TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Serra Peralta, M., Shaw, M. H., & Terhal, B. M. (2026). Decoding across Transversal Clifford Gates in the Surface Code. *PRX Quantum*, *7*(1), Article 010335. https://doi.org/10.1103/sk5y-25b1

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Decoding across Transversal Clifford Gates in the Surface Code

Marc Serra-Peralta,[1,2,*,†] Mackenzie H. Shaw,[1,2,*] and Barbara M. Terhal[1,2]

[1] QuTech, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands

[2] Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands

(Received 6 June 2025; revised 14 October 2025; accepted 4 December 2025; published 19 February 2026)

Transversal logical gates offer the opportunity for fast and low-noise logic, particularly when interspersed by a single round of parity check measurements of the underlying code. Using such circuits for the surface code requires decoding across logical gates, complicating the decoding task. We show how one can decode across an arbitrary sequence of transversal gates for the unrotated surface code, using a fast "logical observable" minimum-weight perfect matching-based decoder, and benchmark its performance in Clifford circuits under circuit-level noise. We propose *windowed* logical observable matching decoders to address the problem of fully efficient decoding: our basic windowed decoder is computationally efficient under the restriction of quiescent (slow) resets. Our "advanced" two-step windowed decoder can be computationally inefficient but allows fast resets. For both windowed decoders we identify errors which scale sublinearly in d—depending on the structure of the circuit—which can lead to logical failure, and we propose methods to adapt the decoding to remove such failures. Our work highlights the complexity and interest in efficient decoding of fast logic for the surface code.

DOI: 10.1103/sk5y-25b1

I. INTRODUCTION

One of the many challenges in performing fault-tolerant quantum computing is that of implementing a fault-tolerant gate set in a quantum error-correction (QEC) code. Transversal gates are an attractive way of implementing gates because they are both fast and inherently low noise. Although it is impossible to construct a *universal* transversal gate set in a single QEC code [1,2], the Clifford group can be implemented transversally in both the 2D color and surface codes [3–6], leaving the T gate to be done by some fault-tolerant state preparation and injection. In the surface code some of these gates have been referred to as "fold"-transversal because they involve two-qubit gates between qubits in the same code block that are adjacent to each other when the surface code is "folded" along its diagonal, see Fig. 1. Because of this beyond-strictly 2D connectivity, transversal gates are particularly attractive in platforms with mobile qubits such as neutral atom qubits [7] or trapped-ion qubit chips (see, e.g., Ref. [8]). The fault-tolerance of transversal gates has been known for a long time, but it has only recently been proven that transversal gates and magic state injections can be performed *quickly* (that is, with only a constant number of QEC rounds between each gate) without compromising the fault-tolerance of the algorithm [9,10]. This stands in contrast with the purely 2D procedure of lattice surgery in the surface code, which can only perform a parity measurement on a subset of logical qubits once every $\Theta(d)$ QEC rounds where d is the distance of the code.

Transversal Clifford gates therefore offer the opportunity for faster fault-tolerant logic—referred by some as "constant-time" logical gates [11]—but comes at the price of more complex decoding. In particular, in order for measurement errors to be correctly diagnosed, any surface code decoder must look at a syndrome record which has a temporal width that grows at least linearly with d [12–14]. In other words, we need to decode *across* transversal logical gates. Because transversal logical gates can map stabilizer generators to *products* of generators, error detectors [15] may need to be set by the joint parity of more than two measurement outcomes. As a consequence, single measurement errors can flip more than two detectors, corresponding to so-called hyperedges in a decoding hypergraph [16]. Even in the surface code, the fast and simple minimum-weight perfect matching (MWPM) algorithm therefore cannot be directly applied to the full decoding problem.

*These authors contributed equally to this work.

†Contact author: m.serraperalta@tudelft.nl

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

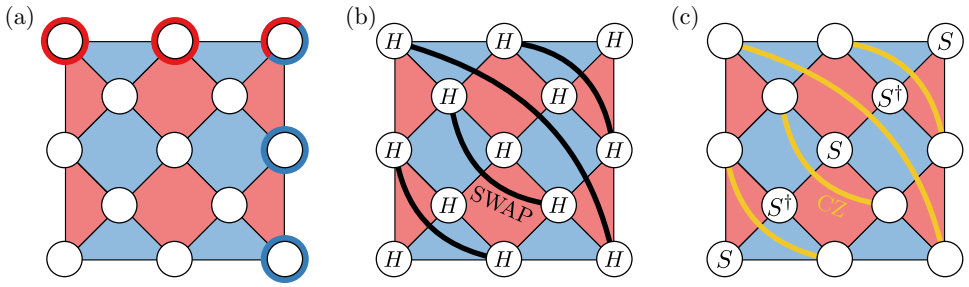


FIG. 1. (a) The $d = 3$ unrotated surface with data qubits as white circles and X- (respectively, Z-) stabilizers as shaded red (respectively, blue) regions. Representatives of the Pauli $\overline{X}$ (respectively, $\overline{Z}$) are shown by the red (respectively, blue) circles around the data qubits on which it has support. (b) and (c) The fold-transversal $\overline{H}$ and $\overline{S}$ gates, respectively. The logical Hadamard gate is implemented as a layer of Hadamard gates followed by a layer of SWAP gates. The long-range SWAPs (respectively, CZs) are shown as thick black (respectively, yellow) lines.

To overcome this issue, Ref. [10] used a minimum-weight *hypergraph* decoder to prove its results. Such a decoder is inefficient both because minimum-weight decoding of a hypergraph is inefficient in general and because the cost of the decoder scales with the depth of the algorithm being run.

In this paper we make progress toward designing an efficient MWPM-based decoder across fast transversal gates in the surface code. After reviewing previous work in Sec. I A, we describe our main setup and concepts in Sec. II. We avoid minimum-weight hypergraph decoding with a logical observing matching (LOM) decoder (Sec. III), which applies MWPM only to matchable subgraphs of the decoding hypergraph. Under a "basic" error model (defined in Sec. II C), the LOM decoder can correct any error of weight $< d/2$ in the unrotated surface code in arbitrary circuits with fast transversal Clifford gates and T gate injections which use fault-tolerantly prepared $|T\rangle$ magic states. This stands in contrast to existing hypergraph decomposition methods in the literature, including both "splitting" decoders and "hierarchical" matching decoders [17–21], whose failures we discuss in Secs. IV C and III C, respectively. One key detail in the LOM decoder is how to handle so-called fragile logical measurement observables that we discuss pedagogically in Sec. III B 2. Our fault-tolerant results are not specific to MWPM and hold for any (distance-preserving) graph-based decoder, such as Union Find [22]. Numerical benchmarks of the LOM decoder show high thresholds and accuracy compared to minimum-weight decoding under both phenomenological and circuit-level depolarizing noise and for both repeated and random logical two-qubit Clifford gate circuits (Sec. IV).

However, the LOM decoder is still inefficient because it has a decoding volume that scales with the depth of the circuit. To this end, we propose a *windowed* version of the LOM decoder in Sec. V. It turns out to be nontrivial to design a windowed matching decoder for circuits with fast gates, resets and measurements, that is both computationally efficient and able to correct all basic errors of weight $< d/2$. Instead, we introduce two variants of the *windowed*-LOM decoder: a "basic" variant that is computationally efficient but cannot decode circuits with fast resets, and a "two-step" variant that can decode circuits with fast resets but is no longer guaranteed to be efficient. Both of these variants require further modifications to be able to correct all low-weight error patterns, one of which requires synchronized resets and measurements (except those in T injection circuits) to occur only every $\Omega(d)$ steps, but we conjecture that with these modifications both variants of the windowed-LOM decoder can correct up to $d/2$ basic errors and have a noise threshold. We end the paper with a discussion in Sec. VI on possible future work, including the use of independent observer decoders more generally in decoding.

