Fast fault-tolerant decoder for qubit and qudit surface codes

Fern H. E. Watson,[1,2,*] Hussain Anwar,[3] and Dan E. Browne[1]

[1]*Department of Physics and Astronomy, University College London, Gower Street, London, WC1E 6BT England, United Kingdom*

[2]*Department of Physics, Imperial College London, Prince Consort Road, London, SW7 2AZ England, United Kingdom*

[3]*Department of Mathematical Sciences, Brunel University, Uxbridge, Middlesex, UB8 3PH England, United Kingdom*

(Received 23 February 2015; published 8 September 2015)

The surface code is one of the most promising candidates for combating errors in large scale fault-tolerant quantum computation. A fault-tolerant decoder is a vital part of the error correction process—it is the algorithm which computes the operations needed to correct or compensate for the errors according to the measured syndrome, even when the measurement itself is error prone. Previously decoders based on minimum-weight perfect matching have been studied. However, these are not immediately generalizable from qubit to qudit codes. In this work, we develop a fault-tolerant decoder for the surface code, capable of efficient operation for qubits and qudits of any dimension, generalizing the decoder first introduced by Bravyi and Haah [Phys. Rev. Lett. **111**, 200501 (2013)]. We study its performance when both the physical qudits and the syndromes measurements are subject to generalized uncorrelated bit-flip noise (and the higher-dimensional equivalent). We show that, with appropriate enhancements to the decoder and a high enough qudit dimension, a threshold at an error rate of more than 8% can be achieved.

DOI: 10.1103/PhysRevA.92.032309 PACS number(s): 03.67.Pp, 03.67.Lx, 05.10.Cc

I. OVERVIEW

Topological quantum codes built from qubits [two-dimensional (2D) quantum systems] play a central role in architectures for fault-tolerant quantum computing at the forefront of current research [1–4]. The surface code [5] and the related toric code [6,7] are prominent examples of such codes. Compared with other quantum error correcting codes, they posses the key experimental benefit of requiring only local interactions and yet, under realistic noise models, they have been shown to achieve the highest reported fault-tolerant thresholds [8,9].

Recent developments have shown that employing d-dimensional quantum systems, or qudits, as the building blocks for fault-tolerant schemes may offer some important advantages. For example, an integral part of many fault-tolerant schemes is the distillation of magic states [10]—a procedure necessary to achieve universal computation—where generalization to higher dimensions has resulted in improved distillation thresholds and lower overheads in the number of qudit magic states [11–13]. Moreover, threshold investigations of the qudit toric code with noise-free syndrome measurements have shown that, for a standard independent noise model, the error correction threshold increases significantly with increasing qudit dimension [14–16], although we caution that it is difficult to fairly compare noise rates between systems of different dimensions. Although it is more challenging to realize qudit quantum systems experimentally, recent work has demonstrated the ability to coherently control and perform operations in single 16-dimensional atomic systems with high fidelity [17,18], with the implementation of high-fidelity multiqudit interactions still to be achieved.

A surface code is a stabilizer code with local stabilizer generators. Qudits are associated with the edges of a 2D square lattice. In order to store the encoded information for an arbitrary length of time, active error detection must be performed periodically in order to prevent the errors from accumulating beyond the capability of the code to correct them (see Fig. 1). In every round of error correction all the stabilizer generators are measured to obtain the *syndrome*. The syndrome is then processed by the *decoder*—the classical algorithm that outputs a correction operator. In a realistic environment both the physical systems and the stabilizer measurements are prone to errors, and hence the decoder must be able to take both of these types of errors into account [7,19].

Decoders are often developed for the simpler case where measurement error is neglected. However, there is a well-established and elegant method for generalizing measurement-noise-free decoders for topological codes to the fully fault-tolerant setting [7]. The noisy syndrome measurements are repeated, extending the two-dimensional surface representing the code to a three-dimensional (3D) data structure, where time represents an extra dimension. Remarkably, the change from two to three dimensions allows most decoder algorithms developed for noise-free measurements to be applied largely unchanged in this more general setting.

The most widely used decoding algorithm for topological codes remains the minimum-weight perfect matching algorithm (MWPMA). However, this algorithm has a number of disadvantages. For a distance L surface code, with error-free measurements, the run time for a basic implementation of the MWPMA scales with $O(L^6)$, and for error-prone measurements this run time increases to $O(L^9)$. A more refined fault-tolerant implementation for the qubit surface code scales with $O(L^2)$ [20], and under certain assumptions a run time complexity that is independent of L can be attained [21]. Nevertheless, the main disadvantage of the MWPMA is that it is not suitable for qudit surface codes with $d > 2$. For these reasons, the development of alternative decoding algorithms is currently a very active research area [14,15,22–27].

In this work, we introduce a fault-tolerant decoding algorithm which overcomes both of the disadvantages of the MWPMA. The algorithm, which extends the hard-decision

*fern.watson10@imperial.ac.uk

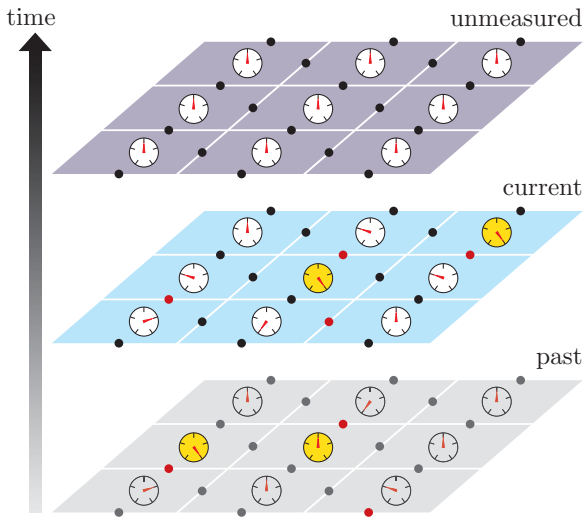

FIG. 1. (Color online) An illustrative picture of the data structure obtained in order to perform fault-tolerant error correction. Each layer represents a single time step where all the stabilizers are measured (only plaquettes are shown here for clarity—depicted by the meter with multiple outcomes) to obtain the syndrome. The yellow meters (dark gray) represent locations where an error has occurred in the measurement procedure itself. After a specified number of time steps a full 3D history of the syndromes will have been be collected. If operating below threshold the decoder then uses this data to infer a correction operator that returns the code to its original state with high probability.

renormalization group (HDRG) decoder proposed by Bravyi and Haah [28], has a fast typical run time of $O(L^3)$ and can be applied to qudit surface codes of any dimension d.

For a given noise model, the error threshold represents an upper bound on the noise level for which increasing the code distance increases the probability of successful error correction. We denote the threshold for a given qudit dimension by $p_{\text{th}}^{(d)}$. A widely studied qubit error model (described below in more detail) is the simple *uncorrelated noise* model where X and Z Pauli errors on individual code qubits and bit-flip errors on the syndrome measurement outcomes each occur independently with probability p. For this noise model, the optimal threshold for the qubit toric code is known to be 3.3% [29] while the threshold obtained with the MWPMA decoder is 2.9% [9,19,30].

The HDRG decoder we study here attains a threshold of $p_{\text{th}}^{(2)} = 2.2\%$ for the qubit code and may also be used with qudit surface codes of any dimension. For the qudit generalization of the uncorrelated noise model (introduced below), the decoder achieves a threshold value which increases monotonically with the qudit dimension d, until it reaches a saturated value of around 4.2%.

We show that this saturating behavior is due to a syndrome percolation effect which upper bounds the achievable threshold. To overcome the percolation threshold we have constructed a procedure executed *before* running the HDRG, which we call the *initialization step* [15]. The algorithm implemented in this "pre-decoding" step disrupts the syndrome percolation and boosts the threshold to 8.3% for sufficiently high qudit dimension. We call the HDRG decoder when augmented with the initialization step the *enhanced*-HDRG decoder.

The structure of this paper is as follows. We start in Sec. II by reviewing the properties of the qudit surface code and fixing our notation. In Sec. III we give a formal description of the noise model investigated and describe how our numerical simulations were performed. In Sec. IV we present our different variations of the HDRG decoder for the fault-tolerant setting, along with the thresholds we obtain. We conclude in Sec. V.

