Perspective

# Focus beyond Quadratic Speedups for Error-Corrected Quantum Advantage

Ryan Babbush,* Jarrod R. McClean,† Michael Newman, Craig Gidney, Sergio Boixo, and Hartmut Neven
*Google Quantum AI, Venice, California 90291, USA*

(Received 10 November 2020; published 29 March 2021)

In this perspective we discuss conditions under which it would be possible for a modest fault-tolerant quantum computer to realize a runtime advantage by executing a quantum algorithm with only a small polynomial speedup over the best classical alternative. The challenge is that the computation must finish within a reasonable amount of time while being difficult enough that the small quantum scaling advantage would compensate for the large constant factor overheads associated with error correction. We compute several examples of such runtimes using state-of-the-art surface code constructions under a variety of assumptions. We conclude that quadratic speedups will not enable quantum advantage on early generations of such fault-tolerant devices unless there is a significant improvement in how we realize quantum error correction. While this conclusion persists even if we were to increase the rate of logical gates in the surface code by more than an order of magnitude, we also repeat this analysis for speedups by other polynomial degrees and find that quartic speedups look significantly more practical.

DOI: 10.1103/PRXQuantum.2.010103

## I. INTRODUCTION

One of the most important goals of the field of quantum computing is to eventually build a fault-tolerant quantum computer. But what valuable and classically challenging problems could we actually solve on such a device? Among the most compelling applications are quantum simulation [1,2] and prime factoring [3]. Quantum algorithms for these tasks give exponential speedups over known classical alternatives but would have limited impact compared with significant improvements in our ability to address problems in broad areas of industrial relevance such as optimization and machine learning. However, while quantum algorithms exist for these applications, the most rigorous results have been able to show a large speedup only in contrived settings or a smaller speedup across a broad range of problems. For example, many quantum algorithms (often based on amplitude amplification [4]) give quadratic speedups for tasks such as search [5], optimization [5–7], Monte Carlo methods [4,8,9], and various machine learning tasks [10,11]. However, attempts to assess the overheads of some such applications within fault tolerance have come up with discouraging predictions for what would be required to achieve a practical advantage against classical algorithms [7,12].

The central issue is that quantum error correction and the device operation time introduce significant constant factor slowdowns to the algorithm runtime (see Fig. 1). These large overheads present many challenges for the practical realization of useful fault-tolerant devices. However, for applications that benefit from an exponential speedup relative to classical algorithms, the exponential scaling of the classical approach quickly catches up to the large constant factors of the quantum approach and so one can achieve a practical runtime advantage for even modest problem sizes. This is borne out through numerous studies on the cost of error-correcting applications with an exponential scaling advantage in areas such as quantum chemistry [14–16], quantum simulation of lattice models [17,18], and prime factoring [19].

In this perspective we discuss when it would be practical for a modest fault-tolerant quantum computer to realize a quantum advantage with quantum algorithms giving only a small polynomial speedup over their classical competition. We see that with only a low-order (e.g., quadratic) speedup, exorbitantly long runtimes are sometimes required for the slightly worse scaling of the classical algorithm to catch up to the slightly better scaling (but worse constant factors) of the quantum algorithm. We argue that the problem is especially pronounced when the best classical algorithms for a problem can also be easily parallelized.

*babbush@google.com
†jmcclean@google.com

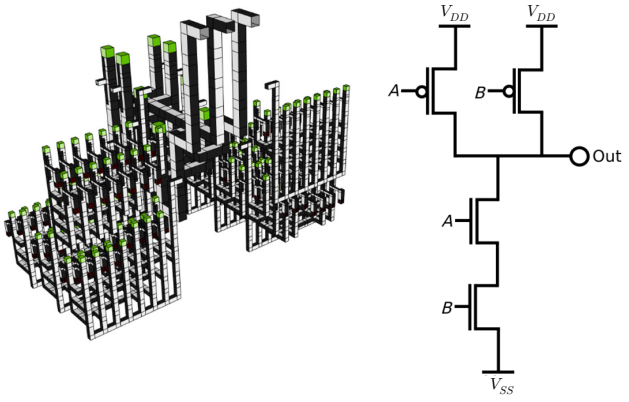


FIG. 1. The primary obstacle in realizing a runtime advantage for low-degree quantum speedups is the enormous slowdown when one is performing basic logic operations within quantum error correction. (a) A surface code Toffoli factory for distilling Toffoli gates (which act as the NAND gate when the target bit is *on*) requires a space-time volume greater than 10 qubit seconds under reasonable assumptions on the capabilities of an error-corrected superconducting qubit platform [13]. (b) A NAND circuit realized in CMOS can be executed with just a few transistors in well under a nanosecond. Thus, there is roughly a difference of 10 orders of magnitude between the space-time volume required for comparable operations on an error-corrected quantum computer and a classical computer.

Our analysis emphasizes current projections within the surface code [20] since it has the highest threshold error rate for a two-dimensional (2D) quantum computing architecture and is generally regarded as the most practical quantum error-correcting code [21]. We focus on a modest realization of the surface code that would involve enough resources to perform classically intractable calculations but support only a few state distillation factories. Our analysis differs from studies analyzing the viability of error-correcting quadratic speedups for combinatorial optimization such as those in Refs. [7,12] by addressing the prospects for achieving quantum advantage via polynomial speedup for a broad class of algorithms rather than for specific problems. Campbell *et al.*[7] were the first to detail poor prospects for error correcting an algorithm achieving a quadratic speedup with a small fault-tolerant processor.

Here we assume that there is some problem that can be solved by a classical computer that makes $M^d$ calls to a "classical primitive" circuit or by a quantum computer that makes $M$ calls to a "quantum primitive" circuit (which is often, but not always, related to the classical primitive circuit). This corresponds to an order $d$ polynomial quantum speedup in the number of queries to these subroutines. For $d = 2$, this is especially evocative of a common class of quantum algorithms leveraging amplitude amplification. This generously assumes no prefactor overhead in a quantum implementation of an algorithm with respect to the number of calls required, and along with other crude assumptions allows us to bound the crossover time.

Our back-of-the-envelope analysis makes many assumptions that are overly optimistic toward the quantum computer and yet we still conclude that the prospects look poor for quadratic speedups with current error-correcting codes and architectures to outperform classical computers in the time to solution. It seems that to realize a quantum advantage with reasonable fault-tolerant resources, one must either focus beyond quadratic speedups or dramatically improve techniques for error correction, or do both. Our conclusion is already "folk wisdom" among some in the small community that studies quantum algorithms and error correction with an eye toward practical realization; however, this reality is not widely appreciated in the broader community that studies algorithms and applications of quantum computers more generally, and there is value in presenting a specific argument to this effect in written form. An encouraging finding is that the prospects for an error-corrected quantum advantage look significantly better with quartic speedups. Of course, there might exist use cases involving quadratic speedups that defy the framework of this analysis. Either way, we hope this perspective will encourage the field to critically examine the prospects for quantum advantage with error-corrected quadratic speedups and either produce examples where it is feasible or focus more effort on algorithms with larger speedups.

