  2. Output a random variable x_j whose expectation value is $\mathcal{R}e\langle\alpha|Q(B)|\alpha\rangle$ using the Hadamard test.
- Let r be the average over all x_j achieved this way. Output $(-A)^{3w(B^{pl})} d^{n-1} r$.

Algorithm Approximate-Jones-Trace-Closure

- 1. Classically, pick a random path $p \in P_{n,k}$ with probability $Pr(p) \propto \lambda_\ell$, where ℓ is the index of the site which p ends at. This can be done to within exponentially good precision, in polynomial time.

2. Repeat for $j = 1$ to $poly(n, m, k)$: Output a random variable x_j whose expectation value is $\mathcal{R}e\langle p|Q(B)|p\rangle$ using the Hadamard test. Let r be the average over all x_j.
3. Output $(-A)^{3w(B^{tr})}d^{n-1}r$.

Claim 4.2 *The above two quantum algorithms can be performed in time polynomial in n, m, k.*

Proof: Algorithm Approximate-Jones-Plat-Closure is clearly efficient, because the Hadamard test can be applied efficiently using Corollary 4.1. To perform the first step of the second algorithm efficiently, we use the following claim.

Claim 4.3 *Given n, k, ℓ, there exists a classical probabilistic algorithm which runs in time polynomial in n and k, and outputs a random path in $P_{n,k,\ell}$ according to a distribution which is exponentially close to uniform.*

Proof: (We thank O. Regev for a discussion that lead to this variant [26].) We first define the following $n \times k$ array A, such that $A_{i,j} = P_{i,k,j}$ (the number of path on G_k of i steps that end at j). $A_{i,j}$ can be calculated recursively efficiently, using the recursive formula:

$$P_{n,k,\ell} = P_{n-1,k,\ell-1} + P_{n-1,k,\ell+1},$$

where if the third argument in $P(n, k, \ell)$ is less than 1 or larger than $k-1$ we count the value of $P_{n,k,\ell}$ as 0. We initialize $P_{1,k,1} = 1$ and $P_{1,k,\ell} = 0$ if ℓ is different than 1, so that all paths start at the leftmost site.

To pick a random path ending at ℓ, we will pick the values of an array p from the last cite of the path to the first, one by one, as follows. Initialize the last site to be $p(n) = \ell$. Now choose $P(n-1)$ randomly with the correct weight according to $A_{i,j}$. Namely, we decide if $P(n-1) = \ell - 1$ or $P(n-1) = \ell + 1$ according to the ratio $P_{n-1,k,\ell-1} : P_{n-1,k,\ell+1}$. To make this decision, we toss a random coin with the correct bias. We can now repeat the same procedure but for paths of length $n-1$. The random choices can be done with exponentially good accuracy, and hence we can approximate the distribution exponentially close. □

The overall distribution is now sampled by picking a random $\ell \in \{1, \ldots, k\}$, with probability proportional to λ_ℓ, and then using the above claim. We note that our calculations involve irrational numbers, λ_ℓ, but these can be approximated to within an exponentially good precision efficiently. □

Theorem 4.1 *Algorithm* Approximate-Jones-Trace-Closure *approximates the Jones polynomial of B^{tr} at $A^{-4} = e^{2\pi i/k}$, to within the precision specified in Theorem 1.1.*

Proof: We will need the following definition.

Definition 4.1 *Define $Tr_n(U)$ for every U in the image of $\varphi(B_n)$ to be:*

$$Tr_n(U) = \frac{1}{N}\sum_{\ell=1}^{k-1} \lambda_\ell Tr(U|_\ell)$$

where by reducing a matrix to ℓ we mean the restriction of U to the subspace $H_{n,k,\ell}$, and Tr denotes the standard trace on matrices. The renormalization is $N = \sum_\ell \lambda_\ell dim(H_{n,k,l})$ where the sum is taken over all ℓs such that $P_{n,k,\ell}$ is non empty.

This definition makes sense because we want to apply the trace to matrices in the image of φ, and indeed, such matrices are block diagonal, with the blocks indexed by ℓ, the last site of the paths:

Claim 4.4 *For any $B \in B_n$, $\varphi(B)H_{n,k,\ell} \subseteq H_{n,k,\ell}$.*

Proof: (Of Claim 4.4) It is easy to see that Φ_i cannot change the final point of a path since it only moves 01 to 10, which does not change the end point. □

Hence, the above trace function simply gives different weights to these blocks (and gives zero weights on strings that aren't paths).

Claim 4.5 *The trace $Tr_n(\varphi(B))$ satisfies the three properties in Claim 2.2.*

Proof: (Of Claim 4.5) We start by verifying that $Tr(\Phi(1)) = 1$. The braid 1 is simply the n strands with no crossings. The path model representation takes it to the identity on the set of the legal paths, $P_{n,k}$. Hence, this property follows from the renormalization.

The fact that $Tr_n(U)$ satsifies the second property follows from Claim 4.4 plus the fact that the standard trace on matrices satisfies this property, so $Tr_n(U)$ satisfies it on each block separately.