A. Previous and simultaneous work

In Ref. [10], it was first proved and numerically demonstrated that one can implement fault-tolerant quantum logic by having $O(1)$ QEC rounds after each transversal gate for, say, the unrotated surface code, supplemented by $|T\rangle$ magic states. The constructions in that paper and the earlier paper on correlated decoding, Ref. [9], invoked minimum-weight decoders or used decoders that can explicitly handle hyperedges, thus not necessarily being very efficient. Our goal here is to show how to construct an efficient fault-tolerant *matching*-based decoder.

Reference [23] proposed an algorithm to decompose hyperedges into edges (see also [24]), but we show that it does not perform well when decoding across transversal gates in Sec. IV C. References [18,25] decompose the hyperedges and iteratively run MWPM to achieve good logical performance. However, no formal guarantees are proven for the number of iterations and the logical performance for an arbitrary circuit and increasing code distance. Furthermore, the hierarchical MWPM decoding methods from Refs. [17,19–21] cannot be used to fault-tolerantly decode an *arbitrary* sequence of transversal logical Clifford gates, as we argue in Sec. III C.

Reference [11] addresses similar questions of decoding as our work, with numerics decoding transversal logical gates such as the CNOT and H for the surface code, interspersed by single QEC rounds. Reference [11] provides a windowed decoder, but it does not reduce decoding to a matching problem, instead it uses heuristic approximate solvers to deal with hyperedges. A limitation for this type of approximate solver is their poor embedding into specific hardware for running real-time decoding. On the other hand, Refs. [26,27] have already performed real-time decoding on experimental data with graph-based decoders, such as MWPM. Windowed, also called sliding-window, matching decoders for memory experiments in the surface code have previously been considered and used in Refs. [12,28–32].

At the time of writing this paper and presenting our work at the APS March meeting 2025, we became aware of similar work on fast matching-based decoding of transversal gates for the surface code [33].

II. PRELIMINARIES

We begin this work with a pedagogical introduction to the concepts we will use throughout the paper.

A. Circuits, (fragile) observables, and observing regions

We consider an arbitrary (unencoded) quantum circuit $\mathcal{C}$ compiled into $|0\rangle$, $|+\rangle$, and $|T\rangle$ state preparation, destructive Z- and X-basis measurements, and Clifford gates that may be conditioned on the outcome of those measurements. If a measurement is used to condition a Clifford gate, we call it a *conditioning* measurement; otherwise, we call it a *final* measurement. Each different set of outcomes of conditioning measurements leads to a different *realization* of the circuit. The outcome of the computation is then determined from some subset of the final measurements.

We consider each layer of resets, gates, and/or measurements to take place at a time labeled by a half-integer, so $t + 1/2$ with $t \in \mathbb{Z}$. With this, we say that a *circuit location* is a point in the circuit in between layers of gates described by a set of coordinates (t, j), where $t \in \mathbb{Z}$ represents time and j is a qubit index.

We make use of *Pauli regions* of the circuit, which is formally a list of Pauli operators $P = [P_{(t,j)}]$, with each element of the list labeled by a circuit location. Each element is itself a single-qubit Pauli operator I, X, Y, or Z (ignoring phases). Two Pauli regions P_1 and P_2 can be multiplied to each other element-wise by multiplying the Pauli operators at each circuit location. We say that two Pauli regions anticommute if their components anticommute at an odd number of circuit locations; otherwise we say they commute. We also consider time slices of Pauli regions denoted as P_t, which is a Pauli operator acting on all of the qubits that are active at time t. Note that a Pauli region would be a Pauli operator if we associated a qubit with each circuit location as is done for space-time codes [34].

We define an *observable* $O = \{M_1, M_2, \dots\}$ as any nonempty subset of measurements in the circuit $\mathcal{C}$ where the outcome of the observable is the *parity* of the outcomes of each measurement M_i in the observable (the measurements M_i can be conditioning and/or final measurements and either X or Z-measurements). In a slight abuse of notation we will also use O to refer to the *Pauli representation* of the observable, which is the Pauli region that has a Pauli Z at the circuit locations immediately preceding each of the Z-measurements in the observable (and likewise a Pauli X preceding an X-measurement), and identity operators elsewhere. This way two observables O_1 and O_2 can be multiplied by each other, constructing $O_1 O_2$ by multiplying their Pauli representations. Later in Sec. V, we will additionally refer to observables that do not correspond to measurements in the circuit and instead are just a Pauli operator in a circuit that hypothetically *could* be measured.

Given a realization of the circuit, we define the *observing region* $O^{\leftarrow}$, which is itself a Pauli region, of the observable O as follows. This object has been called the spackle of a Pauli string in Ref. [34], the back-cumulant of O in Ref. [35], and has also been used in Refs. [15,36]. We consider all the measurement locations M_i of the observable and apply the following rules:

(1) Begin by setting $O^{\leftarrow} = O$, the Pauli representation of the observable; that is, for all circuit locations (t, j), set $O_{(t,j)} = X$ (respectively, Z) if there is a X-measurement (Z-measurement) at $(t + 1/2, j)$ which is in the observable O, otherwise set $O_{(t,j)} = I$.
(2) Starting at the last time step on which the observable has support and proceeding iteratively backward in time, apply the following rule. If the circuit locations $\{(t, j) : j \in J\}$ occur before a layer of Clifford gates C at time $t + 1/2$ for some subset of qubits J, set $O^{\leftarrow}_{(t,J)} = C^{\dagger} O^{\leftarrow}_{(t+1,J)} C$, where $O^{\leftarrow}_{(t+1,J)}$ is the $|J|$-qubit Pauli operator acting on each qubit according to $O^{\leftarrow}_{(t+1,j)}$ for $j \in J$.

Note that when the backpropagation hits a reset or a T-ancilla preparation, nothing backpropagates further. Note also that the observing region can only be calculated after all the conditional gates that occur before the last measurement included in O have been fixed. Loosely speaking, the observing region $O^{\leftarrow}$ is obtained as the "backpropagation" of the Pauli representation of the observable O. Importantly, the observing region also corresponds to the region of the circuit where an error in the circuit flips the outcome of the observable: if $O^{\leftarrow}_{(t,j)} = Z$, then a Pauli X or Y error at the circuit location (t, j) flips the value of the observable.

We also consider *reset stabilizers* of the circuit. Analogous to our definition of an observable O as a set of measurements, we define a reset stabilizer S as a set $\{R_1, \dots\}$ of $|+\rangle$- or $|0\rangle$-state resets in the circuit $\mathcal{C}$. The Pauli representation of a reset stabilizer S is the Pauli region which has X (respectively, Z) operators at the circuit locations immediately following each of the $|+\rangle$-resets (respectively, $|0\rangle$-resets) in S, while the reset stabilizing region $S^{\rightarrow}$ is the "forward-propagation" (cumulant in Ref. [35]) of S using a time-reversed version of Steps (1)–(2) used to define the observing region. An important fact that we prove for completeness (see, e.g., Ref. [36] and Proposition 3 in Ref. [35]) in Sec. I of the Supplemental Material is the following:

Lemma 1. $S^{\rightarrow}$ and O anticommute if and only if S and $O^{\leftarrow}$ anticommute.

Lastly, we define fragile, or unreliable, observables. The relevance of fragile observables relates to the encoded execution of the circuit $\mathcal{C}$ and the existence of fragile time boundaries, see Sec. II C 1, but its definition depends only on $\mathcal{C}$ itself, as follows:

Definition 1 (Fragile observable). We say that an observable O is *fragile* if its observing region $O^{\leftarrow}$ anticommutes with at least one reset stabilizer S of the circuit; or, equivalently via Lemma 1, if the observable O anticommutes with at least one reset stabilizing region $S^{\rightarrow}$ of the circuit.

