Topological Code Autotune

Austin G. Fowler,[1] Adam C. Whiteside,[1] Angus L. McInnes,[1] and Alimohammad Rabbani[2]

[1]*Centre for Quantum Computation and Communication Technology, School of Physics, The University of Melbourne, Victoria 3010, Australia,*
[2]*Sharif University of Technology, Department of Computer Engineering, Tehran, Iran*

Many quantum systems are being investigated in the hope of building a large-scale quantum computer. All of these systems suffer from decoherence, resulting in errors during the execution of quantum gates. Quantum error correction enables reliable quantum computation given unreliable hardware. Unoptimized topological quantum error correction (TQEC), while still effective, performs very suboptimally, especially at low error rates. Hand optimizing the classical processing associated with a TQEC scheme for a specific system to achieve better error tolerance can be extremely laborious. We describe a tool, AUTOTUNE, capable of performing this optimization automatically, and give two highly distinct examples of its use and extreme outperformance of unoptimized TQEC. AUTOTUNE is designed to facilitate the precise study of real hardware running TQEC, with every quantum gate having a realistic, physics-based error model.

Subject Areas: Computational Physics, Quantum Information

Many quantum algorithms now exist, including factoring [1], searching [2], simulating quantum physics [3], problems in knot theory [4], and much more [5]. Large-scale simulations of topological quantum error correction (TQEC) indicate that gate error rates between 0.2% and 0.5% are sufficiently low to enable practical overhead, highly reliability quantum computation [6,7]. This is tantalizingly close to experimentally achieved two-qubit gate error rates of 2% [8], the best achieved to date in a system with the potential to implement the required 2D array of qubits. This motivates the serious study of mapping TQEC schemes to physical hardware to enable realistic engineering trade-offs to be determined and optimizations found. We would very much like to collaborate with any experimentalist with a potentially 2D qubit system and an interest in making use of TQEC. Given the transversely invariant nature of TQEC, experiments involving as few as two qubits can be sufficient to determine whether a system could successfully implement TQEC (see Appendix A).

Neutral atoms in optical lattices [9] motivated the development of TQEC. Optical lattices lack the ability to easily implement arbitrary patterns of two-qubit gates, making them unsuitable for other types of quantum error correction [10–14]. 2D architectures designed for TQEC have since been developed for phosphorus atoms in silicon [15], nitrogen-vacancy color centers in diamond [16], superconducting circuits [17], quantum dots probabilistically entangled using linear optics [18], quantum dots deterministically entangled using nonlinear optics [19], and ion traps [20,21]. Basic TQEC has been experimentally demonstrated using linear optics [22]. In short, the best architectures in all scalable quantum computer technologies now make use of TQEC.

Every effort has been made to make this paper self-contained. The required quantum information background is provided in Sec. I. In Sec. II, TQEC is defined and examples given. Automated methods of analyzing and visualizing the propagation of errors when using TQEC are presented in Sec. III. The extreme performance difference of unoptimized TQEC and autotuned TQEC is numerically demonstrated in Sec. IV. Section V concludes with a discussion of our planned future extensions of AUTOTUNE.

I. QUANTUM INFORMATION

Quantum computers manipulate quantum systems with two relatively stable quantum states that are denoted $|0\rangle$ and $|1\rangle$. These quantum systems are called qubits. Unlike classical bits, which can be either 0 or 1, qubits can be placed in arbitrary superpositions $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. The quantities $|\alpha|^2$ and $|\beta|^2$ represent the probabilities that the qubit, if measured, will be observed to be $|0\rangle$ or $|1\rangle$, respectively.

In addition to initialization to $|0\rangle$ and measurement, we will initially be interested in two quantum gates, Hadamard and controlled-NOT (CNOT). To define their action, we first define

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad (1)$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad (2)$$

leading to

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \qquad (3)$$

Given two qubits in states $|\Psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ and $|\Psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$, the state $|\Psi_1\rangle|\Psi_2\rangle$ corresponds to the outer product

$$|\Psi_1\rangle \otimes |\Psi_2\rangle = \begin{pmatrix} \alpha_1\alpha_2 \\ \alpha_1\beta_2 \\ \beta_1\alpha_2 \\ \beta_1\beta_2 \end{pmatrix}. \qquad (4)$$

Given the above definitions, Hadamard (H) is a single-qubit gate

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad (5)$$

and CNOT (or C_X) is a two-qubit gate:

$$C_X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \qquad (6)$$

Note that H and CNOT are unitary matrices. This is a general property of quantum gates other than initialization and measurement. In this instance, H and CNOT are also Hermitian, and therefore self-inverses. As defined above, CNOT flips the value of the second (target) qubit if the value of the first (control) qubit is $|1\rangle$ and does nothing otherwise.

Very general quantum errors can be expressed in terms of only X and Z errors [10], where

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad (7)$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \qquad (8)$$

A state $|\Psi\rangle$ that contains an X error and is then acted on by H results in the following state: $HX|\Psi\rangle = HXHH|\Psi\rangle = ZH|\Psi\rangle$. This can be verified by simple matrix multiplication. In other words, an X error commuted through an H gate transforms into a Z error. Similarly,

$$Z \xrightarrow{H} X, \qquad (9)$$

$$I \otimes X \xrightarrow{C_X} I \otimes X, \qquad (10)$$

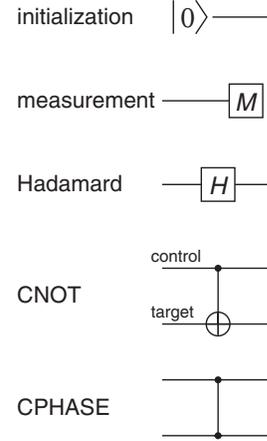$$X \otimes I \xrightarrow{C_X} X \otimes X, \qquad (11)$$

FIG. 1. Quantum circuit symbols for initialization, measurement, Hadamard, CNOT, and CPHASE.

$$I \otimes Z \xrightarrow{C_X} Z \otimes Z, \qquad (12)$$

$$Z \otimes I \xrightarrow{C_X} Z \otimes I. \qquad (13)$$

For brevity, where clear from context, outer products such as $X \otimes I$ are frequently written as XI. We shall use the above rules for propagating errors extensively.

Quantum circuits provide a convenient notation for expressing complex sequences of quantum gates. Figure 1 defines common circuit symbols. Two simple periodic quantum circuits are shown in Fig. 2. When the measurement value of these periodic circuits changes value, the local presence of one or more errors is indicated. These measurement value changes are called detection events. Two similar (sometimes identical) but conceptually distinct circuits are always required to detect all quantum errors requiring correction. We arbitrarily label one of these primal and the other dual. Primal (dual) detection circuits lead to primal (dual) detection events. Errors that lead to primal (dual) detection events are called primal (dual) errors.
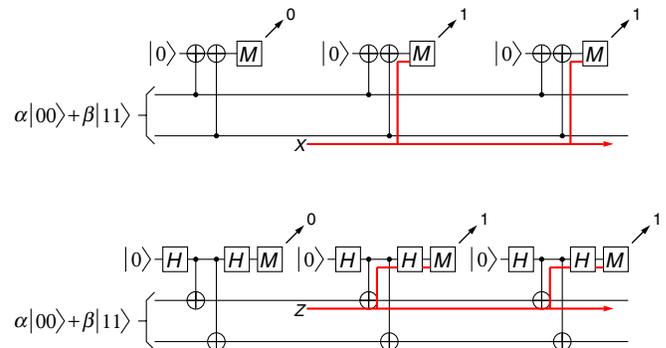