We now turn to the Markov property. We have to show that if $X \in TL_{n-1}(d))$ then $\mathrm{tr}(\Phi(X)\Phi(E_{n-1})) = \frac{1}{d}\mathrm{tr}(\Phi(X))$. We note that for any $X \in TL_{n-1}(d)$, $\Phi(X)$ can be written as a linear combination of terms of the form $|p\rangle\langle p'| \otimes I$, with $p, p' \in P_{n-1,k}$ and the identity operates on the last qubit. By linearity, it suffices to prove the Markovian property on such matrices. Writing $|p\rangle\langle p'| \otimes I = |p0\rangle\langle p'0| + |p1\rangle\langle p'1|$ we require:

$$\mathrm{tr}(|p0\rangle\langle p'0|\Phi_{n-1} + |p0\rangle\langle p'0|\Phi_{n-1}) = \frac{1}{d}\mathrm{tr}(|p0\rangle\langle p'0| + |p0\rangle\langle p'0|).$$

We consider two cases. First, $p \neq p'$. In this case the right hand side is 0. As for the left hand side, $\langle p'0|\Phi_{n-1}$ has a zero component on $\langle p0|$. To see this, we check the two cases: if p' ends with 0 then $\langle p'0|\Phi_{n-1} = 0$, otherwise p' ends with 1. $\langle p'0|\Phi_{n-1}$ is then a sum of two terms, one equals to $\langle p'0|$ and is therefore different than $\langle p0|$, and the other ends with 01 and is thus also different from $\langle p0|$. The same argument works to show that $\langle p'1|\Phi_{n-1}$ has a zero component on $\langle p1|$. Hence, the left hand side is also 0.

It is left to check the equality in the case $p = p'$. We require

$$\mathrm{tr}(|p0\rangle\langle p0|\Phi_{n-1} + |p1\rangle\langle p1|\Phi_{n-1}) = \frac{1}{d}\mathrm{tr}(|p0\rangle\langle p0| + |p0\rangle\langle p0|).$$

Suppose $\ell(p) = \ell$. Then the right hand side is equal to $\frac{1}{d}(\lambda_{\ell-1} + \lambda_{\ell+1}) = \lambda_\ell$, using the properties of the eigenvector λ as in Claim 3.1. To see that the left hand side is the same, we again divide to cases. Suppose first that p ends with 0. In this case $\langle p0|\Phi_{n-1} = 0$. As for the other term, $\langle p1|\Phi_{n-1} = \frac{\lambda_\ell}{\lambda_{\ell+1}}\langle p1|$, using the definition of Φ_{n-1} and the fact that p without its last step ends in $\ell + 1$. The weight in the trace of the left hand side is $\lambda_{\ell+1}$, and so the left hand side is equal to λ_ℓ too. The argument is similar in the case that p ends with 1. □

By the uniqueness of the Markov trace, Lemma 2.1, we have that $Tr_n(\varphi(B)) = tr(\rho_A(B))$. Hence, using Lemma 2.2, we have that for any braid $B \in B_n$

Lemma 4.1

$$V_{B^{tr}}(A^{-4}) = (-A)^{3w(B^{tr})}d^{n-1}Tr_n(\varphi(B)).$$

Due to Lemma 4.1, the correctness of the algorithm follows trivially from the following claim.

Claim 4.6 *With all but exponentially small probability, the variable r used in the algorithm is $\in [\mathcal{R}eTr_n(\varphi(B)) - \varepsilon, \mathcal{R}eTr_n(\varphi(B)) + \varepsilon]$ for ε which is inverse polynomial in n, k, m.*

Proof: The Hadamard test indeed implies that the expectation of x_j for a fixed p is exactly $\mathcal{R}e\langle p|\varphi(B)|p\rangle$. The expectation of the variable x_j taken over a random p is thus

$$\frac{\sum_{\ell,p\in P_{n,k,\ell}} \lambda_\ell \mathcal{R}e(\langle p|\varphi(B)|p\rangle)}{\sum_{\ell,p\in P_{n,k,\ell}} \lambda_\ell} = \frac{\sum_\ell \lambda_\ell \mathcal{R}e(Tr(\varphi(B)|_\ell))}{\sum_\ell \lambda_\ell dim(H_{n,k,l})} = \mathcal{R}e(Tr_n(\varphi((B)))).$$

Since r is the average of polynomially many $i.i.d$ random variables, each taking values between 1 and -1, the result follows by the Chernoff-Hoeffding bound. □

This completes the proof of Theorem 4.1. □

Theorem 4.2 *Algorithm* Approximate-Jones-Plat-Closure *approximates the Jones polynomial of B^{pl} at $A^{-4} = e^{2\pi i/k}$, to within the desired precision as in Theroem 1.2.*

Proof: By the correctness of the Hadamard test, and by the Chernoff-Hoeffding bound, the algorithm outputs a number which is with exponentially good confidence within $\delta = 1/poly(n, m, k)$ from $\mathcal{R}e\langle\alpha|\varphi(B)|\alpha\rangle$. The main point now is the following observation:

Claim 4.7

$$|\alpha\rangle\langle\alpha| = \Phi_1\Phi_3\dots\Phi_{n-1}/d^{n/2}.$$

Proof: We show that $\Phi_1\Phi_3\dots\Phi_{n-1}$ applied to any path except for $|\alpha\rangle$ gives 0, and when applied to $|\alpha\rangle$ it gives the desired factor. To see this, recall that the operators Φ_i commute if their indices are more than one apart. We first apply Φ_1. Since the path starts at the left most site, Φ_1 turns out to simply apply the following rescaled projection: $d|10\rangle\langle 10|$ on the first two coordinates. This projection forces the first two coordinates to be 10, and if they are indeed equal to 10, a factor of d is assigned. otherwise the state is sent to 0. Hence, if the state was not sent to zero, then the path must start at the left most state when it starts its third step. So it is back to the same position it was in its first step. Hence, a similar argument applies when we apply Φ_3 on the next two coordinates. Φ_3 would thus operate as if it is the same rescaled projection $d|10\rangle\langle 10|$. By induction, we get the desired result. □

We thus have, using Definition 4.1 and the above claim:

$$\langle\alpha|\varphi(B)|\alpha\rangle = Tr(\varphi(B)|\alpha\rangle\langle\alpha|) = \frac{N}{\lambda_1}Tr_n(\varphi(B)|\alpha\rangle\langle\alpha|) = \frac{N}{\lambda_1}Tr_n(\varphi(B)\Phi_1\Phi_3\dots\Phi_{n-1}/d^{n/2}). \quad (3)$$

The matrix $\phi(B)\Phi_1\Phi_3\dots\Phi_{n-1} = \Phi(\rho_A(B))\Phi_1\Phi_3\dots\Phi_{n-1}$ is equal to $\varphi(C)$ for C the tangle achieved by concatenating the braid B with $n/2$ capcups. We have that

$$\langle\alpha|\varphi(B)|\alpha\rangle = \frac{N}{\lambda_1}Tr_n(\varphi(C)).$$

To complete the proof, we relate $Tr_n(\varphi(C))$ to $V_{B^{pl}}$, similarly to what was done in the proof of Theorem 4.1. By the uniqueness of the Markov trace, Lemma 2.1, we have that $Tr_n(\varphi(c)) = tr(\rho_A(c))$ for any tangle c. Hence, using Lemma 2.2, we have that for our tangle C,

$$V_{C^{tr}}(A^{-4}) = (-A)^{3w(C^{tr})}d^{n-1}Tr_n(\varphi(c)).$$

We now observe that as links, the trace closure of C is isotopic to the plat closure of B, by simply pulling up the lower halfs of the capcups through the right side of the picture. Hence, $V_{C^{tr}}(A^{-4}) = V_{B^{pl}}(A^{-4})$. We get that the quantity $\langle\alpha|\varphi(B)|\alpha\rangle$ which the algorithm approximates relates to the Jones polyonomial of the plat closure of the braid with the correct factor, and the approximation is indeed as required. This completes the proof of Theorem 4.2. □

5 Acknowledgements

We thank Alesha Kitaev for the clarification regarding the difference between the plat and the trace closure. We are greatful to Umesh Vazirani for helpful remarks regarding the presentation.

References

[1] Alexander, J. W. Topological invariants of knots and links. Trans. Amer. Math. Soc. 30 (1928), no. 2, 275–306.

[2] Artin, E. (1947). Theory of braids. *Annals of Mathematics*, **48** 101–126.

[3] Bernstein E and Vazirani U, 1993, Quantum complexity theory, SIAM Journal of Computation 26 5 pp 1411-1473 October, 1997

[4] Birman, J. (1974). Braids, links and mapping class groups. *Annals of Mathematical Studies*, **82**.

[5] J.H. Conway *An enumeration of knots and links, and some of their algebraic properties.*Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967) (1970) 329–358

[6] W. van Dam and S. Hallgren, Efficient quantum algorithms for shifted quadratic character problems. In quant-ph/0011067.

[7] B. Ewing and K.C. Millett, A load balanced algorithm for the calculation of the polynomial knot and link invariants. The mathematical heritage of C. F. Gauss, 225–266, World Sci. Publishing, River Edge, NJ, 1991.

[8] M. Freedman, P/NP and the quantum field computer, Proc. Natl. Acad. Sci., USA, **95**, (1998), 98–101

[9] M.Freedman, A.Kitaev, M. Larsen, Z. Wang, *Topological quantum computation. Mathematical challenges of the 21st century* (Los Angeles, CA, 2000). Bull. Amer. Math. Soc. (N.S.) 40 (2003), no. 1, 31–38

[10] M. H. Freedman, A. Kitaev, Z. Wang *S*imulation of topological field theories by quantum computers Commun.Math.Phys. 227 (2002) 587-603

[11] M. H. Freedman, M. Larsen, Z. Wang *A modular Functor which is universal for quantum computation* Commun.Math.Phys. 227 (2002) no. 3, 605-622

[12] F.M. Goodman, P. de la Harpe, and V.F.R. Jones, *Coxeter graphs and towers of algebras*, Springer-Verlag, 1989.

