

Quantum Algorithms Beyond the Jones Polynomial

Quantum Algorithms Beyond the Jones Polynomial

by

Elad Eban

under the supervision of

Professor Dorit Aharonov

Thesis Presented to
The School of Computer Science and Engineering
for the Degree of
Master of Science

The Faculty of Science
The Hebrew University of Jerusalem
Israel, September 2007

Acknowledgments

I first want to thank Dorit who has guided me with patience, enthusiasm and devotion from my first step in the field till this very day.

I thank Itai for his contribution to the work here presented and additionally for giving me a great example of the of value persistence.

I thank my close family, my parents and my sister, for all the love they give me. Especially I thank my parents for opening my mind and nurturing me in every way since long before I can remember.

Finally, I thank Tali who is a constant source of support and inspiration for me.

Abstract

The quantum algorithm of *AJL* [3] (following the work of Freedman et al. [10]) to approximate the Jones polynomial, is of a new type: rather than using the quantum Fourier transform, it encodes the local combinatorial structure of the problem by the relations of an algebra, called the Temperley-Lieb algebra, whose matrix representation is then applied, using a quantum computer. By the results of [11], the problems this algorithm solves are the first non trivial BQP-complete problems. A natural question is whether it is possible to generalize this algorithm to other points of the famous Tutte polynomial, of which the Jones polynomial is a special case. However, the only points for which unitary representations of the Temperley-Lieb algebra exist are exactly those points to which the *AJL* algorithms apply. It seems counter-intuitive to be able to apply non-unitary representations in a quantum algorithm, and even more so to be able to show that these non-unitary algorithms are BQP-complete.

In the first part of this work we present two approaches that aim to generalize the results of *AJL* to other points of the Tutte polynomial. The first approach uses interpolation techniques in order to approximate the polynomial on an interval. The second approach uses a transformation of graphs, namely *graph tensor*, in order to approximate a discrete set of points of the Tutte polynomial that were not accessible using the algorithm of *AJL*. Both approaches achieve an approximation scale which is exponential in the number of edges of the graph. This scale is of the same order as the trivial bound and we aimed to improve it.

The main result of the present work is a polynomial quantum additive approximation of the Tutte polynomial, at *any point*, for any planar graph. This is done by first generalizing the Temperley-Lieb algebra so that any planar graph can be dealt with and then observing that non-unitary operators can in fact be applied. We note that by the result of Aharonov, Arad, Landau and the author [2], the approximation of many points of the Tutte polynomial is BQP-hard. However, this result is not described in this work.

An implication is an additive approximation of the partition function of the Potts model with any set of couplings at any temperature. However, the question of the complexity of this approximation is not fully settled; as the techniques used in [2] to prove BQP-hardness are *not* known to apply to the parameters of the Potts model. Thus, the exact quality of our algorithm regarding the Potts model is yet to be determined. In addition, our result implies approximations to many other combinatorial graph properties captured by the (multivariate or not) Tutte polynomial.

Contents

1	Introduction	1
1.1	Classical Algorithms via Particular Representation	1
1.1.1	Algorithms Related to the Braid Group	2
1.2	Quantum Algorithms via a Particular Representation	3
1.3	Our Results: Generalizing <i>AJL</i>	4
1.4	Hardness Result	4
1.5	Open Questions	5
1.6	Related Work	5
1.7	Organization of the Thesis	6
2	Background	7
2.1	Quantum Computation	7
2.2	The Tutte Polynomial	7
2.2.1	Relation with the Standard Tutte Polynomial	8
2.2.2	Relation with the Potts Model	8
2.3	The Kauffman Brackets	10
2.4	The Jones Polynomial	11
I	Initial Attempts for Generalizing <i>AJL</i>	13
3	Interpolation Approaches for Generalizing <i>AJL</i>	15
3.1	Extracting a Polynomial from Noisy Data	17
4	Graph Tensor Approach for Generalizing <i>AJL</i>	19
4.1	<i>AJL</i> Based Quantum Algorithm for Points in the Tutte Plane	20
4.2	Approximation Scale	21
4.3	BQP Hardness of Points in the Tutte Plane	22
II	Generalizing <i>AJL</i> via Representations	23
5	The Generalized Temperley-Lieb Algebra $GTL(d)$	25
5.1	Tangles and Elementary Tangles	26
5.2	Connection between the Kauffman Brackets and the $GTL(d)$ Algebra.	27
5.3	Connection with Representations of $GTL(d)$	27
6	The Path-Model Representation of $GTL(d)$	29
6.1	Diagrams as Operators over Paths on an Auxiliary Graph	29
6.2	The Hilbert Space of Paths on F	29
6.3	Compatible Paths	30

6.4	The Value $\langle p' \rho(\mathfrak{T}) p \rangle$ for Compatible Paths	30
6.5	Extending the Definition beyond Basis Elements	31
6.6	Conditions on $a_{k,\ell}, b_{k,\ell}$ for ρ to Be a Representation	31
6.7	Setting $a_{k,\ell}, b_{k,\ell}$: Path Representations for all d	32
6.8	Hermitian Representations	33
7	The Quantum Algorithm	35
7.1	Proof of the Algorithm	36
7.1.1	Moving Qubits	37
7.1.2	Simulating a Linear Operator	37
7.1.3	Completing the Proof of the Algorithm with the Correct Approximation Window Size	38
7.2	Improving the Approximation Window	38

Chapter 1

Introduction

The Fourier transform was widely used in computer science long before the birth of quantum computation. Some of the edge quantum computers may have over classical computers, is the ability to apply such a transformation on a Hilbert space of exponential size, namely the tensor space of n qubits. From the early days of quantum algorithms, the Fourier transform was used in order to achieve exponential speedup as compared to classical algorithms. The algorithms of Deutsch-Jozsa, Simon, and Shor [21] all use different types of Fourier transforms to find hidden structure in the problem at hand.

The Fourier transform is simply a change of basis to the basis of irreducible representations of a certain group. The set of irreducible representations encapsulates many interesting properties of the group; the algorithms mentioned above use the Fourier transform in order to discover a hidden structure, often a of some function on the group.

A natural question that arises is why use *all* the representations of the group? Maybe it is possible to draw some benefit from one particular representation or another.

Let us first briefly explore some classical algorithms that use representations in such a way.

1.1 Classical Algorithms via Particular Representation

An elementary computational question when dealing with groups is whether two sequences of generators are equal to the same group element. It is straightforward to see that this problem is equivalent to the that of deciding whether an element is equal to the identity element. The latter problem is known as the word problem. More formally, given a set of generators and relations of a group, one must decide whether a certain product of the generators is equivalent to the identity element.

Focusing for instance on the free group. Elements of the free group with two generators a, b are words taken from the alphabet $\{a, a^{-1}, b, b^{-1}\}$, where the only group relation is $a \cdot a^{-1} = b \cdot b^{-1} = 1$. In this group it is clear that the word problem is solvable in polynomial time (simply by repeatedly erasing adjacent elements of the form $x^{-1}x$). A stronger and non-trivial result by Lipton [20] is that the word problem is in fact solvable in logarithmic space. The algorithm suggested uses a faithful representation ρ of the free group with two generators:

$$\rho(a) = \begin{pmatrix} 1 & 2 \\ 1 & 1 \end{pmatrix}$$
$$\rho(b) = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix}$$

We briefly sketch the algorithm: Given elements $A_1, A_2 \dots A_n$ and the representation ρ of this group, compute $\rho(A_1)\rho(A_2)\dots\rho(A_n) \pmod q$ for all values of $q \leq q_n$ where q_n is some function polynomial in n . Since the representation is faithful, the input is equal to the identity element if and only if the

product *without the modulus* is equal to the identity matrix. In Ref. [20] Lipton has proved that if all the products of the matrices $\pmod q$ are equal to the identity element, then so is the product without the modulus. Hence, the correctness of the algorithm follows. We note that for reasons of simplicity, the algorithm described here is a special case of Lipton's algorithm that applies to all linear groups (groups with a faithful representation).

1.1.1 Algorithms Related to the Braid Group

A famous representation that can be used to solve an algorithmic problem is the Lawrence-Krammer representation of the braid group. We briefly introduce the braid group and then discuss the representation and its algorithmic use.


A braid of n strands and m crossings is described pictorially by n strands hanging alongside each other, between two horizontal bars, with m crossings, each of adjacent strands. Here is a four-strand braid with seven crossings:

The set of n -strand braids, B_n has a multiplicative group structure as follows: given two n -strand braids b_1, b_2 place the braid b_1 on top of b_2 , remove the bottom bar of b_1 and the top bar of b_2 , and fuse the bottom of the b_1 strands with the top of the b_2 strands. Two elements in the braid group are considered equal if they are topologically equivalent.

The product of the above four-strand braid and this four-strand braid is:

An algebraic presentation of the braid group due to Artin is as follows [6]: Let B_n be the group with generators $\{1, \sigma_1, \dots, \sigma_{n-1}\}$ 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 .

The Lawrence-Krammer representation is a faithful [7] mapping $\mathcal{K} : B_n \rightarrow GL_{\frac{n(n-1)}{2}}(\mathbb{Z}[t^{\pm 1}, q^{\pm 1}])$. We note that for the standard set of generators for the braid group, the entries of the matrix representation of each generator are all polynomials (in $t^{\pm 1}, q^{\pm 1}$) of degree at most $\frac{n(n-1)}{2}q$. In addition, we note that these matrices can be computed efficiently. This gives us a simple efficient algorithm for the word problem in the braid group. Given a sequence of generators $\sigma_1 \dots \sigma_m$ we compute the product of the relevant matrices. We claim that this computation can be done efficiently. It will be enough to check that the entries of the product of the matrices are of adequate size. Each entry of this product

is a Laurent polynomial in t, q , with a degree bounded by $mn(n-1)/2$, moreover, the coefficients of this polynomial grow only exponentially with m and thus can be handled efficiently. Thus we have now a simple efficient algorithm that solves a highly non-trivial problem, namely the word problem of the braid group.

Another example of the use of the Lawrence-Krammer representation is to break a public-key cryptosystem that uses the braid group. In Ref. [18] a public-key protocol based on the assumed intractability of a problem related to the braid group was suggested. However, Cheon and Jun [8] have shown using the Lawrence-Krammer representation and its faithfulness, that the suggested protocol does not provide any security.

1.2 Quantum Algorithms via a Particular Representation

As we have seen in the classical case, it is possible to use the simple, familiar, and accessible structure of a matrix representation in order to access properties of algebraic objects. We wish to import these ideas to the setting of quantum computation. In order to employ the power of quantum computation, we would naturally prefer matrix representation of very high dimension (where quantum computers seem to have an advantage). In addition, in order for a quantum computer to apply these matrices efficiently, we need them to decompose to matrices having local structure. In other words, the matrices have to be tensor products of small matrices and the identity.

This approach might seem strange, as it is not possible to directly compute the product of large matrices even in the quantum setting, since they are simply too large (exponential in the number of qubits used)! Thus, in order to benefit from the representation we have to apply the matrices on a certain vector. Eventually, we will be able to extract a small amount of information from the product of the matrices, namely the value of one entry of the final product of matrices.

Aharonov, Jones and Landau, in [3] used a famous mathematical structure known as the Temperley-Lieb algebra. This algebra has a tight connection with the braid group discussed above and to the so-called Jones polynomial which is discussed later on. Roughly, the Temperley-Lieb algebra is a finite dimensional algebra of diagrams, generated by elements of the type:

The product of two elements is defined as the diagram that results from placing the first element on top of the second and fusing the strands. Similarly to the braid group, elements that are topologically equivalent are considered equal. Aharonov et al. showed a quantum algorithm that approximates the Jones polynomial of a link that is a plat closure of a braid, i.e., the link that results from connecting each two adjacent strands $(2i-1, 2i)$, on both the lower and upper side of a $2n$ strand braid. The approximation scale guaranteed by the *AJL* algorithm is in the order of $d^{n/2}$ where d is a real number close to two and n is the number of strands of the braid. This result was achieved using the path model representation, of the Temperley-Lieb algebra and by known connections between the braid group and the algebra.

When looking at the *AJL* algorithm it is a bit frustrating to see that the only points that yield to the approximation technique are the roots of the unity of the form: $e^{2\pi i/k}$. The *AJL* algorithm was restricted to these set of points, since these were the ones having a unitary representation.

1.3 Our Results: Generalizing *AJL*

Our initial approach to generalize the result of *AJL* was using interpolation. Intuitively, if we could evaluate an n -degree polynomial at $n + 1$ points, then we could extrapolate the polynomial itself and so evaluate it at any point. We know that we can approximate the Jones polynomial at infinitely many points and thought that this might suffice to give a certain approximation at any point. Using Lagrange interpolation we show that the only approximation that can be achieved is in the same scale of the trivial bound on the Jones polynomial. With the use of a different interpolation technique due to Arora and Khot [5] we show a procedure that can provides us with a non trivial approximation of the Jones polynomial for an entire interval. However, this technique can *not* be applied efficiently since it uses an exponential number of points. Specifically, we did not manage to find an efficient algorithm that approximates the Jones polynomial.

