Commun. Math. Phys. 279, 735–768 (2008)
Digital Object Identifier (DOI) 10.1007/s00220-008-0438-0

Communications in
Mathematical
Physics

On the Exact Evaluation of Certain Instances of the Potts Partition Function by Quantum Computers

Joseph Geraci[1,2], **Daniel A. Lidar**[2]

[1] Department of Mathematics, University of Toronto, Toronto, ON M5S 2E4, Canada.
E-mail: geraci.joseph@gmail.com
[2] Departments of Chemistry, Electrical Engineering, and Physics, Center for Quantum Information Science & Technology, University of Southern California, Los Angeles, CA 90089, USA

Received: 16 March 2007 / Accepted: 5 September 2007
Published online: 1 March 2008 –

Abstract: We present an efficient quantum algorithm for the exact evaluation of either the fully ferromagnetic or anti-ferromagnetic q-state Potts partition function Z for a family of graphs related to irreducible cyclic codes. This problem is related to the evaluation of the Jones and Tutte polynomials. We consider the connection between the weight enumerator polynomial from coding theory and Z and exploit the fact that there exists a quantum algorithm for efficiently estimating Gauss sums in order to obtain the weight enumerator for a certain class of linear codes. In this way we demonstrate that for a certain class of sparse graphs, which we call Irreducible Cyclic Cocycle Code (ICCC$_\epsilon$) graphs, quantum computers provide a polynomial speed up in the difference between the number of edges and vertices of the graph, and an exponential speed up in q, over the best classical algorithms known to date.

1. Introduction

A wealth of results has been obtained since the dramatic early results [1,2] on quantum speedups relative to classical algorithms. A relatively unexplored field is quantum algorithms for problems in classical statistical mechanics. The earliest contribution to this subject [3] obtained a modest speedup in that it avoided critical slowing down [4] in the problem of sampling from the Gibbs distribution for Ising spin glass models. Subsequently Ref. [5] raised the question of providing a classification of classical statistical physics problems in terms of their quantum computational complexity. In this work we shed light on this classification by considering the problem of evaluating the Potts model partition function Z for classical spin systems on graphs. It is known that under particular conditions even certain approximations for Z are unlikely to be efficient, barring an $NP = RP$ surprise [6]. Here we present a class of sparse graphs (which we call ICCC$_\epsilon$) for which *exact* quantum evaluation of Z is possible with a polynomial speedup in the size of the graph and an exponential speedup in the number of per-spin states, over the best classical algorithms available to date.

The Potts partition function of graphs in ICCC_ϵ is equivalent to the weight enumerators of certain linear codes. The evaluation of weight enumerators (in this case) involves the evaluation of Gauss sums or Zeta functions. The evaluation of Gauss sums is in general hard and equivalent to the calculation of discrete log [7]. This suggests that ICCC_ϵ includes cases that are unlikely to be solved as efficiently on classical computers.

1.1. The Potts model. Let $\Gamma = (E, V)$ be a weighted graph with edge set E and vertex set V. The q-state Potts model is a generalization of the Ising model where a q-state spin resides on each vertex. In the Ising model $q = 2$, whereas in the Potts model $q \geq 2$. The edge connecting vertices i and j has weight J_{ij}, which is also the interaction strength between the corresponding spins. The Potts model Hamiltonian for a particular spin configuration $\sigma = (\sigma_1, \ldots, \sigma_{|V|})$ is

$$H(\sigma) = -\sum_{<ij>} J_{ij}\delta_{\sigma_i\sigma_j}, \tag{1}$$

where summation is over nearest neighbors, and where $\delta_{\sigma_i\sigma_j} = 1$ (0) if $\sigma_i = \sigma_j$ ($\sigma_i \neq \sigma_j$). Thus only nearest neighbor parallel spins contribute to the energy. The probability $P(\sigma)$, of finding the spin in the Potts model in some configuration σ at a given temperature T, is given by the Gibbs distribution

$$P(\sigma) = \frac{e^{-\beta H(\sigma)}}{Z(\beta)}, \tag{2}$$

where $\beta = 1/(k_\mathrm{B}T)$ is the inverse temperature in energy units, and k_B is the Boltzmann constant. The normalization factor is the *partition function*

$$Z(\beta) = \sum_{\{\sigma\}} e^{-\beta H(\sigma)}, \tag{3}$$

which plays a central role in statistical physics, since many thermodynamic quantities can be derived from it [7]. When for all configurations $\beta \ll |H(\sigma)|$, the probability distribution becomes flat: $P(\sigma) \approx 1/Z(\beta)$, so that at high temperatures randomness dominates.

The partition function can be rewritten as a polynomial:

$$\begin{aligned} Z(\beta) &= \sum_{\{\sigma\}} e^{\beta \sum_{<ij>} J_{ij}\delta_{\sigma_i\sigma_j}} = \sum_{\{\sigma\}} \prod_{<ij>} e^{\beta J_{ij}\delta_{\sigma_i\sigma_j}} \\ &= \sum_{\{\sigma\}} \prod_{<ij>} (1 + v_{ij}(\beta)\delta_{\sigma_i\sigma_j}), \end{aligned} \tag{4}$$

where

$$v_{ij}(\beta) = e^{\beta J_{ij}} - 1. \tag{5}$$

Now let us consider the case when the interactions J_{ij} are a constant J. Then the Hamiltonian (1) of this system can be written as

$$H(\sigma) = -J|U(\sigma)|, \tag{6}$$

where $U(\sigma)$ is the subset of edges whose vertices have the same spin for a particular spin configuration σ, and $|U(\sigma)|$ is the number of such subsets. If we let

$$y = e^{-\frac{J}{kT}} \tag{7}$$

we can write the Potts partition function as

$$Z(y) = \sum_{\sigma} y^{-|U(\sigma)|}. \tag{8}$$

1.2. Relation between the Potts partition function, knot invariants, and graph theory. There is a rich inter-relation between classical statistical mechanics and topology, in particular, the theory of the classification of knots. The first such connection was established by Jones [8], who discovered the second knot invariant (the Jones polynomial (a Laurent polynomial), after the Alexander polynomial) during his investigation of the topological properties of braids [9]. It is known that the classical evaluation of the Jones polynomial is ♯P-hard [10].

A direct connection between knots and models of classical statistical mechanics was established by Kauffman [11]. Knot invariants are, in turn, also tightly related to graph theory; e.g., the graph coloring problem can be considered an instance of evaluation of the Kauffman bracket polynomial, via the Tutte polynomial [11,12]. The q-state Potts partition function on a graph Γ is connected to the Tutte polynomial $\mathcal{T}_\Gamma$ for the same graph via

$$Z_\Gamma(v) = q^n \mathcal{T}_\Gamma \left(\frac{q+v}{v}, v+1 \right), \tag{9}$$

where as in Eq. (5), $v+1 = e^{-\beta}$. This means that the Potts partition function is equivalent to some, easily computed function times the Tutte polynomial along the hyperbola $\mathcal{H}_q = (x-1)(y-1) = q$. But for planar graphs, when $q > 2$ the Tutte polynomial is ♯P-hard to evaluate at points along $\mathcal{H}_q$ [6]. For a review of the connection between the Potts partition function and the various polynomials mentioned above, see [11] and also [5,13]. It immediately follows from Eq. (9) and complexity results concerning the Tutte polynomial, that the evaluation of the Potts partition function is also ♯P-hard. It is not known whether there is an fpras (fully polynomial randomized approximation scheme) [6] for the q-state fully ferromagnetic Potts partition function, but it is known that if there is an fpras for the fully anti-ferromagnetic Potts partition function then $NP = RP$ [6] and therefore it seems unlikely that an fpras will be found for this case.

1.3. Previous complexity results. The first connection between knots and quantum field theory was established by Witten, who showed that the Jones polynomial can be expressed in terms of a topological quantum field theory [14]. Recently this connection was extended to the possibility of efficient evaluation of the Jones polynomial by Freedman and co-workers, after showing that quantum computers can efficiently simulate topological quantum field theory [15]. More specifically, there are recent results demonstrating the efficacy of quantum computers in approximating the Jones polynomial at primitive roots of unity [16–18]. In Ref. [16] tools from topological quantum field theory [14] were utilized and it was shown that approximating the Jones polynomial at primitive roots of unity is BQP-complete, but no explicit algorithm was provided. More recently in

[17], a combinatorial approach was taken which yielded an explicit quantum algorithm and which extended the results in [16] to all primitive roots of unity. This leads one to hypothesize that quantum computers will also be efficient at estimating partition functions. Indeed, an immediate corollary of the results in [16–19], is that the Potts partition function over any planar graph can be approximated efficiently on a quantum computer at certain *imaginary* temperatures (see also [5]). This follows by noting that in order to obtain an equality between the Potts partition function and the Jones polynomial (up to multiplication by an easily computed function), the Jones variable t and the temperature T must be related by $t = -e^{\pm J_\pm / k_B T}$ [11]. With t a root of unity ($t = e^{\frac{2\pi i}{r}}$) we then find:

$$T = i \frac{J_\pm r}{k_B \pi (2 + r)}, \quad r \in \mathbb{N}.$$

This result is of interest mainly in light of quantum Monte Carlo simulations [20], where one retrieves real time *dynamics* from a simulation in terms of imaginary time, via analytic continuation. Perhaps a similar extrapolation can be achieved here between imaginary and real temperature dynamics. While this is interesting, here we are concerned with *thermodynamics*, and hence evaluations of the Potts partition function at physically relevant, real temperatures.

Most closely related to our work is the very recent result due to Aharonov *et al*. [21] who – generalizing Temperley-Lieb algebra representations used in [17] – provided a quantum algorithm for the additive approximation of the Potts partition function (and other points of the Tutte plane) for any planar graph with any set of weights on the edges. These results are the most impressive to date in the context of approximate evaluations of the Potts partition function, but are also subject to certain caveats.[1] In particular, Ref. [21] leaves as an open problem the complexity of physical instances (real temperature, positive partition function) under the restriction of an additive approximation. Nor is it clear whether the algorithm found in Ref. [21] provides a quantum speedup. The authors state: "We believe that the main achievement here is that we demonstrate how to handle non-unitary representations, and in particular, we are able to prove universality using non-unitary matrices."

Recently Ref. [22] gave a scheme for studying the partition function of classical spin systems including the Potts and Ising model. Their approach involves transforming the problem of evaluating the partition function into the evaluation of a probability amplitude of a quantum mechanical system and then using classical techniques to extract the pertinent information. In essence their method involves moving into a quantum mechanical formalism to obtain a classical result. The scheme is therefore classical and not a quantum algorithm.

[1] To quote from the abstract of Ref. [21]: "Additive approximations are tricky; the range of the possible outcomes might be smaller than the size of the approximation window, in which case the outcome is meaningless. Unfortunately, ruling out this possibility is difficult: If we want to argue that our algorithms are meaningful, we have to provide an estimate of the scale of the problem, which is difficult here exactly because no efficient algorithm for the problem exists!". And: "The case of the Potts model parameters deserves special attention. Unfortunately, despite being able to handle non-unitary representations, our methods of proving universality seem to be non-applicable for the physical Potts model parameters. We can provide only weak evidence that our algorithms are non-trivial in this case, by analyzing their performance for instances for which classical efficient algorithms exist. The characterization of the quality of the algorithm for the Potts parameters is thus left as an important open problem." Finally, quoting from Sect. 1.5 of Ref. [21]: "Proving anything about the complexity of our algorithm for the Potts model remains a very important open problem. It is still possible that this case of the Tutte polynomial, with our additive approximation window, can be solved by an efficient classical algorithm."

In addition, two purely classical results should be mentioned here. One is a state of the art result by Hartmann [23], who provides an algorithm which is well suited to large ferromagnetic systems for either the Potts or Ising model. We do not know the exact complexity of this algorithm, however. The approach taken in our work is to utilize the connection between classical coding theory and the partition function. For this reason we mention the classical algorithm given in [24] for calculating the Zeta function of certain curves. This is also a state of the art algorithm and it can be used to find the Potts partition function via the scheme we present in this paper, though it is slower than using quantum resources.

A quantum algorithm for finding the Zeta function of a curve is given in [25]. One could replace the role that the Gauss sum estimation [26] plays in our scheme with this quantum algorithm for the Zeta function. It seems that using Gauss sums is more efficient but further work is required to make this conclusive.

Finally, we mention that it was recently shown that one can construct interesting classes of graphs for which the Potts model can be computed analytically [46]. These so called n-ladder graphs are recursively defined.

1.4. Structure of the paper. The structure of this paper is as follows. In Sect. 2 we define the class of graphs our quantum algorithm applies to, and present our main theorem. In Sect. 3 we compare the computational complexity of our algorithm with the state of the art in classical algorithms. In Sect. 4 we give a brief summary of the entire algorithm. In Sect. 5 we provide several illustrative examples of graphs and codes to which our algorithm applies. Finally, in Sect. 6 we conclude and discuss future directions. The Appendix provides pertinent background on matroids, irreducible cyclic codes, and Gauss sums.

2. A Theorem about Quantum Computation and Certain Instances of the Potts Model

2.1. Main Theorem. We present here a polynomial time quantum algorithm for the exact evaluation of the q-state (fully ferromagnetic or anti-ferromagnetic) Potts partition function Z for a certain class of graphs. This class of graphs, which we call "Irreducible Cyclic Cocycle Code" ICCC_ϵ graphs, comprises graphs whose incidence matrices generate certain cyclic codes. This and other concepts used below are given precise definitions in Sect. 2.2. The key ingredients used are the connection of Z to the weight enumerators of codes [27] and a quantum algorithm for the approximation of Gauss sums [26].