In the absence of conditional Clifford gates, a fragile observable always has a 50-50 measurement outcome, since applying the reset stabilizer does not change the state while it changes the measurement outcome of the observable. In the presence of conditional-Clifford gates, an observable can be fragile or not depending on which Clifford gates are applied and can even have a *deterministic* outcome [an example is circuit Eq. (2) where the observable O_4 is fragile if both conditional S gates are applied]. Fragile observables have been called "nondeterministic" observables in Ref. [10]. We will call nonfragile observables *reliable* observables. Note that nonfragile observables are not necessarily deterministic, due to the presence of $|T\rangle$-state injections.

B. Fault-tolerant circuit implementation in the surface code

To implement $\mathcal{C}$ fault-tolerantly, we encode each qubit into a distance-d unrotated surface code, using $n = 2d^2 - 2d + 1$ data qubits. Throughout the logical computation we perform one QEC round of parity check measurements after each circuit layer. We compile each circuit element in $\mathcal{C}$ using the following logical operations on the unrotated surface code:

(1) Transversal logical $|\overline{+}\rangle$ (respectively, $|\overline{0}\rangle$) state preparation; that is, the preparation of $|+\rangle^{\otimes n}$ ($|0\rangle^{\otimes n}$) on each data qubit.
(2) Fault-tolerant magic state preparation, which could be achieved by, for example, magic state distillation. Importantly, we assume that the values of all X- and Z-stabilizers are known after the preparation of the state with an error rate less than $\sim p^{d/2}$ for some small $p \ll 1$.
(3) Transversal logical $\overline{X}$ (respectively, $\overline{Z}$) measurement; that is, an X-basis (Z-basis) measurement of each data qubit in the n-qubit code block, which is processed, together with other measurement data, by the decoder, see Sec. II C.
(4) Fold-transversal $\overline{S}$ and $\overline{H}$ gates as depicted in Fig. 1 and the transversal $\overline{\text{CNOT}}$ gate, possibly conditioned on transversal logical measurements.

All Pauli operations—both those arising from decoding and those directly in $\mathcal{C}$—are tracked in software. We denote the implementation of $\mathcal{C}$ encoded in the surface code as $\overline{\mathcal{C}}$, and when required for clarity we refer to $\mathcal{C}$ as the *bare* circuit and $\overline{\mathcal{C}}$ as the *encoded* circuit.

C. Error model, detectors in the pre-gate frame, and decoding graph

For all results in Secs. III and V we assume a *basic* error model in the circuit $\overline{\mathcal{C}}$ which is as follows:

(1) independently apply Pauli X and Pauli Z, each with probability p, to each data qubit before each logical gate (followed by the single QEC round),
(2) apply an X error (respectively, Z error) with probability p before a physical qubit Z-measurement (respectively, X-measurement).

Thus in this error model, no errors are inserted prior to the QEC round, but each parity check measurement in the QEC round can be faulty due to the insertion of an error prior to the ancilla qubit measurement. We will show below that decoding efficiently across transversal gates is a nontrivial problem to solve even with this basic error model. Therefore, our strategy in this paper is to design a fast decoder that can efficiently decode errors in the basic model and prove its fault-tolerance (Theorem 1). However, these results directly bear on circuit-level noise since circuit-level errors have the same effect as products of a *constant* number of errors in the basic model, and we test this in the numerics in Sec. IV where we model more realistic phenomenological and circuit-level depolarizing noise models. One could also apply known heuristic decoding improvements to MWPM to optimize decoding for circuit-level noise; however, this is not the focus of this paper.

We define a set of *detectors*—sets of QEC measurement outcomes whose parity is deterministic in the absence of errors [15]. In the absence of logical operations, each detector simply consists of two measurements of the same ancilla qubit in consecutive QEC rounds, as is standard for decoding memory experiments on the surface code. If the measured outcome of a detector differs from its ideal outcome, we call this a *defect*, and the full set of defects is the error *syndrome*. Given the definition of a detector, one can define its *detecting region* [15], namely the Pauli region obtained by backpropagating the Pauli representation of the detector as observable, defined in Sec. II A.

In the presence of transversal logical gates, we wish to find a set of detectors such that the detecting regions are still local in space-time. Such locality is desirable because it implies that each detector can only be flipped by a constant number of errors in the circuit. Because parity checks are spread nontrivially by the logical gates, the detector definitions that achieve this depend on the logical gates that are applied. In this paper we choose a so-called *pre-gate* frame for the detectors. In this frame, each Z- (respectively, X-) detector only involves a single Z (respectively, X) parity measurement outcome before the logical gate and a possibly nontrivial linear combination of binary measurement outcomes after the logical gate, see its precise definition in Sec. II in the Supplemental Material [37]. A very similar frame choice is the *post-gate* frame where each detector only involves a single parity measurement outcome after the logical gate. These detector frames have been used in other previous works on decoding logical gates, see, e.g., Refs. [9,18].

Given an observable $O = \{M_1, M_2, \dots\}$ of the bare circuit $\mathcal{C}$, we can lift this to a *logical* observable $\overline{O}$ of the encoded circuit $\overline{\mathcal{C}}$ by replacing each bare measurement M_i in O with a set of physical measurements that are in the support of the logical operator in $\mathcal{C}$. Throughout the manuscript we choose a logical representative that consists of a string of X (respectively, Z) operators along a spatial boundary of the surface code, as shown in Fig. 1(a). The encoding gives rise to an observing region $\overline{O}^{\leftarrow}$ of the fault-tolerant circuit $\overline{\mathcal{C}}$ that corresponds to all the physical circuit locations where a Pauli flip would flip the recorded outcome of the observable $\overline{O}$. In a more general circuit-level noise model, $\overline{O}^{\leftarrow}$ can include locations on both data and ancilla qubits.

The product of the outcomes of each of the physical measurements that make up $\overline{O}$ gives the *raw* outcome of $\overline{O}$. The task of the decoder is to take the outcomes of all the detectors, predict the error that occurred, and then to infer which observables are flipped by this error. Combining the decoder's prediction with the raw outcome of $\overline{O}$ gives the *error-corrected* outcome of $\overline{O}$.

Using the basic error model and the definition of detectors, we construct a decoding hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{H})$ consisting of vertices $v \in \mathcal{V}$ for each detector in $\overline{\mathcal{C}}$. There is a hyperedge $h \in \mathcal{H}$ for each X or Z error in the basic error model, such that the endpoints of each hyperedge correspond to the detector(s) that are flipped due to the error [38]. For each observable $\overline{O}$ of the circuit, there is a subset of the hyperedges $\mathcal{H}_O$ that cause a flip to the observable, we call this subset of hyperedges collectively the *observing hyperedge set*, namely $\mathcal{H}_O$ is the set of hyperedges that correspond to an error that anticommutes with the logical observing region $\overline{O}^{\leftarrow}$. In our basic error model, for any observable $\overline{O}$, the observing hyperedge set $\mathcal{H}_O$ is a set of edges, i.e., it consists only of a set of space-like edges on one of the spatial boundaries of $\mathcal{G}$, see an example in Fig. 4. Hence we will refer to $\mathcal{H}_O$ as the observing edge set in the remainder of this paper. For circuit-level noise $\mathcal{H}_O$ can be a set of hyperedges. Note that the observing edge set is closely related to the observing region: the observing region is defined at the level of unencoded circuits, but it has a corresponding expression at the physical qubit and decoding graph level (depending on the error model) that exactly coincides with the observing edge set [39]. The structure of $\mathcal{H}_O$ will be important in the following section when we define our single logical observable matching decoder.