[13] L. Grover, Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Letters, July 14, 1997.

[14] S. Hallgren, Polynomial-time quantum algorithms for Pell's Equation and the principal ideal problem. STOC 2002, pp. 653–658.

[15] Jaeger, F.(F-INPGAM-DS); Vertigan, D. L.(4-OXMT); Welsh, D. J. A.(4-OX) On the computational complexity of the Jones and Tutte polynomials. Math. Proc. Cambridge Philos. Soc. 108 (1990), no. 1, 35–53.

[16] Mark Jerrum, Alistair Sinclair and Eric Vigoda, A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries, In STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing, 712–721, 2001

[17] V.F.R Jones, *A polynomial invariant for knots via von Neumann algebras.* Bull. Amer. Math. Soc. 12 (1985), no. 1 103–111.

[18] V.F.R. Jones, *Index for subfactors*, Invent. Math **72** (1983), 1–25.

[19] V.F.R. Jones, Braid groups, Hecke Algebras and type II factors, in *G*eometric methods in Operator Algebras, Pitman Research Notes in Math., **123** (1986), 242–273

[20] L.Kauffman, *State models and the Jones polynomial.* Topology **26**,(1987),395-407.

[21] Kauffman, Louis H.(1-ILCC-MS) Quantum computing and the Jones polynomial. (English. English summary) Quantum computation and information (Washington, DC, 2000), 101–137, Contemp. Math., 305,

[22] G. Kuperberg, A subexponential-time quantum algorithm for the dihedral hidden subgroup problem, arXiv:quant-ph/0302112.

[23] Markov, A. (1935). Über de freie Aquivalenz geschlossener Zöpfe. *Rossiiskaya Akademiya Nauk, Matematicheskii Sbornik*, **1**, 73–78.

[24] Neilsen, Chuang, Quantum Computation and Quantum Information, Cambridge press, 2000

[25] Podtelezhnikov, Alexei A.(1-NY-K); Cozzarelli, Nicholas R.(1-CA-DCB); Vologodskii, Alexander V.(1-NY-K) Equilibrium distributions of topological states in circular DNA: interplay of supercoiling and knotting. (English. English summary) Proc. Natl. Acad. Sci. USA 96 (1999), no. 23, 12974–129

[26] O. Regev, Private communication, April 2005.

[27] V. Subramaniam, P. Ramadevi, Quantum Computation of Jones' Polynomials, quant-ph/0210095

[28] Simon D 1994 On the power of quantum computation, SIAM J. Comp., 26, No. 5, pp 1474-1483, October 1997

[29] P. W. Shor: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5) 1997, pp. 1484–1509.

[30] P. Vogel, representation of links by braids: A new algorithm, In *Comment. math. Helvetici*, **65** (1) pp. 104–113, 1990

[31] J. Watrous: Quantum algorithms for solvable groups. STOC 2001, pp. 60–67.

[32] D. J. A. Welsh, "The Computational Complexity of Some Classical Problems from Statistical Physics," Disorder in Physical Systems, G.R. Grimmett and D.J.A. Welsh, eds., Clarendon Press, Oxford, 1990, pp. 307-321.

[33] Witten, Edward(1-IASP-NS) Quantum field theory and the Jones polynomial. Comm. Math. Phys. 121 (1989), no. 3, 351–399.

[34] F. Y. Wu, Knot Theory and statistical mechanics, Rev. Mod. Phys. **64**, No. 4., October 1992

A Quantum computation

We have a system of n quantum bits, called qubits. A general state of the n qubits, namely, a superposition, is a unit vector in the complex Hilbert space $\mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$. We choose an orthonormal basis for this space, which we call the computational basis: the 2^n vectors $|i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle$, where $i_j \in \{0, 1\}$. Denote by $|i\rangle$ the basis vector $|i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle$, where $i_1 i_2 \ldots$ is the binary representation of i. A unit vector in the Hilbert space is represented in this basis as $\sum_{i=0}^{2^n-1} c_i |i\rangle$ where $\sum_i |c_i|^2 = 1$.

A measurement of a qubit in the computational basis, applied to a superposition $|\alpha\rangle$, is a randomized process, the outcome of which is 0 or 1. When applied to the superposition $|\alpha\rangle$, the probability for 0 (respectively, for 1) is the norm squared of the projection of a state $|\alpha\rangle$ on the subspace spanned by all basis states where the measured qubit is in the state 0 (respectively, 1). After the measurement is applied, the quantum state collapses to the corresponding projection, renormalized to a unit vector.

A quantum algorithm is specified by a sequence of elementary quantum operations called quantum gates. A quantum gate on k qubits is a unitary $2^k \times 2^k$ matrix U. To apply this matrix on the entire system, we tensor product it with the identity on the other qubits. We restrict ourselves to $k = 2$, which suffices to provide universal quantum computation.

