Accelerating Quantum Algorithms with Precomputation

William J. Huggins and Jarrod R. McClean

Google Quantum AI, Venice, CA, USA

2024-02-15

Real-world applications of computing can be extremely time-sensitive. It would be valuable if we could accelerate such tasks by performing some of the work ahead of time. Motivated by this, we propose a cost model for quantum algorithms that allows quantum precomputation; i.e., for a polynomial amount of "free" computation before the input to an algorithm is fully specified, and methods for taking advantage of it. We analyze two families of unitaries that are asymptotically more efficient to implement in this cost model than in the standard one. The first example of quantum precomputation, based on density matrix exponentiation, could offer an exponential advantage under certain conditions. The second example uses a variant of gate teleportation to achieve a quadratic advantage when compared with implementing the unitaries directly. These examples hint that quantum precomputation may offer a new arena in which to seek quantum advantage.

1 Introduction

In order to efficiently use limited computational resources, it is natural to quantify and minimize their use. In quantum computing, we frequently try to minimize some proxy for the spacetime cost of an algorithm, such as the number of two-qubit gates on an near-term machine or the number of non-Clifford gates on a fault-tolerant device. Focusing on spacetime metrics allows one to easily incorporate the fungibility of additional qubits and time inside error correcting codes [18, 21, 35], as well as elements of algorithmic parallelism. However, in some cases, one is interested in the raw time to solution, or "wall-clock time," given any reasonable resources. As such, in this paper, we explore a different cost model that allows for what we call "quantum precomputation." In the process, we aim to understand the opportunities and challenges inherent in generalizing classical ideas of precomputation, e.g., caching of results, indexing in databases, or creating lookup tables. The precomputation cost model allows for a quantum algorithm to start with access to a specially prepared resource state that depends on the algorithm and some portion (but not all) of its input. We neglect the cost of preparing this resource state, but we demand that it can be prepared efficiently, i.e., that the quantum and classical resources required scale polynomially in the size of the input.

Our precomputation cost model is motivated by real-world problems where the crucial limited resource is the computational power available after the problem is fully specified.

William J. Huggins: whuggins@google.com

For some of these problems, the value of finding a solution as quickly as possible would justify investing extra effort ahead of time preparing to perform a computation. In fields ranging from optimization, to finance, to data analysis, there are tasks that naturally fit into this framework. If we can build useful quantum primitives that accelerate such tasks in the precomputation cost model, it could have a substantial impact even in cases where the overall quantum advantage is modest or non-existent. We study quantum precomputation because of these potential practical applications, and also because it offers the chance to investigate the nature of quantum computation from another angle. Notably, the no-cloning theorem imposes limitations on our ability to reuse the results of earlier quantum computations, which implies that precomputation may occupy a different role in quantum computing than it does classically.

In order for the precomputation cost model to make sense, there must be some components of the computational task that are naturally specified before others. For example, we could be given a classical description of a Hamiltonian now with the understanding that we will want to estimate some properties of its ground state that will be determined at a later time. In such a situation, we could prepare for when these properties are specified by generating and storing a sufficient number of copies of the ground state. In other cases, we might have a classical description of some unitary U available now that we will later wish to apply to a (currently unknown) state $|\psi\rangle$. In this paper, we ask if we can find interesting or useful families of tasks that can be implemented using asymptotically fewer quantum resources in a cost model that allows for free precomputation.

We formalize our definition of the precomputation cost model in Section 2. In Section 3, we discuss some of the connections that quantum precomputation has with prior work on quantum and classical computation. We go on to explore how existing algorithmic primitives can interpreted as tools for quantum precomputation in Section 4. Specifically, we make use of density matrix exponentiation and gate teleportation to accelerate the application of certain unitaries in the precomputation cost model [23, 36], finding the possibility of speedups that range from quadratic to exponential (when comparing the cost in the precomputation model with the usual quantum gate complexity). In Section 5, we present a less straightforward protocol for quantum precomputation that uses a technique known as selective teleportation [18] to yield a quadratic improvement in complexity for a family of diagonal unitaries. We conclude with a discussion of open questions and potential applications in Section 6.

2 The Precomputation Cost Model

2.1 Formalizing the cost model

Analyzing the resources required to execute an algorithm requires a cost model. A good cost model encodes useful assumptions that simplify the analysis, abstracting away irrelevant details while keeping the essential information required to answer the questions at hand. There are a number of different choices one could make in formalizing the intuition behind quantum precomputation into a cost model; i.e., specifying what it means to "allow a reasonable amount of work to be performed for free." In this section, we propose a concrete definition flexible enough to encompass several interesting examples rather than a maximally general abstract definition.

There are many kinds of computational tasks that we might wish to analyze in the precomputation cost model. We will loosely formalize a computational task as an algorithm, which we treat as a map that takes an input from some set of valid inputs and returns a correct output (or a sample from a correct distribution over outputs). Different algorithms may define different notions of valid inputs and correct outputs. For now, we leave these details unspecified, although they may be crucial to determining the complexity of implementing an algorithm. For example, there are some tomographic tasks that are efficient for pure state inputs but prohibitively expensive for general mixed state inputs [22]. In other cases, the computational complexity of a problem may vary depending on the definition of the "correct" output, e.g., what kind of approximation is allowed [19].

To be sufficiently general, we need a notion of a quantum algorithm that can accept both quantum and classical input and can output both quantum and classical data.[1] We also need to allow for the possibility that the input is partitioned into two components that are provided at different times. For simplicity, we assume that the earlier input (that might be used in the precomputation step) is classical, and that the later input may be a combination of classical and quantum data. Let x denote the (classical) input provided at the earlier time and let ρ and y denote the quantum and classical components of the input provided at a later time. For the quantum and classical outputs we use the symbols σ and z respectively.

In the usual situation, where we do not take advantage of the fact that some portion of the input may be available ahead of time, a quantum algorithm $\mathcal{A}$ implements a map

$$\mathcal{A} : x, y, \rho \to z, \sigma. \tag{1}$$

In general, we can understand $\mathcal{A}$ as performing some classical computation that takes x and y as an input, determining a quantum circuit that is subsequently applied to ρ. The portions of the resulting quantum state that are not measured or discarded constitute σ. The classical component of the output, z, is classically computed from x, y, and the measurement outcomes. In a standard cost model, we are concerned with the cost of executing the algorithm $\mathcal{A}$ given access to x, y, and ρ.

In a model that allows for free precomputation, we aim to produce the same (distribution over) outputs by implementing the map

$$\mathcal{P} : \bar{x}(\mathcal{A}, x), \left|\Gamma(\mathcal{A}, x)\right\rangle, y, \rho \to z, \sigma, \tag{2}$$

where $\bar{x}(\mathcal{A}, x)$ and $|\Gamma(\mathcal{A}, x)\rangle$ represent the classical and quantum outputs of some precomputation step. We allow for $\bar{x}(\mathcal{A}, x)$ and $|\Gamma(\mathcal{A}, x)\rangle$ to be generated using a "reasonable" amount of classical and quantum computation performed ahead of time, i.e., with knowledge of $\mathcal{A}$ and x but not ρ or y. In a precomputation cost model, the only cost that we consider directly is the cost of performing the map $\mathcal{P} : \bar{x}(\mathcal{A}, x), \left|\Gamma(\mathcal{A}, x)\right\rangle, y, \rho \to z, \sigma$. In order to fully define a precomputation cost model and compare it to a standard cost model, we therefore have to specify answers to two questions: i) How will we quantify the costs of implementing $\mathcal{A}$ and $\mathcal{P}$? ii) What do we mean when we say that we allow for a "reasonable" amount of classical and quantum computation to be used in the preparation of $|\Gamma(\mathcal{A}, x)\rangle$ and $\bar{x}(\mathcal{A}, x)$?

In this paper, we focus on quantifying the quantum resources used to implement $\mathcal{P}$ (and $\mathcal{A}$ itself) in terms of the quantum circuit complexity (a term that we use interchangeably with "gate complexity"), the number of gates from some elementary set of discrete operations required to implement the algorithm. We consider a discrete set of gates that consists of one- and two-qubit Clifford gates, single-qubit computational basis measurement operations, and T gates. We also choose to count single-qubit identity operations

[1] We could consider all of the inputs and outputs to be quantum states, but treating them separately will help us take a more nuanced approach that differentiates the quantum and classical resources.

as gates in order to include the cost of storage (which is comparable in most architectures to the cost of active workspace). This choice implies that our notion of circuit complexity grows asymptotically as fast as the product of the number of qubits and the circuit depth (the number of layers of gates, executed in parallel).

We could define other related models that allow for free precomputation but account for "cost" differently. Depending on the context, it might be useful to work in an oracle model, or to count only the number of non-Clifford gates, or even to quantify the space-time volume used in a particular error-correcting code. It might also be useful to discuss the number of gates required for the best known implementation of an algorithm, rather than the absolute minimum required. For the examples we consider, this distinction will not be important. We find that discussing the gate complexity is convenient because it allows us to use the same model to consider several different examples, but we will make some comments along the way regarding other notions of cost. As we consider these examples, it will sometimes make sense to allow for $\mathcal{A}$ or $\mathcal{P}$ to be implemented with some error. In the context of this work, when we need to allow for some notion of error, it will be sufficient to focus on the case where the output is a quantum state and we can quantify the error using a single parameter ϵ that bounds the trace distance between the ideal output and the actual output.

By focusing on quantifying the cost in the precomputation model in terms of the number of quantum operations, we are implicitly treating quantum operations as a fundamentally different and more limited resource than classical ones. This decision is motivated by the practical observation that quantum operations on a fault-tolerant computer are expected to be vastly slower and more expensive than classical operations [4]. Nevertheless, we would like a definition of the precomputation cost model that is useful in practice, so we demand that the classical time and space complexity of implementing $\mathcal{P}$ scales as $\mathcal{O}(\text{poly}(\epsilon^{-1}, |x|, |y|, |\rho|))$. Here the notation $|*|$ indicates the size of $*$ in terms of classical or quantum bits.

Besides specifying how we quantify the cost of implementing $\mathcal{A}$ or $\mathcal{P}$, we also need to formalize the notion that the amount of work performed ahead of time is required to be "reasonable." We should bound the quantum gate complexity of the precomputation step, as well the classical time and space complexities. For all of these resources, we allow their usage during the precomputation step to scale as $\mathcal{O}(\text{poly}(\epsilon^{-1}, |x|))$. Although we define our model with this coarse-grained notion of what is allowed during the precomputation step, we will discuss the actual scaling of the various resources in more detail for the particular examples we consider in this paper.

