Efficient Quantum Walk Circuits for Metropolis-Hastings Algorithm

Jessica Lemieux,1 Bettina Heim,2 David Poulin,1,3 Krysta Svore,2 and Matthias Troyer2

1Département de Physique & Institut Quantique, Université de Sherbrooke, Québec, Canada
2Quantum Architecture and Computation Group, Microsoft Research, Redmond, WA 98052, USA
3Canadian Institute for Advanced Research, Toronto, Ontario, Canada M5G 1Z8

(Dated: October 7, 2019)

We present a detailed circuit implementation of Szegedy’s quantization of the Metropolis-Hastings walk. This quantum walk is usually defined with respect to an oracle. We find that a direct implementation of this oracle requires costly arithmetic operations and thus reformulate the quantum walk in a way that circumvents the implementation of that specific oracle and which closely follows the classical Metropolis-Hastings walk. We also present heuristic quantum algorithms that use the quantum walk in the context of discrete optimization problems and numerically study their performances. Our numerical results indicate polynomial quantum speedups in heuristic settings.

I. INTRODUCTION

Markov chain Monte Carlo (MCMC) methods are a cornerstone of modern computation, with applications ranging from computational science to machine learning. The key idea is to sample a distribution \( \pi_x \) by constructing a random walk \( W \) which reaches this distribution at equilibrium \( W \pi = \pi \). One important characteristic of a Markov chain is its mixing time, the time it requires to reach equilibrium. This mixing time is governed by the inverse spectral gap of \( W \), where the spectral gap \( \Delta \) is defined as the difference between the two largest eigenvalues. The runtime of a MCMC algorithm is thus determined by the product of the mixing time and the time required to implement a single step of the walk.

Szegedy [1] presented a general method to quantize reversible walks, resulting in a unitary transformation \( U_W \). The eigenvalues \( e^{i\theta} \) of a unitary matrix all lie on the unit complex circle, and we choose \( \theta_0 \leq \theta_1 \leq \theta_2 \leq \ldots \). The steady state \( |\pi\rangle \) of the quantum walk has \( \theta_0 = 0 \) and is essentially a coherent version \( |\pi\rangle = \sum_x \sqrt{\pi_x} |x\rangle \) of the classical equilibrium distribution \( \pi \). The main feature of the quantum walk is that its spectral gap \( \delta := \theta_1 \geq \sqrt{\Delta} \) is quadratically larger than the one of the corresponding classical walk. Combined with the quantum adiabatic algorithm [2][4], this yields a quantum algorithm to reach the steady state that scales quadratically faster with \( \Delta \) than the classical MCMC algorithm [5].

While at first glance this is an important advantage with far reaching applications, additional considerations must be taken into account to determine if quantum walks offer a significant speedup for any specific application. One of the reasons is that the time required to implement a single step \( U_W \) of the quantum walk can be significantly larger than the time required to implement a single step \( W \) of the classical walk. Thus, quantum walks are more likely to offer advantages in situations with extremely long equilibration times. Moreover, classical walks are often used heuristically out of equilibrium. When training a neural network for instance, where a MCMC method called stochastic gradient is used to minimize a cost function, it is in practice often not necessary to reach the true minimum, and thus the MCMC runs in time less than its mixing time. Similarly, simulated annealing is typically used heuristically with cooling schedules far faster than prescribed by provable bounds – and combined with repeated restarts. Such heuristic applications further motivate the constructions of efficient implementations of \( U_W \), and the development of heuristic methods for quantum computers.

This article addresses these two points. First, we present a detail realization and cost analysis of the quantum walk operator for the special case of a Metropolis-Hastings walk [6][7]. This is a widespread reversible walk, whose implementation only requires knowledge of the relative populations \( \pi_x / \pi_y \) of the equilibrium distribution. While Szegedy’s formulation of the quantum walk builds on a classical walk oracle, our implementation circumvents the direct implementation of the classical walk, which would require costly arithmetic operations. Instead, we directly construct a related but different quantum unitary walk operator with an effort to minimize circuit depth. Second, we suggest heuristic uses of this oracle inspired by the adiabatic algorithm, and study their performances numerically.

II. PRELIMINARIES

A. Quantum Walk

We define a classical walk on a \( d \)-dimensional state space \( X = \{ x \} \) by a \( d \times d \) transition matrix \( W \) where the transition probability \( x \rightarrow y \) is given by matrix element \( W_{yx} \). Thus, the walk maps the distribution \( p_y \) to the distribution \( p'_y = \sum_x W_{yx} p_x \), or in matrix form \( p' = W p \). An aperiodic walk is irreducible if every state in \( X \) is accessible from every other state in \( X \), which implies the existence of a unique equilibrium distribution \( \pi = W \pi \). Finally, a walk is reversible if it obeys the detailed balance...
condition
\[ W_{yx} \pi_x = W_{xy} \pi_y. \]  
(1)

We now explain how to quantize an reversible classical walk \( \mathcal{W} \).

Szegedy’s quantum walk is formulated in an oracle setting. For a classical walk \( \mathcal{W} \), it assumes a unitary transformation \( W \) acting on a Hilbert space \( \mathbb{C}^d \otimes \mathbb{C}^d \) with the following action
\[ W|x\rangle \otimes |0\rangle = |w_x\rangle \otimes |x\rangle =: |\phi_x\rangle, \]
where \( |w_x\rangle := \sum_y \sqrt{W_{yx}} |y\rangle \). Define \( \Pi_0 \) as the projector onto the subspace \( \mathcal{E}_0 \) spanned by states \( \{|x\rangle \otimes |0\rangle\}_{x=1} \). Combining \( W \) to the reflection \( R = 2\Pi_0 - I \) and the swap operator \( \Lambda \), we can construct the quantum walk defined by
\[ U_\mathcal{W} := RW^\dagger \Lambda W = (2\Pi_0 - 1)W^\dagger \Lambda W. \]  
(4)

Szegedy’s walk is defined as \( \Lambda W(RW^\dagger \Lambda W)RW^\dagger \), so it is essentially the square of the operator \( U_\mathcal{W} \) we have defined, but this will have no consequence on what follows aside from a minor simplification.