II. THE QUDIT SURFACE CODE

The qudit surface code is the natural higher-dimensional generalization of the qubit code. This generalization is already present in Kitaev's seminal paper [6] and has been written about extensively elsewhere [7,19,31–33]. For completeness, however, we shall provide an overview of qudit stabilizer codes and the qudit surface code.

We express the computational basis for a single qudit as the set of states $|\alpha\rangle$ where $\alpha \in \mathbb{Z}_d$, and where the d-element cyclic group $\mathbb{Z}_d = \{0, \ldots, d-1\}$ can be conveniently identified with addition over integers modulo d. The conventional single qubit Pauli operators have natural generalizations:

$$X = \sum_{j\in\mathbb{Z}_d} |j \oplus 1\rangle\langle j|, \quad Z = \sum_{j\in\mathbb{Z}_d} \omega^j |j\rangle\langle j|, \tag{1}$$

where $\omega = e^{2\pi i/d}$ and the addition $\oplus$ is taken to be modulo d. Notice that these unitary operators are no longer Hermitian when $d > 2$, but they posses orthogonal eigenspaces with eigenvalues of the form ω^j, for some j. Hence, we can still interpret them as physical observables with measurement outputs labeled by their complex eigenvalues. As a shorthand we will often abbreviate an outcome ω^j simply by its exponent j.

The qudit Pauli operators obey the commutation relation $X^j Z^k = \omega^{-jk} Z^k X^j$ for arbitrary $j,k \in \mathbb{Z}_d$. They generate the single qudit Pauli group $\mathcal{P}_d = \langle X, Z\rangle$ up to a global phase. The n-qudit Pauli group $\mathcal{P}_d^n$ is the n-fold tensor product of the single qudit Pauli group $\mathcal{P}_d^{\otimes n}$. The code space of a stabilizer code is defined as the "+1" eigenspace of an Abelian subgroup $\mathcal{S} \in \mathcal{P}_d^n$, such that $\omega^j \mathbb{1} \notin \mathcal{S}$ for nonzero j. The elements of $\mathcal{S}$ are called the *stabilizers* of the code. A set of generators of $\mathcal{S}$ is identified as the syndrome measurement operators for the code.

In a surface code qudits are identified with the edges of an $L \times L$ lattice with boundaries as shown in Fig. 2. The surface code is a stabilizer code with two types of stabilizer generators $\mathcal{S} = \langle A_s, B_p\rangle$ defined on the lattice as

$$A_s = X_e \otimes X_e^{-1} \otimes X_e^{-1} \otimes X_e \quad \forall\, e \in V, \tag{2}$$

$$B_p = Z_e \otimes Z_e \otimes Z_e^{-1} \otimes Z_e^{-1} \quad \forall\, e \in P, \tag{3}$$

where $e \in V$ are the edges surrounding a vertex V of the lattice and $e \in P$ are the edges surrounding a plaquette P. We refer to A_s as the *vertex* operators, and to B_p as the *plaquette* operators. An example of each is shown in Figs. 2(a) and 2(b). Note that the two boundary different types ("rough" and "smooth") of the lattice lead to deformations of plaquette and vertex operators at the boundary, respectively.

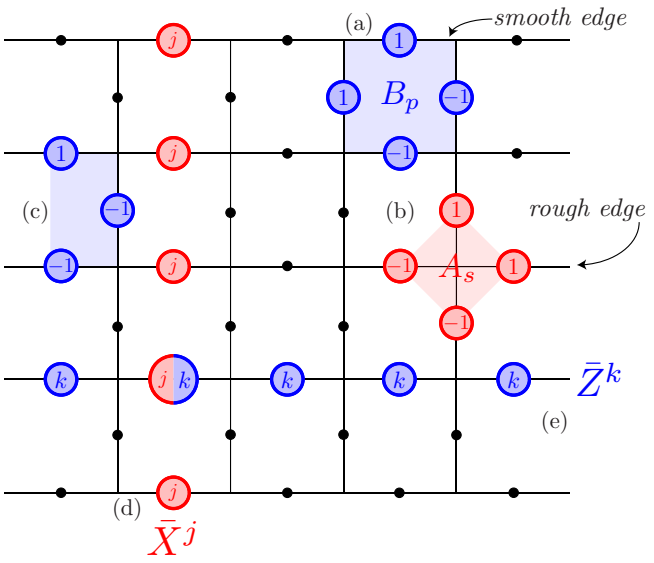

FIG. 2. (Color online) An example of a distance 5 surface code. Qudits are shown as black dots, arranged on the edges of a lattice with two types of boundary: rough and smooth. For clarity, when an arbitrary X^j or Z^k Pauli operator acts on a physical qudit, we only include the exponents j and k on the edges of the figure. We use red for X^j errors and vertex operators, and blue for Z^k errors or plaquette operators. (a) and (b) An example of a single plaquette and vertex operator, respectively. (c) An example of a deformed rough edge plaquette operator (three-body operator). Note that the vertex operators are deformed at smooth edges. (d) and (e) An example of a pair of anticommuting logical operators.

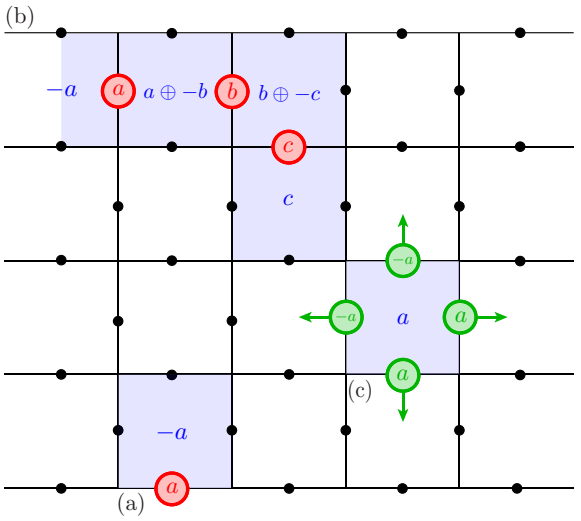

FIG. 3. (Color online) Examples of X-type errors and the syndrome transportation rule. (a) A single boundary error is only detected by one plaquette. (b) An arbitrary string of three errors and the corresponding intermediate plaquette measurement outcomes. (c) An example of how to transport the plaquette with outcome a in any of the indicated directions by applying the relevant X-type operator shown in green.

The surface code supports one logical qudit. The logical operators for the qudit are defined by stringlike X (or Z) operators. The logical $\bar{X}$ operators connect the two opposing smooth edges, whereas the logical $\bar{Z}$ operators connect the two rough edges. An example of each is shown in Figs. 2(d) and 2(e). These operators, together with the stabilizer group, generate the group of Pauli operators which map the code space to itself. We denote this group of logical operators $\mathcal{L}$. We denote the set of logical operators which do not leave the code space invariant as $\mathcal{L} - \mathcal{S}$. The distance of a topological code corresponds to the length of shortest possible logical operator, i.e., the distance is L.

Errors that occur on the qudits are detected by measuring the neighboring stabilizers, with X-type and Z-type errors detected independently by the plaquette and the vertex operators, respectively. This allows us to restrict the discussion to X-type errors since results for Z-type errors will be analogous. A single X-type error is detected by two adjacent plaquettes, except when it occurs on a smooth boundary [see Fig. 3(a)]. In general, a string of X-type errors is detected by plaquettes contiguously along the path of the string, as shown by the example in Fig. 3(b). This is in contrast to the qubit case ($d = 2$) where only the end points of the string give rise to nontrivial plaquette measurements, a situation that can also arise for general d when the errors along the string possess identical errors. This observation suggests that in higher d the syndrome reveals more information about the path of the errors on the lattice. Indeed it is this information that, if exploited correctly by the decoder, can lead to improved error correction performance, as shown by their higher threshold values, as d increases.

We also introduce the concept of syndrome *transportation*: a syndrome can be transported in any direction by applying the appropriate operator as illustrated by the example in Fig. 3(c). Moreover, by transporting one syndrome to the location of a second, they are *fused* into a single syndrome such that their charges are added (modulo d). These concepts will be useful in Sec. IV when describing our decoder.