## II. RELATIONSHIP BETWEEN PRIMITIVE TIMES AND RUNTIME

Many quantum algorithms are built on coherent access to primitives implemented with classical logic. For example, this classical logic might be required to compute the value of a classical cost function for optimization [12], to evaluate a function of a trajectory of some security that one is pricing with a Monte Carlo method [9], or to compute some classical criterion that flags a marked state for which one might be searching [5]. We define the runtimes of the quantum and classical algorithms as

$$\mathcal{T}_Q = M\, t_Q, \quad \mathcal{T}_C = M^d\, t_C, \tag{1}$$

where $\mathcal{T}$ gives the total runtime of the algorithm, $M$ is the number of primitive calls required, $d$ is the order of the polynomial speedup the quantum computer achieves, and $t$ is the time required to perform a call. Throughout this perspective, the subscripts $Q$ and $C$ denote "quantum" and "classical" implementations.

The condition for quantum advantage is

$$\mathcal{T}_Q < \mathcal{T}_C \quad \text{and thus} \quad M > \left(\frac{t_Q}{t_C}\right)^{\frac{1}{d-1}}. \tag{2}$$

We see then that whenever a problem will require enough calls $M$ that a quantum advantage is possible,

$$\mathcal{T}_Q > \mathcal{T}^\star \equiv t_Q \left(\frac{t_Q}{t_C}\right)^{\frac{1}{d-1}}, \tag{3}$$

where $\mathcal{T}^\star$ is the "break-even time," which occurs when $\mathcal{T}_Q = \mathcal{T}_C$, corresponding to onset of quantum advantage. As emphasized in Fig. 1, we see that the fundamental challenge in realizing this runtime advantage against classical computers (for small $d$) is that $t_Q \gg t_C$ in error-corrected contexts, making $\mathcal{T}^\star$ very large.

Rather than using a single CPU for the classical approach, one might instead parallelize the algorithm using $P$ classical CPUs. This will reduce the total classical runtime to

$$\mathcal{T}_C = \frac{M^d\, t_C}{S}, \tag{4}$$

where

$$S = \left(\alpha + \frac{1-\alpha}{P}\right)^{-1}, \tag{5}$$

where $\alpha$ is the fraction of the algorithm that must be executed in series and $S$ is the speedup factor due to parallelization consistent with "Amdahl's law" [22]. Amdahl's law scaling is considered somewhat pessimistic as one can often adjust the size of problems to fully exploit the computing power that becomes available with more parallelism (e.g., see "Gustafson's law" [23] for a more optimistic formula for $S$). But it also seems that in most situations where one might hope to find a quadratic speedup with a quantum computer (e.g., applications such as search, optimization, Monte Carlo sampling, and regression) the corresponding classical approach is embarrassingly parallel (suggesting that $\alpha$ is small enough that $S \approx P$ for reasonable values of $P$). Regardless of the form of $S$, classical parallelism leads to the following revised conditions for quantum advantage:

$$M > \left(\frac{t_Q S}{t_C}\right)^{\frac{1}{d-1}} \text{ and } \mathcal{T}^\star = t_Q \left(\frac{t_Q S}{t_C}\right)^{\frac{1}{d-1}}. \tag{6}$$

While parallel efficiency might be limited for some applications, any implementation of an error-correcting code will also require substantial classical coprocessing in order to perform decoding, and this is likely to require thousands of classical cores. Although many quantum algorithms can also benefit from various forms of parallelism, we consider an early fault tolerance setting where there is likely an insufficient number of logical qubits to exploit a space-time trade-off to the same extent.

## III. IMPLEMENTING ERROR-CORRECTED QUANTUM PRIMITIVES

We now explain the principal overheads believed to be required for the best candidate for quantum error correction on a two-dimensional lattice: the surface code. Toffoli gates are the most commonly used gate for implementing classical logic on a quantum computer but cannot be implemented transversally within practical implementations of the surface code. Instead, one must implement these gates by first distilling resource states. In particular, to implement a Toffoli gate, one requires a controlled controlled Z (CCZ) state ($|\text{CCZ}\rangle = \text{CCZ}\,|+++\rangle$), and these states are consumed during the implementation of the gate. Distilling CCZ states requires a substantial amount of both time and hardware, and thus they are usually the bottleneck in realizing quantum algorithms within the surface code.

Here we focus on the state-of-the-art Toffoli factory constructions of Ref. [13], which are based on applying the lattice surgery constructions of Ref. [24] to the fault-tolerant Toffoli protocols of Refs. [25,26]. With that approach, one Toffoli gate requires $5.5 \times d$ surface code cycles, where $d$ is the code distance. The time per round of the surface code, including the decoding time, is expected to be around 1 $\mu$s in superconducting qubits. Our analysis assumes a code distance on the order of 30. This would be sufficient for an algorithm with billions of gates and physical gate error rates on the order of $10^{-3}$ (as our analysis will reveal, even more than a billion gates would likely be required to obtain quantum advantage with a modest polynomial speedup). With these assumptions, our model predicts a Toffoli gate time $t_G$ of $30 \times 5.5 \times 1$ $\mu$s $\approx 170$ $\mu$s. This rough approximation matches the more detailed resource estimate of Ref. [13]. We discuss these estimates in more detail in Appendix A.

Under the aforementioned assumptions, which are specific to contemporary realizations of the surface code using superconducting qubits, we could express the quantum primitive runtime as $t_Q = t_G \times G = 170$ $\mu$s $\times G$, where $G$ is the number of Toffoli gates required to implement the quantum primitive. Although we have focused on superconducting qubits, we can also contextualize the performance of ion traps—another leading architecture for quantum advantage. Ion qubits have hour-long coherence times [27] but are typically gated by the performance of their two-qubit gate [28]. Gate times within a single ion crystal can range from hundreds of microseconds to submicroseconds [29–33], and can be efficiently parallelized [34,35].

Multiple ion crystals can be connected to form a networked quantum computer, either through a charge-coupled device [36,37] or via photonic interfaces [38,39]. While a charge-coupled device may support thousands of qubits, millions of qubits will likely require photonic interconnects, although large shuttling-based traps have been proposed [40]. For either architecture, a cycle frequency of approximately 10 kHz has been identified as an ambitious but attainable goal [40–42]. Consequently, we can roughly estimate that such a device will be limited by a clock speed about 100 times slower than the 1-$\mu$s decoding throughput limit, commensurate with typical high-fidelity two-qubit gate times [43,44] and corresponding to $t_G \approx 17$ ms. However, in trade, such a device may support the requisite connectivity for non-2D error-corrected codes and fault-tolerant gates. While the advantages of such approaches are speculative, we touch on some of these alternative proposals in Appendix B.

On a very large surface code quantum computer one could instead use multiple Toffoli factories (at a high cost in the number of physical qubits required) to reduce $t_Q$ by performing state distillation in parallel. However, the Toffoli gates are only about 2 orders of magnitude slower than the Clifford gates, and when using multiple factories, one needs to account for routing overhead. Thus, while $t_Q$ can be reduced at the cost of using many more qubits, the reduction is by a factor that is only between about 10 and 10.