To analyze the quantum walk \( U_\mathcal{W} \), let us define the state \( |\psi_x\rangle := \Lambda |\phi_x\rangle = |x\rangle \otimes |w_x\rangle \) and consider the operator
\[ X := \Pi_0 W^\dagger \Lambda W \Pi_0 = \sum_{xy} \langle \phi_y | \psi_x \rangle |y\rangle \langle x| \otimes |0\rangle \langle 0| \]
\[ = \sum_{xy} \sqrt{W_{xy} W_{yx}} |y\rangle \langle x| \otimes |0\rangle \langle 0|. \]  
(7)

At this point, we must assume that the walk is reversible and make use of the detailed balance condition of Eq. (4) to obtain
\[ X = \sum_{xy} \sqrt{\frac{\pi_x}{\pi_y}} W_{yx} |y\rangle \langle x| \otimes |0\rangle \langle 0|, \]
(8)

or, if we restrict the operator \( X \) to its support \( \mathcal{E}_0 \), we get in matrix notation \( X = \text{diag}(\pi^{-\frac{1}{2}}) \mathcal{W} \text{diag}(\pi^\frac{1}{2}) \). The matrices \( \mathcal{W} \) and \( X \) are thus similar so they have the same eigenvalues. Define its eigenvectors
\[ X |\tilde{\gamma}_k\rangle = \lambda_k |\tilde{\gamma}_k\rangle, \]
where \( \lambda_k \) are the eigenvalues of \( \mathcal{W} \). Because the operator \( X \) is obtained by projecting the operator \( W^\dagger \Lambda W \) onto the subspace \( \mathcal{E}_0 \), its eigenvectors with non-zero eigenvalues in the full Hilbert space clearly have the form \( |\gamma_k\rangle = |\tilde{\gamma}_k\rangle \otimes |0\rangle \).

If we consider the action of \( W^\dagger \Lambda W \) without those projections, we get
\[ W^\dagger \Lambda W |\gamma_k\rangle = \lambda_k |\gamma_k\rangle - \beta_k |\gamma_k^\perp\rangle \]
(10)

where \( |\gamma_k\rangle \) is orthogonal to the subspace \( \mathcal{E}_0 \), so in particular it is orthogonal to all the vectors \( |\gamma_{k'}\rangle \). Finally, because \( W^\dagger \Lambda W \) is a unitary, we also obtain that the \( |\gamma_k^\perp\rangle \) are orthogonal to each other and that \( \beta_k = \sqrt{1 - |\lambda_k|^2} \). This implies that the vectors \( \{|\gamma_k\rangle, |\gamma_k^\perp\rangle\} \) are all mutually orthogonal and that \( W^\dagger \Lambda W \) is block diagonal in that basis.

Given the above observations, it is straightforward to verify that
\[ U_\mathcal{W} |\gamma_k\rangle = \lambda_k |\gamma_k\rangle + \sqrt{1 - |\lambda_k|^2} |\gamma_k^\perp\rangle \]
\[ U_\mathcal{W} |\gamma_k^\perp\rangle = \sqrt{1 - |\lambda_k|^2} |\gamma_k\rangle - \lambda_k |\gamma_k^\perp\rangle , \]
(11)
(12)

so the eigenvalues of \( U_\mathcal{W} \) on the subspace spanned by \( \{|\gamma_k\rangle, |\gamma_k^\perp\rangle\} \) are \( e^{\pm i \theta_k} \) where \( \cos \theta_k = \lambda_k \) with corresponding eigenvectors \( |\gamma_k\rangle = \frac{1}{\sqrt{2}}(|\gamma_k\rangle \pm i |\gamma_k^\perp\rangle) \).

B. Adiabatic state preparation

We can use quantum phase estimation \cite{8} to measure the eigenvalue of \( U_\mathcal{W} \). In particular, we want this measurement to be sufficiently accurate to resolve the eigenvalue \( \theta = 0 \), or equivalently \( \lambda_k = 1 \), from the rest of the spectrum. Assuming that the initial state is supported on the subspace \( \mathcal{E}_0 \), the spectral gap of \( U_\mathcal{W} \) is \( \delta = \theta_1 = \arccos(\lambda_1) = \arccos(1 - \Delta) \sim \sqrt{\Delta} \), so we only need about \( 1/\sqrt{\Delta} \) applications of \( U_\mathcal{W} \) to realize that measurement. This is quadratically faster than the classical mixing-time \( 1/\Delta \), which is at the heart of the quadratic quantum speed-up.

A measurement outcome corresponding to \( \theta = 0 \) would produce the coherent stationary distribution \( |\pi\rangle \otimes |0\rangle := \sum_x \sqrt{\pi_x} |x\rangle \otimes |0\rangle \). Indeed, first note that for any \( |\psi\rangle \) such that \( \langle \pi | \psi \rangle = \langle \psi | \pi \rangle = |\langle \psi \rangle \| = 1 \), Eq. (10) implies that \( U_\mathcal{W} (|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |0\rangle \). We can verify that this condition holds for \( |\psi\rangle = |\pi\rangle \):
\[ X \sum_x \sqrt{\pi_x} |x\rangle \otimes |0\rangle = \sum_x \sqrt{\pi_x} W_{yx} \sqrt{\pi_y} |y\rangle \otimes |0\rangle \]
\[ = \sum_x W_{xy} \sqrt{\pi_y} |y\rangle \otimes |0\rangle \]
\[ = \sum_y \sqrt{\pi_y} |y\rangle \otimes |0\rangle \]
(13)
(14)
(15)

where we have used detailed balance Eq. (4) in the second step and \( \sum_y W_{xy} = 1 \) in the last step.

From an initial state \( |\psi\rangle \otimes |0\rangle = \sum_y \alpha_y |\gamma_k\rangle \), the probability of that measurement outcome is \( |\langle \psi \pi \rangle|^2 = |\alpha_0|^2 \), so the initial state \( |\psi\rangle \) must be chosen with a large overlap with the fixed point to ensure that this measurement outcome has a non-negligible chance of success. If no such state can be efficiently prepared, one can use adiabatic state preparation \cite{2, 3} to increase the success probability. In its discrete formulation \cite{3} inspired by the quantum Zeno effect, we can choose a sequence of random
walks \( W^0, W^1, \ldots W^L = W \) with coherent stationary distributions \( |\pi^0\rangle \). The walks are chosen such that \( |\pi^0\rangle \) is easy to prepare and consecutive walks are nearly identical, so that \( \langle (\pi^j|\pi^{j+1})^2 \rangle \geq 1 - \frac{1}{L} \). Thus, the sequence of \( L \) measurements of the eigenstate of the corresponding quantum walk operators \( U_{W^j} \) all yield the outcomes \( \theta = 0 \) with probability \( (1 - \frac{1}{L})^L \sim \frac{1}{e^L} \), which results in the desired state. The overall complexity of this algorithm is

\[
C \sum_{j=1}^{L} \frac{1}{\delta_j}
\] (16)

where \( \delta_j \) is the spectral gap of the \( j \)-th quantized walk \( W^j \) and \( C \) is the time required to implement a single quantum walk operator.

C. Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm [6, 7] is a special class of Markov chains which obey detailed balance Eq. (1) by construction. The basic idea is to break the transition probability \( T_{yx} \). This transition is accepted with probability \( A_{yx} \), and if the transition is rejected the state remains \( x \).

The overall transition probability is thus

\[
W_{yx} = \begin{cases} T_{yx}A_{yx} & \text{if } y \neq x \\ 1 - \sum_y T_{yx}A_{yx} & \text{if } y = x. \end{cases}
\] (17)

The detailed balance condition Eq. (1) becomes

\[
R_{xy} := \frac{A_{yx}}{A_{xy}} = \frac{\pi_y T_{xy}}{\pi_x T_{yx}},
\] (18)

which in the Metropolis-Hastings algorithm is solved with the choice

\[
A_{yx} = \min \left(1, R_{xy}\right).
\] (19)

We note that our quantum algorithm can also be applied to the Glauber, or heat-bath choice

\[
A_{yx} = \frac{1}{1 + R_{yx}}.
\] (20)

The Metropolis-Hastings algorithm is widely used to generate a Boltzmann distribution as used in statistical physics and machine learning. Given a real energy function \( E(x) \) on the configuration space \( X \), the Boltzmann distribution at inverse temperature \( \beta \) is defined as \( \pi_x = \frac{1}{Z(\beta)} e^{-\beta E(x)} \) where the partition function \( Z(\beta) \) ensures normalization. In this setting, it is common practice to choose a symmetrical proposed transition \( T_{yx} = T_{xy} \), so the acceptance probability depends only on the energy difference

\[
A_{yx} = \min \left(1, e^{\beta |E(x) - E(y)|}\right).
\] (21)

III. CIRCUIT FOR WALK OPERATOR

Quantum algorithms built from quantization of classical walks [1, 5, 9, 10] usually assume an oracle formulation of the walk operator, where the ability to implement the transformation \( W \) of Eq. (2) is taken for granted. As we discuss below in Appendix A this transformation requires costly arithmetic operations. One of the key innovations of this article is to provide a detailed and simplified implementation of walk operator along with a detailed cost analysis for Metropolis-Hastings walks. As it will become apparent, our implementation circumvents the use of \( W \) altogether.

For concreteness, we will assume a \((k, d)\)-local Ising model, where \( X = \{+1, -1\}^n \), and the energy function takes some simple form

\[
E(x) = \sum_{\ell} J_\ell \prod_{s \in \Omega_\ell} x_s
\] (22)

where \( \Omega_\ell \) are subsets of at most \( k \) Ising spins and the \( J_\ell \) are real coupling constants, and each spin interacts with at most \( d \) other spins.

As it is always the case for Ising models, we will assume that the proposed transition of the Metropolis-Hastings walk are obtained by choosing a random set of spins and inverting their signs. In other words, \( T_{yx} = f(x \cdot y) \) where the product is taken bit by bit and where \( f(z) \) is some simple probability distribution on \( X - \{1^n\} \) (it does not contain a trivial move), so \( T_{yx} \) is clearly symmetrical. The distribution \( f(z) \) is sparse, in the sense that it has only \( N \ll 2^n \) non-zero entries.

For concreteness, we will suppose that \( f \) is uniform over some set \( \mathcal{M} \) of moves, with \( |\mathcal{M}| = N \):

\[
T_{xy} = \begin{cases} \frac{1}{N} & \text{if } z = x \cdot y \in \mathcal{M} \\ 0 & \text{else} \end{cases}
\] (23)

The most common example consists of single-spin moves, where a single spin is chosen uniformly at random to be flipped. More generally, we will suppose that moves are sparse in the sense that each move \( z_j \in \mathcal{M} \) flips a constant-bounded number of spins and that each spin belongs to a constant-bounded number of different moves \( z_j \in \mathcal{M} \). For \( j = 1, 2, \ldots, N \), we use \( f(j) \) as a shorthand for \( f(z_j) \) where the \( z_j \) are the \( N \) elements of \( \mathcal{M} \). With a further abuse of notation, we will view \( z_j \in \mathcal{M} \) both as Ising spin configurations and as subsets of \([n]\), where the correspondence is given by the locations of \(-1\) spins in \( z_j \).

A direct implementation of the oracle \( W \) generally requires costly quantum circuits using arithmetic operations. The complexity arises from the need to uncompute a move register and a Boltzmann coin when implementing \( W \). This turns out to be non-trivial and costly if a move is rejected. We thus do not implement \( W \), but instead present a circuit which is isometric to the entire walk operator \( U_W \), thus avoiding the problem. In other
words, we construct a circuit for \( \tilde{U}_W := Y^\dagger U_W Y \) where \( Y \) maps
\[
Y: |x\rangle \otimes |y\rangle \rightarrow \begin{cases} |x\rangle \otimes |x \cdot y\rangle & \text{if } x \cdot y \in \mathcal{M} \\ 0 & \text{else.} \end{cases} \tag{24}
\]
To minimize circuit depth, the second register above is encoded in a unary representation, so it contains \( N \) qubits and \( |z\rangle \) is encoded as \( |00\ldots 0100\ldots 0\rangle \) with a 1 at the \( z \)-th position. In addition to these two registers, the circuit acts on an additional coin qubit. Thus, we will denote the System, Move, and Coin registers with corresponding subscripts \( |x\rangle_{S}|j\rangle_{M}|b\rangle_{C} \), and they contain \( n \), \( N \), and 1 qubits respectively.

Our implementation of the walk operator combines 4 components:
\[
\tilde{U}_W = RV^\dagger B^\dagger F BV \tag{25}
\]
where
\[
V: |0\rangle_M \rightarrow \sum_j \sqrt{f(j)} |j\rangle_M, \tag{26}
\]
\[
B: |x\rangle_S|j\rangle_M|0\rangle_C \rightarrow |x\rangle_S|j\rangle_M \left( \sqrt{1-A_{x,z_j}} |0\rangle + \sqrt{A_{x,z_j}} |1\rangle \right)_C, \tag{27}
\]
\[
F: |x\rangle_S|j\rangle_M|b\rangle_C \rightarrow |x\cdot z_j^b\rangle_S|j\rangle_M|b\rangle_C, \tag{28}
\]
\[
R: |0\rangle_M|0\rangle_C \rightarrow -|0\rangle_M|0\rangle_C, |j\rangle_M|b\rangle \rightarrow |j\rangle_M|b\rangle \text{ for } (j,b) \neq (0,0) \tag{29}
\]

While these definitions differ slightly from the ones of Sec. [Ref.1A] it can be verified straightforwardly, similar to our discussion in Sec. [Ref.1A] that these realize the desired walk operator. In what follows, we provide a complete description of each of these components, and their complexity is summarized in Table [Ref.1].