The initial state of a quantum algorithm is a basis state $|i\rangle$, which corresponds to the input for the computation being the string i. The sequence of gates is then applied on this state, and at the end of the algorithm, a measurement of a qubit marked as the output qubit is applied on the final state. The classical outcome of this measurement is the output of the computation.

The running time, or the complexity, of a quantum algorithm is measured in this model in terms of the number of elementary gates.

More generally, a quantum algorithm can be defined in a hybrid classical-quantum model, in which a classical computer controls a quantum computer. In this model, the classical computer decides what the input state for the quantum algorithm is, and what gates are applied. In addition, it can also apply measurements in the middle of the computation, and use the outcome of the measurement as input to the remaining of its computation. The purely quantum model is polynomially equivalent to this hybrid quantum-classical model, but the hybrid quantum-classical model is more convenient to use when designing quantum algorithms. The running time in the hybrid model is the total number of gates, quantum and classical.

B Algebra Background

An algebra is a vector space A with a multiplication. The multiplication must be associative and distributive.

Definition B.1 *A linear function from an algebra to the field of scalars is called a trace if it satisfies*

$$tr(XY) = tr(XY) \tag{4}$$

for every two elements X, Y in the algebra.

If an algebra has an identity 1 we may consider representations of a group G inside the algebra. Such a representation ρ is just a group homomorphisms from G to the group of invertible elements in the algebra, namely, we require $\rho(g_1)\rho(g_2) = \rho(g_1 g_2)$ for any two group elements $g_1, g_2 \in G$.

A representation of G is just a representation inside the algebra of $n \times n$ matrices.

We say a representation is reducible if there exists a proper subspace of vectors which is invariant under the group action. If there is no such subspace, we say the representation is irreducible.

We will also talk about the representation of an algebra. The definition is similar to that of a representation of a group:

Definition B.2 *An r dimensional representation Φ of an algebra is a linear mapping from the algebra into the set of $r \times r$ complex matrices M_r, such that for any two elements X, Y in the algebra, $\Phi(X)\Phi(Y) = \Phi(XY)$.*

If a group is represented inside an algebra then any representation of the algebra gives a representation of the group by composition.

Often, an algebra or a group is defined using a set of generators and relations between them. In this case, a representation may be defined by specifying the images of the generators, provided the same relations hold for the images as for the generators.

C Proofs

Proof: (Of Claim 3.1) We need to check that for all ℓ between 1 and $k-1$, $(M_k\lambda)_\ell = 2cos(\pi/k)\lambda_\ell$. By direct calculation using the definition of M_k we have:

$$(M_k\lambda)_\ell = \begin{cases} \lambda_2 & if \quad \ell = 1 \\ \lambda_{k-2} & if \quad \ell = k-1 \\ \lambda_{\ell-1} + \lambda_{\ell+1} & if \quad otherwise \end{cases} \tag{5}$$

For the case $\ell = 1$, we have $(M_k\lambda)_1 = \lambda_2 = sin(2\pi/k)$. By the trigonometric identity $\sin(2\alpha) = 2\cos(\alpha)\sin(\alpha)$, this is $2cos(\pi/k)sin(\pi/k)$ which is equal to $2cos(\pi/k)\lambda_1$, as we want. For the case $\ell = k-1$, we again use the same trigonometric identity and we have $(M_k\lambda)_{k-1} = \lambda_{k-2} = sin((k-2)\pi/k) = -sin(2\pi/k) = -2cos(\pi/k)sin(\pi/k) = 2cos(\pi/k)sin((k-1)\pi/k) = 2cos(\pi/k)\lambda_{k-1}$, as we want. As for the other cases, we have $(M_k\lambda)_\ell = \lambda_{\ell-1} + \lambda_{\ell+1} = \sin(\pi(\ell-1)/k) + \sin(\pi(\ell+1)/k) = 2sin(\pi\ell/k)cos(\pi/k) = 2cos(\pi/k)\lambda_\ell$ as we wanted. □

Proof: (Of Claim 3.2) We need to show that the Φ_i's satisfy the relations of Definition 2.1. We show the relations one by one, by showing that the operators are equal on computational states $|p\rangle$ for $p \in P_{n,k}$.

1. We want to show that $\Phi_i\Phi_j|p\rangle = \Phi_j\Phi_i|p\rangle$ if $|i-j| \geq 2$. Without loss of generality, $j > i$. Let p be a path. Note that due to the restriction on i, j, the pairs of bits $i, i+1$ and $j, j+1$ are disjoint. We can thus write $p|_j = p|_i p_i p_{i+1} q$ for some string q.