FIG. 2. Examples of error detection circuits. Errors lead to a permanent change in the periodic circuit measurement value. A detection event is associated with any sequential pair of measurements that differ in value.
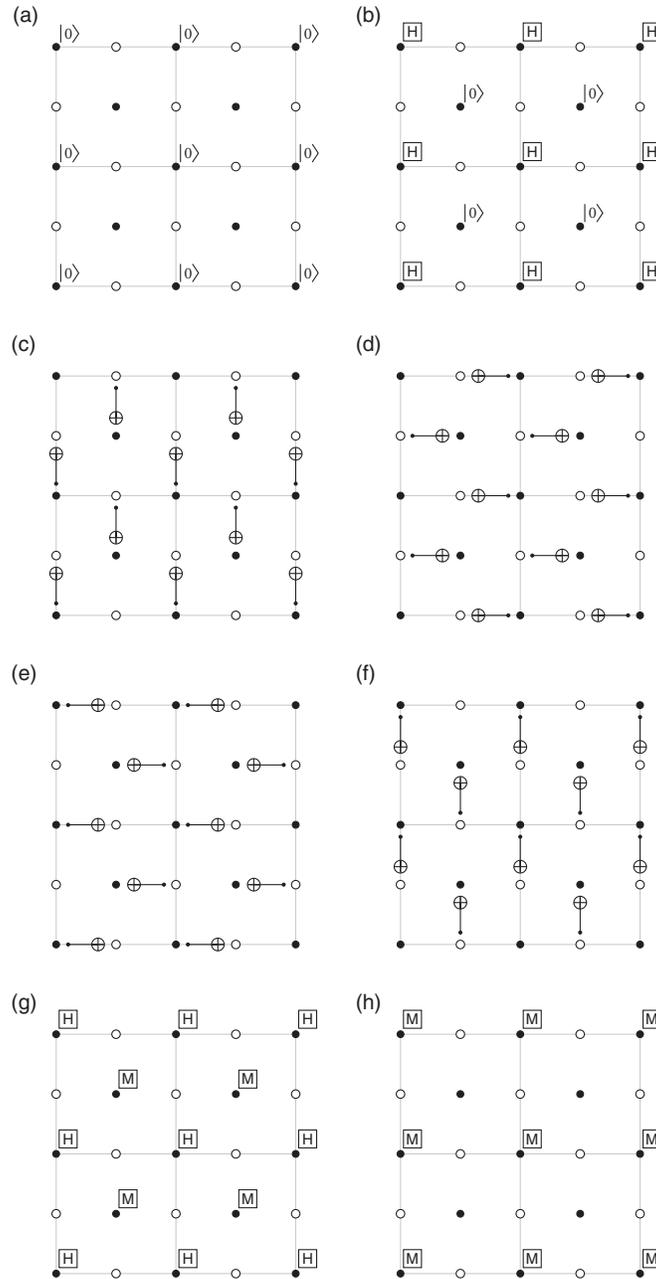
FIG. 3. An eight-layer sequence of quantum gates comprising a single round of surface code quantum error detection. White circles represent data qubits, black circles represent syndrome (error detection) qubits. Grey edges are guides for the eye with no physical significance. Definitions of all quantum gates are given in Fig. 1.

II. TOPOLOGICAL QUANTUM ERROR CORRECTION

For the purposes of this paper, TQEC is defined to be a collection of quantum circuitry on an arbitrary dimensional, nearest-neighbor-coupled lattice of qubits with the property that a single error leads to a pair of primal and/or dual detection events unless the error is near a boundary of the lattice. Near boundaries, a single error can lead to just a single detection event. A 2D planar circuit with these properties is shown in Fig. 3. This circuit is associated with the surface code [23–28]. Examples of errors resulting in a pair of detection events and a single detection event are shown in Fig. 4. Note that error-correction codes have traditionally been defined in terms of stabilizers [29]; however, our definition in terms of quantum circuits is both more general and how AUTOTUNE works internally.
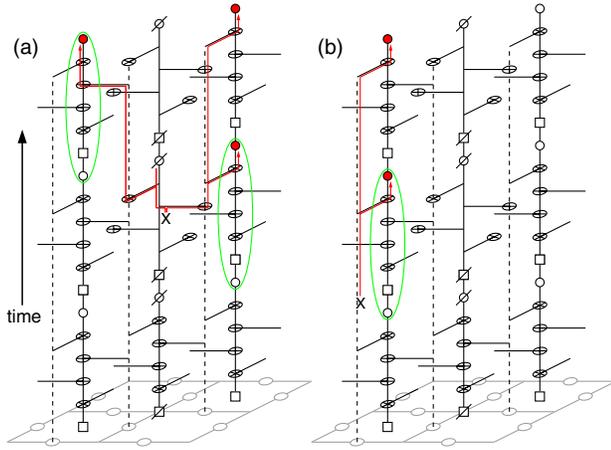
FIG. 4. 2D surface code (grey). Time runs vertically. Squares represent initialization to $|0\rangle$, circles represent Z-basis measurement. Slashed squares represent initialization to $|+\rangle$, slashed circles represent X-basis measurement. (a) A single error leading to a pair of detection events (green ellipses encircle each pair of measurements with differing values associated with each detection event). Red lines show the paths of error propagation, using Eq. (10). (b) An error leading to a single detection event due to its proximity to a boundary of the lattice.

The notion of a detection event as defined above can be significantly generalized. Defining measurement to report the results $+1$ and -1 for $|0\rangle$ and $|1\rangle$, respectively, a detection event can be associated with a set of measurements with product -1. Examples of appropriate sets are shown in Fig. 5(a). This figure also contains arrows indicating nearby temporal and spatial boundaries. This information, namely, set definitions and their boundaries, is required by AUTOTUNE to correctly construct detection events and associate single detection events with the correct boundary.

Armed with these definitions, we can also consider more complex TQEC schemes, such as 3D topological cluster states [26,30,31]. A cluster state [32,33] can be prepared in two stages by first initializing a number of qubits to $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. Physically, this typically corresponds to initializing the qubits to $|0\rangle$ and applying Hadamard. Next, controlled-phase (CPHASE or C_Z) gates are applied to pairs of qubits, where

$$C_Z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{14}$$

A topological cluster state has the form shown in Fig. 5(b). This basic structure is tiled in 3D.

When all topological cluster-state qubits are measured in the X basis, typically achieved by applying Hadamard and then measuring, errors can be detected by multiplying the measurement results in sets of the form shown in Fig. 5(b).

As above, a -1 product indicates a detection event. This is discussed in more detail in [31]. Exactly why this is true is not important for the purposes of this paper. The only feature of TQEC required to read this paper is that detection events are associated with particular locations in space-time and that single errors lead to detection events that are local to one another.

If two errors would lead to a detection event at the same location, their contributions cancel and no detection event is observed at the shared location. Instead, an error chain is formed, with detection events only at the endpoints of the chain. Assuming some characteristic error rate p and
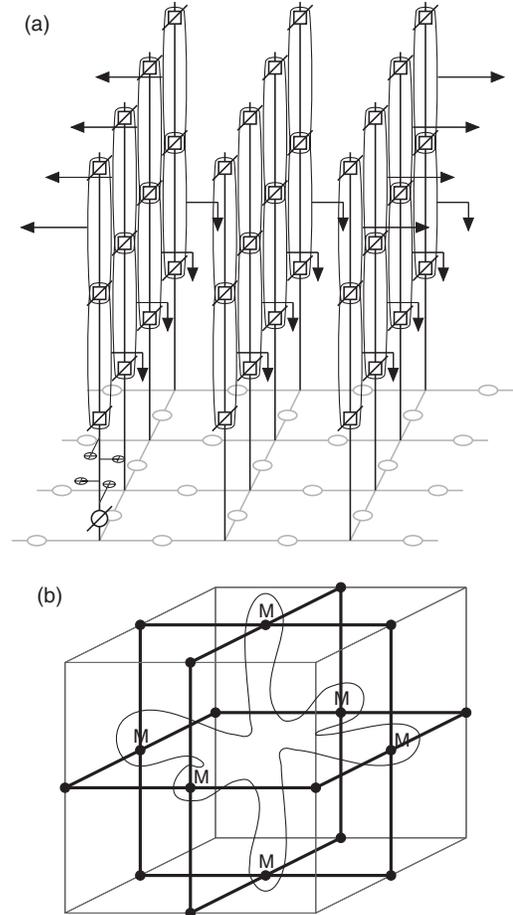

