Compilation of Fault-Tolerant Quantum Heuristics for Combinatorial Optimization

Yuval R. Sanders,[1,2] Dominic W. Berry,[1,*] Pedro C.S. Costa,[1] Louis W. Tessler,[1] Nathan Wiebe,[3,4,5] Craig Gidney,[5] Hartmut Neven,[5] and Ryan Babbush[5,†]

[1] *Department of Physics and Astronomy, Macquarie University, Sydney, NSW 2109, Australia*
[2] *ARC Centre of Excellence in Engineered Quantum System, Macquarie University, Sydney, NSW 2109, Australia*
[3] *Department of Physics, University of Washington, Seattle, Washington 18195, USA*
[4] *Pacific Northwest National Laboratory, Richland, Washington 99354, USA*
[5] *Google Research, Venice, California 90291, USA*

(Received 23 July 2020; accepted 29 September 2020; published 9 November 2020)

Here we explore which heuristic quantum algorithms for combinatorial optimization might be most practical to try out on a small fault-tolerant quantum computer. We compile circuits for several variants of quantum-accelerated simulated annealing including those using qubitization or Szegedy walks to quantize classical Markov chains and those simulating spectral-gap-amplified Hamiltonians encoding a Gibbs state. We also optimize fault-tolerant realizations of the adiabatic algorithm, quantum-enhanced population transfer, the quantum approximate optimization algorithm, and other approaches. Many of these methods are bottlenecked by calls to the same subroutines; thus, optimized circuits for those primitives should be of interest regardless of which heuristic is most effective in practice. We compile these bottlenecks for several families of optimization problems and report for how long and for what size systems one can perform these heuristics in the surface code given a range of resource budgets. Our results discourage the notion that any quantum optimization heuristic realizing only a quadratic speedup achieves an advantage over classical algorithms on modest superconducting qubit surface code processors without significant improvements in the implementation of the surface code. For instance, under quantum-favorable assumptions (e.g., that the quantum algorithm requires exactly quadratically fewer steps), our analysis suggests that quantum-accelerated simulated annealing requires roughly a day and a million physical qubits to optimize spin glasses that could be solved by classical simulated annealing in about 4 CPU-minutes.

DOI: 10.1103/PRXQuantum.1.020312

I. INTRODUCTION

The prospect of quantum-enhanced optimization has driven much interest in quantum technologies over the years. This is because discrete optimization problems are ubiquitous across many industries and faster solutions could potentially revolutionize fields as broad as logistics, finance, machine learning, and more. Since combinatorial optimization problems are often NP hard, we do not expect that quantum computers can provide efficient solutions in the worst case. Rather, the hope is that there may exist ensembles of instances with structure that enable a significant quantum speedup on average, or for which a quantum computer can approximate better solutions.

Among the most studied algorithms for quantum optimization are those that can function as heuristics. The objective of a heuristic algorithm is to produce a solution given a reasonable amount of computational resources that is "good enough" (or at least the best one can afford) for solving the problem at hand. While heuristics are often able to efficiently find the exact solution, sometimes they might fail to do so and instead only approximate the exact solution (potentially in an uncontrolled fashion). But such techniques are still valuable because finding some usable result does not require a prohibitively long time. Accordingly, heuristics are often used without regard for rigorous bounds on their performance. Indeed, the NP hardness of many combinatorial optimization problems makes heuristics the only viable option for many problems that need to be routinely solved in real-world applications.

[*] dominic.berry@mq.edu.au
[†] ryanbabbush@gmail.com

While some heuristic algorithms have a strong theoretical basis, many of the most effective heuristics are based on intuitive principles and then honed empirically through data and experimentation. However, today, our ability to evaluate quantum heuristics through experimentation is limited since the only available quantum computers are small and noisy [1]. We can perform numerics on small instances but extrapolation from those small system size numerics can be potentially misleading [2]. Still, it is reasonable to ask the question: what are some of the most compelling quantum heuristics for optimization that we want to attempt on a small fault-tolerant quantum computer, and how many resources are required to implement their primitives?

There are many prominent approaches to combinatorial optimization on a quantum computer. These include variants of Grover's algorithm [3,4], quantum annealing [5,6], adiabatic quantum computing [7,8], the shortest-path algorithm [9], quantum-enhanced population transfer [10,11], the quantum approximate optimization algorithm [12], quantum versions of classical simulated annealing [13,14], quantum versions of backtracking [15,16] as well as branch and bound techniques [17], among many others. While often these works focus on the asymptotic scaling of exact quantum optimization, in many cases one can use these algorithms heuristically through trivial modifications of the approach. For instance, the quantum adiabatic algorithm requires that one evolve the system for an amount of time scaling polynomially with the inverse of the minimum spectral gap of the adiabatic evolution. However, one can instead use this algorithm as a heuristic by choosing to evolve for a much shorter amount of time, and hoping for the best (this is similar to the strategy usually employed with quantum annealing).

What essentially all forms of quantum optimization have in common is the requirement that the quantum algorithm query some function of the cost function of interest. This is how the quantum computer accesses information about the energy landscape. For instance, if our cost function is H and $H\left|x\right\rangle = E_x\left|x\right\rangle$ so that E_x is the value of the cost function for bit string $\left|x\right\rangle$, then often we need to phase the computational basis by a function $f\left(\cdot\right)$ of E_x, e.g.,

$$\sum_x a_x\left|x\right\rangle \mapsto \sum_x e^{-if\left(E_x\right)} a_x\left|x\right\rangle . \tag{1}$$

For example, $f\left(E_x\right) \propto E_x$ is required to implement the quantum approximate optimization algorithm, quantum-enhanced population transfer, digitized forms of quantum annealing, and the shortest-path algorithm. Alternatively, $f\left(E_x\right) \propto \arccos(E_x)$ describes something related to the quantum walk forms of those algorithms. If $f\left(E_x\right) \propto (-1)^{(E_x \le K)}$ this primitive is the bottleneck subroutine for amplitude amplification to boost our support on energies less than K. In most quantum approaches to optimization, a unitary like this is interleaved with a much cheaper operation, which does not commute with the operation in Eq. (1). Some algorithms instead call for simultaneously evolving under a function of the cost function together with a simple noncommuting Hamiltonian, but still the bottleneck is usually the complexity of the cost function Hamiltonian. The difference between many of these algorithms often comes down to the choice of $f\left(\cdot\right)$ and the choice of the much cheaper noncommuting unitary.

The quantum algorithms for simulated annealing (e.g., Ref. [13]) work slightly differently as those algorithms are based on making local updates to the wavefunction. For instance, the quantum version of a simulated annealing algorithm that updates with single bit flips requires

$$\sum_x a_x\left|k\right\rangle\left|x\right\rangle\left|0\right\rangle \mapsto \sum_x a_x\left|k\right\rangle\Bigg[\sqrt{1-f\left(E_x, E_{x_k}\right)}\left|x\right\rangle\left|0\right\rangle + \sqrt{f\left(E_x, E_{x_k}\right)}\left|x_k\right\rangle\left|1\right\rangle\Bigg], \tag{2}$$

where x_k is defined as the bit string x with the kth bit flipped, i.e., $\left|x_k\right\rangle = \mathrm{NOT}_k\left|x\right\rangle$, with $k=0$ corresponding to no bit flip. But again, these approaches are still typically bottlenecked by our ability to compute these functions of the cost function $f\left(\cdot\right)$.

This paper does *not* address the important question of how well various heuristic quantum optimization approaches might perform in practice. Rather, our main motivation is to compile common bottleneck primitives for these approaches to quantum circuits suitable for execution on a small fault-tolerant quantum computer. In doing this, we see that most contemporary approaches to quantum optimization are actually bottlenecked by the same subroutines [e.g., those required for Eqs. (1) and (2)], and thus improved strategies for realizing those subroutines are likely of interest regardless of which paradigm of quantum optimization is ultimately found to be most effective in practice. In essentially all heuristic approaches to quantum optimization there is a primitive that is repeated many times in order to perform the optimization. Instead of investigating how many times those primitives must be repeated, we focus on the best strategies for realizing those primitives within a fault-tolerant cost model. For all algorithms we consider, we report the constant factors in the leading-order scaling of the Toffoli and ancilla complexity of these primitives.

For some algorithms studied, such as for the quantum algorithms for simulated annealing, this work is the first to give concrete implementations, which determine constant factors in the scaling. In other cases our contribution is to optimize the scaling for certain problem Hamiltonians or improve details of the implementation. We focus on Toffoli complexity since we imagine realizing these algorithms in the surface code [18,19], where non-Clifford gates such as Toffoli or T gates require considerably more time (and physical qubits) to implement than Clifford gates.

A. Overview of results

The goal of this paper is to estimate the performance of an early universal quantum computer for key steps of combinatorial optimization. To achieve this goal, we consider prominent heuristic-based methods for combinatorial optimization on a quantum computer and how their key steps could be executed on early hardware. We consider the following heuristic-based methods: amplitude amplification [20] as a heuristic for optimization and in combination with other approaches; quantum approximate optimization algorithms (QAOA) [12]; time-evolution approaches such as adiabatic algorithms [2] (including a variant incorporating a Zeno-like measurement [21]), quantum-enhanced population transfer [11], and "shortest-path" optimization [9]; and three quantum methods for simulated annealing (QSA), namely, a Szegedy walk-based [22] implementation of Markov chain Monte Carlo [13], a qubitized form of the Metropolis-Hastings approach [23], and simulation of a spectral-gap-amplified Hamiltonian [14]. We review existing approaches in detail and develop several new methods or improvements. For each approach, we compile the primitive operations into quantum circuits optimized for execution in the surface code [19].

For concreteness, we focus our analysis on four families of combinatorial optimization problems: the L-term spin model, in which the Hamiltonian is specified as a real linear combination of L tensor products of Pauli-Z operators; quadratic unconstrained binary optimization (QUBO), which is an NP-hard special case of a two-local L-term spin model; the Sherrington-Kirkpatrick (SK) model, which is a model of spin-glass physics and an instance of QUBO that has been well studied in the context of simulated annealing [24]; and the low autocorrelation binary sequence (LABS) problem, which is a problem with many terms but significant structure that is known to be extremely challenging in practice. For each of the above problems, we design several methods of calculating the cost function on a quantum computer depending on how a given algorithmic primitive is supposed to query and process the cost of a candidate solution. We present these methods in Sec. II.

Our analysis has produced several novel techniques that yield improvements over previous approaches. We recount the main ones here in order of appearance. In Sec. II A 2, we reduce by a logarithmic factor the cost of calculating the Hamming weight of a bit string using our method from Ref. [25]. This new technique leads to improvements in several other parts of our paper. In Sec. II E, we introduce a new technique for evaluating costly arithmetic functions when computational cost matters more than accuracy. Our new technique is based on approximating the function using linear interpolation between classically precomputed points that can be accessed using quantum read-only memory (QROM) [26], or a new variant of QROM designed for sampling at exponentially growing spacings.

In Sec. III B, we introduce a method of cost function evaluation for QAOA based on amplitude estimation. This technique gives a quadratic improvement over the original approach. In Sec. III C, we introduce a heuristic method for adiabatic optimization that is likely to be computationally cheaper for some applications of early quantum computers, although we do not expect an asymptotic advantage over other state-of-the-art approaches. The idea is to simulate the adiabatic path generated by the arccosine of the given Hamiltonian, not by the Hamiltonian directly, by "stroboscopically" simulating time evolution with short time steps produced by evolving under a qubitized walk.

In Sec. III D we give a new method for constructing the Szegedy walk operator suggested in Ref. [13]. Our key technique is a state preparation circuit that avoids expensive on-the-fly calculations by using the techniques introduced in Ref. [27]. In Sec. III E, we introduce an alternative method for executing the controlled qubit-rotation step in the qubitized Metropolis-Hastings approach introduced in Ref. [23]. Our approach is preferable in cases where the Hamiltonian has a higher connectivity; i.e., when the probability of accepting a proposed transition depends on many bits in the candidate solution. In those cases the approach of Ref. [23] have exponential complexity. In Sec. III F, we give an explicit linear combination of unitaries- (LCU) based oracle for the spectral-gap-amplified Hamiltonian introduced in Ref. [14]. This explicit oracle enables a cost analysis of the approach, which we provide. Apart from assisting with our goal of estimating early quantum computer performance, many of these innovations produce asymptotic improvements to the approaches we consider.

Having compiled the primitive operations of our chosen approaches and established how to query cost functions for our chosen problems, we are able to numerically estimate the computational resources needed to execute these primitives on a quantum computer. Based on our assumption that the quantum computer is built from superconducting qubits and employs the surface code to protect the computation from errors, we focus on minimizing the number of ancilla qubits and non-Clifford gates that is required. This approach is founded on the knowledge that non-Clifford operations are significantly harder than Clifford operations to perform in the surface code.

We summarize our ultimate findings in Tables I and II. In Table I we provide the leading-order scaling of the number of Toffoli gates needed to perform an update using five of the heuristics that we consider for two benchmark problems—LABS and SK. In Table II we give asymptotic scalings for the two more general benchmark problems we consider—the L-term spin model and QUBO. The scalings are reproduced from Table VIII and presented in a simplified form where we assume that the working precision for various calculations is a constant.

Table I also reproduces key figures from Tables IX and X to show how we expect these estimated complexity scalings translate into the runtime of an early quantum computer. We show the estimated number of steps of the chosen algorithmic primitive that could be executed in a single day on a quantum computer for a problem size of $N = 256$, a relatively small problem size that is reasonable to execute with only a single Toffoli factory as we assume in Tables IX and X. We also present the estimated number of physical qubits needed. Our estimation is based on the assumption that the runtime of the quantum computer is dominated by the cost of executing non-Clifford gates, and so we have chosen to treat the Clifford operations as free. Future researchers should plan to count Clifford operations as more sophisticated compilers are developed [28,29]. Such research could also allow for performance assessment on other target architectures [30].

We find that, despite great efforts made to optimize our compiled quantum circuits, the costs involved in implementing heuristics for combinatorial optimization is taxing for early quantum computers. Not surprisingly, to implement problems between $N = 64$ and $N = 1024$ we find that hundreds of thousands of physical qubits are required when physical gate-error rates are on the order of 10^{-4} and sometimes over a million are required for physical gate-error rates on the order of 10^{-3}. But even more concerning is that the number of updates that we can achieve in a day (given realistic cycle times for the error-correcting codes) is relatively low, on the order of about ten thousand updates for the smallest instances considered of the cheapest cost functions. With such overheads, these heuristics need to yield dramatically better improvements in the objective function per step than classical optimization heuristics. From this we conclude that, barring significant advances in the implementation of the surface code (e.g., much faster state distillation), quantum optimization algorithms offering only a quadratic speedup are unlikely to produce any quantum advantage on the first few generations of superconducting qubit surface code processors.

B. Organization of paper

Our paper is divided into essentially two parts. In the first part (Sec. II) we introduce and provide explicit compilations for a wide variety of subroutine or “oracle” circuits, which perform operations related to specific problem Hamiltonians. In the second part of our paper (Sec. III) we describe a variety of heuristic algorithms for quantum optimization and discuss how the oracle circuits of Sec. II can be called in order to implement these algorithms. We see that the same “oracle” circuits are required by many algorithms. The results of Sec. III essentially provide query complexities to implement the primitives of common quantum optimization heuristics with respect to the oracles of Sec. II. Thus, while the results of Sec. II are adapted to particular problem Hamiltonians, the results of Sec. III are fairly general. We now describe our results in slightly more detail.

TABLE I. We compare the cost of implementing various types of heuristics optimization primitives in a fault-toleration cost model. For concreteness, we give results for two problems: the SK model and the LABS. We simplify the complexity scaling estimates from Table VIII by treating as constant the bits of precision for numerical values. Note that depending how they are used, it might be appropriate to scale the Hamiltonian walk steps by a factor of λ, which is roughly $\lambda_{\mathrm{SK}} \approx N^2/2$ and $\lambda_{\mathrm{LABS}} \approx N^3/3$. The numerical values from Tables IX and X are based on a problem size of $N = 256$, a surface code cycle time of 1 μs, and a physical gate-error rate of 10^{-3} (there are other assumptions as well, covered in more detail in Sec. IV).

		(Tables IX and X)		(Table VIII)
Problem	Algorithm primitive	Steps per day	Physical qubits	Toffoli count
SK	Amplitude amplification (Sec. III A)	4.8×10^3	8.1×10^5	$2N^2 + N + \mathcal{O}(\log N)$
	QAOA/first-order Trotter (Sec. III B)	4.7×10^3	8.6×10^5	$2N^2 + 4N + \mathcal{O}(1)$
	Hamiltonian walk (Sec. III C)	3.3×10^5	8.0×10^5	$6N + \mathcal{O}(\log^2 N)$
	QSA/qubitized (Sec. III E)	3.3×10^5	8.4×10^5	$5N + \mathcal{O}(\log N)$
	QSA/gap amplification (Sec. III F)	3.9×10^5	8.4×10^5	$5N + \mathcal{O}(\log N)$
LABS	Amplitude amplification (Sec. III A)	3.3×10^3	8.0×10^5	$5N^2/2 + 7N/2 + \mathcal{O}(\log N)$
	QAOA/first-order Trotter (Sec. III B)	3.4×10^3	8.4×10^5	$5N^2/2 + \mathcal{O}(N)$
	Hamiltonian walk (Sec. III C)	4.9×10^5	8.0×10^5	$4N + \mathcal{O}(\log N)$
	QSA/qubitized (Sec. III E)	1.7×10^3	8.8×10^5	$5N^2 + \mathcal{O}(N)$
	QSA/gap amplification (Sec. III F)	1.7×10^3	8.8×10^5	$5N^2 + \mathcal{O}(N)$

TABLE II. A brief summary of our results for the L-term spin model and QUBO; see Table VIII for details. We present the Toffoli complexity of the algorithm primitives to leading order in the problem parameters. Our algorithms require the user to specify the number of bits of precision at various stages of the algorithms, each of which are denoted in Table VIII by a parameter b with a subscript indicating which step requires the specification. Here we simplify by supposing the user chooses a single parameter b that dictates all these other precision parameters. Unlike in Table I, we do not estimate the runtime for these algorithm primitives on a fixed problem size as doing so requires arbitrary choices about which instances to simulate.

Problem	Algorithm primitive	Toffoli count
L-term spin	Amplitude amplification (Sec. III A)	$2bL + N + \mathcal{O}(b)$
	QAOA/first-order Trotter (Sec. III B)	$1.15(\log L + b)L + \mathcal{O}(N + \log L + b^2)$
	Hamiltonian walk (Sec. III C)	$3L + \mathcal{O}(\log L + b)$
	QSA/qubitized (Sec. III E)	$4bL + N + \mathcal{O}(\log N + b^2)$
	QSA/gap amplification (Sec. III F)	$4bL + N + \mathcal{O}(\log N + b^2)$
QUBO	Amplitude amplification (Sec. III A)	$bN^2 + \mathcal{O}(bN)$
	QAOA/first-order Trotter (Sec. III B)	$(1.15 \log N + 0.575b)N^2 + \mathcal{O}(N^2)$
	Hamiltonian walk (Sec. III C)	$(2 \log N + b)N + \mathcal{O}(N)$
	QSA/qubitized (Sec. III E)	$(2b + 1)N + \mathcal{O}(\log N + b^2)$
	QSA/gap amplification (Sec. III F)	$(2b + 1)N + \mathcal{O}(\log N + b^2)$

Section II details strategies for realizing five straightforward oracle circuits, which are detailed therein for each of four problem Hamiltonians in Table IV. The specific problems we focus on are introduced at the beginning of Sec. II. These five oracles correspond to the following: (Sec. II A) the direct computation of a cost function into a quantum register, (Sec. II B) the computation of the difference between the cost of two computational basis states, which differ by a specific single bit, (Sec. III C) an operation that phases the computational basis by an amount proportional to the cost, (Sec. III D) the realization of a qubitized quantum walk [31], which encodes eigenvalues of the cost function, and (Sec. II E) the computation of arithmetic functions of an input value using QROM [26]. Our approach to computing arithmetic operations using QROM is likely useful in other contexts and is a new technique from this work. The culmination of Sec. II is Table V, which gives leading-order constants in the scaling of Toffoli, T, and ancilla complexities for all five of these oracles and for all four of the problems. Even though the first two cost functions we introduce in Sec. II have fairly general specifications, they do not capture exploitable structure in all optimization problems of interest. Still, we imagine that the motifs developed in Sec. II is helpful for any future work seeking to develop similar circuits for other cost functions.

Section III describes how the oracle circuits of Sec. II are queried in order to realize the essential primitives of many fault-tolerant quantum heuristics for optimization. This section contains a mixture of new results and a review of established methods. Sec. III A reviews how one can use amplitude amplification [20] heuristically for optimization and also discusses how and why one might combine amplitude amplification with other algorithms in this section. Section III B discusses strategies for executing QAOA [12] within fault-tolerant cost models. While most of this section is review, we also discuss the combination of QAOA with amplitude-amplification-based methods for more efficiently extracting the cost function value.

Section III C discusses several approaches to quantum optimization that are based on time evolution or quantum walks generated by a cost function and simple driver. First, we review the adiabatic algorithm [2] and well-known methods for how it might be digitized using product-formula-type circuits. We then introduce a method of simulating the adiabatic algorithm based on qubitized quantum walks. Next, we review how the adiabatic algorithm can be combined with a Zeno-like measurement approach, which corresponds to evolution under static Hamiltonians for random durations [21], and give some new results about how to optimally choose the distribution of those durations.

The remainder of Sec. III focuses on three approaches to a quantum algorithm, which accelerates classical simulated annealing. In terms of implementation, these are the most complex algorithms studied in the paper. For the three variants of the quantum simulated annealing algorithms, we provide the first complete compilation of circuits, which execute the heuristic primitive. In Sec. III D we analyze and compile the original version [13] of these algorithms, which is based on Szegedy quantum walks [22]. As anticipated, this approach is the least efficient of the three studied. In Sec. III E we focus on what is essentially a qubitized version of the Szegedy quantum walk. The primary characteristics of this approach were independently described in Ref. [23] (a paper that came out during the preparation of our own) but we go beyond that work to determine (and in some ways improve upon) constant factors in the scaling. Finally, in Sec. III F we compile the algorithm for quantum simulated annealing based on spectral-gap amplification [32], using an improvement based on qubitization. The results of Sec. III are summarized in Tables VII and VIII, which give the query complexities with respect to the oracles of Sec. II and overall gate and ancilla complexities of all algorithms of Sec. III for all of the cost functions of Sec. II.

Finally, we conclude in Sec. IV with a discussion of these results. Our discussion includes an attempt to contextualize the ultimate cost of these heuristic primitives by giving the Toffoli count, ancilla count, and the total number of physical qubits and wallclock time that is required to realize these primitives given various resource budgets and assumptions in the surface code. These concrete resource estimates are given in Tables IX and X. We then finish with a discussion of how these results lead to a fairly pessimistic outlook on the viability of obtaining quantum advantage for optimization by using a small quantum computer unless one is able to obtain significantly better than a quadratic speedup over classical alternatives.

II. ORACLES AND CIRCUIT PRIMITIVES FOR SPECIFIC COST FUNCTIONS

While many paradigms of quantum optimization require the same bottleneck subroutines for their implementation, aspects of these subroutines are always specific to the particular problem that one intends to optimize. Thus, in order to give concrete implementations and develop a sense of how many resources are required for steps of common quantum heuristics, aspects of our work are adapted to particular problem Hamiltonians (equivalently here, "cost functions") of interest. There are four main types of Hamiltonians that we consider in this paper.

The first two types of Hamiltonians we study are of interest because they are programmable instances of optimization problems that one might encounter in practical situations. The second two types of problems we study are of interest more to those who study statistical physics and for different reasons: because they define ensembles of instances for which the average case has known and interesting properties. While solutions to specific instances of the latter two problems are probably not of much value, we anticipate they are interesting problems on which to investigate the performance of a quantum computer. The four problems we study are described below.

1. ***L*-term spin model:** The most general Hamiltonian we consider is the one we refer to simply as the "L-term spin model." This Hamiltonian is a linear combination of L tensor products of Pauli-Z operators,

$$H_L = \sum_{\ell=1}^{L} w_\ell \prod_{i\in q_\ell} Z_i, \tag{3}$$

where w_ℓ are real scalars, Z_i is the Pauli-Z operator on qubit i, N is the number of qubits in the cost function, and q_ℓ is a unique set of up to N integers, which also take values between 1 and N (it is a set of integers corresponding to the indices of qubits on which term ℓ acts). One might anticipate that it is helpful to also specify this Hamiltonian in terms of its many-body order $k = \max|\{q_\ell\}|$. However, perhaps surprisingly, none of the algorithms discussed in this paper have a Toffoli complexity that scales explicitly in k.

2. **Quadratic unconstrained binary optimization:** We also consider an NP-Hard example of $H_{N^2/2}$ known as QUBO. The QUBO Hamiltonian is expressed as

$$\begin{aligned} H_{\text{QUBO}} &= \sum_{i\leq j} w_{ij}\left(\frac{\mathbb{1}-Z_i}{2}\right)\left(\frac{\mathbb{1}-Z_j}{2}\right) \\ &= \sum_{i<j} J_{ij} Z_i Z_j + \sum_i h_i Z_i + K, \end{aligned} \tag{4}$$

where K is a constant term that we ignore from this point forward as this never needs to be explicitly simulated or computed for the purposes of optimizing the model, and the coefficients J_{ij} and h_i can be computed from the w_{ij}. This form of the model is also known as the Ising model but we refer to it here as QUBO since the Ising model can also mean a model with more limited connectivity and regular coefficients in some contexts.

3. **Sherrington-Kirkpatrick:** This problem corresponds to a widely studied model of spin-glass physics [24]. The SK model is an example of the following QUBO Hamiltonian:

$$\begin{aligned} H_{\text{SK}} &= \sum_{i<j} w_{ij} Z_i Z_j, \quad w_{ij} \in \{-1, 1\}, \\ \|H_{\text{SK}}\| &\leq N^2/2, \end{aligned} \tag{5}$$

and the values of w_{ij} are usually chosen at random. The SK model is the focus of many studies on heuristic optimization, especially ones focusing on variants of simulated annealing. There is also a variant of the SK model, which has the same statistical properties where the coefficients are Gaussian distributed real numbers.

4. **Low autocorrelation binary sequences:** We think it is interesting to use a quantum computer to attempt to optimize problems that are very challenging on average. One problem is the LABS problem,

also known as the Bernasconi model in physics [33]:

$$H_{\mathrm{LABS}} = \sum_{k=0}^{N-1} H_k^2 \quad H_k$$
$$= \sum_{i=1}^{N-k} Z_i Z_{i+k}, \quad \|H_{\mathrm{LABS}}\| \approx N^3/3, \qquad (6)$$

which is an instance of H_{N^3}. This model is known to be extremely difficult; in fact the best classical algorithm has scaling like $\Theta(1.73^N)$ and has only been run for problem sizes up to $N = 66$ [34]. However, we note that the model is not really a "problem" in the usual computer science sense because there is only one instance defined for each problem size. A variant of the LABS problem that we consider is when the squared operators are instead replaced with absolute values, as one can verify that the ordering of the low-energy solutions are unchanged by this modification, and it is sometimes less expensive to simulate with a quantum computer.

The remainder of Sec. II discusses concrete circuit realizations for "oracles," which provide information about these cost functions of interest. Here we slightly abuse the term "oracle" to mean a circuit primitive, which is repeatedly queried throughout an algorithm, usually revealing information about the problem we are solving. These oracles are used by multiple algorithms throughout our paper. In Sec. II A, we explain how to implement cost function oracles that are required to return the cost of a specific candidate solution x. We refer to such oracles as "direct-energy oracles." In Sec. II B, we explain how to implement cost function oracles that are required to return the difference in cost between two candidate solutions that differ by exactly one bit. In Sec. III C, we explain how to implement cost function oracles that are required to return the cost function as a phase, rather than as a value written to a separate quantum register. In Sec. III D, we explain how to implement cost function oracles that are required to implement the cost function as a direct application of the Hamiltonian onto a target quantum register. Finally, in Sec. II E, we consider the cost of evaluating functions whose input is the difference in cost of candidate solutions as described in the other parts of this section.

We summarize the content of this section using three tables. In Table III we give a list of the symbols we use for reporting our computational complexity results. This table aids in the interpretation of the following two tables. In Table IV, we summarize the definitions of the various different kinds of oracles considered in this section. Finally, in Table V, we summarize the complexities of each of the 16 cost function oracles (four types of oracles for each of four types of cost functions) as well as the complexity of calculating functions of those oracle outputs. In these tables, and throughout the paper, we use log to indicate logarithms base 2.

A. Oracles for direct cost function evaluation

Many of the algorithms considered in this work are formulated in terms of a query to an oracle, which computes the value of the cost function C (for instance, one of the Hamiltonians discussed above) in a binary register. For instance, if we have a wavefunction $|\psi\rangle = \sum_x \psi_x |x\rangle$ where the computational basis states $|x\rangle$ are eigenstates of C such that $C|x\rangle = E_x |x\rangle$ then we define the direct-energy evaluation oracle O^{direct} as a circuit, which acts as

$$O^{\mathrm{direct}} \sum_x \psi_x |x\rangle |0\rangle^{\otimes b_{\mathrm{dir}}} \mapsto \sum_x \psi_x |x\rangle |\tilde{E}_x\rangle , \qquad (7)$$

where $\tilde{E}_x$ is a binary approximation to E_x using b_{dir} bits. We provide some strategies for how to realize this oracle for specific problems with low Toffoli complexity. We refer to the Toffoli complexity of this oracle as $\mathcal{C}^{\mathrm{direct}}$. However, first we discuss an efficient method for performing reversible in-place addition of a constant. This routine is critical to our implementation.

TABLE III. A list of common symbols we use throughout this paper.

symbol	meaning
x	bit string corresponding to a candidate solution of the optimization problem
N	number of bits needed to specify a candidate solution
E_x	cost (a.k.a. energy) of candidate solution x as specified by a cost function
H_{cf}	Hamiltonian operator corresponding to a cost function "cf"
b	number of bits used to specify the precision of an oracle
L	number of terms in a spin model (type of cost function)
λ	the normalization parameter for LCU methods, related to the Hamiltonian 1-norm
β	inverse temperature in simulated annealing
$\mathcal{C}$	Toffoli or T cost of some oracle
$\mathcal{A}$	ancilla required to implement some oracle that must be kept
$\mathcal{B}$	temporary ancilla required to implement some oracle

TABLE IV. Quick definitions of the most important "oracle" circuits discussed in this work. Here, we slightly abuse the term "oracle" to mean a circuit primitive, which is repeatedly queried throughout an algorithm, usually revealing information about the problem we are solving. Throughout the paper we use $\mathcal{C}$ to denote Toffoli (or occasionally T) complexity while $\mathcal{A}$ and $\mathcal{B}$ denote persistent and temporary ancilla costs, respectively. For some of these oracles there are different Toffoli costs when performing them in the forward and reverse directions. We always pair a forward oracle with a reverse oracle, so give the average cost. In some cases the computation may introduce ancilla qubits not shown here, that are erased in the inverse computation. For the function evaluation oracle we incorporate multiplication by the inverse temperature β. The approximation $\tilde{f}$ is given to $b_{\rm sm}$ bits, but for generality we allow an error $2^{-b_{\rm fun}}$, which may be larger than $2^{-b_{\rm sm}}$.

Oracle	Oracle definition	Precision definition
$O^{\rm direct}$	$O^{\rm direct}\sum_x \psi_x \lvert x\rangle \lvert 0\rangle^{\otimes b_{\rm dir}} \mapsto \sum_x \psi_x \lvert x\rangle \lvert \tilde{E}_x\rangle$	$\left\lvert E_x - \tilde{E}_x\right\rvert \le 2^{-b_{\rm dir}} \max_x \lvert E_x\rvert$
$O_k^{\rm diff}$	$O_k^{\rm diff}\sum_x \psi_x \lvert x\rangle \lvert 0\rangle^{\otimes b_{\rm dif}} \mapsto \sum_x \psi_x \lvert x\rangle \lvert \widetilde{\delta E}_x^{(k)}\rangle, \quad \lvert y\rangle = X_k \lvert x\rangle$	$\left\lvert \widetilde{\delta E}_x^{(k)} - E_x + E_y\right\rvert \le 2^{-b_{\rm dif}} \max_{x,y} \lvert E_x - E_y\rvert$
$O^{\rm phase}(\gamma)$	$O^{\rm phase}(\gamma)\sum_x \psi_x \lvert x\rangle \mapsto \sum_x e^{-i\widetilde{\gamma E}_x}\psi_x \lvert x\rangle$	$\left\lvert \widetilde{\gamma E}_x - \gamma E_x\right\rvert \le 2^{-b_{\rm pha}}$
$O^{\rm LCU}$	$\langle 0\rvert^{\otimes \log L} O^{\rm LCU} \lvert 0\rangle^{\otimes \log L} = \tilde{H}/\lambda, \quad \tilde{H} = \sum_{\ell=1}^{L} \tilde{w}_\ell U_\ell, \quad \lambda = \sum_{\ell=1}^{L} \lvert w_\ell\rvert$	$\left\lvert \sqrt{w_\ell} - \sqrt{\tilde{w}_\ell}\right\rvert \le 2^{-b_{\rm LCU}}$
$O_\beta^{\rm fun}$	$O_\beta^{\rm fun} \lvert z\rangle \lvert 0\rangle^{\otimes b_{\rm sm}} \mapsto \lvert x\rangle \lvert \tilde{f}\,(\beta z)\rangle$	$\left\lvert f\,(\beta z) - \tilde{f}\,(\beta z)\right\rvert \le 2^{-b_{\rm fun}}$

1. Direct-energy oracle for L-term spin model and QUBO

We now explain how to implement the direct-energy oracle for the H_L Hamiltonian with low Toffoli complexity. We represent the energy $\tilde{E}_x$ in the two's-complement binary representation, as this encoding enables efficient methods for addition [35]. In the two's-complement positive integers have a normal binary representation whereas negative integers are the complement of that representation minus one. For instance, in 4-bit two's complement $3_{10} = 0011_2$ whereas $-3_{10} = 1100_2 + 1 = 1101_2$. Zero still corresponds to all bits zero. The fact that we need to add one for negative numbers complicates our approach but this representation is still preferable for our purposes.

The main idea behind our approach is to add or subtract the value of each term's coefficient w_ℓ to a b-bit output register based on the parity of the string $\prod_{i\in q_\ell} Z_i$. To perform addition or subtraction controlled on a qubit, we use the fact that one can switch between addition and subtraction by applying NOT gates to the target register in two's complement representation. That is, applying NOT gates to all qubits of a register change $\lvert v\rangle$ to $\lvert -v-1\rangle$. Adding w to this register gives $\lvert w-v-1\rangle$, then applying NOT gates to all qubits again yields $\lvert v-w\rangle$. To perform addition or subtraction controlled on a qubit, one can use the procedure shown in Fig. 4(a) of Ref. [35] (see Appendix D 2). The complete procedure to compute the energy is then as given in Algorithm 1.

After performing this for L terms one can verify that this produces the intended state $\lvert v\rangle = \lvert \tilde{E}_x\rangle$ in the output register. Toffoli gates enter only through the adder in step 3. Thus, in total our approach has Toffoli complexity $\mathcal{C}_L^{\rm direct}$ and ancilla requirements $\mathcal{A}_L^{\rm direct}, \mathcal{B}_L^{\rm direct}$ given by

$$\mathcal{C}_L^{\rm direct} = L(b_{\rm dir} - 2) < L b_{\rm dir}, \tag{8}$$

$$\mathcal{A}_L^{\rm direct} = b_{\rm dir}, \tag{9}$$

$$\mathcal{B}_L^{\rm direct} = b_{\rm dir} - 1 < b_{\rm dir}, \tag{10}$$

where the ancilla refer to the carry bits for the adder in addition to the $b_{\rm dir}$ bits required to output the energy. We note that for this oracle these costs have no dependence on the many-body order of the Hamiltonian H_L since this only affects the number of CNOT gates used to compute the parity of the terms.

This exact same reasoning can be used to determine the complexity of computing the energies for the QUBO Hamiltonian. Due to the relative lack of structure in QUBO, there is no obvious way to improve over this general complexity. There we have $L = N(N+1)/2$ terms and so from Eqs. (8)–(10) we require a number of Toffolis and ancillas equal to

$$\mathcal{C}_{\rm QUBO}^{\rm direct} = \frac{N^2 b_{\rm dir}}{2} + \frac{N b_{\rm dir}}{2} - N(N+1)$$
$$= \frac{N^2 b_{\rm dir}}{2} + \mathcal{O}(N b_{\rm dir}), \tag{11}$$

$$\mathcal{A}_{\rm QUBO}^{\rm direct} = b_{\rm dir}, \tag{12}$$

$$\mathcal{B}_{\rm QUBO}^{\rm direct} = b_{\rm dir} - 1 < b_{\rm dir}. \tag{13}$$

2. Direct-energy oracle for the SK model

Here we show that the energy for the SK model can be computed with only N^2 Toffolis and a logarithmic number of ancillas. The method we use is a sum of tree sums of bits. It is also possible to just use a tree sum with a Toffoli cost of about $N^2/4$, but the drawback is that this method needs $N^2/2$ ancilla qubits, which is prohibitive.

For the SK model it is convenient to replace -1 with 0, so the sum takes values between 0 and L. That corresponds to dividing the Hamiltonian by 2 and shifting it, which does not change the optimization problem, but means we are only summing bits. If we were to sum the bits in

TABLE V. Summary of complexities for realizing oracles used throughout this paper. Next to the complexity entry is a number indicating the equation in the paper, which gives the full expression in context. The energy difference for H_L and LABS just has twice the Toffoli cost and the same ancilla cost as the direct-energy oracle, because it is found by evaluating the energy twice. These oracles and the meaning of their precision parameters b are defined in Table IV. The Toffoli count is reported except when the oracle type for that cost function is marked with (*), which indicates that T count is reported instead. Here we include only the main terms in the order expressions. We use these costings to determine the complexities in Table VIII.

Cost function	Oracle type	Toffoli (*or T) gate count $\mathcal{C}$	persistent ancilla $\mathcal{A}$	temporary ancilla $\mathcal{B}$
L-term spin model	direct energy	Lb_{dir} (8)	b_{dir} (9)	$b_{\text{dir}}-1$ (10)
H_L	energy difference	$2Lb_{\text{dif}}+\mathcal{O}(1)$ (8)	b_{dif} (9)	$b_{\text{dif}}-1$ (10)
	direct phase*	$1.15L(b_{\text{pha}}+\log L)+\mathcal{O}(\log L)$ (36)	0 (37)	1 (38)
	Hamiltonian walk	$3L+2b_{\text{LCU}}+\mathcal{O}(\log L)$ (56)	$2\log L+2b_{\text{LCU}}+\mathcal{O}(1)$ (57)	$\log L+\mathcal{O}(1)$ (58)
Quadratic	direct energy	$N^2b_{\text{dir}}/2+\mathcal{O}(Nb_{\text{dir}})$ (11)	b_{dir} (12)	$b_{\text{dir}}-1$ (13)
Unconstrained	energy difference	Nb_{dif} (25)	b_{dif} (26)	$b_{\text{dif}}-1$ (27)
Binary optimization	direct phase*	$0.575N^2(b_{\text{pha}}+2\log N)+\mathcal{O}(N^2)$ (39)	0 (40)	1 (41)
H_{QUBO}	Hamiltonian walk	$N(b_{\text{LCU}}+2\log N)+\mathcal{O}(N)$ (73)	$2b_{\text{LCU}}+4\log N+\mathcal{O}(1)$ (75)	$3\log N+\mathcal{O}(\log b_{\text{LCU}})$ (76)
Sherrington-	direct energy	N^2 (16)	$2\log N$ (17)	$4\log N$ (18)
Kirkpatrick model	energy difference	$2N$ (28)	$\log N+1$ (29)	$2\log N+\mathcal{O}(1)$ (30)
H_{SK}	direct phase	$2N^2+{b_{\text{pha}}}^2/2+\mathcal{O}(b_{\text{pha}}\log b_{\text{pha}})$ (42)	$2\log N+b_{\text{pha}}+\mathcal{O}(\log b_{\text{pha}})$ (43)	$4\log N$ (44)
	Hamiltonian walk	$6N+\mathcal{O}(\log^2 N)$ (82)	$2\log N+\mathcal{O}(1)$ (83)	$3\log N+\mathcal{O}(1)$ (84)
Low	direct energy	$5N(N+1)/4$ (20)	$2\log N+1$ (21)	$3\log N+3$ (22)
Autocorrelation	energy difference	$5N(N+1)/2$ (20)	$2\log N+1$ (21)	$3\log N+3$ (22)
Binary sequences	direct phase	$\frac{8}{5}N^2+\min\left(\frac{1}{2}N{b_{\text{pha}}}^2,\frac{9}{10}N^2\right)+\mathcal{O}(Nb_{\text{pha}}\log b_{\text{pha}})$ (49)	$b_{\text{pha}}+\mathcal{O}(\log b_{\text{pha}})$ (46)	$5\log N+\mathcal{O}(\log b_{\text{pha}})$ (50)
H_{LABS}	Hamiltonian walk	$4N+\mathcal{O}(\log N)$ (87)	$3\log N+\mathcal{O}(1)$ (88)	$2\log N+\mathcal{O}(1)$ (89)
	function evaluation	${b_{\text{sm}}}^2+b_{\text{dif}}+\mathcal{O}(b_{\text{sm}}\log b_{\text{sm}}+2^{b_{\text{fun}}/2})$ (95)	$2b_{\text{sm}}+\mathcal{O}(\log b_{\text{sm}})$ (97)	$b_{\text{dif}}-1$ (98)
	arcsine evaluation	$(b_{\text{sm}}+b_{\text{fun}})^2+b_{\text{dif}}+\mathcal{O}(b_{\text{sm}}\log b_{\text{sm}}+2^{b_{\text{fun}}/2})$ (96)	$2b_{\text{sm}}+b_{\text{fun}}+\mathcal{O}(\log b_{\text{sm}})$ (99)	$b_{\text{dif}}-1$ (100)

```
Require: A quantum state $\sum_x a_x |x\rangle$, a vector of weights $\{w_\ell\}$ that specifies the $L$-term spin model or QUBO Hamiltonian.
Ensure: An output state of the form $\sum_x a_x |x\rangle |\tilde{E}_x\rangle$, where $H$ is the relevant Hamiltonian and $\tilde{E}_x$ is the approximate eigenvalue of $H$ corresponding to $|x\rangle$.
Use Clifford gates (CNOT gates) to compute the parity of the term $\prod_{i\in q_\ell} Z_i$ in-place in a single system qubit $|\pi_\ell\rangle$. Specifically, if $x_i$ is the $i^{\rm th}$ bit of computational basis state $x$ then we are using CNOTs to compute $\pi_\ell = (\sum_{i\in q_\ell} x_i) \bmod 2$.
Controlled on $|\pi_\ell\rangle$, use more CNOT gates to negate every bit of the output register. We will refer to this output register as $|v\rangle$. Thus, after this step we will have the state $|0\rangle |v\rangle$ if the first bit holds $\pi_\ell = 0$ and we will have the state $|1\rangle |-v-1\rangle$ if the first bit holds $\pi_\ell = 1$.
Using the strategy described in Appendix D 2 for the addition of a constant, add a $b_{\rm dir}$-bit binary approximation $\tilde{w}_\ell$ to the coefficient $w_\ell$ into the output register. This step has Toffoli complexity $b_{\rm dir} - 2$ where $b_{\rm dir}$ is the size of the output register. After this step we will have the state $|0\rangle |v + \tilde{w}_\ell\rangle$ if $\pi_\ell = 0$ and we will have the state $|1\rangle |\tilde{w}_\ell - v - 1\rangle$ if $\pi_\ell = 1$.
Negate the output register using CNOT gates, controlled on $|\pi_\ell\rangle$. After this step we will have the state $|0\rangle |v + \tilde{w}_\ell\rangle$ if $\pi_\ell = 0$ and we will have the state $|1\rangle |v - \tilde{w}_\ell\rangle$ if $\pi_\ell = 1$.
Using Clifford gates uncompute the parity $\pi_\ell$.
```

Algorithm 1. Energy evaluation for the L-term spin model and QUBO.

the obvious way, the Toffoli complexity is approximately $N^2 \log N$. However, we can take advantage of the fact we are summing bits to reduce the complexity to $\mathcal{O}(N^2)$.

Our methods are based on tree sums of bits. In Ref. [25] it was shown that it is possible to sum L bits using $L-1$ Toffoli gates and $L-1$ ancilla qubits, and this sum can be uncomputed with no Toffoli cost. As discussed in Ref. [25], it is also possible to perform sums in approaches that reduce the number of ancilla at the price of increasing the number of Toffoli gates. In particular, we can subdivide the bits we are summing into about $L/\log L$ groups of size $\log L$, start by using the tree sum approach to sum each of the groups, add it into a running sum, and uncompute it. The number of ancillas needed is reduced to approximately $\log L$ for each of the tree sums. There is also a cost of approximately L for adding the tree sums, giving a total complexity of approximately $2L$.

To be more specific, taking into account that L need not be a power of 2, we can use $M = \lceil L/\lceil \log L \rceil \rceil - 1$ groups of size $\lceil \log L \rceil$, except for a remaining group of size $J \le \lceil \log L \rceil$ such that $M\lceil \log L \rceil + J = L$. That is, there are $\lceil L/\lceil \log L \rceil \rceil$ groups, and J can be smaller than $\lceil \log L \rceil$. The Toffoli cost of computing each of these sums is

$$M\lceil \log L \rceil - M + J - 1 = L - M - 1 = L - \lceil L/\lceil \log L \rceil \rceil. \tag{14}$$

The cost of the additions is

$$\begin{aligned}\sum_{j=1}^{M} &[\lceil \log(J + j\lceil \log L \rceil + 1)\rceil - 1] \\ &\le M[\lceil \log(L+1)\rceil - 1] \\ &\le M\lceil \log L \rceil \\ &< (L/\lceil \log L \rceil)\lceil \log L \rceil = L.\end{aligned} \tag{15}$$

We assume that $L > 1$ and hence $\log L > 0$. The first line of Eq. (15) comes from starting with the sum over J bits and then adding each of the sums over $\lceil \log L \rceil$ to it. After adding j of the sums over $\lceil \log L \rceil$ bits, the maximum value of the sum is $J + j\lceil \log L \rceil$, so the number of bits needed to store the result is $\lceil \log(J + j\lceil \log L \rceil + 1)\rceil$, and the number of Toffolis needed for that sum is one less than that. The inequality in the first line comes from the fact that the total number is never less than L, so the cost of the additions is never greater than $\lceil \log(L+1)\rceil - 1$. The inequality in the second line is because $\lceil \log(L+1)\rceil - 1 \le \log L$. The inequality in the third line is using $M < L/\lceil \log L \rceil$.