The overall structure of the algorithm is the following:

1. Given a graph $\Gamma = (E, V)$, first determine if Γ belongs to the ICCC_ϵ class. This decision problem can be solved efficiently using the quantum discrete log algorithm [1]. If $\Gamma \in \mathrm{ICCC}_\epsilon$ proceed to Step 2, otherwise the algorithm may not evaluate Z_Γ efficiently.
2. Identify the linear code $C(\Gamma)$ for which we shall determine the weight spectrum.
3. Using the quantum Gauss sum estimation algorithm find the weight spectrum of the words in C. This step is believed to be classically hard but the exact complexity is unknown. It is known, however, that this step is at least as hard as determining discrete log [26]. This is the most expensive step of the algorithm due to the large number of words one has to deal with. This is because the number of possible spin configurations grows exponentially in the number of vertices.

4. Take a tally of the weight spectrum obtained in the previous step. Grover's search algorithm can be used to give an additional quadratic speed up but this does not help in reducing the overall complexity since the computational cost of Step 3 is greater than that of the current step.
5. Using the relation given by Eq. (15) between the weight spectrum of a code and Z, use the tally from the previous step to obtain Z (for graphs in ICCC_ϵ).

We now give the main theorem, after the definition of the family of graphs for which the scheme applies.

Definition 1. (ICCC_ϵ) *Given a constant $\epsilon < 1$, ICCC_ϵ is the family of graphs whose cycle matroid matrix (CMM) representation generates a cyclic code whose dual is irreducible cyclic of dimension k and length n, such that*

$$n = \frac{q^k - 1}{\alpha k^{s(k)}} \tag{10}$$

(where $\alpha \in \mathbf{R}$ is chosen so that $n \in \mathbf{N}$ and where $s(k)$ is an arbitrary function whose role will be clarified below) and

$$\theta_{n,k} = \frac{1}{q-1} \min_{0<j\leq \alpha k^{s(k)}} S'(jn) \tag{11}$$

(where $S'(x)$ is the sum of the digits of x in base q) so that

$$\epsilon \leq \frac{q^{\theta_{n,k}-1}}{4\sqrt{q^k}}. \tag{12}$$

Below we define the concepts entering this definition and clarify the role of $\theta_{n,k}$ and of the bound on ϵ.

We work in units such that the Boltzmann constant $k_\mathrm{B} = 1$.

Theorem 1 (***Main Theorem***). *Let $\Gamma = (E, V)$ be a graph, $n = |E|$ and $k = |E| - |V| + c(\Gamma)$, where $c(\Gamma)$ is the number of connected components of Γ. A quantum computer can return the exact q-state fully anti-ferromagnetic or ferromagnetic Potts partition function Z_Γ for graphs in ICCC_ϵ. For each family (ϵ fixed), the overall running time is $O(\frac{1}{\epsilon} k^{2\max[1,s(k)]} (\log q)^2)$ and the success probability is at least $1 - \delta$, where $\delta = [2((q^k - 1)^2\epsilon - 2)]^{-1}$.*

Some remarks:

1. The function $s(k)$ determines the complexity of the schemes. If $s(k) = c \in \mathbf{R}$ (constant) then we have a polynomial time algorithm for the exact evaluation of Z for each family ICCC_ϵ. This restriction is reflected in the graphs by enforcing that $n = O(q^k/k^s)$, i.e., that the number of edges (n) and vertices ($n - k$) is close. We have numerically solved for the number of edges $|E|$ as a function of the number of vertices $|V|$, given by the corresponding transcendental equation $|E| = |V| - c(\Gamma) + \log_q(|E|(|E| - |V| + c(\Gamma))^s + 1)$ [Eq. (10)]. A numerical fit reveals that to an excellent approximation

$$|E| = |V| + a + b \log |V|, \tag{13}$$

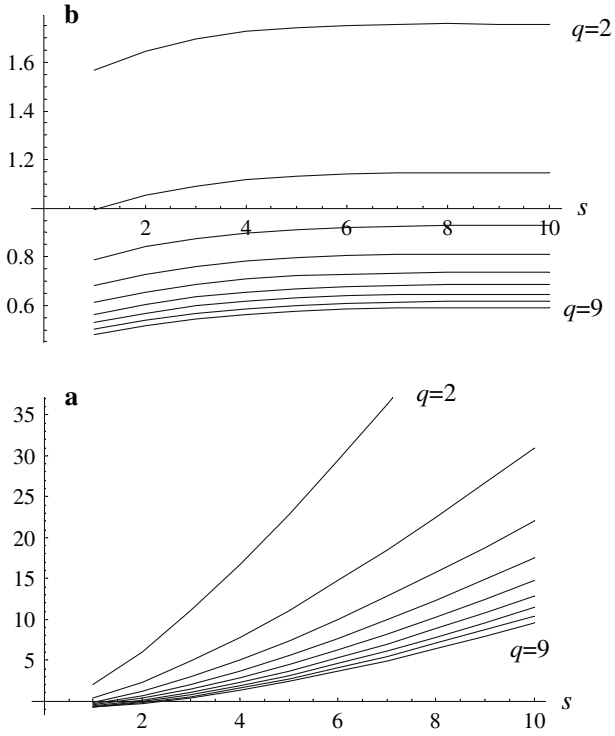

Fig. 1. Coefficients a and b as a function of s, for different values of q. Here $c(\Gamma) = 1$. See text for details

where the constants a and b depend on q and s, and both increase slowly with s, and decrease with q, as shown in Fig. 2.1. By direct substitution of Eq. (13) into the above transcendental equation it can be seen that the analytical solution will have a correction of order $\log\log(|V|)$ to the right-hand side of Eq. (13). The fact that there are logarithmically more edges than vertices in the graphs that are members of ICCC_ϵ is the reason we call these graphs sparse. The important point is that there are families of graphs for which there exist exact polynomial-time evaluation schemes via the methods presented in this paper. As we show below, in these cases we also obtain polynomial speed ups over the best classical algorithms available.

2. Note that if we have an efficient evaluation for $\mathrm{ICCC}_{\epsilon'}$ then we also have an efficient evaluation for ICCC_ϵ, provided $\epsilon > \epsilon'$.
3. We provide a discussion of the computational complexity, both classical and quantum, in Subsect. 3. As argued there, we obtain a polynomial speed up in the difference between the number of edges and vertices and an exponential speed up in q over the best current classical algorithm for the ICCC_ϵ class of graphs.

Corollary 1. *For a given graph Γ, whose CMM is the direct sum of the CMMs of two graphs Γ_1 and Γ_2 in* ICCC_ϵ*, a quantum computer will be able to return Z_Γ with a running time equal to the sum of the running times required to obtain Z_{Γ_1} and Z_{Γ_2}.*

Proofs of the main theorem and the corollary are provided in Sects. 2.4 and 2.5.

2.2. Background. Theorem 1 connects the problem of estimating the Potts partition function to a quantum algorithm for Gauss sums, via weight enumerators for irreducible cyclic codes. In somewhat more detail, the connections we need are as follows. In [27], it was shown that the Potts partition function can be written as the weight enumerator of the cocycle code of the graph Γ, over which the Potts model is defined. Weight enumerators of irreducible cyclic codes are related to Gauss sums via the McEliece Theorem [28].

2.2.1. Cycle matroid matrix representation of a graph A connected component of a graph is any subset of vertices which are all connected to each other via a path along the graph's edges. We denote the number of connected components by $c(\Gamma)$. The incidence matrix of a finite graph $\Gamma(E, V)$ is a $|V| \times |E|$ binary matrix where column c represents edge c with non-zero entries in row i and j if and only if vertices i and j are the boundaries of edge c. Every finite graph Γ also gives rise to a cycle matroid matrix (CMM) [29], which essentially captures the presence and locations of cycles in the graph.

Definition 2. *The cycle matroid matrix of a graph $\Gamma = (E, V)$, CMM(Γ), is formed as follows: write down the incidence matrix of Γ using* 1 *for the i^{th} and* -1 *for the j^{th} rows, where $i < j$. Then apply elementary row operations and Gaussian reduction to obtain a $(|V| - c(\Gamma)) \times |E|$ matrix of the form $[I_{|V|-c(\Gamma)}|X]$, where I_a is the $a \times a$ identity matrix and X is a $(|V| - c(\Gamma)) \times (|E| - |V| + c(\Gamma))$ matrix. This is CMM(Γ) (see Prop. 4.7.14 of* [30]).

We give more details on cycle matroids in Appendix 7. As an example consider the square $[|V| = |E| = 4, c(\Gamma) = 1]$ and its incidence matrix

$$\begin{pmatrix} 1 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}.$$

Applying elementary row operations and Gaussian reduction one obtains the CMM

$$\begin{pmatrix} 1\,0\,0\,-1 \\ 0\,1\,0\,-1 \\ 0\,0\,1\,-1 \end{pmatrix},$$

which is indeed of the form $[I_{|V|-c(\Gamma)}|X]$ with dimensions as in the definition, i.e., X is $(4-1) \times (4-4+1)$. Over $\mathbf{Z}_2$ one would replace all the -1's with $+1$'s. The column space of this matrix represents the cycle structure of the graph where a cycle (or circuit) is a path in the graph for which the first vertex of the path is the same as the last. Any set of columns that are linearly dependent indicate a cycle. The first three columns in the CMM of the square are linearly independent, but together with the fourth column they become linearly dependent, since there is a cycle in the graph involving the corresponding four edges.

What is the equivalence class of graphs with the same CMM? This is answered by the following:

Definition 3. *Two graphs G and G' are called* 2*-isomorphic if there exists a* $1-1$ *correspondence between the edges of G and G' such that the cycle* (*or circuit*) *relationships are preserved.*

Thus all 2-isomorphic graphs have the same CMM (up to elementary row and column operations).

2.2.2. Irreducible cyclic codes We provide some background material in coding theory which allows us to exhibit the connection between Eq. (8) and the so-called cocycle code of the graph Γ. (A good reference is [31].)

Definition 4. *Let* $\mathbf{F}_q$ *be a finite field with* q *prime. A linear code* C *is a* k *dimensional subspace of the vector space* $\mathbf{F}_q^n$ *and is referred to as an* $[n, k]$ *code. The code is said to be of length* n *and of dimension* k.

In our case q is the number of possible states per spin.

Definition 5. *A* $k \times n$ *matrix whose rows are a basis for* C *is called a generator matrix for* C.

Recall from Definition 2 that CMM(Γ) is a $(|V| - c(\Gamma)) \times |E|$ matrix. The $|E|$ columns of CMM(Γ) reflect the cycle structure of the given graph via linear independence in the vector space $\mathbf{F}_q^n$ (see Appendix 7). We now view the $|V| - c(\Gamma)$ rows of the CMM as generating an $[n = |V|, k = |E| - c(\Gamma)]$ "cocycle code" C:

Definition 6. *The cocycle code* $C(\Gamma)$ *of a graph* Γ *is the row space of CMM(*Γ*)* [27].

We focus our attention on a subclass known as *cyclic codes* and a smaller subclass known as *irreducible cyclic codes*.

Definition 7. *A linear code* C *is a cyclic code if for any word* $(c_0, c_1, \ldots, c_{n-1}) \in C$, *also* $(c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \in C$. *If* C *contains no subspace* (*other than* 0) *which is closed under cyclic shifts then it is irreducible cyclic.*

Cyclic codes have an interesting underlying algebraic structure which we review in Appendix 8. In general the generator matrix of an $[n, k]$ cyclic code can be written as

$$\begin{pmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & & & & \cdots & 0 \\ 0 & 0 & \cdots & & g_0 & g_1 & \cdots & g_{n-k} \end{pmatrix}. \tag{14}$$

The non-zero matrix elements can be used to construct the "generator polynomial" $g(x) = g_0 + g_1 x + \cdots g_{n-k} x^{n-k}$. Both can be used to generate a cyclic code; the manner in which this is done via the generator polynomial is reviewed in Appendix 8. For an $[n, k]$ non-degenerate irreducible cyclic code (no words are repeated) the relation between n and k is $k = \mathrm{ord}_q n$, i.e., k is the smallest integer such that $q^k = 1 \bmod n$. Equivalently, $q^k - 1 = nN$, where N counts the number of equivalence classes under cyclic permutations of words, which is an upper bound on the number of different weights. Non-degenerate irreducible cyclic codes have generator polynomials of the form

$$g_i(x) = \frac{x^n - 1}{w_i(x)},$$

where $x^n - 1 = w_1(x) w_2(x) \cdots w_t(x)$ is the decomposition of $x^n - 1$ into irreducible factors. Here t is the number of q-cyclotomic cosets mod n (see Subsect. 2.6).

Next we explain the connection to weight enumerators.

Definition 8. *Let C be a linear code of length n and let A_i be the number of vectors in C having i non-zero entries (Hamming weight of i). Then the weight enumerator of C is the bi-variate polynomial*

$$A(x, y) = \sum_{i=0}^{n} A_i x^{n-i} y^i.$$

The set $\{A_i\}$ is called the weight spectrum of the code.

In this paper our only concern for the weight spectrum is its connection to the Potts partition function, but in coding theory it can be used to reveal information about the effiency of a code [31]. The connection between Eq. (8) and the cocycle code of the graph Γ for the Potts model is given in the following theorem proved in [27].