   Suppose first that $p_i = p_{i+1} = 0$. Then $\Phi_j(\Phi_i|p\rangle) = \Phi_j(0) = 0$. Also, $\Phi_i\Phi_j|p|_j p_j p_{j+1}\rangle = \Phi_i(\frac{\lambda_{z_j \pm 1}}{\lambda_{z_j}}|p|_i 00 q p_j p_{j+1}\rangle + \frac{\sqrt{\lambda_{z_j+1}\lambda_{z_j-1}}}{\lambda_{z_j}}|p|_i 00 q p_{j+1} p_j\rangle)$. But both paths go to 0 by Φ_i and so $\Phi_i\Phi_j|p\rangle = 0$. The same argument works if $p_i = p_{i+1} = 1$ and likewise if $p_j = p_{j+1} = 0$ pr 1.

   It remains to check the cases where $p_j \neq p_{j+1}$, and also $p_i \neq p_{i+1}$. We have four cases to check. In the first case, The first $i+1$ bit of p are $p|_i 01q01$. In this case:

$$\begin{aligned}
&\Phi_i\Phi_j|p|_j 01\rangle = \\
&\Phi_i(\frac{\lambda_{z_j-1}}{\lambda_{z_j}}|p|_i 01q01\rangle + \frac{\sqrt{\lambda_{z_j+1}\lambda_{z_j-1}}}{\lambda_{z_j}}|p|_i 01q10\rangle) = \\
&\frac{\lambda_{z_j-1}}{\lambda_{z_j}}(\frac{\lambda_{z_i-1}}{\lambda_{z_i}}|p|_i 01q01\rangle + \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}}|p|_i 10q01\rangle) + \\
&\frac{\sqrt{\lambda_{z_j+1}\lambda_{z_j-1}}}{\lambda_{z_j}}(\frac{\lambda_{z_i-1}}{\lambda_{z_i}}|p|_i 01q10\rangle \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}}|p|_i 10q10\rangle)
\end{aligned} \tag{6}$$

   where in the last equality we use the fact that z_i, the location the path reached after $i-1$ steps, does not change if the j and $j+1$ bits flip. We get exactly the same expression if we apply Φ_j after Φ_i. This is because of a similar argument: The location the path reached after $j-1$ steps does not change if we swap the i and $i+1$ bits, and hence z_j doesn't change if we first apply Φ_i. This implies that Φ_i commutes with Φ_j for this case. The argument is similar in the other three cases.

2. We now check that $\Phi_{i+1}\Phi_i\Phi_{i+1}|p\rangle = \Phi_{i+1}|p\rangle$. It is clear that both terms are zero if $p_{i+1} = p_{i+2}$. We check the other four cases by restricting p to its first $i+2$ bits.

   (a) **The case of $p|_i 001$.** We have:

$$\Phi_{i+1}|p|_i 001\rangle = \frac{\lambda_{z_{i+1}-1}}{\lambda_{z_{i+1}}}|p|_i 001\rangle + \frac{\sqrt{\lambda_{z_{i+1}+1}\lambda_{z_{i+1}-1}}}{\lambda_{z_{i+1}}}|p|_i 010\rangle = \frac{\lambda_{z_i-2}}{\lambda_{z_i-1}}|p|_i 001\rangle + \frac{\sqrt{\lambda_{z_i}\lambda_{z_i-2}}}{\lambda_{z_i-1}}|p|_i 010\rangle \tag{7}$$

   where we have used the fact that since $p_{i+1} = 0$, $z_{i+1} = z_i - 1$.
   We now apply on this expression Φ_i. $|p|_i 001\rangle$ is sent by this to 0.

$$\Phi_i|p|_i 010\rangle = \frac{\lambda_{z_i-1}}{\lambda_{z_i}}|p|_i 010\rangle + \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}}|p|_i 100\rangle \tag{8}$$

   Finally, we apply on this Φ_{i+1} again. The second term is sent to 0 and we are left with:

$$\begin{aligned}
&\Phi_{i+1}\Phi_i\Phi_{i+1}|p|_i 001\rangle = \\
&\frac{\sqrt{\lambda_{z_i}\lambda_{z_i-2}}}{\lambda_{z_i-1}}\frac{\lambda_{z_i-1}}{\lambda_{z_i}}\Phi_{i+1}|p|_i 010\rangle = \\
&\frac{\sqrt{\lambda_{z_i}\lambda_{z_i-2}}}{\lambda_{z_i-1}}\frac{\lambda_{z_i-1}}{\lambda_{z_i}}(\frac{\lambda_{z_i}}{\lambda_{z_i-1}}|p|_i 010\rangle + \frac{\sqrt{\lambda_{z_i}\lambda_{z_i-2}}}{\lambda_{z_i-1}}|p|_i 001\rangle) = \frac{\sqrt{\lambda_{z_i}\lambda_{z_i-2}}}{\lambda_{z_i-1}}|p|_i 010\rangle + \frac{\lambda_{z_i-2}}{\lambda_{z_i-1}}|p|_i 001\rangle
\end{aligned} \tag{9}$$

Where in the one before last equality we have once again used the fact that the $i+1$th bit of the path is 0 to express z_{i+1} in terms of z_i. We observe we get the desired equality between the two operators in this case.