Generally speaking, the aim of the decoder is to use the information given by the syndrome to return a correction operator that restores the code to its original state. More formally, let us denote an arbitrary configuration of X-type errors on the 2D surface code by the set $\mathbf{e}$, and the corresponding plaquette measurement outcomes by the set $\mathbf{s} = \{s_{x,y}\}$, where $s_{x,y} \in \mathbb{Z}_d$ is the outcome of the measurement and the subscripts x and y are the coordinates of the plaquettes, so that $1 \leqslant x \leqslant L$ and $1 \leqslant y \leqslant (L-1)$. We will often refer to the outcome $s_{x,y}$ as the *charge* of the measurement. Then we say that a decoder $\mathcal{D}$ takes in the syndrome $\mathbf{s}$ and returns a correction configuration $\mathbf{f}$. We denote this map by $\mathcal{D}(\mathbf{s}) \to \mathbf{f}$. The decoder succeeds if $\mathbf{e} \otimes \mathbf{f} \in \mathcal{S}$ and fails if $\mathbf{e} \otimes \mathbf{f} \in \mathcal{L} - \mathcal{S}$.

In the next section we will give a formal description of the noise model and describe the method for the fault-tolerant simulation.

III. THE NOISE MODEL AND SIMULATION METHODS

In the literature it is common to test fault-tolerant decoders with a simple error model described by a single parameter p. For ease of comparison, we shall follow this convention and use the same error model here. Although this model is not likely to be particularly close to the noise which occurs in

physical systems, it has the advantage that it allows X- and Z-type errors and their correction to be modeled independently. It is thus the standard noise model used to benchmark new decoders.

Between each round of syndrome measurements we assume that each physical qudit is independently subject to an error channel which applies error operator X^k such that $1 \leqslant k \leqslant (d-1)$ with equal probability $p/(d-1)$, followed by an error channel which applies error operator Z^k such that $1 \leqslant k \leqslant (d-1)$ with equal probability $p/(d-1)$. We then assume that the outcome of each syndrome measurement j undergoes an error which maps j to $j \oplus k$ for $1 \leqslant k \leqslant (d-1)$ with equal probability $p/(d-1)$. Since X-type errors, Z-type errors, and measurement errors are uncorrelated this is often called the *uncorrelated noise model*.

We estimate the threshold via a Monte Carlo simulation. We shall study a distance L code for a variety of values of L. This corresponds to an $L \times L$ surface code grid. For simplicity, we shall let the number of time steps in our simulation also equal L.

The simulation proceeds by first generating a 3D data structure of L time steps of the accumulated history of the physical qudit errors and the measurement errors. The corresponding syndrome measurement outcomes, taking into account both of these error sources, are then computed.

In order to achieve the close analogy for the relationship between errors and syndromes in the 2D measurement-error-free and 3D general case, Dennis *et al.* [7] showed that it is most convenient to represent the history of the syndrome outcomes as a 3D grid of syndrome *changes*.

Let us denote $\mathbf{s}_t$ as the set of syndrome outcomes at the tth time step. The set of syndrome changes $\mathbf{s}'_t$ at time step t is then defined as the elementwise difference, modulo d, of $\mathbf{s}_t$ and $\mathbf{s}_{t-1}$, i.e., $\mathbf{s}'_t = \mathbf{s}_t \ominus \mathbf{s}_{t-1}$, where $\ominus$ denotes subtraction modulo d and we assume that $\mathbf{s}'_1 = \mathbf{s}_1$. Each set of syndrome changes corresponds to a 2D grid of integers, and we combine these grids into a 3D cubic structure with $t = 1$ at the bottom and $t = L$ at the top. We call this grid the syndrome changes history and denote it $\mathbf{S}'$. It is convenient to introduce a Cartesian coordinate system to refer to the elements of $\mathbf{S}'$, i.e., $s_{t,x,y}$ corresponds to the syndrome change at grid point (x,y) at time step t.

The input to the decoder is the 3D syndrome *changes* history $\mathbf{S}' = \{\boldsymbol{s}_1, \boldsymbol{s}_2 \ominus \boldsymbol{s}_1, \ldots, \boldsymbol{s}_L \ominus \boldsymbol{s}_{L-1}\}$. The decoder takes the syndrome changes history and returns a 3D correction operator $\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_L\}$. To convert this to a physical correction operator that can be applied in two dimensions we ignore timelike edges and combine the two-dimensional layers corresponding to each time step, to form a 2D correction operator $\tilde{\mathbf{f}}$ that corrects the accumulated errors at the last time step of the surface code.

In other words, the resultant correction operator, $\tilde{\mathbf{f}}$, is the sum (modulo d) of the correction at each qudit location at each time step, i.e., $\tilde{\mathbf{f}} = \otimes_t \mathbf{f}_t$. We say the decoder has succeeded when the product of the accumulated errors on the qudits and the returned correction operator is within the stabilizer of the code.

If we are operating below threshold then following the 3D decoding we expect almost all of the errors to have been corrected. There is a finite probability, however, that some small number of errors will remain after the fault-tolerant decoding has been performed. In a realistic setting the error correction would proceed in this way, eliminating all but a small number of errors in each block of L time steps. At the point when the state is read out, these small errors can be accounted for by taking a majority vote on the measurements of the logical operators.

For the purposes of the simulation, however, we need to determine whether the fault-tolerant decoder has introduced a logical error. The conventional way to overcome this problem is to perform an additional round of error correction in two dimensions with noise-free syndrome measurements, after which we can be certain that all the errors are corrected and a parity check will reveal whether any logical errors have been introduced.

IV. HDRG DECODER WITH NOISY SYNDROMES

The HDRG decoder has a simple motivation behind its construction: when the error rate is sufficiently low we expect any errors arising on the surface code lattice to be sparse. This in turn means that syndromes are likely to occur in small, well-separated clusters. The HDRG decoder aims to identify clusters of syndromes generated by such local errors and correct them locally within each cluster. If these clusters have been correctly identified, and the clusters are each small enough that they do not span the lattice, then this strategy results in the decoder computing a correction operator that will correct all errors with high probability. In this section we shall give a formal definition of these concepts in the fault-tolerant setting.

A. Decoder construction

The main concept required for the description of the HDRG decoder is that of a *metric*—a geometric distance function between any pair of elements of a set. In our case, we wish to associate a metric between pairs of syndromes in the set $\mathbf{S}'$. The metric we use is the Manhattan distance, denoted here by δ, which maps two syndromes as follows:

$$\delta(s_{t,x,y}, s_{t',x',y'}) = |t'-t| + |x'-x| + |y'-y|. \quad (4)$$

See Fig. 4 for an illustration for how the region defined by this metric grows.

We say that two syndromes are δ-*connected* if the distance between them is less than or equal to δ. For a given metric value δ, we define a cluster $\mathcal{C}$ to be the set nontrivial syndromes

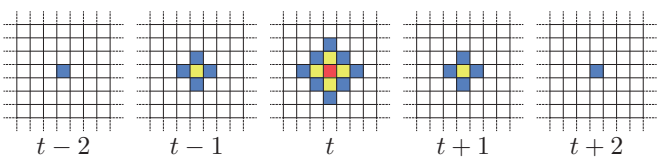

FIG. 4. (Color online) An illustration of the Manhattan distance metric. The figure shows five time steps from the syndrome changes history. The green (light gray) plaquettes are 1-connected to the central red plaquette (central plaquette at time step t, medium gray). Blue (dark gray) and green plaquettes are 2-connected to the central plaquette.

such that every syndrome within the cluster is δ-connected to at least one other syndrome within that cluster. It is easy to see that for a fixed δ the syndrome changes history $\mathbf{S}'$ can always be partitioned into a set of disjoint clusters such that $\mathbf{S}' = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_n$, for some integer n.

Associated to every cluster is a total charge $\oplus_{\mathcal{C}}\, s_{t,x,y}$, where the summation is performed modulo d. If this charge is zero, we call the cluster *neutral*. Such a cluster can be annihilated by fusing all of the syndromes contained in the cluster locally, meaning that the Pauli correction operator will have support only within the cluster. If the charge is nonzero but the cluster is δ-connected to any of the three smooth boundaries (two spatial and one time) then we call the cluster *boundary neutral*. Clusters of this type can be annihilated by fusing the syndromes locally and then connecting the remaining charge to the boundary it overlaps.