Let us discuss the structure of the decoding hypergraph $\mathcal{G}$ for our basic error model. It consists of both space-like hyperedges—those arising from data-qubit errors—that connect detectors at the same time step, and time-like hyperedges—those arising from measurement errors—that connect detectors in two adjacent time steps, see Fig. 2. At each circuit location (t, j), the space-like hyperedges form an identical subgraph consisting of a disconnected X- and Z-component corresponding to the X- and Z-detectors. Some space-like hyperedges have weight one, i.e., they are connected to only one vertex, and these we say are *connected to the boundary*. When decoding with a minimum-weight decoder, one would add a boundary vertex v_{bdy} to the graph and connect each weight-one edge to it, but we do not consider such a modification here.

1. Time-like hyperedges and fragile time boundaries

The time-like hyperedges may touch one, two, *or* three vertices, as shown in Fig. 2. In particular, weight-one time-like edges correspond to measurement errors in Z- (respectively, X-) detectors immediately following a $|+\rangle^{\otimes n}$ (respectively, $|0\rangle^{\otimes n}$) reset, or immediately preceding an X- (respectively, Z-) measurement, and they will also later appear when we consider windowed decoding in Sec. V. In general, we refer to these weight-one edges collectively as constituting an *open* time boundary (as a single defect can be matched to it), as opposed to a closed time boundary to which a single defect cannot be matched. More specifically, if the open time boundary arises from a reset or measurement (instead of from windowed decoding), we refer to it as a *fragile* time boundary. Open time boundaries

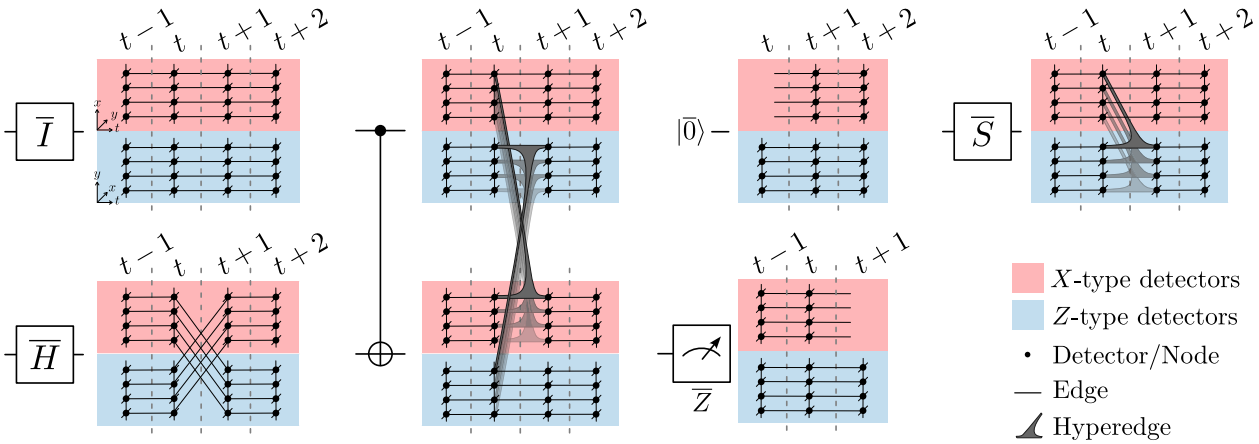


FIG. 2. Decoding hypergraphs $\mathcal{G}$ when executing a logical operation assuming a basic error model, using the pre-gate frame. Time flows from left to right and the vertical gray dashed lines denote half-integer layers when a logical gate followed by a QEC round happens, so that detectors are labeled by integer times $t \in \mathbb{Z}$. In the subfigures, the logical gate + QEC round happens at layer $t + 1/2$ while for all other layers we perform the identity gate. Logical reset and measurement happen at $t + 1/2$ and are only shown for $\overline{Z}$, as the decoding hypergraphs for the $\overline{X}$-basis are the same but with the X and Z detector types swapped. The decoding hypergraphs are three-dimensional for the surface code, but we have drawn a two-dimensional slice for easier visualization. Note that for the logical $|\overline{0}\rangle$ there are no weight-1 time-like edges—the time boundary is closed—while for the X-detectors there are weight-one time-like edges—the time boundary is open.

are potentially dangerous for decoding since only one type of stabilizer has a fixed eigenvalue. Moreover, the observing edge set of any fragile observable, see Definition 1, "borders" at least one fragile time boundary, in the sense that at least one hyperedge in the observing edge set shares a vertex with at least one edge in the fragile time boundary. How to avoid decoding fragile observables is discussed in Sec. III B 2.

A weight-three time-like edge corresponds to a measurement error on an ancillary qubit immediately following a logical $\overline{S}$ or $\overline{\text{CNOT}}$ gate. For example, a measurement error on an ancilla qubit measuring a Z-check immediately following an $\overline{S}$ gate triggers three detectors: two Z-detectors before and after the measurement *and* an X-detector before the measurement whose detecting region includes the ancilla measurement by the definition of the pre-gate frame, see Table I from Supplemental Material [37] and the gray hyperedge in Fig. 3(b).

Clearly, in the absence of the weight-3 time-like hyperedges, $\mathcal{G}$ would be matchable and thus decodable using MWPM. A simple approach to decoding $\mathcal{G}$ is therefore to follow the "splitting-hyperedge" methods of Ref. [23] and Stim [24]; that is, to perform decoding only using the subgraph $\mathcal{G}_{\text{matchable}}$ of $\mathcal{G}$ that contains weight-2 edges such that $\mathcal{G}_{\text{matchable}}$ is matchable. We could then decompose each weight-3 hyperedge into a set of edges in $\mathcal{G}_{\text{matchable}}$ whose

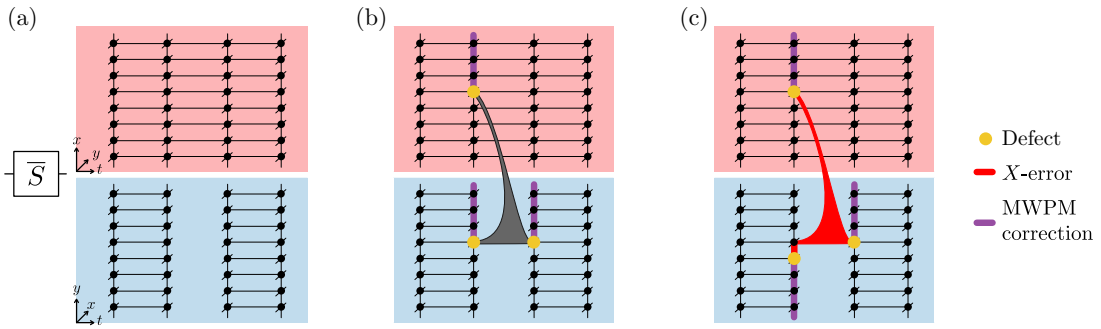


FIG. 3. Hyperedge decomposition and error combination that explains the bad performance of a "splitting-hyperedge" matching decoder in the pre-gate frame. (a) $\mathcal{G}_{\text{matchable}}$ of the $\overline{S}$ gate, which is built by only taking the edges from the decoding hypergraph in Fig. 2. (b) The procedure from Ref. [23] to decompose the gray hyperedge goes as follows: (1) the detectors involved in the hyperedge are triggered (yellow dots), (2) such syndrome is decoded with MWPM using $\mathcal{G}_{\text{matchable}}$, and (3) the correction (purple edges) corresponds to the hyperedge decomposition. Note that the number of purple edges in the decomposition can badly grow with the code distance. (c) The two errors marked in red lead to a logical error when decoded using $\mathcal{G}_{\text{matchable}}$, thus the logical scaling is at most $O(p^2)$ for any distance.

endpoints match the endpoints of the hyperedge and use this decomposition to update the weights of the edges in $\mathcal{G}_{\text{matchable}}$. For example, this method can be applied to Y data-qubit errors which produce four defects in standard circuit-level noise decoding for the surface code.