3 Prior work

While the authors are not aware of prior work that has focused on a cost model that allows for free precomputation in the sense that we consider, there are a number of closely related ideas that we draw inspiration from. The paper that first described gate teleportation speculated that it might be used to mass manufacture resource states for later consumption [23]. For example, one could imagine using magic state distillation to distill a large number of magic states, storing them for use in a later computation [8]. Going beyond the prototypical use of magic state distillation to implement a T gate, state distillation schemes have been proposed for a variety of other few-qubit operations [10, 13, 21, 28, 35]. In Ref. 13, Jones et al. proposed a method that implements an arbitrary single-qubit Z rotation with success probability $1 - \delta$ by precomputing and storing a resource state on $\mathcal{O}(-\log(\delta))$ qubits. More abstractly, measurement based quantum computing has some similarity to quantum precomputation, but it aims to prepare generically useful resource states rather than ones that are tailored to accelerating particular algorithms [40, 43].

The idea of precomputing and storing a reservoir of resource states for single or few-qubit operations is appealing, but it faces serious challenges. In particular, the number of such resource states required for interesting and classically intractable applications appears large [6, 20, 44], while quantum memory has a comparable cost with active workspace in most proposed architectures [49]. For example, Ref. 20 estimates that thousands of logical qubits and billions of Toffoli and T gates would be required to factor a 2048 bit RSA integer using Shor's algorithm. A fault-tolerant quantum computer that large enough to perform this computation, but not too much larger, would be unable to precompute and store more than a tiny fraction of the necessary resource states ahead of time.

Even so, one might ask if precomputing resource states for T or Toffoli gates offers a simple example of asymptotic advantage when the cost of the precomputation itself is neglected. In our definition of the precomputation cost model, the answer is no. This is because, even with access to the appropriate resource state, applying either of these gates still requires a (nonzero) constant number of operations and our model allows T gates to be performed at unit cost. If we instead consider the task of implementing arbitrary single-qubit rotations to within some precision ϵ, Ref. 13 provides an example where allowing for free precomputation does indeed change the asymptotic cost. Specifically, precomputation can be used to remove the dependence on ϵ from the cost (not including the cost of the precomputation step) at the expense of incurring some logarithmic dependence on the allowed failure probability δ.

The idea of supplementing a quantum computer with a specially-prepared resource state has also been considered from a complexity-theoretic perspective. The complexity class BQP/qpoly formalizes the power of a polynomial-time quantum computer augmented with an arbitrary resource state, referred to as "quantum advice," that is allowed to depend on the length of the input. Comparing this complexity class to our model of quantum precomputation requires some care, so we provide a longer discussion in Appendix A and merely summarize the conclusions here. First of all, the model formalized in BQP/qpoly places no restrictions on the computational power used to prepare the resource state, whereas we require that it be preparable in polynomial time. Secondly, the quantum advice states of BQP/qpoly can only depend on the length of the input. We allow for the resource states to depend on a subset of the parameters, denoted by x. Thirdly, the only problems that fit into the framework of BQP/qpoly are decision problems, which have a classical input and a (single bit of) classical output. This is a more limited setting than the one that we consider.[2]

Despite these differences between BQP/qpoly and our notion of quantum precomputation, we can make a useful comparison if we restrict ourselves to considering the power of both models to solve decision problems. One might suspect that our model of quantum precomputation gets some additional power from the fact that we allow the resource state to depend on the input in richer ways than allowed by the complexity class BQP/qpoly. However, any decision problem that is solvable in polynomial time in the precomputation model we have defined is not only a member of BQP/qpoly, but also BQP itself. This is because we only allow a polynomial amount of "free" precomputation, which can't add any power to a machine that is already allowed to run arbitrary polynomial-time quantum computations. Ultimately, our model of quantum precomputation is trying to capture a

[2] One could imagine analogues of BQP/qpoly that use a similar notion of advice but consider problems beyond the setting of decision problems. The main benefit of focusing exclusively on decision problems is that they are simple to formalize precisely.

finer-grained notion of speedup than these particular complexity classes are designed to address. Imprecisely, we could say that we are interested in the power of the "advice that a polynomial time quantum computer can give itself."

In the context of classical computing, the term "precomputation" has been used extensively to describe variations on the idea of performing useful work ahead of time and caching the result. For example, branch-prediction is an essential component of modern computer architecture design [47]. Precomputation is used to optimize certain tasks in computer graphics [46] and computer vision [24]. The precomputation of expensive operations involved in breaking cryptographic schemes is both a practical and theoretical concern [5], which is closely related to the study of advice in classical computational complexity theory [30]. For the most part, these examples seem slightly different than the quantum algorithmic primitives that we will discuss. Classically, some applications of precomputation derive their usefulness from the ability to reuse the precomputed information rather than the time-sensitive nature of the computation. In contrast with the classical case, the resource states that we consider are generally consumed when used, precluding their reuse. It would be interesting if other techniques, perhaps based on gentle measurements [3], can be used to design quantum precomputation protocols that allow for some amount of information reuse.

4 Examples of Precomputation

In this section, we discuss several examples of quantum precomputation. These examples show how existing quantum primitives can be leveraged to obtain an advantage in a cost model that allows for free precomputation. In particular, we study the application of density matrix exponentiation (introduced in Ref. 36, reviewed in Appendix B.1) and gate teleportation (introduced in Ref. 23, reviewed in Appendix B.2) as tools for quantum precomputation.

Before turning towards these examples, it is worth briefly discussing two particularly simple forms of quantum precomputation. One natural example is the case where precomputation is equivalent to performing the first steps of some algorithm and then waiting until the problem is fully specified to perform the rest. For example, many quantum algorithms consist of applying a known unitary to the all zero state and performing a measurement. If we knew the unitary ahead of time but the measurement wasn't yet specified, we could perform the state preparation in advance. More speculatively, there may be settings where it is natural to prepare for the future execution of some quantum machine learning task by encoding data into a quantum state "on the fly" as it streams in. This latter idea is related to rigorous work on quantum algorithms in streaming settings, which is itself connected to the study of quantum communication complexity [29, 32].

It is easy to understand how one might be able to usefully perform precomputation by executing the steps at the beginning of some algorithm ahead of time. We could try to imagine situations where this naturally occurs, but it is unclear if our formal definition of the notion of quantum precomputation adds anything to the understanding of such cases. For this reason, in the other examples that we consider in this paper, we focus instead on the goal of using precomputation to accelerate steps that lie in the middle of an algorithm, rather than at the beginning.

Turning towards a second example, recall that we briefly discussed the idea of precomputing magic states to use as resources for implementing non-Clifford gates in an error correcting code in Section 3. We explained how there is no advantage to this idea in the primary cost model we use throughout this paper because we do not distinguish between

Clifford and non-Clifford gates. This is true, but it is instructive to consider this example in a slightly different model of quantum precomputation, where we instead quantify the amount of spacetime volume required to implement a circuit in a quantum error correcting code. For simplicity, let us work in units where a depth d circuit acting on n qubits has a volume of dn and let us assume that the spacetime volume required to prepare a suitably distilled T state is $\lambda \gg 1$. Furthermore, we will neglect the spacetime cost of qubits that have not yet been initialized and qubits that have already been measured (since they could presumably be used for other purposes).

Under this more nuanced cost model, we can compare the cost of implementing an algorithm with and without the precomputed T states. Let us consider a depth d circuit on n qubits that consumes one magic state per time step. Implementing this algorithm without precomputation would require a spacetime volume of $nd + \lambda d$ in order to account for the cost of the circuit itself and the cost of the magic state distillation. In the precomputation model, we allow ourselves to start with all d magic states already prepared, but we must account for the cost of storing them while the algorithm executes. We are using $d - s$ qubits to store the magic states at each step s from 0 to $d - 1$, so the spacetime volume required is $nd + \frac{d(d+1)}{2}$.

In this cost model, precomputing the magic states removes the dependence on λ but it increases the dependence on d from linear to quadratic. Realistic values of λ are expected to be significantly less than 100, which suggests that only relatively short-depth circuits of this type would benefit from free access to precomputed magic states [34]. This example highlights the fact that our model implicitly penalizes precomputation protocols for the space used to store their precomputed resource states. Because of this penalization, it is not trivially true that a precomputation protocol is at least as efficient as a straightforward approach to executing an algorithm.

4.1 Precomputation with density matrix exponentiation

In this subsection, we consider applications where reflections about an expensive to prepare state, $|b\rangle$, are a dominant contribution to the complexity of an algorithm. As we explain below, an algorithm that requires q calls to the reflection operator $R = \mathbb{I} - 2\,|b\rangle\langle b|$ can be implemented by consuming $\mathcal{O}(q^2)$ copies of $|b\rangle$ (at nearly unit time per consumption) in lieu of making any calls to R directly. A cost model that allows for free precomputation can therefore entirely remove the component of such an algorithm's cost that depends on $|b\rangle$. In the most extreme cases, this could lead to a cost in the precomputation model that is exponentially smaller than the cost in a standard model. For example, preparing or reflecting about the state $|b\rangle$ might require using $\mathrm{poly}(|x|)$ gates to implement a brute-force encoding of some classical input x into $n = \mathrm{polylog}(|x|)$ qubits, while the other components of the algorithm could scale polynomially in n. We consider the quantum algorithm for linear systems as a specific example of an algorithm where such a speedup might prove useful [12, 26].

This type of quantum precomputation makes use of a technique called density matrix exponentiation. Introduced in Ref. 36, density matrix exponentiation allows us to consume copies of some density matrix ρ in order to approximately apply the unitary $e^{-it\rho}$ for some time t. We provide a brief review of density matrix exponentiation in Appendix B.1, but for now we just recall the fact that using density matrix exponentiation to implement $e^{-it\rho}$ to within an error ϵ (in the diamond norm) requires

$$m = \mathcal{O}(t^2/\epsilon) \tag{3}$$

copies of ρ [31].

Before explaining how we can make good use of density matrix exponentiation for quantum precomputation, let us examine why it does not lead to efficient protocols for implementing general unitaries in the precomputation cost model. Imagine that we want to implement a unitary U that corresponds to evolution under a Hamiltonian H for a time t, where $||H||$ (the spectral norm of H) and t are both $\mathcal{O}(1)$. We can shift H by some multiple c of the identity to obtain a positive semidefinite operator $H + c\mathbb{I}$ with $||H + c\mathbb{I}|| = \mathcal{O}(1)$. Applying U using density matrix exponentiation entails evolving under the Hamiltonian corresponding to the normalized state

$$\rho = \frac{H + c\mathbb{I}}{\operatorname{tr}\left[H + c\mathbb{I}\right]} \tag{4}$$

for a time

$$\tilde{t} = t \operatorname{tr}\left[H + c\mathbb{I}\right]. \tag{5}$$

The cost of implementing U using density matrix exponentiation scales quadratically with $\tilde{t}$, which can scale exponentially with the number of qubits in the worst case. This occurs easily even for simple unitaries, for example, when H is a non-trivial Pauli operator.