The HDRG decoder involves multiple levels of decoding to fuse together all the elements in $\mathbf{S}'$ and return the resultant correction operator. Every decoding level ℓ is associated with a distance determining the connectivity of the disjoint clusters at that level. For the metric we have defined we will use $\delta = 2^{\ell}$ starting with $\ell = 0$. This means that the cluster connectivity increases exponentially as we increase the decoding levels. At each level, only the neutral (and boundary-neutral) clusters are fused, leaving any charged clusters to be combined to form neutral clusters at subsequent levels.

The decoding procedure can now be summarized as follows, starting with $\ell = 0$.

(1) Clustering: Identify all the disjoint δ-connected clusters at level ℓ.

(2) Neutral annihilation: Fuse each neutral and boundary-neutral cluster locally and return a correction operator.

(3) Renormalize: If there are clusters that are not annihilated, then increment ℓ by 1 and return to step 1.

The decoder stops when there are no nontrivial syndromes remaining. The crucial feature of this decoder is that part of the total correction operator is fixed after each level of decoding. In classical coding theory, decoding algorithms exhibiting such a feature are referred to as *hard-decision* decoders. An explicit example for a small lattice simulation is illustrated in Appendix A 1.

B. The run time of the HDRG decoder

The dominant parts of our decoder algorithm that contribute to the run time complexity are the identification of the δ-connected cluster of syndromes (clustering) and the determination of the Pauli operator that eliminates the syndrome (fusion). We shall look at each of these processes in turn and argue that for lower error rates we expect a run time scaling of $O(L^3)$ and even in the worst case this scaling will be no greater than $O(L^6)$.

Let us first consider the limit in which error rates are low and the errors are extremely sparse. In the clustering part of the algorithm at a given level ℓ, the algorithm searches a constant number of plaquettes $O(2^{3\ell})$ around every nontrivial syndrome.

In the case of extremely sparse syndromes the total number of syndromes is $O(L^3)$ and the decoder will only need to run at the first level $\ell = 1$. Thus, in this limit, the dependence of the run time complexity on L for this part of the algorithm will be $O(L^3)$.

In the worst case scenario we consider the most pessimistic estimates for the clustering step of the algorithm. In this case the decoder will run the maximum number of levels $\ell = O(\log_2 L)$. There will be $O(L^3)$ syndromes and the dependence of the run time complexity on L for this part of the algorithm will thus be

$$\sum_{\ell=0}^{O(\log_2 L)} 2^{3\ell} L^3 \sim 2^{3\,O(\log_2 L)} L^3 \sim O(L^6), \tag{5}$$

to leading order.

For the fusion part of the algorithm, the syndromes can all be moved to a single point in the box enclosing the cluster. This will take a time that scales with the size of the enclosing box, and the maximum size of the box scales with L^3. Note that, since the time complexity of modular arithmetic is independent of modulus d, this scaling is independent of d.

C. Thresholds estimation and percolation limitation

To estimate the threshold we simulate the entire process of generating L time steps of errors and noisy syndromes, followed by decoding the syndromes. The simulation was done for $N = 10^4$ runs, and repeated for a range of lattice sizes L and error rates p.

We determine the threshold $p_{\text{th}}^{(d)}$ using a rescaling method [19,34]. Selecting data close to the point where the curves of different L cross (for fixed qudit dimension) we perform a fit to a function of the form

$$P_{\text{succ}}(x) = A + Bx + Cx^2 + DL^{-1/\mu}, \tag{6}$$

where $x = (p - p_{\text{th}})L^{1/\nu}$, and the final term in the sum represents a finite-size correction to the fitting.

The success probability of the decoder for the qubit case is shown in Fig. 5, where we find a threshold value of $p_{\text{th}}^{(2)} = 0.0215 \pm 0.0006$. This allows us to directly compare our decoder with other fault-tolerant qubit decoders; for example, the soft-decision renormalization-group decoder by

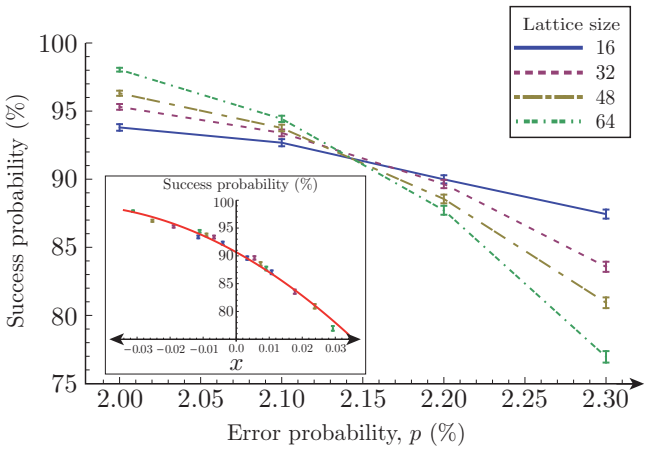

FIG. 5. (Color online) An example for the collected simulation data used to estimate the threshold for qubits. The inset figure shows the fitting of the function $P_{\text{succ}}(x) = A + Bx + Cx^2 + DL^{-1/\mu}$ to the rescaled data.

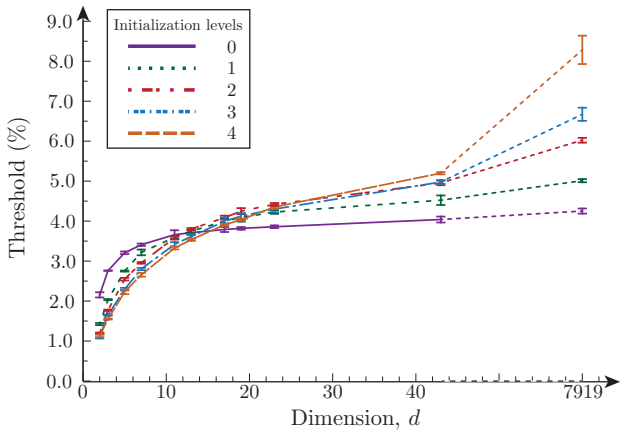

FIG. 6. (Color online) A summary of all qudit thresholds for different numbers of rounds of the initialization step. We have chosen the 1000th prime dimension ($d = 7919$) to represent the asymptotic limit. Although for small qudit dimensions the initialization step disrupts the syndrome too much and reduces the threshold, we see that in the asymptotic limit there is a clear advantage to using this technique.

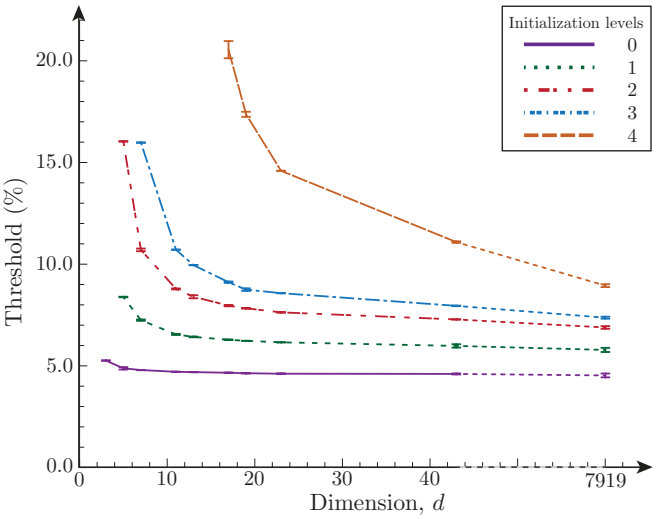

FIG. 7. (Color online) A summary of percolation thresholds for different numbers of rounds of the initialization step. The initialization step disrupts the percolation for low qudit dimensions, meaning that in some cases a threshold cannot be identified.

Duclos-Cianci and Poulin achieves a threshold of $p_{\text{th}}^{(2)} = 0.019 \pm 0.004$ [22].

Using the same technique of rescaling and fitting the function in Eq. (6) we can determine the threshold $p_{\text{th}}^{(d)}$ for further qudit dimensions. Although our HDRG decoder works for arbitrary qudit dimension d we consider the first few prime dimensions, and in order to determine the asymptotic behavior we also consider one very high qudit dimension, $d = 7919$, the 1000th prime number. The results are shown as the plot labeled "Initialization levels 0" in Fig. 6. The plot shows that the threshold achieved by the decoder increases monotonically with increasing qudit dimension, but quickly saturates to a maximum value of $p_{\text{th}}^{(7919)} = 0.042 \pm 0.09$. Previous work performed on the noiseless syndrome measurement version of the HDRG in [15] suggests that this saturation is due to a syndrome percolation effect.