Another approach to extend the results of *AJL* was inspired by the work of Jaeger, Vertigan and Welsh [14]. They have shown that the computational complexity of evaluating the Tutte polynomial at all points (except for nine points and a special curve) is $\#P$ - hard. The Tutte polynomial is a graph (and matroid) invariant of which the Jones polynomial is a special case. Jaeger et al. proved this result by performing a transformation on graphs known as *graph tensor* (see Definition. (4.0.2)); in this transformation a simple graph is used to change an arbitrary graph in a way that preserves some of the Tutte polynomial. The use of graph tensors, specifically, the k -thickening and k -stretch types of graph tensors, enabled us to extend the approximation of the *AJL* algorithm for an additional infinite but discrete set of points. Namely, we present a set of point for which an additive approximation is possible using a quantum computer. Furthermore, using the same technique, we show for a different set of points that approximating them within a certain approximation scale is BQP-hard.

Unfortunately, the approximation scale that can be attained by this method is in the order of d^{km} where m is the number of crossings in the braid and k is some constant whose exact nature will be discussed later. Similarly to the interpolation approach this method provides approximation on the same scale as the trivial bound. To derive better approximation scale (and stronger hardness results) a different method must be used.

The most fruitful approach to extend the work of Aharonov et al. was to define the generalized Temperley-Lieb algebra and use the same representation as Aharonov et al.. Using the generalized Temperley-Lieb algebra and by realizing that it is possible to apply **non-unitary** representation on a quantum computer, we managed to construct a quantum algorithm that approximates the Tutte polynomial at any point. We note that the approximation scale provided by this approach is exponentially smaller than the one that is achieved by the graph tensor approach. In addition we note that our algorithm is a true generalization of the *AJL* algorithm. In the sense that when restricting our algorithm to the points that can be approximated by *AJL* the algorithms are equivalent and achieve the same approximation scale.

1.4 Hardness Result

Approximation algorithms come in many forms. Among the common ones are the *FPTAS* or *FPRAS* types, which stand for fully polynomial time (randomized) approximation scheme. In these types of approaches the output of the algorithm is guaranteed to be within the range $[X(1 - 1/\text{poly}(n)), X(1 + 1/\text{poly}(n))]$, where X is the value to be approximated. However, our results give an additive approximation, namely the output of our algorithm lies within $[X - \Delta/\text{poly}(n), X + \Delta/\text{poly}(n)]$, where Δ is *not* guaranteed to be close to X in any way! The actual size of the approximation scale Δ is of crucial importance: if it is too large then the approximation task might become trivial. For this reason one has to provide evidence to why an additive approximation is non-trivial.

A way to suggest that the algorithm performs a meaningful task (under the assumption that

$BQP \neq BPP$) is by showing that approximating the value of the Tutte polynomial up to a given scale Δ is BQP-hard. Following the path laid down in Refs. [3] and [1] Aharonov, Arad, Landau and the author proved that for many points the computation done by the algorithm presented in this thesis is BQP-hard. This result is presented in paper [2] and does not appear in this thesis.

1.5 Open Questions

- In our *graph tensor* approach to extend the results of *AJL*. We use in fact only two restricted types of graph tensor transformations, namely k -thickening and k -stretch. It interesting to find out whether is it possible to use this approach with more success with other more complex or innovative graphs. Can this technique be used in order to prove BQP-completeness of points of the Tutte plane? From a slightly different point of view, can this approach be used fruitfully in the context of representation based algorithm for the entire Tutte plane? Finally, we would like to know if there are other types of graph transformations that can be used instead of the graph tensor transformation?
- The approach using the Temperley-Lieb algebra relies on a particular representation of the algebra called the path model representation. The construction made in this thesis uses auxiliary graphs which are lines (infinite or finite). We ask whether this is the most suitable representation for approximating the Tutte Polynomial? More specifically, is it possible to get better approximations or stronger hardness results with different auxiliary graphs or with fundamentally different representations?
- The *HOMFLY* polynomial, which is a knot invariant, contains as special cases both the Tutte and the Jones polynomials. Yard and Wocjan [28] have generalized the *AJL* algorithm to the *HOMFLY* polynomial. However similarly to Aharonov et al. Yard and Wocjan used only unitary representations related to the braid group. Thus, their generalization is restricted to a discrete set of points. It would be interesting to find out whether it is possible to use techniques presented in this thesis, namely the use of non-unitary representations, to further determine the complexity of the *HOMFLY* polynomial?
- It is not hard to see that the algorithm described in Chapter 7 works when the loop value d and the graph weights are not necessary complex numbers. Is there any interest in considering these values from an arbitrary field?
- The algorithm described in Chapter 7 is quite wasteful in the quality of the approximation it guarantees. It is possible to group operators together before applying them on the quantum computer. *Any* such grouping will improve the approximation scale of the algorithm! However, we do not know of the optimal (or even approximate) way to perform this task. This issue is discussed briefly in Sec. 7.2 and more thoroughly in [2] but is yet unsettled.
- In this thesis we manage to apply, using a quantum computer, non-unitary representations of the Temperley-Lieb algebra in order to gain information hidden in a certain algebra element. Which other mathematical structures can a quantum computer explore efficiently using non-unitary representations? Is the possibility to apply non unitary transformation on a quantum computer be used in a different context (regardless of representations)?

1.6 Related Work

The connection between quantum computation and the Potts model, via quadratically signed weight enumerators, was explored in Ref. [19], but no explicit quantum algorithm, was given.

Other additive approximations of partition functions of statistical physics models are implicit in the work of Van Den Nest et al. [23], using very different methods. The approximation scale derived using the methods of [23] is not comparable to ours, and so we have not included any discussion regarding the non-triviality of the approximation with this scale.

Yard and Wocjan [28] have succeeded in generalizing the work of Aharonov et al. in the context of the *HOMFLY* polynomial, of which the Tutte and Jones polynomials are special cases. Their quantum algorithm uses *unitary* representation related to the braid group to find an additive approximation for the *HOMFLY* polynomial. Additionally they proved the non triviality of their results by showing it is BQP-hard to compute such approximations. The approximations scale achieved by Yard and Wocjan matches the one of *AJL*; hence it matches ours as well since when restricted to unitary representation, our algorithm is in fact equivalent to *AJL*.

We note that after the publication of the archive version of these results [2], Geraci and Lidar derived quantum algorithmic speedups for the *exact* evaluations of the Potts model for a restricted class of graphs [12] for which the problem is known to be in P (as long as q is fixed).

1.7 Organization of the Thesis

In Chapter 2 we provide some background on the Tutte polynomial and its relation to the Potts model, Jones polynomial and Kauffman brackets. The rest of the paper is divided into two parts. In Part I we present the interpolation and graph tensor approaches to extend the results of *AJL* (Chapters 3 and 4 respectively).

In Part II we present the main achievement of this thesis, a quantum algorithm generalizing *AJL* that uses the path model representation and the generalized Temperley-Lieb algebra. This part consists of Chapters 5, 6 where we describe in details the *generalized Temperley-Lieb* algebra and the *path-model representation*. Furthermore, in the Chapter 5 we make the connection between the Temperley-Lieb algebra and the Kauffman brackets. In Chapter 7 we present the quantum algorithm for the approximation of the product of certain elements of the generalized Temperley-Lieb algebra.

Chapter 2

Background

2.1 Quantum Computation

In this thesis we use the standard model for quantum computation. We allow application of arbitrary two qubit unitary gates. An excellent and detailed description can be found in Ref. [21].

2.2 The Tutte Polynomial

The multivariate Tutte polynomial (often referred to here simply as the Tutte polynomial) is defined for finite graphs with weighted edges and a scalar q . Below we give a short definition and description of this important polynomial. Our treatment and notation closely follow Ref. [22], which we strongly recommend as an introductory text for this subject.

Given a graph $G = (V, E)$ and a subset of edges $A \subseteq E$, we define $k(A)$ to be the number of connected components in the subgraph (V, A) . Note that an isolated vertex is considered a connected component; hence when A is empty, $k(A) = |V|$. The multivariate Tutte polynomial is now defined by

Definition 2.2.1 (The multivariate Tutte polynomial) *Let $G = (V, E)$ be a finite graph with variables $\mathbf{v} = \{v_e\}$ assigned to its edges $e \in E$. Then the multivariate Tutte polynomial of G is a polynomial in \mathbf{v} and an extra variable q , defined by*

$$Z_G(q; \mathbf{v}) = \sum_{A \subseteq E} q^{k(A)} \prod_{e \in A} v_e . \quad (2.1)$$

When considering a particular set of values of the variables $\mathbf{v} = \{v_e\}$, it is customary to call them weights and to refer to G as a weighted graph. It is also common to substitute the variables $\mathbf{v} = \{v_e\}$ with a single variable v and obtain a two-variable polynomial

$$Z_G(q, v) = \sum_{A \subseteq E} q^{k(A)} v^{|A|} . \quad (2.2)$$

As an example, consider the graph $G =$ with all edges set as $v_e = v$. Its expansion contains terms such as the following:

- $k(A) = 1, |A| = 5 \implies \text{Term} = qv^5$
- $k(A) = 2, |A| = 3 \implies \text{Term} = q^2v^3$

- $k(A) = 3, |A| = 1 \implies \text{Term} = q^3 v$

2.2.1 Relation with the Standard Tutte Polynomial

When all the variables $\mathbf{v} = \{v_e\}$ are substituted by a single variable v , it is easy to relate the resulting polynomial $Z_G(q, v)$ to the standard Tutte polynomial $T_G(x, y)$ defined by [25, pp. 45]:

$$T_G(x, y) \stackrel{\text{def}}{=} \sum_{A \subseteq E} (x-1)^{k(A)-k(E)} (y-1)^{|A|+k(A)-|V|} . \quad (2.3)$$

Simple algebra yields

$$T_G(x, y) = (x-1)^{-k(E)} (y-1)^{-|V|} Z_G\left((x-1)(y-1), y-1\right) . \quad (2.4)$$

In other words, $T_G(x, y)$ and $Z_G(q, v)$ are essentially equivalent under the change of variables:

$$x = 1 + q/v , \quad y = 1 + v , \quad (2.5)$$

$$q = (x-1)(y-1) , \quad v = y-1 . \quad (2.6)$$

2.2.2 Relation with the Potts Model

In this section we elaborate on the connection between the Potts model and the Tutte polynomial. A broad review of this model, in a more theoretical physics context, can be found in Refs. [27] and [26].

The Potts model, also known as the q -state Potts model, is a physical model that is defined on a graph $G = (V, E)$ and some $q \in \mathbb{Z}_+$. The vertices of the graph, known as sites, hold physical objects, such as “atoms” or “spins,” which can be in one of q possible states (colors, or spin states) $\{1, \dots, q\}$. A coloring of all sites is called a *configuration* of the system and is denoted by a map $\sigma : V \rightarrow \{1, \dots, q\}$.

The energy of the system is defined by assigning a coupling constant J_e to each edge $e \in E$. If e_1 and e_2 denote the adjacent sites of e , and $\sigma(e_1), \sigma(e_2)$ denote their colors respectively, the energy of the edge is

$$\epsilon_e = -J_e \delta_{\sigma(e_1), \sigma(e_2)} , \quad (2.7)$$

where $\delta_{a,b}$ is the usual Kronecker delta. In other words, the energy of an edge is $-J_e$ if its sites share the same color and is zero otherwise. Finally, the total energy of the configuration σ is the sum of the energies of all the edges:

$$H(\sigma) = \sum_{e \in E} \epsilon_e = - \sum_{e \in E} J_e \delta_{\sigma(e_1), \sigma(e_2)} . \quad (2.8)$$

In statistical physics we assume that the system constantly changes its microscopic configuration. The actual state of the system is therefore of little importance; rather we try to estimate the probability $P(\sigma)$ that the system will be in a configuration σ and use this information to understand its the global properties. In this context, it is common to consider the scenario in which the system is attached to another, much bigger, system (a thermal bath), with a constant temperature T , and that the two systems are allowed to exchange energy (heat). Then the system is described by the so-called *canonical ensemble*; its probability distribution is the well-known *Boltzmann-Gibbs distribution*

$$P(\sigma) = \frac{1}{Z^{\text{Potts}}} e^{-\beta H(\sigma)} . \quad (2.9)$$

Here $\beta = 1/(k_B T)$ is the inverse temperature with k_B being the Boltzmann constant. Z^{Potts} is the normalization factor which is called the *partition function* of the system and is given by

$$Z^{\text{Potts}} \stackrel{\text{def}}{=} \sum_{\sigma} e^{-\beta H(\sigma)} . \quad (2.10)$$

The partition function Z^{Potts} has a central role in statistical physics. Knowing it allows us to calculate important global properties of the system, e.g., its average energy, entropy, and heat capacity.