FIG. 5. (a) 2D surface code (grey). Time runs vertically. In the front left corner, the transversely invariant initialization, 4 CNOT, measurement pattern is shown. For clarity, only the measurement gate of this pattern is shown elsewhere in space-time. Long ovals encircle pairs of measurements and represent sets. Arrows indicate associations with boundaries. The bottom layer of sets is associated with the initial time boundary. The second layer left and right rows of sets are associated with the left and right spatial boundaries, respectively. (b) A single 3D topological cluster-state cell. Sets contain six measurements away from the boundaries of the lattice. Dots represent qubits initialized to $|+\rangle$, heavy black lines connecting dots represent CPHASE gates. The order of application of CPHASE gates is irrelevant as they commute.
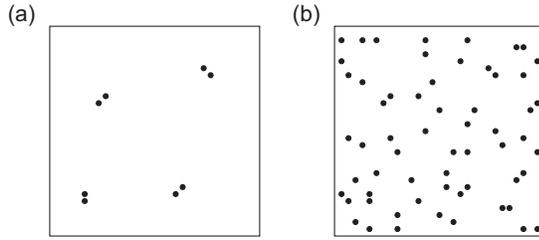
FIG. 6. (a) Typical low error rate pattern of detection events. Such patterns are easy to correct. (b) Typical high error rate pattern of detection events with significant probability of unsuccessful correction.

independent errors, long error chains are exponentially unlikely. Assuming a low error rate, a typical pattern of detection events consists of well-isolated pairs as shown in Fig. 6(a). The higher the error rate, the more difficult it is to guess the location of the errors leading to the detection events. A high error rate pattern of detection events is shown in Fig. 6(b).

A number of different algorithms exist, taking detection events in topological codes and determining locations to apply corrections [34–40]. Edmonds' minimum-weight perfect matching algorithm [7,41–43] is the fastest known algorithm capable of handling practical circuits of the form described in this section. Minimum-weight perfect matching is conceptually simple—after connecting detection events in pairs or individually to nearby boundaries, such that the total length of connecting paths is minimal, corrections are applied along the connecting paths. The errors leading to the detection events in Fig. 6(a) would be corrected with high probability; those in Fig. 6(b) may not be successfully corrected. The error rate at which adding additional qubits fails to improve the probability of successful correction is called the threshold error rate p_{th}. In minimal gate-count fault-tolerant implementations of the surface code, simulations indicate $p_{th} \sim 1\%$ [7,44], which is consistent with the proven lower bound to the threshold error rate of 7.4×10^{-4} [45].

AUTOTUNE uses our own implementation of minimum-weight perfect matching [7,43]. The details of this implementation lie outside the scope of this discussion as AUTOTUNE itself treats matching as a black-box process. The primary question we wish to address is how to define the distance between two detection events. If we assign coordinates to detection events such that neighboring events v_1, v_2 of the same type (primal or dual) differ by one unit in one coordinate, the default metric is the Manhattan metric:

$$D(v_1, v_2) = |i_1 - i_2| + |j_1 - j_2| + |k_1 - k_2|. \quad (15)$$

This metric was used in early TQEC works [25,27,46].

III. TRACKING AND VISUALIZING ERRORS

The Manhattan metric takes no details of the underlying gate sequence or error models of each gate into account. If we imagine every potential location of a primal or dual detection event in space-time, namely, the location of each primal or dual set, the Manhattan metric can be visualized as a cubic lattice. Figure 7 shows the Manhattan lattice associated with the primal sets shown in Fig. 5(a). Each cylinder represents a weight 1 path. A similar lattice exists for the dual sets. The distance D between any given pair of sets is defined to be the shortest (lowest weight) connecting path through the corresponding lattice. Cylinders apparently leading to nowhere actually lead to spatial or temporal boundaries. Given a lattice and randomly generated detection events, minimum-weight perfect matching can be used to match detection events to one another or to nearby boundaries such that the total weight of all connecting paths through the lattice is minimal.

Manhattan lattices are trivial to construct for the surface code and topological cluster states. However, as we shall see in detail in Sec. IV, Manhattan lattices lead to very suboptimal performance. The reason for this can be deduced from Fig. 4(a), in which a single error can be seen to lead to a pair of detection events separated by two units of space and one unit of time. In other words, when using a Manhattan lattice, this single error is treated the same way as some three-error chains. In [47], we showed that this shortcoming leads to a given size surface code only being able to guarantee the correction of half the number of errors it is theoretically capable of always correcting. This shortcoming was removed by laboriously analyzing the surface code circuits and error models by hand and
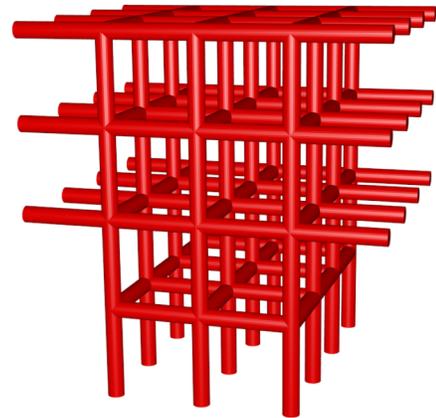

FIG. 7. Manhattan primal lattice of a distance 4-surface code with depolarizing noise. Time runs vertically. Cylinder endpoints represent points in space-time where detection events can occur. Each cylinder has equal diameter, representing an assumed equal probability of detection events at the endpoints of each cylinder. Vertical cylinders leading to nowhere connect to the initial time boundary. Horizontal cylinders leading to nowhere connect to spatial boundaries of the qubit array.

including diagonal weight 1 links (additional diagonal cylinders with the same diameter) in the lattice wherever necessary.

Weight 1 links do not take into account the relative probability of different pairs of detection events. Generally speaking, diagonally separated pairs of detection events are less probable than pairs separated only along one axis of the cubic lattice. In [44], with even more laborious analysis by hand, all distinct types of errors were propagated through the surface code and polynomial expressions were obtained for the probability of each link as a function of a characteristic physical gate error rate p. This analysis was performed only for the body of the lattice, and not attempted near the boundaries. Links to the spatial boundaries were simply assigned the same probability as a horizontal link of an appropriate direction from the body of the lattice. It was deemed too complex and laborious to perform this analysis for the different types of boundaries and corners of the lattice.