Theorem 2. *Let $A(x, y)$ be the weight enumerator of the $[n = |E|, k = |V| - c(\Gamma)]$ cocycle code $C(\Gamma)$ of the graph $\Gamma = (E, V)$, and let the number of states per spin (vertex) in the corresponding Potts model be a prime q. Then*

$$Z_\Gamma(y) = y^{-n} q^{c(\Gamma)} A(1, y). \tag{15}$$

We take q to be prime and not a power of a prime to simplify matters. In this manner the cocycle code has words whose entries are in $\mathbf{F}_q$, as will the corresponding irreducible cyclic code in the trace representation over $\mathbf{F}_{q^r}$ – see Appendix 8 for details.

The connection between the Potts partition function and weight enumerators can also be understood via a previous result which shows that Z is equivalent to the Tutte polynomial (under certain restrictions) and that the weight polynomial of a linear code is also equivalent to the Tutte polynomial [32]. We also note that a relation similar to Eq. (15) was established in [5] for the Ising spin glass partition function and so-called quadratically signed weight enumerators, along with a discussion of computational complexity.

2.3. Testing the graph for membership in the ICCC$_\epsilon$ *class.* We now have the tools to address the issue of whether a graph should be accepted as input into the main algorithm, i.e., whether a graph belongs to the ICCC$_\epsilon$ class. This is handled as follows:

- `Input:` A graph Γ with $|E|$ edges and $|V|$ vertices, the given Galois field of q^k elements and ϵ.
- `Output:` Accept or Reject. Let $n = |E|$ and $k = |E| - |V| + c(\Gamma)$ as in the main theorem.
- `Overall Complexity:` $O(|E| \cdot k^2 \log k \log\log k)$ due the ability to take the discrete log $|E|$ times efficiently with a quantum computer [1].

1. Compute $\theta_{n,k}$ as given in Definition 1.
2. Find CMM(Γ). It is a $(n-k) \times n$ matrix of the form $[I_{n-k}|X]$, where X is a $(n-k) \times k$ matrix. Form the $k \times n$ (transpose parity check) matrix $H^T = [-X^T|I_k]$. H^T generates an $[n, k]$ code $C^\perp(\Gamma)$ that is dual to the cocycle code $C(\Gamma)$.
3. Determine if $\epsilon \leq \frac{q^{\theta_{n,k}-1}}{4\sqrt{q^k}}$ and if k is the multiplicative order of $q \bmod n$ (i.e., k is the smallest integer such that $q^k = 1 \bmod n$). If both are true then go to the next step. Otherwise skip the next step and continue.

4. Main Loop:
   (a) Fix a basis of $GF(q^k)$ over $GF(q)$ and consider the columns of H^T as coordinate vectors of some elements g_i of $GF(q^k)$.
   (b) Calculate the discrete logarithms $\log(g_i)$ of each g_i with respect to a fixed primitive element g (every element in the field can be written as g^l for some l) of $GF(q^k)$ using Shor's algorithm [1] on a quantum computer.[2]
   (c) Accept or Reject Γ based on the fact that $C^{\perp}$ is (equivalent to) an irreducible cyclic code if and only if the numbers $\log(g_i)$ are some permuted list of consecutive integer multiples of $N := (q^k - 1)/n$ in some order. This is due to the fact that by definition the generator matrix of an irreducible cyclic code is equivalent to $(1\ g^{Nj}\ g^{2Nj}\ \ldots\ g^{(n-1)Nj})$ where $\gcd(n, j) = 1$ [31].
5. Step c failed. Using elementary row operations transform H^T to a block diagonal matrix if possible. If not possible then Reject. If possible then go to Step c and input each sub-matrix and continue.

2.4. Proof of the Main Theorem.

2.4.1. Preliminaries We introduce Gauss sums as this is the vital link between the Potts partition function and quantum computation. Appendix 9 contains a more detailed exposition as well as an outline of a proof that there exists a polylog quantum algorithm for estimating Gauss sums. Here however it is essential only to understand the following. Given a field $\mathbf{F}_{q^k}$, there is a multiplicative and additive group associated with it. Namely, the multiplicative group is $\mathbf{F}^*_{q^k} = \mathbf{F}_{q^k} \setminus 0$ and the additive group is $\mathbf{F}_{q^k}$ itself. Associated with each group are canonical homomorphisms from the group to the complex numbers, named the additive and multiplicative characters. The multiplicative character χ is a function of the elements of $\mathbf{F}^*_{q^k}$ and the additive character is a function of $\mathbf{F}_{q^k}$ and is parameterized by $\beta \in \mathbf{F}_{q^k}$. A Gauss sum is then a function of the field $\mathbf{F}_{q^k}$, the multiplicative character χ and the parameter β, and can always be written as

$$G_{\mathbf{F}_{q^k}}(\chi, \beta) = \sqrt{q^k} e^{i\gamma}, \tag{16}$$

where γ is a function of χ and β. It is in general quite difficult to find the angle γ. The complexity of estimating this quantity via classical computation is not known but there is evidence that it is hard [26].

We now introduce the trace function over finite fields.

Definition 9. *Let q be prime, k a positive integer, and let $\mathbf{F}_{q^k}$ be the finite field with $q^k - 1$ non-zero elements. The trace is a mapping* $\mathrm{Tr} : \mathbf{F}_{q^k} \mapsto \mathbf{F}_q$ *and is defined as follows. Let $\xi \in \mathbf{F}_{q^k}$. Then*

$$\mathrm{Tr}(\xi) = \sum_{j=0}^{k-1} \xi^{q^j}. \tag{17}$$

[2] For every non-zero $x \in \mathbf{F}_{q^r}/\{0\}$ the discrete logarithm with respect to a primitive element (i.e., generator) g of $\mathbf{F}_{q^r}$ is given by $\log_g(x) = \log_g(g^j) = j \bmod (q^r - 1)$.

Now let $q^k = 1 + nN$, where n and N are both positive integers, and let γ generate the multiplicative (cyclic) group $\mathbf{F}_{q^k}^* = \mathbf{F}_{q^k}\backslash\{0\}$. Each of the q^k words of an $[n, k]$ irreducible cyclic code may then be uniquely associated with an element $x \in \mathbf{F}_{q^k}$ and may be written as

$$(\mathrm{Tr}(x), \mathrm{Tr}(x\gamma^N), \mathrm{Tr}(x\gamma^{2N}), \ldots, \mathrm{Tr}(x\gamma^{(n-1)N})), \tag{18}$$

where k is the smallest integer such that $q^k = 1 \bmod n$. For a proof of this statement see [33] or [34].

As stated in Theorem 1, we are essentially interested in obtaining the weight spectrum of $[n, k]$ irreducible cyclic codes. The number of words with different non-zero weight is at most N, where $N = (q^k - 1)/n$ (for a proof see Proposition 1 in Appendix 8). Now let $w(x)$ be the Hamming weight of the code word associated with $x \in \mathbf{F}_{q^k}^*$. The McEliece Theorem connects the weights of words of irreducible cyclic codes to Gauss sums.

Theorem 3 (***McEliece Formula***). *Let $w(y)$ for $y \in \mathbf{F}_{q^k}^*$ be the weight of the code word given by Eq. (18), let $q^k = 1+nN$, where q is prime and k, n and N are positive integers, let $d = \gcd(N, (q^k - 1)/(q - 1))$, and let the multiplicative character $\bar{\chi}$ be given by $\bar{\chi}(\gamma) = \exp(2\pi i/d)$, where γ generates $\mathbf{F}_{q^k}^*$. ($\bar{\chi}$ is called the character of order d.) Then the weight of each word in an irreducible cyclic code is given by*

$$w(y) = \frac{q^k(q-1)}{qN} - \frac{q-1}{qN}\sum_{a=1}^{d-1} \bar{\chi}(y)^{-a} G_{\mathbf{F}_{q^k}}(\bar{\chi}^a, 1). \tag{19}$$

For a proof of this see [33].

The important feature here is that if we had the ability to efficiently estimate $G_{\mathbf{F}_{q^k}}(\chi, \beta)$, then we would be able to find the weights of the words in an irreducible cyclic code efficiently under the restrictions mentioned in Theorem 1. This would in turn allow us to find the weight spectrum $\{A_i\}$ of the code. The following theorem reveals that a quantum computer can efficiently approximate Gauss sums.

Theorem 4 (***van Dam & Seroussi*** [26]). *For any $\epsilon > 0$, there is a quantum algorithm that estimates the phase γ in $G_{\mathbf{F}_{q^k}}(\chi, \beta) = \sqrt{q^k}e^{i\gamma}$, with expected error $E(|\gamma - \tilde{\gamma}|) < \epsilon$. The time complexity of this algorithm is bounded by $O(\frac{1}{\epsilon} \cdot (\log(q^k))^2)$ [26].*

(For details see Appendix 9.)

The Gauss sum algorithm allows one to estimate γ in Eq. (16) to within any accuracy ϵ, i.e., the algorithm returns γ' such that $|\gamma' - \gamma| < \epsilon$. The hope is that if one can approximate γ precisely enough then one would get an exact evaluation of the weight. In fact an essential step here is to use a quantum computer to obtain a list of approximate angles $\{\gamma_t'\}$ for $t = 1, \ldots, d - 1$ for d given above.

The next theorem gives some minimum distance between weights so that we can choose an appropriate error that will allow one to be able to distinguish between weights, which allows us to obtain accurate coefficients for $A(1, y)$ and hence exact values for the exponents.

Theorem 5 (***McEliece*** [35]). *All the weights of an $[n, k]$ irreducible cyclic code are divisible by $q^{\theta_{n,k}-1}$, where $\theta_{n,k}$ is given in Definition 1.*

2.4.2. The proof. We are now ready to prove Theorem 1.

Proof. Assume that a given graph $\Gamma = (E, V)$ is a member of ICCC_ϵ, where $n = |E|$ and $k = |E| - |V| + c(\Gamma)$. Hence it is given that $\epsilon \leq \frac{q^{\theta_{n,k}-1}}{4\sqrt{q^k}} \equiv \epsilon_0$. We want to obtain Z_Γ for either the fully ferromagnetic or anti-ferromagnetic Potts model. It follows from Definition 1 that the dual of the cocycle code of Γ is an irreducible $[n, k]$ cyclic code. We must demonstrate that we can obtain the weight enumerator $A(1, y)$ of this dual code within the claimed number of steps. As mentioned above, since $nN = q^k - 1$ there are then at most N different weights, with at least n words of each weight (see the Appendix). In order to find the spectrum $\{A_i\}$, we are faced with the computational task of finding the range of

$$S(i) = \frac{q^k(q-1)}{qN} - \frac{q-1}{qN} \sum_{a=1}^{d-1} \bar{\chi}(\alpha^i)^{-a} \sqrt{q^k} e^{i\widetilde{\gamma_a}} \tag{20}$$

(where again $d = \gcd(N, q^k - 1/q - 1)$ and $i \in \{0, \ldots, N-1\}$) and then performing a tally.

The proof consists of five main parts:

1. Proof that with ϵ bounded by ϵ_0 as given, it is possible to distinguish between weights of the words of the code that corresponds to the given graph. This ability allows for an exact evaluation of Z_Γ.
2. We need to justify our asymptotic approach and show that for a fixed error ϵ there are a countable number of graphs in ICCC_ϵ.
3. Proof that the success probability δ is as stated in the theorem.
4. Proof that the running time is as stated in the theorem.
5. A transformation from the dual (irreducible cyclic) code to the cocycle code of the graph whose Potts partition function we are evaluating.

Let us now prove each of these five parts.

1. The first question we must address is the following: how small do we need to make the error ϵ in the phases returned in the Gauss sum approximation algorithm so that we will be able to distinguish between weights? We now show that $\epsilon \leq \epsilon_0$ is sufficient, and hence that for every member of the class ICCC_ϵ it is possible to distinguish between weights.

Let $\widetilde{w(y)}$ be the approximated weight returned by the quantum Gauss sum algorithm. It follows from Theorem 5 that two consecutive weights are separated by a distance that is an integer multiple of $q^{\theta_{n,k}-1}$. Hence, a sufficient condition for being able to associate $\widetilde{w(y)}$ with the correct weight $w(y)$ (and not another neighboring weight) is:

$$|w(y) - \widetilde{w(y)}| < \frac{q^{\theta_{n,k}-1}}{2}. \tag{21}$$

Let the error between the actual phase γ_i and the approximated phase $\widetilde{\gamma_i}$ be ϵ, i.e.,

$$|\gamma_i - \widetilde{\gamma_i}| < \epsilon.$$

Let us derive a bound on ϵ. Taking $w(y)$ given in Theorem 3 and the necessary bound given in Eq. (21) we find that we need the inequality

$$\left| \sum_a \bar{\chi}(y)^{-a} e^{i\gamma_a} - \sum_a \bar{\chi}(y)^{-a} e^{i\widetilde{\gamma_a}} \right| < \frac{qN}{(q-1)} \frac{q^{\theta_{n,k}-1}}{2} \frac{1}{\sqrt{q^k}} \tag{22}$$

to be satisfied. Now, we have

$$\begin{aligned}\left|\sum_a \bar{\chi}(y)^{-a} e^{i\gamma_a} - \sum_a \bar{\chi}(y)^{-a} e^{i\widetilde{\gamma}_a}\right| &\leq \sum_a \left|e^{i\gamma_a} - e^{i\widetilde{\gamma}_a}\right| \\ &\leq \sum_a \left(|\cos(\gamma_a) - \cos(\widetilde{\gamma}_a))| + |(\sin(\gamma_a) - \sin(\widetilde{\gamma}_a))|\right) \\ &\leq 2(d-1)\,|\gamma_a - \widetilde{\gamma}_a| < 2(d-1)\epsilon,\end{aligned}$$

where the last inequality follows from the Mean Value Theorem of elementary calculus. Therefore, if we impose

$$\epsilon < \frac{qN}{(q-1)(d-1)} \frac{q^{\theta_{n,k}-1}}{4} \frac{1}{\sqrt{q^k}}, \tag{23}$$

then inequality (22) is satisfied. Consider the factor $\frac{qN}{(q-1)(d-1)}$. Noting from $N \leq \alpha k^s$ that $d = \gcd(N, (q^k-1)/(q-1)) \leq \alpha k^s = N$, it follows that $1 < \frac{qN}{(q-1)(d-1)} \leq \frac{q}{q-1} N = O(k^{s(k)})$. Thus we can replace the bound (23) by the tighter bound

$$\epsilon < \frac{q^{\theta_{n,k}-1-k/2}}{4} = \epsilon_0, \tag{24}$$

and this ϵ is definitely small enough to satisfy the required bound given in Eq. (21).[3] Hence, if $\epsilon < \epsilon_0$ it is possible to resolve the weights $\widetilde{w(y)}$ for different words y. This, in turn, gives us the ability to exactly reconstruct the weight enumerator A, and from there the partition function Z_Γ.