(b) **The case of $p|_i 010$.** We have:

$$\Phi_{i+1}|p|_i 010\rangle = \frac{\lambda_{z_{i+1}+1}}{\lambda_{z_{i+1}}}|p|_i 010\rangle + \frac{\sqrt{\lambda_{z_{i+1}+1}\lambda_{z_{i+1}-1}}}{\lambda_{z_{i+1}}}|p|_i 001\rangle = \frac{\lambda_{z_i}}{\lambda_{z_i-1}}|p|_i 010\rangle + \frac{\sqrt{\lambda_{z_i}\lambda_{z_i-2}}}{\lambda_{z_i-1}}|p|_i 001\rangle \tag{10}$$

where we have again used the fact that since $p_{i+1} = 0$, $z_{i+1} = z_i - 1$.

We now apply on this expression Φ_i. $|p|_i 001\rangle$ is sent by this to 0, and $\Phi_i|p|_i 010\rangle$ is as in Equation 8.

Finally, we apply on this Φ_{i+1} again. The second term in Equation 8 is sent to 0 and we are left with:

$$\Phi_{i+1}\Phi_i\Phi_{i+1}|p|_i 010\rangle = \frac{\lambda_{z_i}}{\lambda_{z_i-1}}\frac{\lambda_{z_i-1}}{\lambda_{z_i}}\Phi_{i+1}|p|_i 010\rangle = \Phi_{i+1}\ ketp|_i 010 \tag{11}$$

Which implies the equality we are after in this case.

(c) **The case of $p|_i 101$.** We have:

$$\Phi_{i+1}|p|_i 101\rangle = \frac{\lambda_{z_{i+1}-1}}{\lambda_{z_{i+1}}}|p|_i 101\rangle + \frac{\sqrt{\lambda_{z_{i+1}+1}\lambda_{z_{i+1}-1}}}{\lambda_{z_{i+1}}}|p|_i 110\rangle = \frac{\lambda_{z_i}}{\lambda_{z_i+1}}|p|_i 101\rangle + \frac{\sqrt{\lambda_{z_i+2}\lambda_{z_i}}}{\lambda_{z_i+1}}|p|_i 110\rangle \tag{12}$$

where we have used the fact that since $p_{i+1} = 1$, $z_{i+1} = z_i + 1$.

We now apply on this expression Φ_i. $|p|_i 110\rangle$ is sent by this to 0.

$$\Phi_i|p|_i 101\rangle = \frac{\lambda_{z_i+1}}{\lambda_{z_i}}|p|_i 101\rangle + \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}}|p|_i 110\rangle \tag{13}$$

Finally, we aply on this Φ_{i+1} again. The second term is sent to 0 and we are left with:

$$\begin{aligned}&\Phi_{i+1}\Phi_i\Phi_{i+1}|p|_i 101\rangle = \\ &\frac{\lambda_{z_i}}{\lambda_{z_i-1}}\frac{\lambda_{z_i-1}}{\lambda_{z_i}}\Phi_{i+1}|p|_i 010\rangle = \Phi_{i+1}|p|_i 010\rangle\end{aligned} \tag{14}$$

which is again the equality we need.

(d) **The case of $p|_i 110$.** We have:

$$\Phi_{i+1}|p|_i 110\rangle = \frac{\lambda_{z_{i+1}+1}}{\lambda_{z_{i+1}}}|p|_i 110\rangle + \frac{\sqrt{\lambda_{z_{i+1}+1}\lambda_{z_{i+1}-1}}}{\lambda_{z_{i+1}}}|p|_i 101\rangle = \frac{\lambda_{z_i+2}}{\lambda_{z_i+1}}|p|_i 110\rangle + \frac{\sqrt{\lambda_{z_i+2}\lambda_{z_i}}}{\lambda_{z_i+1}}|p|_i 101\rangle \tag{15}$$

where we have used the fact that since $p_{i+1} = 1$, $z_{i+1} = z_i + 1$.

We now apply on this expression Φ_i. $|p|_i 110\rangle$ is sent by this to 0.

$$\Phi_i|p|_i 101\rangle = \frac{\lambda_{z_i+1}}{\lambda_{z_i}}|p|_i 101\rangle + \frac{\sqrt{\lambda_{z_i+1}\lambda_{z_i-1}}}{\lambda_{z_i}}|p|_i 110\rangle \tag{16}$$

Finally, we aply on this Φ_{i+1} again. The second term is sent to 0 and we are left with:

$$\Phi_{i+1}\Phi_i\Phi_{i+1}|p|_i 110\rangle = \tag{17}$$
$$\frac{\sqrt{\lambda_{z_i+2}\lambda_{z_i}}}{\lambda_{z_i+1}}\frac{\lambda_{z_i+1}}{\lambda_{z_i}}\Phi_{i+1}|p|_i 101\rangle =$$
$$\frac{\sqrt{\lambda_{z_i+2}\lambda_{z_i}}}{\lambda_{z_i+1}}\frac{\lambda_{z_i+1}}{\lambda_{z_i}}(\frac{\lambda_{z_i}}{\lambda_{z_i+1}}|p|_i 101\rangle + \frac{\sqrt{\lambda_{z_i+2}\lambda_{z_i}}}{\lambda_{z_i+1}}|p|_i 110\rangle) =$$
$$\frac{\sqrt{\lambda_{z_i+2}\lambda_{z_i}}}{\lambda_{z_i+1}}|p|_i 101\rangle + \frac{\lambda_{z_i+2}}{\lambda_{z_i+1}}|p|_i 110\rangle$$