If $N$ is the number of qubits on which this problem is defined, then a sensible lower bound would seem to be $G \geq N$, and thus $t_Q \geq 170\ \mu\text{s} \times N$. For example, in Grover's algorithm [5] one must perform a reflection that requires $\mathcal{O}(N)$ Toffoli gates. To achieve a quantum advantage we would need to focus on problem sizes that are sufficiently large that enough calls can be made so that Eq. (2) is satisfied. We find it difficult to imagine satisfying this condition for problem sizes smaller than 100 qubits. Thus, an approximate "lower bound" (using $N = 100$) would be $t_Q \geq 17$ ms.

In addition to this lower bound, we also consider a specific, realistic example to keep our estimates grounded. We focus on the quantum accelerated simulated annealing by the qubitized quantum walk algorithm studied in Ref. [45,46], which appears to provide a quadratic speedup over classical simulated annealing (at least in terms of the best known bounds) in terms of the mixing time of the Markov chain under certain assumptions [47]. This is among the most efficient algorithms compiled in Ref. [12], and for the Sherrington-Kirkpatrick model [48] the implementation complexity is $5N + \mathcal{O}(\log N)$ (ignoring some subdominant scalings that depend on precision), which is worse than the scaling of our lower bound by only a factor of 5. For example, for an $N = 512$ qubit instance, Sanders *et al.* [12] show that only about $2.6 \times 10^3$ Toffoli gates are required to make an update. Thus, for that problem size (which we choose to facilitate a comparison with classical algorithms, which we discuss later) we have $t_Q = 440$ ms.

## IV. IMPLEMENTING CLASSICAL PRIMITIVES

Classical computers are very fast; a typical 3-GHz CPU can perform several billion 64-bit operations (e.g., floating point multiplications) per second. We might crudely write that the classical primitive time $t_C$ is 330 ps $\times L$, where $L$ is the number of classical clock cycles required to implement the classical primitive. For our first example comparison of quantum and classical primitives, we assume that any classical logic operation that would require one Toffoli gate in the quantum primitive can be executed during one classical clock cycle in the classical primitive. This seems generous to the quantum computer since many operations that would take a single clock cycle on a classical computer would actually require thousands of Toffoli gates. (Note that we are not assuming any scaling advantage for the quantum computer in the primitive implementations.) One might worry about memory-bound classical primitives (since calls to main memory can take hundreds of clock cycles) but since problems defined on more than thousands of logical bits would be infeasible to process on a small fault-tolerant quantum computer, we expect that the memory required for the corresponding classical primitives can be held in cache.

Thus, a corresponding bound on the time to realize a classical primitive for a problem where a quantum computer could realize a quantum primitive with anywhere near the lower bound time given in the previous section ($t_Q \geq 170\ \mu\text{s} \times N$) is $t_C \leq 330$ ps $\times N$, and for $N = 100$, $t_C \leq 33$ ns.

Even though the equivalence we make between Toffoli gates and classical compute cycles is seemingly generous to the quantum computer, the assumption of such a cheap primitive on the quantum side (only 100 Toffoli gates) results in what appears to be a fairly cheap primitive on the classical side. However, because Eq. (6) scales worse with $t_Q$ than with $t_C$, this assumption is ultimately optimistic toward the overall crossover time.

Consistent with the previous section, we also discuss the classical primitive time required to apply simulated annealing to an instance of the Sherrington-Kirkpatrick model. With use of the techniques developed in Ref. [49], a high-performance implementation of the classical simulated annealing code for an $N = 512$ instance of the Sherrington-Kirkpatrick model can perform a simulated annealing step in roughly 7 CPU nanoseconds [12] (this accounts for the fact that most updates for the Sherrington-Kirkpatrick model are rejected); thus, in that case, $t_C = 7$ ns. But given the high costs of quantum computing, it is unclear whether we should make a comparison with a single classical core.

## V. MINIMUM RUNTIME FOR QUADRATIC QUANTUM ADVANTAGE

Here we discuss the ramifications that the primitive runtimes discussed in the previous two sections have for the minimum time to achieve an advantage according to Eq. (3) in the case of a quadratic quantum speedup. First, we compare the example of a quantum primitive requiring only $N = 100$ Toffoli gates and $t_Q = 17$ ms. We argued that any such primitive could likely be computed in $t_C = 33$ ns on a single core. For this example, $\mathcal{T}^\star = t_Q^2/t_C = 2.4$ h. One might object to this minimal example on the grounds that it seems unlikely any interesting primitive would require only 100 Toffoli gates. While this is true, we point out that because the quantum runtime is quadratic in the quantum primitive time and only inversely proportional to the classical primitive time, the overall crossover time can only get worse by our assuming that more than 100 Toffoli gates would be required.

Next we make a comparison with the example of quantum accelerated simulated annealing. We focus on this example because the steps of the quantum algorithm have been concretely compiled, appear quite efficient, and have a clear classical analogue. Here, for an $N = 512$ qubit instance we have $t_Q^2/t_C = 320$ days, reproducing the finding in Ref. [12]. Quantum advantage in this case would occur when $M > t_Q/t_C = 6.3 \times 10^7$. This means that $4.0 \times 10^{15}$ calls would be required for the classical algorithm. However, most $N = 512$ Sherrington-Kirkpatrick model instances would require many fewer calls to solve with classical simulated annealing, and so one would need to focus on an even bigger system, for which the numbers will look yet worse for the quantum computer. Notice that our simulated annealing example gives a quantum runtime that is much longer than the resources required for the quantum primitive with $N = 100$ Toffoli gates. This is because the notion that it would take a classical computer an entire clock cycle to do what a quantum computer could accomplish with a single Toffoli gate is very generous to the quantum computer.

At first glance, the quantum runtime of 2.4 h to achieve an advantage for the primitive with just 100 Toffoli gates seems encouraging. Unfortunately, this was just for a single classical core. Even most laptops have on the order of ten cores these days, and again, most of the problems where quantum computers display a quadratic advantage are classically embarrassingly parallel problems. Furthermore, error-corrected quantum computers are likely to use thousands of classical CPUs just for decoding. When $P$ different classical CPUs are used in parallel, the break-even time is given by Eq. (6). Using that equation, if we take $P = 3000$ CPUs for the classical task (rather than using them for error correction), and if the classical algorithm is sufficiently parallelizable ($\alpha^{-1} \ll P$, so $S \approx P$), we see that the break-even time even in this still quantum-generous example becomes 1 year. As we discuss in the next section, there are also ways of parallelizing the quantum computations (e.g., by using multiple quantum computers or distillation factories).