| Gate | 3L depth | 3L count | Total depth | Qubits |
|------|----------|---------|-------------|--------|
| V    | \( \log_2 N + 1 \) | \( 2N \) | \( \log_2 N + 1 \) | \( 2N \) |
| F    | 1        | \( N \)  | \( \log_2 N + 1 \) | \( 2N + n \) |
| R    | 2 log \( N \) | 4N      | 2 log \( N \) | \( 2N \) |
| B    | \( O(2^d \log \frac{1}{\epsilon}) \) | \( O(Nd^4 \log \frac{1}{\epsilon}) \) | \( O(\log(Nd^4 \log \frac{1}{\epsilon})) \) | \( 2N + n + 2 \) |

TABLE I. Upper bound on the complexity of each components of the walk operator. The cost is measured in terms of number of gates in the 3rd level of the Clifford hierarchy, which is equivalent to \( T \) count up to a small multiplicative factor. These are evaluated for a \((k, d)\)-local Ising model with moves consisting of single-spin flips, in which case \( N = n \). These costs could otherwise increase by a constant amount determined by \( k \), \( d \) and the sparsity of the moves \( z \in \mathcal{M} \).

to realize the transformation Eq. [26]. Here, we focus on the case of a uniform distribution. To begin, suppose that \( N \) is a power of 2. Starting in the state \( |00\ldots 01\rangle_M \), the state \( \frac{1}{\sqrt{N}} \sum_j |j\rangle_M \) (in unary) is obtained by applying a sequence of \( N \) gates \( \sqrt{SWAP} \) in a binary-tree fashion. Recall indeed that \( \sqrt{SWAP} |0\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \). The gate \( \sqrt{SWAP} \) is in the third level of the Clifford hierarchy, so it can be implemented exactly using a constant number of \( T \) gates. This represents a substantial savings compared to the method of [11] for general distributions which requires arbitrary rotations obtained from costly gate synthesis.

To avoid such costly rotations, even when \( N \) is not a power of 2, we choose to pad the distribution with additional states and prepare a distribution \( \frac{1}{N} \sum_j |j\rangle_M \) where \( j = [\log_2 N] \). The states \( j = 1, 2, \ldots N \) encode the \( N \) moves \( \mathcal{M} \) of the classical walk \( x \rightarrow y = x \cdot z_j \) while the additional states \( j > N \) correspond to trivial moves \( x \rightarrow x \). This padding has the effect of slowing down the classical walk by a factor \( 2^\ell /N < 2 \), and hence the quantum walk by a factor less than \( \sqrt{2} \), which is less than the additional cost of preparing a uniform distribution over a range which is not a power of 2.

B. Spin flip \( F \)

The operator \( F \) of Eq. [25] flips a set of system spins \( z_j \) conditioned on the coin qubit and on the \( j \)-th qubit of the move register being in state 1. This can be implemented with at most \( Nc \) Toffoli gates (controlled-controlled-NOT), where the constant \( c \) upper-bounds the number of spins that are flipped by a single move of \( \mathcal{M} \).

The coin register acts as one control for each gate, the \( j \)-th bit of the move register acts as the other control, and the targets are the system register qubits that are in \( z_j \), for \( j = 1, 2, \ldots N \). No gate is applied to the padding qubits \( j > N \).

This implementation has the disadvantage of being purely sequential. An alternative implementation uses \( O(N) \) additional scratchpad qubits but is entirely parallel. The details of the implementation depends on the sparsity of the moves \( \mathcal{M} \), and in general there is a tradeoff between the scratchpad size and the circuit depth. When the moves consist of single-spin flips for instance, this uses \( N \) CNOTs in a binary-tree fashion (depth \( \log_2 N \)) to make \( N \) copies of the coin qubit. The Toffoli gates can then be applied in parallel for each move, and lastly the CNOTs are undone.

C. Reflection \( R \)

The transformation \( R \) of Eq. [29] is a reflection about the state \( |00\ldots 00\rangle_M |0\rangle_C \). Using standard phase kickback methods, it can be implemented with a single additional qubit in state \( \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \) and a control\(^{N+1}\)-NOT gate.
The latter can be realized from $4(N - 1)$ serial Toffoli gates \[12\] and linear depth.

Since our goal is to minimize circuit depth, we use a different circuit layout that uses at most $N$ ancillary qubits and $4N$ Toffoli gates to realize the $(N + 1)$-fold controlled-not. The circuit once again proceeds in a binary tree fashion, dividing the set of $N + 1$ qubits into $(N + 1)/2$ pairs and applying a Toffoli gate between every pair with a fresh ancilla in state 0 as the target. The ancillary qubit associated to a given pair is in state 0 if and only if both qubits of the pair are in state 0. The procedure is repeated for the $(N + 1)/2$ ancillary qubits, until a single bit indicates if all qubits are in state 0. The ancillary bits are then uncomputed.

### D. Boltzmann coin B

The Boltzmann coin given in Eq. \[27\] is the most expensive component of the algorithm, simply because it is the only component which requires rotations by arbitrary angles. Specifically, conditioned on move qubit $j$ being 1 and the system register being in state $x$, the coin register undergoes a rotation by an angle

$$\theta_{x,j} = \arcsin \left( \sqrt{\min\{e^{-\beta \Delta_j(x)}, 1\}} \right)$$

for Metropolis-Hastings or

$$\theta_{x,j} = \arcsin \left( \frac{1}{1 + e^{\beta \Delta_j(x)}} \right)$$

for Glauber dynamics, where $\Delta_j = E(x \cdot z_j) - E(x)$. Given the sparsity constraints of the function $E$ and of the moves $z_j \in \mathcal{M}$, the quantity $\Delta_j$ can actually be evaluated from a subset of qubits of the system register, namely $\mathcal{N}_j = \{ k : k \in \Omega_i \& z_j \cap \Omega_i \neq \emptyset, \forall i \}$. For single-spin flips on a $(k, d)$-local Hamiltonian, $|\mathcal{N}_j| \leq d$ by definition. For multi-spin flips $z_j$, we get $|\mathcal{N}_j| \leq |z_j|d$.

Thus, the Boltzmann coin consists of a sequence of $N$ conditional gates $R_j$, where $R_j$ itself is a single-qubit rotation by an angle determined by the qubits in the set $\mathcal{N}_j$. Since each $\mathcal{N}_j$ are of constant-bounded sizes, each $R_j$ can be realized from a constant number of $T$ gates, so the entire Boltzmann coin requires $O(N \log \frac{1}{\epsilon})$ $T$ gates, where $\epsilon$ is the desired accuracy.