Therefore, the total Toffoli cost is less than $2L$. The ancilla cost of each tree sum is $\lceil \log L \rceil - 1$, there are $\lceil \log(L+1)\rceil$ ancilla needed for the total, and $\lceil \log(L+1)\rceil - 1$ temporary ancillas for the addition of the tree sum into the total. Since the ancillas in the tree sum are uncomputed, they contribute to an overall temporary ancilla cost, meaning the temporary ancilla cost is $2\log L + \mathcal{O}(1)$ and the persistent ancilla cost (for the total) is $\log L + \mathcal{O}(1)$.

Since $L = N(N-1)/2$, if we use a tree sum the cost is less than $N^2/2$, but the ancilla cost is approximately $N^2/2$. The sum could be uncomputed without ancillas, giving an average (compute and uncompute) cost of $N^2/4$. We expect that the tradeoff is not worth it in this case. However, by using the sum of tree sums, we get a Toffoli cost less than N^2, and an ancilla cost that is logarithmic in N. That gives costs for the SK model of

$$\mathcal{C}_{\rm SK}^{\rm direct} < N^2, \tag{16}$$

$$\mathcal{A}_{\rm SK}^{\rm direct} \le 2\log N, \tag{17}$$

$$\mathcal{B}_{\rm SK}^{\rm direct} < 4\log N. \tag{18}$$

3. Direct-energy oracle for LABS model

Next we show that for the LABS problem it is possible to compute the energy with a Toffoli cost of $5N(N+1)/4$ for $N \ge 64$, with a logarithmic number of ancilla qubits. We improve over the application of our general technique by specializing the implementation to the LABS problem.

Since the LABS problem has $L = \mathcal{O}(N^3)$ with maximum integer energy values of $\mathcal{O}(N^3)$, we expect a complexity of $\mathcal{O}(N^3)$. Instead, we show that it is possible to perform the direct-energy evaluation at cost $\mathcal{O}(N^2)$. We focus on the form of the LABS Hamiltonian that is expressed as $\sum_{k=1}^{N} |H_k|$, where H_k is as defined in Eq. (6) (as we mention, this form of the problem has the same ordering of the low-energy landscape).

In the following we use E_k to denote the eigenvalue of H_k. It is most efficient to use the sum of the tree sums approach described above. Here we need to find E_k by using $+1$ and -1 rather than $+1$ and 0, because we need to take the absolute value, so we need an extra bit for the sign. Therefore, after summing bits, we need to multiply by 2 (which has no Toffoli cost), followed by subtracting the number of bits. The overall approach is then as follows. We sum k starting at $k = N - 1$ and go down to zero, so the number of bits at each step is minimized. For each value of k we perform Algorithm 2.

In step 1, the Toffoli complexity computing each E_k is approximately $2(N-k)$ plus the cost of subtracting $N-k$. In two's complement we can determine whether the number is negative or positive by looking at the highest bit; if the highest bit is 1 then we know the value is negative. This justifies the operations in step 2. Since step 2 requires no non-Clifford operations, it can be neglected in our cost analysis. In step 3, the state is $|u\rangle\,|u+v\rangle$ if $u \geq 0$ or $|u\rangle\,|v-u\rangle$ if $u < 0$; equivalently we now have the state $|u\rangle\,|v+|u|\rangle$. The output register is of size $\lceil \log[(N-k)(N-k+1)/2+1] \rceil + 1$ so the Toffoli cost is $\lceil \log[(N-k)(N-k+1)/2+1] \rceil$.

The output register is significantly larger than the scratch register. However, with a slight modification of the procedure in Appendix D 2 we can allow this register to be smaller with no additional Toffoli cost. First, consider expanding the number of qubits $|u\rangle$ is encoded on. This is of course trivial for positive numbers. For negative u, for n bits it is encoded as $2^n + u$. Therefore, if we have a number that is negative and we need to map it to a negative number on some larger number of bits n', then we need to map $2^n + u$ to $2^{n'} + u$, which means adding $2^{n'} - 2^n = \sum_{j=n}^{n'-1} 2^j$. This means that bits $n+1$ to n' of the negative number encoded on the n' bits need to be ones. These can be set by using CNOTs controlled by bit n, which means no additional Toffoli cost is needed to encode the number into more qubits. A further simplification can be used to eliminate the need for those extra qubits. First, rearrange the addition circuit as in Fig. 18 so that the qubits of $|u\rangle$ are only used as controls and not changed. Since all of the additional qubits for $|u\rangle$ contain the same value as the sign qubit of $|u\rangle$, we may use that sign qubit as the control instead of any of those additional qubits. Then the additional qubits are not used, and can be omitted.

There is an improvement that we can make when we take into account that each computation needs to be paired with an uncomputation. This is because, in step 5, if we are computing an energy that we later uncompute, then we can use the strategy of Ref. [35] to erase $|u\rangle$ using X measurements and no Toffoli cost. A phase correction is required, but that can be done when we later uncompute the LABS energy. This means that in step 5 we have a cost of $N-k$ in uncomputing the LABS energy, but no Toffoli cost in computing the LABS energy. Because each computation is paired with an uncomputation, it is therefore convenient to give the average complexity of $N-k$. The largest temporary ancilla cost is when we need to uncompute the overall Hamiltonian, when it is $2\log(N-k) + \mathcal{O}(1)$. That is still less than the temporary ancilla cost in step 3, so can be ignored.

After repeating this for the N values of k one can verify that the output register contains the energy of the LABS Hamiltonian. Toffoli gates enter only through steps 1, 3, and 5. The primary contribution to the complexity is the computation of E_k in steps 1 and 5. Ignoring the complexity of subtracting $N-k$, the Toffoli complexity is

$$\sum_{k=0}^{N-1} 3(N-k) = 3N(N+1)/2. \qquad (19)$$

The cost of the subtractions as well as the additions in step 3 increases the cost, but also $2(N-k)$ is an overestimate of the cost of adding $n-k$ bits. In particular, we can use tree sums of as many as approximately $\log N$ bits, rather than just $\log(N-k)$, with no penalty in terms of the

```
Require: A quantum state $\sum_x a_x |x\rangle$, the set of all terms in the LABS Hamiltonian $\{H_k\}$.
Ensure: An output state of the form $\sum_x a_x |x\rangle |E_x\rangle$.
1: Compute for computational basis vector $|x\rangle$ the value of $E_k$ in a scratch register $|u\rangle$ that will require $\lceil \log(N-k+1) \rceil + 1$
   ancilla to store (with +1 for the sign).
2: Controlled by the highest bit of $u$ (the sign bit in two's complement), use CNOT gates to negate the value of the output
   register $|v\rangle$. At this point we have $|u\rangle |v\rangle$ if $u \geq 0$ or $|u\rangle |-v-1\rangle$ if $u < 0$.
3: Add the scratch register into the output register.
4: Use CNOTs to negate the output register controlled on the highest bit of the $|u\rangle$ register.
5: Uncompute $|u\rangle$.
```

Algorithm 2. Energy evaluation for LABS model.

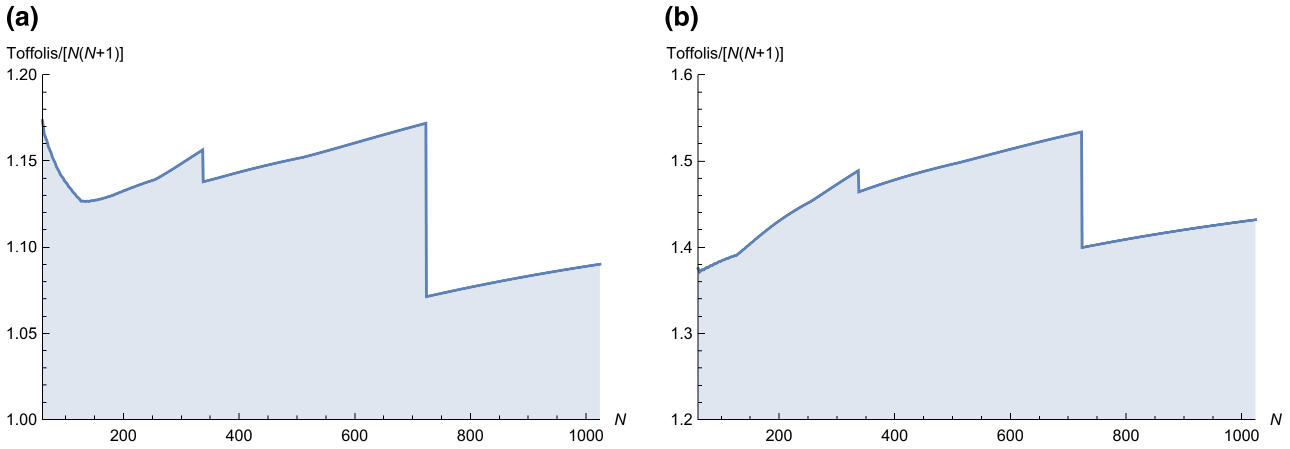

FIG. 1. Toffoli costs for direct evaluation of the LABS Hamiltonian by computing H_k with a sum of tree sums. The average cost when computing and uncomputing the Hamiltonian is shown in (a). The cost of just computing and uncomputing H_k (omitting the cost of summing the absolute values), when we just compute the Hamiltonian, is given in (b).

temporary ancilla cost. The computed costs are shown in Fig. 1(a), and it is found for the range of N we are interested in (64–1024), the constant factor on $N(N+1)$ is less than 1.2, rather than 1.5 (in fact, this bound is good for $N \geq 45$). In particular, the constant factors for $N = 64$, 128, 256, and 1024 are 1.16466, 1.12673, 1.13945, and 1.0901, respectively. To simplify the expressions we give the slightly looser bound in the table

$$\mathcal{C}_{\text{LABS}}^{\text{direct}} < 5N(N+1)/4, \tag{20}$$

with the caveat that it is for $N \geq 45$. The number of ancilla we require is

$$\mathcal{A}_{\text{LABS}}^{\text{direct}} = \lceil \log[N(N+1)/2+1] \rceil \leq 2\log N + 1, \tag{21}$$

$$\mathcal{B}_{\text{LABS}}^{\text{direct}} = \lceil \log[N(N+1)/2+1] \rceil + \lceil \log(N-k+1) \rceil + 2 \leq 3\log N + 3. \tag{22}$$

The persistent ancilla are for the output value. Approximately $2\log N$ of the temporary ancilla are for carry bits in the addition and $\log N$ are for the scratch register. We assume $N > 1$ for the inequalities, which omits the trivial case. This example illustrates how taking advantage of problem structure can lead to advantages over the implementation of an oracle intended to handle a more general case.

B. Energy-difference oracles

For some of the algorithms discussed in this work (specifically the quantum versions of simulated annealing) we often need the direct-energy oracle only as means to compute a difference between the energies of two different states, which differ in only one bit. The ultimate objective in that context is a circuit that performs the mapping

$$O_k^{\text{diff}} \sum_x \psi_x |x\rangle |0\rangle^{\otimes b_{\text{dif}}} \mapsto \sum_x \psi_x |x\rangle |\widetilde{\delta E}_x^{(k)}\rangle, \quad \delta E_x^{(k)} = E_x - E_y, \quad |y\rangle = X_k |x\rangle, \tag{23}$$

where (as usual) X_k is the NOT operation on qubit k and $\widetilde{\delta E}_x^{(k)}$ is a binary approximation to $\delta E_x^{(k)}$ using b_{dif} bits. Especially when the many-body order is 2-local, it is more efficient to consider a specialized implementation of O_k^{diff} than to try to realize this operation using one call to O^{direct} and one call to $O^{\text{direct}} X_k$.

First, we discuss the energy-difference oracle for QUBOs. In this case, $\delta E_x^{(k)}$ is the eigenvalue of the operator

$$\delta H^{(k)} = 2h_k Z_k + 2\sum_{i \neq k} J_{ik} Z_i Z_k. \tag{24}$$

We see that $\delta H^{(k)}$ is itself a simple cost function, which is an example of H_N (the L-term spin model with $L = N$). Thus, to compute the eigenvalue of this operator (equivalent to implementing O_k^{diff}) we require

$$\mathcal{C}_{\text{QUBO}}^{\text{diff}} = N(b_{\text{dif}} - 2) < N b_{\text{dif}}, \tag{25}$$

$$\mathcal{A}_{\text{QUBO}}^{\text{diff}} = b_{\text{dif}}, \tag{26}$$

$$\mathcal{B}_{\text{QUBO}}^{\text{diff}} = b_{\text{dif}} - 1. \tag{27}$$

This scaling is much less than the $N^2 b_{\text{dif}} + \mathcal{O}(N b_{\text{dif}})$ Toffoli gates that are required by making two queries to the direct-energy oracle for QUBO.

For the SK model we can simplify the QUBO result. We then have the difference operator $2\sum_{i \neq k} w_{ik} Z_i Z_k$, so we just need to sum $N-1$ bits, and can take $b_{\rm dif} = \lceil \log N \rceil$. We also need to subtract $N-1$ from the bit sum to obtain the energy difference, but the cost of that subtraction plus the cost of the bit sum is still no more than the upper bound of $2N$ we gave previously on the cost of the bit sum. Therefore, the energy-difference oracle has cost

$$\mathcal{C}_{\rm SK}^{\rm diff} < 2N, \tag{28}$$

$$\mathcal{A}_{\rm SK}^{\rm diff} = \lceil \log N \rceil \leq \log N + 1, \tag{29}$$

$$\mathcal{B}_{\rm SK}^{\rm diff} \leq 2\log N + \mathcal{O}(1). \tag{30}$$

For higher many-body-order Hamiltonians like LABS or the H_L model of many-body order greater than 2, the best strategy probably involves two applications of the direct-energy oracle $O^{\rm direct}$. However, rather than actually use two registers to output the energy and then perform subtraction one can instead just compute the energy of x first and then in the same register compute the energy of y while subtracting all of the terms instead of adding them. There is a slightly greater Toffoli cost because the subtraction is on a slightly larger number of qubits, but that cost is small enough to be ignored. This leads to Toffoli complexity of $2\mathcal{C}^{\rm direct}$ but requires no additional ancilla.

C. Oracles for phasing by cost function

In some contexts our goal is to phase each computational state on which the wavefunction has support by an amount proportional to the energy of that computational basis state (this task is equivalent to performing evolution under a diagonal Hamiltonian for unit time). We refer to circuits that achieve this task as a “phase” oracle and define them to act as

$$O^{\rm phase}(\gamma)\sum_x \psi_x |x\rangle \mapsto \sum_x e^{-i\widetilde{\gamma E_x}}\psi_x|x\rangle \qquad \left|\widetilde{\gamma E_x} - \gamma E_x\right| \leq 2^{-b_{\rm pha}}. \tag{31}$$

To simplify the following discussion, we assume that E_x is shifted such that it is non-negative. Such a shift corresponds to an unobservable global phase.

To realize this oracle, one strategy is to first approximately compute E_x into a register using $O^{\rm direct}$, then multiply by γ, and perform further logic to phase the system by the amount in the register. For instance,

$$\left(O^{\rm direct}\right)^\dagger \left[\mathbb{1}\otimes U^{\rm phase}(\gamma)\right] O^{\rm direct}\sum_x \psi_x|x\rangle \mapsto \sum_x e^{-i\widetilde{\gamma E_x}}\psi_x|x\rangle \tag{32}$$

where the phasing operation needed is

$$U^{\rm phase}(\gamma) = \sum_{k=0}^{2^{b_{\rm dir}}-1} \exp\left(\frac{2\pi i k\tilde{\gamma}}{2^{b_{\rm dir}}}\right) k. \tag{33}$$

The value of $2\pi k\tilde{\gamma}/2^{b_{\rm dir}}$ corresponds to the approximation of γE_x, with k the integer approximating E_x (so $k \approx 2^{b_{\rm dir}} E_x/E_{\max}$) and $\tilde{\gamma} = \gamma E_{\max}/(2\pi)$ is a scaled form of γ. We limit ourselves to simulations where the phase factor is no more than a factor of 2π, so $\tilde{\gamma} \leq 1$. Using the “phase gradient” trick of Refs. [35,36], it is possible to apply a phase by adding into a reusable ancilla register initialized to the state

$$|\phi\rangle = \frac{1}{\sqrt{2^{b_{\rm grad}}}}\sum_{\ell=0}^{2^{b_{\rm grad}}-1} e^{-2\pi i\ell/2^{b_{\rm grad}}}|\ell\rangle. \tag{34}$$

Here we use $b_{\rm grad}$ rather than $b_{\rm dir}$ in this state to allow for needing to use more bits to obtain the required precision in the phase. For details see Appendix A. In this case we need to multiply by the classically specified number $\tilde{\gamma}$ to obtain the required phase. This number can be given by $\log\tilde{\gamma} + b_{\rm pha} + \mathcal{O}(1)$ digits in order to obtain error $< 2^{-b_{\rm pha}}$. There is error due to the finite number of digits for E_x, the finite number of bits for $\tilde{\gamma}$, and the multiplication.

Rather than performing the multiplication by $\tilde{\gamma}$, adding into the phase-gradient state, then uncomputing the multiplication, a more efficient method is to perform the multiplication by repeated addition into the phase-gradient state. For each nonzero bit of $\tilde{\gamma}$, we can add a bit-shifted copy of k into the phase gradient state. Each addition into the phase-gradient state has cost $b_{\rm grad} - 2$, and on average approximately half the bits of $\tilde{\gamma}$ are zero, giving cost roughly $b_{\rm grad}(\log\tilde{\gamma} + b_{\rm pha})/2$. To address cases where more bits of $\tilde{\gamma}$ are nonzero, we can write $\tilde{\gamma}$ as a sum of powers of 2 with plus and minus signs. In that case it is possible to use no more than $(\log\tilde{\gamma} + b_{\rm pha})/2 + \mathcal{O}(1)$ additions, giving cost $b_{\rm grad}(\log\tilde{\gamma} + b_{\rm pha})/2 + \mathcal{O}(b_{\rm grad})$. The error due to omission of bits in the multiplication is no more than approximately $2^{-b_{\rm grad}}(\log\tilde{\gamma} + b_{\rm pha})\pi$, so to obtain error $< 2^{-b_{\rm pha}}$ one should take $b_{\rm grad} = b_{\rm pha} + \mathcal{O}(\log b_{\rm pha})$. That gives an overall cost for the multiplication

$$\frac{b_{\rm pha}(\log\tilde{\gamma} + b_{\rm pha})}{2} + \mathcal{O}(b_{\rm pha}\log b_{\rm pha}). \tag{35}$$

For more details see Appendix A. Note finally that the state $|\phi\rangle$ can be initialized prior to simulation and reused throughout, with a negligible additive one time cost scaling as $\mathcal{O}(b_{\rm grad}^2)$. This one time cost comes from synthesizing $b_{\rm grad}$ arbitrary rotations. However, since this is additive to the overall cost (whereas all other oracle costs are multiplicative with the number of queries), we expect this is negligible.

For the L-term spin Hamiltonians and QUBOs, the cost of the multiplication by γ can be eliminated by simply including it in the coefficients of the problem Hamiltonian. However for these cases an even more efficient approach is to simulate each term explicitly in a Trotter-like fashion and perform rotation synthesis to decompose each rotation into a sequence of T gates. In that case, one requires a number of T gates equal to the number of terms times the cost of rotation synthesis, which gives a complexity of $\mathcal{O}(L(b_{\text{pha}} + \log L))$. Using the repeat until success circuits of Ref. [37], this gives T gate and ancilla complexities of roughly

$$\begin{aligned}\mathcal{C}_L^{\text{phase}} &= 1.15L(b_{\text{pha}} + \log L) + 10.925L + \mathcal{O}(1)\\ &= 1.15L(b_{\text{pha}} + \log L) + \mathcal{O}(L), \qquad (36)\\ \mathcal{A}_L^{\text{phase}} &= 0, \qquad (37)\\ \mathcal{B}_L^{\text{phase}} &= 1. \qquad (38)\end{aligned}$$

There is a single temporary ancilla qubit used by the repeat until success circuits. The measure of error in Ref. [37] is the Frobenius distance $d(U,V) = \sqrt{1 - |\text{Tr}(UV^\dagger)|/2}$. A phase error of $2^{-b_{\text{pha}}}$ gives $|\text{Tr}(UV^\dagger)|/2 = |1 + \exp(2^{-b_{\text{pha}}}i)|/2 = \cos(2^{-b_{\text{pha}}}/2)$. Expanding in a series gives a Frobenius distance of $2^{-b_{\text{pha}}}/\sqrt{8} + \mathcal{O}(2^{-3b_{\text{pha}}})$. That means the cost becomes $1.15\left[b_{\text{pha}} + \log(\sqrt{8})\right] + 9.2 = 1.15b_{\text{pha}} + 10.925$, which is why the second term above is different than in Ref. [37]. Because Toffoli gates require roughly twice the resources to distill as T gates [38], this approach is likely to be more efficient in practice. This gives T and ancilla complexities for QUBO of

$$\begin{aligned}\mathcal{C}^{\text{phase}} &= 0.575N(N+1)\{b_{\text{pha}} + \log[N(N+1)]\}\\ &\quad + 4.9N(N+1) + \mathcal{O}(1)\\ &= 0.575N^2(b_{\text{pha}} + 2\log N) + \mathcal{O}(N^2), \qquad (39)\\ \mathcal{A}^{\text{phase}} &= 0, \qquad (40)\\ \mathcal{B}^{\text{phase}} &= 1, \qquad (41)\end{aligned}$$

assuming $N > b_{\text{pha}}$.

For the SK model it is better to compute the energy, add the energy into the phase gradient state, then uncompute the energy. That has Toffoli complexity $2N^2$, with $2\log N$ persistent ancillas and $4\log N$ temporary ancillas. The cost of the multiplication directly into the phase gradient state is ${b_{\text{pha}}}^2/2 + \mathcal{O}(b_{\text{pha}} \log b_{\text{pha}})$ (with $\bar{\gamma} \leq 1$), with b_{grad} permanent ancillas for the phase-gradient state and $b_{\text{grad}} - 1$ temporary ancillas for the addition. That gives costs for SK of

$$\begin{aligned}\mathcal{C}_{\text{SK}}^{\text{phase}} &= 2N^2 + {b_{\text{pha}}}^2/2 + \mathcal{O}(b_{\text{pha}} \log b_{\text{pha}}), \qquad (42)\\ \mathcal{A}_{\text{SK}}^{\text{phase}} &= 2\log N + b_{\text{pha}} + \mathcal{O}(\log b_{\text{pha}}), \qquad (43)\\ \mathcal{B}_{\text{SK}}^{\text{phase}} &= \max\left[4\log N, b_{\text{pha}} + \mathcal{O}(\log b_{\text{pha}})\right]. \qquad (44)\end{aligned}$$

For the parameters we consider for examples of gate counts, $4\log N \geq b_{\text{pha}}$, so we give that in Table V.

For the LABS model we still need to explicitly compute the partial sum for H_k and then take the absolute value. Instead of adding the absolute value of that to an output register we can CNOT the highest bit (indicating the sign of the partial sum u) into a single ancilla. Then, we can negate the whole partial sum controlled on this ancilla so that we have the state $|u\rangle\,|0\rangle$ if $u \geq 0$ or $|-u-1\rangle\,|1\rangle$ if $u < 0$. Then, we can add this ancilla to the partial sum register giving us either $|u\rangle\,|0\rangle$ if $u \geq 0$ or $|-u\rangle\,|1\rangle$ if $u < 0$. At this point we can multiply by γ and add the value of u to the $|\phi\rangle$ register and perform phase kickback in order to phase the system by the absolute value of the partial sum. Then, we need to invert adding the sign qubit register to the sum register and uncompute $|u\rangle$ and the ancilla.

Using the sum of tree sums, we numerically find that the Toffoli cost to compute and uncompute the partial sums is no greater than $8N(N+1)/5$ for N in the range 64 to 1024 that we consider. The numerically computed ratios are shown in Fig. 1(b), and for 64, 128, 256, and 1024 we obtain 1.35962, 1.38507, 1.45027, and 1.43186. Multiplying by $\bar{\gamma}$ directly into the phase gradient state has cost ${b_{\text{pha}}}^2/2 + \mathcal{O}(b_{\text{pha}} \log b_{\text{pha}})$, giving a total cost

$$\mathcal{C}_{\text{LABS}}^{\text{phase}} \leq 8N(N+1)/5 + N{b_{\text{pha}}}^2/2 + \mathcal{O}(Nb_{\text{pha}} \log b_{\text{pha}}). \qquad (45)$$

The number of ancillas needed is b_{grad} persistent ancillas for the phase-gradient state, $b_{\text{grad}} - 1$ temporary ancillas for the addition, $\log N + \mathcal{O}(1)$ for the temporary ancilla with the partial sum for H_k, and $2\log N + \mathcal{O}(1)$ for the temporary ancillas used for the sum of tree sums. The ancillas for the partial sum for H_k are needed at the same time as those for the addition into the phase-gradient state, but the temporary ancillas for the sum of tree sums are not. The temporary ancillas for the sum of tree sums is less than those for the addition into the phase-gradient state, so can be ignored. That gives us a total of $b_{\text{grad}} + \lceil \log(N+1) \rceil + 1$ temporary ancillas for a total

$$\begin{aligned}\mathcal{A}_{\text{LABS}}^{\text{phase}} &= b_{\text{grad}} = b_{\text{pha}} + \mathcal{O}(\log b_{\text{pha}}), \qquad (46)\\ \mathcal{B}_{\text{LABS}}^{\text{phase}} &= b_{\text{grad}} + \lceil \log(N+1) \rceil + 1 = b_{\text{pha}}\\ &\quad + \log N + \mathcal{O}(\log b_{\text{pha}}). \qquad (47)\end{aligned}$$

For $9N/5 < {b_{\text{pha}}}^2$, it is more efficient to just compute the entire energy, multiply by $\bar{\gamma}$, then uncompute the energy, as explained above. Then we obtain complexity

$$\mathcal{C}_{\text{LABS}}^{\text{phase}} \leq 5N^2/2 + \mathcal{O}(Nb_{\text{pha}} \log b_{\text{pha}}), \qquad (48)$$

where the cost of multiplying by $\bar{\gamma}$ is absorbed into the order term. Because this is smaller than that given above for $9N/5 < {b_{\rm pha}}^2$, we should give the cost as the minimum of the two complexities

$$\mathcal{C}_{\rm LABS}^{\rm phase} \leq 8N(N+1)/5 + \min\left(N{b_{\rm pha}}^2/2, \frac{9}{10}N^2\right) + \mathcal{O}(Nb_{\rm pha}\log b_{\rm pha}). \tag{49}$$

In that case we need $2\log N + \mathcal{O}(1)$ temporary ancillas for the energy, and $b_{\rm grad} - 1$ temporary ancillas for the addition into the phase-gradient state at the same time. There are also $3\log N + \mathcal{O}(1)$ temporary ancillas for computing the energy, which are not used at the same time as $b_{\rm grad} - 1$ temporary ancillas. That gives a number of temporary ancillas increased to

$$\mathcal{B}_{\rm LABS}^{\rm phase} = \max(b_{\rm pha}, 3\log N) + 2\log N + \mathcal{O}(\log b_{\rm pha}). \tag{50}$$

We give this cost in Table V to account for the possibility of using either method. In the table we assume $3\log N \geq b_{\rm pha}$, because that is true for most combinations of parameters we consider.

D. Oracles for linear combinations of unitaries

A number of approaches to quantum simulation are based on accessing the Hamiltonian as a linear combination of unitaries. This so-called LCU query model [39] has been used for Taylor series simulation [40], interaction picture simulation [41], and generalized to block encodings for "qubitization" [31]. These approaches begin from the observation that any Hamiltonian can be decomposed as a linear combination of unitaries,

$$H = \sum_{\ell=1}^{L} w_\ell U_\ell, \tag{51}$$

where w_ℓ are real scalars and U_ℓ are unitary operators.

Here we consider an approach to forming quantum walks known as qubitization [31]. The quantum walk involves LCU using queries to two oracles, followed by a reflection operation as shown in Fig. 2. The first oracle circuit, the "preparation oracle," acts on an empty ancilla register of $\lceil \log L \rceil$ qubits and prepares a particular superposition state related to the notation of Eq. (51),

$$\text{PREPARE}\,|0\rangle^{\otimes \log L} \mapsto \sum_{\ell=1}^{L}\sqrt{\frac{w_\ell}{\lambda}}\,|\ell\rangle\,, \qquad \lambda \equiv \sum_{\ell=1}^{L}|w_\ell|\,. \tag{52}$$

The quantity λ has significant ramifications for the overall algorithm complexity; specifically, the qubitization oracles need to be repeated a number of times proportional to λ in order to realize the intended quantum walk.

The second oracle circuit we require acts on the ancilla register $|\ell\rangle$ as well as the system register $|\psi\rangle$ and directly applies one of the U_ℓ to the system, controlled on the ancilla register. For this reason, we refer to the ancilla register $|\ell\rangle$ as the "selection register" and name the second oracle the "Hamiltonian selection oracle,"

$$\text{SELECT}\,|\ell\rangle\,|\psi\rangle \mapsto |\ell\rangle\,U_\ell\,|\psi\rangle\,. \tag{53}$$

Using two queries to PREPARE and a single query to SELECT we are able to implement a controlled quantum walk $\mathcal{W}$, which encodes the eigenvalues of H as a function of its own eigenvalues [31]. Specifically, in a subspace this quantum walk has eigenvalues equal to the arccosine of the eigenvalues of the problem Hamiltonian divided by λ.We now discuss the realization of this quantum walk for the problems discussed in Sec. II.

1. LCU oracles for L-term Hamiltonian

Using the strategy for unary iteration introduced in Ref. [26] we can implement SELECT for H_L with Toffoli complexity of exactly $L-2$ and $\lceil \log L \rceil - 1$ extra ancilla qubits (or $L-1$ and $\lceil \log L \rceil$ if the operation needs to be controlled by another ancilla, as it is in Ref. [26]). The circuit given there has $\lceil \log L \rceil$ ancilla. The other ancilla is just a control, it is not needed for the iteration. If we do not want to make it controlled, then the number of ancilla needed is $\lceil \log L \rceil - 1$. Also, the Toffoli cost is only $L-2$ if we do not need to make it controlled. The operator we are to implement is

$$\text{SELECT}\,|\ell\rangle\,|\psi\rangle \mapsto |\ell\rangle \prod_{i\in q_\ell} Z_i\,|\psi\rangle\,. \tag{54}$$

A simple way to understand the strategy is to first map the binary representation of $|\ell\rangle$ to a one-hot unary register (a

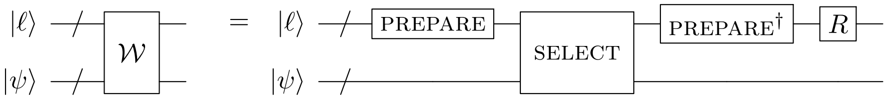

FIG. 2. A circuit realizing the qubitized quantum walk operator $\mathcal{W}$ controlled on an ancilla qubit [26,31]. Here R is a reflection about the zero state for the entire $|\ell\rangle$ register, and therefore has Toffoli complexity $\log L + \mathcal{O}(1)$, where $\lceil \log L \rceil$ is the size of the $|\ell\rangle$ register. However, that overhead is negligible compared to the cost of the PREPARE and SELECT operators in the constructions of this paper.

register that contains L qubits, which are all OFF except for qubit ℓ, which is ON). Then, one could control the application of the Z_i associated with $i \in q_\ell$ on this qubit with only Clifford gates. This strategy has low Toffoli complexity but requires L ancilla. The basic insight of the unary iteration circuits in Ref. [26] is that one can stream through bits of this unary register using just $\lceil \log L \rceil - 1$ extra ancilla. A circuit primitive is repeated L times and at iteration j, a particular ancilla is equal to ON if and only if $\ell = j$. At that point in the circuit we can use Clifford gates to control the application of Hamiltonian terms like $Z_i Z_j Z_k$.

In Ref. [26] a strategy referred to therein as "coherent alias sampling" is introduced and explicit circuits are provided, which allow one to realize PREPARE for an arbitrary model with a Toffoli cost of $L + b_{\rm LCU} + \log L + \mathcal{O}(1)$. We need approximately $\log L$ ancillas for the state being prepared, $\log L$ for the alternate index values, and $\log L$ for the temporary ancillas in the QROM. There are $b_{\rm LCU}$ ancillas for the keep probabilities in the coherent alias sampling and $b_{\rm LCU}$ for the equal superposition state. Another temporary ancilla is used for the result of the inequality test. SELECT uses L Toffolis and $\log L$ temporary ancilla, but these can be reused from the temporary ancilla used by PREPARE. Here, $b_{\rm LCU}$ is a parameter that scales the precision of the cost function. In particular, this strategy generates the state in Eq. (52) but with approximate coefficients $\tilde{w}_\ell$ in place of the exact coefficients w_ℓ such that

$$\left| \sqrt{w_\ell} - \sqrt{\tilde{w}_\ell} \right| \leq 2^{-b_{\rm LCU}}. \tag{55}$$

Per the realization depicted in Fig. 2, the quantum walk of interest is realized using two queries to PREPARE and one query to SELECT. Thus, the strategy we outline requires Toffoli and ancilla counts of

$$\mathcal{C}_L^{\rm LCU} = 3L + 2b_{\rm LCU} + 2\log L + \mathcal{O}(1), \tag{56}$$

$$\mathcal{A}_L^{\rm LCU} = 2\lceil \log L \rceil + 2b_{\rm LCU} + \mathcal{O}(1), \tag{57}$$

$$\mathcal{B}_L^{\rm LCU} = \lceil \log L \rceil = \log L + \mathcal{O}(1). \tag{58}$$

2. LCU oracles for QUBO and using dirty ancilla

In some cases, especially when there is some structure in the Hamiltonian terms and one is willing to reduce gate complexity at the cost of space complexity, another method of implementing PREPARE might be appropriate. In particular, we can combine the coherent alias sampling ideas of Ref. [26] with the on-the-fly "dirty QROAM" of Ref. [42] (which is a concrete realization of an idea in Ref. [43], which builds on the QROM idea of Ref. [26] and is named "QROAM" since it incorporates attributes of both QROM and QRAM). Using Theorem 1 of [42] in conjunction with the coherent alias sampling of Ref. [26] with cost $b_{\rm LCU} + \mathcal{O}(\log N)$, we see that it is possible to implement PREPARE with

$$\frac{2L}{k} + 4b_{\rm LCU}k + \mathcal{O}\left(b_{\rm LCU} + k\log L\right) \tag{59}$$

Toffolis and $(k-1)b_{\rm LCU}$ dirty ancilla in addition to $2b_{\rm LCU} + \log(L/k) + \mathcal{O}(1)$ clean ancilla (not counting the selection register), where $k \in [1, L]$ is a free parameter that must be a power of 2. This sort of QROAM can be uncomputed faster than it can be computed [42]. Combining Theorem 3 in Ref. [42] with coherent alias sampling [26] leads us to the result that the Toffoli cost of uncomputing PREPARE is less than the complexity quoted above by $4(b_{\rm LCU} - 1)k$ and can reuse the same ancilla. The number of dirty ancilla is reduced to $k - 1$, which means that the value of k can be taken to be larger, reducing the Toffoli complexity. See Table VI for detailed costs of various types of QROAM.

We use this dirty QROAM strategy for the QUBO Hamiltonian. Our approach involves indexing the terms and coefficients with two registers, each of size $\lceil \log N \rceil$ so that $|\ell\rangle = |i\rangle\,|j\rangle$. This makes applying SELECT particularly easy as we can use two applications of the unary iteration strategy that we discuss for implementing Eq. (54) to realize SELECT with Toffoli complexity $2N - 4$ and $\lceil \log N \rceil - 1$ ancilla (again, not counting those in the selection register). Because the QROAM strategy needs a single register that takes a contiguous set of values, we need to compute a new register for QUBO. For QUBO where $i \leq j$ one calculates $j(j-1)/2 + i$. (Note that this is with indexing starting from 1, which we do to simplify the sums, but 1 is represented in binary as $00 \ldots 00$, and so forth.) We apply the QROAM to this register, then uncompute it afterwards. The cost of computing and uncomputing this register is $\mathcal{O}(\log^2 N)$ due to the multiplications. Since $L = N(N+1)/2$ for QUBO, the Toffoli cost of implementing SELECT, in addition to implementing (and later uncomputing) PREPARE, is

$$\frac{2N^2}{k} + 4b_{\rm LCU}k + 2N + \mathcal{O}\left(b_{\rm LCU} + \log^2 N\right) \tag{60}$$

and requires $kb_{\rm LCU} + \mathcal{O}(1)$ dirty ancilla and $2b_{\rm LCU} + 2\log(N/k) + 2\log N + \mathcal{O}(1)$ clean ancilla. For simplicity we are taking k to be the same for the computation and uncomputation here, though it is more efficient to take k larger for the uncomputation. Minimizing k by taking the derivative gives us

$$4b_{\rm LCU} - 2N^2/k^2 = 0, \quad k = N/\sqrt{2b_{\rm LCU}}, \tag{61}$$

which leads to Toffoli complexity for the entire walk (including SELECT) going like

$$\begin{aligned} &4N\sqrt{2b_{\rm LCU}} + 2N + \mathcal{O}\left(b_{\rm LCU} + \log^2 N\right) \\ &\quad = 4N\sqrt{2b_{\rm LCU}} + \mathcal{O}(N) \end{aligned} \tag{62}$$

and ancilla complexity for the entire walk going like

$$N\sqrt{b_{\mathrm{LCU}}/2} + 2b_{\mathrm{LCU}} + 2\log b_{\mathrm{LCU}} + 2\log N + \mathcal{O}(1)$$
$$= N\sqrt{b_{\mathrm{LCU}}/2} + \mathcal{O}(\log(b_{\mathrm{LCU}}N)), \tag{63}$$

where the first term in the ancilla scaling corresponds to the dirty ancilla, and thus can use the system qubits. For simplicity we use the exact optimal value of k here; there is a slight increase to the complexity because k needs to be a power of 2 so cannot be taken exactly equal to $N/\sqrt{2b_{\mathrm{LCU}}}$.

While this result optimizes the Toffoli complexity of our implementation it does so at a fairly high cost; we increase the space complexity from $N + \mathcal{O}(b_{\mathrm{LCU}})$ to $N\sqrt{b_{\mathrm{LCU}}/2} + \mathcal{O}(b_{\mathrm{LCU}})$. In many cases this is not a sensible tradeoff and one should instead choose a smaller k so that the total number of qubits is not increased. For instance, $k = N/b_{\mathrm{LCU}}$ never increases the spatial complexity because we always have N system qubits available in the system register that are not acted upon while we apply PREPARE. In some cases (for instance, the quantum simulated annealing algorithm realized by Szegedy quantum walks) we actually have $2N$ qubits available for use during PREPARE and so we can safely take $k = 2N/b_{\mathrm{LCU}}$ without increasing the spatial complexity.

Next we give a more detailed explanation of the costing. The QROAM costings are, for output size M, given in Table VI. The value of L is $L = N(N+1)/2$ for QUBO. The output consists of b_{LCU} qubits for the keep probability in the state preparation, plus $2\lceil \log N\rceil$ qubits for the alternate values of i and j, so

$$M = b_{\mathrm{LCU}} + 2\lceil \log N\rceil. \tag{64}$$

With clean ancilla qubits, the optimal value of k for preparation limited to powers of 2 is

$$k_{c1} = 2^{\mathrm{round}(\log\sqrt{L/M})}, \tag{65}$$

and for inverse preparation is

$$k_{c2} = 2^{\mathrm{round}(\log\sqrt{L})}, \tag{66}$$

The other Toffoli costs in other parts of the LCU (beyond the QROAM) are as follows.

(a) There is $\mathcal{O}(\log N)$ cost to prepare the equal superposition states over i and j with $i \le j$.
(b) There is $2(b_{\mathrm{LCU}} + 2\log N) + \mathcal{O}(1)$ Toffoli cost for the inequality test and controlled swaps for the state preparation and inverse preparation.
(c) The cost of the arithmetic for producing the contiguous ancilla is $\mathcal{O}(\log^2 N)$.
(d) The SELECT has a Toffoli cost of $2N-4$, or $2N-2$ if it needs to be made controlled.

Altogether these costs give a Toffoli cost with clean ancilla of

$$\frac{N(N+1)}{2k_{c1}} + \frac{N(N+1)}{2k_{c2}} + M(k_{c1}-1) + k_{c2}$$
$$+ 2b_{\mathrm{LCU}} + 2N + \mathcal{O}\left(\log^2 N\right), \tag{67}$$

with the values of M, k_{c1}, and k_{c2} in Eqs. (64)–(66). If we ignore the rounding in k_{c1} and k_{c2}, then the Toffoli cost is

$$\sqrt{2b_{\mathrm{LCU}}}N + \mathcal{O}\left(N + b_{\mathrm{LCU}} + b_{\mathrm{LCU}}{}^{-1/2}N\log N\right). \tag{68}$$

The rounding in k_{c1} and k_{c2} can potentially increase the cost by a factor of $1/\sqrt{2} + 1/\sqrt{8}$, or about 6%.

In costing the total number of ancillas for the state preparation, we also need to account for the following (in addition to those in Table VI).

(a) There are $2\lceil\log N\rceil$ qubits needed for the prepared state.
(b) There are b_{LCU} qubits used for the register in equal superposition that we use to perform an inequality test with in the state preparation.
(c) The M output qubits.
(d) There are $\lceil \log L\rceil$ temporary ancilla qubits used for the contiguous register.
(e) There are $b_{\mathrm{LCU}} - 1$ temporary ancillas used in computing the inequality test for the state preparation.

There are also $\log N$ temporary registers needed for the SELECT step, but many of the qubits are only temporarily used by the QROAM, and these can be reused, so we do not get an additional ancilla cost for SELECT. The ancillas additional to those in Table VI can therefore be given as $2M$ persistent ancillas and $\max(\log L, b_{\mathrm{LCU}}) + \mathcal{O}(1)$ temporary ancillas. The ancillas in Table VI are temporary as

TABLE VI. QROAM complexities from Ref. [42], where L is the number of items, k is a power of 2, and M is the output size. This table omits the $\log L$ ancilla from the selection register and the M-qubit output.

Type of ancilla	Type of computation	Toffolis	Clean ancilla	Dirty ancilla
clean	forward	$\lceil L/k\rceil + M(k-1)$	$\lceil \log(L/k)\rceil + M(k-1)$	0
dirty	forward	$2\lceil L/k\rceil + 4M(k-1)$	$\lceil \log(L/k)\rceil$	$M(k-1)$
clean	reverse	$\lceil L/k\rceil + k$	$\lceil \log(L/k)\rceil + k$	0
dirty	reverse	$2\lceil L/k\rceil + 4k$	$\lceil \log(L/k)\rceil + 1$	$k-1$

well, and the $b_{\rm LCU}$ qubits are not needed at the same time, giving a maximum of

$$\log(L/k_{c1}) + M(k_{c1}-1) + \log L + \mathcal{O}(1), \tag{69}$$

temporary ancillas. Ignoring the rounding in k_{c1} for simplicity gives the leading-order term as $N\sqrt{M/2}$ temporary ancillas. Next we consider the cost with N dirty ancilla. The optimal value of k for the QROAM computation is

$$k_{d1} = 2^{\lfloor \log(N/M+1)\rfloor}. \tag{70}$$

For the uncomputation cost it is optimal to take $k_{d2} = \sqrt{L/2}$, which gives a cost of $4\sqrt{2L}$, ignoring rounding of k_{d2} to a power of 2. With $L = N(N+1)/2$, the optimal k_{d2} is $\sqrt{N(N+1)/4} < N$, so there are enough dirty ancilla available. With rounding the value of k_{d2} for uncomputation is

$$k_{d2} = 2^{\mathrm{round}(\log\sqrt{N(N+1)/4})}. \tag{71}$$

Together with the additional Toffoli costs for the state preparation, the Toffoli cost for LCU is

$$\frac{N(N+1)}{k_{d1}} + \frac{N(N+1)}{k_{d2}} + 4M(k_{d1}-1) + 4k_{d2} + 2b_{\rm LCU} + 2N + \mathcal{O}(\log^2 N). \tag{72}$$

To simplify the expression, we use N/M rather than $N/M+1$ in the expression for k_{d1}, and do not take into account rounding k to a power of 2. Then we get a computation Toffoli cost of

$$\mathcal{C}^{\rm LCU}_{\rm QUBO} = N(b_{\rm LCU} + 2\log N) + \mathcal{O}(N). \tag{73}$$

For the ancilla cost, the persistent ancilla cost is again $2M$, and the temporary ancilla cost loses the term $M(k-1)$ because dirty ancilla are used for that, so it does not increase the ancilla cost. The temporary ancilla cost is

$$\max[\log(L/k_{d1}) + \log L, b_{\rm LCU}] + \mathcal{O}(1). \tag{74}$$

Using $L = N(N+1)/2$ and $k_{d1} = N/M$ gives $\log(L/k_{d1}) = \log(N+1) + \log M - 1$. Then $\log(N+1) = \log N + \mathcal{O}(1/N)$. Using $M = b_{\rm LCU} + 2\log N + \mathcal{O}(1)$ then gives

$$\mathcal{A}^{\rm LCU}_{\rm QUBO} = 2b_{\rm LCU} + 4\log N + \mathcal{O}(1), \tag{75}$$

$$\mathcal{B}^{\rm LCU}_{\rm QUBO} = \max(3\log N, b_{\rm LCU}) + \mathcal{O}(\log b_{\rm LCU}). \tag{76}$$

In Table V we just give $3\log N$ for the temporary ancilla cost, because it is true (or close to true) for the combinations of parameters we consider.