In order for density matrix exponentiation to be a useful tool for precomputation, we need to focus on cases where the normalization factor is small. One natural example of a unitary that is efficiently implementable using density matrix exponentiation is the reflection about a state $|b\rangle$,

$$R = \mathbb{I} - 2\,|b\rangle\!\langle b| = e^{-i\pi|b\rangle\langle b|}. \tag{6}$$

In order to implement R up to an accuracy $\tilde{\epsilon}$ using density matrix exponentiation, it suffices to consume $\mathcal{O}(\tilde{\epsilon}^{-1})$ copies of the state $|b\rangle\!\langle b|$. If an algorithm involves q calls to R, we can guarantee a constant overall error ϵ by setting $\tilde{\epsilon} \propto \epsilon q^{-1}$. We can therefore implement all q calls to R to within the desired accuracy by consuming a total of $\mathcal{O}(\epsilon^{-1}q^2)$ copies of $|b\rangle\!\langle b|$.

As an example of a context where this kind of precomputation might be useful, consider the quantum linear systems problem [12, 14, 26, 33, 37, 48]. Given a matrix A and a vector $\vec{b}$, the linear systems problem is to find a vector $\vec{x}$ such that $A\vec{x} = \vec{b}$. The quantum formulation of this problem encodes the vector $\vec{b}$ into the amplitudes of a state $|b\rangle$ and asks that we prepare a state $|x\rangle \propto A^{-1}|b\rangle$. Without loss of generality we can assume that A is Hermitian.[3] The access models for A and $|b\rangle$ can vary, but it is usually assumed that one has access to an oracle that prepares $|b\rangle$ and either i) the ability to perform time evolution by A, ii) oracle access to the non-zero entries of (a sparse) A, or iii) a block encoding of A. Regardless of the access model for A, the most efficient algorithms for this problem query the state preparation oracle for $|b\rangle$ a number of times that scales as $\tilde{\mathcal{O}}(\kappa)$, where κ denotes the condition number of A and the $\tilde{\mathcal{O}}(\cdot)$ notation hides logarithmic factors in κ and the precision. These queries are used to prepare $|b\rangle$ and to implement the reflection R about $|b\rangle$.

In a context where a classical description of $|b\rangle$ is available before A, preparing $\tilde{\mathcal{O}}(\epsilon^{-1}\kappa^2)$ copies of $|b\rangle$ during the precomputation step would allow us to apply one of the standard quantum algorithms for the linear systems problem at a cost that is independent of the cost of preparing $|b\rangle$. As we argued above, it is easy to imagine situations where preparing or reflecting about $|b\rangle$ is exponentially more expensive than any other component of the algorithm. For example, we could take $|b\rangle$ to be a brute force encoding of some classical

[3] One can always solve a linear systems problem on a larger space with the Hermitian $\tilde{A} := \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}$ instead of the original A.

data $|x|$ into $n = \text{polylog}(|x|)$ qubits, such that preparing or reflecting about $|b\rangle$ has a complexity that scales polynomially in $|x|$. We could also make the (sometimes reasonable) assumption that the condition number of A and the gate complexity of implementing A (under whatever notion of access is appropriate) scale polynomially in n. Given these two conditions, the complexity of applying any of the standard quantum algorithms for the linear systems problem would be exponentially better in the precomputation cost model than in the standard one (assuming that the target precision is a constant).

Of course, this separation is entirely due to the fact that we discount the cost of preparing the resource state. In fact, in this form of precomputation, the cost of preparing the resource state would be asymptotically larger than the cost of implementing the reflections in the standard way since we require $\tilde{\mathcal{O}}(q^2)$ copies of $|b\rangle$ to implement the reflection R a total of q times with constant error in the overall algorithm. Furthermore, the optimal algorithms for the quantum linear systems problem have a logarithmic dependence on the target precision [12], whereas our approach introduces a polynomial dependence. Additionally, sufficient storage for the copies of $|b\rangle$ would be required. Nevertheless, in a situation where $\vec{b}$ is specified ahead of time and the solution to the problem is sufficiently valuable and time-sensitive, quantum precomputation could prove useful. Note that there is no significant classical cost in terms of storage or computation for this form of precomputation.

It is worth point out that, if one is willing to prepare $\tilde{\mathcal{O}}(\kappa^2)$ copies of $|b\rangle$ ahead of time, there is a simpler strategy to solving the linear systems problem that does not require density matrix exponentiation. However, this simpler strategy is less efficient with respect to the number of times that A must be queried. Consider the original HHL algorithm of Ref. 26. This algorithm requires starting with the state $|b\rangle$ and time-evolving under the Hamiltonian A for a time that scales as $\tilde{\mathcal{O}}(\kappa)$ (to perform phase estimation). This is followed by a postselection step that succeeds with probability $\Omega(1/\kappa^2)$. Normally one uses amplitude amplification to increase the success probability to $\mathcal{O}(1)$.

Instead of using amplitude amplification, one could instead repeatedly prepare the appropriate state and actually perform the postselection based on the output from phase estimation. This would solve the quantum linear systems problem with high probability using a number of copies of $|b\rangle$ that scales as $\tilde{\mathcal{O}}(\kappa^2)$. However, it would also require a total amount of time evolution under A equal to $\tilde{\mathcal{O}}(\kappa^3)$. The approach we proposed above uses a similar number of copies of $|b\rangle$, but the scaling in terms of A (either time evolution under A or a related notion of access) can be made nearly linear with respect to κ by using the optimal algorithms of, e.g., Ref. 12.

4.2 Precomputing Clifford unitaries with gate teleportation

In this subsection, we consider accelerating the task of implementing an n-qubit unitary from the Clifford group using precomputation. We explain how a well-known construction allows for a quadratic savings in gate complexity (when comparing the cost in the precomputation model to the gate complexity in a standard cost model). This construction is a straightforward application of gate teleportation, a technique introduced in Ref. 23 which we illustrate in Figure 1 and review in more detail in Appendix B.2 (along with the definition of the Clifford group). Although this example of quantum precomputation is particularly simple, it provides a good introduction to some of the concerns relevant in the more technically interesting example that we consider in Section 5.

We recall that an arbitrary unitary from the n-qubit Clifford group can be efficiently implemented using one- and two-qubit Clifford gates arranged in a circuit with depth $\mathcal{O}(n)$ [9], leading to a gate complexity of $\mathcal{O}(n^2)$. A counting argument shows that this

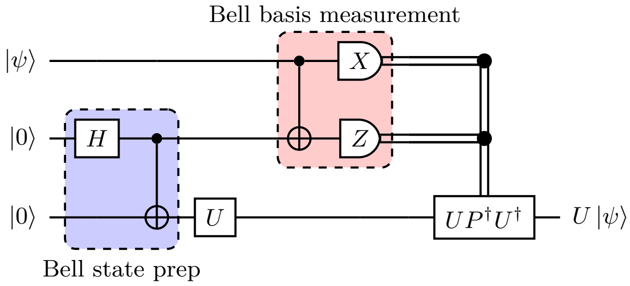


Figure 1: A quantum circuit diagram for the one-qubit version of gate teleportation [23]. The circuit in the blue shaded area prepares a bell pair and the circuit in the red shaded area performs a bell basis measurement (the X/Z in the rounded caps indicate X/Z basis measurements). Based on the outcome of the measurement, a classically controlled operation $UP^\dagger U^\dagger$ is performed, where the "byproduct operator" $P \in \{\mathbb{I}, X, Z, ZX\}$ depends on the measurement outcome. When U is a member of the Clifford group, $UP^\dagger U^\dagger$ is an element of the Pauli group. Single-qubit gate teleportation generalizes naturally to a multi-qubit version. Using multi-qubit gate teleportation to apply an n-qubit unitary from the Clifford group offers a simple example of advantage in the precomputation cost model, reducing the quantum gate complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

asymptotic scaling must be optimal for most elements of the Clifford group. We will show that, in the precomputation cost model, the quantum gate complexity of applying the same unitary is only $\mathcal{O}(n)$.

Let U be an arbitrary unitary in $\mathcal{C}^{(2)}$ (the Clifford group on n qubits) and $|\psi\rangle$ be an arbitrary n-qubit quantum state. Using standard multi-qubit gate teleportation, we can prepare a state $|\Gamma(U)\rangle$ on $2n$ qubits that we can consume to apply U to $|\psi\rangle$ (up to a Pauli correction). This straightforward generalization of the procedure presented in Figure 1 consists of preparing n bell pairs and applying U to a set of n qubits, one taken from each bell pair. Let us consider the steps involved in applying U to $|\psi\rangle$ once $|\Gamma(U)\rangle$ is already prepared. Applying a Clifford unitary using gate teleportation involves making n simultaneous bell-basis measurements of the $3n$-qubit state $|\psi\rangle \otimes |\Gamma(U)\rangle$. The resulting n-qubit state can therefore be obtained in constant depth,

$$|\phi\rangle = UP\,|\psi\rangle\,, \tag{7}$$

where P is the "byproduct operator," a member of the Pauli group that is determined by the measurement outcomes. By the definition of the Clifford group, the correction operator $UP^\dagger U^\dagger$ is also a Pauli operator (up to a possible phase) and can therefore be applied in constant depth to yield the desired state $U|\psi\rangle$. The overall quantum circuit complexity (neglecting the cost of preparing $|\Gamma(U)\rangle$) is therefore $\mathcal{O}(n)$, in contrast with the $\mathcal{O}(n^2)$ cost of applying U without precomputation.

Although we are primarily concerned with the quantum gate complexity of applying U given $|\Gamma(U)\rangle$, we may also wish to consider the classical computational costs of determining which of the 4^n possible correction operators to apply once the measurement outcomes are known. We need to use $2n$ bits initially to store the results of the bell basis measurement that determines the byproduct operator. We could store a classical description of the $\mathcal{O}(n^2)$ Clifford gates in U and apply them to the byproduct operator. This would require $\mathcal{O}(n^2)$ operations (updating a constant number of the $\mathcal{O}(n)$ stored bits each time we conjugate by a gate in the circuit) which could be performed in $\mathcal{O}(n)$ sequential steps by parallelizing across gates in the same layer of the circuit.

We can reduce the depth of the classical computation (although not the overall number of operations) by factorizing the correction operator ahead of time,

$$UP^\dagger U^\dagger = U\left(\bigotimes_{i=1}^{n} X_i^{x_i} Z_i^{z_i}\right)U^\dagger = \left(\prod_{i=1}^{n} UX_i^{x_i}U^\dagger\right)\left(\prod_{i=1}^{n} UZ_i^{z_i}U^\dagger\right), \tag{8}$$

where the x_i and z_i are determined by the measurement outcomes of bell basis measurement. This allows us to classically precompute each of the $2n$ Pauli operators of the form $UX_iU^\dagger$ or $UZ_iU^\dagger$ and store the results using $\mathcal{O}(n^2)$ bits. Once we know the measurement results, we can multiply the appropriate operators together in logarithmic depth using a divide and conquer strategy, ultimately computing the final correction operator using $\mathcal{O}(n^2)$ operations using $\mathcal{O}(\log(n))$ sequential steps (neglecting the classical cost of the precomputation).