2. We prove the following lemma.

Lemma 1. *Given a fixed $\epsilon < 1$ there are countably many graphs in* ICCC_ϵ*, i.e., there are infinitely many corresponding irreducible cyclic codes $[n_i, k_i]$ such that $\{\theta_{n_i,k_i}\}$ satisfies*

$$\frac{1}{4} q^{\theta_{n_i,k_i} - 1 - \frac{k_i}{2}} > \epsilon. \tag{25}$$

What this means is that there is at least one family of graphs for a given fixed ϵ for which one will be able to obtain the exact Potts partition function. This also justifies the complexity arguments used herein.

Proof. We shall construct one such family and show that it satisfies the required relations. For simplicity take $\epsilon = 4\epsilon$. We must construct one family of graphs for which the corresponding irreducible cyclic codes, $[n_i, k_i]$, satisfy

$$\theta_{n_i,k_i} + \log_q(\epsilon^{-1}) > 1 + \frac{k_i}{2}. \tag{26}$$

[3] Note, however, that when $d = 2$, we have in fact $\epsilon < k^{s(k)}\epsilon_0$, where $\epsilon_0 = \frac{q^{\theta_{n,k}-1}}{4\sqrt{q^k}}$, and in this case the computational cost of the algorithm (see Theorem 1) is scaled down from $O(\frac{1}{\epsilon_0} k^{2s(k)} (\log q)^2)$ to $O(\frac{1}{\epsilon_0} k^{s(k)} (\log q)^2)$, where the upper bound (24) still applies. This means that within the family $\mathrm{ICCC}_{\epsilon_0}$ some instances can be solved faster than others by a factor of $k^{s(k)}$, at fixed ϵ_0.

Take q fixed and consider the following countable set of irreducible cyclic codes: $\{[q^m - 1, k_m]\}_{m=1,2,3,\ldots}$. First we must note that

$$\theta_{q^m-1,k_m} = m. \tag{27}$$

This follows from the properties of addition in base q: the k digits of $q^k - 1$ in base q are all $(q - 1)$, and adding integer multiples of $q^k - 1$ will not decrease the digit sum. i.e.,

$$S'(\eta(q^m - 1)) \geq m(q - 1) \quad \forall \eta \in \mathbb{N}.$$

This is important to keep in mind when we consider extending this family later in this proof. Now we must demonstrate that there is at least one k_m that satisfies Eq. (26). Because we are dealing with irreducible cyclic codes we must have

$$q^{k_m} = 1 \mathrm{mod} n = 1 \mathrm{mod}(q^m - 1).$$

This is trivially satisfied by $k_m = m$ and indeed Eq. (26) becomes $m - \frac{m}{2} > 1 + \log_q \epsilon$, which is clearly true for any $\epsilon < 1$. This family is computationally trivial, however, being that $N = 1$.

Let us now extend this family to include many interesting instances. Let us first consider a fixed code $[q^m - 1, k_m]$ (i.e., $N = 1$, m fixed). Let us next generate a family of codes $\{[\eta_j(q^m - 1), k_{mj}]\}_{j=1,2,\ldots,M}$ by taking integer multiples η_j of $q^m - 1$, and picking $k_{mj} \geq k_m$ such that $\eta_j(q^m - 1)N = q^{k_{mj}} - 1$ (this is just the irreducible cyclic code condition $nN = q^k - 1$). We obtain a finite set of codes ($M < \infty$) because it follows from Eq. (25) that eventually the $\{k_{mj}\}_j$ will become too large for the fixed error ϵ, for each m. We then do this for every $m \in \mathbf{N}$ paying special attention to the integer multiples η_j. The η_j are selected in this construction so that two conditions are satisfied: (i) the corresponding $\{k_{mj}\}_j$ are sufficiently small to ensure that Eq. (25) is satisfied, (ii) that N is bounded by $\{O(k_{mj}^s)\}_{m,j}$.

Regarding (i), the steps above are conveniently summarized as the following loop:
Given ϵ:

1. For $m = 1, 2, \ldots$
   2. Repeat $j = 1, 2, \ldots$
   $n := j(q^m - 1)$
   calculate $k_{mj} = \mathrm{ord}_q(n)$
   if $q^{\theta_{n,k_{mj}} - 1 - k_{mj}/2} < \epsilon$ then reject k_{mj}, otherwise accept k_{mj} and let $\eta_j \equiv j$.
   Until $j = M$

Note that we are guaranteed to find such a non-empty finite set $\{k_{mj}\}_j$ due to the fact that if $\gcd(q, \eta_j(q^m - 1)) = 1$, then $\exists k_{mj} \in \mathbf{N}$ such that $q^{k_{mj}} = 1 \bmod \eta_j(q^m - 1)$ (see, e.g., Th. 7-1 of [36]).

Regarding (ii), we still need to show that there exist solutions N_{mj} to $q^{k_{mj}} - 1 = nN_{mj}$ that scale as $O(k_{mj}^s)$. To see why such solutions exist consider solving $q^k - 1 = nN$ with $N = \alpha k^s$ and $n = \eta(q^m - 1)$, where $\alpha \in \mathbf{R}$ (we have dropped the subscripts for simplicity). The solution is

$$m = \log_q[(q^k - 1)/(\alpha \eta k^s) + 1]. \tag{28}$$

In the loop above, only those m's satisfying Eq. (28) are acceptable in terms of the scaling of our algorithm. However, note that asymptotically Eq. (28) yields $m = k - s\log_q k - \log_q \alpha\eta$. This means that for every value of k and s it is possible to adjust α such that m is an integer by letting $s\log_q k = \log_q \alpha\eta$.

At this point we have constructed an infinite family of pairs $[n, k_{mj}]$ [where $n = j(q^m - 1)$ and where m satisfies Eq. (28)], each of which defines a graph which is a member of the set ICCC_ϵ. □

Finally, we should mention without proof, that one can "fill" this family of graphs by considering the multitude of cases which do not conform to the restrictions in this construction, but which do obey relation (26) and obey the asymptotic conditions given in definition (1). Moreover, the graphs we have constructed are quite sparse but they are only a subset of ICCC_ϵ. There are many more interesting graphs that can be handled by this fixed error bound. For example, graphs which are the direct sum of many copies of a smaller graph are excluded from this family. Further one may accept an error ϵ that decreases polynomially in k for example and define a family of graphs in that way. We do not pursue this here.

3. In the van Dam-Seroussi algorithm (Theorem 1 in [26]), a prepared state must must go through a phase estimation. In [37] it is demonstrated that if the number of qubits used in phase estimation is $t = \log 1/\epsilon + \log(2 + 1/(2\delta))$ then the probability of success is at least $1 - \delta$. Ref. [38, p. 7] states that for the Gauss sum algorithm $t = 2\log(q^k - 1)$. After some elementary algebra we obtain $\delta = [2((q^k - 1)^2\epsilon - 2)]^{-1}$. By the Chernoff bound, for fixed problem size k, we only need to pick ϵ such that the probability of failure δ is less than $1/2$.

4. (a) We have that if α is a generator for $\mathbf{F}^*_{q^k}$ and if $i = j \bmod n$, then the code words associated with α^i and α^j are cyclic permutations of each other, and therefore are of the same weight. Let us denote by $[\alpha^i]$ the (equivalence) class of all words $\{\alpha^j\}_j$ with $i = j \bmod n$. In this step we wish to find the weight of $[\alpha^i]$. This weight is given by

$$S(i) = \frac{q^k(q-1)}{qN} - \frac{q-1}{qN}\sum_{a=1}^{d-1} \bar{\chi}(\alpha^i)^{-a}\sqrt{q^k}e^{i\widetilde{\gamma}_a}. \tag{29}$$

Hence (up to irrelevant classical computations) the computational cost of computing $S(i)$ is $d - 1$ times the cost of computing $\widetilde{\gamma}_a$. For any graph in ICCC_ϵ, obtaining these $d - 1$ phases has a (quantum) cost of $O(dk^2(\log q)^2)$, where d is bounded above by N. This comes from the complexity of computing the Gauss sum d times. (Recall that one has to repeat this algorithm $1/\epsilon$ times in order to ensure that we obtain a sufficiently close approximation.)

(b) How many times must we compute $S(i)$? The number of times is the number of different equivalence classes $\{[\alpha^i]\}$. Each equivalence class $[\alpha^i]$ is clearly of size n, and there are $q^k - 1$ words. Recall that $nN = q^k - 1$ for non-degenerate irreducible cyclic codes, and hence N is the number of different equivalence classes. (Actually the answer to "How many times must we compute $S(i)$?" is that one must only do this for the number of cyclotomic cosets of N – see Subsect. 2.6).

(c) For given $S(i)$ we must compute a sum over d terms. The cost of computing each such term is constant once we have obtained the phases $\widetilde{\gamma}_a$ [which we have, in Step (a)]. Combining this with Step (b), we see that the total cost of computing all $S(i)$'s is $(d - 1)N$.

At this point the total computational cost is therefore $\max[O(dk^2(\log q)^2), O(dN)]$. We choose $N = O(k^{s(k)})$ so if one takes $s(k)$ to be a constant, then the algorithm is

polynomial in k. Thus, the overall time complexity is $O(d \cdot k^{\max[2,s(k)]}(\log q)^2)$. Being that $d \leq N = O(k^{s(k)})$, the complexity is ultimately $O(k^{2\max[1,s(k)]}(\log q)^2)$.

(d) We now have the list $\{S(i)\}$. Next, a tally of all the weights has to be done which has complexity $O(k^{s(k)/2})$ using quantum counting [39]. The tally will return all the weights and counts of each weight (see Sect. 4) which are the exponents and coefficients respectively, of the polynomial $A(1, y)$ which is the weight enumerator of the dual of the cocycle code. Note that this step does not effect the overall complexity of the algorithm as it has a smaller running time than the previous steps.

5. Note that so far we have dealt with the $[n, k]$ irreducible cyclic code that is the dual of the cocycle code of Γ, i.e., we have used $n = |E|$ and $k = |E| - |V| + c(\Gamma)$. However, recall that $\Gamma = \Gamma(E, V)$, and hence corresponds to the $[n, n-k] = [|E|, |V| - c(\Gamma)]$ code, i.e., the cocycle code of the graph Γ as desired. (This correspondence means that we can obtain information about interesting graphs by considering codes of smaller dimension.) Thus, in order to complete the proof we need the weight enumerator of the $[n, n-k]$ cocycle code itself, so that we can apply Theorem 2. The relation between the weight enumerator A of a code C over the field $\mathbf{F}_{q^k}$, and the weight enumerator $A^{\perp}$ of the dual code $C^{\perp}$ is given by the MacWilliams Theorem [31]:

$$A^{\perp}(1, x) = q^{k(k-n)} \left(1 + (q^k - 1)x\right)^n A(1, y), \tag{30}$$

where

$$x \equiv \left(\frac{1 - y}{1 + (q^k - 1)y}\right). \tag{31}$$

Applying the MacWilliams theorem and Barg's theorem [specifically Eq. (15) to $A^{\perp}(1, x)$], we arrive at the partition function

$$Z(x) = x^{-n} q^{c(\Gamma)} A^{\perp}(1, x). \tag{32}$$

Recall that $y = e^{-\beta J}$ (where $\beta = \frac{1}{k_B T}$); thus we have the following final expression for the partition function as a function of β:

$$Z(x(\beta)) = q^{c(\Gamma)+k(k-n)} \left[(q^k - 1) + x(\beta)^{-1}\right]^n A(1, y(\beta)). \tag{33}$$

It is simple to verify that given any temperature $T \geq 0$, and for both positive and negative J, $Z(x(\beta))$ is always positive, as it should be. □

2.5. Proof of the corollary. We now give the proof of Corollary 1.

Proof. Assume that we are given a graph $\Gamma(E, V)$ whose CMM is the direct sum of the CMMs of two graphs Γ_1 and Γ_2 in ICCC_ϵ (we call such a graph Γ a "composite graph"). Let C be the code that corresponds to the graph Γ, i.e., C is the cocycle code of Γ. Let C_1 and C_2 be the corresponding cocycle codes of Γ_1 and Γ_2. This means that we may apply our algorithm to each of these sub-graphs and obtain their weight enumerators. To do this we need to obtain the weight enumerators of C_1 and C_2 which we can do efficiently. By definition $C = C_1 \oplus C_2$. If the respective lengths and dimensions of C_1 and C_2 are $[m, l]$ and $[m', l']$, then C is an $[m + m', l + l']$ linear code and its weight enumerator will be $W = W_1 W_2$ [31]. Thus, once one obtains the weight enumerators of the sub-graphs, one has the weight enumerator of Γ and by using the arguments already outlined one can see that we can efficiently compute Z_Γ. □

The above corollary allows the scheme outlined in this paper to be efficiently applied to many graphs because if one knows the generator matrices for C_1 and C_2 then one can efficiently construct the generator matrix for C by just taking the direct sum of the matrices. This gives a way of constructing examples of graphs for which the Potts partition function can be efficiently approximated. On the other hand (recall Subsect. 2.3), we can efficiently check if a generator matrix decomposes into a direct sum of smaller matrices and we can efficiently check if these matrices generate codes whose duals are irreducible cyclic.