Because all gates $R_j$ act on the Coin register, they must be applied sequentially. An alternative to this consists of copying the coin register in the conjugate basis $\sigma_{y_j}$, i.e. $| \pm i \rangle \rightarrow | \pm i \rangle \otimes | N \rangle$ since a sequence of rotations $e^{i \theta_j \sigma_x}$ is equivalent to a tensor product of these rotations under this mapping. Moreover, any set of gates $R_j$ with non-overlapping $\mathcal{N}_j$ can be executed in parallel, so the total depth can be bounded by a constant at the expense of $N$ additional qubits.

The complexity of the Boltzmann coin does scale exponentially with the sparsity parameters of the model however, namely as $O(\max_j 2^{|\mathcal{N}_j|})$. A circuit that achieves $R_j$ consists of a sequence of $2^{|\mathcal{N}_j|}$ single-qubit rotations by an angle given by Eq. \[30\] or Eq. \[31\], conditioned on the bits in $\mathcal{N}_j$ taking some fixed value. Each of these $2^{|\mathcal{N}_j|}$ multi-controlled rotations require $O(|\mathcal{N}_j|)$ Toffoli gates along with $O(\log \frac{1}{\epsilon}) T$ gates, for an overall circuit depth of $O(2^{|\mathcal{N}_j|}|\mathcal{N}_j| \log \frac{1}{\epsilon})$ to realize $R_j$.

Perhaps a more efficient way to realize the Boltzmann coin uses quantum signal processing methods \[13\]–\[16\]. This is a method to construct a unitary transformation $S_2 = \sum_x e^{i \phi_x} |x\rangle\langle x|$ from a controlled version of $S_1 = \sum_x e^{i \lambda x} |x\rangle\langle x|$. In the current setting, $S_1 = \sum_x e^{i \lambda \Delta_j(x)} |x\rangle\langle x|$ and we choose $f(e^{i \lambda \Delta_j(x)}) = e^{i \theta_{2,j}}$ where $\theta_{2,j}$ is given at Eq. \[30\]. Applying a Hadamard to the Coin qubit, followed by a controlled $S_2$ with the Coin acting as control, and followed by a Hadamard on the Coin qubit again results in the transformation that we called $R_j$ above and that build up the Boltzmann coin transformation $B$.

Above, the constant $\lambda$ is chosen in such a way that the argument of the exponential $e^{i \lambda \Delta_j(x)}$ is restricted to some finite interval which does not span the entire unit circle, say in the range $[-\pi/2, \pi/2]$. The exponential can be further decomposed as a product

$$e^{i \lambda \Delta_j(x)} = \prod_{\ell \text{z}_j \neq 0} \exp \left\{ i \lambda 2 J_{\ell} \prod_{s \in \Omega_\ell} x_s \right\}. \quad \text{(32)}$$

Each of these components consists of a rotation by an angle $2J_\ell$, whose sign is conditioned on the parity of the bits in $\Omega_\ell$. The parity bit can be computed using $|\Omega_\ell\rangle$ CNOTs, and the rotation is implemented using gate synthesis, with a $T$-gate count per transformation of $O(\log \frac{1}{\epsilon})$, which is dictated by the accuracy $\epsilon$. The complexity of quantum signal processing depends on the targeted accuracy. More precisely, it scales with the number of Fourier coefficients required to approximate the function $f(e^{i \theta}) = \min(1, e^{-\theta \beta / \lambda})$ or $g(e^{i \theta}) = \frac{1}{1+e^{\beta \theta}}$ to some constant accuracy $\epsilon$ on the domain $\theta \in [-\pi/2, \pi/2]$.

### IV. HEURISTIC USE

The Metropolis-Hastings algorithm is widely used heuristically to solve minimization problems using simulated annealing or related algorithms \[17\]. The objective function is the energy $E(x)$. Starting from a random configuration or an informed guess, the random walk is applied until some low-energy configuration $x$ is reached. The parameter $\beta$ can be varied in time, with an initial low value enabling large energy fluctuations to prevent the algorithm from getting trapped in local minima, and large final value to reach a good (perhaps local) minimum.

In this section, we propose heuristic ways to use the quantum walk in the context of a minimization problem. We first recall the concept of total time to solution \[18\] which we use to benchmark and compare different heuris-
tics. We then present two quantum heuristics which we compare using numerical simulations on small instances.

Since the purpose of our study is to compare a classical walk to its different quantum incarnations – as opposed to optimizing a classical walk – we will use a schedule with a linearly increasing value of \( \beta \) in time up to a fixed final value of \( \beta \) in our comparison and expect our conclusions to hold if an optimized \( \beta \) schedule was used instead in both the classical and the quantum walks.

### A. Total time to solution

When a random walk is used to minimize some function \( E(x) \), the minimum \( x^* \) is only reached with some finite probability \( p \). Starting from some distribution \( q(x) \) and applying the walk \( W \) sequentially \( t \) times, the success probability is \( p(t) = (W^t q)(x^*) \). To boost this probability to some constant value \( 1 - \delta \), it is necessary to repeat the procedure \( L = \frac{\log(1 - \delta)}{\log(1 - p(t))} \) times. The total time to solution is then defined as the duration of the walk \( t \) times the number of repetitions \( L \),

\[
TTS(t) := t \frac{\log(1 - \delta)}{\log(1 - p(t))}.
\]  

(33)

There is a compromise to be reached between the duration of the walk \( t \) and the success probability \( p(t) \) – longer walks can reach a higher success probability and therefore be repeated fewer times, but increasing the duration \( t \) of the walk beyond a certain point has a negligible impact on its success probability \( p(t) \). We thus define the minimum total time to solution as \( \min(TTS) = \min_t TTS(t) \).

### B. Zeno with rewind

In Sec. [3] we explained how to prepare the eigenstate of \( U_W \) with eigenvalue 1 using a sequence of walks \( W^0, W^1, \ldots, W^L = W \). In the setting of Metropolis-Hastings where \( W \) is the walk with parameter \( \beta \), a natural choice of \( W^j \) are given by \( \beta^j = \frac{1}{2} \beta \). An optimized \( \beta \) schedule is also possible, but for a systematic comparison with the classical walk, we choose this fixed schedule, whose only parameter is the number of steps \( L \).