5 Precomputing diagonal unitaries in the Clifford hierarchy with selective gate teleportation

In this section, we show how a more sophisticated form of gate teleportation introduced in Ref. 18 can be used to construct a precomputation protocol for a set of diagonal unitaries in the Clifford hierarchy (reviewed Appendix B.2). We graphically illustrate this selective gate teleportation in Figure 2 and present a more substantial review in Appendix B.3. In Section 4.2, we considered a simple example of quantum precomputation that uses standard gate teleportation to apply some $U \in \mathcal{C}^{(2)}$ (the Clifford group). We explained how the $\mathcal{O}(n^2)$ gate complexity required to implement an arbitrary n-qubit Clifford unitary can be reduced to $\mathcal{O}(n)$ in the precomputation model. The approach is less straightforward, but the generalization that we present in this section achieves the same quadratic compression for a subset of unitaries from higher levels of the Clifford hierarchy. In other words, we show that unitaries from the family $\mathcal{Z}^{(k)}$, defined below, that have a gate complexity of $\tilde{\Theta}(n^k)$ when implemented directly can be implemented with a gate complexity of $\tilde{\mathcal{O}}(kn^{k/2})$ in the precomputation cost model (assuming k is even for simplicity). The basic strategy we use is to apply such a unitary with gate teleportation and then use a series of selective gate teleportation steps to apply the correction operator up to some simpler correction that can be implemented directly.

Before we present our actual proposal, let us consider a naive generalization, where we use gate teleportation to implement some $U \in \mathcal{C}^{(3)}$ (the third level of the Clifford hierarchy). By definition, the correction operator required will be some $R \in \mathcal{C}^{(2)}$. Applying R directly would result in an overall gate complexity of $\mathcal{O}(n^2)$, essentially saving a factor of n compared to the cost of implementing U directly, which is $\Omega(n^3)$ by a counting argument. For a general $U \in \mathcal{C}^{(k)}$, it is unclear if it is possible to obtain an advantage greater than a factor of n in the precomputation model.

However, if we restrict ourselves to considering a smaller set of unitaries, we can do better. Rather than allowing for arbitrary elements of the Clifford hierarchy, we limit ourselves to considering elements of the hierarchy that are also diagonal. To simplify the presentation, we actually restrict ourselves even further in this section, considering only those gates in $\mathcal{C}^{(k)}$ that are composed of products of $\pm\mathbb{I}$, Pauli Z operators, and controlled Z operators with up to $k-1$ controls.[4] We denote this set $\mathcal{Z}^{(k)}$ and in Appendix C, we

[4] The only property of $\mathcal{Z}^{(k)}$ that we leverage, other than the fact that $\mathcal{Z}^{(k)} \subset \mathcal{C}^{(k)}$, is that it forms an Abelian group. This is also true of the full set of diagonal unitaries at each level in the Clifford hierarchy, which suggests that our results may readily generalize to this case.

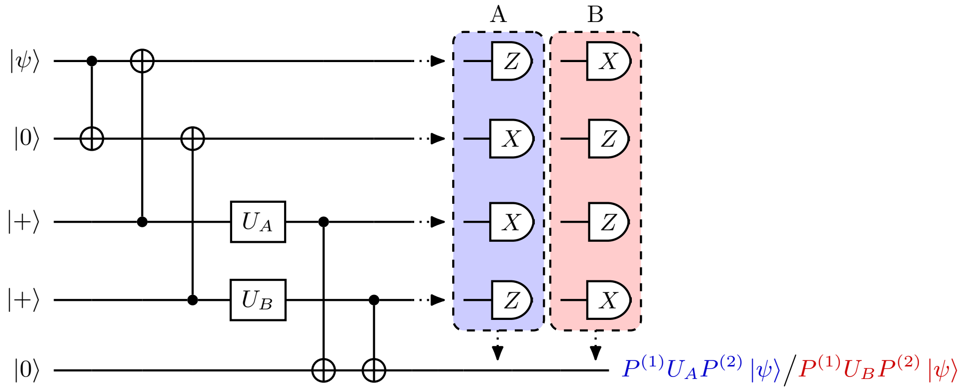


Figure 2: A circuit diagram for the one-qubit version of selective gate teleportation [18]. This protocol allows for the teleportation of a choice of unitaries, U_A or U_B, onto an input state. Which unitary is teleported is controlled by the measurement settings (the four ancilla qubits are each measured in the X or Z basis according to the proscriptions shown in the blue and red shaded areas of the diagram). The possible states of the output qubit are color-coded to match the measurement settings that select for them. In our use of selective teleportation, we take $U_A = U$ and $U_B = \mathbb{I}$. Byproduct operators $P^{(1)}$ and $P^{(2)}$ from the set $\{\mathbb{I}, X, Z, XZ\}$ are randomly applied before and after the selected unitary based on the measurement outcomes.

show that it forms a group. We also note that $\mathcal{Z}^{(j)}$ is a proper subgroup of $\mathcal{Z}^{(k)}$ for $j < k$ and prove the following proposition:

Proposition 1. Consider a gate $G \in \mathcal{Z}^{(k)}$ and a product of single-qubit Pauli X operators that we denote by $X_{\boldsymbol{s}}$ (where $\boldsymbol{s} \in [n]$ indicates the indices of the qubits where $X_{\boldsymbol{s}}$ acts non-trivially). Define G' in the following way,

$$G' := X_{\boldsymbol{s}} G X_{\boldsymbol{s}} G^{\dagger}. \tag{9}$$

Then $G' \in \mathcal{Z}^{(k-1)}$ if $k > 1$ and $G' = \pm\mathbb{I}$ if $k \in \{0, 1\}$. As a corollary, we also have that

$$G X_{\boldsymbol{s}} = X_{\boldsymbol{s}} G' G. \tag{10}$$

Diagonal unitaries commute, and the elements of $\mathcal{Z}^{(k)}$ are all self-inverse. As a result, we can specify a $U \in \mathcal{Z}^{(k)}$ using exactly

$$\sum_{j=0}^{k} \binom{n}{j} = \mathcal{O}(n^k) \tag{11}$$

bits, one to specify the sign and one to specify the presence or absence of each possible $C^{j-1}Z$ gate for each $j \in [1..n]$. A $C^{j-1}Z$ gate can be implemented using $\mathcal{O}(j)$ T gates in depth $\mathcal{O}(\log j)$ [39]. An arbitrary gate $G \in \mathcal{Z}^{(k)}$ can therefore be implemented in depth $\tilde{\mathcal{O}}(n^{k-1})$ and gate complexity $\tilde{\mathcal{O}}(n^k)$, even under reasonable assumptions about qubit connectivity [41]. Furthermore, by counting the number of distinct elements of $\mathcal{Z}^{(k)}$, we can also see that a typical element must have a circuit complexity lower bounded by $\Omega(n^k)$.

We begin our construction by preparing the usual $2n$ qubit resource state for applying the gate $U \in \mathcal{Z}^{(k)}$ using teleportation. If this state were used directly for gate teleportation, we would need to perform a correction of the form $UP^{\dagger}U^{\dagger}$ for some n-qubit byproduct operator P (which we can write as a product of single-qubit X and Z operators). We will perform this correction using selective teleportation. Note that we can neglect the

Z corrections (as they can be trivially commuted to the end of the circuit up to a sign). Factorizing the X component of the corrections, we see that we need to apply the unitary

$$R = \prod_{i=1}^{n} U X_i^{x_i} U^\dagger, \tag{12}$$

where the bits x_i will be chosen based on the measurement outcomes of first gate teleportation step. It is convenient to rewrite each of the terms in the product as

$$U X_i^{x_i} U^\dagger = X_i^{x_i} \left(X_i^{x_i} U X_i^{x_i} U^\dagger \right), \tag{13}$$

i.e., a product of $X_i^{x_i}$ and an operator that is in $\mathcal{Z}^{(k-1)}$ by Proposition 1.

We can use selective gate teleportation to apply the diagonal term $(X_i^{x_i} U X_i^{x_i} U^\dagger)$ from each of the n possible factors of the correction operator. Note that we can do this after applying U to the n bell pairs and before performing the bell basis measurement that completes the gate teleportation. We ignore the $X_i^{x_i}$ terms that precede the diagonal components of the factors of the correction operator in Equation (13) because we can absorb them into the byproduct operators that will arise anyway from the selective teleportation. For each of the correction operators, we need $4n$ additional qubits to implement the selective gate teleportation, so the overall overhead is $4n^2$. When we attempt to use selective teleportation in this way to implement the correction operator, we will actually end up implementing the operator

$$\tilde{R} = P^{(0)} \prod_{i=1}^{n} \left(X_i^{x_i} U X_i^{x_i} U^\dagger P^{(i)} \right), \tag{14}$$

where the $P^{(i)}$ terms represent randomly obtained products of Pauli operators and the $X_i^{x_i} U X_i^{x_i} U^\dagger$ are elements of $\mathcal{Z}^{(k-1)}$. Notice that we can commute the Pauli terms to the left at the cost of requiring a series of corrections $R^{(i)'} \in \mathcal{Z}^{(k-2)}$.

We can proceed recursively. We factored the one byproduct operator to obtain n possible factors of the correction operator, each of which we applied using selective gate teleportation. Implementing these corrections required a total of $4n^2$ additional ancilla qubits and resulted in the addition of Pauli byproduct operators at $n+1$ locations. We can commute these byproduct operators through to the left, starting at the righthand side of our expression. Each time we commute an n-qubit operator of the form $\prod_{i=1}^{n} X_i^{x_i}$ through a diagonal gate we do so by factorizing it and we pick up n possible correction terms one level lower in the $\mathcal{Z}^{(k)}$ hierarchy. The number of corrections that we must apply, and the number of additional ancilla qubits that we require, therefore increases by a factor of n each time we descend the hierarchy by a level. For example, we can use $\mathcal{O}(n^3)$ ancilla qubits to implement each of the n^2 possible second-order corrections using selective teleportation, leaving only corrections that are three or more levels down the hierarchy. More generally, to implement $U \in \mathcal{Z}^{(k)}$ up to a correction $R \in \mathcal{Z}^{(k-a)}$ (and some Pauli X operators), we require a resource state on $\mathcal{O}(n^a)$ qubits.