2.6. Reducing the computational cost of the algorithm via permutation symmetry. We now briefly review the concept of p-cyclotomic cosets. As an introduction see [31]. Consider the set of integers $\{0, 1, 2, \ldots, N-1\}$ and take p a prime number such that p does not divide N. The p-cyclotomic cosets of this set are given by the collection of subsets

$$\{0\}, \{1, p, p^2, \ldots, p^{r(1)}\}, \ldots, \{j, jp, jp^2, \ldots, jp^{r(j)}\}, \ldots,$$

where j is an integer and $r(j)$ is the smallest integer such that $j(p^{r(j)} - 1) = 0 \bmod N$, i.e., $r(j)$ is the smallest integer before one begins to get repeats in the coset indexed by j. The number of cosets is finite so j is finite. As an example consider $N = 16$ and $p = 3$. One obtains

$$\{0\}, \{1, 3, 9, 11\}, \{2, 6\}, \{4, 12\}, \{5, 15, 13, 7\}, \{8\}, \{10, 14\}.$$

With regards to the scheme presented in this paper, we take q-cyclotomic cosets. We are guaranteed that $\gcd(N, q) = 1$, which ensures that in our case the cyclotomic cosets are disjoint. That $\gcd(N, q) = 1$ is due to the fact that there are solutions $x, y \in \mathbf{Z}$ to $Nx + qy = 1$ (Thm. 2-4 of [36]). For example, since $N = \frac{q^k-1}{n}$, one can take $x = n(q-1)$ and $y = 1 + q^{k-1} - q^k$, which are both integers. The relevance of the q-cyclotomic cosets of $\{0, \ldots, N-1\}$ is that each element in a given coset has the same value of $S(i)$. This is because of the fact that the mapping $x \mapsto x^q$ is a permutation of $\mathbf{F}_{q^k}$ and that the additive characters obey the identity $\exp(2\pi i \mathrm{Tr}(b^q)/q) = \exp(2\pi i \mathrm{Tr}(b)/q)$ for all $b \in \mathbf{F}_{q^k}$. Hence $S(i)$ is invariant under the mapping $x \mapsto x^q$. (See Appendix 9.1 for details on additive characters and the trace function Tr.) Therefore we only have to evaluate $S(i)$ for one i in each coset. The computational cost of computing the coset representatives and the number of elements in each coset is linear in N [40]. This has the potential of significantly speeding up the algorithm, but how much will clearly depend on the number of cosets generated by each instance. The number of cosets is given by [41]

$$N_C = \sum_{f|N} \frac{\phi(f)}{\mathrm{ord}_q f}, \tag{34}$$

where $\phi(f)$ is the Euler totient (the number of positive integers which are relatively prime to f and $s = \mathrm{ord}_q f$ means that s is the smallest positive integer such that $q^s = 1 \bmod f$). Note that N_C replaces N in the overall computational cost of our algorithm and $N_C \leq N$. While this can lead to a significant speedup in some cases, for the sake of simplicity and of having uniform bounds we will not pursue this further here.

As an illustration of the power of using cyclotomic cosets, consider the following numerical example. Let $q = 2$, $1/\epsilon \geq 8192$, and consider a binary [113, 85] code which

is the dual to a binary [113, 28] irreducible cyclic code (i.e., 28 is the smallest integer such that $2^{28} = 1 \bmod 113$). This corresponds to either the fully ferromagnetic or fully anti-ferromagnetic Ising model on a graph with 113 edges and 86 vertices. Now note that $nN = 2^{28} - 1$, which implies that $N = 2375535$. Without the use of cyclotomic cosets this value of N would set our computational cost in that it is the number of times that $S(i)$ must be queried. However, it turns out that there are $N_C = 85439$ cyclotomic cosets, and this is the actual number of queries to $S(i)$.

Note that there are instances where $N \ll n$ and cyclotomic cosets are not required. For example consider the binary [13981, 20] irreducible cyclic code. Here $n = 13981$ and $N = 75$. Physically this corresponds to either the fully anti-ferromagnetic or ferromagnetic Ising model over a connected graph with 13981 edges and 13962 vertices (considering the dual code).

3. Classical and Quantum Complexity of the Scheme

Assuming one knew that a given graph was a member of ICCC_ϵ, then classically one could proceed as follows using a state of the art algorithm ZETA for the computation of zeta functions of the family of curves $C_\alpha : y^q - y = \alpha x^N$ [24]. Here N is as given in the relation $nN = q^k - 1$ and the index α is in one-one correspondence with the code words in the given cocycle code (specifically $\alpha \in \mathbf{F}_{q^k}$). The connection between the weights of words of an irreducible cyclic code and the number of rational points on the curves C_α is well known, as is the connection between the zeta functions of such curves and Gauss sums [42]. The complexity of using ZETA to compute the $N = \alpha k^{s(k)}$ different weights is $O(k^{6s(k)+3+\epsilon'}\left(\frac{q}{2}\right)^{5+\epsilon'})$ [24] and a tally of these weights will take $O(k^{s(k)})$ operations (ϵ' is a small real number – unrelated to ϵ which parameterizes the class of graphs in question). The overall complexity of finding the range of $S(i)$ will therefore be

$$\text{classical cost} = O\left(k^{6s(k)+3+\epsilon'}\left(\frac{q}{2}\right)^{5+\epsilon'}\right), \tag{35}$$

assuming that we know that a given graph is a member of ICCC_ϵ. As far as we know this is the fastest classical algorithm for the problem we have considered here.

For a quantum computer we do not need to assume that testing for membership is efficient: we know that this can be done efficiently using the discrete log algorithm [1]. Above we showed that the overall complexity of finding Z is bounded by $O(k^{2\max[1,s(k)]}(\log q)^2)$. This should be contrasted with the best classical result available, (35). For example if we take $s = 2$, (both classical and quantum methods are polynomial when we take $s(k)$ to be a constant) we obtain an $O(k^{11})$ improvement and an exponential speedup in q. One could imagine fixing a graph and calculating the partition function for increasing values of q. In this situation we have an exponential speedup over the best classical algorithm available.

Note that there is a quantum algorithm for finding zeta functions of curves which is exponentially faster in q than the classical algorithm in [24] (as is ours). This is given in [25]. The use of this algorithm instead of the Gauss sum approximation algorithm is left for a future publication.

On a final note, the classification ICCC_ϵ we have chosen is meant to highlight the boundary between BQP and P by fixing the acceptable error in the Gauss sum phases. One could opt for a perhaps more natural class of graphs by bounding the way that $1/\epsilon$

grows instead. For example, one could restrict the class of graphs in such a way that $1/\epsilon \sim q^{\frac{k}{2}-\theta_{n,k}+1}$ grows polynomially in k, in particular such that

$$\frac{1}{\epsilon} < k^{5s(k)+1}.$$

For this class of graphs one would also have a speedup in the quantum case.

4. Detailed Summary

For convenience we recollect our definitions and provide a diagram of our scheme. We are considering the q-state Potts model (fully ferromagnetic or fully anti-ferromagnetic) over a graph $\Gamma = (E, V)$, with q prime. This includes the Ising model ($q = 2$). Every graph Γ has a cycle matroid $M(\Gamma)$ associated with it and every cycle matroid has a $(|V| - c(\Gamma)) \times |E|$ matrix representation G (the CMM), where $c(\Gamma)$ is the number of connected components of Γ. The columns of G encode the cycle structure of the graph and the row space of G generates the cocycle code of length $|E|$ and dimension $|V| - c(\Gamma)$. The length and dimension of the dual code are respectively $n = |E|$ and $k = |E| - |V| + c(\Gamma)$.

Following is a detailed synopsis of the algorithm for computing the partition function.

1. Given a graph, efficiently determine if it belongs to ICCC_ϵ (Definition 1). This step appears to be hard on a classical computer in general, since it is equivalent to computing a discrete log.
2. If the CMM $G = [I_{|V|-c(\Gamma)}|X]$ is the matrix representation over $\mathbf{F}_{q^k}$ of the cycle matroid of Γ, $M(\Gamma)$, then the row space of $H = [-X^T|I_{|E|-|V|+c(\Gamma)}]$ will be the code $C(\Gamma)$.
3. Let $N = O(k^s)$, where s is a constant integer that determines the complexity of the algorithm. Take $C(\Gamma)$ as an irreducible cyclic code of length $n = \frac{q^k-1}{N}$ and dimension k, i.e., we only consider graphs Γ, where $C(\Gamma)$ is an irreducible $[n, k]$ cyclic code.
4. If we can evaluate the weight enumerator of $C(\Gamma)$ we will have successfully approximated the Potts partition function over the corresponding graph Γ. To do so:
   (a) Find the q-cyclotomic cosets of $\{0, 1, \ldots, N-1\}$. This step requires at most linear time in N.
   (b) Using the quantum algorithm for Gauss sums [26] we are be able to estimate the weights of the words. The error in the Gauss sum algorithm can be high in this setting, and therefore we have to restrict the class of graphs further in order to obtain exact evaluations. Use the Gauss sum algorithm to return the phases $\gamma_1, \ldots, \gamma_{d-1}$ [Eq. (16)] and then input these values into the function $S(i)$ [Eq. (20)]. According to the McEliece Theorem (Th. 3) we have to make $d - 1$ (where $d = \gcd(N, \frac{q^k-1}{q-1})$) calls to the quantum oracle and we can use these evaluations for each representative i of the q-cyclotomic cosets of $\{0, 1, \ldots, N-1\}$. This step has time complexity $O(dk^2(\log q)^2)$.
   (c) Let $b_1, b_2, \ldots, b_{N_C}$ be the coset representatives from the N_C cosets. Now each coset has cardinality v_i, i.e., b_i belongs to coset i which has v_i elements. We evaluate $\omega_i = S(b_i)$ for each b_i, remembering that each ω_i occurs v_i times. We end up with a list $(\omega_1, \omega_2, \ldots, \omega_{N_C})$ as well as a list $(v_1, v_2, \ldots, v_{N_C})$ of multiplicities.

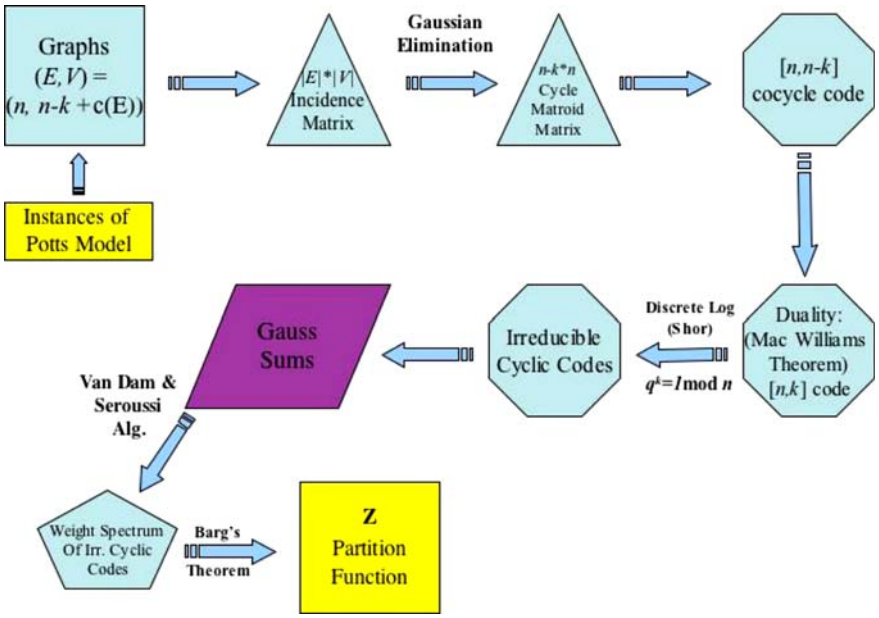

Fig. 2. A diagrammatic overview of the algorithm. (Box shapes do not have a meaning.)

(d) Now perform a tally of repeats of the ω_i for each $i \in \{1, \ldots, N_C\}$. This returns a set of indices $\Lambda_i \equiv \{j_i\} \subseteq \{1, \ldots, N_C\}$. We add the corresponding v_{j_i} which yields $a_i = \sum_{j \in \Lambda_i} v_j$, the number of words of weight ω_i up to cyclic permutations. To account for cyclic permutations due to the fact that we are working over cyclic codes, we have $A_i = na_i$, which is the desired weight spectrum.

5. Now that we have determined the weight spectrum A_i in time $O(k^{2s}(\log q)^2)$, we have the coefficients for $A(1, y)$ and so via the MacWilliams identity (30) we finally obtain the partition function (33).

5. Examples and Discussion

In this section we provide the reader with some simple examples for illustrative purposes.