When given an arbitrary TQEC circuit and arbitrary stochastic error models for each gate, AUTOTUNE performs a full analysis of the propagation of all errors through all parts of the circuit to determine the probability of all pairs of local detection events. As this is not conceptually complex, we shall not describe the details of how this is achieved here. A description of the AUTOTUNE algorithm can be found in Appendix B. The analysis typically takes less than 1 s. A lattice is then constructed with link weights equal to $-\ln(p_{\text{link}})$, where p_{link} is the total probability of all errors leading to a given link; this is discussed in more detail in Appendix B 5. This ensures that low probability links have larger positive weights and are therefore used less often. It also ensures that the sum of weights along a path through the lattice is related to the product
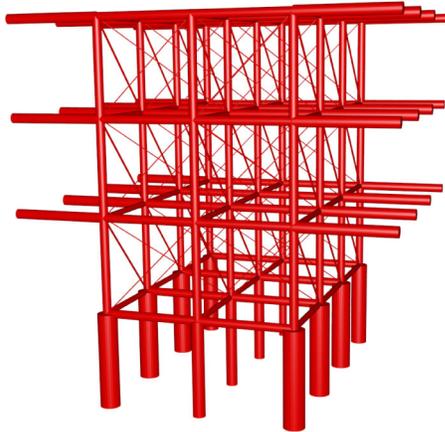

FIG. 8.   AUTOTUNE-generated primal lattice of a distance 4 surface code with depolarizing noise. Note that in contrast to Fig. 7, cylinders do not have equal diameter, accurately representing the diverse range of probabilities of various pairs of detection events. Note also the many additional diagonal cylinders, which are associated with errors that propagate to space-time locations separated by more than one unit of space and/or time (see Fig. 4).
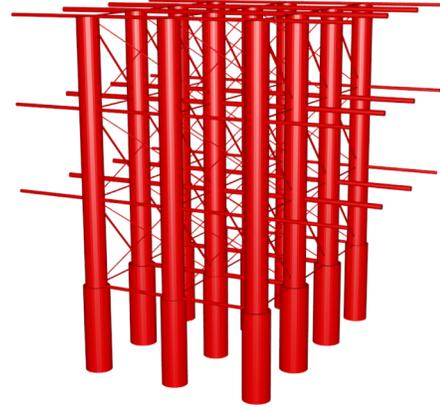

FIG. 9.   AUTOTUNE-generated primal lattice of a distance 4 surface code with measurement error rate $10p$, identity error rate $0.1p$, and a CNOT with total probability of error p but with any error containing Y or Z 100 times more likely than an error containing only I and X. Note that the increased measurement error probability has led to fat vertical cylinders, representing the increased probability that consecutive measurements will differ leading to vertically aligned detection events. Such vertically aligned detection events will be preferentially matched together instead of to nearby boundaries.

of probabilities along that path. Figure 8 shows the AUTOTUNE-generated lattice corresponding to Fig. 5(a). Cylinders now have a diameter proportional to the probability of that link, which we find more useful than the weight for visualization purposes. Note the varying diameters of cylinders to the boundary.

Figure 8 was created with a standard depolarizing error model. Initialization and measurement produce or report the wrong state with probability p; identity and Hadamard gates introduce an X, $Y = XZ$, or Z error each with probability $p/3$; and CNOT introduces one of the 15 non-trivial outer products of I, X, Y, and Z, each with probability $p/15$. Figure 9 shows a lattice generated with measurement error rate $10p$, identity error rate $0.1p$, and a CNOT with total probability of error p but with any error containing Y or Z 100 times more likely than an error containing only I and X. Note the much thicker vertical cylinders due to the high measurement error rate. AUTOTUNE is designed to handle absolutely any set of stochastic error models. Note that after the first few layers, the structure of the lattice repeats. Exactly the same pattern of links and probabilities will be generated since the TQEC circuit is repetitive. This means that we do not need to endlessly perform an analysis of the propagation of all errors; rather, we perform the analysis until it becomes repetitive, then simply continue to generate the repetitive structure without the analysis.

IV. UNOPTIMIZED VERSUS AUTOTUNED TQEC

When we simulate the surface code, we study the situation of a lattice of finite spatial extent and potentially infinite

temporal extent. In practice, the lattice is generated dynamically as gates are simulated and random errors generated. We are interested in the logical error rate per round of error detection. A logical error is a chain of errors after correction that connects distinct boundaries. These errors can be detected in simulations as changes in the logical state.

The distance d of a code is the minimum number of operations required to change the logical state. For each (d, p) pair, we typically let our simulation run until 10 000 logical state changes are observed. Fewer state changes are occasionally permitted at very low p and high d as the runtime of the simulations increases rapidly due to the very low logical error rate and consequent large number of required rounds of simulated error detection. We have recently developed fast analytic methods to calculate the low-p logical error rate for arbitrary even d [48].

The logical-X error rate per round of error correction for a range of values of d and p when using Manhattan lattices is shown in Fig. 10. This should be contrasted with the performance when using AUTOTUNE-generated lattices, which is shown in Fig. 11.

The parallel asymptotic curves for distances 5 and 7 in Fig. 10 are correct. In Fig. 4(a), an example of a single error leading to detection events separated by two units of space and one unit of time is shown. When using the Manhattan lattice, three corrections must be inserted to pair such detection events. Said another way, when using the Manhattan lattice, this single error is indistinguishable from a three-error process. A logical error occurs when detection events are incorrectly matched, forming a logical operator of errors and corrections. Logical operators are associated with paths through the lattice connecting opposing boundaries. The shortest paths make use of the same number of links as the code distance. There exist logical
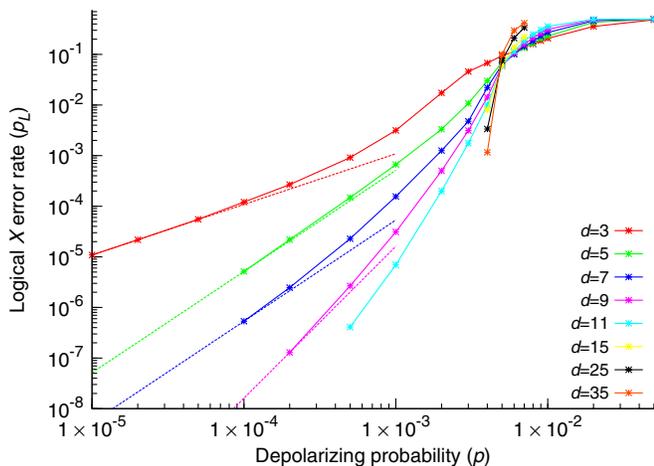

FIG. 11.   Probability of logical logical-X error per round of error correction for various code distances d and physical error rates p when using an AUTOTUNE-generated lattice. The asymptotic curves (dashed lines) are quadratic, cubic, and quartic for distances $d = 3$, 5, and 7, respectively.

operators in a $d = 5$ code that do not follow a shortest path and are associated with a total of 9 links and incorporate two single-triple errors. Given the detection events associated with these single-triple errors, matching will choose to match incorrectly (inserting 3 corrections) rather than correctly (which would require 6 corrections), forming a

FIG. 12.   A unit cell of the first layer of the interaction pattern of a double layer of qubits generating a 3D cluster state. Small black dots correspond to qubits. The qubits in the second layer have been omitted for clarity. The numbers at the top right and bottom left of each qubit indicate the time steps in which it is initialized and measured, respectively. Each line corresponds to a C_Z gate and is labeled with the time step(s) in which this gate is applied. The second layer comprises an identical ordering of qubits and gates, but is shifted in time by three steps (such that a gate labeled 2 is executed in time step 5 and vice versa, etc.) and shifted north, east, and up as indicated by the arrow. Upward C_Z gates in the first layer become downward after shifting.