## VI. THE VIABILITY OF HIGHER POLYNOMIAL SPEEDUPS AND THE IMPACT OF FASTER ERROR CORRECTION

We report values of both $M$ and $\mathcal{T}^\star$ assuming quantum speedups by different polynomial degrees under different amounts of classical parallelism in Table I. While the viability of quantum advantage with cubic speedups is still a bit ambiguous, the prospects of achieving quantum advantage given a quartic speedup are promising. Even the simulated annealing example run with a classical adversary with $S = 10^6$ parallelism would give quantum

TABLE I. Resources required to achieve quantum advantage assuming speedups of various polynomial degrees $d$. We make this comparison against an adversary using distributed classical computing resources that achieve a speedup factor $S$ and report the number of algorithm steps $M$ and the total runtime $\mathcal{T}^\star$ before a quantum speedup is possible. We make this comparison for both the informal resource "lower bound" we argue for in the main text (using $t_Q \geq 17$ ms and $t_C \leq 33$ ns), and for the specific example of quantum simulated annealing applied to the Sherrington-Kirkpatrick model using the quantum and classical implementations discussed in Refs. [12,49] (giving $t_Q = 440$ ms and $t_C = 7$ ns).

| Polynomial degree | Parallelism | Resource "lower bound" | | Simulated annealing | |
|---|---|---|---|---|---|
| | speedup $S$ | Iterations $M$ | Runtime $\mathcal{T}^\star$ | Iterations $M$ | Runtime $\mathcal{T}^\star$ |
| Quadratic, $d = 2$ | 1 | $5.2 \times 10^5$ | 2.4 h | $6.3 \times 10^7$ | 320 days |
| | $10^3$ | $5.2 \times 10^8$ | 100 days | $6.3 \times 10^{10}$ | 880 years |
| | $10^6$ | $5.2 \times 10^{11}$ | 280 years | $6.3 \times 10^{13}$ | 880 millennia |
| Cubic, $d = 3$ | 1 | $7.2 \times 10^2$ | 12 s | $7.9 \times 10^3$ | 58 min |
| | $10^3$ | $2.3 \times 10^4$ | 6.4 min | $2.5 \times 10^5$ | 1.3 days |
| | $10^6$ | $7.2 \times 10^5$ | 3.4 h | $7.9 \times 10^6$ | 40 days |
| Quartic, $d = 4$ | 1 | $8.0 \times 10^1$ | 1.4 s | $4.0 \times 10^2$ | 2.9 min |
| | $10^3$ | $8.0 \times 10^2$ | 14 s | $4.0 \times 10^3$ | 29 min |
| | $10^6$ | $8.0 \times 10^3$ | 2.3 min | $4.0 \times 10^5$ | 4.9 h |

advantage after 5 h of runtime if we assume a quartic speedup (while we do not expect a quartic speedup in that case, the comparison is still instructive).

It is rather surprising just how much of a difference there is for this example between assuming a quadratic speedup (requiring 880 millennia of runtime for an advantage) and a quartic speedup (requiring just 4.9 h of runtime for an advantage). There are not as many examples of quartic speedups in quantum computing but there are a few, such as the tensor principal component analysis algorithm of Hastings [50]. Another example is the quartic query complexity reductions of Ambainis *et al.* [51] and Aaronson *et al.* [52]. We also expect that certain applications of quantum algorithms for linear systems [53] (such as for solving linear differential equations in a high dimension [54]) might lead to modest polynomial speedups higher than quadratic. It is also possible that some heuristic quantum algorithms for optimization might give larger than quadratic improvements for some classes of problems, although this is still speculative.

Another question we might ask is what would happen if we were somehow able to implement Toffoli gates much faster in the surface code? For example, we might achieve this by fanning out and using more physical qubits per factory, by using more Toffoli factories, by inventing significantly more efficient protocols for Toffoli state distillation, or even by switching to a different technology with an intrinsically faster cycle time. We perform this analysis for the case of quadratic speedups; there, the quantum runtime is reduced to $\mathcal{T}_Q = M\, t_Q/R$, where $R \geq 1$ is a speedup factor corresponding to performing Toffoli distillation in time 170 $\mu$s$/R$. In analogy to Eq. (6), this leads to the equations for a quadratic quantum speedup:

$$M > \frac{t_Q S}{t_C R} \text{ and } \mathcal{T}^\star = \frac{t_Q^2 S}{t_C R^2}. \tag{7}$$

In Table II we compute Eq. (7) for our example problems with $R = 10$, $R = 10^2$, and $R = 10^3$ assuming a classical adversary capable of achieving an $S = 10^3$ parallelism. We restrict ourselves to $S = 10^3$ due to the general difficulty in achieving high parallel efficiency described by Amdahl's law. However,for simulated annealing we can achieve $S = 10^6$ in practice (and so these numbers are overly optimistic for that case).

Unfortunately, even if Toffoli distillation rates increase by an order of magnitude it would not be enough to make quantum advantage with a quadratic speedup viable. If Toffoli distillation rates increase by 2 orders magnitude (making them essentially as cheap as Clifford gates) then it would still be challenging to obtain quantum advantage with a quadratic speedup (it would take more than a month for the simulated annealing example despite our limiting the classical parallelism to $S = 10^3$) but we cannot categorically rule it out for all algorithms. For speedup of 3 orders of magnitude, the story would be materially different but this would likely require a significant breakthrough. Even if classical processing and signal propagation were instantaneous, and we could adapt measurements to take advantage of feedforward single-qubit gates being applied only half the time, a single layer of non-Clifford gates would still take a hard limit of the measurement time plus half the single-qubit gate time.

TABLE II. Resources required to achieve quantum advantage under a quadratic speedup assuming a Toffoli distillation time of 170 $\mu$s$/R$ and a classical adversary using classical parallelism with $S = 10^3$. The speedup factor $R$ can account for improvements in error-correction implementations or in our estimates of their overheads. For example, $R = 10$ could be reached by using ten Toffoli factories if routing were very efficient (at the cost of requiring many more qubits).

| Speedup | Resource "lower bound" | | Simulated annealing | |
|---|---|---|---|---|
| factor | Iterations $M$ | Runtime $\mathcal{T}^\star$ | Iterations $M$ | Runtime $\mathcal{T}^\star$ |
| $R = 10^1$ | $5.2 \times 10^7$ | 1.0 day | $6.3 \times 10^9$ | 8.8 years |
| $R = 10^2$ | $5.2 \times 10^6$ | 15 min | $6.3 \times 10^8$ | 32 days |
| $R = 10^3$ | $5.2 \times 10^5$ | 8.8 s | $6.3 \times 10^7$ | 7.7 h |

## VII. CONCLUSION

We investigate simple conditions that must be satisfied to realize a quantum advantage through polynomial speedups on a small fault-tolerant quantum computer. Our ultimate finding is that the prospects are generally poor for a quadratic speedup, consistent with folk knowledge in the error-correction community and recent work such as that in Refs. [7,12]. The comparison with parallel classical resources is particularly damning for quantum computing, and unfortunately many quadratic quantum speedups (especially those leveraging amplitude amplification) apply to problems that are highly parallelizeable. The strongest conclusions in this work assume that one can achieve classical parallelism speedups on the order of $10^3$ or more. But if one can produce a quadratic speedup for a problem where that is not the case, the prospects of quantum advantage would be improved.

These findings do not apply to all polynomial speedups. We find that while one would need to very significantly increase the rate of an error-corrected processor to help the case of quadratic speedups, having a quartic speedup rather than a quadratic speedup is often sufficient to restore the viability of achieving quantum advantage on a modest processor. Thus, we believe that these results suggest that the field should focus beyond quadratic speedups to find viable applications that might produce a quantum advantage on the first several generations of fault-tolerant quantum computers.