5.1. Example. Consider the graph depicted in Fig. 3. This graph depicts three spins, one of which has a self-interaction. It can be verified that this graph corresponds to the dual of a [4, 2] irreducible cyclic code over $GF(3)$, i.e., $q = 3$. The generator matrix for this code is given by

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{pmatrix}.$$

We see that the corresponding graph must have 4 edges and 3 vertices (if the graph is connected), and this is the reason for having the spin with the self-interaction. The second, third, and fourth columns correspond to a triangle (as they sum to zero modulo

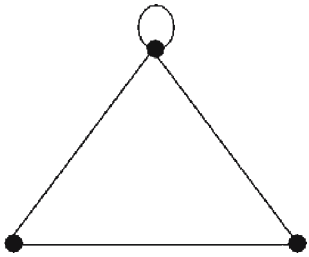

Fig. 3. A graph corresponding to a [4, 2] linear code over $GF(3)$

3) and the first column is the loop at one of the vertices. The self-interaction can be removed once the partition function has been obtained via a simple procedure described below.

We need to find the weight spectrum for this code. After forming the weight enumerator using MacWilliams identity, we apply Barg's theorem which will give the $q = 3$ Potts partition function for this graph.

1. Using a quantum computer we evaluate the necessary Gauss sums. From the identity $q^k - 1 = nN$ (necessarily satisfied by irreducible cyclic codes) we see that $N = 2$. This means that there can be at most two different weights (in fact the number of non-zero cyclotomic cosets is one).
2. Compute the number of times that one must repeat the quantum algorithm for Gauss sums in order to obtain an acceptable accuracy. We see that this number is given by

$$\frac{1}{\epsilon} = \frac{4\sqrt{3^2}}{3^{\theta_{n,k}-1}}.$$

   Since $4 = 11$ base 3 we have that $\theta_{n,k} = \frac{1}{2}[1 + 1] = 1$ and so $\frac{1}{\epsilon} = 12$. This means that the algorithm must be repeated 13 times to ensure the desired accuracy.
3. After evaluating the Gauss sums and plugging them into Eq. (19), we obtain two weights: 0 and 3.
4. As only one word can have zero weight, the remaining $3^2 - 1$ words have weight 3. This means that we have the weight enumerator

$$A(1, y) = 1 + 8y^3.$$

5. Using relation (33) derived earlier, we find that for this graph

$$Z(x(\beta)) = \frac{1}{27}\left[8 + x(\beta)^{-1}\right]^4 [1 + 8y^3(\beta)],$$

   where $x(\beta) = \frac{1-y(\beta)}{1+8y(\beta)}$, $y = e^{-\beta J}$, and $\beta = \frac{1}{k_B T}$.
6. At this point we can remove the self-interaction by dividing $Z(x(\beta))$ by y. This is due to the following theorems.

Theorem 6. *Let T be the Tutte polynomial. If e is a loop then*

$$T(M; x, y) = yT(M - e; x, y),$$

where $M - e$ is the matroid (or graph) with the loop deleted [6].

Theorem 7.

$$A(1, y) = y^{n-k}(1-y)^k T\left(M; \frac{1+(q-1)y}{1-y}, \frac{1}{y}\right).$$

This is known as Greene's identity and one can see either [27] *or* [6] *for details.*

Putting these theorems together one finds that

$$A_{M-e}(1, y) = y^{n-k}(1-y)^k T\left(M-e; \frac{1+q-1y}{1-y}, \frac{1}{y}\right) \tag{36}$$

$$= y^{n-k}(1-y)^k yT\left(M; \frac{1+(q-1)y}{1-y}, \frac{1}{y}\right) \tag{37}$$

$$= yA_M(1, y), \tag{38}$$

and therefore we find that the partition function for the triangle is then given by

$$Z(x(\beta)) = \frac{1}{27}\left[8 + x(\beta)^{-1}\right]^4 \left(\frac{1-x}{1+(q^k-1)x}\right)[1+8y^3(\beta)].$$

One should note that due to Corollary (1), we could form a string of these triangle graphs as shown in Fig. 4, and easily compute the partition function by multiplying the above partition function with itself three times (the number of copies of the triangle in the chain). This property is shared by all instances of the Tutte polynomial defined over direct sums of matroids.

We can extend this to certain types of recursively defined graphs [46] by forming chains made of multiple copies of different graphs. We note however that recursively defined graphs [46] do not always fit into our construction because they may not be members of ICCC_ϵ. For example, consider Fig. 5. This is known as a ladder graph and it is an example of a recursively defined graph. This graph corresponds to a [8,17] binary linear code which is not irreducible cyclic, nor dual to one [27].

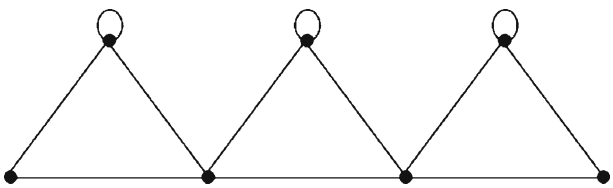

Fig. 4. Chaining of the graph in Fig. 3

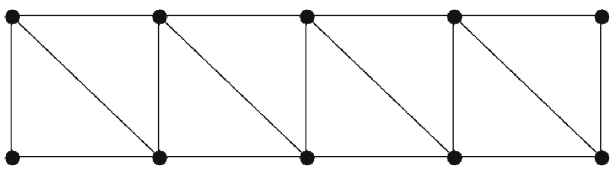

Fig. 5. A ladder graph illustrating a recursively defined graph. This graph corresponds to a [8,17] binary linear code

5.2. Degenerate cyclic codes. Here we introduce an approach to construct examples that will help to classify the types of graphs that our scheme is tailored for. The motivation for this is to clarify the relationship between graphs and codes in the sense used in our scheme. The problem is the fact that many of the irreducible cyclic codes have duals that are not graphic in the sense of cycle matroids. We ask the following question: Given an irreducible cyclic code whose dual is not graphic (and hence does not correspond to a physical Potts model), can we find another code whose dual has a weight spectrum that is simply related to the original code, and which is graphic? We provide some arguments in favor of this idea.

There exist codes whose words consist of several repetitions of a code of smaller length. Of particular interest to us is a class of degenerate codes related to irreducible cyclic codes in the following way. Lemma IV.2 in [45] states that a code of length n is degenerate if $w(x)|x^r - 1$ [i.e, $w(x)$ divides $x^r - 1$] for some $r|n$, where $w(x)$ is the check polynomial (see Appendix 8). In the case of irreducible cyclic codes the check polynomial is the denominator of the generator polynomial introduced earlier, given by $g(x) = \frac{x^r-1}{w(x)}$. This means that if we have an $[r, k]$ irreducible cyclic code with check polynomial $w(x)$, we find some n such that $w(x)|x^n - 1$ such that $r|n$. We then have a degenerate linear $[n, k]$ code generated by $\frac{x^n-1}{w(x)}$. The words in the degenerate code will look like $(c', c', \ldots, c')$, where c' is a word in the non-degenerate code. This means that once we know the weight distribution of the $[r, k]$ code, we can easily construct the weight enumerator of the $[n, k]$ code since the weights of the words of length n will be n/r times the weight of the corresponding word of length r. This construction allows one to loosen the constraints on the dimension and length and therefore on the number of vertices and edges of the corresponding graph. In other words, for many of the codes whose corresponding cycle matroids are not graphic we may use this construction to map these instances to graphic matroids. The definition for ICCC_ϵ can be easily tailored to include these graphs as will be done in future work.

As an example consider the [4, 5] irreducible cyclic code whose check polynomial is $w(x) = 1 + x + x^2 + x^3 + x^4$. The dual of this code is non-graphic, because it requires forming a cycle of five edges with only two vertices. Now notice that $w(x)|x^{15} - 1$ and $5|15$. In this way we form the [4, 15] code generated by $\frac{x^{15}-1}{w(x)}$. The dual of this code is a [11, 15] code and the corresponding graph is given by Fig. 6. The weight enumerator of the [4, 5] code is $A(1, y) = 1 + 10y^2 + 5y^4$ and the weight enumerator of the degenerate code is $1 + 10y^6 + 5y^{12}$. Note that the exponents are just multiplied by $n/r = 3$. The structure of this graph gives one a clue as to the structure of the types of graphs addressed by our approach. They will be graphs which consist of several repetitions of simple cycles of different lengths. In the example above all the simple cycles have length six, as can be seen in Fig. 6. As one explores codes with higher n, one finds that there will be multiple simple cycles of different lengths that will form the corresponding graph. The reason to believe this to be true in general comes from the fact that the weights of the code C correspond to the size of sets of linearly dependent columns of the generator matrix of the code dual to C. For example, the minimum weight of a code C is the size of the smallest set of linearly dependent columns of the code's parity check matrix, which can be used as the generator matrix of the code dual to C. On the other hand, the length of the cycles (number of edges) are given by the weights or sums of the weights.

The relation between codes and graphs is not yet well understood and future work in this regard based on our approach will hopefully reveal new results that will have applications to both statistical mechanics and knot theory.

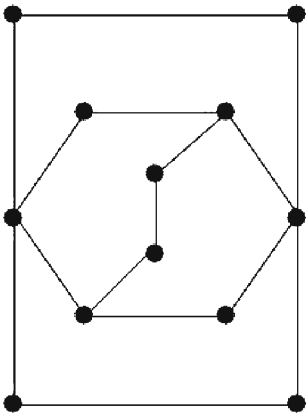

Fig. 6. Example of a graph corresponding to a [11, 15] code related to the [4, 5] irreducible cyclic code

6. Conclusions and Future Directions

In this work we have given a quantum algorithm for the exact evaluation of the fully ferromagnetic or anti-ferromagnetic Potts partition function Z under the restriction to certain sparse graphs (with logarithmically more edges than vertices). The methods we used exploit the connection between coding theory and statistical physics. The motivation for this work is an ongoing effort to identify instances of classical statistical mechanics for which quantum computers will have an advantage over classical machines.

The approach we described involves using the link between classical coding theory and the Potts model via the weight enumerator polynomial A. One should note that A is another instance of the Tutte polynomial and so this connection is not surprising. The weight enumerator encodes information about all the different Hamming weights of the code words in a linear code and the weight of a code word can be given by a formula involving a sum of Gauss sums when dealing with a specific type of linear code. Since there exists an efficient algorithm to approximate Gauss sums via quantum computation [26] we were able to efficiently calculate the weights of code words for certain codes. Much of this paper dealt with the necessary restrictions that one must impose in order to achieve this last step. For example, once an error ϵ in the Gauss sum algorithm is accepted, we demonstrated that there is a family of graphs for which one can find the exact partition function, and therefore the error does not scale within this family. Given a graph Γ, one can map the graph to a corresponding linear code via the incidence structure of Γ. The Potts partition function of Γ (with either fully ferromagnetic or anti-ferromagnetic interactions) is given by some easily computed function times the weight enumerator of the corresponding code. Due to the symmetries inherent in the mathematical structure of linear codes we were able to provide an efficient method to exactly determine Z for a class of graphs (ICCC_ϵ) which has a well defined correspondence to a subset of linear codes.

In [43] it was shown that the exact evaluation of weight enumerators for binary linear codes is hard for the polynomial hierarchy. As our approach involved the exact evaluation of weight enumerators, it is not surprising that we had to make restrictions on the class of graphs so as to make our scheme efficient. The vantage that coding theory gives to this particular problem, however, allows one to utilize the fact that certain graphs have properties that a quantum computer can take advantage of to provide a speed up.

Notice that the related results in [17,21] concern *additive approximations*; the methods used in this paper can be extended to a wider class of graphs if one relaxes the requirement of exact evaluation and instead similarly considers additive approximations of Z. An open question is what instances of the Potts partition function are amenable to an fpras (fully polynomial random approximation scheme). The methods used in [17,21] have proven to be quite powerful. There is hope to extend some of these methods to non-planar graphs. One idea is to extend the algorithm in [17] to the Jones polynomial for virtual knots and then use some correspondence between the virtual knots and non-planar graphs. Another approach may involve seeing things in a new light. Note that the Jones polynomial is the Euler characteristic of a certain chain complex [44]. One can explore how effective quantum computers will be at approximating Euler characteristics in general. Perhaps there is a way of exploiting this in order to obtain knowledge about the Potts partition function.

One may also consider strengthening the results given here by exploiting theorems about the minimal distance of cyclic codes. For example, there are theorems that guarantee a lower bound for the weight between any two words. By enforcing that the generator polynomial of the code be of a certain form, one would be guaranteed a certain distance between words and therefore the error in the Gauss sum approximation will be of little consequence for certain graphs [31]. As already mentioned in the Introduction, another potentially promising approach is to consider the scheme we have presented here but to replace the Gauss sum algorithm with the quantum algorithm for obtaining Zeta functions [25]. Work has to be done on understanding the exact cost of this algorithm when one is restricted to curves that are pertinent for the evaluation of the Potts model.

Corollary 1 deals with the combination of graphs via a direct sum of codes and gives one a way of "tiling" graphs for which one knows the partition function. This gives a quick way of obtaining the partition function of certain graphs that are made of many repeats of a simpler graph. There are other ways of combining codes that may allow one to study the partition function of new graphs, for example the concatenation or direct product of two codes [31].

The coding theoretic approach does give us a way of evaluating the partition function of instances of the Potts model at arbitrary temperatures but precisely the kinds of graphs which are involved is a question for future research. Indeed, the identification of the physical instances represented by the graphs for which our algorithm is efficient will shed light on the question that motivated this work in the first place [5]: what is the quantum computational complexity of classical statistical mechanics?

Acknowledgement. J.G. would like to thank Marko Moisio, Ravi Minhas, and Frank van Bussel for helpful discussions. D.A.L. gratefully acknowledges support under ARO grant W911NF-05-1-0440.