FIG. 10.   Probability of logical logical-X error per round of error correction for various code distances d and physical error rates p when using a Manhattan lattice. The asymptotic curves (dashed lines) are linear, quadratic, quadratic, and cubic for distances $d = 3$, 5, 7, and 9, respectively.
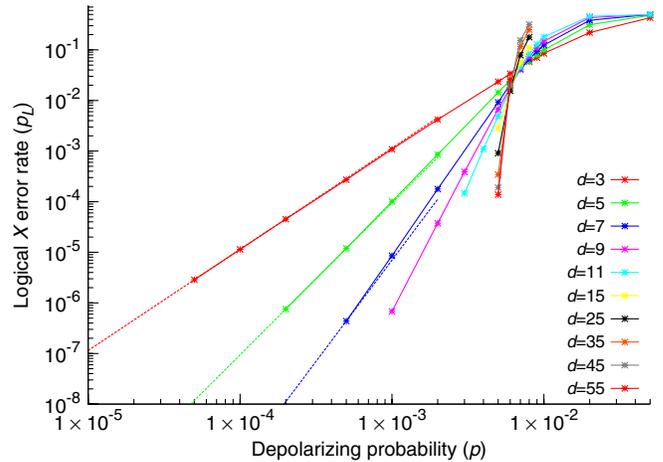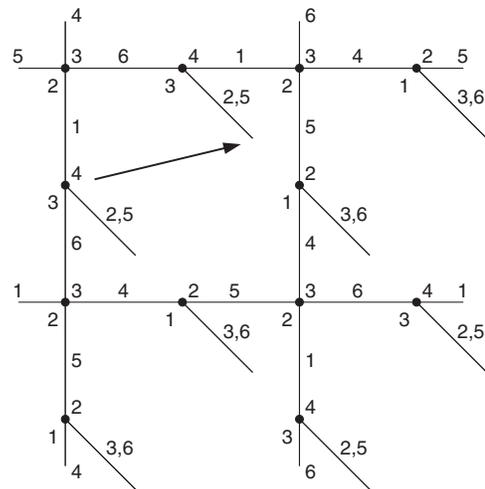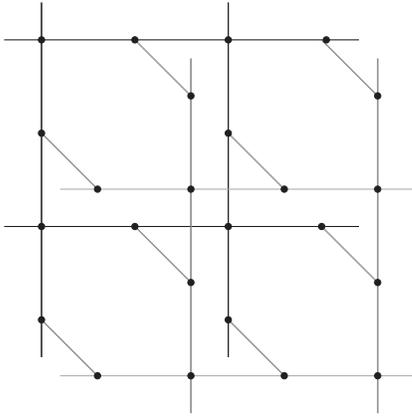
FIG. 13. A single-layer implementation of the two layers used to progressively generate a 3D topological cluster state. Note that while some interactions cross, they are still technically nearest neighbor.

logical error. This is why the asymptotic curve for $d = 5$ is only quadratic. Similarly, a $d = 7$ code contains logical operators consisting of 11 links incorporating two single-triple errors and matching will choose to match incorrectly (inserting 5 corrections) rather than correctly (6 corrections). Fewer combinations of two single-triple errors lead to failure in a $d = 7$ code, explaining the lower logical error rate. AUTOTUNE lattices do not suffer the shortcomings described in this paragraph as they include all necessary diagonal links and therefore guarantee correction of $\lfloor (d - 1)/2 \rfloor$ errors.

If we focus on physical error rates of $p = 10^{-3}$, a reasonable medium-term goal for scalable two-qubit interactions, the distance $d = 5$ autotuned logical error rate is nearly a factor of 10 lower than the Manhattan logical error rate. At $d = 7$, the improvement is over a factor of 40. The ratio of out performance continues to grow with d. The ratio of out performance also grows rapidly as p is decreased. At $p = 10^{-5}$, the out performance is already a factor of 100 at $d = 3$, and over 5 orders of magnitude at $d = 7$. This extreme performance difference makes AUTOTUNE an essential tool for analyzing the surface code.
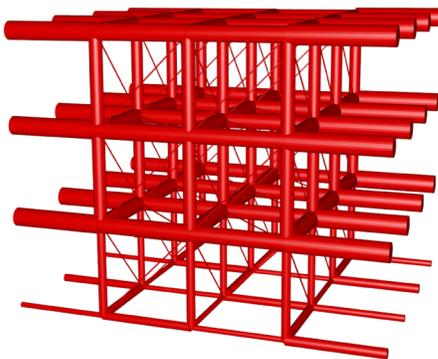

FIG. 14. AUTOTUNE-generated primal lattice of a distance 4 topological cluster state with depolarizing noise.
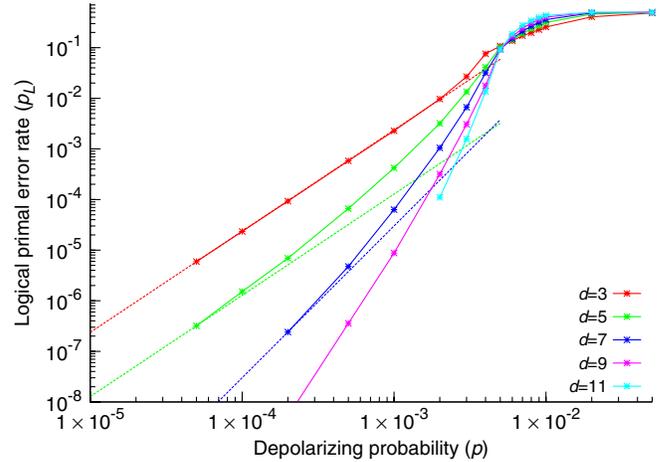

FIG. 15. Probability of logical primal error per round of error correction for various topological cluster-state code distances d and physical error rates p when using a Manhattan lattice. The asymptotic curves (dashed lines) are quadratic, quadratic, and cubic for distances $d = 3$, 5, and 7, respectively.

Turning our attention to topological cluster states, the first step is to describe a way to progressively build the cluster state using only a 2-D lattice of qubits and nearest-neighbor interactions. We start with a double layer of qubits. The required sequence of initialization, measurement, and C_Z gates is somewhat complex; however, an attempt to convey this clearly can be found in Fig. 12. An appropriate single layer of qubits and interactions capable of implementing the same sequence is shown in Fig. 13.

We assume the same error model as that used for the surface code, including the explicit use of Hadamard gates and initialization and measurement in the Z basis only. The primal nest resulting from the described gate sequence and
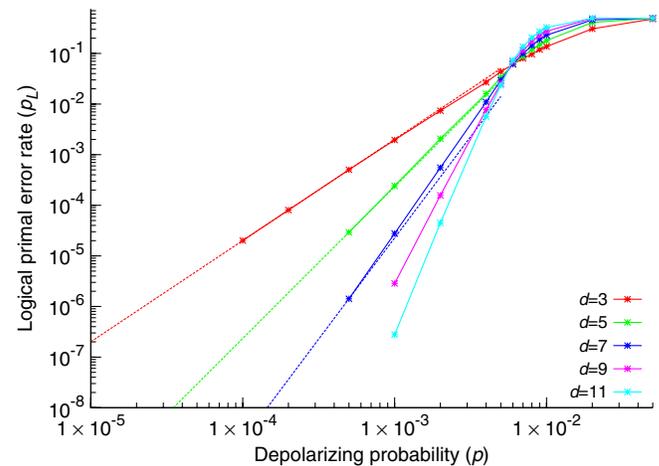

FIG. 16. Probability of logical primal error per round of error correction for various topological cluster-state code distances d and physical error rates p when using an AUTOTUNE-generated lattice. The asymptotic curves (dashed lines) are quadratic, cubic, and quartic for distances $d = 3$, 5, and 7, respectively.

error model for a distance 4 topological cluster state is shown in Fig. 14. Figures 15 and 16 show the Manhattan lattice and autotuned lattice error-correction performance, with the distance 5 logical error rate, in particular, showcasing the advantage of using AUTOTUNE. Note the parallel asymptotic curves for $d = 3$ and 5 in the Manhattan case, caused by single errors leading to detection events separated by two links.

V. CONCLUSION

We have described a tool, AUTOTUNE, that is capable of handling, in a natural manner, both fully fault-tolerant surface codes and 3D topological cluster states, with the full code distance achieved in both cases. This generality is achieved through the definition of sets of measurements that are used to detect error-chain endpoints. Arbitrary syndrome measurement circuits are supported along with arbitrary stochastic error models for each gate. The details of the measurement circuits and gate error models are analyzed before simulation begins to ensure high error-correction performance. The algorithms upon which AUTOTUNE is based are highly efficient, with a per round runtime that is comparable to that reported in [43].

In future work, we plan to take correlations between errors into account, improve the efficiency of AUTOTUNE's handling of qubit loss, parallelize the core matching engine, and develop the capability to analyze complex logical circuits consisting of many braided defects, including the precise simulation of the complete quantum state.