We expect this conclusion will persist under a variety of different cost models (e.g., were we to focus on the

energy consumption of a computation rather than the runtime). However, we also expect that the community will make progress on some of the challenges described here, or perhaps identify circumstances under which the assumptions of this analysis do not apply. Either way, we hope that these arguments will foster further discussion about how we might develop broadly applicable algorithms that can achieve quantum advantage on small error-corrected quantum computers.

## ACKNOWLEDGMENTS

The authors thank Dave Bacon, Dominic Berry, Ken Brown, Eddie Farhi, Austin Fowler, Bill Huggins, Sergei Isakov, Evan Jeffrey, Cody Jones, John Platt, Rolando Somma, Nathan Wiebe, and Will Zeng for helpful discussions and feedback on earlier drafts.

## APPENDIX A: ACCOUNTING FOR ERROR-CORRECTION COSTS

In the main text, we provide an estimate for the time that it takes to perform a single Toffoli gate with optimized factories within the surface code. The crux of the argument in the main text is that this time is so much longer than the classical equivalent, and so there is a massive overhead that must be first overcome. We believe that it is valuable in directing future research in error correction and algorithms to break down the origin of this overhead into its contributions from quantum error correction and the physical device speed itself. Here we do this in some detail for the case of the surface code in superconducting qubits, and in passing for ion traps. We hope that this discussion will elucidate several avenues through which breakthroughs in error correction might materially change the analysis in the main text.

To begin, we assume that there is a physical two-qubit operation and syndrome measurement speed, $\tau$ and $\tau_s$, where $\tau_s > \tau$ as $\tau$ is used to build measurement circuits along with a base physical measurement time $\tau_m$. Modern fault-tolerant error correction proceeds via rounds of syndrome extraction, processing, and correction to implement gates. The core physical operation of these rounds on the device is measurement of syndromes, and we are hence lower-bounded by the measurement time $\tau_s$ in realistic settings. For context, estimates of these times for high-fidelity superconducting qubits that would be realistic on improvement are $\tau \approx 10$ ns and $\tau_m \approx 100$ ns.

For a networked ion trap device, there are extra nuances in estimating a realistic syndrome measurement speed [55]. Currently, high-fidelity two-qubit gates and measurements take $\tau \approx 100$ $\mu$s and $\tau_m \approx 10$ $\mu$s [43,56], although high-fidelity microsecond gates have also been demonstrated [32]. Most proposals are limited by a typical trap frequency of approximately 1 MHz, although this limit is not fundamental [31], and submicrosecond gate times are possible [30].

In addition, communicating between different crystals will likely introduce significant overhead. When photonic interconnects are used, the mean connection rate between different modules will be fundamentally limited by the emission rate, which for typical atomic transitions into free space will be approximately 100 MHz. However, current state-of-the-art entanglement generation occurs in the approximately-200-Hz regime [57]. When accounting for fractional light collection and single-photon detector efficiency, we can ambitiously estimate future mean connection rates of approximately 10 kHz [39,41], which may be amplified by generation of entanglement in parallel at an additional cost in space. Without photonic interconnects, shuttling and cooling will introduce additional slowdowns [36], and can currently take hundreds of microseconds [37]. With photonic interconnects, shuttling may still be required to isolate memory ions from light scattered during entanglement generation, although this can be mitigated by use of a different atomic species for communication [58–60].

None of these components are fundamentally limited below approximately 1 MHz. However, many of them must act several times in concert to measure a single round of syndromes. Consequently, $\tau_s \approx 100$ $\mu$s seems an ambitious goal, and is commensurate with earlier estimates [40–42].

If one had perfect operations, but still performed gates via a synthesized and fault-tolerant protocol, these would lower-bound the achievable runtime for a gate. As our operations are not perfect, however, we will need to encode in an error-correcting code with some distance $d$ that is chosen on the basis of the error rate in our device, the threshold of the code, and the total number of operations we expect to perform. If one is allowed to use numerous ancilla qubits, this need not increase the runtime of individual operations by exploiting parallelism through teleportation and space-time optimization [61,62]. However, more qubit spartan implementations must use $d$ rounds of measurement and correction to protect against measurement errors in the time direction, adding a factor of $\mathcal{O}(d)$ in the time cost. Research into one-shot correction techniques hopes to alleviate this time dependence on $d$ without excessive space overhead [63], but current code constructions are not readily implementable.

On top of each round of these measurements, we must account for the time for this information to leave the device, be processed via decoding, and in some cases implement active recovery after a gate, where this time depends on the hardware and the complexity of the decoding. For error correction to be efficient, it must be possible to process the syndrome data without an accumulation of rounds that grows in time. If we denote this processing

latency as $l_r$, then the time for processing $d$ rounds is lower-bounded approximately by the time it takes to produce those syndrome measurements on the physical device plus this latency, or $d\tau_s + l_r$. Depending on the implementation details, $l_r$ is likely to depend on $d$, but with sufficient classical parallelization, it may be possible to make it effectively independent of $d$.

On top of these costs, each gate has some associated prefactor in the number of rounds that depends on the type of gate and its logical locality, $C_G$. For easy, or Clifford, gates in most codes, $C_G$ can be made near 1. Unfortunately, to perform universal computation, one requires a gate that is not easy to implement [64], and common proposals center on state distillation, where the prefactor $C_G$ is often on the order of 10. Moreover, if one considers synthesis of arbitrary rotations into several of these hard gates, $C_G$ can multiply by a factor of 10 or more depending on the precision, leaving $C_G$ on the order of 100. Putting these together, we can approximate a lower bound on the quantum gate time scaling in terms of error-correction parameters as

$$t_G \propto C_G\,(d\tau_s + l_r). \tag{A1}$$

Now that we have a general picture of how the time overhead enters for quantum error correction, we examine it in a specific gate and context. In particular, we focus on superconducting qubits with feasible error rates and operation times within the surface code. Toffoli gates are required to implement classical logic on a quantum computer but cannot be implemented transversally within practical implementations of the surface code. Instead, one must implement these gates by first distilling resource states. To implement a Toffoli gate, one requires a CCZ state ($|\mathrm{CCZ}\rangle = \mathrm{CCZ}\,|{+}{+}{+}\rangle$), and these states are consumed during the implementation of the gate. Distilling CCZ states requires a substantial amount of both time and hardware, and thus they are usually the bottleneck in realizing quantum algorithms within the surface code.

Here we focus on the state-of-the-art Toffoli factory constructions of [13] that are based on applying the lattice surgery constructions of Ref. [24] to the fault-tolerant Toffoli protocols of Refs. [25,26]. Using that approach, one can distill one CCZ state using two levels of state distillation with 5.5 $d + \mathcal{O}(1)$ surface code cycles and a factory with a data qubit footprint of about 12 $d \times 6\,d$, where $d$ is the code distance (the total footprint includes measurement qubits as well, and is thus roughly double this number). Hence for the Toffoli gate, we take $C_G \approx 5.5$.