However, this approach is not fault-tolerant for our circuits: there are weight-2 error patterns that lead to a logical error, as shown in Fig. 3(c). The key issue is that the weight-3 time-like hyperedge cannot be decomposed into a set of weight-2 time-like edges and instead must use a set of space-like edges. If the hyperedge is located $\sim d/2$ away from each of the spatial boundaries, the number of space-like errors that must be used to decompose the hyperedge is $\sim 3d/2$, see Fig. 3(b). Consequently, with just the weight-2 error pattern shown in Fig. 3(c), the matching decoder on $\mathcal{G}_{\text{matchable}}$ will make a logical error. We demonstrate this numerically in Sec. IV C. We therefore require a different approach to decode $\mathcal{G}$ using MWPM, which we present in the following section.

III. LOGICAL OBSERVABLE MATCHING (LOM) DECODING

In this section we design a matching-based decoder for the hypergraph $\mathcal{G}$. The key observation enabling our decoder is that for any single observable O, it is possible to define a matchable subgraph that "follows" the *observing region* $O^{\leftarrow}$ of the observable. Therefore, by performing matching on this subgraph of the decoding graph, we are able to determine if a logical error occurred that flips the observable O. We call this decoder the *Single Logical Observable Matching* decoder, or single-LOM for short. In Sec. III B, we turn the single-LOM decoder into the LOM decoder: a windowless matching decoder for the full circuit $\mathcal{C}$ that works by running a *separate* iteration of the single-LOM decoder for each of the independent reliable observables in the circuit.

A. Single logical observable matching decoding

Given an observable O, we define the subgraph $G_O = (V_O, E_O)$ of $\mathcal{G}$ as follows, see the example circuit in Fig. 4. To define the vertex subset $V_O \subseteq \mathcal{V}$, we take the observing edge set $\mathcal{H}_O$ of O, which consists of a set of space-like edges on the spatial boundary of the surface code. Each edge $e \in \mathcal{H}_O$ is connected to the boundary, with its only endpoint a vertex that occurs at some logical circuit location (t, j) and corresponds to an X- or a Z-detector. To construct the vertex subset V_O, we include all vertices that occur at the same logical circuit location (t, j) and correspond to a detector of the same Pauli type as an edge in $\mathcal{H}_O$, as shown in Fig. 4. Given the vertex subset V_O we now define the edge subset $E_O = \{h \cap V_O : h \in \mathcal{H}, h \cap V_O \neq \emptyset\}$, which is intuitively the "projection" of the hyperedges

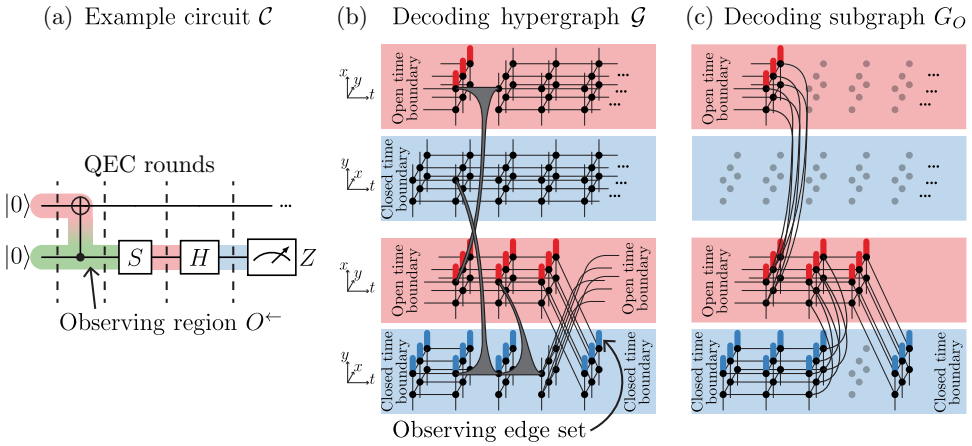


FIG. 4. The decoding hypergraph and subgraph G_O for an observable O for an example circuit. (Actually, the Z measurement observable which is decoded here happens to be fragile, and decoding fragile observables can be avoided for reasons discussed in Sec. III B 2.) (a) The observing region $O^{\leftarrow}$ of the observable O being the Z-measurement is shown, where red, green, and blue shading regions indicate X, Y, and Z support, respectively. (b) The decoding hypergraph $\mathcal{G}$ of the encoded circuit $\overline{\mathcal{C}}$ for the $d = 3$ unrotated surface code. The vertices of $\mathcal{G}$ are partitioned based on the logical qubit to which they pertain and whether they represent X- (red background) or Z- (blue background) detectors. Note that the x and y spatial coordinates have been swapped between the X- and Z-detectors for ease of visualization. Also note that we have only shown the weight-3 hyperedges that appear on the top-front qubit to reduce clutter; weight-3 hyperedges with the same pattern also exist at all the other spatial coordinates in $\mathcal{G}$ but are not shown here. We have also labeled all the edges that make up the observing edge set $\mathcal{H}_O$ (in blue and red), and the open and closed time boundaries that arise from resets and measurements in (a). (c) The decoding subgraph G_O for the observable O. As explained in the text, G_O contains all the vertices V_O of $\mathcal{G}$ that pertain to the same logical qubit and represent a detector of the same Pauli type (X or Z) as an edge in the observing edge set. It clearly has the property that it is a graph with no hyperedges. Vertices that are excluded from V_O are shaded in gray.

in $\mathcal{H}$ onto the vertex subset V_O, and $\mathcal{H}_0 \subseteq E_0$. It is a straightforward exercise to verify that E_O is indeed a set of edges for all possible choices of observables for the $\{\overline{H}, \overline{S}, \overline{\text{CNOT}}\}$ gates, by visual inspection using Fig. 4. A general mathematical argument can be found in Sec. III in the Supplemental Material [37].

The single-LOM decoder then proceeds by applying MWPM to the observed defects in V_O using the edges in E_O. If the returned minimum-weight solution contains an odd number of edges in the observing edge set $\mathcal{H}_O$, then the single-LOM decoder predicts that the observable has been flipped by the error and must be corrected; otherwise, it predicts that the observable has not been flipped by the error. Given the definition of $\mathcal{G}_O$, it is clear that defects in the entire observing edge set are contained in E_O and therefore the single-LOM decoder is looking at all relevant parts of the circuit that could cause a flip in O. Note that one could also apply other graph-based decoders, such as Union Find [22], to decode the observed defects in V_O.

Finally, we discuss the fault-tolerance of this single-LOM decoder applied to any reliable observable, which is summarized as follows.

*Theorem 1 (Fault-tolerance of the single-*LOM *decoder).* Given a reliable observable O, the single-LOM decoder applied to the decoding subgraph $G_O = (V_O, E_O)$ can correctly predict whether any basic error of weight $w < d/2$ has flipped the observable O.

Proof. The proof is straightforward and follows familiar arguments for the surface code, but for completeness we repeat them here.

The errors that occur on the hypergraph can be projected onto the decoding subgraph, let us write $\vec{e} \subseteq E_O$ as the set of edges in G_O that have been flipped. The defects that are visible to the decoder are then $\partial\vec{e} \subseteq V_O$. The single-LOM decoder then works by applying the minimum-weight correction $\vec{c} \subseteq E_O$ to the defects such that $\partial\vec{c} = \partial\vec{e}$. The total error-plus-correction string $\vec{e} \oplus \vec{c}$ does not trigger any detectors, and therefore is either a logical error or logically trivial, depending on whether $\vec{e} \oplus \vec{c}$ intersects the observing edge set $\mathcal{H}_O$ an odd or an even number of times (respectively).