Appendix

7. Matroids

Definition 10. *A matroid M on a set E is the pair (E, I), where I is a collection of subsets of E with the following properties*:

1. *The empty set is in I.*
2. *Hereditary Property: If $A \in I$ and $B \subset A$, then $B \in I$.*
3. *Exchange Property: If A and B are in I and A has more elements than B, then $\exists a \in A$ such that $a \notin B$ but $B \cup \{a\} \in I$.*

The collection of sets in I are called the independent sets and E is referred to as the ground set.

Definition 11. *A cycle matroid of a graph Γ is the set of all edges of Γ as the ground set E together with I as the subsets of E which do not contain a cycle. So the independent sets are collections of edges which do not have cycles.*

Recall that in graph theory one refers to such an edge set (the above independent set) as a forest.

In matroid theory a matrix representation is a matrix whose column vectors have the same dependence relations as the matroid it is representing. More clearly, the column vectors represent the matroid elements and the usual notion of linear dependency determines the dependent sets and therefore the independent sets as well. Thus, the matrix can be said to generate the matroid.

As an example, imagine the triangle graph of three nodes with three edges A, B, and C. The cycle matroid consists of each of the edges individually and any collection of two edges. All three edges form a cycle so it cannot be included. We require our matrix representation to encode this independence structure of the edges. One may work over any field here because we are only concerned with graphic matroids, i.e., matroids which can be represented as a cycle matroid of some graph. (Graphic matroids are representable over any field [29].) Now, if we think of column 1,2 and 3 as edges A, B and C respectively we can take the following matrix as a representation in $\mathbf{F}_2$:

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Since addition is mod 2 here, a cycle is any collection of columns that sum to the 0-vector. We can take all collections where this does not happen and these collections will form I. In this way, this matrix is a representation of the cycle matroid for the triangle graph. In matroid theory one has the familiar notion of a base.

Definition 12. *A base of a matroid $M = (E, I)$ is a maximal independent subset of E.*

It is not a coincidence that the left part of the matrix is the 2×2 identity matrix. In general one can form a representation (known as the standard matrix representation) where one begins with an identity matrix which is $r \times r$, where r is the size of the base of M and append to it columns that capture the dependence structure of the matroid in question. In this way, the columns of the identity matrix represent the chosen basis of M. So M is isomorphic to the matroid induced on the columns of the matrix by linear dependence. A more precise explanation can be found in [29]. What is important for us is that such a matrix representation is possible.

8. Algebraic Approach to (Irreducible) Cyclic Codes

8.1. Irreducible cyclic codes as minimal ideals. Let us recall some definitions from algebra. Take q to be prime or a power of a prime.

Definition 13. *A ring is a set R which is an abelian group $(R, +)$ with 0 as the identity, together with $(R, \times)$, which has an identity element with respect to $\times$ where $\times$ is associative.*

Definition 14. *An ideal I is a subset of a ring R which is itself an additive subgroup of $(R, +)$ and has the property that when $x \in R$ and $a \in I$ then xa and ax are also in I.*

Definition 15. *A principle ideal is an ideal where every element is of the form ar, where $r \in R$.*

Thus, a principle ideal is generated by the one element a and a principal ideal ring is a ring in which every ideal is principle.

There is an important isomorphism between powers of finite fields $\mathbf{F}_q^n$ and a certain ring of polynomials. Recall that the multiples of $x^n - 1$ form a principal ideal in the polynomial ring $\mathbf{F}_q[x]$.

Therefore the residue class ring $\mathbf{F}_q[x]/(x^n - 1)$ is isomorphic to $\mathbf{F}_q^n$ since it consists of the polynomials

$$\{a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} | a_i \in \mathbf{F}_q, 0 \leq i < n\}.$$

Therefore, taking multiplication modulo $x^n - 1$ we can make the following identification:

$$(a_0, a_1, \ldots, a_{n-1}) \in \mathbf{F}_q^n \longleftrightarrow a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \in \mathbf{F}_q[x]/(x^n - 1). \quad (39)$$

This implies the following theorem.

Theorem 8. *A linear code C in $\mathbf{F}_q^n$ is cylic $\iff$ C is an ideal in $\mathbf{F}_q[x]/(x^n - 1)$* [31].

Proof. In one direction this is easy since if C is an ideal in $\mathbf{F}_q[x]/(x^n - 1)$ and $c(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$ is a codeword, then by definition $xc(x) \in C$ as well and so $(a_{n-1}, a_0, a_1, \ldots, a_{n-2}) \in C$. In the other direction, one just has to note that since C is cyclic, $xc(x)$ is in C for every $c(x) \in C$ which means that $x^k c(x)$ is in C for every k. But C is linear by assumption so if $h(x)$ is any polynomial then $h(x)c(x)$ is in C and thus C is an ideal. □

Note that $\mathbf{F}_q[x]/(x^n - 1)$ is a principal ideal ring and therefore the elements of every cyclic code C are just multiples of $g(x)$, the monic polynomial of lowest degree in C; $g(x)$ is called the generator polynomial of C. Because of the correspondence (39) above we know that given $g(x) = g_0 + g_1 x + \cdots g_{n-k}x^{n-k}$ [$g(x)$ divides $x^n - 1$ since otherwise $g(x)$ could not be the monic polynomial of lowest degree in C], we have the vector $(g_0, g_1, \ldots, g_{n-k})$. We then can write the $k \times n$ generator matrix of the code as

$$\begin{pmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & & & & \cdots & 0 \\ 0 & 0 & \cdots & & g_0 & g_1 & \cdots & g_{n-k} \end{pmatrix}.$$

In this way, the row space generates C. If we can write $x^n - 1 = w_1(x)w_2(x)\cdots w_t(x)$ as the decomposition of $x^n - 1$ into irreducible factors, then the code generated by $\frac{x^n-1}{w_i(x)}$ is called an *irreducible* cyclic code. In algebraic terms what this means is that the code C is a *minimal ideal* of $\mathbf{F}_{q^k}[x]/(x^n - 1)$, i.e., C contains no subspace (other than 0) which is closed under the cyclic shift operator [34]. The reason we are interested in irreducible cyclic codes is that there is an established connection between the weights of the code words and Gauss sums.

For convenience, we also introduce the check polynomial and parity check matrix.

Definition 16. *The polynomial* $h(x) = \frac{x^n-1}{g(x)}$ *in an* $[n, k]$ *cyclic code is called the check polynomial.*

It has earned this name due to the following fact. If a word $(v_0, v_1, \ldots, v_{n-1}) \in C$ then

$$(v_0 + v_1 x + \cdots + v_{n-1}x^n - 1)h(x) = 0 \mod x^n - 1.$$

This follows from the observation that every word in C is equal to a polynomial $p(x)$ multiplied by the generator polynomial $g(x)$ and thus we have that

$$(v_0 + v_1 x + \cdots + v_{n-1}x^n - 1)h(x) = p(x)g(x)h(x) = p(x)(x^n - 1) = 0 \mod x^n - 1.$$

Definition 17. *The parity check matrix* H *of a code* C *is the generator matrix for the code dual to* C. *If* $c \in C$ *then* $Hc = 0$.

We now turn to the representation of irreducible cyclic codes, specifically 1) the form that the generator matrix can take, 2) a description of the codewords in terms of the trace function. Issue 1) relates back to the matrix representation of the cycle matroid of graphs and issue 2) will allow us to make the connection to Gauss sums.

8.2. Generator matrix of a cyclic code and the cycle matroid matrix. There is an alternative (but equivalent) way of constructing the generator matrix of a cyclic code which will immediately show its usefulness in its relationship with the cycle matroid matrix representation. Let C be an $[n, k]$ cyclic code and let $g(x)$ be the generator polynomial. Now, divide x^{n-k+i} by $g(x)$ for $0 \leq i \leq k-1$. We have

$$x^{n-k+i} = q_i(x)g(x) + r_i(x),$$

where $\deg r_i(x) < \deg g(x) = n - k$ or $r_i(x) = 0$. What this means is that we have a set of linearly independent code words. Namely, we have the k code words given by

$$x^{n-k+i} - r_i(x) = q_i(x)g(x)$$

in C. More explicitly, take the remainder polynomials $r_i(x)$ after applying the division algorithm and using the correspondence (39) above, form the $k \times (n-k)$ matrix R and append the $k \times k$ identity matrix to it. The rows of R are the coefficients of the $r_i(x)$ and one then has the $k \times n$ generator matrix $[I_k|R]$. This is precisely the form of the matrix representation for matroids discussed above. Thus, we have a correspondence between the generator matrix for an irreducible cyclic code and the matrix representation for the cycle matroid of a graph.

Proposition 1. *In an* $[n, k]$ *irreducible cyclic code there are at most* N *words of different non-zero weight where* $N = (q^k - 1)/n$.

Proof. For any irreducible cyclic code we have the relation $q^k - 1 = nN$ over the field $\mathbf{F}_q$. The length of each word is n and any cyclic permutation of a word preserves the Hamming weight. Therefore, for each word there are $n-1$ other words of equal weight. As there are $q^k - 1$ words of non-zero weight, if we assume that every word that does not arise from the cyclic permutation of another word is of a different weight, then there are $(q^k - 1)/n$ words of different weight. Being however that there is the possibility of repeats in weight among words which are not cyclic permutations of each other, there are at most N different weights. □

9. Gauss Sums and a Quantum Algorithm for the Estimation of Gauss Sums

Gauss sums are sums of products of group characters.

9.1. Characters. A character of a finite group $(G, *)$ is a homomorphism Φ from G to the group of the non-zero complex numbers $\mathbf{C}$.

We are interested in two types of characters, namely the multiplicative and additive characters. Let $\mathbf{F} \equiv \mathbf{F}_{q^k}$ (where k is a positive integer) be a finite field as defined previously, and let $\mathbf{F}^*$ be the multiplicative group of $\mathbf{F}$. Let g be a primitive element of $\mathbf{F}$ (i.e., g generates $\mathbf{F}$). Let

$$\zeta_q = e^{2\pi i/q}$$

denote the q^{th} root of unity. Let $x = g^k \in \mathbf{F}^*$. A multiplicative character $\chi_j(x)$ is a mapping from the set of powers $\{m\}$ in $x = g^m$ to powers of roots of unity. Specifically, the group of multiplicative characters $\chi = \{\chi_j\}_j$ consists of the elements

$$\chi_j(x) = \chi_j(g^m) = \zeta_{q^k-1}^{jm}, \quad m = 0, \ldots, q-2 \in \mathbf{F}_q; \quad j = 0, \ldots, q^k - 2 \in \mathbf{F}.$$

Let $a \in \mathbf{F}$. An additive character $e_j(a)$ is a mapping from $\mathbf{F}$ to powers of roots of unity via the trace function. Specifically, the group of additive characters $e = \{e_\beta\}_\beta$ consists of the elements

$$e_\beta(a) = \zeta_q^{\mathrm{Tr}(\beta a)} \quad \forall a, \beta = 0, \ldots, q^k - 1 \in \mathbf{F},$$

where the trace is defined in Eq. (17).

9.2. Discrete log. For every non-zero $x \in \mathbf{F}^*$ the discrete logarithm with respect to a primitive element $g \in \mathbf{F}$ is given by

$$\log_g(x) = \log_g(g^m) = m \bmod (q^k - 1).$$

This means that every multiplicative character can be written

$$\chi_j(x) = \chi_j(g^m) = \zeta_{q^k-1}^{j \log_g(x)} \tag{40}$$

for $x \neq 0$ and $\chi(0) = 0$.

9.3. Gauss sums. Let e_β and χ_j be an additive and multiplicative character respectively. Then the Gauss sum $G(\chi_j, e_\beta)$ is defined as:

$$G(\chi_j, e_\beta) = \sum_{x \in \mathbf{F}^*} \chi_j(x) e_\beta(x). \tag{41}$$

Gauss sums are used extensively in number theory, e.g., in the study of quadratic residues or Dirichlet L-functions.

To compute a Gauss sum we need to specify the field $\mathbf{F}$ and the indices $\beta \in \mathbf{F}$ and $j \in \mathbf{F}$ of the additive and multiplicative characters respectively. Thus the input size to a Gauss sum computation is $O(k \log q)$ bits. We can now define the Gauss sum over $\mathbf{F}$ as

$$G_{\mathbf{F}}(\chi_j, \beta) = \sum_{x \in \mathbf{F}^*} \chi_j(x) \zeta_q^{\mathrm{Tr}(\beta x)}.$$

It is well known that if $\chi_j \neq 1$ then [33]:

$$G_{\mathbf{F}}(\chi_j, \beta) = \sqrt{q^r} e^{i\gamma}, \tag{42}$$

where $\gamma = \gamma_{\mathbf{F}}(\chi_j, \beta)$. This means that all we need to do is approximate the angle $\gamma \bmod (2\pi)$ in order to approximate the Gauss sum. This is precisely the Gauss sum approximation problem for finite fields.

9.4. Quantum algorithm for Gauss sums. Van Dam and Seroussi devised an efficient quantum algorithm to estimate Gauss sums [26]. The following is an outline of the essentials of the proof; we refer the reader to [26] for a complete description as well as a discussion of the complexity of estimating Gauss sums.

Theorem 9 {***Quantum Amplitude Amplification***}. *Let $f : S \mapsto \{0, 1\}$ be a function for which we know the total weight $\|f\|_{l_1}$ but not those values $x \in S$ for which $f(x) = 1$. Then the corresponding state*

$$|f\rangle = \frac{1}{\|f\|_{l_2}} \sum_{x \in S} f(x)|x\rangle$$

can be efficiently and exactly prepared on a quantum computer where we have to make a number of queries to f of the order $O\left(\sqrt{\frac{|S|}{\|f\|_{l_1}}}\right)$.