We assume a correlated-error minimum weight perfect matching decoder capable of keeping pace with 1-$\mu$s rounds of surface code error detection [65], and capable of performing with a similar latency of feedforward in about 1 $\mu$s for $d$ around 30, and conservatively lower-bound the overall time for $d$ rounds to then be $d\tau_s + l_r \leq 30$ $\mu$s. We also assume physical gate error rates on the order of $10^{-3}$, which we hope will be achievable at scale in the next decade. Since we expect to require on the order of billions of Toffoli gates to achieve quantum advantage for practical applications (we will see this is actually a significant underestimate for the case of quadratic speedups), we assume that a code distance $d$ on the order of 30 will be sufficient (since errors are suppressed exponentially in code distance, this number will be approximately correct).

With these assumptions, our model predicts a Toffoli gate time $t_G = C_G(d\tau_s + l_r) \approx 5.5 \times 30\ \mu\mathrm{s}\ \approx 170\ \mu\mathrm{s}$. This rough approximation matches the more detailed resource estimate that shows the space-time volume required to implement one Toffoli gate is approximately 23 qubit seconds [13]. We discuss the resources required for distillation in terms of qubit seconds because it is generally possible to make trade-offs between space and time but the critical resource to be minimized is actually the product of the two. Under these assumptions, we would be able to distill a Toffoli gate in about 170 $\mu$s using around 130 000 physical qubits (see the resource estimation spreadsheet in Ref. [13] for detailed assumptions). Because of this large overhead we focus on estimates assuming we distill CCZ states in series, which is likely how we would operate early fault-tolerant surface code computers. For comparison, if ion trap devices used a similar surface code implementation and error rates while achieving syndrome measurement time $\tau_s = 100$ $\mu$s in parallel, the gate time $t_G$ assuming $C_G \approx 5.5$ is approximately 17 000 $\mu$s, or roughly a factor of 100 slower.

To make this more concrete, we can convert this to a unitless error-correction overhead for a particular gate of $C_G(d\tau_s + l_r)/\tau_s$. If we keep the 30-$\mu$s overall bound for $d\tau_s + l_r$ and make a reasonable estimate for the improvement of physical syndrome measurement times for superconducting qubits to 100 ns, then the error-correction overhead at this distance is 1700.

This suggests that at present for superconducting qubits, the most fruitful improvements with regard to algorithmic speed are the reduction of the decoding time, the minimization of time overheads in distillation factories, and then the reduction of the number of measurement rounds required to protect the state against errors in the time direction, perhaps through improved gate fidelities for equivalent operation times to result in lower required distances or through single-shot protocols. If this can be achieved, the next milestone would be the reduction of physical syndrome extraction time. However, even such advances would make the prospects for realizing a quantum advantage with quadratic speedups considerably more enticing.

## APPENDIX B: ALTERNATIVE APPROACHES

Throughout this work, we focus on the time cost of surface code implementations of non-Clifford gates, as the

expense of such gates has been highly optimized given the connectivity constraints of a superconducting quantum device. Furthermore, the surface code is one of the few error-correction schemes that can operate efficiently in the high-noise regime, with gate infidelities in the range of $10^{-3}$ [21,66].

However, there are limitations when one is considering a two-dimensional architecture. We chose to report on the time cost of logical gates (while keeping the space cost low). By this metric, transverse gates are minimally expensive, yet non-Clifford gates cannot be implemented transversally in the surface code. In fact, any constant depth circuit on a 2D local stabilizer code must be of Clifford type [67], often leading to a time dependence on $d$. Additional connectivity in the device—and likely a requirement of lower error rates—opens up many more avenues toward universal fault-tolerant logic. Beyond magic state distillation, contemporary approaches include, but are not limited to the following:

1. Computing with three-dimensional local codes, where non-Clifford gates are transverse [68] (and may be realized dynamically in two dimensions [69,70])
2. Code switching between codes with complementary transverse gate sets [71]
3. Fixing the gauge of subsystem codes, where different gauges admit different transverse gates [72,73]
4. Concatenating codes, each supporting a complementary transverse gate set [74]
5. Pieceable fault tolerance, which breaks nontransverse gates into fault-tolerant pieces [75,76]

For non-Clifford gates, there are trade-offs that can be made to mitigate the distance and routing overhead of our bound at the expense of many more qubits. Whether these constructions may yield a lower space and time overhead at reasonable error rates is speculative. Numerical studies of alternative schemes have yet to show a convincing advantage [77,78], and often require error rates of less than $10^{-3}$ to operate efficiently, although their space-time footprint can be smaller [69]. Nonetheless, significant optimizations or new approaches are likely needed to recoup a reasonablely sized quadratic quantum advantage.

A separate avenue toward reducing the space overhead of error-correction are block encodings. While surface codes are robust, they require very many qubits per logical qubit. More generally, 2D local code families are fundamentally restricted to have a vanishing ratio of logical qubits to physical qubits, assuming an underlying growing code distance [79,80]. In an all-to-all connected device, a nonvanishing rate is possible without sacrificing the low stabilizer weight often essential for good performance [81]. In particular, there have been promising numerical studies of high-density memories in idealized noise settings using variations on efficient belief propagation decoding [82,83].

For the purposes of our bound, we assume first-generation fault-tolerant devices will support hundreds of logical qubits each with distance $d \sim 30$, and few distillation factories. The total qubit footprint of such a device will remain in the $10^5$–$10^6$ qubit range. By comparison, there exist intermediate-size families of block codes that can encode hundreds of logical qubits using only $10^3$–$10^4$ qubits. However, it is difficult to predict the consequences of using such encodings in the context of our computation-time bound. The required physical error rates may be untenably low, with relatively few studies predicting performance in circuit-level error models [84]. In addition, performing gates efficiently on such codes is a difficult task [85,86]. Therefore, while a significant reduction in memory space may result, the role of such codes in future fault-tolerant devices remains unclear.


[1] R. P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. **21**, 467 (1982).

[2] S. Lloyd, Universal quantum simulators, Science **273**, 1073 (1996).

[3] P. W. Shor, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994), p. 124.

[4] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, in *Quantum Computation and Information*, edited by Vitaly I Voloshin, Samuel J. Lomonaco, and Howard E. Brandt (American Mathematical Society, Washington D.C., 2002), Chap. 3, p. 53.

[5] L. K. Grover, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96 (ACM, New York, NY, USA, 1996), p. 212.

[6] R. D. Somma, S. Boixo, H. Barnum, and E. Knill, Quantum Simulations of Classical Annealing Processes, Phys. Rev. Lett. **101**, 130504 (2008).

[7] E. Campbell, A. Khurana, and A. Montanaro, Applying quantum algorithms to constraint satisfaction problems, Quantum **3**, 167 (2019).

[8] A. Montanaro, Quantum speedup of Monte Carlo methods, Proc. Royal Soc. A: Math. Phys. Eng. Sci. **471**, 20150301 (2015).

[9] P. Rebentrost, B. Gupt, and T. R. Bromley, Quantum computational finance: Monte Carlo pricing of financial derivatives, Phys. Rev. A **98**, 022321 (2018).

[10] E. Aïmeur, G. Brassard, and S. Gambs, in *Advances in Artificial Intelligence*, edited by L. Lamontagne and M. Marchand (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), p. 431.

[11] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning, Quantum Info. Comput. **15**, 316 (2015).