3. LCU oracles for the SK model

For the SK model we can considerably improve over the naive implementation. Because the SK-model coefficients only need to give a sign, we just need to apply a sign to the terms in the superposition. That corresponds to the phase fixup used for the QROAM uncomputation, and the cost is the same. Another advantage of this approach is that we eliminate the $2(b_{\rm LCU} + 2\log N) + \mathcal{O}(1)$ cost for the inequality test and controlled swaps that is otherwise needed for the coherent alias sampling. Therefore, the Toffoli cost with clean ancilla is

$$\frac{N(N-1)}{2k_{c2}} + k_{c2} + 2N + \mathcal{O}\left(\log^2 N\right). \tag{77}$$

If we ignore the rounding in k_{c2} then we obtain the complexity

$$(2+\sqrt{2})N + \mathcal{O}\left(\log^2 N\right). \tag{78}$$

Beyond the ancillas needed for the QROAM, we just need the $4\log N + \mathcal{O}(1)$ qubits for the i, j, and contiguous registers. Again SELECT can use the same temporary ancillas as the QROAM and does not add to the ancilla cost. Therefore, the ancilla cost is

$$\log(L/k_{c2}) + k_{c2} + 4\log N + \mathcal{O}(1). \tag{79}$$

Ignoring the rounding in k_{c2} for simplicity gives

$$N/\sqrt{2} + \mathcal{O}\left(\log N\right). \tag{80}$$

If we are using dirty ancilla, then the Toffoli cost becomes

$$\frac{N(N-1)}{k_{d2}} + 4k_{d2} + 2N + \mathcal{O}\left(\log^2 N\right). \tag{81}$$

Ignoring the rounding in k_{d2} we obtain the complexity

$$\mathcal{C}^{\rm LCU}_{\rm SK} = 6N + \mathcal{O}\left(\log^2 N\right). \tag{82}$$

The persistent ancilla cost is only $2\log N$ for the i and j registers, and there is temporary ancilla cost of $2\log N$ for the contiguous register and $\log(L/k_{d2}) \approx \log N$ from the QROAM. The total ancilla costs are therefore

$$\mathcal{A}^{\rm LCU}_{\rm SK} = 2\log N + \mathcal{O}(1), \tag{83}$$

$$\mathcal{B}^{\rm LCU}_{\rm SK} = 3\log N + \mathcal{O}(1). \tag{84}$$

4. LCU oracles for the LABS model

The LABS problem has $L = \mathcal{O}(N^3)$ terms in it, which leads to a high complexity quantum walk if our general strategy were applied. Fortunately, there is much structure in this problem. We start by rewriting Eq. (6) as

$$H_{\text{LABS}} = \sum_{k=0}^{N-1}\sum_{j=1}^{N-k}\sum_{i=1}^{N-k} Z_i Z_{i+k} Z_j Z_{j+k}. \tag{85}$$

Instead of linearly indexing all $\mathcal{O}(N^3)$ terms, we use three registers, each of size $\log N$, which store the values of i, j, and k. Thus, our SELECT operation acts as

$$\text{SELECT}\,|i\rangle\,|j\rangle\,|k\rangle\,|\psi\rangle \mapsto |i\rangle\,|j\rangle\,|k\rangle\, Z_i Z_{i+k} Z_j Z_{j+k}\,|\psi\rangle\,. \tag{86}$$

To accomplish this, we simply need four applications of the unary iteration primitive described in Ref. [26]. Each of these primitives require $N-1$ Toffoli gates. The only nuance is that we need to compute the values $i+k$ and $j+k$ before implementing the primitive to perform Z_{i+k} and Z_{j+k}.

These additions can be performed in place (and then uncomputed) in the i and j registers and introduce a negligible additive $4\log N$ cost to the cost of unary iteration, where the cost of addition is $\lceil \log N\rceil - 1 \le \log N$. Thus, the total Toffoli cost of our SELECT implementation is $4N + 4\log N$. We require approximately $3\log N$ persistent ancilla for the i, j, and k registers, another $\log N$ temporary ancilla for computing the $i+k$ and $j+k$ (since they are computed in place), and $\log N$ temporary ancilla for the addition. The unary iteration uses $\lceil \log N\rceil - 2 < \log N$ ancillas, which can be reused from the temporary ancillas for the addition so do not add to the cost. Because all terms have the same coefficient, PREPARE needs to initialize a superposition over a number of items that is not a power of 2. The Toffoli cost is $\mathcal{O}(\log N)$. The only unfortunate aspect is that for the LABS problem the corresponding normalization λ, is quite large and this enters into the complexity of our quantum walks as the number of times the quantum walk must be repeated to realize the intended unitary. In total then the cost to realize the quantum walk in Fig. 2 is

$$\mathcal{C}_{\text{LABS}}^{\text{LCU}} = 4N + \mathcal{O}(\log N), \tag{87}$$

$$\mathcal{A}_{\text{LABS}}^{\text{LCU}} = 3\log N + \mathcal{O}(1), \tag{88}$$

$$\mathcal{B}_{\text{LABS}}^{\text{LCU}} = 2\log N + \mathcal{O}(1), \tag{89}$$

$$\lambda_{\text{LABS}} \approx N^3/3. \tag{90}$$

E. QROM-based function evaluation

Now that we have explained how to implement oracles for various cost functions of interest, we turn to the question of how to calculate functions of the cost. This is important for several possible approaches to heuristic-based combinatorial optimization. In simulated annealing, for instance, the probability of moving from one candidate solution to another is proportional to an exponential of the energy difference between the two solutions, multiplied by an inverse temperature β. We thus require the quantum computer to calculate an exponential of the output of the relevant energy-difference oracle.

Because we are implementing *heuristic* approaches to combinatorial optimization, we do not expect that the functions of the cost need to be calculated to a high degree of accuracy so long as the functions we compute are still monotonic in the cost (to make sure that the energy landscape is not inverted in any way). We instead want to minimize the computational complexity of evaluating these functions given rather weak requirements on the accuracy of the output. Here we describe a general strategy for such cheap approximate function evaluation.

Our overall strategy is to approximate a function f of a b-bit input z by a piecewise linear approximation, $\tilde{f}$. This approximation $\tilde{f}$ is calculated based on a choice of *sample* points $z_0 < z_1 < \cdots < z_g$, where $z_0 \le z < z_g$. These sample points separate the interval $[z_0, z_L)$ into g different subintervals of the form $[z_\ell, z_{\ell+1})$ with $\ell = 0, 1, \ldots, g-1$. The input z belongs to exactly one of these subintervals, and so we find an ℓ such that $z_\ell \le z < z_{\ell+1}$. Having found ℓ, we use some data that can be looked up in order to calculate $\tilde{f}(z) = \alpha f(z_\ell) + (1-\alpha) f(z_{\ell+1})$ for $\alpha = (z_{\ell+1} - z)/(z_{\ell+1} - z_\ell)$. That is, the function $\tilde{f}$ is defined by interpolating between known values $f(z_\ell)$ and $f(z_{\ell+1})$ of the target function f.

QROM [26] can be used to obtain the region that z is in (i.e., the correct value of ℓ above), and for that region the QROM outputs a slope and intercept for the linear approximation. The Toffoli cost of looking up one of g different possible values in the scheme of Ref. [26] is $g-2$, or $g-1$ if the output is controlled by a qubit. This Toffoli count relies on a technique from Ref. [35] in which certain naively expected Toffolis can be replaced with Clifford gates plus measurement. Also note that the Toffoli count of QROM-based lookup is independent of the number of bits of data output, meaning that we are free to choose any number of bits to represent the slope and intercept without introducing a Toffoli cost from the QROM. We choose the number of bits in order to obtain b_{sm} bits for $\tilde{f}$. That is, $\tilde{f}$ may be a rough approximation of f, but we give $\tilde{f}$ to more bits than needed by that approximation so $\tilde{f}$ has smooth behavior.

We do not use QROM precisely as specified in Ref. [26] but rather a variant of it. To explain the distinction, we begin with some terminology. QROM is a method for executing a quantum circuit that operates on two registers, an `input` register and an `output` register. The `input` register has an initial value of ℓ encoded into it and the `output` register starts in the all-zero state. Each value of ℓ corresponds to some piece of data d_ℓ that has been specified classically before the quantum circuit was constructed. The effect of the QROM is

$$\mathrm{QROM}: |\ell\rangle_{\texttt{input}} |0\rangle_{\texttt{output}} \mapsto |\ell\rangle_{\texttt{input}} |d_\ell\rangle_{\texttt{output}} . \quad (91)$$

Our variant of QROM is designed for the case in which there are data collisions. That is to say, we consider the case where $d_\ell = d_{\ell'}$ for several different pairs ℓ and ℓ'. In Fig. 3(a) we explain how this QROM variant works for $L = 16$ in the case where $d_4 = d_5$, $d_6 = d_7$, $d_8 = d_9 = d_{10} = d_{11}$, and $d_{12} = d_{13} = d_{14} = d_{15}$. In this variant, we imagine that we have distinct parts of the iteration: iterate by $\ell \mapsto \ell + 1$, iterate by $\ell \mapsto \ell + 2$, iterate by $\ell \mapsto \ell + 4$, and so on for each power of 2. This variant of QROM is appropriate for our purposes because we want to improve computational efficiency by spacing z_ℓ unevenly. This is equivalent to treating many pieced of data d_ℓ as being equal, as the data is simply the information needed to calculate a linear function. The total number of Toffoli gates is still $g - 2$ for g distinct regions, provided these regions correspond to ignoring bits of the input. For example, we can use a region such as $\{4, 5\}$, but not $\{3, 4\}$, because $4 \equiv 100$ and $5 \equiv 101$, so grouping 4 and 5 corresponds to ignoring the least significant bit, but the least significant bit changes between 3 and 4.

A further subtlety is that all regions need to be a size corresponding to a power of 2 for this cost. In some cases we may wish to have a final region that is larger than half, so it is not a size that is a power of 2. That occurs because we can have a large energy difference, but the exponential gives a transition probability that can just be approximated as zero for a wide range of energies. Then the cost can be larger. For example, if we are distinguishing 0 from 1–15, then it takes three Toffolis. The cost can be seen from the diagram where the size of the regions increases in powers of 2, shown in Fig. 3(b). There one can choose the numbers used for $d_{4–7}$ and $d_{8–15}$ to be equal, which gives a region for 4–15. This choice corresponds to a situation where the gap between neighboring interpolation points z_ℓ grows exponentially.

For many of the piecewise approximations, we can obtain accurate approximations using just powers of 2, as in Fig. 3(b). Two main types of function that we aim to approximate are the exponential and the arcsine of the exponential. For the exponential the piecewise approximation can use points at argument values of 0, 1/2, 1 and so on and achieve a piecewise linear approximation within about 0.03. The arcsine of the exponential is more difficult to approximate because the slope diverges at an argument of 0, but using piecewise linear approximation points starting at $1/2^{11}$ and going up by powers of 2 gives similar precision as for the exponential.

To estimate the number of interpolation points needed for higher precision, note that the error of interpolation of function $f(z)$ is approximately

$$\frac{(\delta z)^2}{8} f''(z), \quad (92)$$

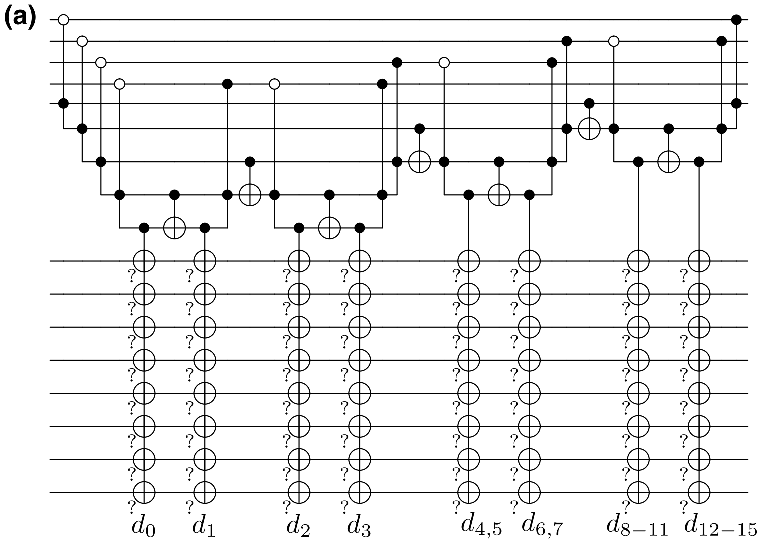

FIG. 3. (a) This figure shows how to perform QROM with variable spacing for the example where there are 4 bits, and we aim to group the input numbers as 0, 1, 2, 3, $\{4, 5\}$, $\{6, 7\}$, $\{8, 9, 10, 11\}$, $\{12, 13, 14, 15\}$. That is, we output the same data for inputs of 4 and 5, and so forth. The first four lines are the four input bits and the fifth is a control register. There are six Toffolis needed in this example for eight data points, with one more Toffoli for a control. (b) This figure shows how to perform QROM with variable spacing for the example where there are 4 bits, and we group the input numbers by powers of 2 as 0, 1, 2–3, 4–7, and 8–15. There are three Toffolis needed in this example for five data points, with one more Toffoli for a control.

where δz is the width of the interval. To obtain error no greater than $2^{-b_{\rm fun}}$, we can therefore take

$$\delta z = \frac{2^{-b_{\rm fun}/2}\sqrt{8}}{\sqrt{f''(z)}}. \tag{93}$$

We can therefore estimate the number of intervals needed to approximate the function by

$$\frac{2^{b_{\rm fun}/2}}{\sqrt{8}} \int_0^\infty dz \sqrt{f''(z)}. \tag{94}$$

In the case where we are approximating $\arcsin[\exp(-z/2)]$, then we get $g \approx 1.31103 \times 2^{b_{\rm fun}/2}$, and if we were approximating $\exp(-z)$, then we have $g \approx 2^{(b_{\rm fun}-1)/2}$. For the three functions used for spectral-gap amplification, $1/\sqrt{1+e^{-z}}$, $e^{-z}/\sqrt{1+e^{-z}}$, and $e^{-z/2}/\sqrt{1+e^{-z}}$ we get $2^{-b_{\rm fun}/2}g$ of 0.346002, 0.566302, and 0.517075, respectively. The variation of $2^{-b_{\rm fun}/2}g$ with $b_{\rm fun}$ is shown in Fig. 4(a). In practice, we need to limit the intervals to sizes that increase by factors of 2 as described above. That increases the values of $2^{-b_{\rm fun}/2}g$ to around 1.0, 1.9, 0.5, 0.8, and 0.7 for the five cases, as can be seen in Fig. 4(b), an increase of around 44%. Nevertheless, it is reasonable to give the scaling of g as $\mathcal{O}(2^{b_{\rm fun}/2})$, with the constant factor somewhere between 0.5 and 2.

In the linear interpolation, the primary cost is that of multiplication of the argument times the slope. This cost depends on how many digits are used for the slope and the argument. For simplicity, consider the case where bits of the argument can be divided between those before the decimal point and those after the decimal point. The maximum value needed for the argument is $\mathcal{O}(b_{\rm sm})$, because beyond that the functions are within $1/2^{b_{\rm sm}+1}$ of their asymptotic values. That means only $\log b_{\rm sm} + \mathcal{O}(1)$ bits are needed before the decimal point. The number of digits after the decimal point depends on the maximum value of the slope. In the case of the exponential the maximum slope is 1, so only $b_{\rm sm}$ bits are needed. Because the slope could be multiplied by an argument that is $\mathcal{O}(b_{\rm sm})$, it could need $b_{\rm sm} + \log b_{\rm sm} + \mathcal{O}(1)$ bits after the decimal point. Both numbers need approximately $b_{\rm sm} + \log b_{\rm sm} + \mathcal{O}(1)$ bits. This gives a cost of multiplication of ${b_{\rm sm}}^2 + \mathcal{O}(b_{\rm sm} \log b_{\rm sm})$ Toffoli gates.

The same result is obtained for all other functions we consider except the arcsine. The arcsine has a slope that goes to infinity, but the linear interpolation only uses a finite slope. The minimum interpolation point needs to be $\mathcal{O}(2^{-2b_{\rm fun}})$, which gives a maximum slope of $\mathcal{O}(2^{b_{\rm fun}})$, so the argument requires another $b_{\rm fun} + \mathcal{O}(1)$ bits after the decimal point. The slope needs $b_{\rm fun} + \mathcal{O}(1)$ bits before the decimal point, and $b_{\rm sm} + \log b_{\rm sm} + \mathcal{O}(1)$ bits after the decimal point to account for the maximum argument. Then both numbers need $b_{\rm fun} + b_{\rm sm} + \log b_{\rm sm} + \mathcal{O}(1)$ bits. We take $b_{\rm fun}$ similar to $b_{\rm sm}$, giving a multiplication cost of $(b_{\rm sm} + b_{\rm fun})^2 + \mathcal{O}(b_{\rm sm} \log b_{\rm sm})$ Toffoli gates.

To estimate the numbers of bits needed, we perform simulation of the technique of Sec. III E with the SK Hamiltonian on 16 qubits, as shown in Fig. 5. In that technique, we need an approximation of the arcsine of the transition probability to control a qubit rotation, rather than the transition probability itself. Choosing interpolation points such that the error in the approximation of the rotation angle is no more than 0.01, the success probabilities are almost unchanged. So far we assume that the energy difference has been multiplied by the inverse temperature β before being input to the procedure. It is possible to bundle the multiplication by β into the oracle, and as shown in Fig. 5 that again has similar performance. There is also the question of how many bits are needed in the function approximating the transition function. We again find that low-precision approximations have very little impact on the success probability.

The overall complexity of the interpolation excluding the QROM is therefore $b_{\rm sm} + \mathcal{O}(b_{\rm sm} \log b_{\rm sm})$ or $(b_{\rm sm} + b_{\rm fun})^2 + \mathcal{O}(b_{\rm sm} \log b_{\rm sm})$ when the arcsine is needed. To

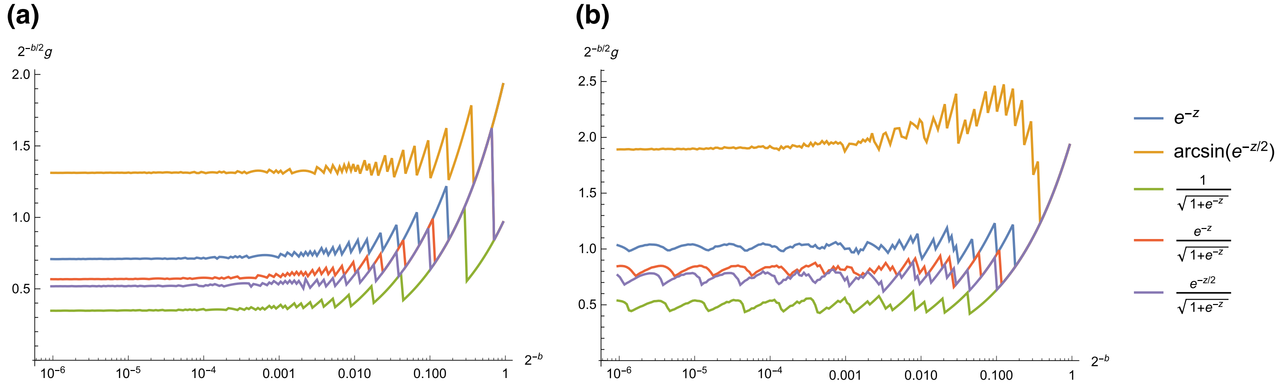

FIG. 4. The numbers of intervals multiplied by $2^{-b_{\rm fun}/2}$ for the five functions we consider. In (a) we allow the intervals to have general endpoints, and in (b) we restrict the intervals to change by factors of 2, to be consistent with the QROM method we use. This demonstrates that the number of intervals scales as $2^{b_{\rm fun}/2}$ with a scaling constant around 1.

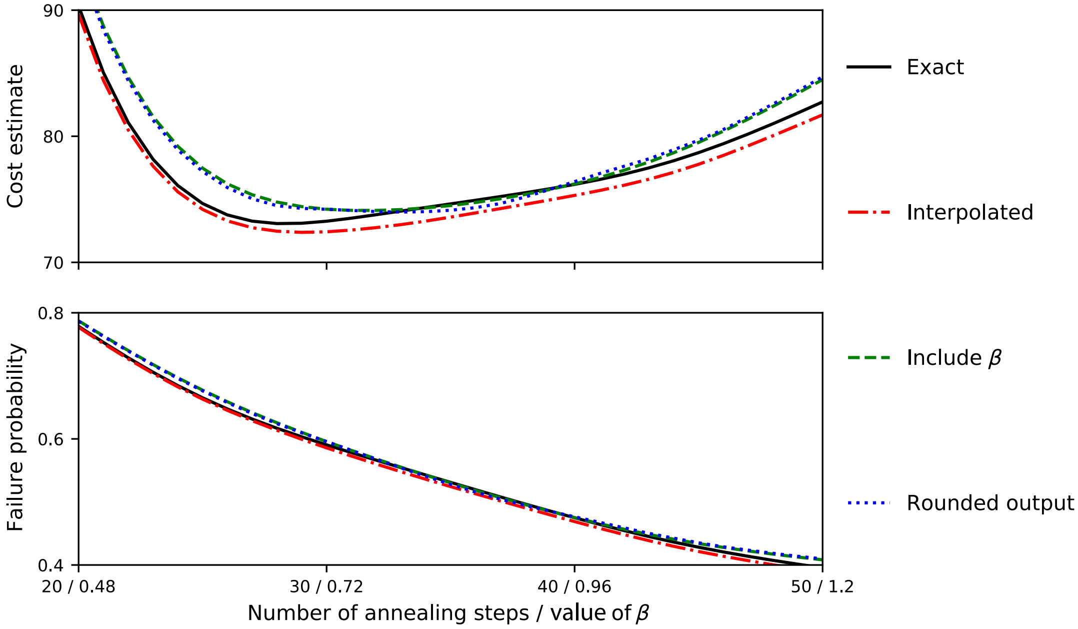

FIG. 5. Effect of various methods of function approximation on optimization performance. We numerically simulate the quantum simulated annealing technique of Sec. III E using various methods of approximating the transition probability. We consider the performance when the rotation angle is calculated to machine precision ("exact"), with piecewise linear approximation chosen to ensure the worst-case error does not exceed 0.01 ("interpolated"), incorporating the inverse temperature β into the definition of the function so that we are interpolating $f(z) = \arcsin[\exp(-\beta z/2)]$ rather than $f(z) = \arcsin[\exp(-z/2)]$ to avoid a multiplication ("include β"), and when we round off the output of the function to 7 bits ("rounded output"). Each of these approximations builds upon the previous approximation, so we perform linear interpolation in all but the exact method. We simulate the performance averaging over 4096 random SK instances on 16 qubits, with β linearly increasing over 50 steps from 0 to 1.2. We report the average failure probability (bottom) as well as an estimate of the computational cost (top) in which we calculate the number of annealing steps divided by the probability of success. We observe that the differences in performance are not meaningfully affected by the method of function approximation, suggesting that we can pick the computationally cheapest option for our cost analysis.

estimate the QROM complexity, we need to account for the final region not being a size that is a power of 2. In the worst case the additional cost can be no larger than $b_{\rm dif}$, which is the total size of the input register. We can therefore bound the QROM complexity as $b_{\rm dif} + \mathcal{O}(2^{b_{\rm fun}/2})$, giving total interpolation complexity of

$$\mathcal{C}^{\rm fun} = {b_{\rm sm}}^2 + b_{\rm dif} + \mathcal{O}(b_{\rm sm}\log b_{\rm sm} + 2^{b_{\rm fun}/2}), \tag{95}$$

or, for the case where the arcsine is needed,

$$\mathcal{C}^{\rm fun} = (b_{\rm sm} + b_{\rm fun})^2 + b_{\rm dif} + \mathcal{O}(b_{\rm sm}\log b_{\rm sm} + 2^{b_{\rm fun}/2}). \tag{96}$$

For the number of ancilla qubits needed, except for the arcsine case there are $2b_{\rm sm} + \mathcal{O}(\log b_{\rm sm})$ needed for the slope and intercept, and $2b_{\rm sm} + \mathcal{O}(\log b_{\rm sm})$ used as temporary ancillas for the arithmetic. We need $b_{\rm dif} - 1$ temporary ancillas for the QROM, which is more than the number used for the arithmetic. The output for the transition probability can be added into the slope, so does not increase the ancilla cost. Therefore, the ancilla costs are

$$\mathcal{A}^{\rm fun} = 2b_{\rm sm} + \mathcal{O}(\log b_{\rm sm}), \tag{97}$$

$$\mathcal{B}^{\rm fun} = b_{\rm dif} - 1. \tag{98}$$

These considerations give the costs for function evaluation in Table V. For the arcsine case we need $2b_{\rm sm} + b_{\rm fun} + \mathcal{O}(\log b_{\rm sm})$ ancillas for the slope and intercept, because we need another $b_{\rm fun}$ ancilla for the slope. Again the temporary ancilla cost is primarily for the QROM, so the ancilla costs are

$$\mathcal{A}^{\rm fun} = 2b_{\rm sm} + b_{\rm fun} + \mathcal{O}(\log b_{\rm sm}), \tag{99}$$

$$\mathcal{B}^{\rm fun} = b_{\rm dif} - 1. \tag{100}$$

III. Optimization Methods

In this section we review proposals for heuristic quantum optimization algorithms and explain how those algorithms can be implemented in terms of the oracles we describe in Sec. II. By this we include methods based on Hamiltonian walks, those based on time evolution, and methods related to simulated annealing. In most cases we suggest improvements to these methods, but an important motivation for this section is to give a complete analysis of the complexity of these algorithms, which includes constant factors so that we can estimate the resources required to realize them in the surface code in Sec. IV. We describe the complexities of these methods in terms of the oracles from the previous section in Table VII, then give the complexity in terms of Toffoli or T gates in Table VIII.

As this section incorporates a wide variety of sophisticated techniques, we begin with a brief summary of the approaches we are considering.

(a) *Amplitude amplification* (Sec. III A). We start by considering amplitude amplification, which can be used to directly amplify the amplitude of the solution. Unlike the other methods, it takes no advantage of the structure of the solution, so is a useful reference point to compare to the other optimization approaches. Amplitude amplification can also be used in combination with the other optimization approaches, by performing amplitude amplification on the output of the optimization.

(b) *The quantum approximate optimization algorithm* (Sec. III B). The steps of this approach (QAOA) are equivalent to Trotter steps, so the costing for QAOA and Trotter steps is given in the same lines in Table VII and Table VIII. Trotter steps can be used for adiabatic approaches, which are considered in the next subsection. But here we also focus on strategies for efficiently estimating the QAOA objective value that are more appropriate for a fault-tolerant cost model than standard approaches.

(c) *Adiabatic quantum optimization* (Sec. III C). We review the quantum adiabatic algorithm [2] and the most straightforward way of implementing that approach using a Trotter method that queries the phase oracles presented in Sec. II. We then suggest a strategy for implementing the adiabatic algorithm using the LCU oracles presented in Sec. II. The LCU oracles have a different costing to Trotter and QAOA, so are given in separate lines in Tables VII and VIII. Next, we review a method for digitizing the adiabatic algorithm while suppressing certain types of errors that is based on inducing quantum Zeno-effect-like projection to the ground state by randomizing phases [21]. We suggest how this approach can be improved by using carefully chosen probability distributions to eliminate the errors that manifest from incorrect measurements in the Zeno approach. The time-evolution oracles used by these methods are also suitable for the quantum-enhanced population-transfer algorithm and the shortest-path algorithm. Since we do not introduce new techniques for those algorithms, but rather review how

TABLE VII. The Toffoli complexities and ancilla complexities needed to implement the basic primitives of various heuristic algorithms, reported in terms of the oracle costs from Table IV. For both the LHPST walk step and the gap-amplified walk step the cost is reduced by $8\log N$ when N is a power of 2. By combining these scalings with the oracle costs given in Table V, we arrive at the resource estimates in Table VIII. For order ρ Suzuki [meaning that the error is $\mathcal{O}(\delta t^{\rho+1})$] we multiply the QAOA cost by $2\times 5^{\rho/2-1}$. For the QAOA and Trotter step, the b_{pha} ancillas are for a phase-gradient state, and may be saved if those are already accounted for in $\mathcal{A}^{\text{phase}}$. The quantity ϵ is an allowable error in synthesizing a rotation in the state preparation.

Algorithm primitive	Toffoli complexity	Ancilla complexity
Amplitude-amplification step	$2\mathcal{C}^{\text{direct}}+N+\mathcal{O}(b_{\text{dir}})$ (108)	$\mathcal{A}^{\text{direct}}+\mathcal{B}^{\text{direct}}+\mathcal{O}(1)$
QAOA and Trotter step	$\mathcal{C}^{\text{phase}}+4N+b_{\text{pha}}{}^2/2+\mathcal{O}(b_{\text{pha}}\log b_{\text{pha}})$ (137)	$\mathcal{A}^{\text{phase}}+\max(\mathcal{B}^{\text{phase}},3\log N)+b_{\text{pha}}+\mathcal{O}(\log b_{\text{pha}})$
Hamiltonian walk step	$\mathcal{C}^{\text{LCU}}+\mathcal{O}(1)$	$\mathcal{A}^{\text{LCU}}+\mathcal{B}^{\text{LCU}}+\mathcal{O}(1)$
Szegedy walk annealing step	$\min[2(N+1)\mathcal{C}^{\text{direct}},2N\mathcal{C}^{\text{diff}}]+2N\mathcal{C}^{\text{fun}}+2N\log N+8Nb_{\text{sm}}+18b_{\text{sm}}{}^2+\mathcal{O}(N)$ (189)	$N\mathcal{A}^{\text{diff}}+N\mathcal{A}^{\text{fun}}+5b_{\text{sm}}+\mathcal{O}(N)$ (190)
LHPST-walk annealing step	$2\mathcal{C}^{\text{diff}}+2\mathcal{C}^{\text{fun}}+N+2b_{\text{dif}}+9\log N+\mathcal{O}(1)$ (213)	$\mathcal{A}^{\text{diff}}+\mathcal{A}^{\text{fun}}+\mathcal{B}^{\text{diff}}+\log N+b_{\text{sm}}+\mathcal{O}(1)$ (214)
Gap-amplified walk step	$2\mathcal{C}^{\text{diff}}+2\mathcal{C}^{\text{fun}}+2b_{\text{sm}}+N+14\log N+\mathcal{O}(b_{\text{rot}})$ (232)	$\mathcal{A}^{\text{diff}}+\mathcal{A}^{\text{fun}}+\mathcal{B}^{\text{diff}}+2\log N+b_{\text{sm}}+\mathcal{O}(1)$ (233)

TABLE VIII. Resource estimates for the various heuristic optimization primitives explored throughout this paper, applied to our four problems of interest. For both the LHPST-walk step and the gap-amplified walk step the Toffoli count is reduced by $8\log N$ when N is a power of 2. In all cases, these algorithms are refined by applying the primitive more times. The parameters used are as follows: N is the number of bits on which our cost function is defined; L is the numbers of terms in an L-term spin Hamiltonian; b_{pha} is the number of bits we use to approximate phases in the implementation of our phase oracle; b_{dir} is the number of bits we use to approximate the value of energies; b_{LCU} is the number of bits used to approximate the square root of Hamiltonian coefficients in LCU methods, and b_{rot} is the number of bits of precision used in rotations. The Trotter step and Hamiltonian walk steps can be used to realize the adiabatic algorithm, the Zeno-phase randomization variant of the adiabatic algorithm, heuristic variants of the short-path algorithm or quantum-enhanced population transfer, and many other heuristics based on Hamiltonian time evolution. These scalings result from combining the query complexities in Table VII with the oracle costs in Table V. When the algorithm type is decorated with (*) we report T complexity rather than Toffoli complexity. We have only given the main terms in the order expressions to simplify them.

Cost function	Algorithm primitive	Toffoli (* or T) count	total ancilla qubits
L-term	Amplitude-amplification step	$2Lb_{\mathrm{dir}} + N + \mathcal{O}(b_{\mathrm{dir}})$	$2b_{\mathrm{dir}} + \mathcal{O}(1)$
Spin	QAOA and Trotter step*	$1.15L(b_{\mathrm{pha}} + \log L) + \mathcal{O}(N + \log L + {b_{\mathrm{pha}}}^2)$	$3\log N + b_{\mathrm{pha}} + \mathcal{O}(\log b_{\mathrm{pha}})$
Model	Hamiltonian walk step	$3L + 2b_{\mathrm{LCU}} + \mathcal{O}(\log L)$	$3\log L + 2b_{\mathrm{LCU}} + \mathcal{O}(1)$
H_L	Szegedy walk annealing step	$2(N+1)Lb_{\mathrm{dir}} + 2N({b_{\mathrm{sm}}}^2 + b_{\mathrm{dif}} + \log N) + \mathcal{O}(Nb_{\mathrm{sm}}\log b_{\mathrm{sm}})$	$Nb_{\mathrm{dif}} + 2Nb_{\mathrm{sm}} + \mathcal{O}(N\log b_{\mathrm{sm}})$
	LHPST-walk annealing step	$4Lb_{\mathrm{dif}} + 2(b_{\mathrm{sm}} + b_{\mathrm{fun}})^2 + 2b_{\mathrm{dif}} + N + 9\log N + \mathcal{O}(b_{\mathrm{sm}}\log b_{\mathrm{sm}})$	$3b_{\mathrm{sm}} + 2b_{\mathrm{dif}} + b_{\mathrm{fun}} + \log N + \mathcal{O}(\log b_{\mathrm{sm}})$
	Gap-amplified walk step	$4Lb_{\mathrm{dif}} + 2{b_{\mathrm{sm}}}^2 + 2b_{\mathrm{dif}} + N + 14\log N + \mathcal{O}(b_{\mathrm{rot}})$	$3b_{\mathrm{sm}} + 2b_{\mathrm{dif}} + 2\log N + \mathcal{O}(\log b_{\mathrm{sm}})$
Quadratic	Amplitude amplification	$N^2 b_{\mathrm{dir}} + \mathcal{O}(Nb_{\mathrm{dir}})$	$2b_{\mathrm{dir}} + \mathcal{O}(1)$
Unconstrained	QAOA and Trotter step*	$0.575N^2(b_{\mathrm{pha}} + 2\log N) + \mathcal{O}(N^2)$	$3\log N + b_{\mathrm{pha}} + \mathcal{O}(\log b_{\mathrm{pha}})$
Binary	Hamiltonian walk step	$N(b_{\mathrm{LCU}} + 2\log N) + \mathcal{O}(N)$	$7\log N + 2b_{\mathrm{LCU}} + \mathcal{O}(\log b_{\mathrm{LCU}})$
Optimization	Szegedy walk annealing step	$2N^2 b_{\mathrm{dif}} + 2N({b_{\mathrm{sm}}}^2 + b_{\mathrm{dif}} + \log N) + \mathcal{O}(Nb_{\mathrm{sm}}\log b_{\mathrm{sm}})$	$Nb_{\mathrm{dif}} + 2Nb_{\mathrm{sm}} + \mathcal{O}(N\log b_{\mathrm{sm}})$
H_{QUBO}	LHPST-walk annealing step	$2Nb_{\mathrm{dif}} + 2(b_{\mathrm{sm}} + b_{\mathrm{fun}})^2 + 2b_{\mathrm{dif}} + N + 9\log N + \mathcal{O}(b_{\mathrm{sm}}\log b_{\mathrm{sm}})$	$3b_{\mathrm{sm}} + 2b_{\mathrm{dif}} + b_{\mathrm{fun}} + \log N + \mathcal{O}(\log b_{\mathrm{sm}})$
	Gap-amplified walk step	$2Nb_{\mathrm{dif}} + 2{b_{\mathrm{sm}}}^2 + 2b_{\mathrm{dif}} + N + 14\log N + \mathcal{O}(b_{\mathrm{rot}})$	$3b_{\mathrm{sm}} + 2b_{\mathrm{dif}} + 2\log N + \mathcal{O}(\log b_{\mathrm{sm}})$
Sherrington-	Amplitude amplification step	$2N^2 + N + \mathcal{O}(\log N)$	$6\log N + \mathcal{O}(1)$
Kirkpatrick	QAOA and Trotter step	$2N^2 + 4N + {b_{\mathrm{pha}}}^2 + \mathcal{O}(b_{\mathrm{pha}}\log b_{\mathrm{pha}})$	$6\log N + b_{\mathrm{pha}} + \mathcal{O}(\log b_{\mathrm{pha}})$
model	Hamiltonian walk step	$6N + \mathcal{O}(\log^2 N)$	$5\log N + \mathcal{O}(1)$
H_{SK}	Szegedy walk annealing step	$4N^2 + 2N({b_{\mathrm{sm}}}^2 + 2\log N) + 8Nb_{\mathrm{sm}} + 18{b_{\mathrm{sm}}}^2 + \mathcal{O}(Nb_{\mathrm{sm}}\log b_{\mathrm{sm}})$	$N\log N + 2Nb_{\mathrm{sm}} + \mathcal{O}(N\log b_{\mathrm{sm}})$
	LHPST-walk annealing step	$5N + 2(b_{\mathrm{sm}} + b_{\mathrm{fun}})^2 + 11\log N + \mathcal{O}(b_{\mathrm{sm}}\log b_{\mathrm{sm}})$	$4\log N + 3b_{\mathrm{sm}} + b_{\mathrm{fun}} + \mathcal{O}(\log b_{\mathrm{sm}})$
	Gap-amplified walk step	$5N + 2{b_{\mathrm{sm}}}^2 + 16\log N + \mathcal{O}(b_{\mathrm{rot}})$	$5\log N + 3b_{\mathrm{sm}} + \mathcal{O}(\log b_{\mathrm{sm}})$
Low	Amplitude-amplification step	$5N(N+1)/2 + N + \mathcal{O}(\log N)$	$5\log N + \mathcal{O}(1)$
Autocorrelation	QAOA and Trotter step	$8N^2/5 + \min\left(N{b_{\mathrm{pha}}}^2/2, 9N^2/10\right) + \mathcal{O}(Nb_{\mathrm{pha}}\log b_{\mathrm{pha}})$	$5\log N + b_{\mathrm{pha}} + \mathcal{O}(\log b_{\mathrm{pha}})$
Binary	Hamiltonian walk step	$4N + \mathcal{O}(\log N)$	$5\log N + \mathcal{O}(1)$
Sequences	Szegedy walk annealing step	$5N(N+1)^2/2 + 2N({b_{\mathrm{sm}}}^2 + 3\log N) + \mathcal{O}(Nb_{\mathrm{sm}}\log b_{\mathrm{sm}})$	$2N\log N + 2Nb_{\mathrm{sm}} + \mathcal{O}(N\log b_{\mathrm{sm}})$
H_{LABS}	LHPST-walk annealing step	$5N^2 + 2(b_{\mathrm{sm}} + b_{\mathrm{fun}})^2 + 6N + 13\log N + \mathcal{O}(b_{\mathrm{sm}}\log b_{\mathrm{sm}})$	$6\log N + 3b_{\mathrm{sm}} + b_{\mathrm{fun}} + \mathcal{O}(\log b_{\mathrm{sm}})$
	Gap-amplified walk step	$5N^2 + 2{b_{\mathrm{sm}}}^2 + 6N + 18\log N + \mathcal{O}(b_{\mathrm{rot}})$	$7\log N + 3b_{\mathrm{sm}} + \mathcal{O}(\log b_{\mathrm{sm}})$

our oracles can be queried within those frameworks, we discuss that content in Appendix E.

(d) *Szegedy walk-based quantum simulated annealing* (Sec. III D). Simulated annealing is a classical algorithm that mimics a physical cooling process via Markov chain Monte Carlo techniques. The quantum algorithm of Somma *et al.* [13] is to replace the Markov chain with a corresponding Szegedy walk. If the spectral gap of the Markov transition operator is Δ, the number of Szegedy walk steps grows as $\mathcal{O}(1/\sqrt{\Delta})$ in contrast with the best known bound on the worst-case scaling of the number of Markov transitions needed in the classical approach, which goes like $\mathcal{O}(1/\Delta)$. Thus, the result appears to be a quadratic speedup over simulated annealing. We note that the $\mathcal{O}(1/\Delta)$ scaling of classical simulated annealing is known to be a very loose bound for a broad class of problems. Typically, simulated annealing is used heuristically by lowering the temperature much faster than suggested by this bound. Our results constitute the first complete cost analysis for this algorithm that involves constant factors in the complexity.

(e) *LHPST qubitized-walk-based quantum simulated annealing* (Sec. III E). Lemieux, Heim, Poulin, Svore, and Troyer (LHPST) [23] give a Metropolis-Hastings-like qubitized-walk approach, which is significantly more efficient than the direct Szegedy approach. We refer to this method by their initials, but we provide an improved technique that is efficient for more complicated problem Hamiltonians with high connectivity. LHPST consider a method that is efficient for simpler problem Hamiltonians with low connectivity, but have exponential cost for the problem Hamiltonians considered here.

(f) *Spectral-gap-amplification-based quantum simulated annealing* (Sec. III F). In Ref. [14], the authors construct an inverse-temperature-dependent Hamiltonian whose ground state in the zero-temperature limit is a superposition of solution configurations. By performing spectral-gap amplification on their Hamiltonian, they obtain a gap that is similar to that for the quantum walk approach, indicating a similar speedup. Our main purpose is to outline these techniques and summarize the work needed to execute such algorithms in general and for specific problems of interest as outlined in the Introduction. We also suggest a variant of this algorithm where one can use qubitized quantum walks rather than time evolution for the adiabatic evolution. In both cases, our results provide the first constant factor bounds on the complexity of implementing these algorithms.

We summarize the outcomes of this section in Table VII. The entries of Table VII show how the Toffoli complexity and ancilla cost of each of the above named algorithm primitives depend on the relevant costs of oracles presented in Table V. We can then use Table VII together with Table V to calculate the overall Toffoli complexity and ancilla cost of each algorithm primitive for each type of cost function. The results of this analysis are summarized in Table VIII. In giving the complexities in this table, we assume $2^{b_{\rm fun}/2} < b_{\rm sm} \log b_{\rm sm} < b_{\rm rot}$ to simplify the order terms, which is reasonable for the examples we consider in Sec. IV.

A. Amplitude amplification

1. Combining amplitude amplification with quantum optimization heuristics

All of the optimization heuristics discussed in this paper can be seen as methods of preparing a quantum state with overlap on a low-energy subspace of interest. We refer to the subspace of interest as $\mathcal{S}$. Sometimes this subspace of interest is actually the lowest-energy state (or states) and other times it is any state with energy less than a certain threshold. Furthermore, all algorithms discussed in this paper are heuristics that can be systematically refined. Let us refer to an algorithm for quantum optimization that is run for duration t as $\mathcal{U}(t)$. Let us assume that these algorithms always begin in the state $|+\rangle^{\otimes N}$ and denote the output state of the algorithm by $|\psi(t)\rangle = \mathcal{U}(t)\,|+\rangle^{\otimes N}$. Thus, after running our algorithm $\mathcal{U}(t)$ and sampling in the computational basis, the probability of measuring a state in the subspace of interest $\mathcal{S}$ is

$$p_0\left(t\right) = \sum_{x\in\mathcal{S}} \left|\langle x\left|\psi\left(t\right)\right\rangle\right|^2 . \tag{101}$$

When we say that these heuristics can be systematically refined what we mean is that we can (on average) increase $p_0(t)$ by increasing t. This refinement comes at a cost $\mathcal{C}(t) > 0$, which we define as the complexity of implementing $\mathcal{U}(t)$. This complexity is greater than zero because preparing the initial state $|+\rangle^{\otimes N}$ requires nonzero time even if we do nothing further. We can also boost the probability of seeing a state in $\mathcal{S}$ by repeating $\mathcal{U}(t)$ more times and sampling. On average we need to run our algorithm $\mathcal{U}(t)$ a number of times equal to $1/p_0(t)$ in order to see a state in $\mathcal{S}$. Thus, on average the cost to sample a state $\mathcal{S}$ is given by

$$\frac{\mathcal{C}\left(t\right)}{p_0\left(t\right)}. \tag{102}$$

There is a compromise to be reached between the duration t of the optimization heuristic $\mathcal{U}(t)$ and the success probability $p_0(t)$; heuristics run for more time can reach a higher success probability and therefore be repeated fewer times, but increasing t beyond a certain point has a negligible impact on its success probability $p_0(t)$. While past work [23] has discussed this dichotomy in terms of a minimum time to solution metric, which is parameterized in terms of a target success probability, here we focus on the mean cost to succeed because this seems more reasonable to consider in a context where $p_0(t)$ is unknown. Still, given knowledge of $p_0(t)$ one could optimize this mean time by choosing t to minimize Eq. (102). But rather than simply repeating the state preparation $1/p_0(t)$ times, one could instead boost the success probability with amplitude amplification.

Amplitude amplification is an idea that generalized Grover search and can be used to boost the probability of a marked state or subspace. For instance, we might define these marked states to be any state in $\mathcal{S}$. In this context, amplitude amplification allows us to perform a series of m reflections [involving two preparations of the state $|\psi(t)\rangle$], which boosts the probability of measuring the marked subspace to

$$p_m(t) = \sin^2\left\{(2m+1)\arcsin\left[\sqrt{p_0(t)}\right]\right\}. \tag{103}$$

For instance, if we hoped to boost the probability to 1 then by using repeated sampling we need roughly $\mathcal{O}(1/p_0(t))$ repetitions. However, by using amplitude amplification we need only

$$m \approx \frac{\pi}{4\arcsin\left[\sqrt{p_0(t)}\right]} - 1 = \mathcal{O}\left(\frac{1}{\sqrt{p_0(t)}}\right) \tag{104}$$

iterations if $p_0(t)$ is small (this is akin the usual quadratic Grover speedup).

For each round of amplitude amplification one needs to reflect about a qubit marking the subspace of interest $\mathcal{S}$. In our context the idea is to amplify either a target energy (if a target energy, e.g., the ground-state energy, is known) or to amplify all states with energy less than a certain threshold. To do this, one needs to compute the energy value into a register and perform either an equality or inequality test to determine whether we have reached a marked state. The energy can be computed simply by using the direct-energy oracles introduced in Sec. II A. However, both that step and the cost of the equality or inequality evaluation typically have a negligible additive cost to the cost $\mathcal{C}(t)$ of actually running the quantum algorithm $\mathcal{U}(t)$. Moreover, the ancilla used for storing the value of the energy can be borrowed from ancilla used in other parts of the algorithm.