In order to verify this hypothesis, we performed a simulation of the syndrome percolation threshold. This was done by generating the qudit noise and noisy syndrome measurements for each qudit dimension in the same way as for the decoder simulation. However, once the syndrome changes were calculated, we performed a check to determine whether any 1-connected clusters in $\mathbf{S}'$ percolated the lattice in the x or y directions. The t direction was not checked since we want to determine whether the percolating cluster is able to support a logical operator once it is collapsed to $\tilde{\mathbf{f}}$, and any stringlike operators in the t direction are unphysical. This information is summarized in the plot labeled "Initialization levels 0" in Fig. 7. The saturated value for the percolation thresholds for dimension 7919 is around 4.5%, agreeing with our prediction that the HDRG decoder thresholds are upper-bounded by the percolation threshold.

In the next section we show how to overcome this syndrome percolation effect and achieve improved qudit thresholds using an initialization step. This is an algorithm which is run before the HDRG to disrupt the percolating clusters.

D. Further enhancement

The initialization step is a subroutine that sweeps through all of the syndromes $\mathbf{S}'$ searching for *neutral subclusters* in order to disrupt the percolating clusters. Unlike the HDRG algorithm, the initialization step does not divide the observed syndrome into disjoint clusters, but simply identifies and eliminates neutral subclusters locally.

As with the decoder, the initialization step has "levels" defined by a metric. However, subclusters are more than δ-connected plaquettes; they are 1-connected *paths* of plaquettes, where the charge of the subcluster is counted along the entire path. This is because of the fact illustrated in Fig. 3, that a

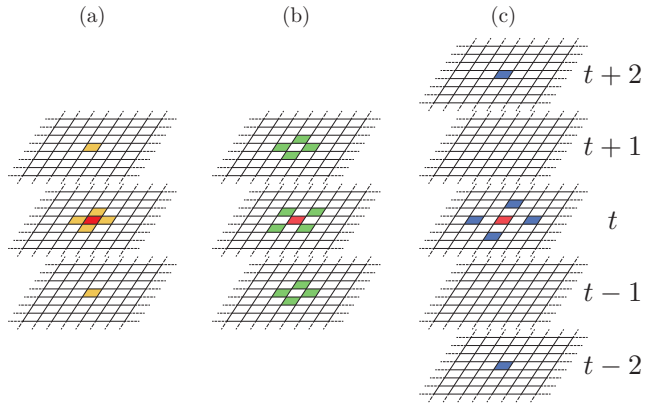

FIG. 8. (Color online) An illustration of the first three initialization levels. (a) First initialization level. Orange plaquettes (light gray) are 1-connected to the central red plaquette (central plaquette at time step t). (b) Second initialization level. Green plaquettes (medium gray) are 2-connected to the central plaquette and paths between the central plaquette and any green plaquette have a degeneracy of 2. (c) Third initialization level. Blue plaquettes (dark gray) are 2-connected to the central plaquette and paths between the central plaquette and any blue plaquette have a degeneracy of 1.

connected path of errors will result in a connected neutral path of syndromes.

An important idea needed to understand the initialization levels is that of *degeneracy* of paths. If there are $\pm h$ steps in the x direction, $\pm v$ steps in the y direction, and $\pm z$ steps in the t direction of the path then its degeneracy is given by

$$D = \frac{(h+v+z)!}{h!\,v!\,z!}. \tag{7}$$

The initialization levels are defined sequentially by distance from the central syndrome and the degeneracy of the paths, favoring those paths with equal distance but higher degeneracy as more likely. In Fig. 8 we show the outer syndromes of the first three initialization levels.

For a given syndrome $s_{t,x,y}$ we denote by $\mathcal{Q} = q_1, q_2, \ldots, q_n$ the set of syndromes with the same distance from $s_{t,x,y}$, and whose paths connecting to $s_{t,x,y}$ have the same degeneracy. Denote by p_i a possible path connecting $s_{t,x,y}$ to q_i and refer to each path as a subcluster at the initialization level k.

The *initialization step* $\mathcal{I}_k$ of *depth* k consists of running all the levels $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k$ where the initialization procedure $\mathcal{I}_j$ consists of the following steps, beginning with the first nontrivial syndrome.

(1) Neutral subcluster: Search over all the paths p_i. If a neutral path (subcluster) is identified go to step 2. If all the paths are searched and none of them are neutral, increment the syndrome index by 1 and repeat step 1.

(2) Subcluster annihilation: Annihilate the neutral subcluster by fusing the syndromes within the subcluster, i.e., along the path.

We refer to the HDRG decoder when augmented with initialization at a certain depth as the enhanced-HDRG. It is clear that the initialization step is not efficient because the number of paths to search over increases factorially as the depth increases, but for small numbers of levels the number of subclusters to search over is still not too high. For example, at the first level of initialization, in the worst case there will be six paths to check for each element in the bulk of $\mathbf{S}'$ [corresponding to the six neighboring syndromes (see Fig. 8)]. In general, the initialization step has an overhead of $C_i L^3$ where C_i is the number of paths for each syndrome for the ith initialization level. Specifically, $C_i = 6, 24, 6$, and 48, for the first four initialization levels, respectively.

We simulated the enhanced-HDRG decoder in the same way described in Sec. IV C. The results are summarized in Fig. 6. We see that the asymptotic threshold achieved for four levels of the initialization step is around 8.2%.

Although the improved thresholds for high d suggest that we are successfully able to disrupt the syndrome percolation using this technique, we still observe some saturation of the thresholds. To test this, we performed syndrome percolation simulations using the initialization step prior to the test for percolation. The results are summarized in Fig. 7. We see that the percolation threshold still upper-bounds the enhanced-HDRG thresholds for the corresponding initialization step.

Despite its success for very high qudit dimensions the enhanced-HDRG is not useful for low qudit dimensions, where the initialization step disrupts the syndromes in a way that results in a *lower* threshold. This can be understood as a result of using a decoding strategy that is too local—the neutral subclusters identified are in fact fragments of larger errors and the syndromes do not contain enough information to reconstruct them correctly. This suggests that the syndromes for very high qudit dimensions contain enough information to allow many rounds of initialization to keep improving the threshold. For smaller qudit dimensions, however, we see there is an optimal number of initialization rounds that should be performed; for example, for $d = 17$ we found that the two initialization levels are optimal.

V. DISCUSSION

We have presented a modified version of the HDRG decoder that was first introduced by Bravyi and Haah in [28] and studied its decoding performance for the surface code with noisy syndrome measurements. The main difference in our version is the use of a more refined metric which has led to an improved threshold. We have chosen the Manhattan distance metric δ, whereas Bravyi and Haah considered the d_∞ metric. In our investigations we discovered that the majority of the syndromes are cleared at the first level of decoding. This means that having a more refined metric matters more at $\ell = 0$ than it does at higher decoding levels. The δ metric ensures that the clusters at the first decoding level are as connected as possible by allowing a single syndrome to be connected only to its six nearest-neighbor plaquettes. This refinement of the metric is the reason for our improved thresholds.

We found that, similarly to the measurement-noise-free setting, for all but the smallest dimension d, syndrome percolation places an upper bound on the decoder threshold for the HDRG decoder. We have demonstrated that this can be overcome by adopting an extra initialization step, which, by scanning for locally neutral subclusters, breaks up the percolated lattice, allowing the decoder to succeed above the percolation threshold. This has a particularly stark effect for high dimensions, increasing the threshold by almost a factor of 2.

The uncorrelated noise model chosen here was adopted for ease of comparison with other decoders. However, an uncorrelated noise model is unlikely to be encountered in experiment. When the dimension is high, in an isotropic depolarizing noise model, there will be a high correlation between the presence of X-type and Z-type errors. A decoder which used this information might achieve significantly higher thresholds. Nevertheless, we expect the decoder presented here to possess an error threshold for any noise model acting independently and identically distributed (i.i.d.) with respect to individual qudits and also non-i.i.d. noise models where the correlation between qudit errors is limited. Testing these possibilities is a pertinent open question.