For a positive temperature $T > 0$ (equivalently $\beta > 0$) and real couplings J_e , the total energy $H(\sigma)$ is a real number; hence Eq. (2.9) is a valid probability distribution. In such a case, configurations with low energies are preferable. An edge with $J_e > 0$ is called “ferromagnetic.” Its adjoint sites “prefer” to be monochromatic. On the other hand, an edge with $J_e < 0$ is called “anti-ferromagnetic,” and its adjoint sites tend to have non-identical colors. An interesting case is when all sites are anti-ferromagnetic and $T \rightarrow 0$ (equivalently $\beta \rightarrow +\infty$). In such a case, if there are configurations of the graph in which adjacent sites have different colors (i.e., “legal coloring”), only those configurations will have a non-vanishing probability. In particular, we find

$$\lim_{T \rightarrow 0} Z^{\text{Potts}} = \# \text{ of } q\text{-colorings} . \quad (2.11)$$

Deciding whether a graph has one or more q -colorings for $q > 2$ is a well-known NP-complete problem. Hence, the above example suggests that calculating or even approximating the partition function can be, at least in some cases, a highly non-trivial task. Currently, most common approximations of this quantity use Monte Carlo types of algorithms (see e.g., Refs. [13, 24]).

If we now define

$$v_e = e^{\beta J_e} - 1 , \quad (2.12)$$

then a simple algebra shows that the partition function can be written as

$$Z^{\text{Potts}}(q, \mathbf{v}) = \sum_{\sigma} \prod_{e \in E} [1 + v_e \delta_{\sigma(e_1), \sigma(e_2)}] . \quad (2.13)$$

It is far from obvious that $Z^{\text{Potts}}(q, \mathbf{v})$, which is defined separately for each positive integer q , is in fact a polynomial in q , but as noted by Fortuin and Kasteleyn in the late 1960’s [9, 15], this is indeed the case:

Theorem 2.2.1 (Fortuin-Kasteleyn representation of the Potts model) *For integer $q > 1$, and for all \mathbf{v} ,*

$$Z^{\text{Potts}}(q, \mathbf{v}) = Z_G(q, \mathbf{v}) . \quad (2.14)$$

A proof of this theorem can be found in Ref. [22].

Approximating the partition function of the q -state Potts model is therefore equivalent to approximating the multivariate Tutte polynomial of the underlying graph with weights given by Eq. (2.12). The complexity of the two problems is identical.

Theorem 2.2.1 enables us to assign a probability to every coloring of a graph for integer $q > 0$ and $v_e > -1$ (which is equivalent to J_e being real for real temperatures). We can actually extend this probabilistic interpretation for non-integer q ’s by using the original Definition. (2.2.1). The idea is to view the elements of the sum in Eq. (2.1) as the new configuration space. In other words, a configuration is defined by a subset of the edges $A \subseteq E$ and is assigned a probability that is proportional to $q^{k(A)} \prod_{e \in A} v_e$. Theorem (2.2.1) guarantees that these two definitions coincide in the appropriate region. We are thus led to define the following:

Definition 2.2.2 (Potts parameters) *We say that \mathbf{v}, q are Potts parameters (or physical parameters) when either:*

- q is an integer greater than 0, and for all edges $v_e > -1$; or
- $q > 0$ and for all edges $v_e > 0$.

Note that the conditions of the second case guarantee that $q^{k(A)} \prod_{e \in A} v_e$ is real and positive. Also note that in the first (ferromagnetic) case $J_e > 0$ corresponds to $v_e > 0$, and $J_e < 0$ (anti-ferromagnetic) corresponds to $-1 < v_e < 0$. In both cases, we are assured that $Z_G(q, \mathbf{v})$ is a real positive number.

Figure 2.1: From a planar graph to a medial graph. In the planar graph, the weights v_i are assigned to the edges. These weights are then transformed to the *crossing* weights u_i in the medial graph, which are related by $v_i = du_i$.

2.3 The Kauffman Brackets

The Kauffman brackets are most easily defined on an object called a *medial graph*. Given any planer G , the medial graph L_G is a four-regular graph that is constructed from G : we first encircle the facets of G with lines and then cross any two lines that surround an edge by putting a 4-regular vertex in its middle. See Fig. 2.1.

We continue by coloring the regions of the medial graph in black and white, which is done uniquely by first painting the outermost area – the area that surrounds the graph – in white and then painting the rest such that regions on two sides of an edge (of the medial graph) are in opposite colors. An example for such coloring is found in Fig. 2.1.

The vertices of the medial graph, which we also call *crossings*, are assigned weights, denoted $\mathbf{u} = \{u_e\}$, where u_e is the weight of the crossing corresponding to the edge e in G . If G is a weighted graph, then one can define a natural, one-to-one, mapping between its weights $\mathbf{v} = \{v_e\}$ and the weights $\mathbf{u} = \{u_e\}$. The nature of this mapping will become clear shortly, when we define the Kauffman brackets of a medial graph.

To define the *Kauffman brackets* of L_G , we first define the notion of a *state*. A state $\sigma = \{\sigma_e | e \in E\}$ of the medial graph L_G is an assignment of 0 or 1 to each crossing. σ is understood as a prescription of how to open every crossing in L_G in one of two ways: $(\sigma_e = 1)$ or $(\sigma_e = 0)$. When $\sigma_e = 1$ the shaded areas are connected (and consequently the original edge e of G is included in the resulting shaded area); when $\sigma_e = 0$ the shaded areas are disconnected (and the edge is not included in the shaded area). Once all the crossings have been opened, the resulting diagram contains only loops. We let $|\sigma|$ denote their total number.

The Kauffman brackets of L_G are defined as the following sum

$$\langle L_G \rangle \stackrel{\text{def}}{=} \sum_{\sigma} \langle L_G | \sigma \rangle, \quad (2.15)$$

where $\langle L_G | \sigma \rangle$ is defined by

$$\langle L_G | \sigma \rangle \stackrel{\text{def}}{=} d^{|\sigma|} \prod u_e^{\sigma_e}. \quad (2.16)$$

In other words, we take the product of $d^{|\sigma|}$ with the weights u_e of all crossings that were opened in such a way that their edge was left alive. We note that our definition of the Kauffman brackets is different from the standard definition found in the theory of knots [16, 25, 17]. The main difference is that in our definition we allow a different weight u_e for each crossing, whereas in the standard definition only one weight is used. It might therefore be more appropriate to call this object the “weighted Kauffman brackets.” However, for the sake of brevity and at the risk of a slight confusion, we will continue to call it simply the Kauffman brackets. A definition of the standard Kauffman brackets and their relation to knots theory, the Jones polynomial, the Potts model, and many other topics in mathematical physics can be found in Refs. [16, 25, 17]. We now show the connection between the Kauffman brackets of L_G and the multivariate Tutte polynomial of G .

Claim 2.3.1 For any scalar d and a set of weights $\mathbf{u} = \{u_e\}$, the following equation holds:

$$\langle L_G \rangle(d, \mathbf{u}) = d^{-|V|} Z_G(d^2, d\mathbf{u}) . \quad (2.17)$$

Proof: The proof is an adaptation of well-known ideas due to Kauffman on the subject of the Jones polynomial in knots theory [16]. Each state σ of the medial graph L_g uniquely defines a subgraph of G whose edges are those with $\sigma_e = 1$. This set of edges is denoted by $A \subseteq E$. At the same time, σ corresponds to a particular opening of the crossings, such that each facet of the subgraph (V, A) is encircled by exactly one loop. Therefore, the number of loops, $|\sigma|$, is exactly the same as the number of facets in (V, A) . Now the number of facets in (V, A) is simply $k(A) + c(A)$, where $c(A)$ is the number of circles in the subgraph. Using the identity $|V| + c(A) = k(A) + |A|$ (see, e.g. [22] Sec. 2)

we have

$$|\sigma| = k(A) + c(A) = 2k(A) + |A| - |V| . \quad (2.18)$$

Therefore,

$$\langle L_G \rangle = \sum_{A \subseteq E} d^{2k(A) + |A| - |V|} \prod_{e \in A} u_e = d^{-|V|} \sum_{A \subseteq E} (d^2)^{k(A)} \prod_{e \in A} (du_e) \quad (2.19)$$

$$= d^{-|V|} Z_G(d^2, d\mathbf{u}) . \quad (2.20)$$

□

To conclude, the Kauffman brackets of L_G are proportional to the Tutte polynomial of G with $q = d^2$ and $\mathbf{v} = d\mathbf{u}$.

2.4 The Jones Polynomial

Another interesting specialization of the multivariate Tutte polynomial of planar graphs is the Jones polynomial, which is a well-known knot invariant that can be defined via the (standard) Kauffman brackets of a link [16]. We define the standard Kauffman brackets, in a manner similar to the one in Sec. 2.3, and then make a simple connection to the Jones polynomial. For clarity, in this section we explicitly identify the Kauffman brackets to be either *standard* or *multivariate*.

Definition 2.4.1 (standard Kauffman brackets) Consider a link L , given by a link diagram. A state σ refers to a choice for every crossing of L from the set $\{ \quad , \quad \}$, of possible openings. We define $|\sigma|$ as the number of loops in the diagram that result from applying the choices of σ to L . Additionally, let σ^+ (resp. σ^-) be the number of crossings of the form \quad for which σ chooses (resp. \quad).

The Kauffman brackets of L are a Laurent polynomial in the variable A , and are defined as the following sum:

$$\langle L \rangle \stackrel{\text{def}}{=} \sum_{\sigma} \langle L | \sigma \rangle , \quad (2.21)$$

where $\langle L | \sigma \rangle$ is defined by

$$\langle L | \sigma \rangle \stackrel{\text{def}}{=} d^{|\sigma| - 1} A^{\sigma^+ - \sigma^-} \quad (2.22)$$

and $d = -A^2 - A^{-2}$.

To define the Jones polynomial, we consider oriented links, namely links with an arrow on each connected component. The connection between the Jones polynomial and the standard Kauffman brackets is given by a property called the *writhe*.

Definition 2.4.2 (Writhe) For an oriented link L and its diagram, assign to each crossing of the type: $\begin{matrix} \nearrow \\ \searrow \end{matrix}$ the value $+1$, and to each crossing that of the type: $\begin{matrix} \searrow \\ \nearrow \end{matrix}$ the value -1 . The writhe of L is the sum over all the crossings of these signs.

Definition 2.4.3 (The Jones polynomial) The Jones polynomial of an oriented link L is defined as

$$V_L(t) = V_L(A^{-4}) = (-A)^{-3w(L)} \langle L \rangle$$

where $w(L)$ is the writhe of the oriented link L , and $\langle L \rangle$ is the brackets' state sum of the link L , ignoring the orientation.

A natural question is whether the standard Kauffman brackets polynomial is a special case of the multivariate one. A medial graph and a link diagram are very similar but not quite the same. For the former, we consider the black and white coloring, and for the latter the topology of the link at hand.

It turns out that if a link has a certain property namely it is alternating, then the two objects are equivalent. An alternating link is link one that has a link diagram with alternating underpasses and overpasses. Let L be a diagram of an alternating link, then if we start by coloring the outside area in black, then all openings that connect the shaded areas are of the same type (either $\begin{matrix} \nearrow \\ \searrow \end{matrix}$ or $\begin{matrix} \searrow \\ \nearrow \end{matrix}$).

Let L be an alternating link diagram, such that the $\begin{matrix} \nearrow \\ \searrow \end{matrix}$ opening will connect all shaded areas, when the outermost region is colored white. Let L_G be the medial graph identical to L except that for each crossing the information as to which strand is on top of the other is disregarded. We want to find the relation between $\langle L_G \rangle(d, u)$ and $\langle L \rangle(d, A)$. Under these assumptions if we set $u_e = A^{-2}$ for each crossing then

$$\frac{A^m}{d} \langle L_G \rangle(d, u) = \langle L \rangle(d, A) \tag{2.23}$$

where m is the total number of crossings in L . We prove this by looking at one state at a time. The connection will follow since both terms are simply a sum over all possible states:

$$\begin{aligned} \frac{A^m}{d} \langle L_G | \sigma \rangle &= \frac{A^m}{d} (d^{|\sigma|} \prod u_e) = \\ &= A^m d^{|\sigma|-1} (A^{-2})^{\sigma^-} \\ &= d^{|\sigma|-1} A^{m-2\sigma^-} \\ &= d^{|\sigma|-1} A^{\sigma^+ - \sigma^-}, \end{aligned}$$

where, as $m = \sigma^+ + \sigma^-$, the last equality holds. The case in which all the shaded areas are connected using the $\begin{matrix} \searrow \\ \nearrow \end{matrix}$ opening is symmetric with $\frac{A^{-m}}{d} \langle L_G \rangle(d, A^2) = \langle L \rangle(d, A)$.