APPENDIX A: TWO-QUBIT EXPERIMENTS DEMONSTRATING TQEC

This Appendix describes generic experiments requiring as few as two qubits that are sufficient to prove that a physical system is capable of implementing TQEC. A large 2D array of qubits is not required, provided the experiment is done in a scalable manner. It must be clear where an arbitrarily large number of additional qubits will go, where all of the control lines will go, where all of the control electronics will go, how scalable cooling will be achieved, and how all aspects of the construction and maintenance of the complete large-scale hypothetical quantum computer will be achieved. No issues can be ignored. This is best achieved with a modular design, such as that described in [16].

The simplest possible modular structure capable of being assembled into a large 2D array of qubits contains just a single qubit with the ability to connect to four neighboring modules. Two such modules connected together are sufficient to demonstrate initialization, measurement, identity, and single- and two-qubit gates. If the error rates of all of these quantum gates are below approximately 1%, the experiment will have demonstrated that nature permits large-scale quantum computation. We believe that the importance of such an experiment cannot be overstated.

A number of trade-offs are possible. For example, measurement error rates of approximately 10% are acceptable provided two-qubit interactions, CNOT or CZ, are possible with an error rate of approximately 0.1%. Precise trade-offs can be determined for specific hardware using AUTOTUNE.

Each module need not be a physically separate device. A pair of solid-state modules may consist of a pair of coupled qubits together with coupling hardware for the six additional nearest-neighbor qubits, which may or may not themselves actually be present. At least the coupling hardware needs to be present to ensure that control and decoherence challenges associated with scaling have indeed been overcome. It is possible that additional qubits and coupling hardware may need to be present if the mechanism achieving qubit interactions significantly couples more than just nearest-neighbor qubits.

To take a specific example, consider a *scalable* ion-trap quantum computer based on a regular array of interaction regions connected by a square grid of transport paths. Provided the design is truly scalable, it would be sufficient to build just two unit cells of the scalable array containing, in total, perhaps just two interaction regions and two transport junctions. Four ions would be required, two data ions and two sympathetic cooling ions. One would then need to demonstrate high fidelity (> 0.99) initialization, measurement, and single-qubit rotations. Most challenging, one would need to demonstrate an interaction process consisting of the transport of one ion to the other interaction region, sympathetic cooling if necessary, combination, high-fidelity interaction, separation, transport, more sympathetic cooling if necessary, and then prove that the entire process had fidelity greater than 0.99. One would also need to show that an identity gate of duration equal to the interaction process could be achieved with fidelity greater than 0.99. Finally, one would need to show that ion loss and leakage to noncomputational states are low probability events, preferably less than 1% per interaction process or other gate. An experiment of this form would put a price tag on a qubit and enable one to calculate, using AUTOTUNE, the precise size, cost, and performance of a large-scale quantum computer based on many of the experimentally demonstrated unit cells. This would reduce the problem of building a quantum computer to one of price performance, a problem sure to attract significant industry interest.

We would be delighted to have arbitrarily detailed discussions with anyone wishing to design a "nature permits quantum computation" experiment with any type of hardware.

APPENDIX B: AUTOTUNE ALGORITHM

The purpose of AUTOTUNE is to enable the precise analysis of real hardware running TQEC. This is achieved by taking user-defined error models for every quantum gate and user-defined quantum circuits implementing TQEC and working out exactly where every possible error on every gate would be detected in space-time and which errors lead to the same detection events. When simulating the operation of the hardware stochastically, the detailed error propagation information produced by AUTOTUNE can be used to reliably guess which errors led to the observed detection events. This results in the extremely effective correction of physical errors. In this Appendix, we describe how AUTOTUNE represents error models, how these errors are propagated through quantum circuits, how errors are detected (sets of measurements), how the detailed error information is visualized (nests of balls and sticks), and how the detailed error information can be reduced to simpler structures that can be generated efficiently during stochastic simulation (lattices of dots and lines).

1. Error models and tracking

AUTOTUNE is capable of handling any error model with outcomes that can be described by a single integer e per qubit. For example, $I = 0$, $X = 1$, $Z = 2$, $Y = 3$, leaked to a noncomputational state $= 4$, qubit lost $= 5$. Only single-qubit and two-qubit gate error models are currently supported; however, this could easily be extended. The user can specify how e is transformed by each gate, for example, controlling whether a CNOT between a leaked qubit and a nonleaked qubit results in two leaked qubits, or results in no effect on the nonleaked qubit, or any other effect describable by single integers.

An example of a Pauli channel CNOT error model file is shown below.

$$
\begin{array}{lcc}
2 & & \\
1.0 & & \\
6 & & \\
81 & 0 & 3 \\
14 & 1 & 0 \\
13 & 2 & 1 \\
28 & 2 & 2 \\
78 & 3 & 0 \\
30 & 3 & 2 \\
1 & &
\end{array}
\tag{B1}
$$

The first line states the number of qubits n_q the gate is applied to. The second line states the value x that the relative strengths s_i of the various errors should be normalized to sum to. This makes it easy to handle different gates with different overall probabilities of error, e.g., $x_{1q} = x_{2q}/10$. The third line states the number of different errors n_e in the model. The next n_e lines contain $n_q + 1$ integers specifying the relative strength s_i of that error and the value of e to apply to each qubit. The user can specify exactly how different errors combine $e_1 e_2 = e_3$. Each time a quantum gate is called, it is passed an error rate p, and an error of any kind is applied with probability px. Error i is then applied with relative probability $s_i / \Sigma s_i$. The final line is the gate duration in arbitrary units. For discussion purposes, we include the internal representation of the error model below, which makes use of explicit relative probabilities rather than integers.

$$
\begin{array}{lcc}
2 & & \\
1.0 & & \\
6 & & \\
0.332 & 0 & 3 \\
0.057 & 1 & 0 \\
0.053 & 2 & 1 \\
0.115 & 2 & 2 \\
0.320 & 3 & 0 \\
0.123 & 3 & 2 \\
1 & &
\end{array}
\tag{B2}
$$

Error models are not only used to generate stochastic errors. Every time a gate is applied, all possible errors are generated and added to the list of errors on each qubit touched by the gate. The probability $px s_i / \Sigma s_i$ is recorded in each error data structure. Each error is given a unique label. Multiple-qubit errors are represented by single-qubit errors on each qubit, each with the same label. If one executes a long sequence of unitary gates, the number of errors per qubit that need to be tracked will grow (linearly) without bound. Each unitary gate transforms all errors present on all of the qubits it touches. An H gate will transform X errors into Z errors and vice versa. Multiple-qubit gates can create new propagated errors that will have the same label as the original and can combine or cancel multiple errors on a single qubit with the same label. For example, CNOT($q_1, q_2, p = 0.01$) applied to qubits

$$q_1 \rightarrow (X, 0.005\,00, 0)$$
$$\hookrightarrow (Z, 0.002\,38, 1), \tag{B3}$$

$$q_2 \rightarrow (Y, 0.002\,38, 1), \tag{B4}$$

where (A, p_{sr}, L) represents the error type, scaled relative probability, and label, and the arrows represent a linked list, will result in

$$q_1 \rightarrow (X, 0.005\,00, 0)$$
$$\hookrightarrow (X, 0.000\,57, 3)$$
$$\hookrightarrow (Z, 0.000\,53, 4)$$
$$\hookrightarrow (Z, 0.001\,15, 5)$$
$$\hookrightarrow (Y, 0.003\,20, 6)$$
$$\hookrightarrow (Y, 0.001\,23, 7), \tag{B5}$$

$$q_2 \rightarrow (X, 0.005\,00, 0)$$
$$\hookrightarrow (Y, 0.002\,38, 1)$$
$$\hookrightarrow (Y, 0.003\,32, 2)$$
$$\hookrightarrow (X, 0.000\,53, 4)$$
$$\hookrightarrow (Z, 0.001\,15, 5)$$
$$\hookrightarrow (Z, 0.001\,23, 7). \tag{B6}$$

2. Sets of measurements

AUTOTUNE currently supports only single-qubit measurements in the X and Z bases. M_X applied to Eq. (B6) will create a measurement:

$$m \rightarrow (Z, 0.002\,38, 1)$$
$$\hookrightarrow (Z, 0.003\,32, 2)$$
$$\hookrightarrow (Z, 0.001\,15, 5)$$
$$\hookrightarrow (Z, 0.001\,23, 7). \tag{B7}$$

The X errors and X components of the Y errors have been removed. All errors will be removed from the qubit. To use the qubit again it must be explicitly initialized. If the qubit does not need to be used immediately after being measured, AUTOTUNE provides a dead gate that advances the qubit in time but does not generate or track any errors. This models incoherent evolution.