[12] Y. R. Sanders, D. W. Berry, P. C. S. Costa, L. W. Tessler, N. Wiebe, C. Gidney, H. Neven, and R. Babbush, Compilation

of fault-tolerant quantum heuristics for combinatorial optimization, PRX Quantum **1**, 020312 (2020).
[13] C. Gidney and A. G. Fowler, Efficient magic state factories with a catalyzed |CCZ> to 2|T> transformation, Quantum **3**, 135 (2019).
[14] I. D. Kivlichan, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, W. Sun, Z. Jiang, N. Rubin, A. Fowler, A. Aspuru-Guzik, H. Neven, and R. Babbush, Improved fault-tolerant quantum simulation of condensed-phase correlated electrons via trotterization, Quantum **4**, 296 (2020).
[15] V. von Burg, G. H. Low, T. Haner, D. Steiger, M. Reiher, M. Roetteler, and M. Troyer, Quantum computing enhanced computational catalysis, arXiv:2007.14460 (2020).
[16] J. Lee, D. Berry, C. Gidney, W. Huggins, J. McClean, N. Wiebe, and R. Babbush, Even more efficient quantum computations of chemistry through tensor hypercontraction, arXiv:2011.03494 (2020).
[17] J. Lemieux, G. Duclos-Cianci, D. Sénéchal, and D. Poulin, Resource estimate for quantum many-body ground state preparation on a quantum computer, arXiv:2006.04650 (2020).
[18] A. M. Childs, D. Maslov, Y. Nam, N. J. Ross, and Y. Su, Toward the first quantum simulation with quantum speedup, Proc. Natl. Acad. Sci. **115**, 9456 (2018).
[19] C. Gidney and M. Ekerå, How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, arXiv:1905.09749 (2019).
[20] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, Annals of Physics **303**, 2 (2003).
[21] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 32324 (2012).
[22] G. M. Amdahl, in *AFIPS '67 (Spring): Proceedings of the April 18-20, 1967, Spring Joint Computer Conference* (1967), p. 483.
[23] J. L. Gustafson, Reevaluating Amdahl's law, Commun. ACM **31**, 532 (1988).
[24] A. G. Fowler and C. Gidney, Low overhead quantum computation using lattice surgery, arXiv:1808.06709 (2018).
[25] C. Jones, Low-overhead constructions for the faulttolerant Toffoli gate, Phys. Rev. A **87**, 22328 (2013).
[26] B. Eastin, Distilling one-qubit magic states into Toffoli states, Phys. Rev. A **87**, 032321 (2013).
[27] Y. Wang, M. Um, J. Zhang, S. An, M. Lyu, J.-N. Zhang, L.-M. Duan, D. Yum, and K. Kim, Single-qubit quantum memory exceeding ten-minute coherence time, Nat. Photonics **11**, 646 (2017).
[28] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, Trapped-ion quantum computing: Progress and challenges, Appl. Phys. Rev. **6**, 021314 (2019).
[29] K. Mølmer and A. Sørensen, Multiparticle Entanglement of Hot Trapped Ions, Phys. Rev. Lett. **82**, 1835 (1999).
[30] J. J. García-Ripoll, P. Zoller, and J. I. Cirac, Speed Optimized Two-qubit Gates with Laser Coherent Control Techniques for Ion Trap Quantum Computing, Phys. Rev. Lett. **91**, 157901 (2003).
[31] J. Wong-Campos, S. Moses, K. Johnson, and C. Monroe, Demonstration of Two-atom Entanglement with Ultrafast Optical Pulses, Phys. Rev. Lett. **119**, 230501 (2017).
[32] V. Schäfer, C. Ballance, K. Thirumalai, L. Stephenson, T. Ballance, A. Steane, and D. Lucas, Fast quantum logic gates with trapped-ion qubits, Nature **555**, 75 (2018).
[33] E. Torrontegui, D. Heinrich, M. I. Hussain, R. Blatt, and J. J. García-Ripoll, Ultra-fast two-qubit ion gate using sequences of resonant pulses, New J. Phys. **22**, 103024 (2020).
[34] N. Grzesiak, R. Blümel, K. Wright, K. M. Beck, N. C. Pisenti, M. Li, V. Chaplin, J. M. Amini, S. Debnath, and J.-S. Chen *et al.*, Ecient arbitrary simultaneously entangling gates on a trapped-ion quantum computer, Nat. Commun. **11**, 1 (2020).
[35] K. A. Landsman, Y. Wu, P. H. Leung, D. Zhu, N. M. Linke, K. R. Brown, L. Duan, and C. Monroe, Two-qubit entangling gates within arbitrarily long chains of trapped ions, Phys. Rev. A **100**, 022332 (2019).
[36] D. Kielpinski, C. Monroe, and D. J. Wineland, Architecture for a large-scale ion-trap quantum computer, Nature **417**, 709 (2002).
[37] J. M. Pino, J. M. Dreiling, C. Figgatt, J. P. Gaebler, S. A. Moses, C. Baldwin, M. Foss-Feig, D. Hayes, K. Mayer, and C. Ryan-Anderson *et al.*, Demonstration of the qccd trapped-ion quantum computer architecture, arXiv preprint arXiv:2003.01293 (2020).
[38] C. Monroe, R. Raussendorf, A. Ruthven, K. Brown, P. Maunz, L.-M. Duan, and J. Kim, Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects, Phys. Rev. A **89**, 022317 (2014).
[39] D. Hucul, I. V. Inlek, G. Vittorini, C. Crocker, S. Debnath, S. M. Clark, and C. Monroe, Modular entanglement of atomic qubits using photons and phonons, Nat. Phys. **11**, 37 (2015).
[40] B. Lekitsch, S. Weidt, A. G. Fowler, K. Mølmer, S. J. Devitt, C. Wunderlich, and W. K. Hensinger, Blueprint for a microwave trapped ion quantum computer, Sci. Adv. **3**, e1601540 (2017).
[41] K. R. Brown, J. Kim, and C. Monroe, Co-designing a scalable quantum computer with trapped atomic ions, Npj Quantum Inf. **2**, 1 (2016).
[42] N. H. Nickerson, J. F. Fitzsimons, and S. C. Benjamin, Freely Scalable Quantum Technologies using Cells of 5-to-50 Qubits with Very Lossy and Noisy Photonic Links, Phys. Rev. X **4**, 041041 (2014).
[43] C. Ballance, T. Harty, N. Linke, M. Sepiol, and D. Lucas, High-Fidelity Quantum Logic Gates using Trapped-Ion Hyperfine Qubits, Phys. Rev. Lett. **117**, 060504 (2016).
[44] J. P. Gaebler, T. R. Tan, Y. Lin, Y. Wan, R. Bowler, A. C. Keith, S. Glancy, K. Coakley, E. Knill, and D. Leibfried *et al.*, High-fidelity Universal Gate Set for be 9+ Ion Qubits, Phys. Rev. Lett. **117**, 060505 (2016).
[45] R. D. Somma, S. Boixo, H. Barnum, and E. Knill, Quantum Simulations of Classical Annealing Processes, Phys. Rev. Lett. **101**, 130504 (2008).
[46] J. Lemieux, B. Heim, D. Poulin, K. Svore, and M. Troyer, Efficient quantum walk circuits for metropolis-hastings algorithm, Quantum **4**, 287 (2020).
[47] S. Boixo, E. Knill, and R. Somma, Quantum state preparation by phase randomization, Quantum Inf. Comput. **9**, 0833 (2009).
[48] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi, Optimization by Simulated Annealing, Science **220**, 671 (1983).
[49] S. V. Isakov, I. N. Zintchenko, T. F. Ronnow, and M. Troyer, Optimized simulated annealing code for Ising spin glasses, Comput. Phys. Commun. **192**, 265 (2015).