Let us revisit the argument of Sec. [3] to establish some notation. Define the binary projection measurements \( \{Q_j, Q_j^\perp\} := \{|\pi^j\rangle\langle\pi^j|, I - |\pi^j\rangle\langle\pi^j|\} \). This binary measurement can be realized from \( U^j \) uses of \( U_W \), where \( \delta_j \) denotes the spectral gap of \( U^j \). Starting from the state \( |\pi^0\rangle \), the Zeno algorithm consists in performing the sequence of binary measurements \( \{Q_j, Q_j^\perp\} \) in increasing value of \( j \). The outcome \( Q_j \) on state \( |\pi^{j-1}\rangle \) yields state \( |\pi^j\rangle \) and occurs with probability \( F_j^2 := \langle |\pi^{j-1}\rangle |\pi^j\rangle^2 \). The sequence of measurements succeeds if they all yield this outcome, which occurs with probability \( \prod_{j=1}^{L} F_j^2 \) and requires \( \sum_{j=1}^{L} \frac{1}{\delta_j} \) applications of quantum walk operator. For the algorithm to be successful, the final measurement in the computational basis must also yield the optimal outcome \( x^* \), which occurs with probability \( \pi^L(x^*) \). Thus, the total time to solution for an \( L \)-step algorithm is

\[
TTS(L) = \frac{\log(1 - \delta)}{\log(1 - \pi^L(x^*)) \sum_{j=1}^{L} \frac{1}{\delta_j}}.
\]  

(34)

In the method outlined above, a measurement outcome \( Q_j^\perp \) requires a complete restart of the algorithm to \( \beta = 0 \). There exists an alternative to a complete restart which we call rewind. It was first described in the context of Zeno state preparation in Ref. [19], but originates from Refs. [20, 21]. It consists of iterating between the measurements \( \{Q_j, Q_j^\perp\} \) and \( \{Q_j, Q_j^\perp\} \) until the measurement \( Q_j \) is obtained. It can easily be shown that a transition between outcomes \( Q_j \) and \( Q_j \) is \( F_j^2 \) while the probability of a transition between outcomes \( Q_j \) and \( Q_j \) is \( 1 - F_j^2 \). Given the cost \( \frac{1}{\delta_j} \) of each of these measurements, we obtain a simple recursion relation for the expected cost of a successful \( |\pi^{j-1}\rangle \rightarrow |\pi^j\rangle \) transition with rewind, and thus for the total time to solution for an \( L \)-step Zeno protocol with rewind. The minimal total time to solution is obtained by minimizing over \( L \). In Ref. [19], it was found that rewinding yields substantial savings compared to the regular Zeno strategy for the preparation of quantum many-body ground states.

### C. Unitary implementation

We propose another heuristic use of the quantum walk which does not use measurement. Starting from state \( |\pi^0\rangle \), it consists in applying the quantum walk operators \( U_W \) sequentially, resulting in the state

\[
|\psi(L)\rangle = U_{W^L} \ldots U_{W^2} U_{W^1} |\pi^0\rangle,
\]  

(35)

and ending with a computational basis measurement. The algorithm is successful if a computational basis measurement yields the outcome \( x^* \) on state \( |\psi(L)\rangle \) (rewind could be used otherwise), so the total time to solution for the \( L \)-step algorithm is

\[
TTS(L) = \frac{\log(1 - \delta)}{\log(1 - \langle x^* | U_{W^L} \ldots U_{W^2} U_{W^1}|\pi^0\rangle^2)}.
\]  

(36)

While we do not have a solid justification for this heuristic use, in Ref. [22], a protocol was proposed which used a similar sequence of unitaries, but where each unitary was applied a random number of times. The motivation for these randomized transformations was to phase randomize in the eigenbasis of the instantaneous unitary operator. When the spectral gap of a unitary operator is \( \delta \) and that unitary is applied a random number of times in the interval \([0, \frac{1}{\delta}]\), then the relative phase between the eigenstate with eigenvalue 1 and the other
FIG. 1. Quantum versus classical minimum total time to solution (min(TTS)) for a one dimensional Ising model of length ranging from \( n = 3 \) to \( n = 12 \) at \( \beta = 2 \). The line \( x = y \) is shown for reference of a quantum speedup.

FIG. 2. Quantum versus classical minimum total time to solution (min(TTS)) for a random sparse Ising model at \( \beta = 2 \). 100 random problem instances are chosen for each size, ranging from \( n = 4 \) to \( n = 14 \).

eigenstates is randomized over the unit circle, thus mimicking the effect of a measurement (but with an unknown outcome). From this analogy, we could expect that the unitary implementation yields a minimal total time to solution roughly equal to the Zeno-based algorithm with no rewind. But as we will see in the next section, its behavior is much better than anticipated – this method is more efficient than the Zeno algorithm with rewind, which itself is more efficient than Zeno without rewind.

D. Numerical results

We have numerically benchmarked three heuristic algorithms: the classical walk with a variable-length linear interpolation between \( \beta = 0 \) and \( \beta = 2 \) and starting from a uniform distribution; the discrete, or Zeno-based adiabatic algorithm with rewind; and the unitary algorithm of the last subsection. The first system considered is a one dimensional Ising model. Figure 1 shows the quantum versus classical minimal total time to solution. The results clearly indicate a polynomial advantages of the quantum algorithms over the classical algorithm. Surprisingly, both quantum approaches show a similar improvement over the classical approach that exceed the expected quadratic speedup, with a power law fit of 0.42 using the unitary algorithm and 0.39 using the Zeno algorithm.

In both the one-dimensional and the random graph Ising model, the unitary quantum algorithm achieves very similar and sometimes superior scaling to the Zeno with rewind algorithm. This is surprising given the observed improvement obtained from rewind in Ref. [19] and our expectation that the unitary algorithm behaves essentially like Zeno without rewind.

V. DISCUSSION

In Appendix A, we discuss the difficulty of implementing the quantum walk oracle \( W \). Our conclusion, and perhaps one of the key messages of this Article, is that even though the quantum walk is traditionally defined with the help of a walk oracle, its circuit implementation does not necessarily require this walk oracle, and this can lead to substantial savings. Appendix B presents an improved parallelized heuristic classical walk for discrete sparse optimization problems which could potentially lead to significant improvements on a quantum computer. Unfortunately this walk is not reversible.
which motivates further generalization of Szegedy’s quantization to include irreversible classical walks. In the rest of this section, we discuss the prospect of using the quantum walk to outperform a classical supercomputer.

We have proposed heuristic quantum algorithms based on the Szegedy walk for solving discrete optimization problems. Theoretical bounds show that the quantum algorithm can benefit from a quadratic speed-up \((x^{0.5})\) over its classical counterpart. Our numerical simulations on small problem instances indicate a super-quadratic speed-up \((\approx x^{0.42})\) for the Ising chain and sub-quadratic speed-up \((\approx x^{0.75})\) for random sparse Ising graphs. It remains an interesting question to understand more broadly what type of problems can benefit from what range of speed-up, and with these crude estimates in hand we can already look into the feasibility of a quantum speed-up on realistic devices.