Part I

Initial Attempts for Generalizing *AJL*

Chapter 3

Interpolation Approaches for Generalizing *AJL*

Dealing with polynomials it is an elementary fact that knowing the value of a polynomial, of bounded degree m , at $m + 1$ points is enough to identify the polynomial by interpolation. However, we do not have access to the exact value of the Jones polynomial at any point; moreover, by the results of Jaeger et al. [14] an exact evaluation of the Jones polynomial even at a single point (except for nine points and a curve where the task is trivial) is $\#P$ hard. If it is possible to approximate a polynomial at many points, then it seems natural to think that an approximation of the entire polynomial is possible as well. A naive approach is to perform the Lagrange interpolation as if we had the exact points. The Lagrange interpolation formula is the following:

$$p(x) = \sum_{i=1}^{m+1} \frac{\prod_{j:j \neq i} x - x_j}{\prod_{j:j \neq i} x_i - x_j} y_i \quad (3.1)$$

where the pairs $\{(x_i, y_i)\}_1^{m+1}$ are the evaluations of the polynomial $p(x)$ of degree m at $m + 1$ points.

It will be useful to recall that the maximal degree of the Jones polynomial is easy to bound as it is straightforward to see from its definition that it is $O(m)$, where m is the number of crossings in the link. Additionally we note that the Jones polynomial is in fact a Laurent polynomial in the variable \sqrt{t} ; thus we consider a multiplication of it by $t^{m/2}$ so as to work with a proper polynomial of degree $O(m)$. For clarity we abuse notation and in this section call the resulting polynomial the Jones polynomial as well.

We present convincing evidence that even in between the points where we have approximation Lagrange interpolation does not provide a good approximation. Let us consider a polynomial $p(x)$ of degree $n = 2r$ which we define so that it vanishes at every integer point between 1 and $2r + 1$ except the point r and $p(r) = 1$. We show that this polynomial receives large values in the interval $[1, n + 1]$, specifically we show that $|p(\frac{3}{2})| = 2^{O(n)}$. In Fig. 3.1 one can see an example of the function when $n = 26$.

Figure 3.1: Example of a 26-degree polynomial vanishing at integer points $[1..27]$ except for 13 where it is equal to 1.

$$|p(\mathbf{x})| = \left| \sum_i \frac{\prod_{j:j \neq i}(x - x_j)}{\prod_{j:j \neq i}(x_i - x_j)} y_i \right| = \left| \frac{\prod_{j:j \neq r}(x - j)}{\prod_{j:j \neq r}(r - j)} \right| \quad (3.2)$$

$$= \left| \frac{\prod_{j:j \neq r}(\frac{3}{2} - j)}{\prod_{j:j < r}(r - j) \prod_{j:j > r}(j - r)} \right| \quad (3.3)$$

$$= \left| \frac{\prod_{j:j=1\dots(n+1)}(\frac{3}{2} - j)}{(3/2 - r)((r - 1)!)^2} \right| \quad (3.4)$$

$$> \left| \frac{n!}{\frac{1-n}{2}((r - 1)!)^2} \right| \quad (3.5)$$

$$= 2^{\mathcal{O}(n)} \quad (3.6)$$

The last equality holds since $\left| \frac{(n-1)!}{\frac{n-1}{2}((r-1)!)^2} \right| \approx \binom{n}{n/2} = 2^{\mathcal{O}(n)}$.

Let us denote by $\bar{V}(t)$ the approximation of the Jones polynomial that results from performing Lagrange interpolation, to values with inaccuracy of Δ_{AJL} . We define $p(x)$ an m -degree polynomial $ERR(x) = (V(t) - \bar{V}(t))/\Delta_{AJL}$. This polynomial represents the error made by $\bar{V}(t)$ compare to $V(t)$. By the previous claim it follows that $|ERR(x)|$ can be in the order of $2^{\mathcal{O}(m)}$ and thus the difference between the polynomial is $\Delta_{AJL}2^{\mathcal{O}(m)}$. We consider the approximation of $\bar{V}(t)$ to be almost trivial since by direct properties of the Jones polynomial it is possible to bound its values by $2^{\mathcal{O}(m)}$ at every fixed interval.

We do not explore in depth the quantity of this approximation since in other results presented in this work we succeeded in achieving a much better approximation scale. Namely, an error in the scale of $2^{\mathcal{O}(n)}$ where n is the number of strands of the braid (which in every interesting case is much smaller than the number of crossings m).

It is clear that if we hope to gain any approximation for the rest of the polynomial then we have to use the information that the approximation algorithm provides us more wisely. Arora and Khot [5], have shown that the task at hand is indeed possible under certain conditions. We briefly review their results and then proceed to use them.

3.1 Extracting a Polynomial from Noisy Data

We begin by formulating a computational problem: We are given pairs of points $(x_1, y_1) \dots (x_k, y_k) \in [-1, 1] \times [-1, 1]$, a noise parameter $\delta > 0$, a degree bound m and a threshold ρ . We consider all polynomials h of degree at most m such that:

$$|h(x_i) - y_i| \leq \delta \quad \text{for at least } \rho \cdot m \text{ indexes}$$

An algorithm that finds a polynomial that is δ close in the l_∞ norm to all of the above polynomials is presented in Ref. [5]:

Theorem 3.1.1 *Fitting Algorithm (Thm. 1 from Ref. [5])* *There is a polynomial time algorithm that solves the above problem when $k = \Omega(m^2/\delta)$ points are used.*

At first glance this interesting and elegant result seems promising. When considering a polynomial whose values are not bounded to the interval $[-1, 1]$ we realize that we must rescale it by dividing it by this largest value. Before we apply these results to the Jones polynomial, we first note that although this result is given for polynomials over the real numbers, it can be generalized to the complex number. We do not describe here such a generalization due to the poor results which can be achieved in our context using this approach. For this reason we ignore in this section the fact the Jones polynomial takes complex values.

Dealing with the Jones polynomial we realize that no bound on its values is known except of the trivial bound which is exponential in its degree, namely $2^{\mathcal{O}(m)}$. We therefore rescaling the polynomial by dividing it by its bound. The approximation error guaranteed by *AJL* is possible but then from Thm. 3.1.1 it follows that it is possible to approximate the rescaled polynomial $V(t)/2^{\mathcal{O}(m)}$ up to $\frac{\Delta}{2^{\mathcal{O}(m)}}$ using $\Omega(m^2/\frac{\Delta}{2^{\mathcal{O}(m)}})$ samples generated from the *AJL* algorithm. Clearly trying to apply this approach *does not* result in an efficient algorithm since the number of samples is exponential.

Chapter 4

Graph Tensor Approach for Generalizing AJL

For simplicity we restrict ourselves in this section to alternating links (see definition in Sec. 2.4). In this section we discuss the standard Tutte polynomial $T_G(x, y)$ as defined in Sec. 2.2.1 and use the known relation between the Tutte polynomial and the Jones polynomial of an alternating link [17]:

$$V_L(t) = (-1)^w t^{\frac{m-n+3w}{4}} T_G(-t, -\frac{1}{t}) \quad (4.1)$$

where G is obtained from L by connecting shaded areas in the medial graph of L , as described in Sec. 2.4; w is the writhe of L and m, n are, respectively, the number of edges and vertices in G .

In order to present and prove our results we need some further definitions: For a graph G and an edge $f \in G$ the graph that results from removing the edge f from G by $G - f$ and the graph that results from identifying both endpoints of f while maintaining the original connectivity of G by G/f .

Definition 4.0.1 (graph 2-sum) *Given graphs G and H and edges $e \in G, f \in H$, we denote by $G_e \oplus H_f$ the 2-sum of G and H with respect to e and f . The graph $G_e \oplus H_f$ is obtained from G and H by first identifying the edges e and f and then removing this edge from the combined graph.*

For this definition to hold we must either have the graph H symmetric with respect to f , or consider oriented edges f and e . We note, however, that we will only use graphs that are clearly symmetric in the right-hand side of the 2-sum. An example for the 2-sum can be seen in Fig. 4.1.

Definition 4.0.2 (graph tensor) *For graphs G and H and an edge $f \in H$ we denote by $G \otimes H_f$ the tensor of the graphs, which is obtained by performing a 2-sum between G and H_f with all edges $e \in G$.*

We focus on graph tensors of two types which will result in two graph transformations that we call k -stretch and k -thickening. The k -stretch of a graph G will simply be $G \otimes C_{k+1}$ where C_l is the cycle of length l . We call this the k -stretch of G since every edge of G is transformed to a path of length k

Figure 4.1: Example for 2-sum of graphs

in $G \otimes C_{k+1}$. Similarly, the k -thickening of G is $G \otimes B_{k+1}$, where B_l is a graph with two vertices and l parallel edges between them.

A crucial property of the Tutte polynomial that was used by Jaeger et al. [14], is the known relation between the Tutte polynomial of G and $G \otimes H_f$.

$$T_{G \otimes H}(x, y) = P_H(x, y)^{m-n+1} Q_H(x, y)^{n-1} T_G(X, Y) \quad (4.2)$$

where H is assumed to be connected, m is the number of edges in G and

$$X = \frac{(x-1)P_H(x, y)}{Q_H(x, y)} + 1, \quad Y = \frac{(y-1)Q_H(x, y)}{P_H(x, y)} + 1 \quad (4.3)$$

Additionally, P_H , Q_H are polynomials that are determined by the equations:

$$(x-1)P_H(x, y) + Q_H(x, y) = T_{H-f} \quad (4.4)$$

$$P_H(x, y) + (y-1)Q_H(x, y) = T_{H/f} \quad (4.5)$$

Note that taking the tensor product with any graph H defines a transformation $(x, y) \rightarrow (X, Y)$ on the plane; however, for any choice of H , it holds that $(X-1)(Y-1) = (x-1)(y-1)$. In other words the taking the graph tensor transforms the hyperbole $(x-1)(y-1) = q$ to itself. This fact is used directly in the paper of [14] but in this work we do not elaborate on this property any more.

Finally, we note that when we perform a k -thickening or k -stretch of a graph, i.e., a tensor product with the graphs C_{k+1}, B_{k+1} the form of the polynomials P, Q is quite simple to calculate [14].

$$\begin{aligned} P_{C_{k+1}} &= \frac{x^k - 1}{x - 1} & Q_{C_{k+1}} &= 1 \\ P_{B_{k+1}} &= 1 & Q_{B_{k+1}} &= \frac{y^k - 1}{y - 1} \end{aligned} \quad (4.6)$$

4.1 AJL Based Quantum Algorithm for Points in the Tutte Plane

The above definitions and observations suggest a way to approximate more points in the Tutte plane, using the *AJL* algorithm as a black box.

Given a graph G , compute the k -stretch (or k -thickening) of G classically to obtain a graph G' . Run the *AJL* algorithm on G' in order to approximate the Jones polynomial at the r 'th root of the unity that by Eq. (4.1) is $T_{G'}(-e^{2\pi i/r}, -e^{-2\pi i/r})$. According to Eq. (4.2) this value is equivalent to $T_G(X, Y)$ up to a factor of

$$[P_{C_{k+1}}(x, y)]^{m-n+1} [Q_{C_{k+1}}(x, y)]^{n-1} = \left[\frac{(-e^{2\pi i/r})^k}{-e^{2\pi i/r} - 1} \right]^{m-n+1} \quad (4.7)$$

for the k -stretch; and

$$[P_{B_{k+1}}(x, y)]^{m-n+1} [Q_{B_{k+1}}(x, y)]^{n-1} = \left[\frac{(-e^{-2\pi i/r})^k}{-e^{-2\pi i/r} - 1} \right]^{n-1} \quad (4.8)$$

for the k -thickening.

The points X, Y that can be approximated via a k -stretch of the graph :

$$X_{k,r} = (-e^{2\pi i/r} - 1) \frac{(-e^{2\pi i/r})^k - 1}{-e^{2\pi i/r} - 1} + 1 = -e^{2\pi i k/r} \quad (4.9)$$

$$Y_{k,r} = 1 + (y-1) \frac{x-1}{x^k - 1} = 1 + (-e^{-2\pi i/r} - 1) \frac{e^{2\pi i/r} + 1}{e^{2\pi i k/r} + 1} \quad (4.10)$$

Similarly, for the k -thickening $Y_{k,r} = -e^{-2\pi i k/r}$ and $X_{k,r} = 1 + (-e^{2\pi i/r} - 1) \frac{e^{-2\pi i/r} + 1}{e^{-2\pi i k/r} + 1}$. We note that when using the r 'th root of the unity, then we do not use k -stretch or k -thickening when $k \geq r$, since the points achieved in such a manner can also be reached using $k' = k \bmod r$.