For amplitude amplification to be most effective one should have an estimate of the overlap $p_0(t)$ in order to avoid "overshooting" the peak of the function in Eq. (103). Unfortunately, a reliable estimate of $p_0(t)$ is not known in advance in general. In some rare cases one might instead have a somewhat tight estimate of a lower bound to $p_0(t)$ and in those cases some advantages can be realized by using a variant of amplitude amplification known as fixed-point amplitude amplification [44]. However, one can confirm that fixed-point amplitude amplification has no advantages in our context compared to the exponential search heuristic proposed in Ref. [20] when the best lower bound that is available is $p_0(t) > 0$. The idea behind the approach in Ref. [20] is to run amplitude amplification for $m = 2^j$ iterations for $j = 0, 1, 2, 3, \ldots$ and so on until we sample a marked state. The cost of each iteration of amplitude amplification is $2\mathcal{C}(t)$ and so if we need to repeat this procedure until $m = 2^k$ it has a total cost that goes like

$$2\mathcal{C}(t)\sum_{j=0}^{k} 2^j = 2\mathcal{C}(t)\left(2^k - 1\right). \tag{105}$$

Therefore, since the probability of failure in a single run with $m = 2^j$ iterations is $1 - p_{2^j}(t)$, we see that the overall mean cost of the procedure is

$$2\mathcal{C}(t)\sum_{k=1}^{\infty}\left(2^k - 1\right)\prod_{j=1}^{k-1}\left[1 - p_{2^{j-1}}(t)\right] = \mathcal{O}\left(\frac{\mathcal{C}(t)}{\sqrt{p_0(t)}}\right). \tag{106}$$

Though the left side of this expression cannot be simplified analytically, it converges quickly and can be easily numerically computed for any $p_0(t) > 0$.

Comparing Eq. (102) to Eq. (106) we can see that there is a clear asymptotic advantage to using amplitude amplification over classical sampling and expect this advantage is realizable in practice in many contexts of interest for us. Like with Eq. (102), if one has knowledge of $p_0(t)$ then one can minimize Eq. (106) with respect to t to make the optimal tradeoff between running the algorithm $\mathcal{U}(t)$ for longer and using more rounds of amplitude amplification. In some cases it might actually be the case that the optimal choice is $t = 0$, which corresponds to using amplitude amplification directly as a heuristic for optimization. The only downside to using amplitude amplification in conjunction with other heuristic quantum algorithms for optimization is that we trade incoherent repetitions of the primitive of $\mathcal{U}(t)$ for coherent repetitions of the primitive of $\mathcal{U}(t)$. In some cases this means that we need to target a higher error rate to make the calculation fault tolerant by using an error-correcting code.

2. Directly using amplitude amplification

In the prior section we describe how amplitude amplification can be combined with any of the other optimization heuristics in this paper in order to boost overlap on a target low-energy subspace of interest. However, one can also use amplitude amplification by itself as a heuristic for optimization. This heuristic provides an interesting point of comparison to other algorithms because it offers a quadratic advantage over classical brute-force search without leveraging any structure that might be available in a particular optimization problem. Thus, it is asymptotically the optimal strategy for solving totally unstructured problems like those described by the typical Grover oracle (all computational basis states have energy zero except for a solution with energy -1) or the random energy model (all computational basis states have a unique, Gaussian distributed energy).

To use amplitude amplification on its own all one needs to do is to regard the algorithm $\mathcal{U}(t)$ as the preparation of the symmetric superposition state $|+\rangle^{\otimes N}$, which requires only Clifford gates. In the analysis of Sec. III 1 we assume that the cost of directly computing the energy and then performing the comparison operation is negligible compared to the cost of applying $\mathcal{U}(t)$ but that is not the case when we aim to directly apply amplitude amplification. Here, the main cost of a step is the cost to compute (and then later uncompute) the energy. Following Eq. (106), in this context we find that the mean cost of applying amplitude amplification directly then scales like

$$\left[2\mathcal{C}^{\text{direct}} + N + \mathcal{O}\left(b_{\text{dir}}\right)\right] \sum_{k=1}^{\infty} \left(2^k - 1\right) \prod_{j=1}^{k-1} \left\{1 - \sin^2\left[\left(2^j + 1\right) \arcsin\left(\sqrt{\frac{1}{2^N}}\right)\right]\right\} = \mathcal{O}\left(\left[\mathcal{C}^{\text{direct}} + N + b_{\text{dir}}\right] \sqrt{2^N}\right), \tag{107}$$

where we use $p_0(t) = 1/2^N$. The cost $2\mathcal{C}^{\text{direct}} + N + \mathcal{O}\left(b_{\text{dir}}\right)$ comes from cost $\mathcal{C}^{\text{direct}}$ to directly compute the energy, cost $\mathcal{O}\left(b_{\text{dir}}\right)$ to apply the inequality operator to determine whether the energy is below the target threshold, cost $\mathcal{C}^{\text{direct}}$ to uncompute the energy, and cost $N - 2$ to reflect about the equal superposition state. Note that this procedure is exactly the heuristic approach introduced in Ref. [20]. When the subspace $\mathcal{S}$ contains only a single state this algorithm reduces exactly to standard Grover search [3]. For later comparisons in this paper we refer to the cost of a single step of amplitude amplification as having

$$2\mathcal{C}^{\text{direct}} + N + \mathcal{O}\left(b_{\text{dir}}\right) \tag{108}$$

Toffoli complexity and requiring $\mathcal{A}^{\text{direct}} + \mathcal{B}^{\text{direct}} + \mathcal{O}(1)$ ancilla.

B. The quantum approximate optimization algorithm

The QAOA is another popular approach to quantum optimization, introduced in Ref. [12]. The QAOA initially attracted significant interest after it was shown to produce a better approximation ratio for a specific combinatorial optimization problem of bounded occurrence, Max E3LIN2, than any known efficient classical method [12]. While a more efficient classical algorithm was presented shortly afterwards [45], interest in QAOA has only increased since then. While bounds on the performance of QAOA are sometimes available, in most contexts it is studied as a heuristic in the sense that the intention is to use the algorithm without knowing how well it performs in practice. Part of the appeal of QAOA has been that it is an easy-to-implement algorithm that can be tested on noisy intermediate-scale quantum (NISQ) devices even before fault tolerance is available [46]. Nonetheless, QAOA is still interesting algorithm to perform within error correction.

The QAOA is more straightforward than other algorithms discussed in this work. The QAOA consists of two components that are repeatedly applied. The first component is parameterized evolution under the diagonal problem Hamiltonian C,

$$U_C\left(\gamma\right) = e^{-i\gamma C} = O^{\text{phase}}\left(\gamma\right), \tag{109}$$

where in the last equality we emphasize that $U_C(\gamma)$ is equivalent to the phase oracle $O^{\text{phase}}(\gamma)$ that we introduce and provide explicit circuit constructions for in Sec. III C.

The second component is parameterized evolution under a local transverse-field driver Hamiltonian B,

$$U_B\left(\beta\right) = e^{-i\beta B} \quad B = \sum_{j=1}^{N} X_j. \tag{110}$$

The QAOA is a variational algorithm that uses repeated application of these unitaries to prepare a parameterized state that is then optimized. The depth of the variational algorithm is usually denoted as "p" in the QAOA literature. Specifically, for depth p we prepare a state parameterized by $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$,

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U_B\left(\beta_p\right) U_C\left(\gamma_p\right) \ldots U_B\left(\beta_1\right) U_C\left(\gamma_1\right) |+\rangle^{\otimes N}, \tag{111}$$

where $|+\rangle^{\otimes N}$ is the symmetric superposition of all 2^N computational basis states.

For a given p, we attempt to find parameters that minimize the expectation value of the cost

$$\langle C\rangle = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}| C |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle. \tag{112}$$

The QAOA proposes to use the quantum computer to estimate this expectation value and then to use a classical processor to perform a classical optimization, in a fashion similar to other variational algorithms [47,48]. In general finding the globally optimal values of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ could prove to be very challenging. However, QAOA is a heuristic algorithm and the idea is that even locally optimal parameter settings might provide good approximations.

The original implementation of QAOA suggested that one directly sample the cost function C to estimate $\langle C\rangle$. Using this method, if one wishes to converge an unbiased estimator $\widetilde{\langle C\rangle}$ so that $|\widetilde{\langle C\rangle} - \langle C\rangle| \leq \Delta_C$ then the state $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ must be prepared and sampled a number of times equal to

$$\sigma^2/\Delta_C^2 \quad \text{where} \quad \sigma^2 = \langle C^2\rangle - \langle C\rangle^2. \tag{113}$$

While one does not know σ^2 in advance, one can obtain a reasonable estimate of σ^2 after only a handful of measurements and use that to determine how many more measurements are required.

The cost of QAOA is always dominated by the number of times that one must repeat the unitary $U_C(\gamma)$; the cost to implement $U_B(\beta)$ is essentially free in comparison. Thus, if J is the number of outer-loop optimization iterations, which each require a query of the energy accurate to within Δ_C then in total we require Toffoli complexity

$$\frac{pJ\mathcal{C}^{\text{phase}}\sigma^2}{\Delta_C^2}. \tag{114}$$

It is difficult to say what an appropriate choice of the quantities J and Δ_C should be as this depends on the problem, the choice of optimizer one is using, and how aggressively one is attempting to optimize. However, in many circumstances one might not need to perform the outer-loop optimization at all and can thus take $J = 1$. This is the case when optimal (or "good enough") parameters can be inferred before running the algorithm. Such a situation often arises when running large instances of optimization problems that are characteristic of a well-defined ensemble (for example, if one is running instances of the Sherrington-Kirkpatrick model). This is due to the observation that normalized energy landscapes (proportional to $\langle C\rangle$ as a function of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$) concentrate to instance and size-independent average values for large N [49,50]. Thus, surprisingly, it is possible to find the optimal values of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ by optimizing much smaller (presumably classically tractable) instances of these problems. Another possibility is that one simply use $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ parameters that are obtained from a Trotterization of the quantum adiabatic algorithm; in fact, there is evidence that these parameters become optimal as one increases p [51]. Thus, for problems where it is appropriate to forgo the outer-loop optimization step of QAOA, we can approximate the Toffoli complexity as $pM\mathcal{C}^{\text{phase}}$ where M is the number of samples we desire. The number of logical qubits required for its implementation is N (not counting any extra ancilla used for the phase oracle).

Within the context of NISQ computations it makes sense to use this method of sampling to estimate the cost function expectation value and then to perform the optimization on a classical computer. The reason is because both strategies minimize the size of each quantum circuit that must be executed, although potentially at a cost of needing a larger number of repetitions compared to other strategies. However, within cost models appropriate for fault tolerance the primary resource to consider is the total number of gates required by the computation and no particular distinction is made whether those gates are involved in repeated applications of short quantum circuits or a single application of a longer quantum circuit. Thus, on a fault-tolerant quantum computer it may make sense to consider more elaborate versions of QAOA in which the expectation-value estimation and potentially even the optimization is also performed on a quantum computer. For instance, perhaps the variational parameters $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ can be stored in a quantum register on which the QAOA unitary is controlled. Such a scheme is considered in Ref. [52] where it is shown that such a method can enable quadratically faster resolution of the gradient than otherwise required, however with significant constant overhead. Similarly, by using the amplitude amplification based Monte Carlo techniques discussed in Ref. [53] (see Theorem 5 therein) one can reduce the number of state preparations needed for an estimate of the cost function to $\mathcal{O}((\sigma/\Delta_C)\log^{3/2}(\sigma/\Delta_C)\log\log(\sigma/\Delta_C))$, an almost quadratic improvement over the naive sampling strategy. However, as that method requires a number of copies of the system register scaling as $\mathcal{O}(\log(\sigma/\Delta_C)\log\log(\sigma/\Delta_C))$, it might prove to be prohibitively expensive for realization on small fault-tolerant quantum computers. We now consider two alternative ways to measure the energy in QAOA, which might prove more practical for small fault-tolerant quantum computers.

1. *Amplitude-estimation-based direct-phase oracle evaluation*

Apart from sampling, the next most natural algorithm for estimating the energy is using amplitude estimation to compute the expectation value of each term in the cost function in sequence. Let us assume that the cost function takes the form, $C = \sum_{\ell=1}^{L} w_\ell U_\ell$, where U_ℓ is a unitary operator (and is typically a sum of diagonal Pauli operators), as in Eq. (51). Further we take $\lambda = \sum_\ell |w_\ell|$. The algorithm that we employ is simple, for each ℓ from 1 to L we compute the quantity $\langle\psi|\, U_\ell\, |\psi\rangle$ within error $\Delta_C/(L|w_\ell|)$. An unbiased estimate of the cost function is then given by $\sum_\ell w_\ell \langle\psi|\, U_\ell\, |\psi\rangle$ and from the triangle inequality the error is at most Δ_C.

An estimate of $\langle\psi|\, U_\ell\, |\psi\rangle$ can be obtained by performing the Hadamard test (as shown in Fig. 6). Specifically, the probability of measuring the ancillary qubit to be zero is $(1 + \mathrm{Re}(\langle\psi|\, U_\ell\, |\psi\rangle))/2$. If amplitude amplification is used to mark the zero state for this circuit then the eigenphases of the resultant walk operator (within the two-dimensional space spanned by the initial state and the marked state) is

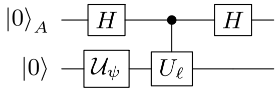

FIG. 6. Hadamard test circuit for computing the expectation value of one of the terms in the cost function. Here $\mathcal{U}_\psi$ is a unitary operation that prepares the ansatz state: $\mathcal{U}_\psi\,|0\rangle = |\psi\rangle$.

[20]

$$\phi = \pm 2\arcsin(\sqrt{P_0}) = \pm 2\arcsin\left[\sqrt{\frac{1+\mathrm{Re}\left(\langle\psi|\,U_\ell\,|\psi\rangle\right)}{2}}\right]. \tag{115}$$

We then have that

$$2\sin^2(\phi/2) - 1 = \mathrm{Re}(\langle\psi|\,U_\ell\,|\psi\rangle). \tag{116}$$

From calculus we then see that

$$\partial_\phi[2\sin^2(\phi/2)] = 2\sin(\phi/2)\cos(\phi/2) \le \phi. \tag{117}$$

Thus from Taylor's remainder theorem we have that for any $\delta \ge 0$

$$|2\sin^2(\phi/2) - 2\sin^2[(\phi+\delta)/2]| \le \delta \tag{118}$$

and if $\phi \to \phi + \delta$ for some error δ we have that the uncertainty that propagates to the expectation value is at most

$$|\mathrm{Re}(\langle\psi|\,U_\ell\,|\psi\rangle) - \mathrm{Re}(\langle\psi|\,U_\ell\,|\psi\rangle)_{\mathrm{est}}| \le \delta. \tag{119}$$

Therefore, if we wish to estimate the energy of a configuration within error ϵ it suffices to use phase estimation with an error of ϵ on the Grover operator. Finally, as discussed above we take $\epsilon = \Delta_C/(L|w_\ell|)$ to ensure that the error sums up to Δ_C as required.

Using the quantum-Fourier-transform- (QFT) based phase-estimation algorithm in Ref. [26] we find that, if we neglect the cost of the QFT and any additional costs due to additional precision required in the QROM then the number of queries to the Grover oracle needed is (for $\epsilon \le \pi$) $2^m \le 2\lfloor\pi/\epsilon\rfloor \le 2\pi/\epsilon$. Here the factor of 2 comes from the fact that the need to round to a power of 2 leads to, in the worst-case scenario, a factor of 2 in the number of iterations required.

Next the Grover oracle requires two reflection operators, one that reflects about the state yielded by the Hadamard test circuit and another that reflects about the target space, which is marked by the top qubit in Fig. 6 being zero (i.e., $R_0 = \mathbb{1} - 2\,|0\rangle\langle 0| \otimes \mathbb{1}$). The Grover walk operator is a product of these two operators $W = -R_1R_0$ and as a result, if we neglect the cost of the additional Hadamard and Toffoli gates needed to implement the conditional phase flip, the costs of this process are entirely due to the reflection about the initial state, which requires two applications of the preparation of the initial state. We further follow the assumption in the previous section that the cost of state preparation dwarfs the cost of applying prepare or select. Thus under these assumptions, and taking the uncertainty in the objective function to be Δ_C the Toffoli complexity for the entire simulation is approximately

$$\sum_{\ell=1}^{L} \frac{4pJ\pi\,|w_\ell| LC^{\mathrm{phase}}}{\Delta_C} = \frac{4pJ\pi\lambda LC^{\mathrm{phase}}}{\Delta_C}. \tag{120}$$

Thus, under these assumptions, direct-energy evaluation yields an advantage over sampling if

$$\sigma^2 \ge 4\pi\lambda\Delta_C L. \tag{121}$$

We expect this to occur when the error tolerance is small and the number of terms is relatively modest. On the other hand if the variance is small, target uncertainty is large, or L is large then sampling is preferable to the direct-phase oracle-evaluation process.

2. Amplitude-estimation-based LCU evaluation

One inexpensive approach that can be used to estimate the expectation value comes from combining the Hadamard test circuit and amplitude estimation [20]. Here we use a slightly generalized form of a generalized Hadamard test circuit shown in Fig. 7. The expectation value of the first qubit for the above circuit is $1/2 + \mathrm{Re}(\langle\psi|\,C\,|\psi\rangle)/2$. In order to see this, consider the following,

$$\begin{aligned}|0\rangle\,|0\rangle\,|\psi\rangle &\mapsto |0\rangle\left(\sum_\ell \sqrt{\frac{w_\ell}{\lambda}}\,|\ell\rangle\right)|\psi\rangle \mapsto \frac{|0\rangle + |1\rangle}{\sqrt{2}}\left(\sum_\ell \sqrt{\frac{w_\ell}{\lambda}}\,|\ell\rangle\right)|\psi\rangle \\ &\mapsto \frac{|0\rangle}{\sqrt{2}}\left(\sum_\ell \sqrt{\frac{w_\ell}{\lambda}}\,|\ell\rangle\right)|\psi\rangle + \frac{|1\rangle}{\sqrt{2}}\left(\sum_\ell \sqrt{\frac{w_\ell}{\lambda}}\,|\ell\rangle\,U_\ell\,|\psi\rangle\right)\end{aligned}$$

$$\begin{aligned}
&\mapsto \frac{|0\rangle}{2}\left[(\text{PREPARE}\,|0\rangle)\,|\psi\rangle + \sum_\ell \sqrt{\frac{w_\ell}{\lambda}}\,|\ell\rangle\, U_\ell\,|\psi\rangle\right] + \frac{|1\rangle}{2} \\
&\times \left[(\text{PREPARE}\,|0\rangle)\,|\psi\rangle - \sum_\ell \sqrt{\frac{w_\ell}{\lambda}}\,|\ell\rangle\, U_\ell\,|\psi\rangle\right] \\
&\mapsto \frac{|0\rangle}{2}\left(|0\rangle\,|\psi\rangle + \text{PREPARE}^\dagger \sum_\ell \sqrt{\frac{w_\ell}{\lambda}}\,|\ell\rangle\, U_\ell\,|\psi\rangle\right) + \frac{|1\rangle}{2}\left(|0\rangle\,|\psi\rangle - |\text{junk}\rangle\right).
\end{aligned} \tag{122}$$

Therefore, the probability of measuring 0 in the top-most qubit in Fig. 7 is

$$\begin{aligned}
&\frac{1}{4}\left[2 + \langle 0|\,\langle\psi|\,\text{PREPARE}^\dagger \sum_\ell \sqrt{\frac{w_\ell}{\lambda}}\,|\ell\rangle\, U_\ell\,|\psi\rangle + \left(\langle 0|\,\langle\psi|\,\text{PREPARE}^\dagger \sum_\ell \sqrt{\frac{w_\ell}{\lambda}}\,|\ell\rangle\, U_\ell\,|\psi\rangle\right)^*\right] \\
&= \frac{1 + \text{Re}\left(\frac{\langle\psi|C|\psi\rangle}{\lambda}\right)}{2}.
\end{aligned} \tag{123}$$

If amplitude estimation is used, the number of invocations of PREPARE and SELECT needed to estimate this probability within ϵ error is $\mathcal{O}(\lambda/\epsilon)$, which is a quadratic improvement over the sampling bound in Eq. (113).

Following the same reasoning used to derive Eq. (119) we find that the overall Toffoli count is then, under the assumptions that the Toffoli count is dominated by applications of the PREPARE, SELECT, and phase-circuit operations and further that the cost of adding an additional control to SELECT is negligible, given by

$$\frac{4\pi p J \lambda(\mathcal{C}_{\text{phase}} + \mathcal{C}_{\text{Sel}} + 2\mathcal{C}_{\text{Prep}})}{\Delta_C}. \tag{124}$$

Here $\mathcal{C}_{\text{Sel}}$ and $\mathcal{C}_{\text{Prep}}$ are the Toffoli counts for SELECT and PREPARE, respectively.

Equation (124) shows that the favorable scalings of the sampling approach and the direct-phase evaluation methods can be combined together in a single method. However, this advantage comes potentially at the price of a worse prefactor owing to the additional complexity of the PREPARE and SELECT circuits. In particular, we find that this approach is preferable to sampling and direct-phase estimation, respectively, when

$$\sigma^2 \geq 4\pi\lambda\Delta_C\left(\frac{\mathcal{C}_{\text{phase}} + \mathcal{C}_{\text{Sel}} + 2\mathcal{C}_{\text{Prep}}}{\mathcal{C}_{\text{phase}}}\right), \tag{125}$$

$$L \geq \frac{\mathcal{C}_{\text{phase}} + \mathcal{C}_{\text{Sel}} + 2\mathcal{C}_{\text{Prep}}}{\mathcal{C}_{\text{Phase}}}. \tag{126}$$

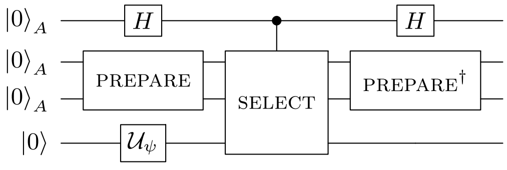

FIG. 7. Generalized form of a Hadamard test that we use for our QAOA implementation using LCU oracles. Here $\mathcal{U}_\psi$ is a unitary operation that prepares the ansatz state: $\mathcal{U}_\psi\,|0\rangle = |\psi\rangle$.

In general, we suspect that in fault-tolerant settings this approach is preferable to direct-phase oracle evaluation because the costs of the prepare and select circuits is often comparable, or less than, that of $\mathcal{U}_\psi$ as we see in the following section where we provide explicit constructions for the PREPARE and SELECT oracles.

C. Adiabatic quantum optimization

1. Background on the adiabatic algorithm

The adiabatic algorithm [54] works by initializing a system as an easy-to-prepare ground state of a known Hamiltonian, and then slowly (adiabatically) deforming that system Hamiltonian into the Hamiltonian whose ground state we wish to prepare. For instance, we might use a Hamiltonian parameterized by $s \in [0, 1]$,

$$H(s) = (1 - s)H_0 + sH_1, \tag{127}$$

where H_0 is a Hamiltonian with an easy-to-prepare ground state and H_1 is a Hamiltonian whose ground state we wish to prepare. We start the system in the ground state of $H(0) = H_0$ and then slowly deform the Hamiltonian by increasing s from 0 to 1 until $H(1) = H_1$. If this is performed slowly enough, then the system is in the ground state of H_1 at the end of the evolution.

The main challenge with the adiabatic algorithm is that we may need to turn s on extremely slowly in order for the procedure to succeed. The rate at which we can turn on s depends on features of the spectrum of $H(s)$, including its derivatives and the minimum gap Δ between the ground-state eigenvalue and first excited-state eigenvalue. It is often empirically observed that the total time of the evolution T should scale as $\mathcal{O}(1/\Delta^2)$. Indeed, this result has been proven using the so-called boundary adiabatic theorem. This result analyzes the adiabatic algorithm in terms of phase randomization between the different paths that describe quantum dynamics for a slowly varying time-dependent Hamiltonian. This randomization causes paths that lead different excitations to destructively interfere, which effects a mapping from the eigenvectors of an initial Hamiltonian to the corresponding eigenvectors of the target Hamiltonian in the limit of slow evolution relative to a relevant gap in the instantaneous eigenvalues of the time-dependent Hamiltonian. The boundary adiabatic theorem holds that if we let $|\psi_k(s)\rangle$ be the kth instantaneous eigenvector of any Gevrey-class time-dependent $H(s)$ then we have that [55]

$$\left\| \mathcal{T} e^{-i\int_0^1 H(x)T dx} |\psi_k(0)\rangle - |\psi_k(1)\rangle \right\| \in \widetilde{\mathcal{O}}\left(\frac{1}{\Delta^2 T}\right), \quad (128)$$

where Δ is the minimum eigenvalue gap between the state $|\psi_k(s)\rangle$ and the remainder of the spectrum. It then follows if we pick an appropriate value for $T \in \mathcal{O}(1/\Delta^2\epsilon)$ then we can make the error less than ϵ for an arbitrary gapped adiabatic path. Alternatively, if very high precision is required then the time required for adiabatic state preparation can also be improved for analytic Hamiltonians to $\widetilde{\mathcal{O}}(\mathrm{poly}(\|\dot{H}\|, \|\ddot{H}\|, \ldots)(1/\Delta^2 + \log(1/\epsilon)/\Delta))$ by adaptively choosing the adiabatic path to have to obey $\|\partial_s^q H(0)\| = \|\partial_s^q H(1)\| = 0$ for all positive integers less than $Q(\epsilon) \in \mathcal{O}(\log(1/\epsilon))$; however, this approach requires small error tolerance on the order of $\epsilon \in \mathcal{O}(\Delta)$ in order to see the benefits of these improved adiabatic paths [56–58].

Note that the boundary adiabatic theorem only tells us about the state at the end of the evolution, and does not actually tell us anything about the state we are in at the middle of the evolution. For that there are “instantaneous” adiabatic theorems, which bound the probability of being in the ground state throughout the entire evolution. For instance, one such way to show this is based on the Zeno-stabilized adiabatic evolutions described in Sec. III 3 [21]. These instantaneous adiabatic theorems have complexity $\mathcal{O}(L^2/(\epsilon\Delta))$, where

$$L = \int_0^1 \|\dot{\psi}(s)\| ds \quad (129)$$

is the path length. In the case of simulated annealing, one can show that the path length is independent of Δ, whereas in general the worst-case bound is $L \leq \|\dot{H}\|/\Delta$, which yields $\mathcal{O}(\|\dot{H}\|^2/\Delta^3)$ complexity [21]. It is not completely clear which style of adiabatic evolution gives the best results when using the approach as a heuristic, and so we discuss both here. With either approach we typically take H_1 to be the cost function of interest and take H_0 to be a simple-to-implement Hamiltonian that does not commute, with an easy-to-prepare ground state. For instance, a common choice is to take $H_0 = \sum_{i=1}^N X_i$ where X_i is the Pauli-X operator, so that the initial state is $|+\rangle^{\otimes N}$. Other H_0 Hamiltonians (or more complicated adiabatic paths) are also possible.

The simplest way to use the adiabatic algorithm as a heuristic is to discretize the evolution using product formulas. For instance, if we assume the adiabatic schedule in Eq. (127) then we could attempt to prepare the ground state as

$$\prod_{k=1}^{M} \exp\left[-i\left(\frac{M-k}{M^2}\right)H_0 T\right] \exp\left[-i\left(\frac{k}{M^2}\right)H_1 T\right] |\psi_0(0)\rangle, \quad (130)$$

where M is the number of first-order Trotter steps used to discretize the adiabatic evolution. The idea of the heuristic is to choose M based on available resources. T also needs to be chosen heuristically rather than based on knowledge of the gap, which we do not expect to have in general. For fixed M, smaller T enables more precise approximation of the continuous-time algorithm, but smaller T also means the system is less likely to stay adiabatic.

Of course, one can also easily extend this strategy to using higher-order product formulas, or to using either different adiabatic interpolations or adiabatic paths. For example, if we define

$$U_2\left(\frac{k-1}{M}, \frac{k}{M}\right) = \exp\left[-i\left(\frac{M-k-1/2}{2M^2}\right)H_0 T\right] \times \exp\left[-i\left(\frac{k+1/2}{M^2}\right)H_1 T\right] \exp\left[-i\left(\frac{M-k-1/2}{2M^2}\right)H_0 T\right], \quad (131)$$

then we have that $\left\|\prod_{k=1}^M U_2\left(\frac{k-1}{M}, \frac{k}{M}\right) - \mathcal{T}\exp[-i\int_0^T H(t)dt]\right\| \in \mathcal{O}(T^3/M^2)$. Higher-order versions of such integrators of order can be formed via Suzuki’s recursive construction (for any $s \in [0,1]$):

$$\begin{aligned} U_\rho(s, s+\delta) := {} & U_{\rho-2}[s+(1-\gamma_\rho)\delta, s \\ & + \delta]U_{\rho-2}[s+(1-2\gamma_\rho)\delta, s+(1-\gamma_\rho)\delta] \\ & \times U_{\rho-2}[s+2\gamma_\rho\delta, s+(1-2\gamma_\rho)\delta]U_{\rho-2}(s \\ & + \gamma_\rho\delta, s+2\gamma_\rho\delta)U_{\rho-2}(s, s+\gamma_\rho\delta). \quad (132) \end{aligned}$$

Here $\gamma_\rho = (4 - 4^{1/(\rho-1)})^{-1}$, which approaches $1/3$ as the order of the formula, ρ, goes to infinity. Furthermore, we have that the error in the Trotter-Suzuki algorithm scales as

$$\left\|\prod_{k=1}^{M} U_\rho\left(\frac{k-1}{M},\frac{k}{M}\right) - \mathcal{T}\exp[-i\int_0^T H(t)dt]\right\| \in \mathcal{O}\left(T^{\rho+1}/M^\rho\right), \tag{133}$$

which results in near linear scaling with T in the limit as T approaches infinity.

In practice, however, since the number of exponentials in the Trotter-Suzuki formula grows exponentially with the order there is in practice an optimal tradeoff in gate complexity that is satisfied by a finite-order formula for a fixed ϵ and T. For simplicity, we assume that the time evolution under H_0 is much cheaper to implement than the time evolution under H_1. As H_1 can be implemented using the phase oracles O^{phase} discussed in Sec. II, the total cost of the procedure is approximately $M\mathcal{C}^{\text{phase}}$. This implies that, for a finite value of M, the cost of performing the heuristic optimization using the above adiabatic sequence is approximately

$$\mathcal{C}_{\text{adiabatic}} = 2M5^{\rho/2-1}\mathcal{C}^{\text{phase}}. \tag{134}$$

Again, assuming that our target error in the adiabatic sweep is ϵ and $\Delta^2 \in \mathcal{O}(\epsilon)$ then it suffices to take $T \in \mathcal{O}(1/\Delta^2\epsilon)$ and further after optimizing the cost by setting M equal to $2^{\rho/2-1}$ we find that $M \in (T^{1+o(1)}/\epsilon^{o(1)})$. Therefore, the total cost obeys

$$\mathcal{C}_{\text{adiabatic}} \in \frac{\mathcal{C}_{\text{phase}}}{(\epsilon\Delta^2)^{1+o(1)}}. \tag{135}$$

Similarly, if we are interested in the limit where $\Delta^2 \in \omega(\epsilon)$, then boundary cancellation methods [56,57] can be used to improve the number of gates needed to reach the global optimum to

$$\mathcal{C}_{\text{adiabatic}} \in \frac{\mathcal{C}_{\text{phase}}\log^{1+o(1)}(1/\epsilon)}{\Delta(\epsilon\Delta)^{o(1)}}. \tag{136}$$

These results show that, provided the eigenvalue gap is polynomial, we can use a simulation routine for $e^{-iH_1(t)}$ and $e^{-iH_0(t)}$ to find the local optimum in polynomial time. However, in practice we are likely to want to use such an algorithm in a heuristic fashion wherein the timesteps do not precisely conform to the adiabatic schedule.

To give the cost for a single step a little more precisely, we can also include the cost of implementing a transverse driving field. Since that involves applying a phase to b_{pha} bits to each of N qubits, using repeat-until-success circuits, this has cost $1.15Nb_{\text{pha}} + \mathcal{O}(N)$ in terms of T gates, with a single ancilla qubit. It is also possible to sum the bits with Toffoli cost N, then phase by the sum with cost ${b_{\text{pha}}}^2/2 + \mathcal{O}(b_{\text{pha}}\log b_{\text{pha}})$ (accounting for multiplying the phase by a constant factor), though that has large ancilla cost. Using the sum of tree sums approach gives complexity $4N + {b_{\text{pha}}}^2/2 + \mathcal{O}(b_{\text{pha}}\log b_{\text{pha}})$, with $3\log L + \mathcal{O}(1)$ temporary ancillas. There is $b_{\text{grad}} = b_{\text{pha}} + \mathcal{O}(\log b_{\text{pha}})$ persistent ancillas needed for a phase-gradient state as well, but in many cases that state is the same as in other steps of the procedure, so does not increase the ancilla cost. Using this approach, and omitting the factor of $2 \times 5^{\rho/2-1}$ for order ρ Suzuki, gives Toffoli cost

$$\mathcal{C}^{\text{phase}} + 4N + {b_{\text{pha}}}^2/2 + \mathcal{O}(b_{\text{pha}}\log b_{\text{pha}}) \tag{137}$$

for a single step.

2. Heuristic adiabatic optimization using quantum walks

While the procedure we describe for heuristically using the adiabatic algorithm with Trotter-based methods is well known, it is less clear how one might heuristically use LCU methods with the adiabatic algorithm. One reason we might try doing this is because the qubitized quantum walks that we discuss in Sec. II are sometimes cheaper to implement than Trotter steps for some problems. One approach to using LCU methods for adiabatic state preparation might be to directly attempt to simulate the time-dependent Hamiltonian evolution using a Dyson series approach, as was recently suggested for the purpose of adiabatic state preparation in Ref. [59]. However, this requires fairly complicated circuits due to the many time registers that one must keep to index the time-ordered exponential operator. In principle, we could always use quantum-signal processing (or more generally quantum singular value transformations) to convert the walk operator at time t into the form $e^{-iH(t)\delta}$ for some timestep δ.

Instead, here we suggest a strategy, which is something of a combination between using qubitized quantum walks and using a product formula approximation. Our method is unlikely to be asymptotically optimal for this purpose but it is simple to implement and we suspect it is cheaper than either a Dyson series approach or a Trotter approach for some applications on a small error-corrected quantum computer. The idea is to stroboscopically simulate time evolution as a short-time evolved "qubitized" walk. The result is that we actually simulate the adiabatic path generated by the arccosine of the normalized Hamiltonian $H(s)$ rather than the adiabatic path generated directly by $H(s)$, but we expect that the relevant part of the eigenspectrum is in the linear part of the arccosine, which means there is not much effect on the dynamics. The main challenge in this approach is to artificially shrink the effective duration of these quantum walk steps so that the method can be refined.

In the following we assume that $\text{SELECT}^2 = 1$, which is to say that every Hamiltonian in the decomposition is self-adjoint (consistent with the problem Hamiltonians we consider). For every eigenvector $|\psi_k(t)\rangle$ of $H(t)$ with $H\,|\psi_k(t)\rangle = E_k(t)$, if we define $|L\rangle = \textsc{prepare}\,|0\rangle$ then we can write

$$W = (I - 2I \otimes |L\rangle\langle L|)\textsc{select}. \tag{138}$$

The walk operator can be seen as a direct sum of two different walk operators, $W = W_H \oplus W_\perp$, where W_H is the portion of the walk operator that acts nontrivially on $|\psi_k(t), L\rangle = |\psi_k(t)\rangle \otimes |L\rangle$ and $W_\perp$ is the operator that acts on the remaining states. Next, if for each k and t we define $|\psi_k^\perp(t)\rangle$ such that

$$\begin{aligned}|\psi_k^\perp(t)\rangle &= \frac{\left[W - \frac{E_k(t)}{\lambda(t)}\right]|\psi_k(t), L\rangle}{\sqrt{1 - \frac{E_k^2(t)}{\lambda^2(t)}}}\\ &= \frac{\left[W_H - \frac{E_k(t)}{\lambda(t)}\right]|\psi_k(t), L\rangle}{\sqrt{1 - \frac{E_k^2(t)}{\lambda^2(t)}}},\end{aligned} \tag{139}$$

then we can express

$$W_H(t) = \exp\left\{-i\left[\sum_k i\,|\psi_k^\perp(t)\rangle\langle\psi_k(t), L| - i\,|\psi_k(t), L\rangle\langle\psi_k^\perp(t)|\right]\arccos\left[\frac{E_k(t)}{\lambda(t)}\right]\right\}. \tag{140}$$

It may be unclear how to implement a time step for $W(t)$ since the operation is only capable of applying unit-time evolutions. Fortunately, we can address this by taking for any $r \geq 1$

$$H(t) = \sum_k \lambda_k(t) U_k \mapsto \sum_k \lambda_k(t) U_k + \frac{(r-1)\lambda(t)}{2}\,(I - I)\,. \tag{141}$$

In this case we can block encode the Hamiltonian using a unary encoding of the extra two operators via

$$|L(t,r)\rangle = \sum_k \sqrt{\frac{\lambda_k(t)}{\lambda(t) r}}\,|k\rangle\,|00\rangle + \sqrt{\frac{r-1}{2r}}\,|0\rangle\,(|10\rangle\, +\, |11\rangle)\,. \tag{142}$$

The select oracles for this Hamiltonian require one additional control for each of the original terms in the Hamiltonian and the additional terms only need a single Pauli-Z gate to implement. We define this operator to be $\textsc{select}'$.

With these two oracles defined, we can then describe the walk operator $W_r(t)$ for any fixed value of t to be

$$W_r(t) = [I - 2I \otimes |L(t,r)\rangle\langle L(t,r)|]\textsc{select}'. \tag{143}$$

This new Hamiltonian has exactly the same eigenvectors, however its value of λ is greater by a factor of r. In particular, we can express the walk operator [restricted to the eigenspace supported by the instantaneous eigenvectors of $H(t)$] is

$$W_{H,r}(t) = \exp\left(\left\{-i\left[\sum_k i\,|\psi_k^\perp(t)\rangle\langle\psi_k(t), L(t,r)| - i\,|\psi_k(t), L(t,r)\rangle\langle\psi_k^\perp(t)|\right] r\arccos\left[\frac{E_k(t)}{r\lambda(t)}\right]\right\}\frac{1}{r}\right). \tag{144}$$

Using the fact that $\arccos(x) = \pi/2 - \arcsin(x)$ we have that, up to an irrelevant global phase this operator can be written as

$$V_{H,r}(t) = \exp\left(\left\{i\left[\sum_k i\,|\psi_k^\perp(t)\rangle\langle\psi_k(t), L(t,r)| - i\,|\psi_k(t), L(t,r)\rangle\langle\psi_k^\perp(t)|\right] r\arcsin\left[\frac{E_k(t)}{r\lambda(t)}\right]\right\}\right). \tag{145}$$

Thus the operator $V_{H,r}(t)$ can be seen to generate a short time step of duration $1/r$ for an effective Hamiltonian

$$H_r(t) := \left[\sum_k i\,|\psi_k^\perp(t)\rangle\langle\psi_k(t), L(t,r)| - i\,|\psi_k(t), L(t,r)\rangle\langle\psi_k^\perp(t)|\right] r\arcsin\left[\frac{E_k(t)}{r\lambda(t)}\right]. \tag{146}$$

Note that as $r \to \infty$ the eigenvalues of this Hamiltonian approach $\pm E_k(t)/\lambda(t)$ and more generally

$$\left| r \sin^{-1}\{E_k(t)/[r\lambda(t)]\} - E_k(t)/\lambda(t) \right| \in O(1/r^2).$$

For any fixed value of r we can choose an adiabatic path between an initial Hamiltonian and a final Hamiltonian. The accuracy of the adiabatic approximation depends strongly on how quickly we traverse this path so it is customary to introduce a dimensionless time $s = t/T$, which allows us to easily change the speed without altering the shape of the adiabatic path. In Appendix B we are able to show that the adiabatic theorem then implies that the number of steps of the quantum walk required to achieve error ϵ in an adiabatic state preparation for a maximum rank Hamiltonian with gap Δ is in

$$\widetilde{\mathcal{O}}\left[\frac{1}{\epsilon^{3/2}}\sqrt{\frac{\max_s\left(\|\ddot{H}\| + |\ddot{\lambda}|\right)\max_s\left(|\dot{\lambda}| + \|\dot{H}\|\right)}{\min(\Delta, \min_k |E_k|)^2} + \frac{\lambda \max_s\left(|\dot{\lambda}| + \|\dot{H}\|\right)^3}{\min(\Delta, \min_k |E_k|)^4}}\right]. \tag{147}$$

The reason why this result depends on the minimum value of E_k is an artifact of the fact that several of the eigenvalues of the walk operator can be mapped to 1 under repeated application of W_r. This potentially can alter the eigenvalue gaps for eigenvalues near zero, which impacts the result.

The key point behind this scaling is that it shows that as the number of time slices increases this heuristic converges to the true adiabatic path. Just as the intuition behind Trotterized adiabatic state preparation hinged on this fact, here this result shows that we can similarly use a programmable sequence of parameterizable walk operators to implement the dynamics. The main advantage relative to Trotter methods is that the price that we have to pay using this technique does not depend strongly on the number of terms in the Hamiltonian, which can lead to advantages in cases where the problem or driver Hamiltonians are complex.

This scaling can be improved by using higher-order splitting formulas for the time evolution [60] and by using boundary cancellation methods to improve the scaling of the error in adiabatic state preparation. In general, if we assume that $\Delta \in \mathcal{O}(1)$ for the problem at hand then it is straightforward to see that we can improve the scaling from $\mathcal{O}(1/\epsilon^{3/2})$ to $1/\epsilon^{o(1)}$ [56–58]. It is also worth noting that the bounds given above for the scaling with respect to the derivatives of the Hamiltonian and the coefficients of the Hamiltonian is expected to be quite loose owing to the many simplifying bounds used to make the expression easy to use. On the other hand, the scaling with the gap and error tolerance is likely tighter.

3. Zeno projection of adiabatic path via phase randomization

The principle of the Zeno approach is to increment the parameter for the Hamiltonian s or β by some small amount such that the overlap of the ground state of the new Hamiltonian with that of the previous Hamiltonian is small. One can then perform phase estimation to ensure that the system is still in the ground state. This approach was used in Refs. [23,61], and combined with a rewind procedure to give a significant reduction in gate complexity compared to other approaches. An alternative approach was proposed in Ref. [21,62], where the measurement was replaced with phase randomization. Here we summarize this method and show how to further optimize it.

When using phase estimation, if it verifies that the system is still in the ground state, one continues with incrementing the parameter. If the ground state is not obtained from the phase estimation, one could abort, in which case no output is given and one needs to restart. Because the probability of failure is low, one could just continue regardless, and check at the end. That means that the result of the phase estimation is discarded.

The phase estimation is performed with control qubits controlling the time of the evolution, then an inverse quantum Fourier transform on the control qubits to give the phase. But, if the result of the measurement is ignored, then one can simply ignore the inverse quantum Fourier transform, and regard the control qubits as being measured in the computational basis and the result discarded. That is equivalent to randomly selecting values for these control qubits in the computational basis at the beginning. But, if these qubits take random values in the computational basis, one can instead just classically randomly generate a time, and perform the evolution for that time.

In performing a phase measurement using control qubits, one uses a superposition state on those control qubits, and the error in the phase measurement corresponds to the Fourier transform of those amplitudes. That is, with b control qubits, we have a state of the form

$$|\chi_\phi\rangle = \sum_{z=0}^{2^b-1} e^{iz\phi}\chi_z\,|z\rangle\,, \tag{148}$$

where ϕ is a phase that corresponds to $-E\delta t$, the energy eigenvalue of the Hamiltonian times the shortest evolution time. Then the phase measurement using the quantum inverse Fourier transform corresponds to the positive operator-valued measurement (POVM) $|\hat{\phi}\rangle\,\langle\hat{\phi}|$, with

$$|\hat{\phi}\rangle = \frac{1}{\sqrt{2\pi}}\sum_{z=0}^{2^b-1} e^{iz\hat{\phi}}\,|z\rangle\,. \tag{149}$$

The probability distribution for the error $\delta\phi = \hat{\phi} - \phi$ is then given by

$$\Pr(\delta\phi) = \left|\langle\hat{\phi}|\chi_\phi\rangle\right|^2 = \frac{1}{2\pi}\left|\sum_{z=0}^{2^b-1} e^{iz\delta\phi}\chi_z\right|. \tag{150}$$

These measurements are equivalent to the theory of window functions in spectral analysis. A particularly useful window to choose is the Kaiser window, because it has exponential suppression of errors [63].

In the case where the evolution time is chosen classically, it can be given by a real number, and we do not need any bound on the evolution time. Then the the expected cost is the expectation value of $|t|$

$$\langle |t|\rangle = \int dt|t|p_{\text{time}}(t). \tag{151}$$

Because there is no upper bound on t, we can obtain a probability distribution for the error that drops strictly to zero outside the given interval, rather than being exponentially suppressed. Still considering a coherent superposition for the moment, the state is given by

$$|\psi_E\rangle = \int dt e^{-iEt}\chi_t\,|t\rangle\,, \tag{152}$$

where E is the energy, t is the evolution time, and $p_{\text{time}}(t) = |\chi_t|^2$. Then the POVM is $|\hat{E}\rangle\,\langle\hat{E}|$ with

$$|\hat{E}\rangle = \frac{1}{\sqrt{2\pi}}\int dt e^{-i\hat{E}t}\,|t\rangle\,. \tag{153}$$

The probability distribution for the error in the measurement of E is

$$\Pr(\delta E) = \frac{1}{2\pi}\left|\int dt e^{it\delta E}\chi_t\right|^2. \tag{154}$$

An alternative description is to describe the system as being in state

$$|\psi\rangle = \sum_j \langle\psi_j|\psi\rangle\,|\psi_j\rangle\,, \tag{155}$$

where $|\psi_j\rangle$ is an eigenstate of the Hamiltonian with energy E_j. Then evolving for time t with probability $p_{\text{time}}(t)$ gives the state

$$\sum_{j,k}\langle\psi_j|\psi\rangle\langle\psi|\psi_k\rangle\tilde{p}_{\text{time}}(E_j - E_k)\,|\psi_j\rangle\langle\psi_k|\,, \tag{156}$$

where

$$\tilde{p}_{\text{time}}(E_j - E_k) = \int dt p_{\text{time}}(t)e^{-i(E_j-E_k)t}. \tag{157}$$

If the width of the Fourier transform of the probability distribution p_{time} is less than the spectral gap Δ, then the state is

$$\sum_j |\langle\psi_j|\psi\rangle^2\,|\psi_j\rangle\langle\psi_j|\,. \tag{158}$$

In comparison, if $\Pr(\delta E)$ is equal to zero for $|\delta E| \le E_{\max}$, then the same result is obtained for $2E_{\max} = \Delta$. This is what is expected, because if $p_{\text{time}}(t) = \chi_t^2$, then the Fourier transform of p_{time} is the autocorrelation of the Fourier transform of χ_t, and therefore has twice the width.