To do so we will compare performances to the special-purpose supercomputer “Janus” \([23, 24]\) which consists of a massive parallel field programable gate array (FPGA). This system is capable of performing \(10^{12}\) Monte Carlo spin updates per second on a three-dimensional Ising spin glass of size \(n = 80^3\). A calculation that lasts a bit less than a month will thus realize \(10^{15}\) Monte Carlo steps. On the one hand, assuming that the theoretically predicted quadratic speed-up holds, the quantum computer must realize at least \(10^9\) steps per month in order to keep up with the classical computer. This requires that a single step of the quantum walk be realized in a few milliseconds. On the other hand, the super-quadratic speed-up we have observed would allow almost a tenth of a second to realize a single quantum step, while the sub-quadratic speed-up would require that a single step be realized within 0.1 microseconds.

Taking the circuit depth reported in Table I as reference with \(d = 6\) for a three-dimensional lattice leads to a circuit depth of \(\log(80^3) \times 2^6 \approx 1000\). To avoid harmful error accumulation, the gate synthesis accuracy \(\epsilon\) should be chosen as the inverse volume of the quantum circuit, roughly \(\epsilon^{-1} \approx 80^3 \times \log(80^3) \times 10^9 \approx 10^{16}\), so on the order of \(4 \log \frac{1}{\epsilon} \approx 200\) logical \(T\) gates are required per fine-tuned rotation \([25, 24]\), for a total logical circuit depth of \(200,000\). With these estimates, the three scenarios described above require logical gate speeds ranging from an unrealistically short 0.5 picoseconds (sub-quadratic speed-up), to an extremely challenging 1 nanosecond (quadratic speed-up), and allow 0.5 microseconds for super-quadratic speed-up.

We could instead compile the rotations offline and teleport them in the computation \([27]\), which requires at least \(4 \log \frac{1}{\epsilon} \approx 200\) more qubits, but increase the time available for a logical gate by the same factor. Under this scenario, the time required for each logical gate would range from 0.1 nanoseconds (sub-quadratic speed-up), to 20 microseconds (quadratic speed-up), and to 1 milliseconds (super-quadratic speed-up). These estimates are summarized in Table \(III\). The latter is a realistic logical gate time for many qubit architectures, while there is no current path to achieve nanosecond logical gate times. It is thus important to understand the class of problems for which heuristic super-quadratic speed-ups can be achieved and to continue to optimize circuit implementations.

### VI. ACKNOWLEDGEMENTS

We thank Jeongwan Haah, Thomas H"aner, Matt Hastings, and Guang Hao Low for stimulating discussions.

#### Appendix A: Walk oracle

Our implementation of the walk operator does not make use of the walk oracle \(W\) of Eq. \(2\). Since the transition matrix elements \(W_{xy}\) can be computed efficiently, we know that \(W\) can be implemented in polynomial time. But this requires costly arithmetics which would yield a substantially larger complexity than the approach presented above.

To see how this complexity arises, consider the following implementation of \(W\), which uses much of the same elements as introduced above. The computer comprises two copies of the system register, which we now label Left and Right. As before, it also comprises a Move register and a Coin register. Begin with the Left register in state \(x\) and all other registers in state 0. Use the transformation \(V\) to prepare the state \(|x\rangle_L \otimes |0\rangle_R \otimes |f\rangle_M \otimes |0\rangle_C\). Using \(n\) CNOTs, copy the state of the Left register onto the Right register, resulting in \(|x\rangle_L \otimes |x\rangle_R \otimes |f\rangle_M \otimes |0\rangle_C\). Apply the move \(z_j\) proposed by the Move register to the Right register. If the Move register is encoded in unary representation as above, this requires \(O(N)\) CNOTs, and results in the state

\[
|x\rangle_L \otimes \sum_{j \in \mathcal{M}} \sqrt{f(z_j)} |x \cdot z_j\rangle_R \otimes |z_j\rangle_M \otimes |0\rangle_C. \tag{A1}
\]

Using a version of the Boltzmann coin transformation on

| Quantum speedup | Synthesis online | Synthesis offline |
|-----------------|------------------|------------------|
| Sub-quadratic \(x^{0.75}\) | 0.5 ps | 0.1 ns |
| Quadratic \(x^{0.5}\) | 1 ns | 20 μs |
| Super-quadratic \(x^{0.42}\) | 0.5 μs | 1 ms |

TABLE II. Logical gate time required to outperform a supercomputer capable of realizing \(10^{12}\) Monte Carlo updates per nanosecond in a computation that lasts one month. Arbitrary single-qubit rotations can be synthesized online or offline at an additional qubit cost.
the quantum walk operator
\[ W \]
state detail. When the acceptance register is in state 0, the
\[ f(z) \]
with controls from the Left and Right registers. At this
\[ f(z) \]
rotation clearly depends on all the coefficients
\[ f(z) \]
needs to be mapped to the state
\[ f(z) \]
conditioned on the Coin qubit being in state 1, resulting in
\[ f(z) \]
This is quite similar to the state that would result from
\[ f(z) \]
all implementations we could envision used arithmetic operations that compute \( A_{xy} \).

Appendix B: Irreversible parallel walk

Note that the Boltzmann operator \( B \) has a total number of gates that scales with the system size \( n \), even though it is used to implement a single step of the quantum walk and that on average, a single spin is modified per step of the walk. This contrasts with the classical walk where in a single step of \( W \), a spin transition \( x \to x \cdot z \) is chosen with probability \( f(z) \), the acceptance probability is computed, and the move is either accepted or rejected. Each transition \( x \to x \cdot z \) typically involves only a few spins (one in the setting we are currently considering), so implementing such a transition in the classical walk does not require an extensive number of gates. The complexity in that case is actually dominated by the generation of a pseudo-random number selecting the location of the spin to be flipped. As a consequence, the quantum algorithm suffers an \( n \)-fold complexity increase compared to the quantum algorithm.

This motivates the construction of a modified classical walk for the lattice spin model which also affects every spin of the lattice, putting the classical and quantum walks on equal footing in terms of gate count. For simplicity, suppose that the set of moves \( z_i \in \mathcal{M} \) consist in single-spin flips. We define a parallel classical walk with transition matrix
\[ W_{yx} = \prod_{j=1}^N [q B_i(x)] (1 - \frac{\beta}{|x|}) [1 - q B_i(x)] (1 + \frac{\beta}{|x|}), \] (B1)
where \( B_i(x) = \min\{1, e^{\beta[E(x) - E(x,z_i)]}\} \) and \( z_i \) is the transition which consist of flipping the \( i \)-th spin only, so only spin \( i \) differ in \( x \) and \( x \cdot z_i \). The variable \( 0 \leq q \leq 1 \) is a tunable parameter of the walk. In other words, a single step of this walk can be decomposed into a sequence over spins \( i \), and consists of flipping \( i \) with probability \( q \) and accepting the flip with probability \( B_i(x) \). Importantly, even if the moves are applied sequentially, the acceptance probability \( B_i(x) \) is always evaluated relative to the state at the beginning of the step, even though other spins could have become flipped during the sequence.