Figure 4.2: A 2-thickening of the left most braid results in the addition of two more strands to the braid

4.2 Approximation Scale

We now have an approximation algorithm for any pair X, Y as mentioned above, but what is the scale of this approximation? It turns out that it is not very good. It follows from the above discussion that the overall approximation factor is, Δ_{AJL}/F where Δ_{AJL} is the approximation scale guaranteed by the AJL algorithm and F is the factor that translates between values of the Tutte polynomials of the original and modified graphs (see Eqs.(4.7,4.8)).

We examine more closely the definition of F and the graph resulting from a graph tensor transformation. We focus on the k -stretch of a graph, for which $F = \left(\frac{(-e^{2\pi i/r})^k}{-e^{2\pi i/r} - 1}\right)^{m-n+1}$. Using the fact that $r \geq 5$:

$$|F| = \left| \left(\frac{(-e^{2\pi i/r})^k}{-e^{2\pi i/r} - 1} \right)^{m-n+1} \right| \quad (4.11)$$

$$= \left(\frac{1}{|-e^{2\pi i/r} - 1|} \right)^{m-n+1} \quad (4.12)$$

$$= \left(\frac{1}{2 + 2 \cos(2\pi/r)} \right)^{m-n+1} \quad (4.13)$$

$$< \left(\frac{2}{3} \right)^{m-n+1} \quad (4.14)$$

The approximation scale is in fact even larger because Δ_{AJL} is exponentially large (in the number of crossings). To see this we consider the example in Fig. 4.2. Let m be the number of crossings in the braid equivalent to the original graph, and let m_0 (m_1) be the number of crossings of the form σ_{2j} (resp. σ_{2j-1}) for some j . It is not hard to see that the number of strands grows by $2(k-1)m_0$ after a k -stretch and by $2(k-1)m_1$ after a k -thickening. We recall that the approximation scale guaranteed by the AJL algorithm depends exponentially in the number of strands, namely $2^{\mathcal{O}(n)}$. It follows that Δ_{AJL} is in the order of $2^{\mathcal{O}(n+(k-1)m)}$ and thus the overall approximation scale is $2^{\mathcal{O}(m)}$.

For completeness, we note that any graph H can be used to produce $G \otimes H$ and furthermore, this process can be repeated an arbitrary number of times. Although many other points can be approximated this way, the approximation scale will deteriorate in such a way that the approximation task will eventually become trivial.

4.3 BQP Hardness of Points in the Tutte Plane

It is not surprising that using the same technique we can find new points in the Tutte plane that are BQP-hard to approximate. We use as “base” BQP-hard points $(-e^{2\pi i/r}, -e^{-2\pi i/r})$ for $k \geq 5$, which are known to be hard to approximate due to the work presented in [11] and [1]. Then the new BQP-hard points are exactly point (x, y) such that there exists a graph H that

$$T_{G \otimes H}(x, y) = P_H(x, y)^{m-n+1} Q_H(x, y)^{n-1} T_G(X, Y) \quad (4.15)$$

for (X, Y) equal to one of the known base points. Let us denote $(X_{k,r}^{-1}, Y_{k,r}^{-1})$ as the pair of points that using a k -stretch transforms them to $(-e^{2\pi i/r}, -e^{-2\pi i/r})$. It is possible but tedious to write down explicitly the formulas for these points similarly to Eqs. (4.9, 4.10) however we do not present them here. We note that the approximate scale for which it is BQP-hard to approximate these points must be chosen appropriately in order to match the known scale of the “base points” and the conversion polynomials Q_H and P_H .

Part II

Generalizing *AJL* via Representations

Chapter 5

The Generalized Temperley-Lieb Algebra $GTL(d)$

The $GTL(d)$ algebra is generated by diagrams of the form shown in Fig. 5.1. Its precise definition is as follows:

Definition 5.0.1 (The $GTL(d)$ algebra) *The $GTL(d)$ algebra is an infinite dimensional algebra that is defined for every scalar d . Its basis elements are represented by classes of diagrams that are made from a finite number of strands that connect n lower pegs to m upper pegs. The diagrams must not contain either any crossings or loops, but they may contain local minima/maxima. Two diagrams belong to the same class if they are topologically equivalent or if one diagram can be obtained from the other by adding a number of straight strands to its right. See Fig. 5.1. We denote algebra elements by capital calligraphic letters such as $\mathcal{T}, \mathcal{A}, \mathcal{B}, \dots$ and diagrams by Gothic letters $\mathfrak{T}, \mathfrak{A}, \mathfrak{B}, \dots$. Note that every $GTL(d)$ basis element is represented by an infinite number of diagrams, but every diagram represents only one basis element.*

Product Rule: *For every two basis elements $\mathcal{T}_1, \mathcal{T}_2$, the product $\mathcal{T}_3 = \mathcal{T}_2 \cdot \mathcal{T}_1$ is defined by the diagram that we obtain by placing a diagram that represents \mathcal{T}_2 on top of a diagram that represents \mathcal{T}_1 and matching their pegs starting from the left. If the number of lower pegs in \mathcal{T}_2 is different than the number of upper pegs in \mathcal{T}_1 we pad one of them with straight strands until they match.*

Identity Element: *From the above definition it is easy to see that the identity element of the algebra, \mathcal{I} , corresponds to diagrams with only straight strands, in which the i 'th bottom peg is connected to the i 'th upper peg.*

Relations:

- *Two isotopic elements in the $GTL(d)$ algebra are equal.*

Figure 5.1: Three different diagrams that represent the *same* $GTL(d)$ basis element. The first two have three in-pegs and five out-pegs and are isotopic to each other. The third one has five in-pegs and seven out-pegs and is a result of adding two straight strands to the right of the second diagram.

Figure 5.2: An illustration of the elementary tangles (from left to right) \mathcal{A}_3 , \mathcal{B}_3 and $\mathcal{C}_3(u)$.

- Two elements in the $GTL(d)$ algebra are equal if a diagram of one element can be obtained from a diagram of the other element padded with straight strands at its right side.
- A single loop is equal to the scalar d times the identity \mathcal{I} . The parameter d is a (complex) number that is a fixed parameter of the algebra and is called the loop value.
- An element in the algebra that contains an empty loop, is equal to the same element, except that the loop is removed, and the result is multiplied by the loop value d .

We extend the $GTL(d)$ algebra to linear combinations of basis elements by linear extensions of the product rule.

5.1 Tangles and Elementary Tangles

It is convenient to talk about *tangles*. These are diagrams that are similar to those that represent basis elements of the $GTL(d)$ algebra, except that they may also contain loops and weighted crossings. We use them to denote certain linear combinations of basis elements in $GTL(d)$:

Loops: In accordance with Definition. (5.0.1), a diagram with ℓ loops denotes the $GTL(d)$ basis element of the corresponding diagram without the loops, multiplied by a factor of d^ℓ .

Crossings: We define a crossing in accordance with the definition of the Kauffman brackets. For a tangle \mathfrak{T} with exactly one crossing and a complex weight u . Consider the black-and-white coloring of the tangle, starting from its leftmost-area, which is always painted white. We then look at the two possible ways of opening the crossing: connecting or disconnecting its two shaded areas. Let \mathcal{X} be the $GTL(d)$ element that corresponds to the diagram in which the shaded areas are connected, and let $\mathcal{Y} \in GTL(d)$ correspond to the diagram in which the shaded areas are disconnected. We then use the diagram \mathfrak{T} to denote the following $GTL(d)$ element:

$$\mathcal{T} = u\mathcal{X} + \mathcal{Y} . \quad (5.1)$$

Of course, if the tangle contains more than one crossing, we can apply the above procedure on all crossings recursively, thereby associating \mathcal{T} with a linear combination of basis elements of $GTL(d)$ and loops. It is easy to convince oneself that this definition is independent of the order in which we open each crossing, and so the mapping from tangles with weighted crossings to $GTL(d)$ is well defined.

Next, we define the notion of *elementary tangles*, which are simple tangles that can generate all tangles.

Definition 5.1.1 (Elementary tangles) *A tangle is called elementary if it is represented by one of the following diagrams (up to isotopy and/or addition of straight strands at the right)*

1. **Cup \mathcal{A}_i** : The cup diagram \mathcal{A}_i is the diagram that is identical to the identity \mathcal{I} diagram except for the upper $i, i+1$ pegs, which are connected to each other. The diagram thus contains a minimum after the $i-1$ strand.

2. **Cap** \mathcal{B}_i : Similar to the cup diagram, \mathcal{B}_i is identical to the identity diagram except for having a maximum that is created by connecting the lower $i, i + 1$ pegs.
3. **Crossing** $\mathcal{C}_i(u)$: The crossing diagram $\mathcal{C}_i(u)$ contains a crossing with weight u between the $i, i + 1$ strands, while all the other strands are trivial.

An illustration of these three diagrams is given in Fig. 5.2.

Note that the three elementary tangles are not independent. Indeed, it is easy to see that for odd i , $\mathcal{C}_i(u) = u\mathcal{I} + A_i \cdot \mathcal{B}_i$, whereas for even i , $\mathcal{C}_i(u) = \mathcal{I} + uA_i \cdot \mathcal{B}_i$. It turns out that it is only the cup and cap tangles that are needed to generate the full $GTL(d)$ algebra. Nevertheless, we will find the crossing operator extremely useful to describe the algorithm and the universality result.

For a tangle that corresponds to a basis element we can define the number of *in-pegs* and *out-pegs* to be, respectively, the number of lower and upper pegs in the diagram (see Fig. 5.1). This definition can be extended to general tangles, since by opening a crossing we obtain two diagrams with the same number of in-pegs and the same number of out-pegs. We note that these numbers are well defined for tangles but not for $GTL(d)$ elements, since the definition of the $GTL(d)$ elements allows adding a number of straight strands to the right without changing the element.

5.2 Connection between the Kauffman Brackets and the $GTL(d)$ Algebra.

Our goal is to calculate the Kauffman brackets $\langle L_G \rangle(d, \mathbf{u})$ of a medial graph using the $GTL(d)$ algebra. We do this by defining a map Ψ_d that associates a medial graph with an element in $GTL(d)$. This map was essentially defined in the previous section. We can simply consider an embedding of L_G in \mathbb{R}^2 as a tangle. Then, according to the previous section, $\Psi_d(L_G)$ is a linear combination of basis elements of $GTL(d)$, and hence is an element of $GTL(d)$.

Moreover, as noted in the previous section, tangles have a well-defined number of in-pegs and out-pegs. It is easy to see that $\Psi_d(L_G)$ has zero in-pegs and zero out-pegs, and is actually equal to a scalar times \mathcal{I} . This is because as we open up its crossings, we end up with tangles that contain only loops, which are proportional to \mathcal{I} by the loop-value relation. It turns out that the coefficient in front of the \mathcal{I} element is exactly the scalar we are looking for!

Proposition 5.2.1 $\Psi_d(L_G) = \langle L_G \rangle(d, \mathbf{u})\mathcal{I}$.

Proof: Opening all the crossings of the L_G element in accordance with Eq. (5.1), we are left with a sum of $2^{|E|}$ terms. Each term corresponds to a possible opening of all the crossings, i.e., to a state σ of the Kauffman brackets, and as it is a diagram with only loops it must be proportional to \mathcal{I} . We claim that the proportionality factor is exactly $\langle L_G | \sigma \rangle$. Assume that the opened diagram contains ℓ loops and corresponds to a state σ . Then the proportionality factor is d^ℓ times u_e for every crossing with $\sigma_e = 1$, because such an opening joins the shaded areas and is therefore multiplied by a u_e factor according to Eq. (5.1). Summing up all the terms we obtain the desired result. \square

This shows that if we want to approximate the Tutte polynomial, or rather, $\langle L_G \rangle(d, \mathbf{u})$, it suffices to approximate the coefficient in front of \mathcal{I} in $\Psi_d(L_G)$.

5.3 Connection with Representations of $GTL(d)$

To approximate the above coefficient, we consider representations of the $GTL(d)$ algebra. We define the representation *for diagrams* rather than for abstract $GTL(d)$ elements, as it would depend on the particular in-pegs and out-pegs of the diagrams.

Definition 5.3.1 Let $\{H_n\}$ be a series of Hilbert spaces (which do not have to be different from each other). A map ρ that maps tangles to linear operators over the Hilbert spaces is said to be a representation of $GTL(d)$ if it maps every tangle diagram \mathfrak{T} with n in-pegs and m out-pegs to a linear operator $\rho(\mathfrak{T}) : H_n \rightarrow H_m$ such that:

- $\rho(\mathfrak{T})$ depends only on the isotopy class of \mathfrak{T}
- For diagram \mathfrak{T} with n in-pegs and m out-pegs, $\rho(\mathfrak{T})$ is a linear transformation from H_n to H_m .
- **The multiplicative structure is preserved:** If \mathfrak{T}_1 has n in-pegs and m out-pegs and \mathfrak{T}_2 has m in-pegs and ℓ out-pegs then $\rho(\mathfrak{T}_2 \cdot \mathfrak{T}_1) = \rho(\mathfrak{T}_2) \cdot \rho(\mathfrak{T}_1)$.
- **The additive structure is preserved:** If $\mathfrak{T}_1, \mathfrak{T}_2$ represent the $GTL(d)$ elements $\mathcal{T}_1, \mathcal{T}_2$ and \mathfrak{T}_3 represents the $GTL(d)$ element $c_1\mathcal{T}_1 + c_2\mathcal{T}_2$, then $\rho(\mathfrak{T}_3) = c_1\rho(\mathfrak{T}_1) + c_2\rho(\mathfrak{T}_2)$.