We will now show that if the weight $w(\vec{e})$ of $\vec{e}$ is less than $d/2$, then $\vec{e} \oplus \vec{c}$ intersects $\mathcal{H}_O$ an even number of times. First, note that because $\vec{c}$ is a minimum-weight correction to $\partial\vec{e}$, then we have $w(\vec{e} \oplus \vec{c}) \leq w(\vec{e}) + w(\vec{c}) < d$. Because the observable is reliable, all the time boundaries in G_O are closed and the only weight-one edges in G_O are the spatial boundaries on the surface code, with the observing edge set $\mathcal{H}_O$ containing all the weight-one edges on one of these boundaries. If $\vec{e} \oplus \vec{c}$ were to intersect $\mathcal{H}_O$ an odd number of times, it therefore must traverse from the "top" to the "bottom" spatial boundary. However, the minimum-weight path from the "top" spatial boundary to the "bottom" has weight d; this is even the case in the presence of time-like edges arising from weight-3 hyperedges because these edges connect vertices with the same spatial coordinates. Therefore, any possible string $\vec{e} \oplus \vec{c}$ that intersects $\mathcal{H}_O$ an odd number of times must use at least d edges. This is a contradiction since $w(\vec{e} \oplus \vec{c}) \geq d$, and hence $\vec{e} \oplus \vec{c}$ is logically trivial, and the proof is complete. ■

Interestingly, there exist error patterns of weight at least d that are undetectable errors by the single-LOM decoder, but are detectable in the full decoding hypergraph and therefore may be correctable by a minimum-weight hypergraph decoder. For example, if the error string uses an edge that corresponds to a hyperedge in the full decoding hypergraph, the string may be detectable in the full hypergraph but not in the single-LOM instance.

B. Multiple logical observable matching decoding

In this section, we explain how to turn the single-LOM decoder into a decoder for an entire quantum circuit $\mathcal{C}$ which we call the LOM decoder. We start by discussing how to decode *reliable* circuits—that is, circuits in which all observables are reliable—before arguing that fragile observables do not need to be decoded in Sec. III B 2.

1. Decoding reliable circuits

In the case where all observables in the circuit are reliable, the LOM decoder is simple: decode every logical measurement individually—both conditioning and final measurements—using an independent instance of the single-LOM decoder. Since we only decode reliable observables, each instance of the single-LOM decoder can correct every basic error of weight $<d/2$. Therefore, the LOM decoder also has this property for the full reliable circuit $\mathcal{C}$.

It is worth discussing some properties of the LOM decoder. First, the LOM decoder does *not* pass any information between each instance of the single-LOM decoder. Thus, measurements that occur at the same time can be decoded in parallel. A disadvantage however is that the same location in the circuit is typically decoded multiple times by independent instances of the single-LOM decoder, increasing the total space-time cost of the decoder in classical hardware. Another important property of the LOM decoder is that it does *not* explicitly provide a physical correction, i.e., a set of errors consistent with the syndrome, as corrections returned by each single-LOM decoder may not be consistent with each other. This independence also poses issues for decoding near fragile time boundaries as we will see when handling fragile observables in Sec. III B 2 and when developing a windowed decoder in Sec. V.

One can easily generalize the preceding definition of the LOM decoder and run single-LOM decoder instances

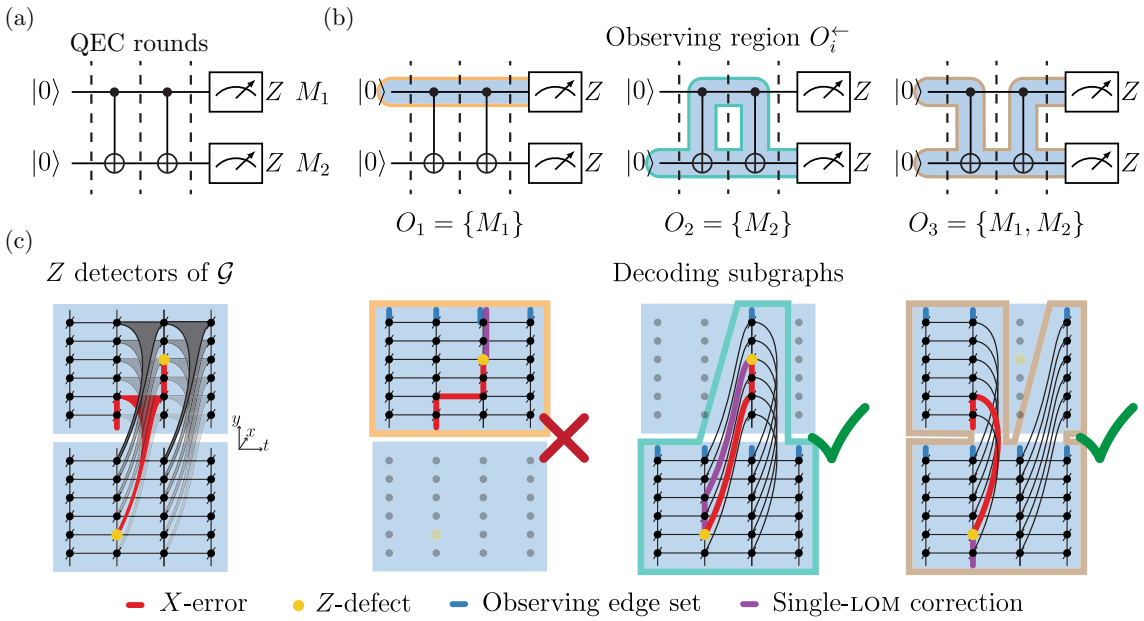


FIG. 5. An example of a weight-5 error in the $d = 7$ surface code in which the decoded outcomes for the reliable observables O_1, O_2, and $O_3 = O_1O_2$ are inconsistent. (a) The circuit consists of two consecutive CNOT gates, with both qubits initialized and measured in the Z-basis. The logical action of the circuit is trivial here and only serves as a simple example. (b) The observing regions of the three observables. (c) The Z-detectors in the decoding hypergraph $\mathcal{G}$. For simplicity, only a slice of $\mathcal{G}$ is shown for a fixed value of the spatial coordinate x. When each of the observables is decoded by the single-LOM decoder, it predicts a flip in the observable O_1 but not in O_2 or O_3, despite the fact that $O_3 = O_1O_2$. This is only possible because the weight of the error—five—is greater than $d/2 = 7/2$. Note that in this figure and throughout, we will refer to measurement errors of Z-stabilizers as X-errors (and vice versa for X-stabilizer errors) for simplicity.

over any independent generating set of reliable observables—that is, any minimal set of reliable observables that generates all possible reliable observables in the circuit. The independence here is desirable since the single-LOM decoder may not return consistent results for observables that are dependent on each other if there are more than $d/2$ errors in the LOM decoding volume. For example, in Fig. 5, the single-LOM decoder predicts a flip to O_1 but not to O_2 or O_3, despite the fact that $O_3 = O_1O_2$. Hence, if in the final measurements of some algorithm, the final outcome is some XOR of final measurements, it is advisable to include this as an observable in the generating set to minimize its logical error rate, see the numerical evidence for this choice of decoding in Sec. IV D 2. This freedom to decode a circuit using any set of (independent) reliable observables is also relevant for handling fragile observables as discussed in the next section.

Before going into this, we note that more realistic circuit-level noise will introduce new hyperedges in the decoding subgraph G_O for the observable O and can make $\mathcal{H}_O$ an observing hyperedge set. However, such hyperedges are decomposable via splitting into a *constant* number of edges that are already present in the decoding subgraph G_O, see Sec. IV B, or decoding G_O can be done with other hypergraph decoding techniques, see Sec. VI.