Every measurement is associated with either two sets or a single set and a boundary. A set of measurements has the property that the product of the measurement results (+ 1 or −1) indicates whether a chain of errors has ended nearby. In the standard surface code, sets contain consecutive pairs of syndrome qubit measurements. In a 3D topological cluster state, sets contain the measurements on the faces of individual primal and dual cells. See Fig. 5. Sets must be specified by the user.

Sets can also be associated with boundaries. The bottom layer of the sets in Fig. 5(a) is associated with the primal initial time boundary. In the second layer of sets, the sets in the left row are associated with the left primal boundary; those in the right row are associated with the right primal boundary. We use the terminology primal and dual instead of rough and smooth as used in [27] to ensure uniform terminology when discussing both the surface code and 3D topological cluster states. Note that the middle row of sets

in the second layer is not associated with any boundary. The association of sets with boundaries is currently manually user specified.

3. Detection events

When all measurements in a set have been performed, further processing is triggered. A measurement may contain many errors. A set may contain many measurements. AUTOTUNE determines which errors with the same label appear an odd number of times. For each such label, a detection event is generated. Figure 4(a) contains an example of a single error leading to a pair of detection events. Detection events are stored in a hash table to enable one to quickly determine whether a detection event with a given label has already been generated. Pairs of detection events immediately trigger the creation of sticks, which are described in the following subsection.

Errors near a boundary can lead to single detection events [Fig. 4(b)]. Such single detection events must generate sticks leading to the nearby boundary. One must decide with care when to conclude that a detection event is unique and that no matching detection event will be generated in the future. To deal with this, we define a measure of error-detection progress, big_t, to increment only when every stabilizer of the code has been measured at least once. An example of an error leading to detection events two big_t in the future is shown in Fig. 17. This is the maximum delay possible. Detection events three big_t in the past that are unique are guaranteed to remain so.

Detection events can be primal or dual depending on whether they are associated with primal or dual sets, which are in turn associated with primal or dual stabilizer measurements. It is a good idea to design stabilizer
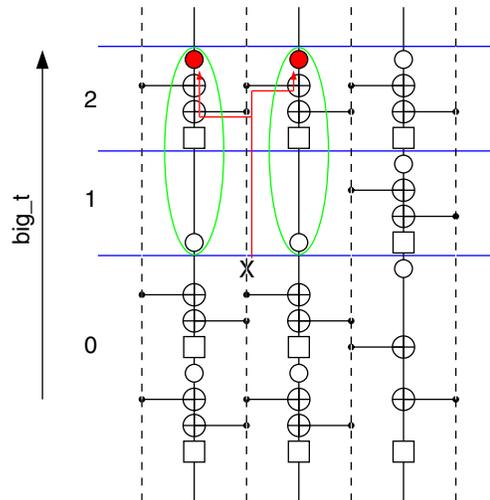

FIG. 17. Example of a single error leading to two detection events two big_t in the future. A detection event has a big_t equal to that of its latest measurement. big_t increments after all stabilizers have been measured at least once (circles represent measurement, squares represent initialization).

measurement circuits with the property that single errors do not lead to the creation of more than one pair of primal and one pair of dual detection events. This ensures that minimum-weight perfect matching [41,42] is well suited to correcting errors generated during the execution of the circuits.

4. Nests of balls and sticks

When a set is processed to generate detection events, it is said to be finalized. At this point in time, a ball is generated and associated with the finalized set. This ball represents a location in space-time. A pair of detection events leads to the creation of a stick between its associated balls. A stick represents a potential connection between a pair of space-time locations arising from a single error. Many single errors can lead to the same stick [6]. The probability of a given stick, given a list of errors leading to it each with its own probability (e_i, p_i), is therefore, to first order

$$p_{\text{stick}} = \sum_i p_i \prod_{j \neq i} (1 - p_j). \quad \text{(B8)}$$

We have a simple Blender-based visualization tool for nests, which is used to generate Figs. 7–9 and 14. A nest contains the full tracking information of which gate leads to which collection of errors and which of those errors leads to which sticks. This is highly useful, however, it is also computationally cumbersome. AUTOTUNE deletes all errors, measurements, sets, detection events, and balls and sticks when they are no longer required, which keeps total memory required finite; however, it remains challenging to generate the nest fast enough to obtain good statistics in simulations, let alone keep pace with a real quantum computer. Note that nest generation is computationally efficient in the computer science sense, with constant memory and $O(n^2)$ time required to generate each layer of the nest on n^2 qubits; however, mere efficiency is insufficient for practical purposes.

5. Exploiting regularity

Motivated by the difficulty of rapidly generating nests, we first cut down the data stored therein to the minimum required by the minimum-weight perfect matching algorithm. For every ball in a nest we create a dot, which again simply represents a space-time location. For every stick, we create a line. Lines connect the dots corresponding to the balls that the stick connected; however, they contain just one number, the weight $w = -\ln(p_{\text{stick}})$.

The lattice is kept for much longer than the nest. With low probability, the matching algorithm can require all prior matching history to correctly match the latest data. A sufficiently long history must be kept to ensure that the probability of requiring more is negligible. This is possible as the probability of requiring additional data in the past decreases exponentially with kept history size. The simplicity of the lattice keeps the memory required low.

All the nests shown have a great deal of regularity in their structure. This can be exploited to enable direct generation of lattices, avoiding the need to generate expensive nests on the fly. During a boot-up phase, AUTOTUNE analyzes each new stick and stores new unique sticks as an offset that contains only the geometric information associated with the stick—just enough to create a line. Furthermore, each new ball is analyzed to determine if it corresponds to a new pattern of offsets. Such unique patterns are stored as blocks of offsets. Finally, each round of error detection is analyzed with structurally unique rounds stored as layers of blocks. All data is then stored in a recipe that contains all necessary information to rapidly create any part of the lattice.

Currently, AUTOTUNE is capable of analyzing either error detection circuitry that eventually leads to identical repeated layers or a finite number of rounds of error detection that can have any structure whatsoever. The former could easily be extended to a finite number of cyclically repeated layers; however, it remains unclear whether one could avoid generating full nests when simulating probabilistic error detection in which all syndrome measurements can take a randomly variable amount of time.

The performance of AUTOTUNE has been described elsewhere [43]. Its complexity is optimal, requiring only $O(n^2)$ time to simulate and perform the necessary classical processing associated with each round of error detection on an array of n^2 qubits. Given constant computing power per unit area, all algorithms within AUTOTUNE can be parallelized to $O(1)$. This optimal parallelization is a direct consequence of the topological nature of the codes used and the fact that, on average, this implies that one only needs local information to correctly process a given space-time region of measurement results. The local runtime does not depend in any way on the global size of the computer.

---

[1] P. W. Shor, in *Proc. 35th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society Press, Los Alamitos, CA, 1994), pp. 124–134.