Previously, in Definition. (6.4.1) we presented a particular representation, the *path model representation*. Nevertheless, any representation will do to evaluate the desired coefficient:

Claim 5.3.1 If ρ is a representation of $GTL(d)$, then $\rho(\Psi_d(L_G)) = \langle L_G \rangle(d, \mathbf{u})\mathbb{1}$.

Proof: This is true because any representation will take the identity \mathcal{I} to the identity operator $\mathbb{1}$. \square

We can thus perform the approximations we want on the image of a representation of the $GTL(d)$ algebras.

Remark: We note an important difference between the connection that is made here between the representation of the $GTL(d)$ algebra and the Kauffman brackets versus a similar connection that was used in [3]. In that paper, a similar result was true for $TL_n(d)$, a restricted version of $GTL(d)$ in which the number of strands was finite and fixed. The representation, however, was required to exhibit the Markov property for the connection to the Kauffman brackets to hold. Here, due to the fact that a single element of the algebra is associated with a class of diagrams that have different in-peg and out-peg numbers, no such property is needed, and any representation will do. This is an advantage of working with the $GTL(d)$ algebra.

Chapter 6

The Path-Model Representation of $GTL(d)$

We use the path-model representations. They are essentially the representations used in Ref. [3], except here we also allow non-Hermitian representations and consider the infinite rather than the finite algebra. In addition, in accordance with Definition. (5.3.1), the representation that we describe is defined for tangles, which are diagrams, rather than for the $GTL(d)$ elements that are represented by these diagrams. The reason is that the definition depends on the number of in-pegs and out-pegs of the diagram; hence different diagrams that represent the same $GTL(d)$ element might be mapped to different operators. The representation, however, does not depend on the particular geometry of the diagram – only on its topology. Specifically, for each diagram we define a linear transformation with *different* domain and target spaces, the dimensions of which, depend on the number of in-pegs and out-pegs of the diagram. We begin by giving motivation to the definition and then move to the exact definitions.

6.1 Diagrams as Operators over Paths on an Auxiliary Graph

Let $\mathcal{T} \in GTL(d)$ be a basis element that is represented by a diagram \mathfrak{T} with m in-pegs and n out-pegs.

In order to associate the diagram with an operator, we draw \mathfrak{T} and confine it to a rectangular box, thereby dividing the box into a set of disjoint regions. Then the n lower-pegs of the diagram, which are connected to the lower edge of the box, divide it into $n + 1$ gaps, while the upper pegs divide the upper edge of the box into $m + 1$ gaps. We would like to treat \mathfrak{T} as an operator taking a sequence of labels of the lower gaps to a sequence of labels of the upper gaps. For this, we use the notion of an *auxiliary graph* F and allow the labels of the different regions of the boxed diagram to be vertices of F . Furthermore, we restrict the labeling of the diagram to be such that two adjacent regions are labeled by adjacent vertices in F . We observe that under those conditions, the sequence of labels of the upper gaps, as well as that of the lower gaps, is a *path* on the auxiliary graph F , and that the diagram can be seen as a transformation that takes the path of the lower gaps to the path of the upper gaps. As an example, see Fig. 6.1.

We thus view \mathfrak{T} (and through it \mathcal{T}) as an operator on paths on F . We can now formally define the Hilbert space of paths.

6.2 The Hilbert Space of Paths on F

For simplicity, we denote the vertices of the auxiliary connected graph F by $\{1, 2, 3, \dots\}$, and define a *family* of Hilbert spaces $\{H_n\}$ that correspond to paths of length n of F as follows:

Figure 6.1: Labeling of a diagram according to the vertices of an auxiliary graph F .

Definition 6.2.1 (The spaces H_n) For any integer $n \geq 0$, the n -step path space H_n of an auxiliary graph F is a Hilbert space that is defined as follows: let $p = \{p_1, p_2, \dots, p_{n+1}\}$ denote an n -step path on F that starts at the vertex p_1 and ends at the vertex p_{n+1} . We specify that $p_1 = 1$ and that p_i be adjacent to p_{i+1} in F (denoted as $p_i \sim p_{i+1}$). Then we associate every path p with a vector $|p\rangle$ and consider the set $\{|p\rangle\}$ of all n -step paths to be an orthonormal basis for the space H_n .

Note that H_0 is a one dimensional space that is spanned by the vector $|1\rangle$.

6.3 Compatible Paths

Let \mathfrak{T} be a diagram with m in-pegs and n out-pegs that represents a basis element of $GTL(d)$. We associate \mathfrak{T} with a transformation $\rho(\mathfrak{T}) : H_m \rightarrow H_n$, by defining $\langle p' | \rho(\mathfrak{T}) | p \rangle$ for every n -step path $|p\rangle = |p_1, \dots, p_{n+1}\rangle$ and m -step path $|p'\rangle = |p'_1, \dots, p'_{m+1}\rangle$. The term $\langle p' | \rho(\mathfrak{T}) | p \rangle$ will be zero unless the two paths are compatible in the following sense:

Definition 6.3.1 (compatible paths) Let \mathfrak{T} be a diagram of a basis element, or a basis element with closed loops, with n in-pegs and m out-pegs. The n -step path p and the m -step path p' are then called compatible with respect to \mathfrak{T} if, when labeling \mathfrak{T} 's lower gaps with the vertices of the path p and the upper gaps with vertices of the path p' , the result is a diagram where all connected surfaces have the same label. The labeling is done from left to right with respect to the order of the vertices in each path.

It is clear that the definition depends only on the isotopy class of the diagram. As an example for of definition, the upper path and lower path in Fig. 6.1 are compatible.

As noted before, we define the value of $\langle p' | \rho(\mathfrak{T}) | p \rangle$ to be zero if the two paths are not compatible. Compatible paths are defined in the next section.

6.4 The Value $\langle p' | \rho(\mathfrak{T}) | p \rangle$ for Compatible Paths

Consider a diagram \mathfrak{T} of a basis element and two compatible paths p, p' with respect to it. As \mathfrak{T} does not contain any loops, it is easy to see that p, p' determine the labeling of all areas in \mathfrak{T} uniquely. To define $\langle p' | \rho(\mathfrak{T}) | p \rangle$, we consider all the local minima and maxima of \mathfrak{T} . The area near each such point is marked by two labels – adjacent vertices from F . A local minimum labeled ℓ from above and k from below is associated with a constant $a_{\ell,k}$, and a maximum with the same labeling is associated with $b_{\ell,k}$ (see Fig. 6.2).

We will soon assign values to $a_{\ell,k}$ and $b_{\ell,k}$ and discuss their properties. As an example, the minima and maxima of the diagram in Fig. 6.1 are associated (from top to bottom) with the variables: $a_{4,2}, a_{2,1}, b_{1,2}, a_{1,2}$ and $b_{2,3}$. We define $\langle p' | \rho(\mathfrak{T}) | p \rangle$ to be the product of all the constants that appear in the diagram. The definition of $\rho(\cdot)$ can thus be summarised as follows:

Definition 6.4.1 (The path-model representation) Let \mathfrak{T} be a diagram that represents a basis element (i.e., it has no loops and crossings), and let p, p' be two compatible paths with respect to \mathfrak{T} .

Figure 6.2: Local minimum and maximum, associated with $a_{k,\ell}$ and $b_{k,\ell}$ respectively.

Then p, p' determine a unique labeling of \mathfrak{T} and $\langle p | \rho(\mathfrak{T}) | p' \rangle$ is the product over all local maximums and minimums of \mathfrak{T} of the corresponding $a_{\ell,k}$ and $b_{\ell,k}$ coefficients that are so determined.

For non compatible p, p' , $\langle p | \rho(\mathfrak{T}) | p' \rangle \stackrel{\text{def}}{=} 0$.

6.5 Extending the Definition beyond Basis Elements

Thus far we have only defined $\rho(\cdot)$ for diagrams of basis elements. We now extend the definition to tangles, which is easily achieved by linearity. Recall that the tangle diagrams have a well-defined number of in-pegs and out-pegs, which is equivalent to saying that they are a linear combination of diagrams of basis elements with the *same* number of in-pegs and the *same* number of out-pegs. Consequently, the operators that represent these diagrams have the same domain and range spaces, so we can define the representation of the tangle's diagram by linearity.

This defines $\rho(\mathfrak{T})$ for all tangle diagrams and satisfies the additivity requirement in Definition. (5.3.1). For $\rho(\cdot)$ to be a $GTL(d)$ representation according to Definition. (5.3.1), we still need to verify that it depends only on the isotopy class of the diagrams and that it preserves the multiplicative structure of the algebra. This is done in the next section.

6.6 Conditions on $a_{k,\ell}, b_{k,\ell}$ for ρ to Be a Representation

To show that $\rho(\cdot)$ is a representation of the $GTL(d)$ algebra we must assert that isotopic elements define the same transformation – after all, our definition of $\rho(\mathfrak{T})$ relied on a particular geometrical representation of \mathfrak{T} . Then, we must show that the representation preserves the product rule, i.e., $\rho(\mathfrak{T}_2 \mathfrak{T}_1) = \rho(\mathfrak{T}_2) \rho(\mathfrak{T}_1)$. The following claim provides sufficient conditions for this to hold.

Claim 6.6.1 *The following two conditions guarantee that $\rho(\cdot)$ is a representation of $GTL(d)$:*

$$\forall \ell, k : \quad b_{\ell,k} \cdot a_{\ell,k} = 1 , \quad (6.1)$$

$$\forall k : \quad \sum_{\ell: \ell \sim k} a_{\ell,k} b_{k,\ell} = d , \quad (6.2)$$

Here, the sum $\sum_{\ell: \ell \sim k}$ is over all vertices ℓ in the auxiliary graph that are adjacent to k .

Proof: To show that two isotopic elements are given the same image by $\rho(\cdot)$, we use the fact that two isotopic elements are linked by a series of moves that eliminate or create a pair of a local maximum and a local minimum. Accordingly it will suffice to prove invariance for a move that creates or eliminates a single pair.

Let \mathfrak{T}' be a diagram that results from \mathfrak{T} by removing a single maximum/minimum pair. A pair of terms $b_{\ell,k}, a_{\ell,k}$ will be missing from the expression for $\langle p' | \rho(\mathfrak{T}') | p \rangle$. Now, by using Eq. (6.1) we have $b_{\ell,k} \cdot a_{\ell,k} = 1$ and thus $\rho(\mathfrak{T}') = \rho(\mathfrak{T})$.

Next, we prove that $\rho(\cdot)$ is a homomorphism. Given two diagrams $\mathfrak{T}_1, \mathfrak{T}_2$ that represent basis elements, we would like to prove that

$$\rho(\mathfrak{T}_2 \cdot \mathfrak{T}_1) = \rho(\mathfrak{T}_2) \cdot \rho(\mathfrak{T}_1) . \quad (6.3)$$

The diagram $\mathfrak{T}_2 \cdot \mathfrak{T}_1$ can either represent a basis element or contain some loops. Assume the first case and let the paths p, p' be compatible with it. Then there is only one path p^* , on the border between the two diagrams, such that p, p^* are compatible with \mathfrak{T}_1 and p^*, p' are compatible with \mathfrak{T}_2 . Therefore, if $\sum_{p''}$ denotes the summation over all possible paths on the border between the diagrams, then

$$\sum_{p''} \langle p' | \rho(\mathfrak{T}_2) | p'' \rangle \langle p'' | \rho(\mathfrak{T}_1) | p \rangle = \langle p' | \rho(\mathfrak{T}_2) | p^* \rangle \langle p^* | \rho(\mathfrak{T}_1) | p \rangle = \langle p' | \rho(\mathfrak{T}_2 \mathfrak{T}_1) | p \rangle . \quad (6.4)$$

The first equality is because p^* is the only path that is compatible with both p' and p ; the second equality follows from the definition of $\langle p' | \rho(\mathfrak{T}_2) | p^* \rangle$ and $\langle p^* | \rho(\mathfrak{T}_1) | p \rangle$: they are products of the $a_{k,\ell}, b_{k,\ell}$ coefficients that correspond to the labeling induced by p, p' . Multiplying them gives the product of all the $a_{k,\ell}, b_{k,\ell}$ coefficients in the composite diagram of $\mathfrak{T}_2 \cdot \mathfrak{T}_1$.