2. Handling fragile observables

Let us first understand what the problem is when we decode fragile observables using the single-LOM decoder, before we argue that for any circuit we only need to decode an independent generating set of reliable observables. The issue arises due to a combination of two factors. The first of these is a transversal logical reset providing a fragile (open) time boundary (defined in Sec. II C 1) that is followed by a nontrivial logical gate after only $O(1)$ QEC rounds. Second, we decode the corresponding fragile time-boundary multiple times by *independent* instances of the single-LOM decoder. If either of these factors are removed, no issues arise for fragile observables. The issue is closely related to those explored in other recent works [10,11].

Recall that a fragile observable O is such that $O^{\leftarrow}$ anticommutes with a reset stabilizer S, see Definition 1 and Lemma 1. Because $O^{\leftarrow}$ anticommutes with S, this also means that the corresponding logical observing region $\overline{O}^{\leftarrow}$ anticommutes with a weight-2 space-time stabilizer at the start of the circuit, at the fragile time boundary, as shown in Figs. 6(a) and 6(b) [40]. This leads to a somewhat paradoxical statement: applying the weight-2 space-time stabilizer causes an undetectable flip in the decoded outcome of the fragile logical observable $\overline{O}$. Since it is a space-time stabilizer, applying it does not change the state at that

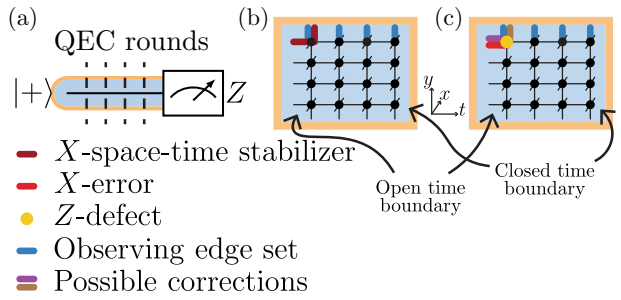
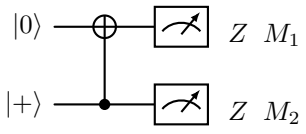


FIG. 6. A simple example of a fragile observable. (a) A $|+\rangle$ reset followed by a Z-measurement, with four QEC rounds in between for ease of visualization. (b) The Z-detectors of the decoding hypergraph $\mathcal{G}$, with an X-space-time stabilizer and the observing edge set of the observable containing the Z-measurement. Because the observable is fragile, the observing edge set has odd overlap with the edges contained in the X-space-time stabilizer. (c) There are two equal-weight corrections to the weight-one measurement error shown, one of which flips the observable while the other one does not. In this simple circuit, it does not matter which decision the decoder makes because the observable has 50-50 measurement statistics anyway.

time step in the circuit, and therefore it cannot change the measurement statistics of any observable. As a result this operator does not constitute a *logical* fault. It does, however, still flip the decoded outcome of $\overline{O}$—this is possible since the statistics of O are 50-50. To illustrate this point, suppose that a weight-one error occurred on one of the two edges contained in the space-time stabilizer, as shown in Fig. 6(c). Then, a decoder has two equally likely corrections it can make, one of which flips the observable and the other of which does not.

At this point, such an ambiguity is not a problem, since the decision the decoder makes does not change the measurement statistics of the observable. A problem does arise, however, when the same fragile time boundary is decoded multiple times by independent instances of a decoder, and the decoder makes a *different* correction in each of these iterations, while the outcomes of the logical observables are supposed to be correlated. Care must therefore be taken to ensure that the decoder decodes all fragile observables in a consistent way.

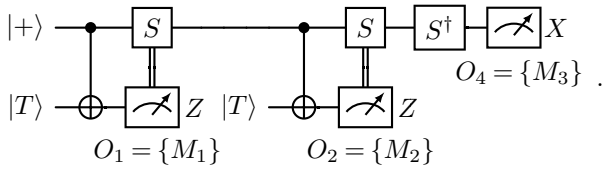
To illustrate the problem, consider a pair of fragile observables O_1 and O_2 whose observing regions both border the same fragile time boundary but whose product is *not* fragile. A simple example is given by the circuit



(1)

in which the observables $O_1 = \{M_1\}$ and $O_2 = \{M_2\}$ are both fragile, but their product $O_3 = \{M_1, M_2\}$ is not. Moreover, in this circuit, O_3 has a deterministic $+1$ measurement outcome. Now, suppose that we use two instances of single-LOM decoders to decode O_1 and O_2 and take their product to infer the outcome of O_3. Because O_1 and O_2 are fragile, a low-weight error ($< d/2$) can cause them to flip. For example, the decoder may correctly infer O_1 but incorrectly infer O_2. This then leads to a flip—and this time a *genuine* logical error—to the product operator $O_3 = O_1O_2$, even though O_3 is not fragile. In Fig. 7, we show two examples of error patterns that can lead to a logical error in O_3 if O_1 and O_2 are decoded using the single-LOM decoder. We can get around this issue by running the LOM decoder as follows. It first randomly selects an outcome for O_1, then decodes the reliable observable O_3 using the single-LOM decoder, and then uses the product $O_2 = O_1O_3$ to infer the value of O_2. This is possible here because both O_1 and O_2 are 50-50 random, but correlated through the outcome of O_3.

In the circuit in Eq. (1) the solution to the problem is fairly straightforward because the pair of fragile observables was measured at the same time; however this need not be the case. For example, the first fragile observable O_1 could condition a Clifford gate that needs to be committed to before continuing the circuit, as shown in the circuit



(2)

In this simple example, the observables O_1 and O_2 are fragile, while their product $O_3 = \{M_1, M_2\}$ is reliable (note that the outcome of O_3 is not deterministic due to the presence of $|T\rangle$-state injections). Note also that the circuit in Eq. (2) simply amounts to preparing $S^\dagger TT|+\rangle = |+\rangle$ before measuring in the X-basis, so that the final observable O_4 is deterministically $+1$. Note that O_4 may or may not be fragile depending on how many of the conditional S-gates have been applied.

Equation (2) causes an issue for both the LOM and a minimum-weight decoder because the decoder needs to be run, first to decide on an outcome for the fragile observable O_1, and then a second time to decide on an outcome for the fragile observable O_2. If the two instances of the decoder give different corrections in the vicinity of the fragile time boundary—for example, due to a weight-1 error string similar to the one in Fig. 6(c)—then this will cause a logical error in the product observable O_3. This is inherently a problem because a logical error to O_3 means that an error has occurred somewhere in the observing region $O_3^{\leftarrow}$, and every possible location has a nontrivial effect on the state