If we were to descend the hierarchy all the way to the point where the only remaining corrections were Pauli corrections ($a = k-1$), we would obtain only a modest compression in circuit complexity (compared with directly applying U). This is because, although the circuit depth would be merely $\mathcal{O}(k)$, we would require $\mathcal{O}(n^{k-1})$ qubits. However, consider what happens when we stop at the level $a = \lfloor k/2 \rfloor$. To simplify the presentation we assume that k is even. We can use a resource state on $\mathcal{O}(n^{k/2})$ qubits to implement U up to a correction $R \in \mathcal{Z}^{(k/2)}$ (and some additional Pauli terms) in $k/2$ rounds of measurement.

We can implement the remaining correction directly with a gate complexity of $\tilde{\mathcal{O}}(kn^{k/2})$ in depth $\tilde{\mathcal{O}}(k)$ with no additional space overhead using the constant depth fanout and unfanout circuits of Ref. 42. Therefore, the overall gate complexity of implementing an arbitrary $U \in \mathcal{Z}^{(k)}$ in the precomputation model (i.e., neglecting the cost of preparing the resource state) is $\tilde{\mathcal{O}}(kn^{k/2})$.

Recall that a fanout operation takes an n-qubit state $|\psi\rangle$ and performs the map

$$|\psi\rangle = \sum_{i=1}^{2^n} c_i |i\rangle \rightarrow \sum_{i=1}^{2^n} c_i |i\rangle^{\otimes m} \tag{15}$$

for some integer $m > 1$, where the states in $\{|i\rangle\}$ are the computational basis states. Unfanout reverses this mapping. Ref. 42 explains how both of these operations can be implemented using constant depth quantum circuits and classical feedback. We can parallelize the implementation of m diagonal unitaries by performing a fanout, applying each unitary to a separate fanned out copy of $|\psi\rangle$, and then performing an unfanout.

We can take advantage of this capability by partitioning the individual terms that make up an arbitrary $R \in \mathcal{Z}^{(k/2)}$ into $\mathcal{O}(k)$ sets of gates, where each set contains only terms that act on disjoint qubits. By setting $m = n^{k/2-1}$, we can apply the terms from each of the sets in parallel. We can therefore apply all of the terms with the desired gate complexity and depth. Because the fanout and unfanout operations are constant depth, they do not increase the asymptotic scaling of the gate complexity. The remaining Pauli correction can then be applied to complete the implementation of U.

Now let us consider the classical computational cost associated with applying U this way in the precomputation model. Applying U up to a correction at level $\mathcal{Z}^{k-a}$ is trivial for $a = 1$. For $a = 2$, we apply some subset of the n possible corrections that corresponds directly to the bits we obtained from the first set of measurements. For $a = 3$, we need to repeatedly XOR one n bit string into another $\mathcal{O}(n)$ times in order to determine the measurement settings, using $\mathcal{O}(n^2)$ classical operations. This growth continues, and we find that we need to perform $\mathcal{O}(n^{k/2-1})$ classical operations to determine which corrections to perform at the level that leaves us with a final correction in $\mathcal{Z}^{k/2}$. Actually computing the final correction $R \in \mathcal{Z}^{k/2}$ requires determining the $\mathcal{O}(n^{k/2})$ elements of $\mathcal{Z}^{k/2}$ that arise from commuting the byproduct operators through and then taking their product, which ultimately takes $\mathcal{O}(n^k)$ XOR operations. The classical postprocessing involved in the fanout operation is negligible compared to these costs, so the overall classical complexity is $\mathcal{O}(n^k)$.

We can also ask about the quantum and classical complexities of performing the precomputation step. Neglecting the operations involved in setting up the teleportation and selective teleportation gadgets themselves since they contribute negligibly to the overall complexity, we can just consider the gate complexities of performing one operation from $\mathcal{Z}^{(k)}$, n operations from $\mathcal{Z}^{(k-1)}$, and so on, down to $n^{k/2-1}$ operations at the level $\mathcal{Z}^{(k/2+1)}$. The only clear way to apply these operations is to work serially (since the use of selective teleportation may prevent us from using fanout and unfanout operations to parallelize). This means that, although we only require $\tilde{\mathcal{O}}(kn^k)$ non-identity gates, our definition of gate complexity (which attempts to account for storage space by counting the single-qubit identity operation as a gate) implies that the overall gate complexity of the precomputation step is $\tilde{\mathcal{O}}(kn^{3k/2})$. This may not be a fundamental requirement, and it is also true that most of the $\mathcal{O}(n^{k/2})$ qubits are not required at all until the very last portions of the precomputation step, so they could be used for other things in the meantime. The classical complexity of the precomputation step arises from computing the various correction operators and is not substantially larger than would be expected from the need to generate some kind of classical description of the circuits involved anyway.

In many ways, the techniques of this section are a generalization of the simpler scheme for applying Clifford operators using gate teleportation that we presented in Section 4.2. In order to make a comparison easy, we summarize the various scalings of these two examples of quantum precomputation in Table 1.

	Typical $U \in \mathcal{C}^{(2)}$ (Sec. 4.2)	Typical $U \in \mathcal{Z}^{(k)}$ (Sec. 5)
Gate complexity, standard	$\Theta(n^2)$	$\tilde{\Theta}(n^k)$
Gate complexity, precomputation	$\mathcal{O}(n)$	$\tilde{\mathcal{O}}(kn^{k/2})$
Resource state size	$\mathcal{O}(n)$	$\mathcal{O}(n^{k/2})$
Gate complexity, preparing $\lvert\Gamma(U)\rangle$	$\mathcal{O}(n^2)$	$\tilde{\mathcal{O}}(kn^{3k/2})$
Classical operations, consuming $\lvert\Gamma(U)\rangle$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^k)$

Table 1: A summary of the scalings for applying arbitrary Clifford operators using gate teleportation (Section 4.2) and arbitrary elements of $\mathcal{Z}^k$ (products of Z and controlled Z operators with up to $k-1$ controls) using selective gate teleportation (Section 5). For simplicity we assume that k is even. For the gate complexity, we count the number of one- and two-qubit gates from the Clifford + T gate set (counting single-qubit identity operations as gates). The quoted gate complexity in the precomputation model includes only those quantum operations required to consume the resource state $\lvert\Gamma(U)\rangle$. The (quantum) cost of preparing the resource state is provided separately, as is the number of classical operations required to consume the resource state to apply U.

6 Discussion

In this paper, we introduced a new cost model for quantum computation that allows for "quantum precomputation." This model is motivated by practical scenarios where it is highly valuable to perform a time-sensitive computation as quickly as possible, and where some portion of the problem's input is naturally known ahead of time. In the precomputation cost model, we allow a reasonable (polynomial in the input size) amount of effort to be spent "for free" preparing a resource state before the input is fully specified. The cost of an algorithm in the precomputation cost model is determined solely by the resources required to implement the algorithm given access to the resource state. We presented three realizations of quantum precomputation that require asymptotically fewer resources in the precomputation cost model than in a standard one.

The first realization uses density matrix exponentiation to implement reflections about a state by consuming copies of that state. We explained how, in some cases, this type of quantum precomputation can offer an exponential advantage (in the sense that the complexity required to execute an algorithm by consuming the resource state can be exponentially smaller than the complexity required to execute an algorithm directly). As a particular example, we considered the task of accelerating quantum algorithms for linear systems in cases where it is natural to prepare copies of the state $\lvert b\rangle$ ahead of time. In the future, we hope to find practical examples where this type of precomputation is useful, either for solving particular linear systems of equations, or for executing some other quantum algorithm whose cost might be dominated by the cost of implementing low-rank reflections. In practice, the advantage need not be exponential to be useful. It would be especially interesting if we could find situations where the ability to accelerate an algorithm using precomputation was the deciding factor that made it worth solving a particular problem using quantum rather than classical computation.

As a second example, we pointed out that standard techniques for implementing Clifford unitaries using gate teleportation constitute a simple illustration of an asymptotic advantage in the precomputation cost model. These techniques allow for unitaries with a gate complexity of $\Theta(n^2)$ to be implemented in $\mathcal{O}(1)$ (quantum gate) depth by consuming a state on $2n$ qubits. This example highlights the importance of choosing an appropriate notion of cost when defining a model of quantum precomputation. Under a definition of cost that treated Clifford operations as free, there could be no value in using precomputation to apply a Clifford unitary more efficiently. However, as schemes for magic state distillation continue to improve, it is becoming less clear if quantifying the cost of a fault-tolerant quantum algorithm solely in terms of the number of non-Clifford gates is an accurate approximation [34]. This motivated our particular definition of a precomputation cost model (that counts gate complexity, including Clifford gates), but it is possible that a metric of cost even closer to the hardware might be more appropriate. For instance, one could imagine squeezing some additional benefit out of a scheme for quantum precomputation by preparing the resource states using shorter distance error correcting codes (and therefore, fewer physical qubits and less actual time) in conjunction with error detection and postselection.

Even within the particular cost model we have defined, there are many degrees of freedom to explore in defining precomputation protocols. For example, the technique we used to implement an arbitrary Clifford unitary could be modified to apply an n-qubit circuit U that interleaved Clifford operations with a small number (t) of T gates. Such a modified scheme could use a combination of gate teleportation and selective teleportation to apply the Clifford gates as normal, while selectively implement the possible corrections after each T gate. This would require an $\mathcal{O}(n + t)$ qubit resource state that would be consumed in $\mathcal{O}(t)$ rounds of measurement to apply U up to a final Pauli correction.

The most novel example of precomputation that we proposed in this paper uses selective teleportation to achieve a quadratic reduction in the complexity of implementing a family of diagonal unitaries from the Clifford hierarchy (when comparing the cost in the precomputation model with the standard cost). Our scheme is likely generalizable to all diagonal unitaries that are members of the Clifford hierarchy, but this is still a relatively restricted class of unitaries. This naturally raises the question, are there ways to compile existing algorithms such that they would make heavy use of the kinds of diagonal unitaries that have shown can be accelerated by precomputation? Diagonal unitaries appear in a variety of places, oftentimes as a natural way of encoding the output of a classical function into a phase. For example, the Forrelation problem [2], IQP circuits [45], QAOA [17], and Grover's algorithm itself [25], can all be formulated to involve heavy use of diagonal unitaries. In the future, we hope that extensions of our precomputation protocols can be used to accelerate some such algorithms for interesting and time-sensitive applications.

More broadly, does quantum precomputation have anything to teach us about the nature or power of quantum computation? The power of advice (computation supplemented by a resource state) has been studied both in classical and quantum contexts [1, 30], but, as we discuss in Section 2, the precomputation model we introduced differs from these prior works in that we require that the extra resource state be efficient to prepare. In this finer-grained setting, what can we say about the difference between quantum and classical computation? Are there classical analogues of the kinds of quantum precomputation that we have proposed, or are there some types of precomputation are uniquely quantum mechanical? Conversely, classical precomputation is widely applicable in situations where the precomputed information is used multiple times. One could interpret recent shadow tomography proposals as examples of quantum precomputation that allow for information reuse [3, 11], and it would be interesting to see if techniques from that domain can be adapted to enable such reuse in the context of other types of quantum precomputation.