Assume now that $\mathfrak{T}_2 \cdot \mathfrak{T}_1$ contains one loop that, by isotopy, has only one maximum and one minimum, and let \mathfrak{T}_3 be equal to $\mathfrak{T}_2 \cdot \mathfrak{T}_1$ without the loop (hence $d\mathcal{T}_3 = \mathcal{T}_2 \cdot \mathcal{T}_1$ for the corresponding $GTL(d)$ elements). There is more than one labeling that is compatible with p, p' . Consequently, the sum

$$\sum_{p''} \langle p' | \rho(\mathfrak{T}_2) | p'' \rangle \langle p'' | \rho(\mathfrak{T}_1) | p \rangle , \quad (6.5)$$

is equal to the summation over all possible labelings of the diagram $\mathfrak{T}_2 \cdot \mathfrak{T}_1$ that are compatible with p, p' . All are identical except for the label of the internal region of the loop. Therefore if k is the label of the external region of the loop, and ℓ is the label of the internal region, then we may write

$$\sum_{p''} \langle p' | \rho(\mathfrak{T}_2) | p'' \rangle \langle p'' | \rho(\mathfrak{T}_1) | p \rangle = \langle p' | \rho(\mathfrak{T}_3) | p \rangle \sum_{\ell: \ell \sim k} a_{\ell,k} b_{k,\ell} . \quad (6.6)$$

By Eq. (6.2), the RHS of the equation is equal to $d \langle p' | \rho(\mathfrak{T}_3) | p \rangle$, and as $\mathcal{T}_1 \mathcal{T}_2 = d\mathcal{T}_3$ for the corresponding $GTL(d)$ tangles, we get

$$\sum_{p''} \langle p' | \rho(\mathfrak{T}_2) | p'' \rangle \langle p'' | \rho(\mathfrak{T}_1) | p \rangle = \langle p' | \rho(\mathfrak{T}_2 \cdot \mathfrak{T}_1) | p \rangle , \quad (6.7)$$

as required. The case of more than one loop follows easily by similar arguments. \square

6.7 Setting $a_{k,\ell}, b_{k,\ell}$: Path Representations for all d

In the previous section we proved that given an auxiliary F and a corresponding set of coefficients $a_{k,\ell}, b_{k,\ell}$ that satisfy the conditions in Definition. (6.4.1) with respect to d , we can construct a representation of the $GTL(d)$ algebra. We now show how to find such coefficients for every complex d .

Consider the graph in Fig. 6.3, which is a single-sided infinite line. Denote this graph by F_∞ and let M_∞ be its adjacency matrix. Given any complex d , we define the infinite-dimensional vector $\bar{\pi} = (\pi_1, \pi_2, \pi_3, \dots)$, as follows:

$$\pi_1 = 1 , \quad (6.8)$$

$$\pi_2 = d , \quad (6.9)$$

$$\pi_3 = d^2 - 1 , \quad (6.10)$$

$$\vdots \quad (6.11)$$

$$\pi_n = d\pi_{n-1} - \pi_{n-2} , \quad (6.12)$$

$$\vdots$$

Figure 6.3: The auxiliary graph F_∞ is a single-sided infinite line graph.

It is easy to see that $\bar{\pi}$ is an eigenvector of M_∞ with an eigenvalue d . When we apply the adjacency matrix M_∞ as an operator on $\bar{\pi}$ we get $(M_\infty \bar{\pi})_1 = \pi_2 = d = d\pi_1$, and for all $n \geq 2$: $(M_\infty \bar{\pi})_n = \pi_{n-1} + \pi_{n+1} = \pi_{n-1} + (d\pi_n - \pi_{n-1}) = d\pi_n$.

We use $\bar{\pi}$ to define the path representation for almost all values of d :

Definition 6.7.1 *Let d be such that in the above definition all coordinates of $\bar{\pi}$ are non vanishing. For such d , the path representation of $GTL(d)$ using the graph F_∞ is defined by:*

$$a_{\ell,k} \stackrel{\text{def}}{=} \sqrt{\frac{\pi_\ell}{\pi_k}}, \quad (6.13)$$

$$b_{\ell,k} \stackrel{\text{def}}{=} \sqrt{\frac{\pi_k}{\pi_\ell}}. \quad (6.14)$$

Lemma 6.7.1 *The map given by Definition. (6.7.1) satisfies the constraints (6.1,6.2), and thus, is a representation of $GTL(d)$.*

Proof: Equation (6.1) holds trivially. For Eq. (6.2),

$$\forall k : \quad \sum_{\ell:\ell \sim k} a_{\ell,k} b_{k,\ell} = \sum_{\ell:\ell \sim k} \frac{\pi_\ell}{\pi_k} = \frac{1}{\pi_k} \sum_{\ell} [M_\infty]_{k,\ell} \pi_\ell = \frac{1}{\pi_k} [M_\infty \bar{\pi}]_k = d. \quad (6.15)$$

In the first equality we used the definition of $a_{\ell,k}, b_{k,\ell}$. In the second equality we replaced the summation over the ℓ 's that are adjacent to k by a summation over all ℓ , using the fact that the adjacency matrix $[M_\infty]_{k,\ell}$ is equal to one when $k \sim \ell$ and vanishes otherwise. Finally, in the last equality we used the fact that $\bar{\pi}$ is an eigenvector of M_∞ with an eigenvalue d . \square

We now need to deal with cases in which the vector $\bar{\pi}$ vanishes at some points. To do this, we simply cut the graph F_∞ before the first location where the vector vanishes. Say that m sites remain on the graph; call this graph F_m . It is easy to see that the vector given by $\bar{\pi}$ cut at that point, namely the first m coordinates of $\bar{\pi}$, is an eigenvector of eigenvalue d of the adjacency matrix of F_m , which is simply the left uppermost $m \times m$ block of the matrix M_∞ .

The same construction of Definition. (6.7.1) will now work for these values of d , except that now the graph being used is the finite graph F_m .

6.8 Hermitian Representations

We conclude this section with a definition of an Hermitian representation of the $GTL(d)$ algebra. These types of representations have a few nice properties, which are particularly important for the BQP-hardness result (see [2]).

Definition 6.8.1 (An Hermitian representation of the $GTL(d)$ algebra) *A representation ρ of the $GTL(d)$ algebra is called Hermitian if $\rho(\mathfrak{A}_i) = \rho(\mathfrak{B}_i)^\dagger$ for every i . Here \mathfrak{A}_i and \mathfrak{B}_i are diagrams that represent the elementary cup and cap tangles $\mathcal{A}_i, \mathcal{B}_i$, which were defined in Chapter 5. For the above condition to make sense, we require that the number of out-pegs of \mathfrak{A}_i be equal to the number of in-pegs of \mathfrak{B}_i .*

For the family of path representations we have the following corollary:

Corollary 6.8.1 *A necessary and sufficient condition for a path representation to be Hermitian is that the coordinates of the eigenvector $\bar{\pi}$ are all positive.*

Proof: It is a simple exercise to verify that a path representation is Hermitian if and only if $a_{\ell,k} = b_{k,\ell}^*$ for every $\ell \sim k$. Therefore by Eqs. (6.13) and (6.14), we get

$$\sqrt{\frac{\pi_\ell}{\pi_k}} = \left(\sqrt{\frac{\pi_\ell}{\pi_k}} \right)^* , \quad (6.16)$$

which is equivalent to the condition that π_ℓ/π_k is positive for every $\ell \sim k$. Assuming that the auxiliary graph is connected and that we have normalized $\bar{\pi}$ such that $\pi_1 = 1$, we conclude that the representation is Hermitian if and only if $\bar{\pi} > 0$. \square

We finally remark, without proof, that for the general path-representation, which uses F_∞ , a *sufficient* condition for hermiticity is that $d = 2\cos(\pi/k)$ for integer k , or that $d > 2$. These are the representations that are generally used in the literature. As we already noted, there is no need for the representation to be Hermitian for our quantum algorithm to work. However, when we are interested in proving that the problem at hand is BQP-hard, there are important implications to the question of whether a representation is Hermitian or not. This question is treated in details in In [2].

This concludes the section on the path-model representation. Throughout this and the previous section, we paid special attention to the distinction between $GTL(d)$ elements and the diagrams that represent them, since the path-model representation is only defined for the diagrams – not for the elements themselves. From here on, however, we do not make that distinction when it is clear from the context that we have already chosen a particular diagram to represent a $GTL(d)$ element. We therefore occasionally talk about $\rho(\mathcal{T})$ and about the in-pegs and out-pegs of \mathcal{T} , even though these concepts are not-well defined.

Chapter 7

The Quantum Algorithm

To state the result, we assume that the graph G is given to us not as an adjacency matrix, but rather is embedded in the plane in some generic way.

Definition 7.0.2 (Nicely Embedded) *A medial graph L_G of a planar graph G is said to be nicely embedded if it is given as a diagram in \mathbb{R}^2 in such a way that if we sweep a horizontal line from the bottom of the tangle to the top, the horizontal line meets only one elementary tangle at a time – the minimum of a cup, the maximum of a cap, or a crossing.*

Like all isotopic facts, it is easy to see but probably much harder to prove that any medial graph can be so embedded. Moreover, any graph can be given as a diagram in \mathbb{R}^2 such that the resulting medial graph is nicely embedded. Therefore we abuse language and say in this case that a graph G is nicely embedded when we implicitly refer to a particular nice embedding of L_G in \mathbb{R}^2 . From now on we assume that the graph G is given to us in such a nicely embedded way. Obviously, such a nice embedding induces an order on the elementary tangles of the medial graph. A simple example is given in Fig. 7.1.

The algorithm works by assigning a linear operator to each of the elementary tangles (a crossing, a cap or a cup), according to the representation ρ , and applying them in the above-mentioned order.

Definition 7.0.3 (From a medial graph to elementary tangles) *Given a nicely embedded medial graph L_G , we translate it to a product of elementary tangles by decomposing it into a product of cups, caps and crossings according to the order that is determined by its (nice) embedding in \mathbb{R}^2 .*

The scale in which the algorithm will work is defined by

Definition 7.0.4 (Algorithmic Scale) *Let $G = (V, E)$ be a nicely embedded planar graph with weights $\mathbf{v} = \{v_e\}$ and q a complex number. In addition, let $\rho(\cdot)$ be the path representation of $GTL(d)$*

Figure 7.1: A nicely embedded planar graph G is a which is attached with a particular nicely embedded medial graph.

with $d^2 = q$. Then its medial graph, L_G can be translated into a product of basic tangles $\mathcal{T}_1 \cdot \dots \cdot \mathcal{T}_N$. The scale of the algorithmic problem of approximating $Z_G(q, \mathbf{v})$ is then

$$\Delta_{alg} \stackrel{\text{def}}{=} q^{|V|/2} \|\rho(\mathcal{T}_1)\| \cdot \dots \cdot \|\rho(\mathcal{T}_N)\| . \quad (7.1)$$

In other words, Δ_{alg} is the product of the norms of the operators that correspond to the elementary tangles that make up L_G , multiplied by, an overall factor $q^{|V|/2}$, which connects $\langle L_G \rangle(d, \mathbf{u})$ to $Z_G(q, \mathbf{v})$ (see Claim 2.3.1).

In accordance with the last paragraph of Chapter 6, when we talk about $\rho(\mathcal{T}_i)$, we are actually talking about the path-model representation of a *diagram* that represents the tangle \mathcal{T}_i . That diagram, however, is implicitly defined by the medial graph L_G ; hence $\rho(\mathcal{T}_i)$ is well defined.

We note that Δ_{alg} , depends not only on the graph but also on its embedding. A different way to orient the graph, for example, might lead to significant changes in the scale. We do not know of any algorithm that optimizes the embedding of the graph to get the best (smallest) possible scale. We can now give the precise version of the principal theorem of this thesis.

Theorem 7.0.1 (Quantum Algorithm)

Given a finite graph $G = (V, E)$ with a nice embedding in \mathbb{R}^2 , with weights on the edges, \mathbf{v} , and a complex number q , there is an efficient quantum algorithm that approximates the Tutte polynomial of the graph at those weights and q to within an additive approximation $\Delta_{alg}/\text{poly}(|E|)$.

After we prove this theorem, it will become clear that we have been quite wasteful in our bounding the quality of approximation and that the size of the approximation scale can be made significantly smaller if one is dealing with non-unitary parameters. We discuss this matter briefly in Sec. 7.2; in [2] there is a more rigorously treatment of the matter.

7.1 Proof of the Algorithm

To prove Theorem 7.0.1, we essentially apply the sequence of linear operators defined earlier by a using a quantum computer. It is convenient to use the following term:

Definition 7.1.1 (Operator circuit: see Ref. [4]) *An operator circuit is defined just like a quantum circuit, except that the gates are only restricted to be linear operators from k to ℓ quantum registers, with no unitarity restriction.*

Consider a medial graph with a nice embedding. There is an operator circuit associated with L_G , which is simply the circuit one gets by replacing each of the elementary tangles (cap, cup or crossing) with the corresponding linear operator by Definition. (7.0.3). We denote the resulting operator circuit by Q . Note that this operator circuit acts on the Hilbert space H_0 and takes it to H_0 .