Next we consider appropriate probability distributions. A probability distribution for t that was suggested in Ref. [21] was

$$p_{\text{time}}(t) = \frac{8\pi\,\text{sinc}^4(t\Delta/4)}{3\Delta}. \tag{159}$$

That gives $\langle|t|\rangle = 12\ln 2/(\pi\Delta)$, so $\langle|t|\rangle\Delta \approx 2.648$. There $\Pr(\delta E)$ is equivalent to the square of a triangle window, but greater performance can be obtained by using the triangle window

$$\Pr(\delta E) = \frac{2}{\Delta}(1 - |2\delta E/\Delta|). \tag{160}$$

Then the corresponding ψ_t is obtained from the Fourier transform of $\sqrt{\Pr(\delta E)}$ as

$$\chi_t = \frac{\sin(\Delta t/2)C(\sqrt{\Delta t/\pi}) - \cos(\Delta t/2)S(\sqrt{\Delta t/\pi})}{(\Delta t/2)^{3/2}}, \tag{161}$$

where C and S are Fresnel integral functions. That gives $\langle|t|\rangle = 7/(3\Delta)$, so $\langle|t|\rangle\Delta \approx 2.333$.

To find the optimal window, we can take

$$\frac{1}{\sqrt{2\pi}}\int dt e^{itx}\chi_t = (1 - x^2)\sum_\ell a_\ell x^{2\ell}, \tag{162}$$

for x the difference in energy divided by $E_{\max}$. We use only even orders, so it is symmetric, and the factor of $(1 - x^2)$

ensures that it goes to zero at ±1. Then

$$\chi_t = \frac{1}{\sqrt{2\pi}} \sum_\ell a_\ell \int_{-1}^{1} dx \cos(xt)(1-x^2)x^{2\ell}. \qquad (163)$$

Then the expectation of the absolute value of the time is

$$\int dt|t||\chi_t|^2 = \frac{1}{2\pi} \sum_{k,\ell} a_k a_\ell A_{k\ell}, \qquad (164)$$

where

$$A_{k\ell} = \int dt|t| \left[\int_{-1}^{1} dx \cos(xt)(1-x^2)x^{2k} \right] \times \left[\int_{-1}^{1} dz \cos(zt)(1-z^2)z^{2\ell} \right]. \qquad (165)$$

We also need, for normalization,

$$1 = \sum_{k,\ell} a_k a_\ell \int_{-1}^{1} dx(1-x^2)^2 x^{2(k+\ell)} = \sum_{k,\ell} a_k a_\ell B_{k\ell}, \qquad (166)$$

where

$$B_{k\ell} = \frac{16}{[2(k+\ell)+1][2(k+\ell)+3][2(k+\ell)+5]}. \qquad (167)$$

Then defining $\vec{b} = B^{1/2}\vec{a}$, the normalization corresponds to $\|\vec{b}\| = 1$. Then the minimum $\langle |t| \rangle$ corresponds to minimizing $\vec{a}^T A \vec{a}/\pi$, which is equivalent to minimizing $\vec{a}^T B^{-1/2} A B^{-1/2} \vec{a}/\pi$, so we need to find the minimum eigenvalue of $B^{-1/2} A B^{-1/2}$. That gives $\langle |t| \rangle E_{\max} \approx 1.1580$ with terms up to a_{22} (a 46th-order polynomial).

This explanation is for the case where there is Hamiltonian evolution for a time t, which can take any real value. In the case of steps of a quantum walk with eigenvalues $e^{\pm i \arccos(H/\lambda)}$, the number of steps take an integer value. For the Hamiltonian evolution it could be implemented by steps of a quantum walk as well but it is more efficient to simply use the steps of that quantum walk directly without signal processing. To obtain the corresponding probability distribution for a discrete number of steps, we simply take the probability distribution for t at points separated by $1/\lambda$. That yields a probability distribution for the error that is the same as for the continuous distribution, except with a periodicity of λ. That periodicity has no effect on the error, because it is beyond the range of possible values for the energy. The reason for this correspondence is that taking the probability distribution at a series of discrete points is like multiplying by a comb function, equivalent to convolving the error distribution with a comb function.

D. Szegedy walk-based quantum simulated annealing

In the remainder of Sec. III we consider quantum simulated annealing, where the goal is to prepare a coherent equivalent of a Gibbs state and cool to a low temperature. More specifically, the coherent Gibbs state is

$$|\psi_\beta\rangle := \sum_{x \in \Sigma} \sqrt{\pi_\beta(x)}\, |x\rangle\,, \quad \pi_\beta(x) \propto \exp(-\beta E_x), \qquad (168)$$

where β is the inverse temperature. For annealing, we have transition probabilities of obtaining y from x denoted $\Pr(y|x)$, which must satisfy the detailed balance condition

$$\Pr(y|x)\pi_\beta(x) = \Pr(x|y)\pi_\beta(y). \qquad (169)$$

The detailed balance condition ensures that π_β is the equilibrium distribution with these transition probabilities. For the costings in this work we take for y differing from x by a single bit flip,

$$\Pr(y|x) := \min\left\{1, \exp\left[\beta\left(E_x - E_y\right)\right]\right\}/N, \qquad (170)$$

and $\Pr(x|x) = 1 - \sum_{y \neq x} \Pr(y|x)$. This choice is similar to that in Ref. [23]. Another choice, used in Ref. [14], is $\Pr(y|x) = \chi \exp\left[\beta\left(E_x - E_y\right)\right]$ for χ chosen to prevent sums of probabilities greater than 1. If one were to construct a Hamiltonian as

$$\langle x| H_\beta |y\rangle = \delta_{x,y} - \sqrt{\Pr(x|y)\Pr(y|x)}, \qquad (171)$$

then the detailed balance condition ensures that the ground state is $|\psi_\beta\rangle$ with eigenvalue zero. One can then apply an adiabatic evolution under this Hamiltonian to gradually reduce the temperature (increase β).

In the approach of Ref. [13], the method used is to instead construct a quantum walk where the quantum Gibbs state is an eigenstate. One could change the value of β between each step of the quantum walk similarly to the adiabatic algorithm for the Hamiltonian. Alternatively, for each value of β one can apply a measurement of the walk operator to project the state to $|\psi_\beta\rangle$ via the quantum Zeno effect. Reference [13] also proposes using a random number of steps of the walk operator to achieve the same effect as the measurement. The advantage of using the quantum walk is that the complexity scales as $\mathcal{O}(1/\sqrt{\delta})$, where δ is the spectral gap of H_β, rather than $\mathcal{O}(1/\delta)$, which is the best rigorous bound for the scaling of (classical) simulated annealing.

The quantum walk used in Ref. [13] is based on a Szegedy walk, which involves a controlled state preparation, a SWAP between the system and the ancilla, and inversion of the controlled state preparation. Then a reflection on the ancilla is required. The sequence of operations is as shown in Fig. 8. The dimension of the ancilla needed is the same as the dimension as the system. The reflection

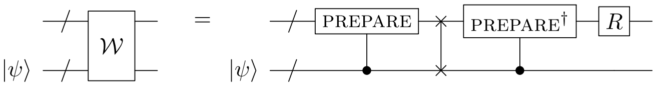

FIG. 8. The qubitized quantum walk operator $\mathcal{W}$ using the Szegedy approach.

and SWAP have low cost, so the Toffoli cost is dominated by the cost of the controlled state preparation.

The Szegedy approach builds a quantum walk in a similar way as the LCU approach in Fig. 2, where there is a block encoded operation followed by a reflection [31]. That is, preparation of the ancilla in the state $|0\rangle$, followed by unitary operations U and projection onto $|0\rangle$ on the ancilla yields the block encoded operator $A = \langle 0| U |0\rangle$. Instead of performing a measurement on the ancilla, the reflection about $|0\rangle$ results in a joint operation that has eigenvalues related to the eigenvalues of A as $e^{\pm i \arccos a}$, where a is an eigenvalue of A.

Here the controlled state preparation is of the form

$$\text{CPREP}\,|x\rangle\,|0\rangle = \sum_y \sqrt{\Pr(y|x)}\,|x\rangle\,|y\rangle \equiv |\alpha_x\rangle\,, \qquad (172)$$

where the sum is taken over all y that differ from x by at most one bit. As a result, the block-encoded operation is

$$\langle 0|\,\text{CPREP}^\dagger\text{SWAP CPREP}\,|0\rangle = \sum_{x,y} \sqrt{\Pr(x|y)\Pr(y|x)}\,|y\rangle\langle x|\,. \qquad (173)$$

Thus the block-encoded operation has a matrix representation of the form $\sqrt{\Pr(x|y)\Pr(y|x)}$, which is equivalent to $\mathbb{1} - H_\beta$. Therefore, the quantum Gibbs state $|\psi_\beta\rangle$ is an eigenstate of this operation with eigenvalue 1. Combining this operation with the reflection gives a step of a quantum walk with eigenvalues corresponding to the arccosine of the block-encoded operator [64,65]. It is this arccosine that causes a square-root improvement in the scaling with the spectral gap. This is because if the block-encoded operation has gap δE from the eigenvalue of 1 for the target state, taking the arccosine yields a gap of approximately $\sqrt{2\delta E}$ for the quantum walk. This gap governs the complexity of the algorithm based on the quantum walk.

In implementing the step of the walk, the state preparation requires calculation of each of the $\Pr(y|x)$ for a given x. In turn these require computing the energy difference under a bit flip, and the exponential. The probability $\Pr(x|x)$ is computed from the formula $\Pr(x|x) \equiv 1 - \sum_{y\neq x}\Pr(y|x)$ required for normalization of the probabilities. To prepare the state one can first prepare a state of the form

$$|\psi_x\rangle = \sum_k \sqrt{\Pr(x_k|x)}\,|x\rangle\,|k\rangle\,, \qquad (174)$$

where x_k indicates that bit k of x has been flipped with $k=0$ indicating no bit flip, and $|k\rangle$ is encoded in one-hot unary. The state $|\alpha_x\rangle$ can then be prepared by applying CNOTs between the respective bits of the two registers.

In order to prepare the state $|\psi_x\rangle$ in unary, an obvious method is to perform a sequence of controlled rotations depending on the transition probabilities. However, that tends to be expensive because our method of performing rotations involves multiplications, and high precision is required because the error in each rotation accumulates. A better method can be obtained by noting that the amplitudes for $k > 0$ are limited. We can then perform the state preparation by the following method.

1. Compute $N\Pr(x_k|x)$ for all N bit flips, and subtract those values from N to obtain $N\Pr(x|x)$. Note that $N\Pr(x_k|x) \leq 1$, and we compute this value to b_{sm} bits. The value of $N\Pr(x|x)$ needs $\lceil \log N\rceil + b_{\text{sm}}$ bits, but only the leading b_{sm} bits can be regarded as reliable. The complexity of the subtractions is $N(\lceil \log N\rceil + b_{\text{sm}})$.
2. We have N qubits in the target system we need to prepare the state and five ancillas,
$$|0\rangle_{\mathtt{A}}\,|0\rangle_{\mathtt{K}}\,|0\rangle_{\mathtt{Z}}\,|0\rangle_{\mathtt{ZZ}}\,|0\rangle_{\mathtt{B}}\,|0\rangle_{\mathtt{C}}\,, \qquad (175)$$
where $\mathtt{K}$ is the target system, $\mathtt{A}$, $\mathtt{B}$, and $\mathtt{C}$ are single-qubit ancillas, and $\mathtt{Z}$ and $\mathtt{ZZ}$ are s-qubit ancillas. Apply Hadamards to the ancillas to give equal superpositions on all except $\mathtt{ZZ}$ and $\mathtt{B}$.
$$|+\rangle_{\mathtt{A}}\,|0\rangle_{\mathtt{K}}\,\frac{1}{2^{s/2}}\sum_{z=0}^{2^s-1}|z\rangle_{\mathtt{Z}}\,|0\rangle_{\mathtt{ZZ}}\,|0\rangle_{\mathtt{B}}\,|+\rangle_{\mathtt{C}}\,. \qquad (176)$$
3. Controlled on ancilla A, prepare an equal superposition state on $\lceil \log N\rceil$ qubits of $\mathtt{K}$. If N is a power of 2, then it can be performed with $\log N$ controlled Hadamards, each of which can be performed with two T gates. It is also possible to prepare an equal superposition for N not a power of 2 with complexity $\mathcal{O}(\log N)$. For more details see Sec. III 2.
4. We can map the binary to unary in place, with cost no more than $N - \log N$ (see Appendix C), to give
$$\frac{1}{2^{s/2}\sqrt{2}}\left(|0\rangle_{\mathtt{A}}\,|0\rangle_{\mathtt{K}} + \frac{1}{\sqrt{N}}\,|1\rangle_{\mathtt{A}}\sum_{k=1}^{N}|k\rangle_{\mathtt{K}}\right) \times \sum_{z=0}^{2^s-1}|z\rangle_{\mathtt{Z}}\,|0\rangle_{\mathtt{ZZ}}\,|0\rangle_{\mathtt{B}}\,|+\rangle_{\mathtt{C}}\,, \qquad (177)$$
where $|k\rangle_{\mathtt{K}}$ is a value in one-hot unary.

5. Compute the approximate square of z, denoted $\tilde{z}^2$, placing the result in register ZZ, to give

$$\frac{1}{2^{s/2}\sqrt{2}}\left(|0\rangle_{\mathtt{A}}|0\rangle_{\mathtt{K}}+\frac{1}{\sqrt{N}}|1\rangle_{\mathtt{A}}\sum_{k=1}^{N}|k\rangle_{\mathtt{K}}\right)\times\sum_{z=0}^{2^s-1}|z\rangle_{\mathtt{Z}}|\tilde{z}^2\rangle_{\mathtt{ZZ}}|0\rangle_{\mathtt{B}}|+\rangle_{\mathtt{C}}. \tag{178}$$

The complexity is no greater than $s^2/2$, as discussed in Appendix 6. To obtain $b_{\rm sm}$ bits of precision in the square, we need to take $s=b_{\rm sm}+\mathcal{O}(\log b_{\rm sm})$, giving complexity ${b_{\rm sm}}^2/2+\mathcal{O}(b_{\rm sm}\log b_{\rm sm})$.

6. For each $k=1,\ldots,N$, perform an inequality test between $N\Pr(x_k|x)$ and z^2 in the ZZ register, controlled by qubit k in K, placing the result in B. This has cost $Nb_{\rm sm}$ Toffolis.
7. Controlled on ancilla A being zero, perform an inequality test between $N\Pr(x|x)$ and Nz^2, with the output in B. The inequality test has complexity $b_{\rm sm}$. In the case where N is not a power of 2, multiplying by N has complexity approximately ${b_{\rm sm}}^2+\mathcal{O}(b_{\rm sm}\log b_{\rm sm})$ to obtain $b_{\rm sm}$ bits, and we incur this cost twice, once for computation and once for uncomputation. If N is a power of 2 the multiplication by N has no cost. We obtain the state

$$\frac{1}{2^{s/2}\sqrt{2}}\left[|0\rangle_{\mathtt{A}}|0\rangle_{\mathtt{K}}\sum_{z=0}^{2^s\sqrt{\widetilde{\Pr}(x|x)}-1}|z\rangle_{\mathtt{Z}}|\tilde{z}^2\rangle_{\mathtt{ZZ}}|0\rangle_{\mathtt{B}}+|0\rangle_{\mathtt{A}}|0\rangle_{\mathtt{K}}\sum_{z=2^s\sqrt{\widetilde{\Pr}(x|x)}}^{2^s-1}|z\rangle_{\mathtt{Z}}|z^2\rangle_{\mathtt{ZZ}}|1\rangle_{\mathtt{B}}\right.$$
$$\left.+\frac{1}{\sqrt{N}}|1\rangle_{\mathtt{A}}\sum_{k=1}^{N}|k\rangle_{\mathtt{K}}\sum_{z=0}^{2^s\sqrt{N\widetilde{\Pr}(x_k|x)}-1}|z\rangle_{\mathtt{Z}}|\tilde{z}^2\rangle_{\mathtt{ZZ}}|0\rangle_{\mathtt{B}}+\frac{1}{\sqrt{N}}|1\rangle_{\mathtt{A}}\sum_{k=1}^{N}|k\rangle_{\mathtt{K}}\sum_{z=2^s\sqrt{N\widetilde{\Pr}(x_k|x)}}^{2^s-1}|z\rangle_{\mathtt{Z}}|\tilde{z}^2\rangle_{\mathtt{ZZ}}|1\rangle_{\mathtt{B}}\right]|+\rangle_{\mathtt{C}}, \tag{179}$$

where $\widetilde{\Pr}$ indicates an approximation of the probability, with the imprecision primarily due to imprecise squaring of z.

8. Uncompute z^2 in register ZZ with complexity no more than $s^2/2$.
9. Use a sequence of CNOTs with the N qubits of K as controls and ancilla A as target. This resets A to zero.
10. Perform Hadamards on the qubits of K, giving a state of the form

$$\frac{1}{2}|0\rangle_{\mathtt{A}}\left[\sqrt{\widetilde{\Pr}(x|x)}|0\rangle_{\mathtt{K}}+\frac{1}{\sqrt{N}}\sum_{k=1}^{N}\sqrt{\widetilde{\Pr}(x_k|x)}|k\rangle_{\mathtt{K}}\right]\times|0\rangle_{\mathtt{Z}}|0\rangle_{\mathtt{ZZ}}|0\rangle_{\mathtt{B}}|0\rangle_{\mathtt{C}}+|\psi^{\perp}\rangle, \tag{180}$$

where $|\psi^{\perp}\rangle$ is the component of the state perpendicular to zero states on Z, B, and C.

11. Now conditioned on $|0\rangle_{\mathtt{Z}}|0\rangle_{\mathtt{B}}|0\rangle_{\mathtt{C}}$, we have the correct state with amplitude approximately 1/2. We simply need to perform one round of amplitude amplification. We reflect about $|0\rangle_{\mathtt{Z}}|0\rangle_{\mathtt{B}}|0\rangle_{\mathtt{C}}$, invert steps 10 to 2, reflect about zero, then perform steps 2 to 10 again. In the limit of large s we then have the correct state. As well as incurring three times the cost of steps 2 to 10, we have a cost of $N+\mathcal{O}(b_{\rm sm})$ for the reflection.

The overall Toffoli complexity of this procedure, excluding the computation of $\Pr(x_k|x)$, is

$$N(\lceil\log N\rceil+b_{\rm sm})+N+3\times\left[N+{b_{\rm sm}}^2+2{b_{\rm sm}}^2+(N+1)b_{\rm sm}\right]+\mathcal{O}(\log N+b_{\rm sm}\log b_{\rm sm}). \tag{181}$$

Here the first term is for the subtractions in step 1, the second term N is for the reflection, then the terms inside the square brackets are from steps 2 to 10. In the square brackets N is for the binary to unary conversion, ${b_{\rm sm}}^2$ is for computation and inverse computation of z^2, $2{b_{\rm sm}}^2$ is for multiplication by N (computation and uncomputation), which is only needed for N not a power of 2, and $(N+1)b_{\rm sm}$ is for the $N+1$ inequality tests. The cost $\log N$ in the order term is for the controlled preparation of an equal superposition state, and $b_{\rm sm}\log b_{\rm sm}$ is the order term for the squaring and multiplication.

Note that the preparation is performed perfectly, because the initial amplitude is not exactly 1/2. We use a flag qubit to indicate success, which controls the SWAP. To see the effect of this procedure, suppose the system is in basis state

x. Then the state that is prepared is

$$\text{CPREP}\,|0\rangle\,|x\rangle\,|0\rangle = \mu_x\,|1\rangle\,|x\rangle \sum_y \sqrt{\Pr(y|x)}\,|y\rangle + \nu_x\,|0\rangle\,|x\rangle\,|\phi_x\rangle\,, \tag{182}$$

where the first qubit flags success, μ_y is an amplitude for success, ν_x is an amplitude for failure, and ϕ_x is some state that is prepared in the case of failure and can depend on x. Here we ignore the imperfect approximation of $\Pr(y|x)$, and are focusing just on the imperfect success probability. Then the SWAP is only performed in the case of success, which gives

$$\text{SWAP CPREP}\,|0\rangle\,|x\rangle\,|0\rangle = \mu_x\,|1\rangle \sum_y \sqrt{\Pr(y|x)}\,|y\rangle\,|x\rangle + \nu_x\,|0\rangle\,|x\rangle\,|\phi_x\rangle\,. \tag{183}$$

Then we can write

$$\langle 0|\,\langle y|\,\langle 0|\,\text{CPREP}^\dagger = \mu_y\,|1\rangle \sum_x \sqrt{\Pr(x|y)}\,\langle y|\,\langle x| + \nu_y\,\langle 0|\,\langle y|\,\langle \phi_y|\,, \tag{184}$$

so

$$\begin{aligned}\langle 0|\,\langle y|\,\langle 0|\,\text{CPREP}^\dagger\text{SWAP CPREP}\,|0\rangle\,|x\rangle\,|0\rangle &= \mu_x\mu_y\sqrt{\Pr(y|x)\Pr(x|y)} + \delta_{x,y}\nu_x^2 \\ &= \sqrt{\Pr'(y|x)\Pr'(x|y)},\end{aligned} \tag{185}$$

where we define

$$\Pr'(x|y) = \begin{cases}\mu_y^2\Pr(x|y), & x \neq y \\ 1-\mu_y^2\sum_{z\neq y}\Pr(z|y), & x = y.\end{cases} \tag{186}$$

That is, the effect of the imperfect preparation is that the qubitized step corresponds to a slightly lower probability of transitions, which should have only a minor effect on the optimization.

The cost of the quantum walk in this approach is primarily in computing all transition probabilities $N\Pr(x_k|x)$. If we were only concerned with the inequality tests for $k > 0$, then we could incur that cost only once with a simple modification of the above scheme. The problem is that we also need $N\Pr(x|x)$, which requires computing all $N\Pr(x_k|x)$. The steps of computing each $N\Pr(x_k|x)$ are as follows.

1. Query the energy-difference oracle to find the energy difference δE of a proposed transition to b_{dif} bits.
2. Calculate $\exp(-\beta\delta E)$ to b_{sm} bits using the QROM and interpolation method from Sec. II E.

The costs for the energy-difference oracles are discussed in Sec. II A, and are as in Table V. In this table, the costs for the energy-difference oracles for the L-term spin model and LABS problem are obtained by evaluating the energy twice. Computing N values of the energy difference suggests we multiply this cost by N, but we can save computation cost by just calculating the energy for x once, and computing the energy for each of the x_k. That means the cost for these problem Hamiltonians can be given as the cost for a single energy evaluation multiplied by $N+1$. For QUBO and the SK model it is considerably more efficient to compute the energy difference than the energy, so in these cases we simply compute the energy difference N times. The number of output registers is increased by a factor of N in all cases. For the cases where we compute the starting energy and the N energies under bit flips, we can compute the starting energy first, copy it into the N outputs, and subtract the energy under the bit flip from each of the output registers. In summary, the complexity can be given as the minimum of $N+1$ times the cost of the energy oracle, and N times the cost of the energy-difference oracle.

To perform the state preparation, we need to compute the energy differences, use those to compute the transition probabilities, prepare the state, then uncompute the transition probabilities and energy differences. In each step of the Szegedy walk as shown in Fig. 8, we need to do the controlled preparation and inverse preparation, which means that the energy differences need to be computed four times for each step. That gives a cost of

$$\min[4(N+1)\mathcal{C}^{\text{direct}}, 4N\mathcal{C}^{\text{diff}}] + 4N\mathcal{C}^{\text{fun}}. \tag{187}$$

However, we can save a factor of 2 by taking the controlled preparation and moving it to the end of the step, as shown in Fig. 9. The reason why we can save a factor of 2 is that then, in between the controlled inverse preparation and preparation, there is a reflection on the target, but the control is not changed. That means we can keep the values of the energy differences and transition probabilities computed in the controlled inverse preparation without uncomputing them, then only uncompute them after the controlled preparation.

This approach does not change the effect of a sequence of steps if β is kept constant. However, if β is changed between steps, then the procedure as shown in Fig. 8 is different to that taking the controlled preparation and moving it to the end of the state. That is, the value of β is changed

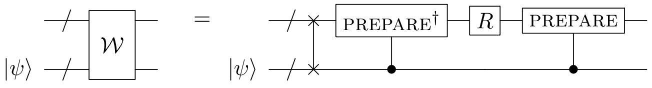

FIG. 9. The quantum walk operator using the Szegedy approach, where we have moved the controlled preparation to the end.

at the swap operation, rather than the reflection. Because there is only a factor of 2 rather than 4, the resulting cost is

$$\min\left[2(N+1)\mathcal{C}^{\text{direct}}, 2N\mathcal{C}^{\text{diff}}\right] + 2N\mathcal{C}^{\text{fun}}. \tag{188}$$

Now adding twice the complexity of the state preparation from Eq. (181) gives complexity

$$\min\left[2(N+1)\mathcal{C}^{\text{direct}}, 2N\mathcal{C}^{\text{diff}}\right] + 2N\mathcal{C}^{\text{fun}} + 2N\log N \\ + 8Nb_{\text{sm}} + 18{b_{\text{sm}}}^2 + \mathcal{O}(N). \tag{189}$$

Here we omit $b_{\text{sm}}\log b_{\text{sm}}$ in the order term because it is smaller than N for the parameter ranges we are interested in. The term $9{b_{\text{sm}}}^2$ includes $3{b_{\text{sm}}}^2$ from squaring and $6{b_{\text{sm}}}^2$ from multiplication. In the case where N is a power of 2 the cost of $6{b_{\text{sm}}}^2$ can be omitted.

To evaluate the numbers of ancillas needed, we need to distinguish between the persistent ancillas and temporary ancillas in Table V. This is because the persistent ancillas need to be multiplied by N, whereas the temporary ancillas are reused, so we only need to take the maximum. Considering the persistent ancillas first, the ancilla costs are as follows.

1. The N qubits for the Szegedy walk for the copy of the system.
2. N times the ancilla cost for the energy evaluation.
3. N times the ancilla cost for the function evaluation.
4. The ancillas Z, A, B, C in the state preparation use $b_{\text{sm}} + \mathcal{O}(\log b_{\text{sm}})$ qubits.

For the temporary ancillas, we have contributions from the energy-difference evaluation, the function evaluation, and the state preparation. Since these operations are not done concurrently, we can take the maximum of the costs. The most significant is that for the state preparation. In the state preparation we have the following costs.

1. Ancilla ZZ has $b_{\text{sm}} + \mathcal{O}(\log b_{\text{sm}})$ qubits, and it is temporary because it is uncomputed.
2. If N is not a power of 2 then we need another $b_{\text{sm}} + \mathcal{O}(\log b_{\text{sm}})$ qubit for an ancilla with Nz^2.
3. We use $b_{\text{sm}} + \mathcal{O}(\log b_{\text{sm}})$ qubits for squaring, or $2b_{\text{sm}} + \mathcal{O}(\log b_{\text{sm}})$ qubits if we are performing the multiplication by N.

As a result, the temporary ancilla cost is $2b_{\text{sm}} + \mathcal{O}(\log b_{\text{sm}})$ qubits if N is a power of 2, or $4b_{\text{sm}} + \mathcal{O}(\log b_{\text{sm}})$ otherwise. Considering the worst case that N is not a power of 2, this temporary ancilla cost is larger than that for the difference function evaluation, giving a total ancilla cost

$$N\mathcal{A}^{\text{diff}} + N\mathcal{A}^{\text{fun}} + 5b_{\text{sm}} + \mathcal{O}(\log b_{\text{sm}}). \tag{190}$$

E. LHPST-qubitized walk-based quantum simulated annealing

The same quantum walk approach to quantum simulated annealing can be achieved using an improved form of quantum walk given by Lemieux, Heim, Poulin, Svore, and Troyer [23] that requires only computation of a *single* transition probability for each step. Here we provide an improved implementation of that quantum walk that can be efficiently achieved for more general types of cost Hamiltonians than considered in Ref. [23]. The operations used to achieve the step of the walk are

$$\tilde{U}_W = RV^\dagger B^\dagger FBV, \tag{191}$$

where

$$V:\ |0\rangle_M \rightarrow \frac{1}{\sqrt{N}}\sum_j |j\rangle_M, \tag{192}$$

$$B:\ |x\rangle_S|j\rangle_M|0\rangle_C \rightarrow |x\rangle_S|j\rangle_M \\ \times\left(\sqrt{1-p_{x,x_j}}\,|0\rangle_C + \sqrt{p_{x,x_j}}\,|1\rangle_C\right), \tag{193}$$

$$F:\ \begin{array}{l}|x\rangle_S|j\rangle_M|0\rangle_C \rightarrow |x\rangle_S|j\rangle_M|0\rangle_C,\\ |x\rangle_S|j\rangle_M|1\rangle_C \rightarrow |x_j\rangle_S|j\rangle_M|1\rangle_C,\end{array} \tag{194}$$

$$R:\ \begin{array}{l}|0\rangle_M|0\rangle_C \rightarrow -|0\rangle_M|0\rangle_C,\\ |j\rangle_M|c\rangle_C \rightarrow |j\rangle_M|c\rangle_C \text{ for } (j,c)\neq(0,0).\end{array} \tag{195}$$

Here $p_{x,y} = N\Pr(y|x)$ in the notation used above, and we specialize to an equal superposition over j and only single bit flips.

This walk is equivalent to the Szegedy approach of Ref. [13] because it yields the same block-encoded operation. That is, $\langle 0|V^\dagger B^\dagger FBV|0\rangle$ has matrix representation $\sqrt{\Pr(x|y)\Pr(y|x)}$. To show this fact, the sequence of operations gives

$$V|0\rangle_M|0\rangle_C = \frac{1}{\sqrt{N}}\sum_{j=1}^N |j\rangle_M|0\rangle_C, \tag{196}$$

$$BV|0\rangle_M|0\rangle_C = \frac{1}{\sqrt{N}}\sum_x\sum_{j=1}^N |x\rangle\langle x|\otimes|j\rangle\left(\sqrt{1-p_{x,x_j}}\,|0\rangle_C + \sqrt{p_{x,x_j}}\,|1\rangle_C\right), \tag{197}$$

$$FBV|0\rangle_M|0\rangle_C = \frac{1}{\sqrt{N}}\sum_x\sum_{j=1}^N |x\rangle\langle x|\otimes|j\rangle\sqrt{1-p_{x,x_j}}\,|0\rangle_C$$

$$
\begin{aligned}
&+ \frac{1}{\sqrt{N}} \sum_{x} \sum_{j=1}^{N} |x_j\rangle\langle x| \otimes |j\rangle \sqrt{p_{x,x_j}}\, |1\rangle_C \\
&= \frac{1}{\sqrt{N}} \sum_{x} \sum_{j=1}^{N} |x\rangle\langle x| \otimes |j\rangle \sqrt{1 - p_{x,x_j}}\, |0\rangle_C \\
&+ \frac{1}{\sqrt{N}} \sum_{x} \sum_{j=1}^{N} |x\rangle\langle x_j| \otimes |j\rangle \sqrt{p_{x_j,x}}\, |1\rangle_C , \qquad (198)
\end{aligned}
$$

$$
\begin{aligned}
{}_M\langle 0|_C \langle 0| V^{\dagger} B^{\dagger} F B V |0\rangle_M |0\rangle_C &= \frac{1}{N} \sum_{x} \sum_{j=1}^{N} |x\rangle\langle x| \,(1 - p_{x,x_j}) + \frac{1}{N} \sum_{x} \sum_{j=1}^{N} |x\rangle\langle x_j| \sqrt{p_{x,x_j} p_{x_j,x}} \\
&= \sum_{x} |x\rangle\langle x| \left(1 - \frac{1}{N} \sum_{j=1}^{N} p_{x,x_j}\right) + \frac{1}{N} \sum_{x} \sum_{j=1}^{N} |x_j\rangle\langle x| \sqrt{p_{x,x_j} p_{x_j,x}} \\
&= \sum_{x,y} |y\rangle\langle x| \sqrt{\Pr(y|x)\, \Pr(x|y)}. \qquad (199)
\end{aligned}
$$

Just as with the Szegedy approach, most operations are trivial to perform, and the key difficulty is in the operation B, which depends on the transition probability. However, B only depends on *one* transition probability, whereas the Szegedy approach requires computing all the transition probabilities for a state preparation. Lemieux *et al.* [23] propose a method for the B operation that is not useful for the cost Hamiltonians considered here, but is useful for Hamiltonians with low connectivity. Instead of computing the energy difference then the exponential, they consider an approach where the required angle of rotation is found from a database.

That is, one considers the qubits that the transition probability for the move (here a bit flip) depends on, and classically precomputes the rotation angle for each basis state on those qubits. For each value of j, one sequentially performs a multiply controlled Toffoli for each computational basis state for these qubits, and performs the required rotation on the ancilla qubit C. The complexity that is given by Ref. [23] is $\mathcal{O}(2^{|\mathcal{N}_j|}|\mathcal{N}_j|\log(1/\epsilon))$, where $|\mathcal{N}_j|$ is the number of qubits that the transition probability for move j depends on. That complexity is a slight overestimate, because each multiply controlled Toffoli has a cost of $|\mathcal{N}_j|$, then the cost of the rotation synthesis is $\mathcal{O}(\log(1/\epsilon))$. It should also be noted that this is the cost for each value of j, and there are N values of j, giving an overall cost $\mathcal{O}(N2^{|\mathcal{N}_j|}[|\mathcal{N}_j| + \log(1/\epsilon)])$.

To improve the complexity, one can divide this procedure into two parts, where first a QROM is used to output the desired rotation in an ancilla, and then those qubits are used to control a rotation. Using the QROM procedure of Ref. [26] to output the required rotation, the cost in terms of Toffoli gates is $\mathcal{O}(N2^{|\mathcal{N}_j|})$. Then one can apply rotations using the phase-gradient state, which was discussed above in Sec. III C. Addition of the register containing the rotation to an ancilla with state $|\phi\rangle$ from Eq. (34) results in a phase rotation. To rotate the qubit, simply make the addition controlled by this qubit, and use Clifford gates before and after so that the rotation is in the y direction. The cost of this rotation is $\mathcal{O}(\log(1/\epsilon))$ Toffolis; for more details see Appendix A. With that improvement the complexity is reduced to $\mathcal{O}(N2^{|\mathcal{N}_j|} + \log(1/\epsilon))$.

Even with that improvement, any procedure of that type is exponential in the number of qubits that the energy difference depends on, $|\mathcal{N}_j|$. That is acceptable for the types of Hamiltonians considered in Ref. [23], but here we consider Hamiltonians typical of real-world problems where the energy difference depends on most of the system qubits, because the Hamiltonians have high connectivity. We thus propose alternative procedures to achieve the rotation B.

1. Rotation B

We propose a completely different method to perform the rotation B than that of LHPST [23]. We can first compute the energy difference $E_x - E_{x_j}$, then the rotation $\arcsin\sqrt{p_{x,x_j}}$ with the result put in an ancilla register. The rotation of the qubit ancilla C is controlled on the value of this ancilla as explained above, then the value of $\arcsin\sqrt{p_{x,x_j}}$ is uncomputed. There are many possible approaches to the computation of $\arcsin\sqrt{p_{x,x_j}}$, for example that of Ref. [66]. For the purposes of quantum optimization, we expect that we do not need to compute this function to high precision as long as the function we compute is still monotonic in the actual energy, so there is the opportunity to use methods that are very efficient for low precision but is not suitable for high precision. We propose a method based on using a piecewise linear approximation, with the coefficients output by a QROM, as described in Sec. II E.

One could then apply the controlled rotation with cost $b_{\rm sm}$ Toffolis using the phase-gradient state in Eq. (34), as described in detail in Appendix A. Then after uncomputing the rotation angle we implement B. That approach then means that a single step of the walk has four times the cost of computing $\arcsin\sqrt{p_{x,x_j}}$, because it needs to be computed and uncomputed for B, and the operation B is applied twice in each step.

It is possible to halve that cost by only computing and uncomputing once in a step, and retaining the value of $\arcsin\sqrt{p_{x,x_j}}$ during the F operation. Because F is a controlled flip of bit j of x, it reverses the role of x and x_j, and the sign of $E_x - E_{x_j}$ is flipped. In more detail, the procedure is as follows.

1. Compute the energy difference between x and x_j, $E_x - E_{x_j}$.
2. Compute $\arcsin\sqrt{p_{x,x_j}}$ based on $|E_x - E_{x_j}|$.
3. If $E_{x_j} < E_x$ then perform an X operation on the qubit C. That can be achieved with a CNOT (Clifford) controlled by the sign bit of $E_x - E_{x_j}$.
4. The remaining rotations for the case of $E_{x_j} > E_x$ need to be controlled on -1 for the sign bit.
5. When we apply F, as well as applying the Toffolis to change x to x_j, we need to flip the sign bit on $E_x - E_{x_j}$ controlled on qubit C. This is another CNOT, with no non-Clifford cost.
6. Then at the end we uncompute $\arcsin\sqrt{p_{x,x_j}}$ and $E_x - E_{x_j}$.

This procedure assumes that $E_x - E_{x_j}$ is represented as a signed integer. The computation of $E_x - E_{x_j}$ uses two's complement, so there is an additional cost of $b_{\rm dif}$ to switch to a signed integer. Because there is only a factor of 2 instead of 4, the overall cost is then $2\mathcal{C}^{\rm diff} + 2\mathcal{C}^{\rm fun} + 2b_{\rm dif} + \mathcal{O}(1)$. Next we consider the other (simpler) operations used in the step of the quantum walk.

2. Equal superposition V

The operation V generates the equal superposition starting from a zero state

$$V : |0\rangle_M \to \frac{1}{\sqrt{N}} \sum_j |j\rangle_M . \qquad (200)$$

In the case where N is a power of 2, then we can create the equal superposition over binary by using Hadamards (and no Toffolis). More generally, if we wish to create an equal superposition where the number of items is *not* a power of 2, we can rotate an ancilla qubit such that the net amplitude is $1/2$ for $|1\rangle|1\rangle$ on the result of the inequality test and the ancilla qubit. We can then perform a single step of amplitude amplification to obtain the superposition state. Our procedure is explained below and gives a complexity of $4\log N + \mathcal{O}(1)$ Toffolis.

Our method for V is also very different to that of LHPST [23]. There they proposed encoding the M register in unary, whereas here we use binary, which greatly reduces the ancilla cost (which is sublinear in N). Moreover, LHPST did not consider using equal superpositions in cases where N is not a power of 2, and instead just allowed for a constant factor overhead in the complexity.

Our procedure to create an equal superposition over $N < 2^k$ items is as follows. With Hadamards we prepare

$$\frac{1}{\sqrt{2^k}} \sum_{j=0}^{2^k-1} |j\rangle . \qquad (201)$$

Then we have an inequality test between j and N to give

$$\frac{1}{\sqrt{2^k}} \sum_{j=0}^{2^k-1} |j\rangle|1\rangle + \frac{1}{\sqrt{2^k}} \sum_{j=N}^{2^k-1} |j\rangle|0\rangle . \qquad (202)$$

This is an inequality test on k bits, and since it is an inequality test with a *constant* we save a Toffoli gate. The cost is therefore $k-2$ Toffolis as per the explanation in Ref. [42]. We have an amplitude of $\sqrt{N/2^k}$ for success, and aim to multiply it by another amplitude of approximately $\frac{1}{2}\sqrt{2^k/N}$ so the amplitude is $1/2$ and we can use a single step of amplitude amplification. For an amplitude of $\frac{1}{2}\sqrt{2^k/N}$, we can rotate another register according to the procedure in Appendix A to give

$$\cos\theta\,|0\rangle + \sin\theta\,|1\rangle . \qquad (203)$$

We can then perform a single step of amplitude amplification for $|1\rangle$ on both this qubit and the result of the inequality test.

The steps needed in the amplitude amplification and their costs are as follows. If we use the procedure for the rotation with s bits, it takes $s-3$ Toffolis because the angle of rotation is given classically.

1. A first inequality test ($k-2$ Toffolis) and a rotation on a qubit (cost $s-3$).
2. A first reflection on the rotated qubit and the result of the inequality test. This just needs a controlled phase (a Clifford gate).

3. Inverting the rotation and inequality test has cost $k+s-5$.
4. Hadamards then reflection of the k qubits and the single qubit ancilla about zero ($k-1$ Toffolis).
5. Applying the inequality test ($k-2$ Toffolis).

The total cost is $4k+2s-13$ Toffolis.

Conditioned on success of the inequality test, the state is

$$\frac{1}{\sqrt{2^k}}\sum_{j=0}^{N-1}|j\rangle\left[\left(1-\frac{4N\sin^2\theta}{2^k}\right)|0\rangle+2\sin\theta(\sin\theta\,|0\rangle+\cos\theta\,|1\rangle)\right]|1\rangle\,. \tag{204}$$

The probability for success is then given by the normalization

$$\frac{N}{2^k}\left[\left(1-\frac{4N\sin^2\theta}{2^k}+2\sin^2\theta\right)^2+4\sin^2\theta\cos^2\theta\right]. \tag{205}$$

It is found that highly accurate results are obtained for $s=7$, as shown in Fig. 10. This procedure enables construction of equal superposition states flagged by an ancilla qubit for N not a power of 2. If we take $s=7$, then the cost is $4k+1$.

3. Controlled bit flip F

We also need to modify the operation F compared to that in LHPST to account for the M register being encoded in binary. This operation flips bit j on x for the control qubit C being in the state $|1\rangle$,

$$F:\quad \begin{aligned}&|x\rangle_S\,|j\rangle_M\,|0\rangle_C\to|x\rangle_S\,|j\rangle_M\,|0\rangle_C\,,\\ &|x\rangle_S\,|j\rangle_M\,|1\rangle_C\to|x_j\rangle_S\,|j\rangle_M\,|1\rangle_C\,.\end{aligned} \tag{206}$$

This operation can be achieved using the iteration procedure of Ref. [26] with Toffoli complexity N, which allows us to perform the operation with register M encoded in binary.

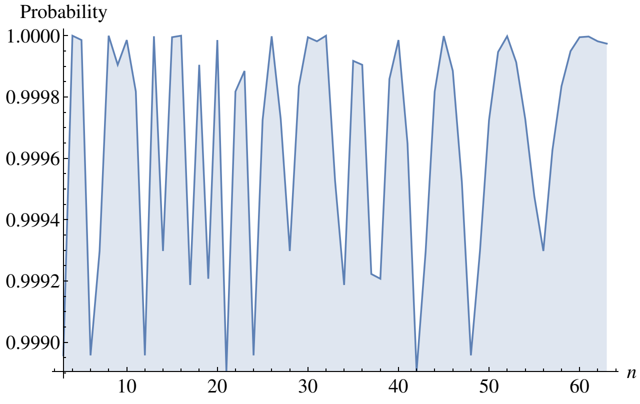

FIG. 10. The probability for success using a rotation of the form $2\pi/2^s$ with $s=7$.

A complication is that, in the case where N is not a power of 2, there is a nonzero cost of the state preparation in V failing. We should only perform the operation F in the case where we have success of the state preparation. We include another two Toffolis to create and erase a register that gives a control qubit that flags whether the C register is in the state $|1\rangle$ and there is success of the state preparation. Because the other operations are inverted, in the case that the state preparation does not work the net operation performed is the identity.

To be more specific, V prepares a state of the form

$$V\,|0\rangle=|1\rangle\,|\psi_1\rangle+|0\rangle\,|\psi_0\rangle\,, \tag{207}$$

with the first register flagging success. Since we only perform F for the flag qubit in the state $|1\rangle$, we obtain

$$B^\dagger FBV\,|0\rangle=B^\dagger FB\,|1\rangle\,|\psi_1\rangle+\mathbb{1}\,|0\rangle\,|\psi_0\rangle\,. \tag{208}$$

To determine the block-encoded operation

$$\langle 0|\,V^\dagger B^\dagger FBV\,|0\rangle\,, \tag{209}$$

we note that

$$\langle 0|\,V^\dagger=\langle 1|\,\langle\psi_1|+\langle 0|\,\langle\psi_0|\,, \tag{210}$$

so

$$\langle 0|\,V^\dagger B^\dagger FBV\,|0\rangle=\langle\psi_1|\,B^\dagger FB\,|\psi_1\rangle+\mathbb{1}\langle\psi_0|\psi_0\rangle, \tag{211}$$

where $\mathbb{1}$ indicates the identity on the target system. The first term is the desired operation we obtain if the equal superposition state is obtained exactly (with a multiplying factor corresponding to the probability of success), and the second term is proportional to the identity. This small offset by the identity just gives a trivial shift to the eigenvalues.

4. Reflection *R*R

This operation applies a phase flip for zero on the ancillas as

$$R: \; |0\rangle_M |0\rangle_C \to -|0\rangle_M |0\rangle_C, \\ |j\rangle_M |c\rangle_C \to |j\rangle_M |c\rangle_C \text{ for } (j,c) \neq (0,0). \tag{212}$$

As well as the ancillas M and C, this reflection also needs to be on any ancilla qubits used to encode the ancilla for the preparation of the equal superposition state, and the flag qubit. There are $\lceil \log N \rceil$ qubits used to encode j, one qubit for C, and one ancilla used for the rotation, for a total of $\lceil \log N \rceil + 2$ qubits. Therefore, the number of Toffolis needed for reflection about zero is $\lceil \log N \rceil$.

5. Total costs

The total Toffoli costs of implementing $\tilde{U}_W = RV^\dagger B^\dagger FBV$ are as follows.

1. The cost of V and $V^\dagger$ is $8 \log N + \mathcal{O}(1)$.
2. The cost of F is N Toffolis.
3. The cost of R is $\lceil \log N \rceil$.
4. The cost of two applications of B is $2\mathcal{C}^{\text{diff}} + 2\mathcal{C}^{\text{fun}} + 2b_{\text{dif}} + \mathcal{O}(1)$.

The total cost of a step is then

$$2\mathcal{C}^{\text{diff}} + 2\mathcal{C}^{\text{fun}} + N + 2b_{\text{dif}} + 9 \log N + \mathcal{O}(1). \tag{213}$$

Note that $8 \log N$ of this cost is for preparing equal superposition states, and can be omitted if N is a power of 2. The ancilla qubits needed are as follows.

1. The ancilla registers M and C need $\lceil \log N \rceil + 1$ qubits.
2. The resource state used to implement the controlled rotations needs b_{sm} qubits.
3. The ancilla requirements of the energy-difference and function-evaluation oracles.