Finally, are there other, perhaps more general, classes of quantum computation that we can accelerate in the precomputation cost model? Many proposed applications of quantum machine learning techniques to classical data rely on quantum random access memory (QRAM) to obtain a computational advantage [7]. Are there real-world applications where it would be natural to circumvent the need for QRAM by encoding some classical data into quantum states ahead of time?

Acknowledgements

The authors thank Rolando Somma, Robin Kothari, Nathan Wiebe, Craig Gidney, Yigit Subasi, Ryan Babbush, and others for helpful discussions and feedback.

References

[1] S Aaronson. Limitations of quantum advice and one-way communication. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004*, pages 320–332. IEEE, 2004. ISBN 9780769521206. DOI: 10.1109/ccc.2004.1313854.

[2] Scott Aaronson and Andris Ambainis. Forrelation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, STOC '15, pages 307–316, New York, NY, USA, 14 June 2015. ACM. ISBN 9781450335362. DOI: 10.1145/2746539.2746547.

[3] Scott Aaronson and Guy N Rothblum. Gentle measurement of quantum states and differential privacy. 18 April 2019. URL http://arxiv.org/abs/1904.08747.

[4] Ryan Babbush, Jarrod R McClean, Michael Newman, Craig Gidney, Sergio Boixo, and Hartmut Neven. Focus beyond quadratic speedups for error-corrected quantum advantage. *PRX quantum*, 2(1):010103, 29 March 2021. ISSN 2691-3399. DOI: 10.1103/prxquantum.2.010103.

[5] Daniel J Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In *Advances in Cryptology - ASIACRYPT 2013*, Lecture notes in computer science, pages 321–340. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 9783642420443,9783642420450. DOI: 10.1007/978-3-642-42045-0“‘17.

[6] Dominic W Berry, Craig Gidney, Mario Motta, Jarrod R McClean, and Ryan Babbush. Qubitization of arbitrary basis quantum chemistry leveraging sparsity and low rank factorization. 6 February 2019. URL http://arxiv.org/abs/1902.02134.

[7] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, September 2017. ISSN 0028-0836,1476-4687. DOI: 10.1038/nature23474.

[8] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Phys. Rev. A*, 71(2):022316, 22 February 2005. ISSN 1050-2947,1094-1622. DOI: 10.1103/physreva.71.022316.

[9] Sergey Bravyi and Dmitri Maslov. Hadamard-free circuits expose the structure of the clifford group. *IEEE Trans. Inf. Theory*, 67(7):4546–4563, July 2021. ISSN 0018-9448,1557-9654. DOI: 10.1109/tit.2021.3081415.

[10] Earl T Campbell and Joe O'Gorman. An efficient magic state approach to small angle rotations. 14 March 2016. URL http://arxiv.org/abs/1603.04230.

[11] Sitan Chen, Jordan Cotler, Hsin-Yuan Huang, and Jerry Li. Exponential separations between learning with and without quantum memory. In *2021 IEEE 62nd*

Annual Symposium on Foundations of Computer Science (FOCS). IEEE, February 2022. DOI: 10.1109/focs52979.2021.00063.

[12] Andrew M Childs, Robin Kothari, and Rolando D Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM J. Comput.*, 46(6):1920–1950, 1 January 2017. ISSN 0097-5397. DOI: 10.1137/16M1087072.

[13] N Cody Jones, James D Whitfield, Peter L McMahon, Man-Hong Yung, Rodney Van Meter, Alán Aspuru-Guzik, and Yoshihisa Yamamoto. Faster quantum chemistry simulation on fault-tolerant quantum computers. *New J. Phys.*, 14(11):115023, 27 November 2012. ISSN 1367-2630. DOI: 10.1088/1367-2630/14/11/115023.

[14] Pedro C S Costa, Dong An, Yuval R Sanders, Yuan Su, Ryan Babbush, and Dominic W Berry. Optimal scaling quantum linear-systems solver via discrete adiabatic theorem. *PRX quantum*, 3(4):040303, 7 October 2022. ISSN 2691-3399. DOI: 10.1103/prxquantum.3.040303.

[15] Jordan Cotler, Hsin-Yuan Huang, and Jarrod R McClean. Revisiting dequantization and quantum advantage in learning tasks. 1 December 2021. URL http://arxiv.org/abs/2112.00811.

[16] Shawn X Cui, Daniel Gottesman, and Anirudh Krishna. Diagonal gates in the clifford hierarchy. *Phys. Rev. A*, 95(1), 26 January 2017. ISSN 2469-9926,2469-9934. DOI: 10.1103/physreva.95.012329.

[17] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. 14 November 2014. URL http://arxiv.org/abs/1411.4028.

[18] Austin G Fowler. Time-optimal quantum computation. 17 October 2012. URL http://arxiv.org/abs/1210.4626.

[19] Sevag Gharibian and François Le Gall. Dequantizing the quantum singular value transformation: hardness and applications to quantum chemistry and the quantum PCP conjecture. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 19–32, New York, NY, USA, 9 June 2022. ACM. ISBN 9781450392648. DOI: 10.1145/3519935.3519991.

[20] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5(433):433, 15 April 2021. ISSN 2521-327X. DOI: 10.22331/q-2021-04-15-433.

[21] Craig Gidney and Austin G Fowler. Flexible layout of surface code computations using AutoCCZ states. 21 May 2019. URL http://arxiv.org/abs/1905.08916.

[22] András Gilyén and Alexander Poremba. Improved quantum algorithms for fidelity estimation. 29 March 2022. URL http://arxiv.org/abs/2203.15993.

[23] Daniel Gottesman and Isaac L Chuang. Quantum teleportation is a universal computational primitive. 2 August 1999. URL http://arxiv.org/abs/quant-ph/9908010.

[24] Leo Grady and Ali Kemal Sinop. Fast approximate random walker segmentation using eigenvector precomputation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2008. ISBN 9781424422425. DOI: 10.1109/cvpr.2008.4587487.

[25] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, STOC '96, pages 212–219, New York, New York, USA, 1996. ACM Press. ISBN 9780897917858. DOI: 10.1145/237814.237866.

[26] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103(15):150502, 9 October 2009. ISSN 0031-9007,1079-7114. DOI: 10.1103/PhysRevLett.103.150502.

[27] Hsin-Yuan Huang, Michael Broughton, Jordan Cotler, Sitan Chen, Jerry Li, Masoud Mohseni, Hartmut Neven, Ryan Babbush, Richard Kueng, John Preskill, and Jarrod R McClean. Quantum advantage in learning from experiments. *Science*, 376(6598):1182–1186, 10 June 2022. ISSN 0036-8075,1095-9203. DOI: 10.1126/science.abn7293.

[28] Cody Jones. Distillation protocols for fourier states in quantum computing. 12 March 2013. URL http://arxiv.org/abs/1303.3066.

[29] John Kallaugher. A quantum advantage for a natural streaming problem. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 897–908. IEEE, February 2022. DOI: 10.1109/focs52979.2021.00091.

[30] Richard M Karp and Richard J Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual ACM symposium on Theory of computing - STOC '80*, STOC '80, pages 302–309, New York, New York, USA, 28 April 1980. ACM Press. ISBN 9780897910170. DOI: 10.1145/800141.804678.

[31] Shelby Kimmel, Cedric Yen-Yu Lin, Guang Hao Low, Maris Ozols, and Theodore J Yoder. Hamiltonian simulation with optimal sample complexity. *Npj Quantum Inf.*, 3 (1):1–7, 30 March 2017. ISSN 2056-6387,2056-6387. DOI: 10.1038/s41534-017-0013-7.

[32] François Le Gall. Exponential separation of quantum and classical online space complexity. In *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, SPAA '06, pages 67–73, New York, NY, USA, 30 July 2006. ACM. ISBN 9781595934529. DOI: 10.1145/1148109.1148119.

[33] Lin Lin and Yu Tong. Optimal polynomial based quantum eigenstate filtering with application to solving quantum linear systems. *Quantum*, 4(361):361, 11 November 2020. ISSN 2521-327X. DOI: 10.22331/q-2020-11-11-361.

[34] Daniel Litinski. Magic state distillation: Not as costly as you think. *Quantum*, 3 (205):205, 2 December 2019. ISSN 2521-327X. DOI: 10.22331/q-2019-12-02-205.

[35] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, 3(128):128, 5 March 2019. ISSN 2521-327X. DOI: 10.22331/q-2019-03-05-128.

[36] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nat. Phys.*, 10(9):631–633, 27 September 2014. ISSN 1745-2473,1745-2481. DOI: 10.1038/nphys3029.

[37] John M Martyn, Zane M Rossi, Andrew K Tan, and Isaac L Chuang. Grand unification of quantum algorithms. *PRX quantum*, 2(4):040203, 3 December 2021. ISSN 2691-3399. DOI: 10.1103/prxquantum.2.040203.

[38] Iman Marvian and Seth Lloyd. Universal quantum emulator. 8 June 2016. URL http://arxiv.org/abs/1606.02734.

[39] F Motzoi, M P Kaicher, and F K Wilhelm. Linear and logarithmic time compositions of quantum many-body operators. *Phys. Rev. Lett.*, 119(16):160503, 20 October 2017. ISSN 0031-9007,1079-7114. DOI: 10.1103/PhysRevLett.119.160503.

[40] Michael A Nielsen. Optical quantum computation using cluster states. *Phys. Rev. Lett.*, 93(4):040503, 23 July 2004. ISSN 0031-9007,1079-7114. DOI: 10.1103/PhysRevLett.93.040503.

[41] Bryan O'Gorman, William J Huggins, Eleanor G Rieffel, and K Birgitta Whaley. Generalized swap networks for near-term quantum computing. 13 May 2019. URL http://arxiv.org/abs/1905.05118.

[42] Paul Pham and Krysta M Svore. A 2D nearest-neighbor quantum architecture for factoring in polylogarithmic depth. 27 July 2012. URL http://arxiv.org/abs/1207.6655.

[43] R Raussendorf and H J Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86(22):5188–5191, 28 May 2001. ISSN 0031-9007,1079-7114. DOI: 10.1103/PhysRevLett.86.5188.

[44] Yuval R Sanders, Dominic W Berry, Pedro C S Costa, Louis W Tessler, Nathan Wiebe, Craig Gidney, Hartmut Neven, and Ryan Babbush. Compilation of fault-tolerant quantum heuristics for combinatorial optimization. *PRX quantum*, 1(2): 020312, 9 November 2020. ISSN 2691-3399. DOI: 10.1103/prxquantum.1.020312.