A remarkable feature of this decoder is the independence of its run time complexity with respect to qudit dimension. This is in stark contrast to other known qudit decoders. For example, the soft-decision renormalization-group decoder in the fault-tolerant setting [22] has a straightforward implementation in higher dimensions but comes with a cost of a polynomial overhead in d which means its applicability is limited to low dimensions.

To make our comparisons with other decoding algorithms more concrete in the qubit case, further research should focus on a full gate-error simulation of the HDRG in the low-noise regime. A comparison of success probability versus error rate in this regime would allow one to compare overheads, the most relevant figure of merit for judging the relative performance of these decoders.

Given the excellent performance of the MWPMA, the most important applications of the methods here will be for codes where MWPMA is not a suitable decoder. The surface codes studied in this paper are not the only quantum error correcting codes for which HDRG type decoders could be beneficial. An efficient decoding algorithm for the more exotic low density parity check (LDPC) code, the four-dimensional hyperbolic code, was introduced by Hastings with similar "greedy local matching" principles as the HDRG decoder [35]. Other LDPC codes exist for which efficient decoders have not yet been identified [36,37]. The development of computationally light fault-tolerant decoders for these codes is essential if they are to be practical. HDRG decoders have demonstrated the efficiency needed to support large scale fault-tolerant error correction on the surface code and a flexibility which may make them well suited to unlock the potential of future novel topological codes.

Note added. Recently, the authors learned of a similar investigation of the HDRG decoder with nonperfect syndrome measurements (without the initialization step) [23].

ACKNOWLEDGMENTS

We would like to thank Earl Campbell and James Wootton for useful discussions and helpful comments on this manuscript. We acknowledge the Imperial College High-Performance Computing Service and Legion at University College London for computational resources and associated support services. F.H.E.W. and H.A. acknowledge the financial support of the Engineering and Physical Sciences Research Council (Grants No. EP/G037043/1 and No. EP/K022512/1, respectively).

APPENDIX: EXPLICIT EXAMPLES

In this section we present two explicit examples outlining all the steps of the 3D fault-tolerant simulation for a single sample of errors. The reader may find the figures below more transparent in explaining how the HDRG decoder works in comparison to the description provided in Sec. IV. In both examples we choose a lattice of distance 5 and qudit dimension $d = 5$.

1. Example 1: HDRG decoding

In this example we present the simulation for the HDRG decoder without any initialization step. We describe the steps of the simulation in the captions of Figs. 9–14.

2. Example 2: Enhanced-HDRG decoding with a depth 1 initialization step

In Figs. 15–19 we present the simulation for the HDRG decoder when augmented with the first level of initialization, $\mathcal{I}_1$. The initialization step is shown in Fig. 16, and all the remaining steps are similar to those shown in the previous example.

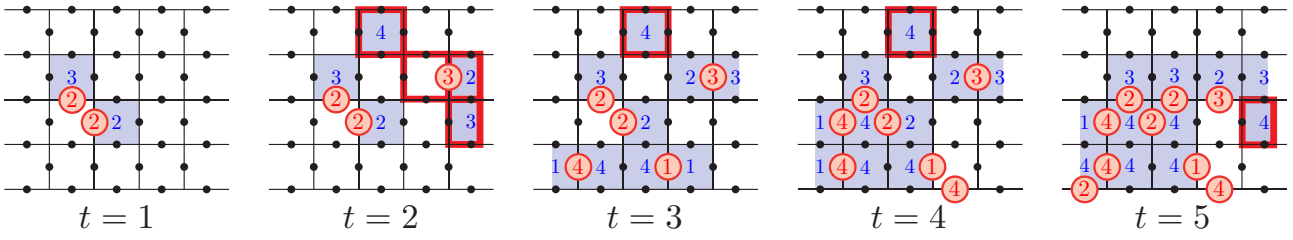

FIG. 9. (Color online) Error and syndrome histories. The first step is to generate the full history of errors and noisy syndrome measurements $\mathbf{S}$ for $L = 5$ time steps. The red circles and squares indicate the location of errors. Notice how the errors accumulate at each time step. The goal of the decoder is to correct the final error configuration $t = 5$.

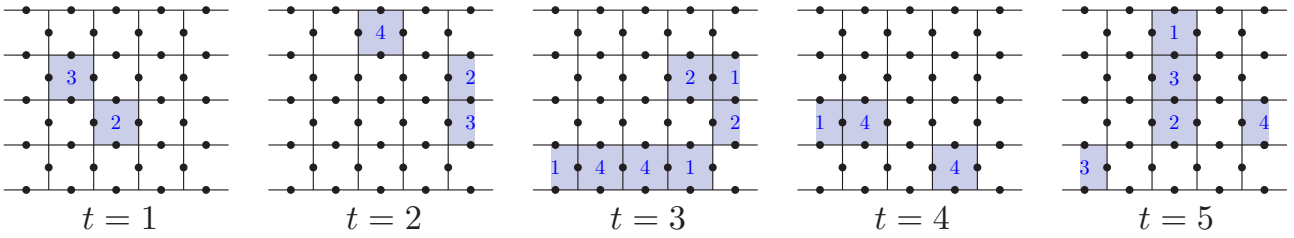

FIG. 10. (Color online) Syndrome changes history. The second step is to evaluate the syndrome changes history $\mathbf{S}' = \{\mathbf{e}_1, \mathbf{e}_2 \ominus \mathbf{e}_1, \mathbf{e}_3 \ominus \mathbf{e}_2, \mathbf{e}_4 \ominus \mathbf{e}_3, \mathbf{e}_5 \ominus \mathbf{e}_4\}$, where the subtraction is performed modulo d. The changes history is passed to the decoder, which must infer a correction operator from the information in $\mathbf{S}'$ alone.

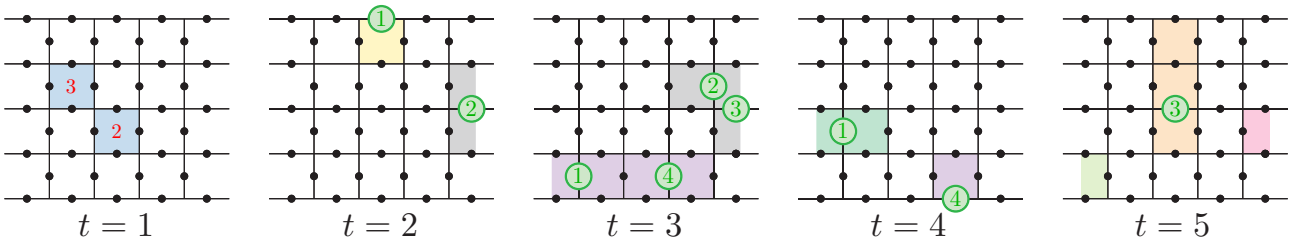

FIG. 11. (Color online) HDRG $\ell = 1$. The first level of the HDRG decoder divides the set $\mathbf{S}'$ into disjoint 1-connected clusters (i.e., $\delta = 1$). There are three different types of clusters shown: *non-neutral*, *neutral*, and *boundary-neutral*. Specifically, there are two single element non-neutral clusters shown in blue with the their charge displayed. These clusters cannot be fused at this level. Moreover, there are two neutral clusters in the bulk (gray and dark green), meaning that their total charge adds to zero (modulo 5). The elements of each neutral cluster are fused together to the vacuum. Finally, there are five boundary-neutral clusters (yellow, purple, light green, orange, and pink). The total charges of these clusters do not add up to zero, but since they are 1-connected to one of the boundaries they can be fused with that boundary. When the cluster is fused with the time boundary, no physical correction is applied. The resultant correction operator from the fusion of the neutral and boundary-neutral clusters is shown in green.