For the temporary ancilla cost, we need to take the maximum of that for the energy difference and function evaluation, giving the total ancilla cost of

$$\mathcal{A}^{\text{diff}} + \mathcal{A}^{\text{fun}} + \max(\mathcal{B}^{\text{diff}}, \mathcal{B}^{\text{fun}}) + \log N + b_{\text{sm}} + \mathcal{O}(1). \tag{214}$$

F. Spectral-gap-amplification-based quantum simulated annealing

An alternative, and potentially simpler, approach to preparing a low-temperature thermal state is given by Ref. [14]. The idea behind this approach is to construct a Hamiltonian whose ground state is a purification of the Gibbs state. Similarly to the case with the quantum walk, one can start with an equal superposition state corresponding to infinite temperature, and simulate the Hamiltonian evolution starting from $\beta = 0$ and gradually increase β. This approach can correspond to using an adibatic approach on this Hamiltonian, or one can also apply a quantum Zeno approach by phase measurements on the Hamiltonian evolution, or apply Hamiltonian evolutions for randomly chosen times.

A simple choice of Hamiltonian is similar to the block-encoded operation for the quantum walks, so has a small spectral gap. In order to obtain a speedup, one needs to construct a new Hamiltonian with the square root of the spectral gap of the original Hamiltonian, thus yielding the same speedup as the quantum walks. That procedure, from Ref. [14], is called spectral-gap amplification. Simulating the time-dependent Hamiltonian, for example using a Dyson series, has significant complexity.

To avoid that complexity, here we suggest that one instead construct a step of a quantum walk using a linear combination of unitaries. Such a quantum walk could be used to simulate the Hamiltonian evolution, but as discussed in Refs. [64,65] one can instead just perform steps of the quantum walk, which has eigenvalues that are the exponential of the arccosine of those for the Hamiltonian. By applying the steps of the quantum walk we can obtain the advantage of the spectral-gap amplification, without the difficulty of needing to simulate a time-dependent Hamiltonian. Unlike the quantum walks in the previous subsections, the arccosine does not yield a further square-root amplification of the spectral gap, because the relevant eigenvalue for the amplified Hamiltonian is not at 1. However, it potentially gives other scaling advantages (for instance, in avoiding the need for quantum-signal processing when using certain oracles) compared to other proposals in the literature for realizing quantum simulated annealing via spectral-gap amplification.

1. Spectral-gap-amplification Hamiltonian

Here we summarize the method of spectral-gap amplification from Ref. [14], but specialise to the case where only single bit flips are allowed to make the method clearer. As discussed above, one can use a Hamiltonian simulation approach with Hamiltonian H_β given in Eq. (171) with ground state corresponding to the quantum Gibbs state $|\psi_\beta\rangle$. Because the complexity depends on the spectral gap, it is advantageous to increase the spectral gap as much as possible, which was done via a quantum walk in the previous subsections. The proposal in Ref. [14] is to construct a different Hamiltonian whose spectral gap has been amplified relative to H_β. To define this new Hamiltonian, they introduce states equivalent to

$$|\lambda_{x,y}\rangle := \sqrt{\frac{p_{x,y}}{p_{x,y}+p_{y,x}}}\,|y\rangle - \sqrt{\frac{p_{y,x}}{p_{x,y}+p_{y,x}}}\,|x\rangle, \tag{215}$$

where as before $p_{x,y} = N\,\mathrm{Pr}(y|x)$. This is the normalized form of the unnormalized kets $|\mu_\beta^{\sigma_i,\sigma_j}\rangle$ presented in Eq. (21) of Ref. [14]. One can then write

$$H_\beta = \frac{1}{2N}\sum_{x,y}\left(p_{x,y}+p_{y,x}\right)|\lambda_{x,y}\rangle\langle\lambda_{x,y}|\,. \tag{216}$$

In this work we consider only transitions with single bit flips, so the coefficient $\left(p_{x,y}+p_{y,x}\right)$ is nonzero only if x and y differ by exactly one bit. We include a factor of $1/2$ to account for the symmetry between x and y. We can use this condition to express H_β as a sum of 2-sparse matrices. To do so, recall that each x is an N-bit string. Then for each $k = 1,\ldots,n$ we define

$$H_{\beta,k} := \frac{1}{2N}\sum_{x}\left(p_{x,x_k}+p_{x_k,x}\right)|\lambda_{x,x_k}\rangle\langle\lambda_{x,x_k}|\,, \tag{217}$$

where $x_k = \mathrm{NOT}_k(x)$, the result of flipping the kth bit of x. Then $H_\beta = \sum_k H_{\beta,k}$. The operators $H_{\beta,k}$ here are equivalent to $O_{\beta,k}$ in Ref. [14], except we specialize to the case where only transitions with single bit flips are allowed.

One can then define a new Hamiltonian [Eq. (25) in Ref. [14]]

$$A_\beta := \sum_{k=1}^{N}\sqrt{H_{\beta,k}}\otimes(|k\rangle\langle 0| + |0\rangle\langle k|). \tag{218}$$

The projector structure of the Hamiltonian allows the square root to be easily implemented via

$$\sqrt{H_{\beta,k}} = \frac{1}{2\sqrt{N}}\sum_{x}\sqrt{p_{x,x_k}+p_{x_k,x}}\,|\lambda_{x,x_k}\rangle\langle\lambda_{x,x_k}|\,. \tag{219}$$

Here the $1/2$ is still included to account for the symmetry between i and $i^{(k)}$. Following Eq. (32) in Ref. [14], a coherent Gibbs distribution can be seen to be the ground state of the following Hamiltonian:

$$\tilde{H}_\beta := A_\beta + \sqrt{\Delta_\beta}(\mathbb{1} - |0\rangle\langle 0|), \tag{220}$$

where Δ_β is a lower bound for the spectral gap of H_β. This means that by preparing the minimum energy configuration of this Hamiltonian one, in effect, is capable of drawing a sample from the distribution that is seen by running a simulated annealing procedure for sufficient time.

2. Implementing the Hamiltonian

In order to implement the Hamiltonian, we use a linear combination of unitaries. We can rewrite the square root of the Hamiltonian as

$$\sqrt{H_{\beta,k}} = \frac{1}{2\sqrt{N}}\sum_{x}\left(p_{x,x_k}+p_{x_k,x}\right)^{-1/2} \left[p_{x,x_k}\,|x_k\rangle\langle x_k| + p_{x_k,x}\,|x\rangle\langle x| - \sqrt{p_{x,x_k}p_{x_k,x}}\,(|x\rangle\langle x_k| + |x_k\rangle\langle x|)\right]. \tag{221}$$

This is a 2-sparse Hamiltonian, then summing over k to obtain A_β gives a $2N$-sparse Hamiltonian. To express A_β as a linear combination of unitaries, we can express $\sqrt{H_{\beta,k}}$ as

$$\begin{aligned}\sqrt{H_{\beta,k}} &= \frac{1}{\sqrt{N}}\sum_{x} q_{xk}\,|x\rangle\langle x| - \frac{1}{2\sqrt{2N}}\sum_{x} r_{xk}\,(|x\rangle\langle x_k| + |x_k\rangle\langle x|)\\ &= \frac{1}{\sqrt{N}}\sum_{x}\int_0^1 dz(-1)^{2z>1+q_{xk}}\,|x\rangle\langle x| - \frac{1}{2\sqrt{2N}}\sum_{x}\int_0^1 dz(-1)^{2z>1+r_{xk}}\,(|x\rangle\langle x_k| + |x_k\rangle\langle x|)\,,\end{aligned} \tag{222}$$

where

$$q_{xk} = \frac{p_{x_k,x}}{\sqrt{p_{x,x_k}+p_{x_k,x}}}, \tag{223}$$

$$r_{xk} = \sqrt{\frac{2p_{x_k,x}p_{x,x_k}}{p_{x,x_k}+p_{x_k,x}}}, \tag{224}$$

and we are taking the inequality test to yield a numerical value of 0 for false and 1 for true. Note that with these definitions, q_{xk} and r_{xk} can take values in the range $[0,1]$. We use the procedure from Ref. [67] (Lemma 4.3) to obtain a linear combination of unitaries. The operator is then approximated as a sum

$$\begin{aligned}\sqrt{H_{\beta,k}} \approx{}& \frac{1}{2^s\sqrt{N}}\sum_{z=0}^{2^s-1}\sum_x(-1)^{z/2^{s-1}>1+q_{xk}}\,|x\rangle\langle x| \\ &- \frac{1}{2^{s+1}\sqrt{2N}}\sum_{z=0}^{2^s-1}\sum_x(-1)^{z/2^{s-1}>1+r_{xk}}\,(|x\rangle\langle x_k| + |x_k\rangle\langle x|)\,. \end{aligned} \tag{225}$$

The operator A_β is then approximated by

$$\begin{aligned}A_\beta \approx{}& \frac{1}{\sqrt{N}}\sum_{k=1}^{N}\left\{\left[\frac{1}{2^s}\sum_{z=0}^{2^s-1}\sum_x(-1)^{z/2^{s-1}>1+q_{xk}}\,|x\rangle\langle x|\right.\right. \\ &\left.\left.-\frac{1}{2^{s+1}\sqrt{2}}\sum_{z=0}^{2^s-1}\sum_x(-1)^{z/2^{s-1}>1+r_{xk}}\,(|x\rangle\langle x_k| + |x_k\rangle\langle x|)\right]\otimes(|k\rangle\langle 0| + |0\rangle\langle k|)\right\}. \end{aligned} \tag{226}$$

For the part $\sqrt{\Delta_\beta}(\mathbb{1} - |0\rangle\langle 0|)$, we can write it as

$$\begin{aligned}\sqrt{\Delta_\beta}(\mathbb{1} - |0\rangle\langle 0|) &= \frac{\sqrt{2N\Delta_\beta}}{(N-1)(\sqrt{2}-1)}\frac{1}{\sqrt{N}}\left(1-\frac{1}{\sqrt{2}}\right)(N-1)(\mathbb{1} - |0\rangle\langle 0|) \\ &= \frac{\delta_\beta}{\sqrt{N}}\sum_{k=1}^{N}\left(1-\frac{1}{\sqrt{2}}\right)\sum_{\ell>0,\ell\neq k}|\ell\rangle\langle\ell| \\ &= \frac{1}{\sqrt{N}}\frac{1}{2^s}\sum_{z=0}^{2^s-1}(-1)^{z/2^{s-1}>1+\delta_\beta}\sum_{k=1}^{N}\left(1-\frac{1}{\sqrt{2}}\right)\sum_{\ell>0,\ell\neq k}|\ell\rangle\langle\ell| \end{aligned} \tag{227}$$

where

$$\delta_\beta := \frac{\sqrt{2N\Delta_\beta}}{(N-1)(\sqrt{2}-1)}. \tag{228}$$

Therefore, the complete approximation of the Hamiltonian with spectral-gap amplification is

$$\begin{aligned}\tilde{H}_\beta \approx{}& \frac{1}{2^s\sqrt{N}}\sum_{k=1}^{N}\sum_{z=0}^{2^s-1}\left\{\left[\sum_x(-1)^{z/2^{s-1}>1+q_{xk}}\,|x\rangle\langle x|\otimes(|k\rangle\langle 0| + |0\rangle\langle k|) + (-1)^{z/2^{s-1}>1+\delta_\beta}\,\mathbb{1}\otimes\sum_{\ell>0,\ell\neq k}|\ell\rangle\langle\ell|\right]\right. \\ &\left.-\frac{1}{\sqrt{2}}\left[\frac{1}{2}\sum_x(-1)^{z/2^{s-1}>1+r_{xk}}\,(|x\rangle\langle x_k| + |x_k\rangle\langle x|)\otimes(|k\rangle\langle 0| + |0\rangle\langle k|) + (-1)^{z/2^{s-1}>1+\delta_\beta}\,\mathbb{1}\otimes\sum_{\ell>0,\ell\neq k}|\ell\rangle\langle\ell|\right]\right\}. \end{aligned} \tag{229}$$

Here we group the terms such that the operations in square brackets are unitaries. Summing the coefficients in the sums gives a λ value of

$$\lambda = \left(1+\frac{1}{\sqrt{2}}\right)\sqrt{N}. \tag{230}$$

To implement the operator by a linear combination of unitaries, we need two single qubit ancillas, a register with z and a register with k. The PREPARE operation is trivial, and just needs to prepare the state

$$\frac{1}{\sqrt{\lambda 2^s}}\sum_{k=1}^{N}|k\rangle\sum_{z=0}^{2^s-1}|z\rangle\left(|0\rangle_F+\frac{1}{2^{1/4}}|1\rangle_F\right). \tag{231}$$

The roles of these registers are as follows.

1. The register with k selects terms in the sum over k in Eq. (229).
2. The register with z selects terms in the sum over z in Eq. (229).
3. The F register selects between the terms in square brackets in the first and second lines of Eq. (229).

There are registers containing k for both this prepared control state and the target state. We call these the control and target K registers. In the PREPARE operation, creating the superposition over z can be trivially achieved with Hadamards. The superposition over N can be achieved similarly if N is a power of 2, but otherwise the procedure outlined in Sec. III 2 can be used with cost $4\log N+\mathcal{O}(1)$. The rotation on qubit F can be achieved with precision ϵ using $1.15b_{\rm rot}+\mathcal{O}(1)$ T operations, where $b_{\rm rot}=\log(1/\epsilon)$.

The SELECT procedure for the linear combinations of unitaries may be performed as follows.

1. Perform a test of whether the target system K register is in the space $\{|0\rangle, |k\rangle\}$, placing the result in an ancilla qubit, call this qubit E.
2. Controlled on E being $|1\rangle$ and F, compute q_{xk} or r_{xk}.
3. Controlled on E being $|0\rangle$, place the value δ_β into the output register also used for q_{xk} or r_{xk}.
4. Perform the inequality test between $z/2^{s-1}$ and $1+q_{xk}$, $1+r_{xk}$, or $1+\delta_\beta$.
5. Apply a Z gate to the output of the inequality test.
6. Controlled on the E register being $|1\rangle$ *and* the register F being $|1\rangle$, apply X to qubit k of the target system.
7. Apply a NOT between $|0\rangle$ and $|k\rangle$ for the target system. That gives $|k\rangle\langle 0|+|0\rangle\langle k|$.
8. Invert the inequality test from step 4.
9. Invert step 3.
10. Invert step 2 uncomputing q_{xk} or r_{xk}.
11. Invert step 1.
12. Apply a Z gate to F to introduce the -1 sign.

Here we call the register that carries $|k\rangle$ for the target system the K register. The cost of these steps may be quantified as follows, ignoring $O(1)$ costs.

Steps 1 and 11. We need an equality test between the K register for the ancilla and the K register for the system, with cost $\log N+\mathcal{O}(1)$. We also test if the system has 0 in its K register, with cost $\log N+\mathcal{O}(1)$, and perform an OR on the results of the two comparisons with cost 1. Since the comparisons needs to be computed and uncomputed, there is cost $4\log N+\mathcal{O}(1)$ for the two steps.

Steps 2 and 10. Computing q_{xk} and r_{xk} may be performed by first computing the energy difference, then using a QROM to output coefficients for linear interpolation. The cost estimation is as given in Sec. II E, and we pay the QROM lookup cost twice for q_{xk} and r_{xk}, but we pay the multiplication cost only once. Since that is the dominant cost, the cost may be regarded as that of a single function oracle. The computation and uncomputation in the two steps means we pay twice the cost of the energy difference and function oracles. Note that q_{xk} and r_{xk} are unchanged under the bit flip in step 6 (since there is no bit flip for q_{xk} and r_{xk} is symmetric under the bit flip). There is $\mathcal{O}(1)$ cost to making the computation controlled on the ancilla in E.

Steps 3 and 9. Outputting δ_β controlled on a single ancilla may be performed with CNOTS (no Toffoli cost) because δ_β is classically computed.

Steps 4 and 8. The inequality test is simply performed in the form $z<2^{s-1}(1+q_{xk})$ and similarly for r. There are no multiplications involved, because q_{xk} and r_{xk} are output as integer approximations. The inequality test has cost s Toffolis, so computation and uncomputation for the two steps has cost $2s$.

Step 5. This is just a Z gate with no Toffoli cost.

Step 6. The cost is two Toffolis to prepare a control qubit that flags whether the conditions required are satisfied. Then this qubit is used as a control register for the QROM on the value of k to apply a X operation to the target system. That QROM has complexity N.

Step 7. Controlled on the system K register being equal to k, we subtract k from it, and controlled on the system K register being 0 we add k to it. We then swap the registers with the results of these two equality tests. Since we still have the qubits with the results of the equality tests from step 1, we have no additional cost for that here. The cost of the two additions is $2\log N+\mathcal{O}(1)$.

The Toffoli cost of the steps is therefore $2s+N+6\log N+\mathcal{O}(1)$, plus two times the cost of the function evaluation and energy-difference oracles. Note that we pay four times the cost of the QROM lookup within the function-evaluation oracle, but we are regarding the cost as two function oracles because the QROM lookup cost is a smaller cost given in an order term. The cost of the preparation and inverse preparation is $8\log N+\mathcal{O}(1)$ Toffolis and $2.3b_{\rm rot}+\mathcal{O}(1)$ T gates, or just $2.3b_{\rm rot}+\mathcal{O}(1)$ T gates if N is a power of 2. Taking $s=b_{\rm sm}+\mathcal{O}(1)$, that gives total cost

$$2\mathcal{C}^{\rm diff}+2\mathcal{C}^{\rm fun}+2b_{\rm sm}+N+14\log N+\mathcal{O}(b_{\rm rot}), \tag{232}$$

where we have put the T cost in the order term. The ancilla cost is as follows.

1. Two qubits for the E and F ancillae.
2. Two qubits from the results of the two equality tests for the system K register.
3. The register with k for the control ancilla and that with k for the system each need $\lceil \log N \rceil$ qubits.
4. The register with z for the control ancilla needs s qubits.
5. The ancillas for the energy-difference oracle.
6. The ancillas for the function-evaluation oracle.

The number of qubits s used for z can be taken to be within $\mathcal{O}(1)$ of the number of qubits c used for q_{xk} or r_{xk}. We need temporary qubits for working, but the same working qubits as for the oracles can be used, so we do not count these ancilla costs again. The function-evaluation oracle may use more or less temporary ancilla than the energy difference, so we need to take the maximum of these two costs. That gives an ancilla cost of $2 \log N + b_{\rm sm} + \mathcal{O}(1)$ plus the ancilla costs of the two oracles, or

$$\mathcal{A}^{\rm diff} + \mathcal{A}^{\rm fun} + \max(\mathcal{B}^{\rm diff}, \mathcal{B}^{\rm fun}) + 2 \log N + b_{\rm sm} + \mathcal{O}(1). \tag{233}$$

IV. ERROR-CORRECTION ANALYSIS AND DISCUSSION

Previous sections of this paper discuss and optimize the compilation of various heuristic approaches to quantum optimization into cost models appropriate for quantum error correction. Specifically, we focus on reducing the Toffoli (and in some cases T) complexity of these algorithms while also keeping the number of ancilla qubits reasonable. This cost model is motivated by our desire to assess the viability of these heuristics within the surface code (the most practical error-correcting code suitable for a 2D array of physical qubits) [19,68–70]. T gates and Toffoli gates cannot be implemented transversely within practical implementations of the surface code. Instead, one must implement these gates by first distilling resource states. In particular, to implement a T gate one requires a T state ($|\mathrm{T}\rangle = \mathrm{T}\,|+\rangle$) and to implement a Toffoli gate one requires a controlled-controlled-Z (CCZ) state ($|\mathrm{CCZ}\rangle = \mathrm{CCZ}\,|+ + +\rangle$); in both cases these states are consumed during the implementation of the associated gates. Distilling T or CCZ states requires a substantial amount of both time and hardware.

Here, we analyze the cost to implement our various heuristic optimization primitives using the constructions of Ref. [38], which are based on applying the lattice surgery constructions of Ref. [71] to the fault-tolerant Toffoli protocols of Ref. [72,73]. We further assume a correlated-error minimum weight perfect-matching decoder capable of keeping pace with 1-μs rounds of surface code error detection [74], and capable of performing feedforward in about 15 μs. We assume that our physical hardware gates have error rates of either 10^{-3} or 10^{-4}, the former consistent with the best error rates demonstrated in state-of-the-art intermediate-scale superconducting qubit platforms [1] and the latter consistent with improvements in the technology that we hope is feasible in the next decade. Under these assumptions the spacetime volume required to implement one Toffoli gate or two T gates with two levels of state distillation and code distance $d = 31$ (which is safely sufficient for the computations we analyze here) is equal to roughly 26 qubitseconds [38]. For instance, to distill one CCZ state using the approach in Ref. [38] requires $5.5d + \mathcal{O}(1)$ cycles using a factory with a data-qubit footprint of about $12d \times 6d$ (the total qubit count includes measurement qubits, and so is roughly double this figure). Specifically, in our estimates we assume that executing a Toffoli gate requires about 170 microseconds and 150 000 physical qubits (see the resource-estimation spreadsheet included within the Supplemental Material of Ref. [38] for more detailed assumptions). Due to this large overhead we focus on estimates assuming that we distill CCZ states in series, which is likely how we would operate the first generation of fault-tolerant surface code computers.

In Tables IX and X we estimate the resources that are required to implement various heuristic optimization primitives within the surface code (given the assumptions of the prior paragraphs) for the Sherrington-Kirkpatrick and low autocorrelation binary sequences problems, respectively. We perform this analysis for the primitives of amplitude amplification, a first-order Trotter step (which can be used for QAOA, population transfer, the adiabatic algorithm, etc.), a qubitized Hamiltonian walk realized from the linear combinations of unitaries query model (which can be used for measuring energies in QAOA, performing population transfer, the adiabatic algorithm, etc.), the qubitized quantum walk approach to quantum simulated annealing ("LHPST walk") and the spectral-gap-amplified approach to quantum simulated annealing. The only primitive discussed in this paper omitted from these tables is the Szegedy walk approach to quantum simulated annealing. This is because we can see from Table VIII that the Szegedy walk approach is strictly less efficient than the qubitized variant, and requires so many ancilla that analyzing it under the assumption of serial state distillation seems unreasonable. Because we do not know how many times one needs to repeat these primitives to solve the various optimization problems, in Tables IX and X we report how many times one is able to implement these primitives for various system sizes, assuming maximum run times of one hour or one day (24 hours). We also report how many physical qubits are required to realize these computations assuming physical gate-error rates of 10^{-3} or (10^{-4}).

We focus on the SK and LABS cost functions primarily for concreteness. As seen in Tables V and VIII, the choice

TABLE IX. Estimates of resources required to implement steps of various heuristic algorithms for the SK model within the surface code. All error-correction overheads are reported assuming a single Toffoli factory using state distillation constructions from Ref. [38]. Surface code overheads in parenthesis assume a physical error rate of 10^{-4} whereas the overheads not in parenthesis assume a physical error rate of 10^{-3}. The target success probability is 0.9. These estimates are based on Table VIII where we somewhat arbitrarily choose to set all values of the parameter quantifying the number of bits of precision (b) that appear in the table to 20 except for b_{fun} and b_{sm}, which can be smaller so we take $b_{\mathrm{fun}} = b_{\mathrm{sm}} = 7$.

				One-hour runtime		One-day runtime	
Algorithm applied to Sherrington-Kirkpatrick model	Problem size, N	Logical qubits	Toffolis per step	Maximum steps	Physical qubits	Maximum steps	Physical qubits
Amplitude amplification	64	100	6.3×10^3	3.3×10^3	3.1×10^5 (1.8×10^5)	7.9×10^4	3.7×10^5 (2.0×10^5)
	128	170	2.6×10^4	7.9×10^2	4.2×10^5 (2.1×10^5)	1.9×10^4	5.2×10^5 (2.3×10^5)
	256	304	1.0×10^5	2.0×10^2	7.2×10^5 (3.0×10^5)	4.8×10^3	8.1×10^5 (3.0×10^5)
	512	566	4.6×10^5	4.5×10^1	1.2×10^6 (4.3×10^5)	1.1×10^3	1.4×10^6 (4.3×10^5)
	1024	1084	1.8×10^6	1.1×10^1	2.2×10^6 (7.0×10^5)	2.7×10^2	2.9×10^6 (8.8×10^5)
QAOA and first-order Trotter e.g., for population transfer or adiabatic algorithm	64	120	6.8×10^3	3.1×10^3	3.4×10^5 (1.9×10^5)	7.3×10^4	4.1×10^5 (2.1×10^5)
	128	190	2.7×10^4	7.7×10^2	5.0×10^5 (2.4×10^5)	1.9×10^4	5.6×10^5 (2.4×10^5)
	256	324	1.1×10^5	2.0×10^2	7.6×10^5 (3.1×10^5)	4.7×10^3	8.6×10^5 (3.1×10^5)
	512	586	4.6×10^5	4.5×10^1	1.2×10^6 (4.5×10^5)	1.1×10^3	1.4×10^6 (4.5×10^5)
	1024	1104	1.8×10^6	1.1×10^1	2.2×10^6 (7.1×10^5)	2.7×10^2	2.9×10^6 (8.9×10^5)
Hamiltonian walk e.g., for population transfer or adiabatic algorithm	64	94	3.8×10^2	5.4×10^4	3.0×10^5 (1.8×10^5)	1.3×10^6	3.5×10^5 (2.0×10^5)
	128	163	7.7×10^2	2.7×10^4	4.1×10^5 (2.1×10^5)	6.5×10^5	5.0×10^5 (2.3×10^5)
	256	296	1.5×10^3	1.4×10^4	7.0×10^5 (3.0×10^5)	3.3×10^5	8.0×10^5 (3.0×10^5)
	512	557	3.1×10^3	6.8×10^3	1.2×10^6 (4.3×10^5)	1.6×10^5	1.4×10^6 (4.3×10^5)
	1024	1074	6.1×10^3	3.4×10^3	2.2×10^6 (6.9×10^5)	8.1×10^4	2.9×10^6 (8.7×10^5)
LHPST-walk quantum simulated annealing	64	117	6.7×10^2	3.1×10^4	3.3×10^5 (1.9×10^5)	7.5×10^5	4.0×10^5 (2.1×10^5)
	128	185	9.0×10^2	2.3×10^4	4.4×10^5 (2.2×10^5)	5.6×10^5	5.5×10^5 (2.4×10^5)
	256	317	1.5×10^3	1.4×10^4	7.4×10^5 (3.1×10^5)	3.3×10^5	8.4×10^5 (3.1×10^5)
	512	577	2.6×10^3	8.1×10^3	1.2×10^6 (4.4×10^5)	2.0×10^5	1.4×10^6 (4.4×10^5)
	1024	1093	4.8×10^3	4.4×10^3	2.2×10^6 (7.0×10^5)	1.0×10^5	2.9×10^6 (8.9×10^5)
Spectral-gap-amplified walk-based quantum-simulated annealing	64	116	4.0×10^2	5.2×10^4	3.3×10^5 (1.9×10^5)	1.2×10^6	4.0×10^5 (2.1×10^5)
	128	185	6.4×10^2	3.3×10^4	4.4×10^5 (2.2×10^5)	7.8×10^5	5.5×10^5 (2.4×10^5)
	256	318	1.3×10^3	1.6×10^4	7.4×10^5 (3.1×10^5)	3.9×10^5	8.4×10^5 (3.1×10^5)
	512	579	2.3×10^3	9.0×10^3	1.2×10^6 (4.4×10^5)	2.2×10^5	1.4×10^6 (4.4×10^5)
	1024	1096	4.5×10^3	4.6×10^3	2.2×10^6 (7.0×10^5)	1.1×10^5	2.9×10^6 (8.9×10^5)

to focus on these specific problems rather than QUBO or the H_L model means that we do not need to choose a precision parameter in some cases. For example, with amplitude amplification we know how many bits of precision we should compute the energy to since SK and LABS both have integer-valued energies in a well-defined range. However, in order to produce specific numerical estimates for other primitives it is necessary to assume values for the precision parameters b appearing Table V (defined in Table IV); e.g., for the Trotter steps one must realize time evolutions of noninteger duration so that the phase is accurate to within some precision b_{pha}, which we must choose independently of the particular problem. Thus, in order to produce actual numerical estimates, in Tables IX and X we choose to set many variants of the free precision parameter b to 20; thus, $b = 20$ bits of precision. However, as discussed in previous sections, the parameters b_{fun} and b_{sm} can generally be chosen to be smaller than the other values of b without compromising precision; here we take $b_{\mathrm{fun}} = b_{\mathrm{sm}} = 7$.

It is tempting to directly compare the costs of the various primitives shown in Tables IX and X. While comparisons of the same primitives between the two problem types are straightforward (e.g., SK is more efficient than LABS in most, but not all, cases), comparisons between the different primitive types are challenging. Quantum simulated annealing, amplitude amplification, QAOA, population transfer, and the adiabatic algorithm are simply different algorithms so it is difficult to compare the relative values of a step of these algorithms.

It seems more reasonable to compare the Trotter steps to the qubitized Hamiltonian walk steps since these primitives can be used for the same ends (e.g., population transfer or the adiabatic algorithm). But first, the choice of $b = 20$ means something different for these two algorithms. And second, while the Hamiltonian walks are capable of more precise evolutions (scaling as $\mathcal{O}(\log 1/\epsilon)$ in terms of precision compared to the $\mathcal{O}(\mathrm{poly}(1/\epsilon))$ scaling of fixed-order Trotter-based methods), for heuristic optimization the evolution does not necessarily need to be

TABLE X. Estimates of resources required to implement steps of various heuristic algorithms for the LABS problem within the surface code. All overheads are reported assuming a single Toffoli factory using state distillation constructions from Ref. [38]. Surface-code overheads in parenthesis assume a physical error rate of 10^{-4} whereas the overheads not in parenthesis assume a physical error rate of 10^{-3}. Target success probability is 0.9. These estimates are based on Table VIII where we somewhat arbitrarily choose to set all values of the parameter quantifying the number of bits of precision (b) that appear in the table to 20 except for b_{fun} and b_{sm}, which can be smaller, so we take $b_{\mathrm{fun}} = b_{\mathrm{sm}} = 7$.

				One-hour runtime		One-day runtime	
Algorithm applied to LABS problem	Problem size, N	Logical qubits	Toffolis per step	Maximum steps	Physical qubits	Maximum steps	Physical qubits
Amplitude amplification	64	98	9.8×10^3	2.1×10^3	3.0×10^5 (1.8×10^5)	5.1×10^4	3.6×10^5 (2.0×10^5)
	128	167	3.7×10^4	5.6×10^2	4.1×10^5 (2.1×10^5)	1.3×10^4	5.1×10^5 (2.3×10^5)
	256	300	1.5×10^5	1.4×10^2	7.1×10^5 (3.0×10^5)	3.3×10^3	8.0×10^5 (3.0×10^5)
	512	561	6.1×10^5	3.4×10^1	1.2×10^6 (4.3×10^5)	8.2×10^2	1.4×10^6 (4.3×10^5)
	1024	1078	2.3×10^6	9.0×10^0	2.2×10^6 (6.9×10^5)	2.2×10^2	2.9×10^6 (8.8×10^5)
QAOA and first-order Trotter e.g., for population transfer or adiabatic algorithm	64	114	1.0×10^4	2.1×10^3	3.3×10^5 (1.9×10^5)	5.0×10^4	4.0×10^5 (2.1×10^5)
	128	183	3.8×10^4	5.5×10^2	4.4×10^5 (2.1×10^5)	1.3×10^4	5.5×10^5 (2.4×10^5)
	256	316	1.5×10^5	1.4×10^2	7.4×10^5 (3.1×10^5)	3.4×10^3	8.4×10^5 (3.1×10^5)
	512	577	5.0×10^5	4.2×10^1	1.2×10^6 (4.4×10^5)	1.0×10^3	1.4×10^6 (4.4×10^5)
	1024	1094	1.7×10^6	1.2×10^1	2.2×10^6 (7.0×10^5)	2.9×10^2	2.9×10^6 (8.9×10^5)
Hamiltonian walk e.g., for population transfer or adiabatic algorithm	64	94	2.6×10^2	8.1×10^4	3.0×10^5 (1.8×10^5)	2.0×10^6	3.5×10^5 (2.0×10^5)
	128	163	5.1×10^2	4.1×10^4	4.1×10^5 (2.1×10^5)	9.8×10^5	5.0×10^5 (2.3×10^5)
	256	296	1.0×10^3	2.0×10^4	7.0×10^5 (3.0×10^5)	4.9×10^5	8.0×10^5 (3.0×10^5)
	512	557	2.0×10^3	1.0×10^4	1.2×10^6 (4.3×10^5)	2.4×10^5	1.4×10^6 (4.3×10^5)
	1024	1074	4.1×10^3	5.1×10^3	2.2×10^6 (6.9×10^5)	1.2×10^5	2.9×10^6 (8.7×10^5)
LHPST-walk quantum simulated annealing	64	132	2.0×10^4	1.0×10^3	3.6×10^5 (2.0×10^5)	2.5×10^4	4.4×10^5 (2.1×10^5)
	128	202	7.5×10^4	2.8×10^2	5.3×10^5 (2.5×10^5)	6.7×10^3	5.9×10^5 (2.5×10^5)
	256	336	3.0×10^5	6.9×10^1	7.8×10^5 (3.2×10^5)	1.7×10^3	8.8×10^5 (3.2×10^5)
	512	598	1.2×10^6	1.7×10^1	1.3×10^6 (4.5×10^5)	4.1×10^2	1.5×10^6 (4.5×10^5)
	1024	1116	4.6×10^6	5.0×10^0	2.2×10^6 (7.1×10^5)	1.1×10^2	3.0×10^6 (9.0×10^5)
Spectral-gap-amplified walk-based quantum-simulated annealing	64	131	2.0×10^4	1.1×10^3	3.6×10^5 (2.0×10^5)	2.5×10^4	4.3×10^5 (2.1×10^5)
	128	202	7.5×10^4	2.8×10^2	5.3×10^5 (2.5×10^5)	6.7×10^3	5.9×10^5 (2.5×10^5)
	256	337	3.0×10^5	6.9×10^1	7.8×10^5 (3.2×10^5)	1.7×10^3	8.8×10^5 (3.2×10^5)
	512	600	1.2×10^6	1.7×10^1	1.3×10^6 (4.5×10^5)	4.1×10^2	1.5×10^6 (4.5×10^5)
	1024	1119	4.6×10^6	5.0×10^0	2.2×10^6 (7.2×10^5)	1.1×10^2	3.0×10^6 (9.0×10^5)

precise, so the Trotter approach may be more efficient by using large steps. The Trotter steps can be made arbitrarily large without increasing gate count (although at a cost of less precision), whereas the Hamiltonian walk effectively simulates time of at most $1/\lambda$ where $\lambda_{\mathrm{SK}} \approx N^2/2$ and $\lambda_{\mathrm{LABS}} \approx N^3/3$ (but it does so quite precisely). Thus, although the Hamiltonian walk steps require the fewest Toffolis in Table X, they may still be less efficient than other approaches.

For the various forms of quantum simulated annealing, the number of steps needed is governed by the spectral gap. The qubitized annealing (LHPST) and Szegedy approaches are directly comparable because they have the same gap, which means the same number of steps should be sufficient. This means that the smaller step cost of LHPST means that it is more efficient. The spectral-gap-amplified approach has a similar gap as the LHPST and Szegedy approaches, because it provides a similar square-root improvement. The problem is that the Hamiltonian has a λ value proportional to $\sqrt{N}$, as shown in Eq. (230). This increases the cost of implementing the Hamiltonian by a factor of $\sqrt{N}$, so the cost given for a single step should be multiplied by $\sqrt{N}$ for a fair comparison with the other simulated annealing approaches. When that is taken into account, the spectral-gap-amplified approach is less efficient.

With these caveats and context, we believe that Tables IX and X give a rough sense for the feasibility of implementing these various heuristic optimization primitives on a small fault-tolerant surface code quantum processor. In most cases one can attempt these algorithms up to roughly a thousand bits with around a million physical qubits or less (especially given 10^{-4} error rates). However, we can see that the significant overheads of state distillation make the execution of these algorithms painfully slow. The quantum simulated annealing steps are often more efficient to implement than most other steps. The one exception is the Hamiltonian walk steps, which are highly efficient. But again, there it is likely that the large value of λ means that many more Hamiltonian walk steps are required.

We see that for SK-model problem sizes between $N = 64$ and $N = 1024$ one can perform between about 4×10^3 and 3×10^4 quantum simulated annealing updates per hour. As a comparison, the work of Ref. [75] discusses the implementation of a very performant classical simulated annealing code for optimizing sparse spin glasses. This same code deployed to an $N = 512$ spin instance of SK is capable of performing a simulated annealing update step in an average of 7 CPU ns [76] (this average accounts for the fact that most updates for the Sherrington-Kirkpatrick model are rejected). This works out to about 6×10^{11} attempted updates per core hour, or about one-hundred million times more steps than the quantum computer can implement in that same amount of time for an $N = 512$ spin SK model. The state produced after the 2×10^5 quantum simulated annealing steps that our quantum computer can make in one day for the $N = 512$ spin SK model could be produced by a single classical core in about 4 CPU min, assuming that the classical algorithm requires exactly quadratically more (4×10^{10}) steps. The comparison is even less favorable for quantum computing if we consider larger problem sizes. Furthermore, given the high costs of quantum computing, it is unclear why we should restrict the classical competition to one core rather than to millions of cores.

The quantum computer must give a speedup for a sufficiently difficult problem if we assume a quadratic speedup in the number of annealing steps required. For the $N = 512$ spin SK model, by comparing the number of steps that the classical algorithm from Ref. [75] can make in one hour (5×10^{11}) to the number of steps that the quantum algorithm can make in one hour (8×10^3), we can estimate a crossover point. In particular, solving $M/(8 \times 10^3) = M^2/(5 \times 10^{11})$ yields $M \approx 7 \times 10^7$ as the minimum number of steps that are required for the quantum algorithm to give an advantage. Unfortunately, this means the quantum computer needs to run for a number of hours that is $7 \times 10^7/(8 \times 10^3)$, which works out to about one year. Moreover, this analysis is very favorable to the quantum computer in that (1) it does not adjust the surface code distance (and thus, resource overheads) for runtimes longer than an hour, (2) it compares to a single classical core, and (3) it assumes that $N = 512$ is a large enough instance to warrant this many steps in some cases. Of course, most $N = 512$ instances of the SK model can be solved with much less than a CPU year of simulated annealing run time, thus precluding the possibility of a quantum speedup for most instances at that size under the assumptions of our analysis.

Comparisons for amplitude amplification are similarly discouraging. For these two problems one can perform between about ten and three thousand steps of amplitude amplification using between about one-hundred thousand and one-million qubithours of state distillation. In the same amount of time one could conservatively check hundreds of billions of solutions on even a single core of a classical computer. Assuming the quantum computer requires quadratically fewer steps of amplitude amplification (still at least a hundred thousand steps) compared to random classical energy queries, we still need roughly billions of qubithours of state distillation in order to compete with what a single core of a classical computer can do in one hour. Once again, if we instead make our comparisons to a classical supercomputing cluster rather than to a single classical core, the overheads appear even more daunting.

The LABS problem is an example where the scaling of the best known classical algorithm is worse than $\mathcal{O}(2^{N/2})$ and thus, an approach based on amplitude amplification has better scaling. In particular, the best scaling method in the literature goes as $\Theta(1.73^N)$ [34]. That scaling is obtained for a branch-and-bound type method that queries the effect of local spin flips (and thus, not the entire objective function). Each of these queries is slightly faster than requiring 7 CPU μs with an optimized classical implementation for $N = 64$ (about 5×10^8 steps per hour). If we were to compete with this approach using amplitude amplification on a quantum computer (where we can perform about 2×10^3 steps per hour at $N = 64$) then we can approximate the crossover point as $2^{M/2}/(2 \times 10^3) = 1.73^M/(5 \times 10^8)$ so long as we remember that these numbers are only valid in the vicinity of $M \approx N = 64$. Coincidentally, that is the case as we find that $M = 62$, which corresponds to about 2×10^9 queries, which would take about 116 years. Once again, here we are being generous to the quantum computer by making comparisons to a single core and not adjusting the code distance for long runtimes. Still, we again see that a small error-corrected quantum computer cannot compete with classical methods under such a modest scaling advantage.

The heuristics based on Trotter steps or qubitized-walk LCU queries are more difficult to compare to classical competition since algorithms such as QAOA, the adiabatic algorithm, or population transfer lack a clear classical analog. In that sense, it is not straightforward to predict what being able to perform a few hundred Trotter steps or a few thousand qubitized walk steps in an hour might buy us, but it is clear that these are able to perform only very short quantum walks or time evolutions, or very inaccurate time evolutions. Eventually, it is at least possible to find out by using our constructions to realize these algorithms on a small fault-tolerant quantum computer and experimentally discovering what happens. We note that for these algorithms the number of steps should be interpreted as the product of the number of repetitions of the primitive and the total number of times the algorithm is repeated. For instance, we see that for either the SK model or LABS at $N = 256$, slightly more than 100 Trotter steps can be implemented in an hour. In the context of QAOA, this could mean that we run QAOA at $p = 100$ and draw one sample, or we run QAOA at $p = 10$ and draw ten samples or we run QAOA at $p = 1$ and draw one-hundred samples, etc. However, as we have explained in Sec. III A and Sec. III B one is probably better off using coherent repetitions in the context of an amplitude-amplification-like scheme rather than making classical repetitions.

Although we try to optimize the realization of these heuristic primitives for the cost functions considered in this paper, clever improvements to our approaches might further reduce the resources required. However, we expect the complexity of these primitives to be no better than N. In particular, LCU-based methods require a minimum of $N - 1$ Toffolis just to access N qubits in a controlled way. For Trotter step methods, evolution under the problem Hamiltonian could be below N for a particularly simple problem Hamiltonian, but then the evolution under the transverse-field driver Hamiltonian is the dominant cost and requires $\mathcal{O}(N)$ non-Clifford gates. For amplitude amplification, one could again have a small cost for a particularly simple problem Hamiltonian, but amplitude amplification requires a reflection on at least N qubits, with cost at least $N - 2$ Toffolis.

We are already at about $5N$ for the LHPST walk with SK, so we do not expect more than about a factor of 5 improvement even for the easiest problem. If we were to use the sum of bits directly as in Ref. [25], then the complexity is about $2N$, but another N ancilla qubits is needed. One could also propose to use a larger fault-tolerant quantum computer and distill more Toffoli states in parallel. But even if this strategy is pursued to the fullest extent possible (requiring tens or hundreds of millions of physical qubits) and parallelized near optimally, the surface code is then bottlenecked by Clifford gates (or the overhead of routing), which are, at best, only about a hundred to a thousand times faster to implement.

In conclusion, we optimize and compile the basic primitives required for many popular heuristic algorithms for quantum optimization to a cost model appropriate for practical quantum error-correction schemes. This allows us to assess and compare the cost of several quantum algorithms that have not previously been compiled in such detail. We focus on doing this for only a subset of the possible cost function structures that one might hope to algorithmically exploit for more efficient implementations, but our constructions led to the development of various methodologies, which we expect is useful in a more general context. For instance, we expect that work outside the context of quantum optimization might benefit from our strategy of interpolating arithmetic functions using an adaptive QROM. However, despite our attempts at optimization, the concrete resource estimates from Tables IX and X are predictably discouraging. The essential reason for this is the substantial constant factor slowdown between error-corrected quantum computation and classical computation. Based on these numbers we strongly suspect that in order for early fault-tolerant quantum computers to have a meaningful impact on combinatorial optimization, we either need quantum optimization algorithms that afford speedups, which are much better than quadratic, or we need significant improvements in the way that we realize error correction.

ACKNOWLEDGMENTS

The authors thank Sergio Boixo, Austin Fowler, Sergei Isakov, Kostyantyn Kechedzhi, Mária Kieferová, Jessica Lemieux, Jarrod McClean, John Platt, and Vadim Smelyanskiy for helpful discussions. Y.R.S., D.W.B., P.C.S.C, and N.W. acknowledge funding for this work from a Grant from Google Quantum. D.W.B. is also funded by an Australian Research Council Discovery Project DP190102633.

APPENDIX A: ADDITION FOR CONTROLLED ROTATIONS

Here we give more details on how to perform phase rotations using the method from Ref. [35,36]. Prior to the simulation the following state is prepared

$$|\phi\rangle = \frac{1}{\sqrt{2^{b_{\mathrm{grad}}}}} \sum_{k=0}^{2^{b_{\mathrm{grad}}}-1} e^{-2\pi i k/2^{b_{\mathrm{grad}}}} |k\rangle . \tag{A1}$$

This state is a tensor product of the form

$$|\phi\rangle = \frac{1}{\sqrt{2^{b_{\mathrm{grad}}}}} \bigotimes_{j=1}^{b_{\mathrm{grad}}} \left(|0\rangle + e^{-2\pi i/2^j} |1\rangle\right) . \tag{A2}$$

It can be prepared using standard techniques for performing rotations on qubits. To obtain overall error ϵ, each rotation should be performed with error $\epsilon/b_{\mathrm{grad}}$, which has complexity $\mathcal{O}(\log(b_{\mathrm{grad}}/\epsilon))$ [37], giving overall complexity $\mathcal{O}(b_{\mathrm{grad}} \log(b_{\mathrm{grad}}/\epsilon))$ to prepare this state. Because this state only need be prepared once, this complexity is negligible compared to the complexities in other parts of the algorithm.

Adding a value ℓ into this register gives

$$\begin{aligned} &\frac{1}{\sqrt{2^{b_{\mathrm{grad}}}}} \sum_{k=0}^{2^{b_{\mathrm{grad}}}-1} e^{-2\pi i k/2^{b_{\mathrm{grad}}}} |k+\ell\rangle \\ &\quad = \frac{1}{\sqrt{2^{b_{\mathrm{grad}}}}} \sum_{k=0}^{2^{b_{\mathrm{grad}}}-1} e^{-2\pi i (k-\ell)/2^{b_{\mathrm{grad}}}} |k\rangle = e^{2\pi i \ell/2^{b_{\mathrm{grad}}}} |\phi\rangle . \end{aligned} \tag{A3}$$

This is why the addition yields a phase factor. Moreover, the value of ℓ can be stored in a quantum register, in order to make this a controlled phase. In order to make a controlled rotation on a qubit, we can perform the addition controlled by this qubit. Then one obtains

$$(\mu\,|0\rangle + \nu\,|1\rangle)\,|\ell\rangle\,|\phi\rangle \mapsto (\mu\,|0\rangle + e^{2\pi i\ell/2^{b_{\rm grad}}}\nu\,|1\rangle)\,|\ell\rangle\,|\phi\rangle\,. \tag{A4}$$

This approach is somewhat inefficient, because controlled addition has twice the complexity of addition.