[45] Dan Shepherd and Michael J Bremner. Temporally unstructured quantum computation. *Proc. Math. Phys. Eng. Sci.*, 465(2105):1413–1439, 8 May 2009. ISSN 1364-5021,1471-2946. DOI: 10.1098/rspa.2008.0443.

[46] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 527–536, New York, NY, USA, 1 July 2002. ACM. ISBN 9781581135213. DOI: 10.1145/566570.566612.

[47] James E Smith. A study of branch prediction strategies. In *25 years of the international symposia on Computer architecture (selected papers)*, ISCA '98, pages 202–215, New York, NY, USA, 1 August 1998. ACM. ISBN 9781581130584. DOI: 10.1145/285930.285980.

[48] Rolando D Somma and Yiğit Subaşı. Complexity of quantum state verification in the quantum linear systems problem. *PRX quantum*, 2(1):010315, 27 January 2021. ISSN 2691-3399. DOI: 10.1103/prxquantum.2.010315.

[49] Barbara M Terhal. Quantum error correction for quantum memories. *Rev. Mod. Phys.*, 87(2):307–346, 7 April 2015. ISSN 0034-6861,1539-0756. DOI: 10.1103/revmodphys.87.307.

[50] Xinlan Zhou, Debbie W Leung, and Isaac L Chuang. Methodology for quantum logic gate construction. *Phys. Rev. A*, 62(5), 18 October 2000. ISSN 1050-2947,1094-1622. DOI: 10.1103/physreva.62.052316.

A Precomputation and quantum advice

The purpose of this appendix is to relate our proposed model of quantum precomputation to the notion of quantum advice and the complexity class BQP/qpoly. We do not aim to provide a self-contained introduction to quantum complexity theory, but we will briefly mention some basic definitions that will aid in making the comparison. The most well-studied computational problems in complexity theory are decision problems, questions that have a yes or no answer. We can formalize a decision problem as a language, a set of bitstrings that encode the inputs to the problem for which the answer is yes. Informally, a decision problem is in the complexity class BQP if it can be solved in polynomial time on a quantum computer. Formally, we have the following definition:

Definition A.1. Let $\{0,1\}^*$ denote the set of all binary strings. A language $L \subseteq \{0,1\}^*$ is in BQP if these exists a uniform family of polynomial-size quantum circuits, $\{C_n\}$, such that the following conditions hold for all $x \in \{0,1\}^n$:

1. If $x \in L$, then the probability that the first qubit is measured to be $|1\rangle$ after C_n is applied to the input $|x\rangle \otimes |0 \cdots 0\rangle$ is at least $2/3$.

2. If $x \notin L$, then the probability that the first qubit is measured to be $|1\rangle$ after C_n is applied to the input $|x\rangle \otimes |0 \cdots 0\rangle$ is at most $1/3$.

Note that the circuit C_n depends only on n, the size of the input. The condition that the family of circuits is uniform essentially requires that a polynomial time classical computer can generate the description of the circuit that the quantum computer will execute.

Like our model of quantum precomputation, the complexity class BQP/qpoly is intended to capture the power of a polynomial-time quantum machine augmented with an additional resource state. Formally, the class can be defined as follows:

Definition A.2. A language $L \subseteq \{0,1\}^*$ is in BQP/qpoly if there exists a uniform family of polynomial-size quantum circuits, $\{C_n\}$, and a family of polynomial-size quantum states, $\{|\psi_n\rangle\}$, such that the following conditions hold for all $x \in \{0,1\}^n$:

1. If $x \in L$, then the probability that the first qubit is measured to be $|1\rangle$ after C_n is applied to the input $|x\rangle \otimes |0 \cdots 0\rangle \otimes |\psi_n\rangle$ is at least 2/3.

2. If $x \notin L$, then the probability that the first qubit is measured to be $|1\rangle$ after C_n is applied to the input $|x\rangle \otimes |0 \cdots 0\rangle \otimes |\psi_n\rangle$ is at most 1/3.

It is important to note that the additional quantum resources afforded to the polynomially powerful quantum machine can be arbitrarily complex states on poly(n) qubits. However, these states are only allowed to depend on the size of the input.

There are therefore three key differences between the model of computation considered in BQP/qpoly and the model we consider when we allow for “free” polynomial-time quantum precomputation. First of all, we have defined quantum precomputation to allow inputs and outputs that are combinations of classical and quantum information. BQP/qpoly is concerned with machines that take a classical bitstring as an input and return (with some probability of failure) a single classical bit as output. Secondly, in the precomputation cost model, we require that the quantum resources states are preparable in polynomial time, whereas the quantum advice states allowed in BQP/qpoly can be arbitrary quantum states. Finally, in the precomputation model, we partition the input into two subsets and allow for the resource state to depend on one subset, but not the other. The complexity class BQP/qpoly only allows for the resource states to depend on the size of the input, but none of its other features.

B Algorithmic Primitives

B.1 Density matrix exponentiation

Density matrix exponentiation is a technique that allows one to consume copies of a mixed quantum state ρ in order to approximately implement the unitary $e^{-it\rho}$ [36]. In Ref. 36, Lloyd et al. gave a protocol for implementing $e^{-it\rho}$ to within an error ϵ (in the diamond norm) by consuming

$$m = \mathcal{O}(t^2/\epsilon) \tag{16}$$

copies of ρ. This scaling is optimal with respect to ϵ, and optimal with respect to t for general ρ (but not necessarily for pure states) [31]. Furthermore, the protocol is relatively simple to implement. In order to act on an input state σ, one repeatedly consumes a single copy of ρ to apply an approximation to $e^{it\rho/m}$. This is done by performing a partial swap operator (with a small angle) on the joint system $\rho \otimes \sigma$ and discarding the first register. The entire evolution can be performed using $\mathcal{O}(nt^2/\epsilon)$ one- and two-qubit gates [31].

Density matrix exponentiation is a basic algorithmic primitive that has been applied in a variety of ways [22, 36, 38]. In the original paper, Ref. 36, it was used as a building block in the quantum principle component analysis algorithm. Quantum principle component analysis allows one to (approximately) sample the eigenvectors of ρ corresponding to large eigenvalues exponentially more quickly than any classical algorithm that has access only to single copies of ρ [15, 27]. In Ref. 38, density matrix exponentiation was used to efficiently emulate the action of a unitary U on a small subspace by consuming samples of the form $|b\rangle \otimes U|b\rangle$, where the input states $|b\rangle$ span the subspace. This type of application closely resembles a sort of quantum lookup table, and shares some features with our proposed use of density matrix exponentiation for precomputation, although the aim of that work is different.

B.2 Gate teleportation and the Clifford hierarchy

Our work makes heavy use of the concept of gate teleportation [23]. We illustrated the single-qubit version of gate teleportation in Figure 1 in the main text, but we present a more detailed review here. Given a unitary U, gate teleportation allows us to prepare a resource state $\Gamma(U)$ that we can later consume to apply UP to an arbitrary state $|\psi\rangle$, where the "byproduct operator" P is an element of the Pauli group randomly determined by the measurement outcomes of the teleportation protocol. The state obtained when using gate teleportation to apply U (actually UP) to $|\psi\rangle$ can be written as $\left(UPU^\dagger\right) U|\psi\rangle$. Multiplying by $UP^\dagger U^\dagger$ yields $U|\psi\rangle$.

Gate teleportation can be especially useful when $UP^\dagger U^\dagger$ is simpler to apply than U itself. This is the case in the canonical application of gate teleportation, implementing T gates in a quantum error correcting code that supports fault-tolerant Clifford gates [8]. The problem of applying T gates without error is reduced to the problem of preparing high-fidelity "magic states," because, for all possible byproduct operators P, $TPT^\dagger$ is a Clifford gate despite the fact that T is not.[5] Just as state teleportation trivially generalizes to multiple qubits, gate teleportation can likewise be straightforwardly applied to multiple qubits. In the n-qubit case, the byproduct operator is an n-qubit Pauli operator (up to a phase) that depends on the $2n$-bit measurement outcome obtained from n simultaneous bell basis measurements.

The notion that gate teleportation is most useful when $UP^\dagger U^\dagger$ is easier to implement than U itself led Gottesman and Chuang to define an infinite hierarchy of unitaries now known as the Clifford hierarchy [23]. The first level of the Clifford hierarchy, which we denote by $\mathcal{C}^{(1)}$, is defined to be the Pauli group. The kth level of the Clifford hierarchy is defined inductively,

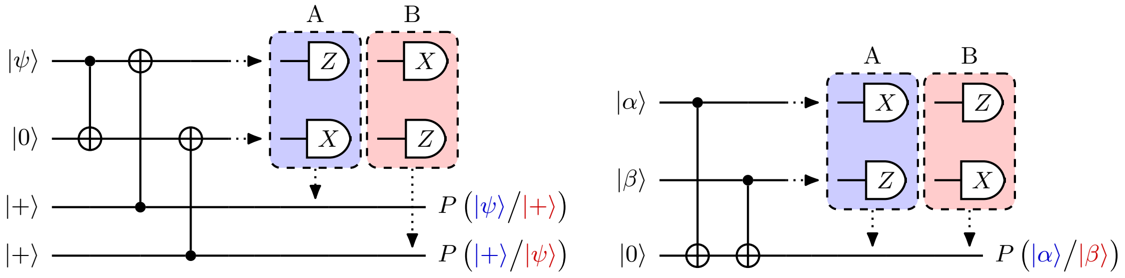
$$\mathcal{C}^{(k)} := \left\{U | UPU^\dagger \in \mathcal{C}^{(k-1)} \; \forall P \in \mathcal{C}^{(1)}\right\}. \tag{17}$$

The second level of the hierarchy is therefore the usual Clifford group. The higher levels of the Clifford hierarchy are harder to characterize in familiar terms, but we can give some examples. For instance, T gates, Toffoli gates, and CCZ gates belong to $\mathcal{C}^{(3)}$. More generally, multi-controlled $C^{k-1}NOT$ and $C^{k-1}Z$ gates are in $\mathcal{C}^{(k)}$, as are the single-qubit rotations Z_k,

$$Z_k := \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi 2^{-k+1}} \end{bmatrix}. \tag{18}$$

It is an open problem to fully characterize the higher levels of the hierarchy, although the diagonal elements are well-understood algebraically in terms of polynomials and roots of unity [16].

[5] In practice, T gates can actually be implemented using a simpler and more specialized form of gate teleportation known as one-bit teleportation [50], but for our purposes we can ignore this detail.