[2] L. K. Grover, in *Proc. Twenty-Eighth Annual ACM Symposium on the Theory of Computing* (ACM Press, New York, 1996), pp. 212–219.

[3] S. Lloyd, *Universal Quantum Simulators*, Science **273,** 1073 (1996).

[4] V. Subramaniam and P. Ramadevi, *Quantum Computation of Jones' Polynomials*, arXiv:quant-ph/0210095.

[5] Stephen Jordan, Quantum Algorithm Zoo, http://math .nist.gov/quantum/zoo/.

[6] D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg, *Surface Code Quantum Computing with Error Rates over 1%*, Phys. Rev. A **83,** 020302(R) (2011).

[7] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, *Towards Practical Classical Processing for the Surface Code*, Phys. Rev. Lett. **108,** 180501 (2012).

[8] T. Monz, P. Schindler, J. T. Barreiro, M. Chwalla, D. Nigg, W. A. Coish, M. Harlander, W. Hänsel, M. Hennrich, and R. Blatt, *14-Qubit Entanglement: Creation and Coherence*, Phys. Rev. Lett. **106**, 130506 (2011).

[9] D. Jaksch, *Optical Lattices, Ultracold Atoms and Quantum Information Processing*, Contemp. Phys. **45**, 367 (2004).

[10] P. W. Shor, *Scheme for Reducing Decoherence in Quantum Computer Memory*, Phys. Rev. A **52**, R2493 (1995).

[11] A. R. Calderbank and P. W. Shor, *Good Quantum Error-Correcting Codes Exist*, Phys. Rev. A **54**, 1098 (1996).

[12] A. M. Steane, *Multiple Particle Interference and Quantum Error Correction*, Proc. R. Soc. A **452**, 2551 (1996).

[13] E. Knill, *Quantum Computing with Realistically Noisy Devices*, Nature (London) **434**, 39 (2005).

[14] D. Bacon, *Operator Quantum Error-Correcting Subsystems for Self-Correcting Quantum Memories*, Phys. Rev. A **73**, 012340 (2006).

[15] L. C. L. Hollenberg, A. D. Greentree, A. G. Fowler, and C. J. Wellard, *Two-Dimensional Architectures for Donor-Based Quantum Computing*, Phys. Rev. B **74**, 045311 (2006).

[16] S. J. Devitt, A. G. Fowler, A. M. Stephens, A. D. Greentree, L. C. L. Hollenberg, W. J. Munro, and Kae Nemoto, *Architectural Design for a Topological Cluster State Quantum Computer*, New J. Phys. **11**, 083032 (2009).

[17] D. P. DiVincenzo, *Fault Tolerant Architectures for Superconducting Qubits*, Phys. Scr. **T137**, 014020 (2009).

[18] D. A. Herrera-Martí, A. G. Fowler, D. Jennings, and T. Rudolph, *A Photonic Implementation for the Topological Cluster State Quantum Computer*, Phys. Rev. A **82**, 032332 (2010).

[19] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, *Layered Architecture for Quantum Computing*, Phys. Rev. X **2**, 031007 (2012).

[20] R. Stock and D. F. V. James, *A Scalable, High-Speed Measurement-Based Quantum Computer Using Trapped Ions*, Phys. Rev. Lett. **102**, 170501 (2009).

[21] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, *Large Scale Modular Quantum Computer Architecture with Atomic Memory and Photonic Interconnects*, arXiv:1208.0391.

[22] X.-C. Yao, T.-X. Wang, H.-Z. Chen, W.-B. Gao, A. G. Fowler, R. Raussendorf, Z.-B. Chen, N.-L. Liu, C.-Y. Lu, Y.-J. Deng, Y.-A. Chen, and J.-W. Pan, *Experimental Demonstration of Topological Error Correction*, Nature (London) **482**, 489 (2012).

[23] S. B. Bravyi and A. Yu. Kitaev, *Quantum Codes on a Lattice with Boundary*, arXiv:quant-ph/9811052.

[24] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, *Topological Quantum Memory*, J. Math. Phys. (N.Y.) **43**, 4452 (2002).

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

[26] R. Raussendorf, J. Harrington, and K. Goyal, *Topological Fault-Tolerance in Cluster State Quantum Computation*, New J. Phys. **9**, 199 (2007).

[27] A. G. Fowler, A. M. Stephens, and P. Groszkowski, *High Threshold Universal Quantum Computation on the Surface Code*, Phys. Rev. A **80**, 052312 (2009).

[28] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, *A Primer on Surface Codes: Developing a Machine Language for a Quantum Computer*, Phys. Rev. A **86**, 032324 (2012).

[29] D. Gottesman, Ph.D. thesis, Stabilizer Codes and Quantum Error Correction, California Institute of Technology, 1997, arXiv:quant-ph/9705052.

[30] R. Raussendorf, J. Harrington, and K. Goyal, *A Fault-Tolerant One-Way Quantum Computer*, Ann. Phys. (N.Y.) **321**, 2242 (2006).

[31] A. G. Fowler and K. Goyal, *Topological Cluster State Quantum Computing*, Quantum Inf. Comput. **9**, 721 (2009).

[32] R. Raussendorf and H.-J. Briegel, *A One-Way Quantum Computer*, Phys. Rev. Lett. **86**, 5188 (2001).

[33] R. Raussendorf, D. E. Browne, and H.-J. Briegel, *Measurement-Based Quantum Computation on Cluster States*, Phys. Rev. A **68**, 022312 (2003).

[34] D. S. Wang, A. G. Fowler, C. D. Hill, and L. C. L. Hollenberg, *Graphical Algorithms and Threshold Error Rates for the 2d Colour Code*, Quantum Inf. Comput. **10**, 780 (2010).

[35] G. Duclos-Cianci and D. Poulin, *Fast Decoders for Topological Quantum Codes*, Phys. Rev. Lett. **104**, 050504 (2010).

[36] G. Duclos-Cianci and D. Poulin, in *Information Theory Workshop (ITW), 2010*, (IEEE, New York, 2010), pp. 1–5.

[37] A. J. Landahl, J. T. Anderson, and P. R. Rice, *Fault-Tolerant Quantum Computing with Color Codes*, arXiv:1108.5738.

[38] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, *Strong Resilience of Topological Codes to Depolarization*, Phys. Rev. X **2**, 021004 (2012).

[39] J. R. Wootton and D. Loss, *High Threshold Error Correction for the Surface Code*, arXiv:1202.4316 [Phys. Rev. Lett. (to be published)].

[40] P. Sarvepalli and R. Raussendorf, *Efficient Decoding of Topological Color Codes*, Phys. Rev. A **85**, 022317 (2012).

[41] J. Edmonds, *Paths, Trees and Flowers*, Can. J. Math. **17**, 449 (1965).

[42] J. Edmonds, *Maximum Matching and a Polyhedron with 0,1-Vertices*, J. Res. Natl. Bur. Stand. Sect. A **69**, 125 (1965).

[43] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, *Towards Practical Classical Processing for the Surface Code: Timing Analysis*, arXiv:1202.5602 [Phys. Rev. A (to be published)].

[44] D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg, *Surface Code Quantum Computing with Error Rates over 1%*, Phys. Rev. A **83**, 020302(R) (2011).

[45] A. G. Fowler, *Proof of Finite Surface Code Threshold for Matching*, arXiv:1206.0800 [Phys. Rev. Lett. (to be published)].

[46] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg, *Threshold Error Rates for the Toric and Planar Codes*, Quantum Inf. Comput. **10**, 456 (2010).

[47] A. G. Fowler, D. S. Wang, and L. C. L. Hollenberg, *Surface Code Quantum Error Correction Incorporating Accurate Error Propagation*, Quantum Inf. Comput. **11**, 8 (2011).

[48] A. G. Fowler, *Analytic Asymptotic Performance of Topological Codes*, arXiv:1208.1334.