Instead we can use the trick described in Sec. II 1, which enables a qubit to control whether addition or subtraction is performed with only Clifford gates. The qubit simply needs to control CNOTs on the target system before and after the addition. Then we obtain

$$(e^{-2\pi i\ell/2^{b_{\rm grad}}}\mu\,|0\rangle + e^{2\pi i\ell/2^{b_{\rm grad}}}\nu\,|1\rangle)\,|\ell\rangle\,|\phi\rangle\,. \tag{A5}$$

This procedure therefore enables us to perform the rotation $e^{-2\pi i\ell Z/2^{b_{\rm grad}}}$ with $b_{\rm grad}-2$ Toffolis. This approach is far more efficient than techniques based on rotation synthesis with T gates when the rotation angle is given in a quantum register, because those techniques need a separate rotation controlled on each bit. When the rotation angle is given classically, this technique is slightly less efficient than rotation synthesis with T gates as in Ref. [37], because Toffolis have a cost equivalent to two T gates in magic state distillation [38]. On the other hand, rotation angles that are integer multiples of $2\pi/2^{b_{\rm grad}}$ can be performed exactly, up to the accuracy of synthesizing the resource state $|\phi\rangle$.

To obtain a rotation that performs the mapping

$$|0\rangle \mapsto \cos(2\pi\ell/2^{b_{\rm grad}})\,|0\rangle + \sin(2\pi\ell/2^{b_{\rm grad}})\,|1\rangle\,, \tag{A6}$$

one can simply perform the operations $SHe^{-2\pi i\ell Z/2^{b_{\rm grad}}}H$. Here the Hadamard H and S gates are Clifford gates, so this gives the state preparation with the only Toffoli cost in synthesizing the Z rotation.

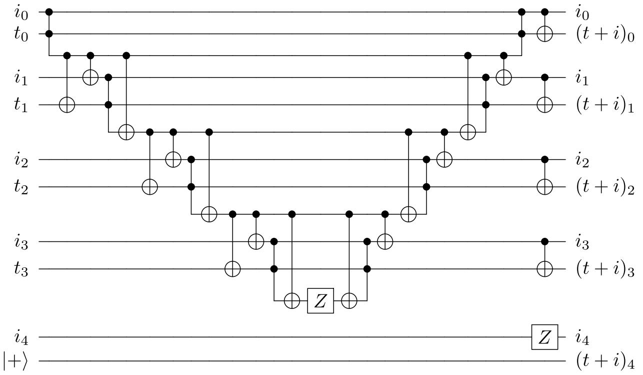

FIG. 11. A circuit to perform addition on 5 qubits modulo 2^5 when the most significant target qubit is in a $|+\rangle$ state.

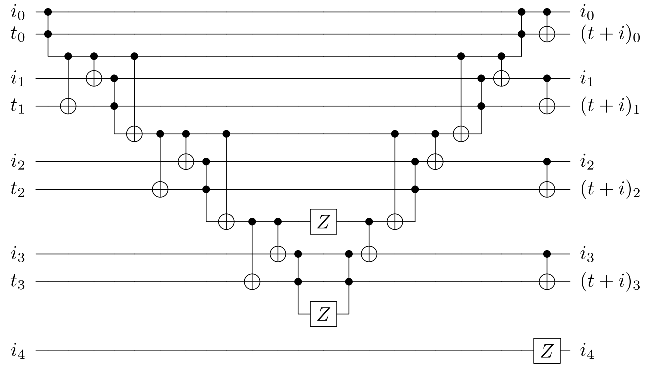

FIG. 12. A simplification of Fig. 11 to eliminate the CNOTs on the last carry qubit. The $|+\rangle$ state is omitted here because it is not acted upon.

The complexity of performing the addition is only $b_{\rm grad}-2$ rather than $b_{\rm grad}-1$, as is normally the case for addition of $b_{\rm grad}$-bit numbers (modulo $2^{b_{\rm grad}}$). The reason is that the most significant qubit of $|\phi\rangle$ is in a $|+\rangle$ state, so NOT gates on this qubit can be replaced with phase gates, and this qubit can be discarded. Doing that yields the circuit shown in Fig. 11. The Toffoli is not immediately saved, but the CNOTs and Z gate on the final carry qubit can be replaced with two Z gates as shown in Fig. 12. Then the Toffoli used on the final carry qubit can simply be replaced with a controlled phase, as shown in Fig. 13. The resulting complexity is $b_{\rm grad}-2$ Toffolis. If the angle to be rotated by is given as a classical variable, then the cost is further reduced to $b_{\rm grad}-3$ Toffolis, because addition of a classical number takes one fewer Toffoli. This means that $b_{\rm grad}=4$, which gives a T gate, takes one Toffoli.

Next we consider the case that we need to multiply an integer k with b bits by a constant $\tilde{\gamma}$ to give the phase. Given that $\tilde{\gamma}$ is represented on n bits, we can write $\tilde{\gamma}$ as a sum of no more than $\lceil(n+1)/2\rceil$ powers of 2, with positive and negative signs. This formula is checked in Fig. 14. To prove the formula, assume that it is true for numbers

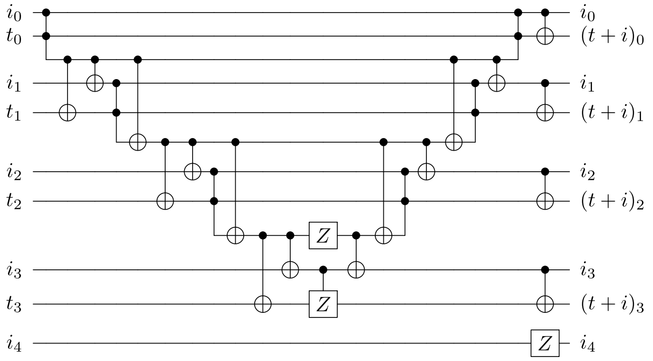

FIG. 13. A simplification of Fig. 12 where the last carry qubit is eliminated entirely and the Toffoli is replaced with a controlled phase.

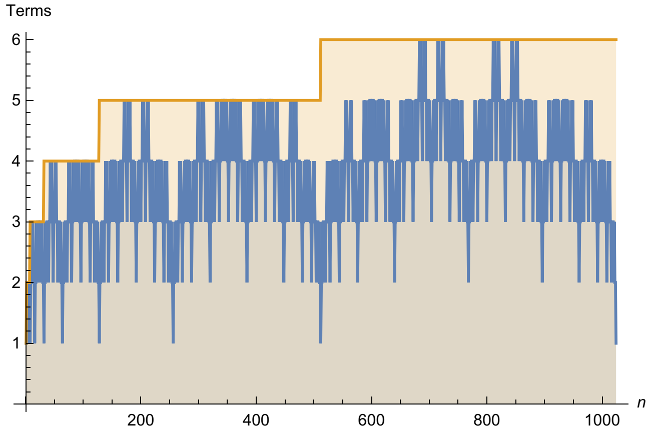

FIG. 14. In orange is the number of powers of 2 needed to give integer m, when we allow additions and subtractions. The formula $\lceil (n+1)/2 \rceil$ is shown in orange, where the number of bits required to represent m is $n = \lceil \log(m+1) \rceil$.

with $\le n_0$ bits, and consider a number m with $n = n_0 + 2$ bits (so the most significant bit must be a 1). There are then three cases to consider.

1. For $m < (3/4)2^n$, we find that $m - 2^{n-1} < 2^{n-2}$, so $m - 2^{n-1}$ has no more than $n - 2 = n_0$ bits, and so can be written as a sum of at most $\lceil (n_0+1)/2 \rceil$ powers of 2. That means m can be written as a sum of at most $\lceil (n_0+1)/2 \rceil + 1 = \lceil (n+1)/2 \rceil$ powers of 2.
2. For $m > (3/4)2^n$, we have $2^n - m < 2^{n-2}$, so $2^n - m$ has no more than $n - 2 = n_0$ bits, and can be written as a sum of $\le \lceil (n_0+1)/2 \rceil$ powers of two. Since m can be written as 2^n minus $2^n - m$, it can be written as at most $\lceil (n_0+1)/2 \rceil + 1 = \lceil (n+1)/2 \rceil$ powers of 2.
3. The last case is that where $m = (3/4)2^n$, so $m = 2^{n-1} + 2^{n-2}$. Since $n = n_0 + 2 \ge 2$, $\lceil (n+1)/2 \rceil \ge 2$, so again m is written as a sum of at most $\lceil (n+1)/2 \rceil$ powers of two.

Since we check that the formula is true for small numbers of bits in Fig. 14, the formula is true for all n by induction. To perform the multiplication, we take each term in the sum for $\tilde{\gamma}$, and add or subtract a bit-shifted copy of k to the phase-gradient state. We have no more than $(n+2)/2$ additions/subtractions, each of which is into the phase-gradient state with b_{grad} bits, which gives a cost of no more than $(b_{\text{grad}} - 2)(n+2)/2$.

The error due to omitted bits in the multiplication (those omitted in bit-shifting k) can be bounded as follows. First, note that the error for the additions is entirely in underestimating the product, since we are omitting digits. For the subtractions the error is in overestimating the product. Therefore, to obtain the maximum error we need to consider the case with entirely additions, since the subtractions cancel the error. For each addition the error is upper bounded by $2\pi/2^{b_{\text{grad}}}$, because we omit adding bits that correspond to phase shifts of $2\pi/2^{b_{\text{grad}}+1}$, $2\pi/2^{b_{\text{grad}}+2}$, and so forth. That means the upper bound on the error from $(n+2)/2$ additions is $(n+2)\pi/2^{b_{\text{grad}}}$. To make the error in the multiplication no larger than ϵ we should take

$$b_{\text{grad}} = \lceil \log[(n+2)\pi/\epsilon] \rceil = \log(n/\epsilon) + \mathcal{O}(1). \qquad \text{(A7)}$$

APPENDIX B: DISCRETIZING ADIABATIC STATE PREPARATION WITH QUBITIZATION

Here we place bounds on the error for the method of adiabatic evolution from Sec. III 2. For any fixed value of r we can choose an adiabatic path between an initial Hamiltonian and a final Hamiltonian. The accuracy of the adiabatic approximation depends strongly on how quickly we traverse this path, so it is customary to introduce a dimensionless time $s = t/T$, which allows us to easily change the speed without altering the shape of the path.

Using Trotter-Suzuki formulas for time-ordered operator exponentials we have that

$$\left\| \mathcal{T} e^{-iT\int_s^{s+1/r} H_{\text{eff}}(s)ds} - e^{-iH_{\text{eff}}(s+1/2r)T/r} \right\| \in \mathcal{O}\left(\frac{\max_s \|\partial_s^2 H_{\text{eff}}(s)\| T + \max_s \|\partial_s H_{\text{eff}}(s)\| \|H_{\text{eff}}(s)\| T^2}{r^3} \right). \qquad \text{(B1)}$$

However, the Hamiltonian H_{eff} for the time-evolution operator in this case is not known except in terms of its action on the space containing the instantaneous eigenvectors of H. In order to use this result, we need to bound the derivatives acting on the entire space. In order to find an asymptotic bound on these derivatives we define,

$$H_{\text{eff}}(s) = \frac{ir}{4} \ln\left[W_r^4(s)\right] = \frac{ir}{4} \ln \left\{ \left[(I - 2I \otimes |L(r,s)\rangle\langle L(r,s)|)\, \text{SELECT}\right]^4 \right\}. \qquad \text{(B2)}$$

It is then clear from the unitarity of $W_r(s)$ that for any $|s| \in \mathcal{O}(1)$, if we choose the principal logarithm for H_{eff} then $\|H_{\text{eff}}(s)\| \in \mathcal{O}(1)$. The derivatives of the Hamiltonian are more involved to estimate.

1. Derivatives of matrix logarithms of unitary matrices

In order to compute the derivatives of the effective Hamiltonian, we need to compute the derivatives of the logarithm function. Such an analysis is usually based on differentiating the Mercator series for the matrix logarithm; however, the Mercator series of $\log(A)$ does not converge for $\|A\| \ge 1$. For greater generality we use an integral representation for the matrix logarithm $\log(A)$ from Ref. [77],

$$\log(A) = \int_0^1 dt(A - \mathbb{1})[t(A - \mathbb{1}) + \mathbb{1}]^{-1}. \tag{B3}$$

This representation converges unless there is a real nonpositive eigenvalue. For the case where A is unitary, this requirement prohibits matrices that have any eigenvalues equal to precisely -1. Next, defining $V := [t(A - \mathbb{1}) + \mathbb{1}]^{-1}$, and in turn $(A - \mathbb{1}) = t^{-1}(V^{-1} - \mathbb{1})$ this expression simplifies to

$$\begin{aligned}\log(A) &= \int_0^1 dt(A - \mathbb{1})V \\ &= \int_0^1 dt\frac{1}{t}\left(\mathbb{1} - V\right).\end{aligned} \tag{B4}$$

Next, note that for any invertible matrix-valued function $A(s)$ we have from the product rule that

$$\partial_s[A(s)A^{-1}(s)] = 0 \Rightarrow \partial_s[A^{-1}(s)] = -A^{-1}(s)\dot{A}(s)A^{-1}(s). \tag{B5}$$

Using $\partial_s V = -tV\dot{A}V$ we get

$$\partial_s \log[A(s)] = \int_0^1 dtV\dot{A}V. \tag{B6}$$

Taking the derivative of Eq. (B6) gives

$$\begin{aligned}\partial_s^2 \log(A(s)) &= \int_0^1 dt\left(\dot{V}\dot{A}V + V\ddot{A}V + V\dot{A}\dot{V}\right) \\ &= \int_0^1 dt\left(-tV\dot{A}V\dot{A}V + V\ddot{A}V - tV\dot{A}V\dot{A}V\right) \\ &= \int_0^1 dtV\left(\ddot{A} - 2t\dot{A}V\dot{A}\right)V.\end{aligned} \tag{B7}$$

We can use the fact that A is unitary to see that

$$\begin{aligned}\|V\|^2 &= \left\|[t(A - \mathbb{1}) + \mathbb{1}][t(A^\dagger - \mathbb{1}) + \mathbb{1}]\right\|^{-1} \\ &= \left\|[t^2 + (t-1)^2]\mathbb{1} + (A + A^\dagger)\left(t - t^2\right)\right\|^{-1}.\end{aligned} \tag{B8}$$

In the case where A is close to the identity, if the absolute values of the phases of the eigenvalues are no greater than Γ, then

$$\|\partial_s \log[A(s)]\| \le \frac{\Gamma}{\sin\Gamma}\|\dot{A}\|, \tag{B9}$$

$$\|\partial_s^2 \log[A(s)]\| \le \frac{\Gamma}{\sin\Gamma}\|\ddot{A}\| + \frac{1}{\cos^2(\Gamma/2)}\|\dot{A}\|^2. \tag{B10}$$

Next we consider $W_r^2(t)$ in the case where the Hamiltonian is a linear combination of self-inverse unitaries so $\text{SELECT}'^2 = \mathbb{1}$, which is the case for all Hamiltonians considered here. Expanding it out we have

$$\begin{aligned}W_r^2(s) &= [\mathbb{1} - 2\mathbb{1}\otimes|L(s,r)\rangle\langle L(s,r)|]\text{SELECT}'[\mathbb{1} - 2\mathbb{1}\otimes|L(s,r)\rangle\langle L(s,r)|]\text{SELECT}' \\ &= [\mathbb{1} - 2\mathbb{1}\otimes|L(s,r)\rangle\langle L(t,r)|][\mathbb{1} - 2\text{SELECT}'\,|L(s,r)\rangle\langle L(s,r)|\,\text{SELECT}'] \\ &= \mathbb{1} - 2\mathbb{1}\otimes|L(t,r)\rangle\langle L(t,r)| - 2\text{SELECT}'\,|L(s,r)\rangle\langle L(s,r)|\,\text{SELECT}' \\ &\quad + 4\,|L(s,r)\rangle\langle L(s,r)|\,\text{SELECT}'\,|L(s,r)\rangle\langle L(s,r)|\,\text{SELECT}'.\end{aligned} \tag{B11}$$

Using

$$|L(s,r)\rangle = \sum_k \sqrt{\frac{\lambda_k(s)}{\lambda(s)r}}\,|k\rangle\,|00\rangle + \sqrt{\frac{r-1}{2r}}\,|0\rangle\left(|10\rangle + |11\rangle\right), \tag{B12}$$

we have

$$\langle L(s,r)|\,\text{SELECT}'\,|L(s,r)\rangle = \frac{H(s)}{\lambda(s)r}. \tag{B13}$$

Then squaring again gives

$$\begin{aligned}W_r^4(s) &= \mathbb{1} + 4\mathbb{1}\otimes|L(s,r)\rangle\langle L(s,r)| + 4\text{SELECT}'\,|L(s,r)\rangle\langle L(s,r)|\,\text{SELECT}' \\ &\quad + 16\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]^3\langle L(s,r)|\,\text{SELECT}'\end{aligned}$$

$$
\begin{aligned}
&-4\mathbb{1}\otimes|L(s,r)\rangle\langle L(s,r)| - 4\textsc{select}'\,|L(s,r)\rangle\langle L(s,r)|\,\textsc{select}'\\
&+8\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]\langle L(s,r)|\,\textsc{select}'\\
&+4\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]\langle L(s,r)|\,\textsc{select}' + 4\textsc{select}'\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]\langle L(s,r)|\\
&-8\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]\langle L(s,r)|\,\textsc{select}' - 8\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]^2\langle L(s,r)|\\
&-8\textsc{select}'\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]^2\langle L(s,r)|\,\textsc{select}' - 8\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]\langle L(s,r)|\,\textsc{select}'\\
=\;&\mathbb{1}+16\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]^3\langle L(s,r)|\,\textsc{select}' - 8\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]^2\langle L(s,r)|\\
&-8\textsc{select}'\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]^2\langle L(s,r)|\,\textsc{select}'\\
&-4\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]\langle L(s,r)|\,\textsc{select}' + 4\textsc{select}'\,|L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]\langle L(s,r)|\\
=\;&\mathbb{1}+4\left\{\textsc{select}', |L(s,r)\rangle\left[\frac{H(s)}{\lambda(s)r}\right]\langle L(s,r)|\right\}+\mathcal{O}\left(\frac{1}{r^2}\right).
\end{aligned}
\tag{B14}
$$

This means that $\|W_r^4(s)-\mathbb{1}\|\le 8/r+16/r^2+16/r^3$, so for $r\gtrsim 5.7$, W_r^4 does not have negative real eigenvalues, and our expression for the matrix logarithm holds. This also implies that

$$
\|\log[W_r^4(s)]\|\le\int_0^1 dt\|(A-\mathbb{1})V\|\le 8/r+\mathcal{O}(1/r^2).
\tag{B15}
$$

Next, under these assumptions we can use Eqs. (B9) and (B14) to show that [neglecting terms of $\mathcal{O}(r^{-2})$, which are negligible for large r and using $\|H\|/\lambda\le 1$]

$$
\begin{aligned}
\|\partial_s\log[W_r^4(s)]\| &\in \mathcal{O}\left(\|\partial_s W_r^4(s)\|\right)\\
&\subseteq \mathcal{O}\left(\big\||\dot{L}(s,r)\rangle\big\|\left\|\frac{H(s)}{\lambda(s)r}\right\|+\left\|\frac{\partial}{\partial s}\frac{H(s)}{\lambda(s)r}\right\|\right)\\
&\subseteq \mathcal{O}\left(\frac{\big\||\dot{L}(s,r)\rangle\big\|}{r}+\frac{|\dot{\lambda}|+\|\dot{H}\|}{\lambda r}\right).
\end{aligned}
\tag{B16}
$$

We observe from Eq. (142) and the definition of the Euclidean norm it follows that if the Hamiltonian is chosen to be independent of r then

$$
\|\,|\dot{L}\rangle\,\|\in\mathcal{O}\left(\sqrt{\sum_k\left|\frac{\partial}{\partial s}\sqrt{\frac{\lambda_k}{\lambda r}}\right|^2}\right)\subseteq\mathcal{O}(1/\sqrt{r}).
\tag{B17}
$$

Thus neglecting terms of order $\mathcal{O}(r^{-3/2})$ we find from substituting this expression into Eq. (B16) that

$$
\|\partial_s\log[W_r^4(s)]\|\in\mathcal{O}\left(\frac{|\dot{\lambda}|+\|\dot{H}\|}{\lambda r}\right).
\tag{B18}
$$

It further follows from Eqs. (B10) and (B14) that the second derivative of the matrix logarithm obeys [neglecting terms order $\mathcal{O}(r^{-3/2})$ and higher]

$$\begin{aligned}
\|\partial_s^2 \log[W_r^4(s)]\| &\in \mathcal{O}\left(\|\partial_s^2 W_r^4(s)\| + \|\partial_s W_r^4(s)\|^2\right)\\
&= \mathcal{O}(\|\partial_s^2 W_r^4(s)\|)\\
&\subseteq \mathcal{O}\left(\frac{\| |\ddot{L}(s,r)\rangle \|}{r} + \left\|\frac{\partial^2}{\partial s^2}\frac{H}{\lambda r}\right\| + \| |\dot{L}\rangle \| \left\|\frac{\partial}{\partial s}\frac{H}{\lambda r}\right\|\right)\\
&\subseteq \mathcal{O}\left(\frac{\| |\ddot{L}(s,r)\rangle \|}{r} + \left\|\frac{\partial^2}{\partial s^2}\frac{H}{\lambda r}\right\| + \frac{1}{r^{3/2}}\right)\\
&\subseteq \mathcal{O}\left(\frac{\| |\ddot{L}(s,r)\rangle \|}{r} + \left(\frac{\|\ddot{H}\| + (\|\dot{H}\| + |\dot{\lambda}|)\left(\frac{|\dot{\lambda}|}{\lambda}\right) + |\ddot{\lambda}|}{\lambda r}\right)\right).
\end{aligned} \tag{B19}$$

Note that in the above derivation terms of the form $\| |\dot{L}(s,r)\rangle \| \|\partial_s H/(\lambda r)\|$ are dropped because they are $\mathcal{O}(r^{-3/2})$. Again, if the Hamiltonian is chosen to be independent of r, then

$$\| |\ddot{L}(s,r)\rangle \| \in \mathcal{O}\left(\sqrt{\sum_k \left|\frac{\partial^2}{\partial s^2}\sqrt{\frac{\lambda_k}{\lambda r}}\right|^2}\right) \subseteq \mathcal{O}\left(\frac{1}{\sqrt{r}}\right), \tag{B20}$$

which implies that, neglecting terms of $\mathcal{O}(r^{-3/2})$ and higher

$$\|\partial_s^2 \log[W_r^4(s)]\| \in \mathcal{O}\left(\frac{\|\ddot{H}\| + (\|\dot{H}\| + |\dot{\lambda}|)\left(\frac{|\dot{\lambda}|}{\lambda}\right) + |\ddot{\lambda}|}{\lambda r}\right). \tag{B21}$$

Next we bound the error that arises from approximating the time-ordered operator exponential by the exponential of the effective Hamiltonian evaluated at the midpoint. From the analysis of the midpoint rule for integration, we intuitively expect that the error should scale as $\mathcal{O}(1/r^3)$; however, such an analysis cannot be directly applied here because of the fact that the derivatives of the Hamiltonian need not commute with the Hamiltonian. It can be seen by performing a Taylor series expansion of the effective Hamiltonian to second order and substituting the result into the Dyson series that

$$\left\|\mathcal{T}e^{-iT\int_s^{s+4/r} H_{\text{eff}}(s')ds'} - e^{-i\frac{4T}{r}H_{\text{eff}}(s+2/r)}\right\| \in \mathcal{O}\left(\max_s \frac{\|\partial_s^2 H_{\text{eff}}(s)\| T}{r^3} + \max_s \frac{\|\partial_s H_{\text{eff}}(s)\| \|H_{\text{eff}}(s)\| T^2}{r^3}\right). \tag{B22}$$

We then can bound the scaling of the error in the midpoint approximation by substituting (B18) and (B21) into (B22) and noting from (B15) that $\|H_{\text{eff}}(s)\| = (r/4)\| \log[W_r^4(s)]\| \in \mathcal{O}(1)$ to find

$$\begin{aligned}
\left\|\mathcal{T}e^{-iT\int_s^{s+4/r} H_{\text{eff}}(s')ds'} - e^{-i\frac{4T}{r}H_{\text{eff}}(s+2/r)}\right\| &= \mathcal{O}\left(\frac{\|\partial_s^2 W_r^4(s)\| T + \|\partial_s W_r^4(s)\| T^2}{r^2}\right)\\
&= \mathcal{O}\left(\frac{\max_s\left(\|\ddot{H}\| + (\|\dot{H}\| + |\dot{\lambda}|)\left(\frac{|\dot{\lambda}|}{\lambda}\right) + |\ddot{\lambda}|\right) T + \max_s\left(|\dot{\lambda}| + \|\dot{H}\|\right) T^2}{\lambda r^3}\right).
\end{aligned} \tag{B23}$$

Since errors are subadditive the error in performing a simulation from $s = 0$ to $s = 1$ is at most $\mathcal{O}(r)$ times the error given above. This results in the following bound on the scaling of the value of r that suffices to guarantee simulation error at most ϵ

$$r \in \mathcal{O}\left(\sqrt{\frac{\max_s\left[\|\ddot{H}\| + (\|\dot{H}\| + |\dot{\lambda}|)\left(\frac{|\dot{\lambda}|}{\lambda}\right) + |\ddot{\lambda}|\right] T + \max_s\left(|\dot{\lambda}| + \|\dot{H}\|\right) T^2}{\lambda\epsilon}}\right). \tag{B24}$$

The adiabatic theorem then implies that, under reasonable assumptions about the derivatives of the Hamiltonian [55] (specifically that the Hamiltonian is Gevrey class G^α for $\alpha \geq 1$), the value of T needed to achieve error ϵ, given that the minimum eigenvalue gap for the effective Hamiltonian is $\Delta_{\rm eff}$, scales at most as

$$T \in \widetilde{\mathcal{O}}\left[\frac{\max_s \|\dot{H}_{\rm eff}(s)\|}{\Delta_{\rm eff}^2 \epsilon}\right] \subseteq \widetilde{\mathcal{O}}\left[\frac{\max_s \left(\|\dot{H}(s)\| + |\dot{\lambda}|\right)}{\lambda \Delta_{\rm eff}^2 \epsilon}\right]. \tag{B25}$$

This implies that if $\lambda \in \Omega(1)$ and $\Delta_{\rm eff} \in o(1)$

$$\begin{aligned} r &\in \widetilde{\mathcal{O}}\left\{\frac{1}{\epsilon^{3/2}}\sqrt{\frac{\max_s \left[\|\ddot{H}\| + (\|\dot{H}\| + |\dot{\lambda}|)\left(\frac{|\dot{\lambda}|}{\lambda}\right) + |\ddot{\lambda}|\right] \max_s \left(|\dot{\lambda}| + \|\dot{H}\|\right)}{\Delta_{\rm eff}^2 \lambda^2} + \frac{\max_s \left(|\dot{\lambda}| + \|\dot{H}\|\right)^3}{\Delta_{\rm eff}^4 \lambda^3}}\right\} \\ &\subseteq \widetilde{\mathcal{O}}\left\{\frac{1}{\epsilon^{3/2}}\sqrt{\frac{\max_s \left(\|\ddot{H}\| + |\ddot{\lambda}|\right) \max_s \left(|\dot{\lambda}| + \|\dot{H}\|\right)}{\Delta_{\rm eff}^2 \lambda^2} + \frac{\max_s \left(|\dot{\lambda}| + \|\dot{H}\|\right)^3}{\Delta_{\rm eff}^4 \lambda^3}}\right\}. \end{aligned} \tag{B26}$$

If the Hamiltonian H is maximum rank then the spectral gap of the effective Hamiltonian is on the order of $\Delta_{\rm eff} \in \Omega[\min(\Delta, \min_k |E_k|)/\lambda]$ where Δ is the minimum spectral gap of the Hamiltonian H. The minimum over energy comes from the fact that the eigenvalues of W_r in the set $\{\pm 1, \pm i\}$ are mapped to 1, which can lead to degeneracies in the effective Hamiltonian that were absent in the original Hamiltonian. Thus the final scaling that we obtain is

$$r \in \widetilde{\mathcal{O}}\left[\frac{1}{\epsilon^{3/2}}\sqrt{\frac{\max_s \left(\|\ddot{H}\| + |\ddot{\lambda}|\right) \max_s \left(|\dot{\lambda}| + \|\dot{H}\|\right)}{\min(\Delta, \min_k |E_k|)^2} + \frac{\lambda \max_s \left(|\dot{\lambda}| + \|\dot{H}\|\right)^3}{\min(\Delta, \min_k |E_k|)^4}}\right]. \tag{B27}$$

This confirms that by taking the number of steps sufficiently large that we can force the diabatic error to become arbitrarily small. Thus we can use the walk operator in place of a Trotterized sequence for adiabatic state preparation and in turn as a heuristic that converges to the global optima given a large enough r. It should be noted, however, that the bounds used in this analysis are extremely loose and if a quantitatively correct estimate of the scaling is desired then many of the simplifications used above can be eschewed at the price of increasing the complexity of the expression.

Note that in practice, the adiabatic paths can be chosen such that the second derivative of the Hamiltonian is zero and similarly we can choose paths such that λ is constant by absorbing it into the definition of the evolution time for each infinitesimal step. However, we give the above expression for generality. Higher-order versions of this can also be derived using time-dependent Trotter-Suzuki formulas [60].

APPENDIX C: IN-PLACE BINARY TO UNARY CONVERSION

Here we present a quantum circuit (B2U^N) for converting a binary-encoded integer k ($0 \leq k < N$) into one-hot unary on N bits. Recall that the one-hot unary encoding should have k encoded as $|0\rangle^{\otimes(k-1)}|1\rangle|0\rangle^{\otimes(N-k)}$. An overview of the circuit is depicted in Fig. 15 in the special case that N is a power of 2. First we sketch a proof that the circuit is correct. Then we explain how to generalize the circuit to the case where N is not a power of 2. Finally, we count the non-Clifford gates needed to perform our binary-to-unary conversion circuit.

We give a sketch of a proof that the circuit is correct for N a power of 2. Our proof works by induction and we begin by explaining the trivial case $N = 1$. In this case, the output can only be the 1-bit, one-hot unary encoding of 0 and hence the output should be a single qubit, $|1\rangle$. The only input to the circuit is an ancilla initialized to $|0\rangle$ and so we can perform B2U^1 with a single NOT gate.

Now that we have explained the trivial case, we next explain our recursion and why it works [see Fig. 15(b)]. The idea is as follows. First, we apply $\text{B2U}^{N/2}$ to $k' := k - 2^{n-1}k_{n-1}$, where k_{n-1} is the most significant bit of k. This input is simply the last $n-1$ bits of k and the output of $\text{B2U}^{N/2}$ is $N/2$ qubits. Then, controlled on k_{n-1}, we swap bits 1 through $N/2 - 1$ (counting from zero) of the output of $\text{B2U}^{N/2}$ with $N/2 - 1$ ancilla qubits initialized to 0. Note that this step does not execute a controlled SWAP on position 0 of the one-hot unary encoding of k'. Having performed these controlled SWAPs, we next wish to erase qubit $N/2$ if $k > N/2$. We do this by performing $N/2 - 1$ CNOT

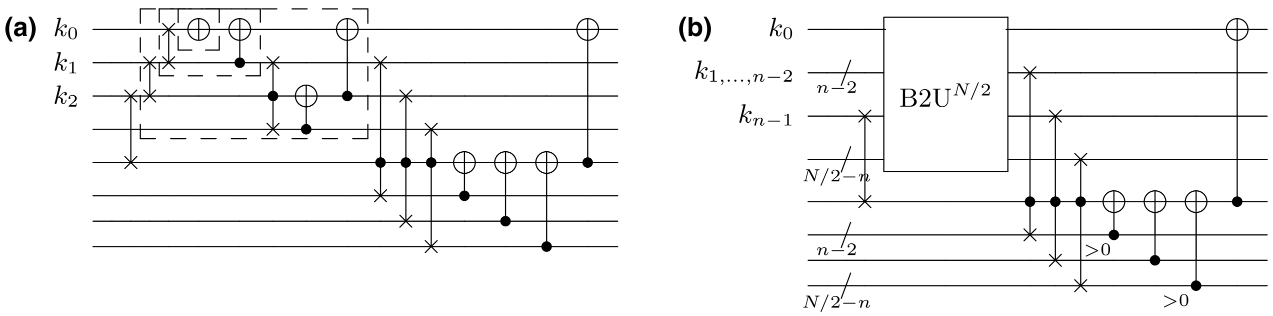

FIG. 15. A depiction of the binary-to-unary circuit mapping an n-bit binary number to an N-bit ($N = 2^n$) unary encoding of the input. In (a) we have the specific example where $N = 8$ (i.e., $\mathrm{B2U}^8$). In (b) we have the circuit ($\mathrm{B2U}^N$) defined recursively (in terms of $\mathrm{B2U}^{N/2}$), In (b) the controlled-SWAP symbols are used to represent many controlled-SWAPs, one for each qubit in the relevant registers. The symbol "> 0" signifies that the multi-CNOT is activated on any state other than the state of all zeros, which can be implemented with a cascade of CNOTs because the input is promised to have at most one nonzero qubit. The labeled rails in both circuit diagrams refer to the bits of the binary-encoded input k, and the unlabeled inputs are fresh ancillae.

gates targeted on the qubit at position $N/2$ and controlled by each of the qubits at positions above $N/2$. Finally, we have to resolve the special cases where k is $N/2$ or 0. We do this with one more CNOT, with qubit $N/2$ as the control and qubit 0 as the target.

Having given an explanation of our recursive construction, we next explain how to prove that the recursion works. We consider three distinct cases.

1. If $k < N/2$, we have $k_{n-1} = 0$ and hence none of the controlled SWAPs or CNOTs do anything. This is correct behavior because the one-hot unary encoding of k is the one-hot unary encoding of k' with $N/2$ ancilla qubits appended to it.
2. If $k = N/2$, the controlled SWAPs again do nothing but this is now because they are swapping pairs of identical qubits in the $|0\rangle$ state. The CNOTs targeted on the qubit at position $N/2$ also do nothing because the control qubits are 0. The final CNOT then erases the 1 encoded in position 0 of the output of $\mathrm{B2U}^{N/2}$, which is there because the input was $k' = 0$.
3. If $k > N/2$, the controlled SWAPs swaps the one-hot unary encoding of k' into the final $N/2$ qubits of the output register. The CNOTs targeted on qubit $N/2$ then erase that qubit, leaving the correct unary encoding of k. The final CNOT does nothing, as the control qubit was erased.

The proof sketch demonstrates that our recursive binary-to-unary circuit works when N is a power of two. Next we explain how to modify the circuit when N is not a power of 2. If N is not a power of 2, define $n := \lceil \log N \rceil$ and $N' = 2^n$. Apply $\mathrm{B2U}^{N'/2}$ to the least significant $n - 1$ bits of k. Then perform the controlled SWAPS and CNOTS involving the remaining $N - N'/2$ ancilla qubits, removing any operations that involve deleted qubits. For $N = 7$, for example, we delete the bottom rail from Fig. 15(a) as well as the controlled SWAP and the CNOT involving that final rail. To see that this works, observe that the circuit also works if we performed $\mathrm{B2U}^{N'}$ and then remove the final $N' - N$ qubits, which are guaranteed to be zero. Our construction simply eliminates unnecessary gates from $\mathrm{B2U}^{N'}$.

Our final task is to count the number of non-Clifford gates needed by our $\mathrm{B2U}^N$ circuit. The only non-Clifford gates are the controlled-SWAP operations, which can be executed with a single Toffoli gate and two CNOTs. We prove that the number of controlled-SWAP gates is

$$C_N := N - \lceil \log N \rceil - 1. \tag{C1}$$

First, it is clear that $C_1 = 0$ as required. Next, it is clear from Fig. 15(b) that $C_{N'} = N'/2 - 1 + C_{N'/2}$. Based on our analysis above, $C_N = C_{N'} - (N' - N)$ and hence

$$C_N = N - N'/2 - 1 + C_{N'/2}. \tag{C2}$$

Next assume Eq. (C1) is true for some particular value $N'/2$. Then by substitution in Eq. (C2),

$$\begin{aligned} C_N &= N - N'/2 - 1 + N'/2 - \lceil \log N'/2 \rceil - 1 \\ &= N - n - 1 = N - \lceil \log N \rceil - 1 \end{aligned} \tag{C3}$$

thus satisfying Eq. (C1) for N as required. Therefore, by induction Eq. (C1) is correct for all N.

APPENDIX D: COST OF MULTIPLICATION

As the multiplication operation is a major contributor to the overall complexity of our algorithms, we need to be quite careful in our analysis of the operation. We also frequently require only low-precision arithmetic, meaning that we can make our multiplications less accurate and therefore computationally cheaper. This appendix presents our algorithms for performing four variations of the multiplication task, with modifications to be used when one of the inputs is given classically rather than quantumly.

Our strategy is to use schoolbook multiplication. In schoolbook multiplication, the product $\gamma := \kappa \times \lambda$ is calculated by writing $\kappa = \sum_\ell 2^\ell \kappa_\ell$ with $\kappa_\ell \in \{0,1\}$ and then calculating the sum $\gamma = \sum_\ell 2^\ell \kappa_\ell \lambda$. This reduces the task of multiplication to two very simple multiplications and the task of adding a list of numbers. The two multiplications are simple because multiplication by a power of 2 can be accomplished by an appropriate bit-shift operation, and multiplying by a single bit can be accomplished by using that bit as a control for the addition operation. That is, we perform that part of the addition if and only if the control bit is one.

We begin in Appendix 1 by reviewing the parts of the main text where we need to multiply two numbers together. In Appendix D 2 we explain how to add a constant value to a quantum register, which is used separately in Algorithm 1 but is also used through the rest of this appendix in order to multiply a quantum variable to a classical constant. We then explain the simplest variant of multiplication in Appendix 3, where we must multiply two integers together. We then explain the remaining variants by modifying the integer-integer multiplication algorithm as appropriate. In Appendix 4, we explain the case where we multiply an integer to a real number. In Appendix 5, we explain the case where we multiply a real number to another real number. Finally, in Appendix 6, we explain the case where we calculate the square of a given real number. In all cases we indicate how the algorithm is to be modified when one of the inputs is classically specified.

1. Uses of multiplication in this paper

In Sec. III C we need to multiply a quantum register by a classical constant $\tilde{\gamma}$ or γ to obtain the phase to apply. The multiplication is performed directly into the phase-gradient state, so we cannot use the savings where the multiplication result is placed in an initially zero register. The fastest method seems to be to write the classical constant as a sum of powers of 2 with plus and minus signs.

In Sec. II E we consider QROM for interpolation of functions, and we need to multiply the input register by the slope. In that case, both registers are quantum. The input register is given to $b_{\rm dif}$ bits, and the goal is to give the approximation to the function to $b_{\rm sm}$ bits. This may require giving the slope to $b_{\rm sm} + \mathcal{O}(\log b_{\rm sm})$ bits, or $b_{\rm sm} + b_{\rm fun} + \mathcal{O}(\log b_{\rm sm})$ bits in the case of the arcsine. For Szegedy walks, we need to take the square of a quantum register, and need to multiply a quantum register by a constant.

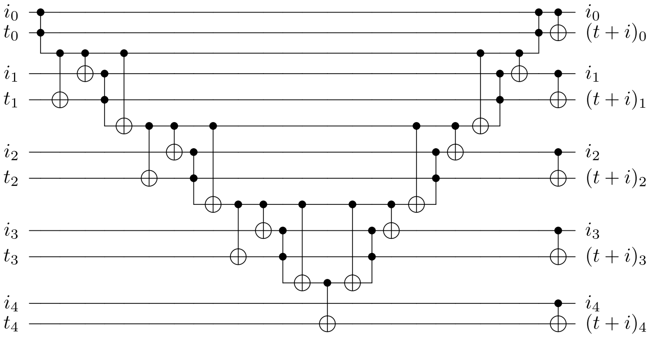

FIG. 16. A circuit to perform addition on 5 qubits modulo 2^5 from Ref. [35].

2. Methods for addition

When adding a classically given constant to a quantum register, it is possible to save the qubits that are used to store this classical constant. Consider the quantum circuit for addition following Ref. [35], as shown in Fig. 16 where i is the classically given integer and t is the quantum register. For this diagram we use the convention of Ref. [35] where a Toffoli with a target known to be initially zeroed is shown with a ⌞ for the target. That is the first operation on the left in Fig. 16. The Toffolis with targets that are known to be zero afterwards are shown with ⌟ for the target. These may be performed with measurements and Cliffords so do not add to the non-Clifford cost.

The circuit for the adder contains a subsection where a CNOT gate is performed on a qubit of i, say i_1, as shown in Fig. 17(a). The state after the CNOT can alternatively be obtained on the control by switching the control and target for the CNOT. Then for the following Toffoli where i_1 is the control, we switch the control to the carry register at the top. After that the carry register needs to be used as a control where it should take its original value, so we need another CNOT to undo the first. The resulting section of the circuit is as shown in Fig. 17(b). Replacing all these sections of the circuit in this way, we obtain an addition circuit as shown in Fig. 18. This adder only uses the i_j registers as controls. Since these registers have classically known values, all controls by these qubits may be replaced with classical controls, and these qubits need not be used. This also reduces the Toffoli cost by 1, because the first Toffoli is replaced with a CNOT. The Toffoli cost is therefore the number of bits minus 2. The number of ancillas needed is the number of bits minus 1.

3. Multiplying two integers

In this variant of the multiplication task, we are to multiply the d_A-bit integer κ to the d_B-bit integer λ. These integers are encoded into quantum registers $\mathtt{A}$ and $\mathtt{B}$, respectively. Thus our task is to prepare a $(d_A + d_B)$-qubit

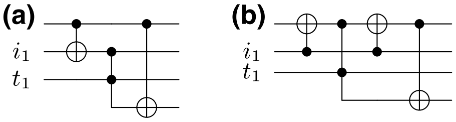

FIG. 17. (a) The component of the adder circuit where the qubit containing classical data is the target of a CNOT. (b) The circuit may be rewritten so the value on the second qubit is never changed.

register `out` as follows:

$$|\kappa\rangle_{\mathtt{A}}\,|\lambda\rangle_{\mathtt{B}}\,|0\rangle_{\mathtt{out}} \mapsto |\kappa\rangle_{\mathtt{A}}\,|\lambda\rangle_{\mathtt{B}}\,|\gamma := \kappa \times \lambda\rangle_{\mathtt{out}}\,. \tag{D1}$$

We explain how to perform this multiplication using schoolbook multiplication and the Gidney adder [35], and we explain how to reduce the computational cost if one of the inputs is presented to us classically rather than quantumly.

We now explain the schoolbook multiplication algorithm in some detail. Let the bits of κ and λ be denoted as follows:

$$\kappa := \sum_{\ell=1}^{d_A} 2^{d_A-\ell}\kappa_\ell; \quad \lambda := \sum_{\ell=1}^{d_B} 2^{d_B-\ell}\lambda_\ell; \quad \kappa_\ell, \lambda_\ell \in \{0,1\}. \tag{D2}$$

Thus λ_{d_B} refers to the least-significant bit of λ. Our procedure is then as follows.

1. Controlled on the final qubit of `B`, copy all the qubits of `A` into the final d_A bits of `out`.
   Result: $|0\rangle_{\mathtt{out}} \mapsto |\lambda_{d_B}\kappa\rangle_{\mathtt{out}}$.
   Cost: d_A Toffolis.
2. For each $\ell = d_B - 1, \ldots, 1$, add 2^ℓ times the value of `A` to `out` in place, controlled on the $(d_B - \ell)$th qubit of `B`. This can be done by using the control to copy the d_A bits to an ancilla, and adding this ancilla. The ancilla can be erased with no Toffoli cost. We add `A` to `out` with the final ℓ qubits of `out` ignored. Note that the number of nonzero bits is always no greater than d_A.
   Result: $|\xi\rangle_{\mathtt{out}} \mapsto |\xi + 2^\ell \lambda_{d_B-\ell}\kappa\rangle_{\mathtt{out}}$, where ξ is the integer encoded in `out` before this step.
   Cost: $2d_A$ Toffolis.

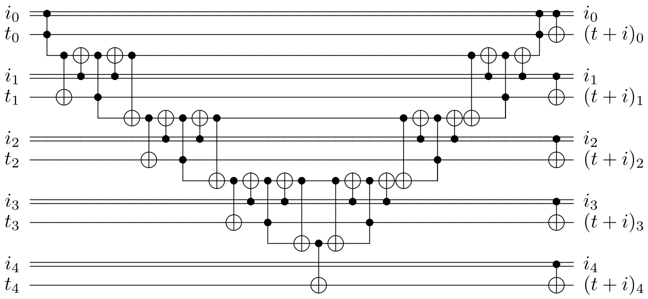

FIG. 18. A circuit to perform addition on 5 qubits modulo 2^5 such that the i_j registers are only used as controls. Because they are only used as controls, if the number i is given classically the addition can be performed entirely using classical control, without using any ancillas to store i.

The total number of Toffolis is $2d_Ad_B - d_A$, and the total number of temporary ancilla qubits needed is $2d_A - 1$ since we are copying d_A qubits out to an ancilla as well as using $d_A - 1$ temporary qubits in the addition.

We now consider how the cost of the algorithm can be reduced when one of the inputs is presented classically, rather than quantumly. The effect on the algorithm is different depending on whether κ or λ is known classically. In the case that κ is known classically, we can replace all the Toffolis in the copy operation in step 1 with CNOTs or identity gates, depending on whether the relevant bit of κ is 1 or 0. More interestingly, each addition step involves adding a known constant rather than an unknown variable to be read from a quantum register during computation. The effect on computational complexity depends on the classical constant κ; in particular, on the largest power of 2 that divides κ. In the worst case ($\kappa \mod 2 = 1$), we save one Toffoli per addition step. In the best case ($\kappa = 0$), we have zero computational cost because we are multiplying by zero and we know we are multiplying by zero.