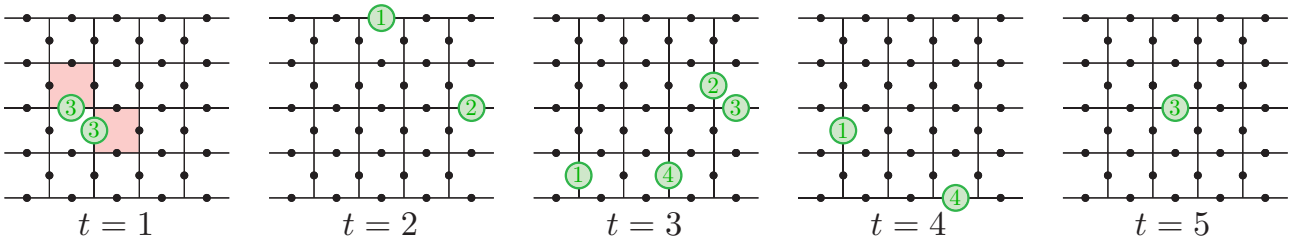

FIG. 12. (Color online) HDRG $\ell = 2$. The only remaining non-neutral cluster from the previous level is now 2-connected (shown in red). Its elements are fused together and a local correction is returned.

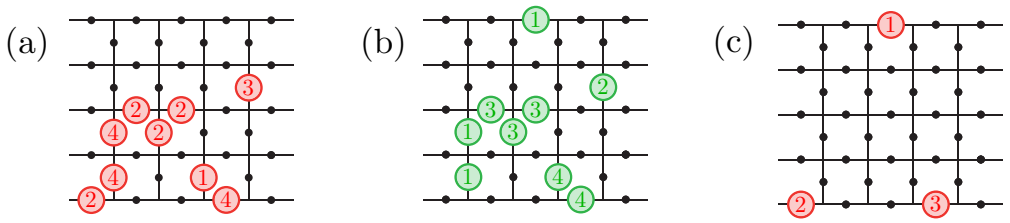

FIG. 13. (Color online) Projected correction. (a) The final (physical) error layer at $t = 5$. (b) Projected correction operator from corrections identified in Fig. 12, $\tilde{\mathbf{f}} = \mathbf{f}_1 \otimes \mathbf{f}_2 \otimes \mathbf{f}_3 \otimes \mathbf{f}_4 \otimes \mathbf{f}_5$, which is equivalent to summing the exponents of the operators modulo d. (c) The product of the accumulated error and the projected correction operators. The correction has resulted in a small number of remaining errors.

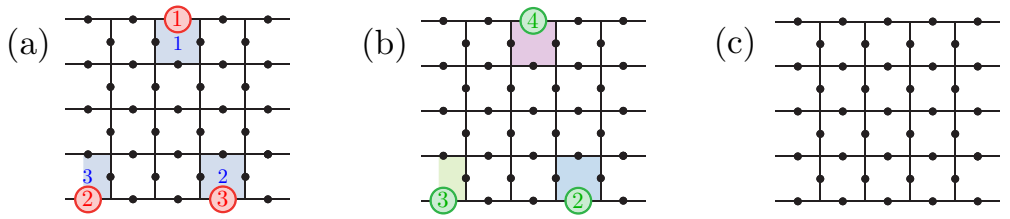

FIG. 14. (Color online) Noise-free decoding. To confirm whether the decoder has succeeded or failed, we must perform an additional round of decoding with noise-free syndrome measurements. (a) The outcomes of the noise-free syndrome measurements. (b) Clustering and correction operators. (c) Result of noise-free decoding. As we can see in this instance all the errors have been eliminated, no logical error has been introduced, and the decoding has succeeded.

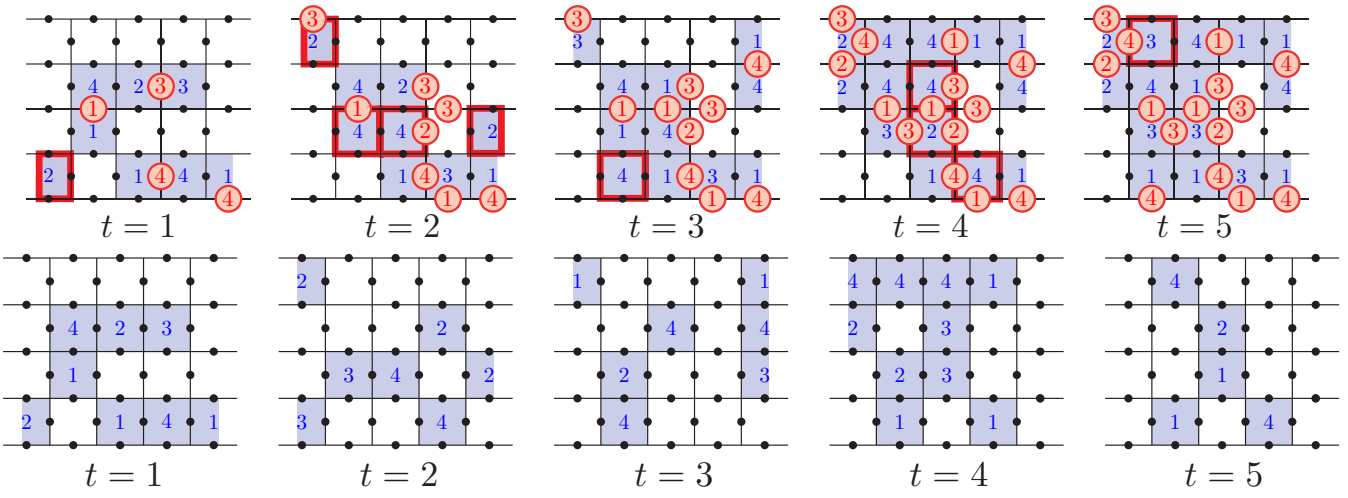

FIG. 15. (Color online) First row: The generation of the error and syndrome histories, similar to Fig. 9. Second row: The syndrome changes history $\mathbf{S}'$, similar to Fig. 10. Notice how $\mathbf{S}'$ contains a percolating cluster of syndromes.

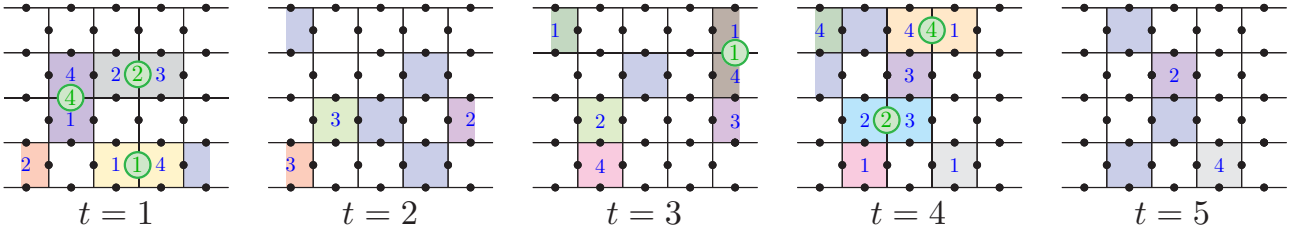

FIG. 16. (Color online) Initialization $\mathcal{I}_1$. At the first level of initialization there are only six subclusters to search around each plaquette. The search works by searching over every nontrivial syndrome and checking its six neighboring plaquettes sequentially to see if any of them forms a two-element neutral subcluster. Once a neutral pairing is found, the two plaquettes are fused together and a single correction operator is returned. Note that in this step we do not pair plaquettes to the physical or the time boundary. In the figure each subcluster is colored differently.

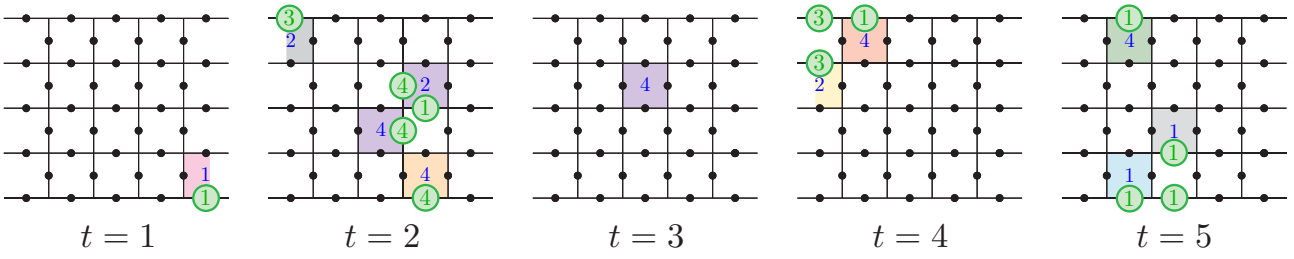

FIG. 17. (Color online) The initialization step has disrupted the percolating cluster. HDRG decoding. Neutral and boundary-neutral clusters identified by running two levels of the HDRG decoder with the correction operator returned are shown in green.