[50] M. B. Hastings, Classical and Quantum Algorithms for Tensor Principal Component Analysis, Quantum **4**, 237 (2020).

[51] A. Ambainis, K. Balodis, A. Belovs, T. Lee, M. Santha, and J. Smotrovs, Separations in Query Complexity Based on Pointer Functions, arXiv:1506.04719 (2015).

[52] S. Aaronson, S. Ben-David, R. Kothari, and A. Tal, Quantum Implications of Huang's Sensitivity Theorem, arXiv:2004.13231 (2020).

[53] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum Algorithm for Linear Systems of Equations, Phys. Rev. Lett. **103**, 150502 (2009).

[54] D. W. Berry, High-order quantum algorithm for solving linear differential equations, J. Phys. A: Math. Theor. **47**, 105301 (2014).

[55] C. J. Trout, M. Li, M. Gutiérrez, Y. Wu, S.-T. Wang, L. Duan, and K. R. Brown, Simulating the performance of a distance-3 surface code in a linear ion trap, New J. Phys. **20**, 043038 (2018).

[56] S. Crain, C. Cahall, G. Vrijsen, E. E. Wollman, M. D. Shaw, V. B. Verma, S. W. Nam, and J. Kim, High-speed low-crosstalk detection of a 171 yb+ qubit using superconducting nanowire single photon detectors, Commun. Phys. **2**, 1 (2019).

[57] L. Stephenson, D. Nadlinger, B. Nichol, S. An, P. Drmota, T. Ballance, K. Thirumalai, J. Goodwin, D. Lucas, and C. Ballance, High-rate, High-fidelity Entanglement of Qubits Across an Elementary Quantum Network, Phys. Rev. Lett. **124**, 110501 (2020).

[58] T. R. Tan, J. P. Gaebler, Y. Lin, Y. Wan, R. Bowler, D. Leibfried, and D. J. Wineland, Multielement logic gates for trapped-ion qubits, Nature **528**, 380 (2015).

[59] C. Ballance, V. Schäfer, J. P. Home, D. Szwer, S. C. Webster, D. Allcock, N. M. Linke, T. Harty, D. A. Craik, and D. N. Stacey *et al.*, Hybrid quantum logic and a test of bell's inequality using two different atomic isotopes, Nature **528**, 384 (2015).

[60] V. Negnevitsky, M. Marinelli, K. K. Mehta, H.-Y. Lo, C. Flühmann, and J. P. Home, Repeated multi-qubit readout and feedback with a mixed-species trapped-ion register, Nature **563**, 527 (2018).

[61] A. G. Fowler, Time-optimal quantum computation, arXiv:1210.4626 (2012).

[62] C. Gidney and A. G. Fowler, Flexible layout of surface code computations using autoccz states, arXiv:1905.08916 (2019).

[63] N. Delfosse, B. W. Reichardt, and K. M. Svore, Beyond single-shot fault-tolerant quantum error correction, arXiv:2002.05180 (2020).

[64] B. Eastin and E. Knill, Restrictions on Transversal Encoded Quantum Gate Sets, Phys. Rev. Lett. **102**, 110502 (2009).

[65] A. G. Fowler, Optimal complexity correction of correlated errors in the surface code, arXiv:1310.0863 (2013).

[66] R. Raussendorf and J. Harrington, Fault-Tolerant Quantum Computation with High Threshold in Two Dimensions, Phys. Rev. Lett. **98**, 190504 (2007).

[67] S. Bravyi and R. König, Classification of Topologically Protected Gates for Local Stabilizer Codes, Phys. Rev. Lett. **110**, 170503 (2013).

[68] M. Vasmer and D. E. Browne, Three-dimensional surface codes: Transversal gates and fault-tolerant architectures, Phys. Rev. A **100**, 012312 (2019).

[69] B. J. Brown, A fault-tolerant non-clifford gate for the surface code in two dimensions, Sci. Adv. **6**, eaay4929 (2020).

[70] H. Bombin, 2d quantum computation with 3d topological codes, arXiv preprint arXiv:1810.09571 (2018).

[71] J. T. Anderson, G. Duclos-Cianci, and D. Poulin, Fault-Tolerant Conversion between the Steane and Reed-Muller Quantum Codes, Phys. Rev. Lett. **113**, 080501 (2014).

[72] A. Paetznick and B. W. Reichardt, Universal Faulttolerant Quantum Computation with only Transversal Gates and Error Correction, Phys. Rev. Lett. **111**, 090505 (2013).

[73] H. Bombín, Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes, New J. Phys. **17**, 083002 (2015).

[74] T. Jochym-O'Connor and R. Laflamme, Using Concatenated Quantum Codes for Universal Faulttolerant Quantum Gates, Phys. Rev. Lett. **112**, 010505 (2014).

[75] T. J. Yoder, R. Takagi, and I. L. Chuang, Universal Fault-Tolerant Gates on Concatenated Stabilizer Codes, Phys. Rev. X **6**, 031039 (2016).

[76] T. J. Yoder, Universal fault-tolerant quantum computation with bacon-shor codes, arXiv preprint arXiv:1705.01686 (2017).

[77] C. Chamberland, T. Jochym-O'Connor, and R. Laflamme, Overhead analysis of universal concatenated quantum codes, Phys. Rev. A **95**, 022313 (2017).

[78] M. E. Beverland, A. Kubica, and K. M. Svore, The cost of universality: A comparative study of the overhead of state distillation and code switching with color codes, arXiv preprint arXiv:2101.02211.

[79] S. Bravyi, D. Poulin, and B. Terhal, Tradeoffs for Reliable Quantum Information Storage in 2d Systems, Phys. Rev. Lett. **104**, 050503 (2010).

[80] S. Bravyi, Subsystem codes with spatially local generators, Phys. Rev. A **83**, 012320 (2011).

[81] J.-P. Tillich and G. Zémor, Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the blocklength, IEEE Trans. Inf. Theory **60**, 1193 (2013).

[82] P. Panteleev and G. Kalachev, Degenerate quantum ldpc codes with good finite length performance, arXiv preprint arXiv:1904.02703 (2019).

[83] A. Grospellier, L. Grouès, A. Krishna, and A. Leverrier, Combining hard and soft decoders for hypergraph product codes, arXiv preprint arXiv:2004.11199 (2020).

[84] O. Higgott and N. P. Breuckmann, Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead, arXiv preprint arXiv:2010.09626 (2020).

[85] D. Gottesman, Fault-tolerant quantum computation with constant overhead, arXiv preprint arXiv:2010.09626 (2013).

[86] A. Krishna and D. Poulin, Fault-tolerant gates on hypergraph product codes, arXiv preprint arXiv:1909.07424 (2019).