In the case that λ is presented to us classically rather than quantumly, we can make the addition controlled by performing the (noncontrolled) addition circuit in the case of 1, or doing nothing when the classical control is 0. The number of quantum-to-quantum additions thus depends on the number of nonzero classical bits—the greater the Hamming weight of the classical input, the greater the number of additions to be performed. Note that this is distinct from the cost of performing classical-to-quantum multiplication when κ is the classical variable, in which case the complexity is determined by the number of zeros on the far right of the number.

The case where λ is given classically is more relevant for this paper. We are unlikely to be dealing with classically specified integers that are a multiple of a large power of 2. On the other hand, we frequently have some information about the Hamming weight $\sum_\ell \lambda_\ell$ of the classically known number λ. Each addition costs at most d_A Toffolis, we perform $\sum_\ell \lambda_\ell \leq d_B$ such additions, and no other operations require Toffoli gates. We therefore have a total Toffoli cost of at most d_Ad_B, which can be replaced with $d_A \sum_\ell \lambda_\ell$ if we can assume knowledge of the Hamming weight of λ. Thus we save a factor of 2 if one of the inputs is classical.

In the following subsections, we explain how to modify the above procedure for variants of the multiplication task. These variants have at least one of the inputs being a real number between 0 and 1, rather than an integer. Thus the task is not to calculate the multiplication exactly, as this involves infinitely many bits for real numbers. Instead, we truncate the binary expansions of real numbers to ensure that an error threshold is achieved.

4. Multiplying an integer to a real number

Now we consider a variant of the multiplication task where one of the inputs is a real number between zero and one. For reasons that become clear below, we specify κ to be the real number and λ to be the integer. We assume that the real number is defined to infinitely many digits and that our task is to approximate $\gamma := \kappa \times \lambda$ to within an error tolerance ε. Thus our task is to calculate some $\tilde{\gamma}$ such that $|\gamma - \tilde{\gamma}| < \varepsilon$. That is to say, we are to prepare a new quantum register out as follows:

$$|\kappa\rangle_{\mathtt{A}} |\lambda\rangle_{\mathtt{B}} |0\rangle_{\mathtt{out}} \mapsto |\kappa\rangle_{\mathtt{A}} |\lambda\rangle_{\mathtt{B}} |\tilde{\gamma}\rangle_{\mathtt{out}} . \tag{D3}$$

Here we are free to choose the number of bits for the register A and hence the number of bits for the register out. This choice naturally depends on the error tolerance ε.

We begin by specifying symbols for the bits of the inputs κ and λ. Note that the indexing differs somewhat from the previous section. We define

$$\kappa := \sum_{\ell=1}^{\infty} \kappa_\ell / 2^\ell; \quad \lambda := \sum_{\ell=1}^{d_B} 2^{d_B-\ell} \lambda_\ell; \quad \kappa_\ell, \lambda_\ell \in \{0,1\}. \tag{D4}$$

We then select an integer $d_A \geq d_B$ (presuming $\varepsilon < 1$) that counts the number of bits of the input κ we plan to use. We use $d_A - 1$ bits of κ. We explain our plan by first representing the ideal product as

$$\gamma = \left(\begin{array}{ccccccccccccc|cc}
 & \lambda_1 & \times & \kappa_1 & \kappa_2 & \kappa_3 & \cdots & \kappa_{d_B-2} & \kappa_{d_B-1} & . & \kappa_{d_B} & \cdots & \kappa_{d_A-1} & \kappa_{d_A} & \cdots \\
+ & \lambda_2 & \times & & \kappa_1 & \kappa_2 & \cdots & \kappa_{d_B-3} & \kappa_{d_B-2} & . & \kappa_{d_B-1} & \cdots & \kappa_{d_A-2} & \kappa_{d_A-1} & \cdots \\
+ & \lambda_3 & \times & & & \kappa_1 & \cdots & \kappa_{d_B-4} & \kappa_{d_B-3} & . & \kappa_{d_B-2} & \cdots & \kappa_{d_A-3} & \kappa_{d_A-2} & \cdots \\
 & & \vdots & & & & & & & & & & & & \\
+ & \lambda_{d_B-1} & \times & & & & & & \kappa_1 & . & \kappa_2 & \cdots & \kappa_{d_A-d_B+1} & \kappa_{d_A-d_B+2} & \cdots \\
+ & \lambda_{d_B} & \times & & & & & & 0 & . & \kappa_1 & \cdots & \kappa_{d_A-d_B} & \kappa_{d_A-d_B+1} & \cdots
\end{array}\right), \tag{D5}$$

where the vertical line denotes where we truncate the binary expansion of κ. We thus calculate

$$\tilde{\gamma}_{d_A} := \sum_{\ell=1}^{d_B} \lambda_\ell \lfloor \kappa 2^{d_A-\ell} \rfloor 2^{d_B-d_A}; \quad \gamma - \tilde{\gamma}_{d_A} \leq d_B 2^{d_B-d_A}. \tag{D6}$$

To ensure that the error tolerance ε is achieved, we should choose $d_A > d_B + \log(d_B/\varepsilon)$. We therefore choose

$$d_A = d_B + \lceil \log(d_B/\varepsilon) \rceil. \tag{D7}$$

We follow a similar strategy to that described in Appendix 3, meaning that we are to perform a sequence of controlled additions. We work bottom to top in Eq. (D5).

We start with the bottom line by copying $d_A - d_B$ bits into the output register, with Toffoli cost $d_A - d_B$. After that the number of Toffolis is twice the number of bits. The total number of Toffolis is then

$$\begin{aligned}
&(d_A - d_B) + 2(d_A - d_B + 1) + \cdots + 2(d_A - 1) \\
&\quad = d_A - d_B + \sum_{\ell=d_A-d_B+1}^{d_A-1} \ell \\
&= d_A(2d_B - 1) - {d_B}^2 \\
&= [d_B + \lceil \log(d_B/\varepsilon) \rceil](2d_B - 1) - {d_B}^2 \\
&= {d_B}^2 + (2d_B - 1)\lceil \log(d_B/\varepsilon) \rceil - d_B.
\end{aligned} \tag{D8}$$

Hence the Toffoli cost of multiplying a real number to an integer on a quantum computer is no more than

$${d_B}^2 + (2d_B - 1)\lceil \log(d_B/\varepsilon) \rceil - d_B, \tag{D9}$$

where d_B is the number of bits used to specify the integer and ε is the allowable error in the overall multiplication. The algorithm requires that the real number is specified to $d_A = \lceil d_B \log(d_B/\varepsilon) \rceil$ bits and uses $d_A - 1$ ancilla qubits.

5. Multiplying two different real numbers

In this subsection we consider the task where we are to multiply two real numbers κ $(0 \leq \kappa < 1)$ and λ

$(0 \leq \kappa < 1)$. Our task is to calculate an approximation $\tilde{\gamma}$ to $\gamma := \kappa \times \lambda$ such that $|\gamma - \tilde{\gamma}| < \varepsilon$, where $\varepsilon > 0$ is some given error tolerance. That is to say, we are to prepare a new quantum register `out` as follows:

$$|\kappa\rangle_{\mathtt{A}} |\lambda\rangle_{\mathtt{B}} |0\rangle_{\mathtt{out}} \mapsto |\kappa\rangle_{\mathtt{A}} |\lambda\rangle_{\mathtt{B}} |\tilde{\gamma}\rangle_{\mathtt{out}}. \tag{D10}$$

We are free to choose the number of qubits in each of the registers `A`, `B`, and `out` to ensure that the output encodes a value for $\tilde{\gamma}$ that approximates γ to within the error tolerance ε. We begin by discussing these choices of register size, starting with the size of `A` and `B`.

To explain our choice for the numbers of qubits for registers `A` and `B`, we begin by introducing notation for the inputs κ and λ. As before, we define the bits of the inputs according to the equations

$$\kappa := \sum_{\ell=1}^{\infty} \kappa_\ell / 2^\ell; \quad \lambda := \sum_{\ell=1}^{\infty} \lambda_\ell / 2^\ell; \quad \kappa_\ell, \lambda_\ell \in \{0, 1\}. \tag{D11}$$

This suggests our strategy for calculating γ. As before, we have

$$\gamma = \left(\begin{array}{ccccccccccccc|cc} & \lambda_1 & \times & .0 & \kappa_1 & \kappa_2 & \kappa_3 & \cdots & \kappa_{d_B-2} & \kappa_{d_B-1} & \kappa_{d_B} & \cdots & \kappa_{d_A-1} & \kappa_{d_A} & \cdots \\ + & \lambda_2 & \times & .0 & 0 & \kappa_1 & \kappa_2 & \cdots & \kappa_{d_B-3} & \kappa_{d_B-2} & \kappa_{d_B-1} & \cdots & \kappa_{d_A-2} & \kappa_{d_A-1} & \cdots \\ + & \lambda_3 & \times & .0 & 0 & 0 & \kappa_1 & \cdots & \kappa_{d_B-4} & \kappa_{d_B-3} & \kappa_{d_B-2} & \cdots & \kappa_{d_A-3} & \kappa_{d_A-2} & \cdots \\ & & \vdots & & & & & & & & & & & & \\ + & \lambda_{d_B-1} & \times & .0 & 0 & 0 & 0 & \cdots & 0 & \kappa_1 & \kappa_2 & \cdots & \kappa_{d_A-d_B+1} & \kappa_{d_A-d_B} & \cdots \\ + & \lambda_{d_B} & \times & .0 & 0 & 0 & 0 & \cdots & 0 & 0 & \kappa_1 & \cdots & \kappa_{d_A-d_B} & \kappa_{d_A-d_B-1} & \cdots \\ & & \vdots & & & & & & & & & & & & \end{array} \right), \tag{D12}$$

where solid lines indicate where we truncate the calculation in order to produce the approximation $\tilde{\gamma}$ instead of γ. Here we assume that $d_A \geq d_B$; if $d_A < d_B$, our repeated addition procedure involves several additions by zero. The repeated addition strategy has a Toffoli cost of

$$\begin{aligned} (d_A - d_B + 1) + 2(d_A - d_B + 2) + \cdots + 2(d_A - 1) &= (d_A - d_B + 1) + 2 \sum_{\ell = d_A - d_B + 2}^{d_A - 1} \ell \\ &= d_A(2d_B - 3) - (d_B - 1)^2. \end{aligned} \tag{D13}$$

It seems reasonable to set $d_A = d_B$, and numerical evidence indicates that this choice makes the optimal trade-off between computational complexity and error tolerance. Setting $d := d_A = d_B$, the Toffoli cost is simply $d^2 - d - 1$.

We now consider the error of the sum in Eq. (D12). There

$$\tilde{\gamma} = \sum_{\substack{n,m=1 \\ n+m \leq d}}^{\infty} \kappa_n \lambda_m 2^{-(n+m)}, \tag{D14}$$

so

$$\gamma - \tilde{\gamma} = \sum_{\substack{n,m=1 \\ n+m > d}}^{\infty} \kappa_n \lambda_m 2^{-(n+m)} \leq \sum_{\substack{n,m=1 \\ n+m > d}}^{\infty} 2^{-(n+m)} = \frac{d+1}{2^d}. \tag{D15}$$

We can ensure that the error of the approximation $\tilde{\gamma}$ is within tolerance ε by setting

$$\frac{d+1}{2d} \leq \varepsilon. \tag{D16}$$

Though the above could be solved exactly using a Lambert-W function, it is satisfied with

$$d = 1 + \log(1/\varepsilon) + \log[1 + \log(1/\varepsilon)]. \tag{D17}$$

Figure 19 justifies this choice by depicting the value of $(d+1)/2d\varepsilon$ as a function of ε with d chosen as per Eq. (D17). To choose d, we take the ceiling of this expression, because d must be chosen to be an integer.

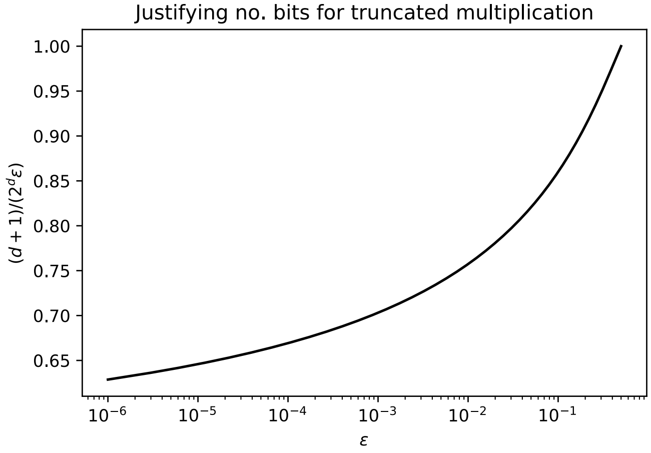

FIG. 19. Numerical justification for our choice of d in Eq. (D17). We plot the ratio of $(d+1)/2d$ to the error bound ε. A value less than 1 ensures that our choice of d yields a multiplication result whose error is less than ε.

Hence our strategy for multiplying two real numbers uses

$$d^2 - d - 1 = \log^2(1/\varepsilon) + 2\log(1/\varepsilon)\log\log(1/\varepsilon) + \mathcal{O}(\log(1/\varepsilon)) \tag{D18}$$

Toffoli gates to achieve an output with error less than ε. The ancilla cost is $d-1$ bits for a copy of the bits of κ for the controlled addition, and another $d-1$ bits for the addition itself. Thus the ancilla cost is

$$2\log(1/\varepsilon) + \mathcal{O}(\log\log(1/\varepsilon)). \tag{D19}$$

6. Squaring a real number

We are given a quantum register A with a real number κ that satisfies $0 \le \kappa < 1$. Our task is to calculate an approximation $\tilde{\gamma}$ of $\gamma := \kappa^2$ such that $|\gamma - \tilde{\gamma}| < \varepsilon$, where ε is given $(0 < \varepsilon < 1)$. That is to say, we are to prepare a new quantum register out as follows:

$$|\kappa\rangle_{\mathtt{A}}|0\rangle_{\mathtt{out}} \mapsto |\kappa\rangle_{\mathtt{A}}|\tilde{\gamma}\rangle_{\mathtt{out}}. \tag{D20}$$

We include d bits in the sum, so the sum can be expressed as

$$\tilde{\gamma} = \sum_{\substack{n,m=1\\ n+m\le d}}^{\infty} \kappa_n\kappa_m 2^{-(n+m)}. \tag{D21}$$

We take advantage of symmetry to rewrite the sum as

$$\begin{aligned}\tilde{\gamma} &= 2\sum_{\substack{n,m=1\\ n+m\le d, n>m}}^{\infty} \kappa_n\kappa_m 2^{-(n+m)} + \sum_{n=1}^{\lfloor d/2\rfloor} \kappa_n 2^{-2n}\\ &= 2\sum_{n=1}^{d-1}\kappa_n \sum_{m=1}^{\min(n-1,d-n)} \kappa_m 2^{-(n+m)} + \sum_{n=1}^{\lfloor d/2\rfloor}\kappa_n 2^{-2n}.\end{aligned} \tag{D22}$$

The first term in this sum contains the parts where $n > m$, and is multiplied by 2 because those parts with $n < m$ are equal by symmetry. The second term is that for $n = m$. This sum is more efficient, because only about half as many terms appear. Now the term in the second sum for $n = \lfloor d/2\rfloor$ is half the size of any of the other terms, so it is convenient to omit it.

The form of the sum can be shown, for the odd example $d = 15$,

$$\tilde{\gamma} = \left(\begin{array}{ccccccccccccccccc}
 & \kappa_1 & \times & .0 & \kappa_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
+ & \kappa_2 & \times & .0 & \kappa_1 & 0 & \kappa_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
+ & \kappa_3 & \times & .0 & 0 & \kappa_1 & \kappa_2 & 0 & \kappa_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
+ & \kappa_4 & \times & .0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & 0 & \kappa_4 & 0 & 0 & 0 & 0 & 0 & 0\\
+ & \kappa_5 & \times & .0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & 0 & \kappa_5 & 0 & 0 & 0 & 0\\
+ & \kappa_6 & \times & .0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5 & 0 & \kappa_6 & 0 & 0\\
+ & \kappa_7 & \times & .0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5 & \kappa_6 & 0 & \kappa_7\\
+ & \kappa_8 & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5 & \kappa_6 & \kappa_7\\
+ & \kappa_9 & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5 & \kappa_6\\
+ & \kappa_{10} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5\\
+ & \kappa_{11} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4\\
+ & \kappa_{12} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3\\
+ & \kappa_{13} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2\\
+ & \kappa_{14} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1
\end{array}\right). \tag{D23}$$

Here we show terms from the second sum from Eq. (D22) in blue. In the case where d is odd, $\lfloor d/2 \rfloor = (d-1)/2$, and we can write the sum as

$$\tilde{\gamma} = \sum_{n=1}^{(d-1)/2} \kappa_n \left[2^{-2n} + 2 \sum_{m=1}^{n-1} \kappa_n \kappa_m 2^{-(n+m)} \right] + 2 \sum_{n=(d+1)/2}^{d-1} \kappa_n \sum_{m=1}^{d-n} \kappa_m 2^{-(n+m)}. \tag{D24}$$

The first sum in Eq. (D24) corresponds to the part above the horizontal line in Eq. (D23), and the second sum in Eq. (D24) corresponds to the part below the line. To compute Eq. (D24), we start at the least significant digit, and move to the most significant digit [corresponding to moving from the bottom row to the top row in Eq. (D29)], as

$$\tilde{\gamma} = \sum_{n=d-1}^{(d+1)/2} \kappa_n \sum_{m=1}^{d-n} \kappa_m 2^{-(n+m-1)} + \sum_{n=(d-1)/2}^{1} \kappa_n \left[2^{-2n} + \sum_{m=1}^{n-1} \kappa_n \kappa_m 2^{-(n+m-1)} \right]. \tag{D25}$$

To compute the sum we start with $n = d-1$, and copy the value $\kappa_{d-1}\kappa_1$ into the output at position $d-1$ [to initialize the output as $\kappa_{d-1}\kappa_1 2^{-(d-1)}$] with Toffoli cost 1. Next, we use κ_{d-2} to control addition of $\kappa_1 2^{-(d-2)} + \kappa_2 2^{-(d-1)}$ into the output. This controlled addition has cost 2×2 because it is for 2 bits. At step $j = d-n \leq (d-1)/2$, the cost of controlled addition of j bits is $2j$. The cost of that part is therefore

$$1 + \sum_{j=2}^{(d-1)/2} 2j = (d^2 - 5)/4. \tag{D26}$$

For the remaining steps with $n = (d-1)/2$ to 2, we have a cost of $n-1$ to produce the $n-1$ values of $\kappa_n\kappa_m$, and there are $n+1$ bits that need to be added into the output. That includes the bit for $\kappa_n 2^{-2n}$, which is spaced by one bit from the remaining bits for $\kappa_n\kappa_m$. That gives cost $(n-1)+(n+1) = 2n$. For $n = 1$, we just have a cost of one Toffoli to add in the single bit (without a control, because it is just κ_n). That gives the same cost as the first half, for a total cost of

$$d^2/2 - 5/2. \tag{D27}$$

In the case where d is even, $\lfloor d/2 \rfloor = d/2$, and we can write the sum as

$$\tilde{\gamma} = \sum_{n=1}^{d/2-1} \kappa_n \left[2^{-2n} + 2 \sum_{m=1}^{n-1} \kappa_n \kappa_m 2^{-(n+m)} \right] + 2\kappa_{d/2} \sum_{m=1}^{d/2-1} \kappa_m 2^{-(d/2+m)} + 2 \sum_{n=d/2+1}^{d} \kappa_n \sum_{m=1}^{d+1-n} \kappa_m 2^{-(n+m)}. \tag{D28}$$

The form of the sum for an even example $d = 16$ is

$$\tilde{\gamma} = \left(\begin{array}{cccccccccccccccccc}
 & \kappa_1 & \times & .0 & \kappa_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
+ & \kappa_2 & \times & .0 & \kappa_1 & 0 & \kappa_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
+ & \kappa_3 & \times & .0 & 0 & \kappa_1 & \kappa_2 & 0 & \kappa_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
+ & \kappa_4 & \times & .0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & 0 & \kappa_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
+ & \kappa_5 & \times & .0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & 0 & \kappa_5 & 0 & 0 & 0 & 0 & 0 \\
+ & \kappa_6 & \times & .0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5 & 0 & \kappa_6 & 0 & 0 & 0 \\
+ & \kappa_7 & \times & .0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5 & \kappa_6 & 0 & \kappa_7 & 0 \\
\hline
+ & \kappa_8 & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5 & \kappa_6 & \kappa_7 & 0 \\
\hline
+ & \kappa_9 & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5 & \kappa_6 & \kappa_7 \\
+ & \kappa_{10} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5 & \kappa_6 \\
+ & \kappa_{11} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 & \kappa_5 \\
+ & \kappa_{12} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 & \kappa_4 \\
+ & \kappa_{13} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 & \kappa_3 \\
+ & \kappa_{14} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1 & \kappa_2 \\
+ & \kappa_{15} & \times & .0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \kappa_1
\end{array} \right). \tag{D29}$$

Again we have shown terms from the second sum from Eq. (D22) in blue. The first sum in Eq. (D28) corresponds to the part above the first horizontal line in Eq. (D29), the second sum in Eq. (D28) corresponds to the part between the two lines in Eq. (D29), and the third sum in Eq. (D28) corresponds to the part below the second horizontal line in Eq. (D29). To compute Eq. (D28), we again start at the least significant digit, and move to the most significant digit, as

$$\tilde{\gamma} = \sum_{n=d}^{d/2+1} \kappa_n \sum_{m=1}^{d+1-n} \kappa_m 2^{-(n+m-1)} + \kappa_{d/2} \sum_{m=1}^{d/2-1} \kappa_m 2^{-(d/2+m-1)} + \sum_{n=d/2-1}^{1} \kappa_n \left[2^{-2n} + \sum_{m=1}^{n-1} \kappa_n \kappa_m 2^{-(n+m-1)} \right]. \tag{D30}$$

For the costing of the additions, we have the same costing for the first sum in Eq. (D30) as in the odd case, giving cost

$$1 + \sum_{j=2}^{d/2-1} 2j = (d^2 - 2d - 4)/4. \tag{D31}$$

The middle sum in Eq. (D30) has cost $d-2$, then the final sum has cost $2n$ for $n = 2$ to $d/2 - 1$, and cost 1 for $n = 1$, giving the same cost as the first sum. That gives a total complexity

$$d^2/2 - 4. \tag{D32}$$

Thus in both the odd and even cases the complexity is less than $d^2/2$.

To estimate the error, we have

$$\begin{aligned} \gamma - \tilde{\gamma} &= 2 \sum_{\substack{n,m=1 \\ n+m>d, n>m}}^{\infty} \kappa_n \kappa_m 2^{-(n+m)} + \sum_{n=\lfloor d/2 \rfloor}^{\infty} \kappa_n 2^{-2n} \\ &\le 2 \sum_{\substack{n,m=1 \\ n+m>d, n>m}}^{\infty} 2^{-(n+m)} + \sum_{n=\lfloor d/2 \rfloor}^{\infty} 2^{-2n} \\ &= \sum_{\ell=d+1}^{\infty} \lfloor (\ell-1)/2 \rfloor 2^{-\ell} + \frac{4}{3} 2^{-2\lfloor d/2 \rfloor}. \end{aligned} \tag{D33}$$

In the case of even d we get

$$\frac{1}{2} d 2^{-d} + \frac{1}{3} 2^{-d} + \frac{4}{3} 2^{-d} = \frac{1}{2} d 2^{-d} + \frac{5}{3} 2^{-d}, \tag{D34}$$

and in the case of odd d we get

$$\frac{1}{2} d 2^{-d} + \frac{1}{6} 2^{-d} + \frac{8}{3} 2^{-d} = \frac{1}{2} d 2^{-d} + \frac{17}{6} 2^{-d}. \tag{D35}$$

We find that we can limit the error to ϵ using

$$d = \lceil \log(1/\epsilon) + \log[11/3 + \log(1/\epsilon)] \rceil. \tag{D36}$$

The Toffoli cost of squaring is then (regardless of whether d is odd or even)

$$\begin{aligned} d^2/2 = \frac{1}{2} \log^2(1/\varepsilon) + \log(1/\varepsilon) \log\log(1/\varepsilon) \\ + \mathcal{O}(\log(1/\varepsilon)). \end{aligned} \tag{D37}$$

The ancilla cost in the case where d is even has a maximum of $d - 2$. When $n = d/2$ [corresponding to the part between the two horizontal lines in Eq. (D29)], there are $d/2 - 1$ bits to add in a controlled way, so there are $d/2 - 1$ bits for the copy and another $d/2 - 1$ bits for the addition itself. When $n = d/2 - 1$, there are $d/2 - 2$ bits to add in a controlled way, and a range of $d/2$ bits to add in which give an ancilla cost of $d/2$. In both these cases, the ancilla cost is $d - 2$. When d is odd, the ancilla cost is $d - 1$. When $n = \lceil d/2 \rceil$ [the part just below the horizontal line in Eq. (D23)], there are $\lceil d/2 \rceil - 1$ bits to add in a controlled way, which takes $2(\lceil d/2 \rceil - 1) = d - 1$ ancillas for d odd. When $n = \lceil d/2 \rceil - 1$, there are $\lceil d/2 \rceil - 2$ bits to add in a controlled way, and a range of $\lceil d/2 \rceil$ bits to add, for a total ancilla cost of $d - 1$. Hence the ancilla cost of squaring is only half that for multiplication, and is

$$\log(1/\varepsilon) + \mathcal{O}(\log\log(1/\varepsilon)). \tag{D38}$$

APPENDIX E: OTHER APPROACHES TO HAMILTONIAN EVOLUTION-BASED OPTIMIZATION

Here we outline two other approaches in the literature to optimization based on Hamiltonian evolution. We consider "shortest-path" optimization in Appendix 1 and we consider quantum-enhanced population transfer in Appendix 2. In both cases, we review the techniques and explain how the algorithmic primitives we develop in this paper could be applied in each approach.

1. Heuristic variant of the shortest-path algorithm

Hastings' "shortest-path algorithm" [9] (SPA) is an interesting approach to quantum optimization that is also based on time evolution under a cost function with some noncommuting driver Hamiltonian. Perhaps the most intriguing property of the SPA is that Hastings was able to rigorously show that the SPA gives a super-Grover (i.e., better than quadratic) speedup for certain classical optimization problems—e.g., for an arbitrary instance of the problem MAX-2-LIN2 (which is a problem very closely related to QUBO) [78]. The results of Refs. [9] and [78] also rigorously (and in some cases, empirically) show similar speedups under a variety of assumptions about related problems.

The SPA essentially involves applying amplitude amplification to a variant of the adiabatic algorithm, which uses the time-dependent Hamiltonian

$$H(s) = C + sB\left(\frac{\sum_p X_p}{N}\right)^K, \tag{E1}$$

where C is the diagonal cost function of interest. Here, K is a positive integer and B is a scalar, and in order to rigorously show super-Grover speedups, both are chosen carefully based on properties of C. In Algorithm 1 of Ref. [9], the system is initialized in $|+\rangle^{\otimes N}$ with $s = 1$ and then the transverse field is adiabatically turned off. Then, one computes the energy C in a quantum register, and the idea is to apply amplitude amplification using this state preparation in order to amplify outcomes for which the computed energy is below some target threshold. In order to simplify analysis of the algorithm Hastings proposes to use a measurement-based scheme similar to the Zeno approach described in Sec. III 3. For the cases considered in Ref. [9] this combination reduces to the following very simple algorithm (Algorithm 3 of Ref. [9]) on which amplitude amplification is applied:

1. Initialize the system in the state $|\psi\rangle = |+\rangle^{\otimes N}$.
2. Perform phase estimation on $|\psi\rangle$ under the Hamiltonian $H(1)$ defined in Eq. (E1). If the energy is greater than a threshold $E_{\text{threshold}}$, terminate the algorithm and return failure to the amplitude amplification flag.
3. If the previous step has succeeded, use a direct-energy oracle to measure the energy of the state into a quantum register. If the energy is equal to E_0, return success to the amplitude amplification flag (else return failure).

The algorithm is to use amplitude amplification to boost the flag bit to near unit success. The work of Ref. [9] points out that the algorithm could work either by using a quantum walk such as qubitization, or with time evolution.

We note that it is possible to simplify the implementation of this algorithm with a technique that also marginally improves performance (by increasing the success probability by an exponentially small factor). Our modification is to suggest that one proceed to step 3 regardless of whether or not step 2 succeeds. In doing this, we see that because the result of the phase-estimation measurement is never used, we do not actually need the ancilla or controls involved in phase estimation. Instead, we can follow similar logic to Ref. [21] to see that the procedure becomes equivalent to performing time evolution (or applying a quantum walk) for randomly chosen duration. We can choose the probability distribution to suppress phase-measurement errors as large as the energy gap, as described in Sec. III 3.

To explain the effect of this approach in a different way, consider writing the initial state as

$$|\psi\rangle = \sum_j \langle\psi_{j,1}|\psi\rangle\,|\psi_{j,1}\rangle, \tag{E2}$$

where $|\psi_{j,1}\rangle$ are the eigenstates of $H(1)$. Then the evolution for time t gives

$$|\psi\rangle = \sum_j \langle\psi_{j,1}|\psi\rangle e^{-iE_{j,1}t}\,|\psi_{j,1}\rangle. \tag{E3}$$

The squared overlap with the desired solution state $|\psi_{0,0}\rangle$ is

$$p_{\text{succ}}(t) = \sum_{j,k} \langle\psi_{j,1}|\psi\rangle\langle\psi|\psi_{k,1}\rangle e^{-i(E_{j,1}-E_{k,1})t} \times \langle\psi_{0,0}|\psi_{j,1}\rangle\langle\psi_{k,1}|\psi_{0,0}\rangle. \tag{E4}$$

This expression corresponds to the probability of measuring the solution state after the evolution. If we average over t with probability $p_{\text{time}}(t)$, then we have

$$p_{\text{succ}} = \sum_{j,k} \langle\psi_{j,1}|\psi\rangle\langle\psi|\psi_{k,1}\rangle\tilde{p}_{\text{time}}(E_{j,1} - E_{k,1}) \times \langle\psi_{0,0}|\psi_{j,1}\rangle\langle\psi_{k,1}|\psi_{0,0}\rangle, \tag{E5}$$

where

$$\tilde{p}_{\text{time}}(E_{j,1} - E_{k,1}) = \int dt p_{\text{time}}(t) e^{-i(E_{j,1}-E_{k,1})t}. \tag{E6}$$

Thus $\tilde{p}_{\text{time}}$ corresponds to a Fourier transform of p_{time}. If p_{time} is chosen such that its Fourier transform goes to zero before the minimum energy gap, then $\tilde{p}_{\text{time}}(E_{j,1} - E_{k,1})$ is nonzero only for $j = k$. That is equivalent to having a measurement of phase with zero probability of error as large as the energy gap. Then the average probability is

$$p_{\text{succ}} = \sum_j |\langle\psi_{j,1}|\psi\rangle|^2 |\langle\psi_{0,0}|\psi_{j,1}\rangle|^2 \geq |\langle\psi_{0,1}|\psi\rangle|^2 |\langle\psi_{0,0}|\psi_{0,1}\rangle|^2. \tag{E7}$$

Thus the average probability of success is at least as large as $|\langle\psi_{0,1}|\psi\rangle|^2|\langle\psi_{0,0}|\psi_{0,1}\rangle|^2$ as given by Hastings' approach. A minor drawback as compared to Hastings' approach is that only a single time is used, so if it happens that this time gives $p_{\text{succ}}(t)$ significantly smaller than average the amplitude amplification does not give the solution.

The original motivation for the SPA seems to be primarily to produce an algorithm where a rigorous analysis can be performed, and so it is debatable whether one actually wants to try to use the algorithm heuristically rather than via some other approach. If one did want to use this approach heuristically there are many ways that could be accomplished; for instance, by choosing E_{target}, B, and K heuristically and then resolving to use a fixed number of rounds of amplitude amplification. Note that in the variant we describe it as no longer necessary to have an $E_{\text{threshold}}$, although one still needs to choose the precision to which one performs phase estimation. One can see that such a heuristic variant of this algorithm could be implemented by using either our Hamiltonian walk or Trotter step oracles for the evolution, followed by our direct-energy oracles for computing the amplitude amplification target.

2. Quantum-enhanced population transfer

Another heuristic algorithm for optimization, which has been proposed is the quantum-enhanced population transfer (QEPT) method of Ref. [10,11]. Unlike quantum heuristics, which begin in a uniform superposition state, QEPT proposes to use quantum dynamics to evolve from one low-energy solution of an optimization problem to other low energy solutions of similar energy. The idea is motivated by the search of configuration space in the classically nonergodic phase associated with hard optimization problems. Such energy landscapes contain an extensive number of local minima separated by large Hamming distances. Algorithms relying on classical dynamics satisfying the detailed balance condition, such as simulated annealing, tend to get trapped at local minima of these landscapes. Thus, one could alternatively apply classical simulated annealing until the algorithm becomes trapped, then apply QEPT starting from that state, then again apply simulated annealing starting from the QEPT solutions, and so on.

Specifically, the context studied in Ref. [10,11] is as follows. Consider a cost function C on N qubits and bit string x with energy E_x (so that $C\,|x\rangle = E_x\,|x\rangle$). The problem solved by QEPT is to produce another bit string y within a small energy window $E_y \in [E_x - \delta/2, E_y + \delta/2]$ such that the Hamming distance $d_{x,y}$ between x and y is $\mathcal{O}(N)$. In the presence of a spin-glass-type energy landscape finding such states y using a classical algorithm takes exponential resources. The QEPT procedure suggests solving the above computational task as follows.

1. Prepare the system in the initial state $|x\rangle$.
2. Turn on a transverse field Hamiltonian $\sum_{k=1}^{N} X_i$ up to some optimal field strength $B_\perp = \mathcal{O}(\|C\|/N)$ with ramp up time polynomial in N.
3. Evolve for time T under the fixed Hamiltonian
$$H = C + B_\perp \sum_{k=1}^{N} X_k. \tag{E8}$$
4. Measure in the computational basis and check the classical energy of the observed state.

In general, we expect that T scales exponentially in order for the procedure to succeed with fixed probability. However, for the worst-case scenario when there are M states with energy -1 and $2^N - M$ states of energy 0, the work of Ref. [11] was able to show that this procedure succeeds with high probability for $T = \mathcal{O}(\sqrt{2^N/M})$, which is the same as the Grover scaling. However, unlike Grover, this protocol does not require any fine tuning of the transverse field or computation time. The procedure has also been shown empirically to produce similar results for the random energy model (where each bit string has a totally random energy).

The suggestion to use this algorithm heuristically is simply to choose T, as well as the accuracy with which we implement the time evolution, heuristically. Like with the adiabatic algorithm, this essentially corresponds to the number of steps that we take in either a product formula, or quantum walk approach to simulating the evolution. We propose that when using the quantum walk form of the algorithm, one does not use signal processing and instead perform population transfer directly on the quantum walk. We note that since the norm of the problem and driver Hamiltonians are similar in magnitude, there is no advantage to performing simulation in the interaction picture and so an approach based on qubitization is likely the best LCU style algorithm for QEPT.

[1] F. Arute *et al.*, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).

[2] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem, Science **292**, 472 (2001).

[3] L. K. Grover, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96 (Association for Computing Machinery, New York, NY, USA, 1996), p. 212.

[4] C. Durr and P. Hoyer, A quantum algorithm for finding the minimum, arXiv:quant-ph/9607014 (1996).

[5] P. Ray, B. K. Chakrabarti, and A. Chakrabarti, Sherrington-Kirkpatrick model in a transverse field: Absence of replica symmetry breaking due to quantum fluctuations, Phys. Rev. B **39**, 11828 (1989).

[6] T. Kadowaki and H. Nishimori, Quantum annealing in the transverse Ising model, Phys. Rev. E **58**, 5355 (1998).

[7] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution, arXiv:quant-ph/0001106 (2000).

[8] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, Adiabatic quantum computation is equivalent to standard quantum computation, SIAM J. Comput. **37**, 166 (2007).

[9] M. B. Hastings, A short path quantum algorithm for exact optimization, Quantum **2**, 78 (2018).

[10] K. Kechedzhi, V. Smelyanskiy, J. R. McClean, V. S. Denchev, M. Mohseni, S. Isakov, S. Boixo, B. Altshuler,

and H. Neven, in *13th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2018)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 111, edited by S. Jeffery (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018), p. 9:1.

[11] V. N. Smelyanskiy, K. Kechedzhi, S. Boixo, S. V. Isakov, H. Neven, and B. Altshuler, Nonergodic Delocalized States for Efficient Population Transfer within a Narrow Band of the Energy Landscape, Phys. Rev. X **10**, 011017 (2020).

[12] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028 (2014).

[13] R. D. Somma, S. Boixo, H. Barnum, and E. Knill, Quantum Simulations of Classical Annealing Processes, Phys. Rev. Lett. **101**, 130504 (2008).

[14] S. Boixo, G. Ortiz, and R. Somma, Fast quantum methods for optimization, Eur. Phys. J. Spec. Top. **224**, 35 (2015).

[15] A. Montanaro, Quantum walk speedup of backtracking algorithms, arXiv:1509.02374 (2015).

[16] E. Campbell, A. Khurana, and A. Montanaro, Applying quantum algorithms to constraint satisfaction problems, Quantum **3**, 167 (2019).

[17] A. Montanaro, Quantum speedup of branch-and-bound algorithms, Phys. Rev. Res. **2**, 013056 (2020).

[18] A. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. **303**, 2 (2003).

[19] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[20] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, in *Quantum Computation and Information*, edited by Vitaly I. Voloshin, Samuel J. Lomonaco, and Howard E. Brandt (American Mathematical Society, Washington, DC, 2002), Chap. 3, p. 53.

[21] S. Boixo, E. Knill, and R. Somma, Quantum state preparation by phase randomization, Quantum Inf. Comput. **9**, 0833 (2009).

[22] M. Szegedy, in *45th Annual IEEE Symposium on Foundations of Computer Science*, (2004), p. 32.

[23] J. Lemieux, B. Heim, D. Poulin, K. Svore, and M. Troyer, Efficient quantum walk circuits for metropolis-hastings algorithm, Quantum **4**, 287 (2020).

[24] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by simulated annealing, Science **220**, 671 (1983).

[25] I. D. Kivlichan, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, W. Sun, Z. Jiang, N. Rubin, A. Fowler, A. Aspuru-Guzik, H. Neven, and R. Babbush, Improved fault-tolerant quantum simulation of condensed-phase correlated electrons via trotterization, Quantum **4**, 296 (2020).

[26] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity, Phys. Rev. X **8**, 041015 (2018).

[27] Y. R. Sanders, G. H. Low, A. Scherer, and D. W. Berry, Black-Box Quantum State Preparation without Arithmetic, Phys. Rev. Lett. **122**, 020502 (2019).

[28] P. Gokhale, S. Koretsky, S. Huang, S. Majumder, A. Drucker, K. R. Brown, and F. T. Chong, Quantum fan-out: Circuit optimizations and technology modeling, arXiv:2007.04246 (2020).

[29] J. M. Baker, C. Duckering, and F. T. Chong, Efficient quantum circuit decompositions via intermediate qudits, arXiv:2002.10592 (2020).

[30] P. Gokhale, J. M. Baker, C. Duckering, N. C. Brown, K. R. Brown, and F. T. Chong, in *Proceedings of the 46th International Symposium on Computer Architecture* (ACM, Phoenix, Arizona, 2019).

[31] G. H. Low and I. L. Chuang, Hamiltonian simulation by qubitization, Quantum **3**, 163 (2019).

[32] S. Boixo, T. F. Ronnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, Quantum annealing with more than one hundred qubits, Nat. Phys. **10**, 218 (2014).

[33] J. Bernasconi, Low autocorrelation binary sequences: Statistical mechanics and configuration space analysis, J. Phys. **48**, 559 (1987).

[34] T. Packebusch and S. Mertens, Low autocorrelation binary sequences, J. Phys. A: Math. Theor. **49**, 165001 (2016).

[35] C. Gidney, Halving the cost of quantum addition, Quantum **2**, 74 (2018).

[36] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi, *Graduate Studies in Mathematics* (American Mathematical Society, Providence, Rhode Island, 2002), Vol. 47.

[37] A. Bocharov, M. Roetteler, and K. M. Svore, Efficient Synthesis of Universal Repeat-Until-Success Quantum Circuits, Phys. Rev. Lett. **114**, 080502 (2015).

[38] C. Gidney and A. G. Fowler, Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2|T\rangle$ transformation, Quantum **3**, 135 (2019).

[39] A. M. Childs and N. Wiebe, Hamiltonian simulation using linear combinations of unitary operations, Quantum Inf. Comput. **12**, 901 (2012).

[40] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, Simulating Hamiltonian dynamics with a truncated Taylor series, Phys. Rev. Lett. **114**, 090502 (2015).

[41] G. H. Low and N. Wiebe, Hamiltonian simulation in the interaction picture, arXiv:1805.00675 (2018).

[42] D. W. Berry, C. Gidney, M. Motta, J. R. McClean, and R. Babbush, Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization, Quantum **3**, 208 (2019).

[43] G. H. Low, V. Kliuchnikov, and L. Schaeffer, Trading T-gates for dirty qubits in state preparation and unitary synthesis, arXiv:1812.00954 (2018).

[44] T. J. Yoder, G. H. Low, and I. L. Chuang, Fixed-Point Quantum Search with an Optimal Number of Queries, Phys. Rev. Lett. **113**, 210501 (2014).

[45] B. Barak, A. Moitra, R. O'Donnell, P. Raghavendra, O. Regev, D. Steurer, L. Trevisan, A. Vijayaraghavan, D. Witmer, and J. Wright, in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24–26, 2015, Princeton, NJ, USA*, LIPIcs, Vol. 40, edited by N. Garg, K. Jansen, A. Rao, and J. D. P. Rolim (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015), p. 110.

[46] F. Arute *et al.*, Quantum approximate optimization of non-planar graph problems on a planar superconducting processor, arXiv:2004.04197 (2020).

[47] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien,

A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).

[48] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).

[49] F. G. S. L. Brandao, M. Broughton, E. Farhi, S. Gutmann, and H. Neven, For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances, arXiv:1812.04170 (2018).

[50] E. Farhi, J. Goldstone, S. Gutmann, and L. Zhou, The quantum approximate optimization algorithm and the Sherrington-Kirkpatrick model at infinite size, arXiv:1910.08187 (2019).

[51] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices, Phys. Rev. X **10**, 021067 (2020).

[52] A. Gilyén, S. Arunachalam, and N. Wiebe, in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, San Diego, California, 2019), p. 1425.

[53] A. Montanaro, Quantum speedup of Monte Carlo methods, Proc. R. Soc. A **471**, 20150301 (2015).

[54] E. Farhi, J. Goldstone, and S. Gutmann, A numerical study of the performance of a quantum adiabatic evolution algorithm for satisfiability, arXiv:quant-ph/0007071 (2000).

[55] A. Elgart and G. A. Hagedorn, A note on the switching adiabatic theorem, J. Math. Phys. **53**, 102202 (2012).

[56] D. A. Lidar, A. T. Rezakhani, and A. Hamma, Adiabatic approximation with exponential accuracy for many-body systems and quantum computation, J. Math. Phys. **50**, 102106 (2009).

[57] N. Wiebe and N. S. Babcock, Improved error-scaling for adiabatic quantum evolutions, New J. Phys. **14**, 013024 (2012).

[58] M. Kieferová and N. Wiebe, On the power of coherently controlled quantum adiabatic evolutions, New J. Phys. **16**, 123034 (2014).

[59] K. Wan and I. Kim, Fast digital methods for adiabatic state preparation, arXiv:2004.04164 (2020).

[60] N. Wiebe, D. Berry, P. Høyer, and B. C. Sanders, Higher order decompositions of ordered operator exponentials, J. Phys. A: Math. Theor. **43**, 065203 (2010).

[61] J. Lemieux, G. Duclos-Cianci, D. Sénéchal, and D. Poulin, arXiv:2006.04650 (2020).

[62] H.-T. Chiang, G. Xu, and R. D. Somma, Improved bounds for eigenpath traversal, Phys. Rev. A **89**, 012314 (2014).

[63] J. Kaiser and R. Schafer, On the use of the I0-sinh window for spectrum analysis, IEEE Trans. Acoust. **28**, 105 (1980).

[64] D. W. Berry, M. Kieferová, A. Scherer, Y. R. Sanders, G. H. Low, N. Wiebe, C. Gidney, and R. Babbush, Improved techniques for preparing eigenstates of fermionic Hamiltonians, npj Quantum Inf. **4**, 22 (2018).

[65] D. Poulin, A. Kitaev, D. S. Steiger, M. B. Hastings, and M. Troyer, Quantum Algorithm for Spectral Measurement with a Lower Gate Count, Phys. Rev. Lett. **121**, 010501 (2018).

[66] T. Häner, M. Roetteler, and K. M. Svore, Optimizing quantum circuits for arithmetic, arXiv:1805.12445 (2018).

[67] D. W. Berry, A. M. Childs, R. Cleve, R. Kothari, and R. D. Somma, in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14 (ACM, New York, NY, USA, 2014), p. 283.

[68] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, arXiv:quant-ph/9811052 (1998).

[69] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

[70] R. Raussendorf and J. Harrington, Fault-Tolerant Quantum Computation with High Threshold in Two Dimensions, Phys. Rev. Lett. **98**, 190504 (2007).

[71] A. G. Fowler and C. Gidney, Low overhead quantum computation using lattice surgery, arXiv:1808.06709 (2018).

[72] N. C. Jones, J. D. Whitfield, P. L. McMahon, M.-H. Yung, R. V. Meter, A. Aspuru-Guzik, and Y. Yamamoto, Faster quantum chemistry simulation on fault-tolerant quantum computers, New J. Phys. **14**, 115023 (2012).

[73] B. Eastin, Distilling one-qubit magic states into Toffoli states, Phys. Rev. A **87**, 032321 (2013).

[74] A. G. Fowler, Optimal complexity correction of correlated errors in the surface code, arXiv:1310.0863 (2013).

[75] S. V. Isakov, I. Zintchenko, T. Rønnow, and M. Troyer, Optimised simulated annealing for Ising spin glasses, Comput. Phys. Commun. **192**, 265 (2015).

[76] S. V. Isakov, Personal communication about simulated annealing code in [75] (2020).

[77] A. Wouk, Integral representation of the logarithm of matrices and operators*, J. Math. Anal. Appl. **11**, 131 (1965).

[78] M. B. Hastings, Weaker assumptions for the short path optimization algorithm, arXiv:1807.03758 (2018).