Where in the one before last equality we have once again used the fact that the $i+1$th bit of the path is 1 to express z_{i+1} in terms of z_i. We observe we get the desired equality between the two operators also in this case.

3. We check that $\Phi_i^2|p\rangle = d\Phi_i|p\rangle$. It is here that we use the fact that the coefficients are defined using an eigenvector of M with eigenvalue d.

   Once again, the case in which the ith and $i+1$th bits in p are equal is trivial, since both sides are equal to 0.

   Suppose therefore that $p_i \neq p_{i+1}$. We check the two possible cases simultaneuously.

$$\Phi_i|p|_i 01\rangle = \frac{\lambda_{z_i-1}}{\lambda_{z_i}}|p|_i 01\rangle + \frac{\sqrt{\lambda_{z_i-1}\lambda_{z_i+1}}}{\lambda_{z_i}}|p|_i 10\rangle.$$
$$\Phi_i|p|_i 10\rangle = \frac{\lambda_{z_i+1}}{\lambda_{z_i}}|p|_i 10\rangle + \frac{\sqrt{\lambda_{z_i-1}\lambda_{z_i+1}}}{\lambda_{z_i}}|p|_i 01\rangle.$$

   We now apply Φ_i once more:

$$\Phi_i^2|p|_i 01\rangle = \frac{\lambda_{z_i-1}}{\lambda_{z_i}}(\frac{\lambda_{z_i-1}}{\lambda_{z_i}}|p|_i 01\rangle + \frac{\sqrt{\lambda_{z_i-1}\lambda_{z_i+1}}}{\lambda_{z_i}}|p|_i 10\rangle) + \tag{18}$$
$$+\frac{\sqrt{\lambda_{z_i-1}\lambda_{z_i+1}}}{\lambda_{z_i}}(\frac{\lambda_{z_i+1}}{\lambda_{z_i}}|p|_i 10\rangle + \frac{\sqrt{\lambda_{z_i-1}\lambda_{z_i+1}}}{\lambda_{z_i}}|p|_i 01\rangle) =$$
$$(\frac{\lambda_{z_i-1}^2}{\lambda_{z_i}^2} + \frac{\lambda_{z_i-1}\lambda_{z_i+1}}{\lambda_{z_i}^2})|p|_i 01\rangle + \frac{\sqrt{\lambda_{z_i-1}\lambda_{z_i+1}}}{\lambda_{z_i}}\frac{\lambda_{z_i-1}+\lambda_{z_i+1}}{\lambda_{z_i}}|p|_i 10\rangle)$$

   The equality $\Phi^2|p|_i 01\rangle = d\Phi|p|_i 01\rangle$ follows from the fact that $\lambda_{z_i-1} + \lambda_{z_i+1} = d\lambda_{z_i}$ due to the fact that λ is an eigenvector of M with eigenvalue d. The same argument works with $|p|_i 10\rangle$.

□

Proof: (Of Claim 3.4) We use the same notation as in the proof of Claim 3.2. It is here that we use the fact that all the entries of the eigenvector λ are non-negative. Note that all eigenvectors other than the principal eigenvector do not satisfy this property.

We show that for any two paths p, p', $\langle p'|\Phi_i|p\rangle = \langle p'|\Phi_i^\dagger|p\rangle$. We only have to check the cases where p' is equal to p or is derived from p by swapping the ith and $i+1$th coordinates. For $p = p'$

the equality is trivial if the relevant bits are equal, since both terms are 0. If the relevant bits are not equal, we require

$$\langle p|\Phi_i|p\rangle = \frac{\lambda_{z_i\pm1}}{\lambda_{z_i}} = \langle p|\Phi_i^\dagger|p\rangle = \frac{\lambda^*_{z_i\pm1}}{\lambda^*_{z_i}}.$$

This follows from the fact that the entries of λ are real.

We proceed to the case when p and p' are different and are derived from each other by swapping the bits. In this case, the two relevant bits cannot be equal, and so there is just one case to check. We have

$$\langle p'|\Phi_i|p\rangle = \frac{\sqrt{\lambda_{z_i-1}\lambda_{z_i+1}}}{\lambda_{z_i}} = \frac{\sqrt{\lambda_{z_i-1}\lambda_{z_i+1}}^*}{\lambda^*_{z_i}} = \langle p'|\Phi_i^\dagger|p\rangle$$

where the middle equality follows from the fact that the entries of λ are non negative. □