If instead the acceptance probability was evaluated conditioned on the previously accepted moves – i.e., \( B_i(x) = \min\{1, e^{\beta[E(x) - E(x,z_i)]}\} \) where \( z_i^{\text{tot}} \) is the total transition accumulated up to step \( i \) – then this acceptance probability would be the same as used in the Metropolis-Hastings algorithm. Note that for a local spin model with, e.g., nearest-neighbor interactions, the two acceptance probabilities only differ if a neighbor of site \( i \) has been flipped prior to attempting to flip spin \( i \). Because a transition on each spin is proposed with probability \( q \), the probability of having two neighboring spins flipped is \( \mathcal{O}(q^2) \). Thus, we essentially expect a single step of this modified walk to behave like \( q \) steps of the original Metropolis-Hastings walk, with a systematic error that scales like \( nq^2 \). Moreover, this systematic error is expected to decrease over time since once the walk settles in a low-energy configuration, very few spin transitions will turn out to be accepted, thus further decreasing the probability of a neighboring pair of spin flips.

To verify the above expectation, we have performed numerical simulations on a Ising model
\[ H = \sum_{i,j} J_{i,j} x_i x_j \]
where \( J_{i,j} \) were randomly chosen from \{+1, -1\}. Results are shown on Fig. 3. What we observe is that, for an equal amount of computational resources, the parallelized walk outperforms the original walk. This is true both in terms in reaching a quick pseudo minimum configuration at short times and in terms of reaching the true minimum at longer times. Thus, while this parallelization was introduced to ease the quantization procedure, it appears to be of interest on its own.
In this case, the quantum walk oracle $W$ can easily be applied. We first proceed as in the previous subsection and use CNOTs to copy the Right register onto the Left register, yielding state $|x\rangle_L \otimes |x\rangle_R$. Then, sequentially over all spins $i$, apply a rotation to spin $i$ of the Left register conditioned on the state of the spin $i$ and its neighbors on the Right register. This rotation transforms $|x_i\rangle \rightarrow \sqrt{1-qB_i(x)}|x_i\rangle + \sqrt{qB_i(x)}|\pi_i\rangle$. Note that the function $B_i(x)$ only depends on the bits of $x$ that are adjacent to site $i$, so this rotation acts on a constant number of spins so requires a constant number of gates. Thus, the cost of the classical and the quantum parallel walks have the same scaling in $n$. Combined to its observed advantages over the original classical walk, the parallel walk thus appears as the ideal version for a quantum implementation.

Unfortunately, the parallel walk is not reversible – it does not obey the detail-balance condition Eq. (1). Thus, it is not directly suitable to quantization à la Szegedy. While quantization of non-reversible walks were considered in [10], they require an implementation of time-reversed Markov chain $W^*$ defined from $W$ and its fixed point $\pi$ as

$$W^*_{xy} \pi_x = W_{yx} \pi_x.$$  

Unfortunately, we do not know how to efficiently implement a quantum circuit for the time-reversed walk $W^*$, so at present we are unable to quantize this parallel walk.

[1] M. Szegedy, in Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (2004).
[2] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, “Quantum computation by adiabatic evolution,” (2000), quant-ph/0001106.
[3] D. Aharonov and A. Ta-Shma, Proc. 35th Annual ACM Symp. on Theo. Comp., 20 (2003).
[4] S. Boixo, E. Knill, and R. D. Somma, “Fast quantum algorithms for traversing paths of eigenstates,” (2010), 1005.3034.
[5] R. Somma, S. Boixo, H. Barnum, and E. Knill, Phys. Rev. Lett., 101, 130504 (2008), arXiv:0804.1571.
[6] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, J. Chem. Phys., 21, 1087 (1953).
[7] W. Hastings, Biometrika, 57, 97 (1970).
[8] A. Kitaev, “Quantum measurements and the abelian stabilizer problem,” (1995), quant-ph/9511026.
[9] A. Ambainis, in Proc. 45th Annual IEEE Symp. on Foundations of Computer Science (FOCS) (2004) quant-ph/0311001.
[10] P. Magniez, A. Nayak, J. Roland, and M. Santha, SIAM journal on computing, 40, 142 (2011).
[11] T. Rudolph and L. Grover, “A 2–rebit gate universal for quantum computing,” (2002), quant-ph/0210187.
[12] D. Barenco, C. H. Bennett, R. Cleve, D. P. DiVinci, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, Phys. Rev. A, 52, 3457 (1995), quant-ph/9503016.
[13] G. Low, T. Yoder, and I. L. Chuang, Phys. Rev. X, 6, 041067 (2016).
[14] G. H. Low and I. L. Chuang, “Hamiltonian simulation by qubitization,” (2016), arXiv:1610.06546.
[15] G. H. Low and I. L. Chuang, Phys. Rev. Lett., 118, 010501 (2017).
[16] J. Haah, “Product decomposition of periodic functions in quantum signal processing,” (2018), arXiv:1806.10236.
[17] S. Kirkpatrick, C. J. Gelatt, and M. Vecchi, Science, 220, 4598 (1983).
[18] T. Ronnow, S. Wang, J. Job, S. Boixo, S. Isakov, D. Wecker, J. Martinis, D. Lidar, and M. Troyer, Science, 345, 420 (2014), arXiv:1401.2910.
[19] J. Lemieux and D. Poulin, “Resource estimate for quantum many-body ground state preparation on a quantum computer,” In preparation (2019).
[20] C. Marriott and J. Watrous, Computational Complexity, 14, 122 (2005), cs/0506068.
[21] K. Temme, T. Osborne, K. Vollbrecht, D. Poulin, and F. Verstraete, Nature, 471, 87 (2011), arXiv:0911.3635.
[22] S. Boixo, E. Knill, and R. Somma, Quant. Info. and Comp., 9, 833 (2009).
[23] Janus Collaboration, Comp. Sci. Eng., 11, 48 (2009).
[24] Janus Collaboration, Eur. J. Phys., 310, 33 (2012).
[25] N. Ross and P. Selinger, Quant. Info. and Comp., 16, 901 (2016), arXiv:1403.2975.
[26] A. Bocharov, M. Rotteler, and K. Svore, Phys. Rev. A, 91 (2015).
[27] N. C. Jones, J. D. Whitfield, P. L. McMahon, M.-H. Yung, R. Van Meter, A. Aspuru-Guzik, and Y. Yamamoto, New Journal of Physics, 14, 115023 (2012).