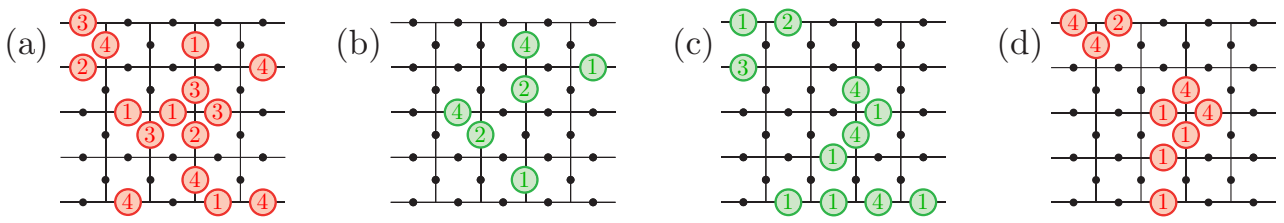

FIG. 18. (Color online) Projected correction. (a) The accumulated layer of errors at $t = 5$ (see the top row of Fig. 15). (b) The projected correction operator from running the initialization $\mathcal{I}_1$ (see Fig. 16). (c) The projected correction operator obtained from the HDRG decoder (see Fig. 17). (d) The resultant errors after taking the operator product of the two correction layers and the accumulated layer of errors.

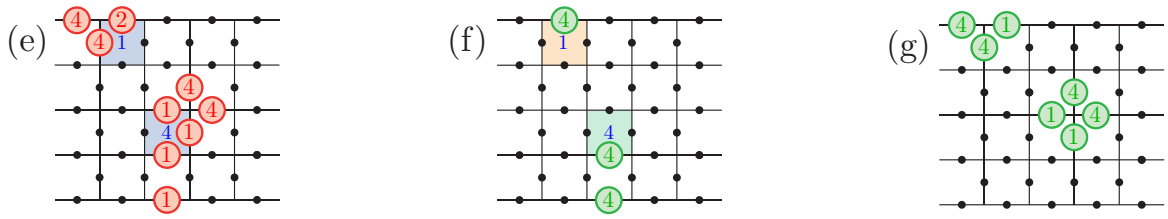

FIG. 19. (Color online) Noise-free decoding. (e) Noise-free syndrome measurements. (f) Clustering and correction operators. (g) The result of noise-free decoding. In this case the resultant operators are all members of the stabilizer group so once again the decoding has been successful.

[1] R. Raussendorf, J. Harrington, and K. Goyal, Topological fault-tolerance in cluster state quantum computation, New J. Phys. **9**, 199 (2007).

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

[3] S. J. Devitt, A. G. Fowler, A. M. Stephens, A. D. Greentree, L. C. L. Hollenberg, W. J. Munro, and K. Nemoto, Architectural design for a topological cluster state quantum computer, New J. Phys. **11**, 083032 (2009).

[4] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[5] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, arXiv:quant-ph/9811052.

[6] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. **303**, 2 (2003).

[7] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

[8] 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 (2011).

[9] A. M. Stephens, Fault-tolerant thresholds for quantum error correction with the surface code, Phys. Rev. A **89**, 022321 (2014).

[10] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, Phys. Rev. A **71**, 022316 (2005).

[11] H. Anwar, E. T. Campbell, and D. E. Browne, Qutrit magic state distillation, New J. Phys. **14**, 063006 (2012).

[12] E. T. Campbell, H. Anwar, and D. E. Browne, Magic-State Distillation in All Prime Dimensions Using Quantum Reed-Muller Codes, Phys. Rev. X **2**, 041021 (2012).

[13] E. T. Campbell, Enhanced Fault-Tolerant Quantum Computing in d-Level Systems, Phys. Rev. Lett. **113**, 230501 (2014).

[14] G. Duclos-Cianci and D. Poulin, Kitaev's $\mathbb{Z}_d$-code threshold estimates, Phys. Rev. A **87**, 062338 (2013).

[15] H. Anwar, B. J. Brown, E. T. Campbell, and D. E. Browne, Fast decoders for qudit topological codes, New J. Phys. **16**, 063038 (2014).

[16] R. S. Andrist, J. R. Wootton, and H. G. Katzgraber, Error thresholds for Abelian quantum double models: Increasing the bit-flip stability of topological quantum memory, Phys. Rev. A **91**, 042331 (2015).

[17] A Smith, B. E. Anderson, H. Sosa-Martinez, C. A. Riofrío, I. H. Deutsch, and P. S. Jessen, Quantum Control in the Cs $6S_{1/2}$ Ground Manifold Using Radio-Frequency and Microwave Magnetic Fields, Phys. Rev. Lett. **111**, 170502 (2013).

[18] B. E. Anderson, H. Sosa-Martinez, C. A. Riofrío, I. H. Deutsch, and P. S. Jessen, Accurate and Robust Unitary Transformations of a High-Dimensional Quantum System, Phys. Rev. Lett. **114**, 240401 (2015).

[19] C. Wang, J. Harrington, and J. Preskill, Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory, Ann. Phys. **303**, 31 (2003).

[20] 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).

[21] A. G. Fowler, Minimum weight perfect matching of fault-tolerant topological quantum error correction in average O(1) parallel time, Quant. Info. and Comput. **15**, 0145 (2015).

[22] G. Duclos-Cianci and D. Poulin, Fault-tolerant renormalization group decoder for Abelian topological codes, Quant. Info. and Comput. **14**, 721 (2014).

[23] A. Hutter, D. Loss, and J. R. Wootton, Improved HDRG decoders for qudit and non-Abelian quantum error correction, New J. Phys. **17**, 035017 (2015).

[24] A. Hutter, J. R. Wootton, and D. Loss, Efficient Markov chain Monte Carlo algorithm for the surface code, Phys. Rev. A **89**, 022326 (2014).

[25] M. Herold, E. T. Campbell, J. Eisert, and M. J. Kastoryano, Cellular-automaton decoders for topological quantum memories, arXiv:1406.2338.

[26] S. Bravyi, M. Suchara, and A. Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, Phys. Rev. A **90**, 032326 (2014).

[27] J. R. Wootton, A simple decoder for topological codes, Entropy **17**, 1946 (2015).

[28] S. Bravyi and J. Haah, Quantum Self-Correction in the 3D Cubic Code Model, Phys. Rev. Lett. **111**, 200501 (2013).

[29] T. Ohno, G. Arakawa, I. Ichinose, and T. Matsui, Phase structure of the random-plaquette Z_2 gauge model: Accuracy threshold for a toric quantum memory, Nucl. Phys. B **697**, 462 (2004).

[30] J. W. Harrington, Analysis of quantum error-correcting codes: Symplectic lattice codes and toric codes, Ph.D. thesis, California Institute of Technology, 2004.

[31] A. Kitaev, Topological quantum codes and anyons, Proc. Sym. Appl. Math. **58**, 267 (2002).

[32] S. S. Bullock and G. K. Brennen, Qudit surface codes and gauge theory with finite cyclic groups, J. Phys. A **40**, 3481 (2007).

[33] A. G. Fowler, D. S. Wang, and L. C. L. Hollenberg, Surface code quantum error correction incorporating accurate error propagation, Quant. Info. and Comput. **11**, 8 (2011).

[34] F. H. E. Watson and S. D. Barrett, Logical error rate scaling of the toric code, New J. Phys. **16**, 093045 (2014).

[35] M. B. Hastings, Decoding in Hyperbolic Spaces: LDPC Codes With Linear Rate and Efficient Error Correction, arXiv:1312.2546.

[36] J.-P. Tillich and G. Zemor, Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength, IEEE Trans. Inf. Theor. **60**, 1193 (2014).

[37] A. A. Kovalev and L. P. Pryadko, Improved quantum hypergraph-product LDPC codes, in *Proceedings of the IEEE International Symposium on Information Theory, Cambridge, MA* (IEEE, New York, 2012), pp. 348–352.