A
B
$|\psi\rangle$
$|0\rangle$
$|+\rangle$
$|+\rangle$
Z
X
X
Z
$P\left(|\psi\rangle/|+\rangle\right)$
$P\left(|+\rangle/|\psi\rangle\right)$
A
B
$|\alpha\rangle$
$|\beta\rangle$
$|0\rangle$
X
Z
Z
X
$P\left(|\alpha\rangle/|\beta\rangle\right)$

(a) Selective destination teleportation

(b) Selective source teleportation

Figure 3: Circuit diagrams for the one-qubit versions of selective destination and source teleportation [18]. Both protocols allow for a choice that is made by selecting between two measurement settings (indicated by the blue and red shaded areas of the diagrams). Selective destination teleportation teleports the state of one qubit to a choice of two different qubits. Selective source teleportation allows one to choose which of two qubits will have its state teleported to a fixed target. The possible states of the output qubit(s) are color-coded to match the measurement settings that select for them. In both cases, a byproduct operator P drawn from the set $\{\mathbb{I}, X, Z, XZ\}$ is randomly applied based on the measurement outcomes.

B.3 Review of selective teleportation

When gate teleportation is used to implement a unitary U that is not in the Clifford group, the resulting correction operator $UP^\dagger U^\dagger$ is not, in general, a Pauli operator. For example, consider the use of gate teleportation to implement a T gate. With probability $\frac{1}{2}$, correcting for the byproduct operator requires the subsequent implementation of a phase gate ($S \in \mathcal{C}^{(2)}$). Naively, this means that after applying a T gate using gate teleportation it is necessary to determine and apply the correction before performing additional Clifford gates. However, in Ref. 18, Fowler showed how a generalization of quantum teleportation can be used to selectively implement this phase gate correction using a small number of ancilla qubits measured in a classically controlled choice of the X or Z basis.

Fowler's selective teleportation relies on two related constructions, selective source teleportation and selective destination teleportation. Selective destination teleportation allows one to teleport a single qubit's state to either one of two destination qubits. Selective source teleportation allows for teleportation from a choice of two different source qubits to a fixed destination qubit. Both types of selective teleportation are controlled by making an appropriate choice of measurement basis and both introduce a Pauli byproduct operator $P \in \{\mathbb{I}, X, Z, XZ\}$ that can be inferred from the (uniformly random) measurement outcomes. We give circuit diagrams for the single-qubit versions of these primitives in Figure 3. The multi-qubit versions are straightforward generalizations.

Together, selective source and destination teleportation can be used to implement a primitive that we refer to as selective gate teleportation. We illustrated the single-qubit version of this selective gate teleportation in Figure 2 in the main text. Selective gate teleportation allows us to apply our choice of unitaries U_1 or U_2 to an unknown n-qubit state $|\psi\rangle$ by choosing how to measure some set of $4n$ ancilla qubits. As a special case, we can use selective gate teleportation to defer the choice of whether or not to apply a unitary U by taking $U_1 = U$ and $U_2 = \mathbb{I}$. Selective gate teleportation randomly introduces the byproduct operators $P^{(1)}$ and $P^{(2)}$ (both n-qubit Pauli operators) before and after the location at which the choice of unitaries is to be applied. For example, let $s \in \{0, 1\}$ denote the classical bit that determines whether or not to perform the teleportation that applies U. Rather than obtaining the desired $U^s|\psi\rangle$, we instead obtain the state $|\phi\rangle = P^{(1)}U^sP^{(2)}|\psi\rangle$. To obtain $U^s|\psi\rangle$, we would need to subsequently apply the correction operator $U^sP^{(2)^\dagger}U^{s\dagger}P^{(1)^\dagger}$.

In the case that Fowler originally consider in Ref. 18, one first uses gate teleportation to implement a T gate (up to a possible S gate correction) and then selectively applies the S gate. Because S is a Clifford gate, $S^s P^{(2)^\dagger} S^{s\dagger} P^{(1)^\dagger}$ is a Pauli operator regardless of the choice of s or the measurement outcomes. As a consequence, the measurements for both teleportation steps can be deferred or performed while applying additional Clifford gates and the necessary Pauli correction can be propagated through the resulting circuit afterwards. This type of optimization has been used to create efficient surface code layouts for a variety of algorithmic primitives [18, 21, 35].

C The $\mathcal{Z}^{(k)}$ hierarchy

In Section 5, we defined $\mathcal{Z}^{(k)}$ to be the set of n-qubit unitaries generated by arbitrary products of controlled Z gates with up to $k-1$ control qubits (including the case with 0 controls, Z gates themselves) and $\pm\mathbb{I}$. For convenience, we define $\mathcal{Z}^{(0)} := \{\pm\mathbb{I}\}$. Let $\mathcal{D}^{(k)}$ denote the elements of the k-th level of the Clifford hierarchy that are also diagonal. As sets, we have that $\mathcal{Z}^{(k)} \subseteq \mathcal{D}^{(k)} \subset \mathcal{C}^{(k)}$. While $\mathcal{C}^{(k)}$ does not form a group for $k > 2$, Ref. 16 showed that $\mathcal{D}^{(k)}$ is a group for all k.

The set $\mathcal{Z}^{(k)}$ can also be shown to form a group under composition. By definition, $\mathcal{Z}^{(k)}$ is closed under composition (which is associative) and includes the identity element. Because diagonal unitaries commute and C^kZ gates are self-inverse for all k, we can see that each element of $\mathcal{Z}^{(k)}$ is its own inverse. Therefore, $\mathcal{Z}^{(k)}$ is a group.

The following proposition will be useful:

Proposition 1. Consider a gate $G \in \mathcal{Z}^{(k)}$ and a product of single-qubit Pauli X operators that we denote by $X_{\boldsymbol{s}}$ (where $\boldsymbol{s} \in [n]$ indicates the indices of the qubits where $X_{\boldsymbol{s}}$ acts non-trivially). Define G' in the following way,

$$G' := X_{\boldsymbol{s}} G X_{\boldsymbol{s}} G^\dagger. \tag{9}$$

Then $G' \in \mathcal{Z}^{(k-1)}$ if $k > 1$ and $G' = \pm\mathbb{I}$ if $k \in \{0, 1\}$. As a corollary, we also have that

$$G X_{\boldsymbol{s}} = X_{\boldsymbol{s}} G' G. \tag{10}$$

Proof. We will prove this proposition by induction. The $k = 0$ case is clear by inspection and the $k = 1$ case follows from the fact that Pauli operators either commute or anti-commute. Now let us assume that the proposition is true for all $j < k$ and prove that it must also hold for $j = k$. Consider an arbitrary $G \in \mathcal{Z}^{(k)}$ and $\boldsymbol{s} \in [n]$.

First of all, we can simplify the proof by considering a single Pauli X operator acting on arbitrary qubit i rather than the product $X_{\boldsymbol{s}}$. This is because we can expand $X_{\boldsymbol{s}} G X_{\boldsymbol{s}} G^\dagger$ as $X_{\boldsymbol{s}} X_{\boldsymbol{s}_1}\ X_{\boldsymbol{s}_1} G X_{\boldsymbol{s}_1} G^\dagger\ X_{\boldsymbol{s}_2}\ X_{\boldsymbol{s}_2} G X_{\boldsymbol{s}_2} G^\dagger \cdots G^\dagger$ through repeated resolutions of the identity. If we can show that $X_i G X_i G^\dagger \in \mathcal{Z}^{(k-1)}$ for all i, then it would follow that

$$X_{\boldsymbol{s}} G X_{\boldsymbol{s}} G^\dagger = X_{\boldsymbol{s}} X_{\boldsymbol{s}_1} G'_1 X_{\boldsymbol{s}_2} G'_2 \cdots \tag{19}$$

for some set of $\{G'_1, G'_2, \cdots\} \subseteq \mathcal{Z}^{(k-1)}$. We could then use the inductive hypothesis to commute the various X operators through to the left, incurring additional terms from the $\mathcal{Z}^{(j)}$ hierarchy with $j < k$. These are all elements of $\mathcal{Z}^{(k-1)}$, which is a group, and therefore their product is also in $\mathcal{Z}^{(k-1)}$. The X terms would cancel, completing the proof.

With that simplification established, the task that remains is to show that

$$X_i G X_i G^\dagger \in \mathcal{Z}^{(k-1)} \tag{20}$$

for an arbitrary qubit i. We can further simplify by expanding G as a product of m unitaries that are either $\pm\mathbb{I}$, single-qubit Z gates, or C^jZ gates (for $j < k$),

$$G = \prod_{\ell=1}^{m} G_\ell. \tag{21}$$

We will proceed by showing that $G_\ell X_i = X_i G'_\ell G_\ell$ for some $G'_\ell \in \mathcal{Z}^{(k-1)}$. If this statement holds, then we can commute X_i to the left through the each of the G_ℓ terms that make up G in Equation (20) and cancel it, picking up a collection of additional G'_ℓ terms from $\mathcal{Z}^{(k-1)}$. Because diagonal unitaries commute, we could also commute these additional terms to the left through the G_ℓ terms, allowing G and $G^\dagger$ to cancel and leaving us with a product of G'_ℓ terms. Because $\mathcal{Z}^{(k-1)}$ is a group, this product of G'_ℓ terms would be in $\mathcal{Z}^{(k-1)}$ and we would therefore be done.

Now all that remains is to show that

$$G_\ell X_i = X_i G'_\ell G_\ell \tag{22}$$

for some $G'_\ell \in \mathcal{Z}^{(k-1)}$. First consider the case where G_ℓ and X_i have support on disjoint qubits. Then we trivially have $G_\ell X_i = X_i G_\ell$, which shows that the equality in Equation (22) holds if we take $G'_\ell = \mathbb{I}$. Now we address the case where X_i acts on one of the qubits that G_ℓ also acts non-trivially on. Let $\boldsymbol{x}$ denote the indices of the qubits where G_ℓ acts non-trivially.

Consider the action of the operator $X_i G_\ell X_i G_\ell$ on an arbitrary state $|\psi\rangle$. Applying G_ℓ flips the sign of those computational basis states where the qubits index by $\boldsymbol{x}$ are all in the 1 state. Applying X_i flips the state of the ith qubit. Applying G_ℓ once again flips the sign of those basis states where the qubits index by $\boldsymbol{x}$ are all in the 1 state. Applying X_i unflips the state of the ith qubit. The cumulative result of these operations is to flip the sign of those states index by the qubits in the set $\boldsymbol{x}\setminus i$. In other words, $X_i G_\ell X_i G_\ell$ acts as a controlled Z operator with one fewer controls than G_ℓ (the control on qubit i is removed). Letting G'_ℓ denote this new operator, we have that $G'_\ell \in \mathcal{Z}^{(k-1)}$ by the definition of $\mathcal{Z}^{(k-1)}$. We can multiply the expression $G'_\ell = X_i G_\ell X_i G_\ell$ by X_i on the left and G_ℓ on the right to obtain the desired result,

$$X_i G'_\ell G_\ell = G_\ell X_i. \tag{23}$$

This completes the proof. □