Claim 7.1.1 $\langle L_G \rangle(d, \mathbf{u}) = \langle 1|Q|1 \rangle$.

Proof: By definition, $Q = \rho(\Psi(L_G))$. By Proposition 5.2.1, $\Psi(L_G) = \langle L_G \rangle(d, \mathbf{u})\mathcal{I}$, and since ρ is a representation, $\rho(\mathcal{I}) = \mathbb{1}$. Therefore, $\langle 1|Q|1 \rangle = \langle L_G \rangle(d, \mathbf{u}) \langle 1| \mathbb{1} |1 \rangle = \langle L_G \rangle(d, \mathbf{u})$. \square

In order to prove Theorem 7.0.1, we have to show that we can approximate the value of $\langle 1|Q|1 \rangle$ to within an additive approximation of the scale Δ_{alg} . To do this, we simply create the state $|1\rangle \in H_0$ and apply the operators in Q on this state one by one. To approximate the inner product of the resulting vector with $|1\rangle$, we use the Hadamard test [21].

7.1.1 Moving to Qubits

We first note that we need not consider sites in F with indices that are bigger than the number of edges in $G = (V, E)$, since we start from 1 and each crossing in L_G can only increase the path by one more site. This means that we can encode the name of a vertex in F with logarithmically many qubits (in $|E|$). We view the Hilbert space now as composed of registers; each register contains logarithmically many qubits and can hold a label of one vertex in the range $1 \rightarrow |E|$.

Note that each elementary crossing acts locally only on three labels of a path. Hence its operator under the path representation is actually the tensor product of the identity with an operator that acts on three local registers. The other elementary tangles, namely the cup and the cap, also yield local three-register operators, as we will see at the end of the next section. Therefore all the operators that we apply act non-trivially only on a logarithmic number of local qubits.

7.1.2 Simulating a Linear Operator

Claim 7.1.2 *Given a linear operator $M : H_k \rightarrow H_k$ that acts on a constant number of local registers and a quantum computer, it is possible to efficiently transform any normalized vector $|\alpha\rangle \in H_k$ to a normalized vector $|\beta\rangle \in H_k \otimes \mathbb{B}$, with \mathbb{B} being the space of an auxiliary qubit, such that the following holds: when the auxiliary qubit is projected to $|0\rangle$, the resulting state on H_k becomes $\frac{1}{\|M\|} M |\alpha\rangle$, where $\|M\|$ is the operator norm of M . In other words, $|\beta\rangle = \frac{1}{\|M\|} M |\alpha\rangle \otimes |0\rangle + \mathbf{c} |\gamma\rangle \otimes |1\rangle$, with $|\gamma\rangle$ being some residual state and \mathbf{c} a constant such that the overall state has unit norm.*

From this claim it follows that we can apply Q efficiently up to some normalization factor. We first prove the claim, and then worry about the normalization factors.

Proof of Claim 7.1.2:

According to the discussion in the previous section, if M acts on a constant number of local registers, then it can be viewed as an operator that acts on $\mathcal{O}(\log(|E|))$ local qubits. By the polar decomposition lemma, we can write M as a product of two matrices, one unitary and the other positive definite, $M = PU$. Both matrices will still act non-trivially only on $\mathcal{O}(\log(|E|))$ local qubits.

The simulation of unitary operators that act on logarithmically many qubits is a standard procedure in quantum computation, and can be done in polynomial time. Hence, we can simulate U efficiently.

We turn to the simulation of the positive definite matrix P . As P can be diagonalized by an orthonormal basis, we assume without loss of generality that P is diagonal in the computational basis; otherwise we can always change the basis back and forth efficiently. Let $r_1 \geq r_2 \geq \dots \geq r_m \geq 0$ be the eigenvalues of P , with the corresponding eigenvectors $|1\rangle, |2\rangle, \dots, |m\rangle$. We wish to apply P/r_1 . To do this we set the auxiliary qubit to $|0\rangle$, and apply a *unitary* transformation P_u that satisfies

$$P_u(|i\rangle \otimes |0\rangle) = \frac{r_i}{r_1} |i\rangle \otimes |0\rangle + \sqrt{1 - (r_i/r_1)^2} |i\rangle \otimes |1\rangle . \quad (7.2)$$

We think of the auxiliary qubit as an indicator of the validity of the original transformation; up to the overall factor r_1 , we get the original transformation only if the auxiliary qubit is $|0\rangle$. This is achieved exactly by projecting the auxiliary qubit onto $|0\rangle$. Therefore, the application of P is recovered by applying P_u , projecting the auxiliary qubit onto $|0\rangle$, and multiplying the final result by r_1 . Finally, note that as r_1 is the maximal eigenvalue of P , it follows that $\|M\| = r_1$. \square

We conclude that we can apply any desired operator, provided we *book-keep* its norm: we actually apply a matrix with a unit norm. Moreover, for each operator we get an extra qubit which must be projected onto $|0\rangle$ for the rest of the state to hold the desired result.

To complete the algorithm we have to deal with the operators that correspond to the cup and cap tangles, which are not square operators. Their action, however, is local. They can be viewed as operators that transform one register to three registers (the cup operator), or vice versa (the cap operator), while leaving the rest of the registers intact.

The cup operator is simply the concatenation of the operator $|k\rangle \mapsto |k, 1, 1\rangle$, with another 3×3 linear operator: $|k, 1, 1\rangle \mapsto \sum_{\ell: \ell \sim k} a_{k,\ell} |k, \ell, k\rangle$, and the rest of the vectors go to (the scalar) 0. We can apply the first operator easily by allocating two more registers. The second operator can be applied by the previous lemma. Likewise, the cap operator is defined by $|k, \ell, m\rangle \mapsto \delta_{k,m} b_{k,\ell} |k, 1, 1\rangle$, which can be applied by the lemma, after which we simply discard the last two registers.

7.1.3 Completing the Proof of the Algorithm with the Correct Approximation Window Size

Once we have applied all the operators, we have to compute the inner product of the final state, with the state $|1\rangle$ tensor with $|0\rangle$ on all the ancillary qubits. The final state is the following normalized state:

$$\frac{1}{\Delta'_{alg}} \left(Q |1\rangle \right) \otimes |0\rangle \otimes |0\rangle \dots |0\rangle + c |\gamma\rangle, \quad (7.3)$$

where $|\gamma\rangle$ is some residual state in which one or more of the ancillary qubits are different from $|0\rangle$, and $\Delta'_{alg} = q^{|V|/2} \Delta_{alg}$, i.e., it is simply the product of all the norms of the elementary tangles that make up Q .

It is a well-known fact that a quantum computer can efficiently estimate the inner product of such a state with $\langle 1| \otimes \langle 0| \dots \otimes \langle 0|$ to get an approximation of $\frac{1}{\Delta'_{alg}} \langle 1| Q |1\rangle$, using the Hadamard test. Repeating this test $\text{poly}(|E|)$ times would result in an approximation of $\frac{1}{\Delta'_{alg}} \langle 1| Q |1\rangle$ to within an additive window of size $1/\text{poly}(|E|)$ with exponentially good confidence. Finally, we use the Eq. (2.17) and Claim 7.1.1 to see that

$$\frac{1}{\Delta'_{alg}} \langle 1| Q |1\rangle = \frac{1}{\Delta'_{alg}} \langle L_G \rangle(d, \mathbf{u}) = \frac{1}{\Delta_{alg}} Z_G(q, \mathbf{v}). \quad (7.4)$$

This completes the proof of Theorem 7.0.1.

7.2 Improving the Approximation Window

When the crossings operators are all unitary, the scale Δ_{alg} does not increase by applying them; the norm of a product of unitary operators is equal to the product of their norms – which is equal to 1. However, the quality of the approximation might worsen severely when we apply non-unitary operations. It is easy to be convinced that the approximation scale we define and, in fact, our algorithm itself, are quite wasteful. For example, if one *groups* several operators together, the product of their norms is very likely to be larger than the norm of their products, and we will lose a lot in our approximation scale. As long as the product of the operators in the group is still an operator on poly-logarithmically many qubits, there is no problem in calculating the product of them all as one matrix (this calculation is done classically on the side) and then performing it as a whole using the quantum computer in the same way as in the previous section. This changes the contribution in the approximation window from the product of the norms to the norm of the product.

Although the problem of finding an optimal grouping is unresolved some progress was made in [2] as part of proving the BQP hardness of the approximation achieved in this thesis. A simple greedy algorithm was introduced that performs such grouping. This leads to an improved approximation scale that is necessary for the completeness result.

Bibliography

- [1] D. Aharonov and I. Arad. The BQP-hardness of approximating the Jones Polynomial. *Arxiv preprint quant-ph/0605181*, 2006.
- [2] D. Aharonov, I. Arad, E. Eban, and Z. Landau. Polynomial Quantum Algorithms for Additive approximations of the Potts model and other Points of the Tutte Plane. *Arxiv preprint quant-ph/0702008*, 2007.
- [3] D. Aharonov, V. Jones, and Z. Landau. A polynomial quantum algorithm for approximating the Jones polynomial. *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 427–436, 2006.
- [4] D. Aharonov, Z. Landau, and J. Makowsky. The quantum FFT can be classically simulated. *Arxiv preprint quant-ph/0611156*, 2006.
- [5] S. Arora and S. Khot. Fitting algebraic curves to noisy data. *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 162–169, 2002.
- [6] E. Artin. Theory of Braids. *The Annals of Mathematics*, 48(1):101–126, 1947.
- [7] S.J. Bigelow. Braid Groups are Linear. *Journal of the American Mathematical Society*, 14(2):471–486, 2001.
- [8] J.H. Cheon and B. Jun. A Polynomial Time Algorithm for the Braid Diffie-Hellman Conjugacy Problem. *CRYPTO*, 2003.
- [9] CM Fortuin and PW Kasteleyn. On the random-cluster model: I. Introduction and relation to other models. *Physica*, 57(4):536–564, 1972.
- [10] M.H. Freedman, A. Kitaev, and Z. Wang. Simulation of topological field theories by quantum computers. *Communications in Mathematical Physics*, 227(3):587–603, 2002.
- [11] M.H. Freedman, M. Larsen, and Z. Wang. A Modular Functor Which is Universal for Quantum Computation. *Communications in Mathematical Physics*, 227(3):605–622, 2002.
- [12] J. Geraci and D.A. Lidar. On the Exact Evaluation of Certain Instances of the Potts Partition Function by Quantum Computers. *Arxiv preprint quant-ph/0703023*, 2007.
- [13] AK Hartmann. Calculation of Partition Functions by Measuring Component Distributions. *Physical Review Letters*, 94(5), 2005.
- [14] F. Jaeger, DL Vertigan, and DJA Welsh. On the computational complexity of the Jones and Tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society*, 108:35–53, 1990.
- [15] PW Kasteleyn and CM Fortuin. Phase Transitions in Lattice Systems with Random Local Properties. *Physical Society of Japan Journal*, 1969.

- [16] L.H. Kauffman. State models and the Jones polynomial. *Topology*, 26(3):395–407, 1987.
- [17] L.H. Kauffman et al. *Knots and Physics*. World Scientific, 1991.
- [18] K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J.S. Kang, and C. Park. New Public-Key Cryptosystem Using Braid Groups. *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 166–183, 2000.
- [19] D.A. Lidar. On the quantum computational complexity of the Ising spin glass partition function and of knot invariants. *New Journal of Physics*, 6(1):167, 2004.
- [20] R.J. Lipton and Y. Zalcstein. Word Problems Solvable in Logspace. *Journal of the ACM (JACM)*, 24(3):522–526, 1977.
- [21] MA Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [22] A.D. Sokal. The multivariate Tutte polynomial (alias Potts model) for graphs and matroids. *Arxiv preprint math.CO/0503607*, 2005.
- [23] M. Van den Nest, W. Dür, and HJ Briegel. Classical spin models and the quantum stabilizer formalism. *Arxiv preprint quant-ph/0610157*, 2006.
- [24] F. Wang and DP Landau. Efficient, Multiple-Range Random Walk Algorithm to Calculate the Density of States. *Physical Review Letters*, 86(10):2050–2053, 2001.
- [25] DJA Welsh. *Complexity: knots, colourings and counting*. Cambridge University Press New York, NY, USA, 1993.
- [26] DJA Welsh and C. Merino. The potts model and the tutte polynomial. *Journal of Mathematical Physics*, 41:1127, 2000.
- [27] FY Wu. The potts model. *Reviews of Modern Physics*, 54(1):235–268, 1982.
- [28] J. Yard and P. Wocjan. The Jones polynomial: quantum algorithms and applications in quantum complexity theory. *Arxiv preprint quant-ph/0603069*, 2006.