This is an essential ingredient in Grover's quantum search algorithm. For a proof and details see [2]. It follows from Eq. (40) and Shor's discrete log algorithm [1] that given g, q^k and j, we can efficiently create the state $|\chi_j\rangle$. The following lemma is essential in this regard. First note that for any set S we define

$$|S\rangle \equiv \frac{1}{\sqrt{|S|}} \sum_{x \in S} |x\rangle.$$

Lemma 2. *For a finite field $\mathbf{F}_{q^k}$ and the triplet (q^k, g, r) (the specification of a multiplicative character χ_r), the state*

$$|\chi_r\rangle = \frac{1}{\sqrt{q^k - 1}} \sum_{x \in \mathbf{F}_{q^k}} \chi_r(x)|x\rangle$$

and its Fourier transform $|\hat{\chi}_r\rangle$ can be created in $\mathrm{polylog}(q^k)$ *time steps on a quantum computer.*

Proof. We first create the state

$$|\mathbf{F}_{q^k}^*\rangle|\hat{1}\rangle = \frac{1}{\sqrt{q^k(q^k-1)}} \sum_{x\in\mathbf{F}_{q^k}^*} |x\rangle \sum_{j=0}^{q^k-2} \zeta_{q^k-1}^{j} |j\rangle$$

by using Grover's amplitude amplification on $\mathbf{F}_{q^k}$ and the Fourier transform. Next, in superposition over all $x \in \mathbf{F}_{q^k}^*$, we calculate $\log_g(x)$ and subtract $r \log_g(x)$.

$$|\mathbf{F}_{q^k}^*\rangle|\hat{1}\rangle \longrightarrow \frac{1}{\sqrt{q^k(q^k-1)}} \sum_{x\in\mathbf{F}_{q^k}^*} |x\rangle \sum_{j=0}^{q^k-2} \zeta_{q^k-1}^{j} |j - r\log_g(x)\rangle \tag{43}$$

$$= \frac{1}{\sqrt{q^k(q^k-1)}} \sum_{x\in\mathbf{F}_{q^k}^*} |x\rangle \sum_{j=0}^{q^k-2} \zeta_{q^k-1}^{j} \zeta_{q^k-1}^{r\log_g(x)} |k\rangle \tag{44}$$

$$= \frac{1}{\sqrt{q^k-1}} \sum_{x\in\mathbf{F}_{q^k}^*} \zeta_{q^k-1}^{r\log_g(x)} |x\rangle|\hat{1}\rangle \tag{45}$$

$$= |\chi_r\rangle|\hat{1}\rangle. \tag{46}$$

To get $|\hat{\chi_r}\rangle$ we just need to apply the Fourier transform. □

The technique used in the above proof is known as the *phase kickback trick*. Now we are ready for the following.

Theorem 10. ***Algorithm for approximating Gauss sums.*** *Consider $\mathbf{F}_{q^k}$, a nontrivial multiplicative character χ_r and $\beta \in \mathbf{F}_{q^k}^*$. If we apply the quantum Fourier transform over this field to $|\chi_r\rangle$, followed by a phase change*

$$|y\rangle \longrightarrow \chi_r^2(y)|y\rangle, \tag{47}$$

then we generate an overall phase change given by

$$|\chi_r\rangle = \frac{1}{\sqrt{q^k-1}} \sum_{x\in\mathbf{F}_{q^k}} \chi_r(x)|x\rangle \longrightarrow \frac{G_{\mathbf{F}_{q^k}}(\chi_r,\beta)}{\sqrt{q^k}}|\chi_r\rangle.$$

Proof. After a Fourier transform we have

$$|\hat{\chi_r}\rangle = \frac{1}{\sqrt{q^k(q^k-1)}} \sum_{y\in\mathbf{F}_{q^k}^*} \left(\sum_{x\in\mathbf{F}_{q^k}} \chi_r(x)\zeta_q^{\mathrm{Tr}(\beta xy)} \right) |y\rangle$$

$$= \frac{1}{\sqrt{q^k(q^k-1)}} \sum_{y\in\mathbf{F}_{q^k}^*} G_{\mathbf{F}_{q^k}}(\chi_r,\beta y)|y\rangle$$

$$= \frac{1}{\sqrt{q^k(q^k-1)}} \sum_{y\in\mathbf{F}_{q^k}^*} \chi_r(y^{-1}) G_{\mathbf{F}_{q^k}}(\chi_r,\beta)|y\rangle.$$

Then

$$|\chi_r\rangle = \frac{G_{\mathbf{F}_{q^k}}(\chi_r, \beta)}{\sqrt{q^k(q^k-1)}} \sum_{y \in \mathbf{F}^*_{q^k}} \chi_r(y^{-1})|y\rangle.$$

Now we know that we can efficiently (and exactly) create the phase change given by (47). Doing so gives us

$$|\hat{\chi}\rangle \longrightarrow \frac{G_{\mathbf{F}_{q^k}}(\chi_r, \beta)}{\sqrt{q^k(q^k-1)}} \sum_{y \in \mathbf{F}^*_{q^k}} \chi_r(y^{-1})\chi_r^2(y)|y\rangle = \frac{G_{\mathbf{F}_{q^k}}(\chi_r, \beta)}{\sqrt{q^k}}|\chi_r\rangle,$$

since $|\chi_r\rangle = \frac{1}{\sqrt{q^k-1}} \sum_{y \in \mathbf{F}^*_{q^k}} \chi_r(y)|y\rangle$ and $\chi_r(y^{-1})\chi_r(y) = 1$. Thus, the coefficient of $|\chi_r\rangle$ is just $e^{i\gamma}$. It is well known that one can efficiently estimate the phase of such a function to within an expected error of $O(1/n)$, where n is the number of copies of $e^{i\gamma}|\chi_r\rangle$ we sample. Therefore we arrive at an estimate of γ and hence of the Gauss sum in question. □

This gives way to the following theorem about the time complexity of the algorithm and is the culmination of the first part of the paper [26].

Theorem 11. *For any $\epsilon > 0$, there is a quantum algorithm that estimates the phase γ in $G_{\mathbf{F}_{q^k}}(\chi_r, \beta) = \sqrt{q^k}e^{i\gamma}$, with expected error $E(|\gamma - \tilde{\gamma}|) < \epsilon$. The time complexity of this algorithm is bounded by $O(\frac{1}{\epsilon} \cdot \text{polylog}(q^k))$* [26].

Note that the "poly" in polylog refers to a quadratic polynomial.

References

1. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. on Comp. **26**, 1484 (1997)
2. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, New York: ACM, 1996, p. 212
3. Lidar, D.A., Biham, O.: Simulating Ising spin glasses on a quantum computer. Phys. Rev. E **56**, 3661 (1997)
4. Swendsen, R.H., Wang, J.-S.: Phys. Rev. Lett. **58**, 86 (1987)
5. Lidar, D.A.: On the Quantum Computational Complexity of the Ising Spin Glass Partition Function and of Knot Invariants. New J. Phys. **6**, 167 (2004)
6. Welsh, D.J.A.: *Complexity: Knots, Colourings and Counting*. Volume **1** of London Mathematical Society Lecture Note Series **186**. London: Cambridge University Press, 1993
7. Reichl, L.E.: *A Modern Course in Statistical Physics*. New York: John Wiley & Sons, 1998
8. Jones, V.F.R.: On Knot Invariants Related to Some Statistical Mechanical Models. Pacific J. Math. **137**, 311 (1989)
9. Jones, V.F.R.: A Polynomial Invariant for Knots via von Neumann Algebras. Bull. Amer. Math. Soc. **12**, 103 (1985)
10. Jaeger, F., Vertigen, D., Welsh, D.: On the Computational Complexity of the Jones' and Tutte polynomials. Math. Proc. Cambridge Philos. Soc. **108**, 35 (1990)
11. Kauffman, L.H.: *Knots and Physics*. Volume **1** of *Knots and Everything*. Singapore: World Scientific, 2001
12. Alon, N., Frieze, A.M., Welsh, D.: *Polynomial Time Randomised Approximation Schemes for Tutte-Gröthendieck Invariants: The Dense Case*. Electronic Colloquium on Computational Complexity, **1**(5) (1994), available at http://eccc.hpi-web.de/eccc-reports/1994/TR94-005/index.html, 1994
13. Nechaev, S.: *Statistics of knots and entangled random walks*. http://arxiv.org/list/cond-mat/9812205, 1998

14. Witten, E.: Topological quantum field theory. Commun. Math. Phys. **117**, 353 (1988)
15. Freedman, M.H., Kitaev, A., Wang, Z.: Simulation of topological field theories by quantum computers. Commun. Math. Phys. **227**, 587 (2002)
16. Freedman, M.H., Kitaev, A., Larsen, M.J., Wang, Z.: *Topological Quantum Computation*. http://arxiv.org/list/quant-ph/0101025, 2001
17. Aharonov, D., Jones, V., Landau, Z.: *A Polynomial Quantum Algorithm for Approximating the Jones Polynomial*. http://arxiv.org/list/quant-ph/0511096, 2005
18. Wocjan, P., Yard, J.: *The Jones polynomial: quantum algorithms and applications in quantum complexity theory*. http://arxiv.org/list/quant-ph/0603069, 2006
19. Kauffman, L.H., Lomonaco, S.J.: q-Deformed spin networks, knot polynomials and anyonic topological computation. J. of Knot Theory and Its Ramifications **16**, 267 (2007)
20. Bonca, J., Gubernatis, J.E.: *Real-Time Dynamics from Imaginary-Time Quantum Monte Carlo Simulations: Test on Oscillator Chains*. http://arxiv.org/list/cond-mat/9510098, 1995
21. Aharonov, D., Arad, I., Eban, E., Landau, Z.: *Polynomial Quantum Algorithms for Additive approximations of the Potts model and other Points of the Tutte Plane*. http://arxiv.org/list/quant-ph/0702008, 2007
22. Van den Nest, M., Dur, W., Briegel, H.J.: Classical spin models and the quantum stabilizer formalism. Phys. Rev. Lett. **98**, 117207 (2007)
23. Hartmann, A.K.: Calculation of partition functions by measuring component distributions. Phys. Rev. Lett. **94**, 050601 (2005)
24. Denef, J., Vercauteren, F.: *Counting Points on C_{ab} Curves using Monsky-Washnitzer Cohomology*. http://citeseer.ist.psu.edu/denef04counting.html, 2004
25. Kedlaya, K.: Quantum Computation of zeta functions of curves. Comput. Complex. **15**, 1–19 (2006)
26. van Dam, W., Seroussi, G.: *Efficient Quantum Algorithms for Estimating Gauss Sums*. http://arxiv.org/list/quant-ph/0207131, 2002
27. Barg, A.: On some polynomials related to Weight Enumerators of Linear Codes. SIAM J. Discrete Math. **15**, 155 (2002)
28. Baumert, L., McEliece, R.: Weights of Irreducible Cyclic Codes. Inform. and Control **20**, 158 (1972)
29. Welsh, D.J.A.: *Matroid Theory*. London: Academic Press Inc, 1976
30. Gross, J., Yellen, J.: *Graph theory and its applications*. Discrete mathematics and its applications. Boca Raton, FL: CRC Press, 1999
31. van Lint, J.H.: *Introduction to Coding Theory*. Berlin-Heidelberg-New York: Springer-Verlag, 1982
32. Jaeger, F.: The Tutte Polynomial and Link Polynomials. Proc. Amer. Math. Soc. **103**, 647 (1998)
33. Evans, J., Berndt, B.C., Williams, K.S.: *Gauss and Jacobi Sums*. New York: Wiley-Interscience, 1998
34. Moisio, M.: *Exponential Sums, Gauss Sums and Cyclic Codes*. 1997. Available at http://lipas.uwasa.fi/~mamo/vaitos.pdf, 1997
35. Aubry, Y., Langevin, P.: *On the weights of irreducible cyclic codes*. 2005. Available at http://iml.univ-mrs.fr/~aubry/LNCS.pdf, 2005
36. Andrews, G.E.: *Number Theory*. New York: Dover Publications Inc., 1994
37. Nielsen, M.A., Chuang, I.L.: *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press, 2000
38. van Dam, W., Seroussi, G.: *Quantum algorithms for estimating Gauss sums and calculating discrete logarithms*. Available at http://www.cs.ucsb.edu/~vandam/gausssumdlog.pdf
39. Brassard, G., Hoyer, P., Tapp, A.: *Quantum Counting*. http://arxiv.org/list/quant-ph/9805082, 1998
40. van Bussel, F., Geraci, J.: *A Note on Cyclotomic Cosets and an Algorithm for finding Coset Representatives and Size and a theorem on the quantum evaluation of weight enumerators for a certain class of cycliccades*. http://arxiv.org/list/cs.0703129, 2007
41. Lidl, R., Niederreiter, H.: *Finite Fields*, Volume **20** of Encyclopedia of Mathematics. Cambridge: Cambridge University Press, 1997
42. Van Der Glugt, M.: Hasse-Davenport Curves, Gauss Sums and Weight Distributions of Irreducible Cyclic Codes. J. Number Theory **55**, 145 (1995)
43. Vyalyi, M.N.: *Hardness of approximating the weight enumerator of a binary linear code*. http://arxiv.org/list/cs.CC/0304044, 2003
44. Khovanov, M.: A categorification of the Jones polynomial. Duke Math. J. **101**, 359 (2000)
45. Martinez-Perez, C., Willems, W.: Is the Class of Cyclic Codes Asymptotically Good?. IEEE Trans. Inf. Theory **52**(2), 696 (2006)
46. Shrock, R.: Exact Potts model partition functions on ladder graphs. Physica A **283**, 388 (2000)

Communicated by M.B. Ruskai