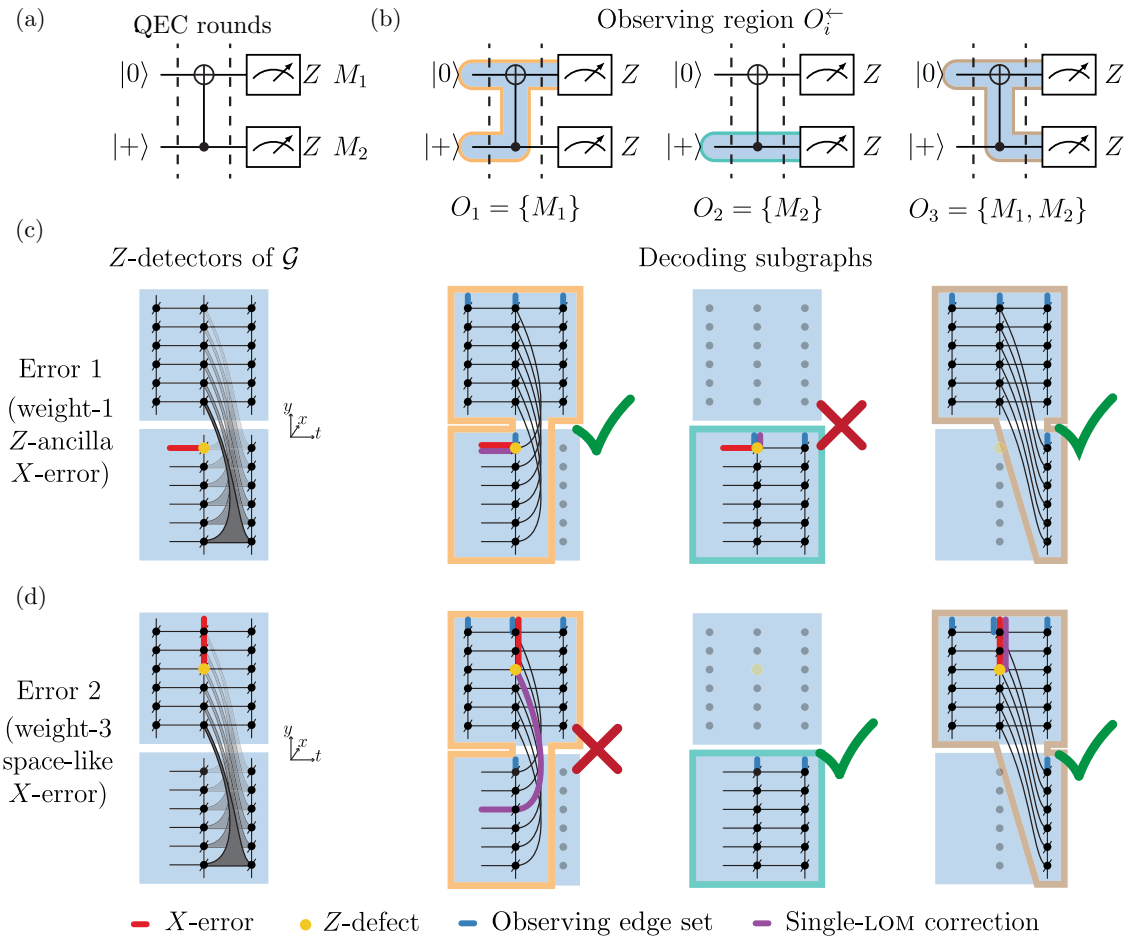


FIG. 7. Examples in the $d = 7$ surface code of the single-LOM decoder applied to potentially hazardous error configurations around a fragile time boundary. In the bare circuit there are three observables that single-LOM decoder can be ran on: O_1 and O_2 are fragile, while O_3 is not fragile and moreover has deterministic $+1$ statistics. We show two examples of errors in the $d = 7$ surface code that do *not* lead to a logical error in O_3 when O_3 is decoded directly by the single-LOM decoder, but *can* lead to a logical error when O_1 and O_2 are decoded separately using single-LOM decoders to infer O_3. Error 1 is a weight-1 measurement error that causes an error in O_3 if the decoders for O_1 and O_2 make different choices about which correction to make—which will occur 50% of the time if minimum-weight corrections are chosen with uniform probability. Meanwhile Error 2 is a weight-3 space-like X-error that will deterministically cause a decoding error in O_1 and O_2, this logical error is even constant weight as the distance of the code increases.

of the system. Indeed, in Eq. (2), an error to O_3 is consistent with an X-error after the first T-injection, meaning that the final state is $S^\dagger TXT\,|+\rangle = S^\dagger\,|+\rangle$, which has 50-50 measurement statistics when measured in the X-basis.

There are multiple solutions to this problem. Perhaps the simplest is to wait $\Theta(d)$ rounds after the transversal logical reset, allowing the state to reach the "quiescent" state [41], and perform an initial window of decoding that commits to a correction in the first $\Theta(d)$ rounds. This works because all of the subsequent instances of the decoder will use the same correction in the vicinity of the fragile time boundary. In the case of the minimum-weight decoder, Ref. [10] proposes a more efficient solution: an additional "consistency check" in the decoder, in which the second instance of the decoder checks whether its correction is consistent with the first and makes an adjustment if necessary before continuing. Related issues around T-injections in the context of a windowed decoder can also be solved by using an additional ancilla per T-injection as proposed in Ref. [11]. Here, we provide yet another solution that is tailored to the LOM decoder.

To decode the circuit in Eq. (2) using the LOM decoder, we use essentially the same procedure as for Eq. (1). We begin by applying the circuit up to and including the measurement M_1. Conventional wisdom would now say that we need to decode the observable O_1 and commit to applying the conditional S gate or not. However, because O_1 is fragile, it can be flipped by a space-time stabilizer without causing a logical error. There is therefore no need to apply a decoder to O_1, and we can uniformly at random select an outcome for O_1 without decoding it, regardless of what the "true" outcome of the measurement was. Then, we

continue executing the circuit up to and including the second measurement M_2. The observable $O_2 = \{M_2\}$ is also fragile and has a 50-50 measurement outcome. However, the product $O_3 = O_1 O_2$ is not fragile and must be decoded. We therefore decode the observable $O_3 = \{M_1, M_2\}$, which is "peeled" off the fragile time boundary, and infer the value of O_2 from the product $O_2 = O_1 O_3$. We can then finish running the circuit, but we now have to interpret the final observable O_4. As mentioned, O_4 may or may not be fragile depending on how many of the conditional S-gates have been applied. If it is not fragile, it can be decoded directly; if it is fragile, we first decode the product observable $O_1 O_4 = \{M_1, M_3\}$ and then multiply its outcome with the outcome of O_1 to obtain the outcome of O_4. In all cases, only a basic error of weight $> d/2$ can cause a flip in the outcome of O_4.

We can generalize the procedure as follows. Each time a measurement M occurs in the circuit, we first check whether the observable $O = \{M\}$ is fragile or not. If O is reliable, then we simply decode it using the single-LOM decoder and continue. If on the other hand O is fragile, we check whether there exists a product with another, earlier fragile observable O' (which can itself be a product of individually measured observables) such that OO' is reliable. If no such O' exists, we uniformly at random select an outcome for O and continue. If such an O' does exist, then we decode OO' and use the outcome of O' to infer the outcome of O. Continuing this way through the circuit allows us to reproduce the measurement statistics of the original circuit fault-tolerantly, without ever decoding a fragile observable. Note that there is no guarantee on the minimum weight of such an operator O', which we will find later to cause problems with the efficiency of the windowed-LOM decoder in Sec. V C 1.

In many cases, we do not need to *actually* randomly sample the measurement outcomes of the fragile measurement. Take, for example, the fragile observable O_1 in Eq. (2). We are not interested directly in the measurement statistics of O_1 and only use it to decide whether the conditional S gate is applied or not. However, the T-injection gadget is *deterministic* after the conditional S gate is applied: it will always apply a T gate to the input state. Therefore, it does not matter whether we choose the outcome to be $+1$ or -1, and we can therefore choose it to be $+1$ without randomly sampling the result. Note that such a trick only works for the first few T-injections in a circuit whose corresponding observables are fragile.

What we have described above is only one way of explaining why fragile observables do not need to be decoded. Here we briefly outline an alternative, entirely equivalent way of understanding this procedure. It turns out that for a fragile observable O that is modeled using a coin toss above, any S gates that are conditioned on O can actually be replaced by a Pauli gate conditioned on O. A simple example of this is the S gate conditioned on O_1 in Eq. (2): because the T gate is being applied to the $|+\rangle$ state, the conditional S gate can be replaced by a conditional X gate. One can generalize this argument to general circuits and show that this replacement can be done for precisely the same observables that we modeled with a coin toss above. One can then track this conditional X gate in software through the rest of the algorithm, possibly updating future observables to include M_1 when they would be flipped by the conditional gate [for example, O_2 in Eq. (2)]. This last process turns out to be exactly equivalent to the procedure described above of multiplying fragile observables together to make new reliable observables.

C. Issues with hierarchical matching decoding

In contrast to the LOM decoder introduced above, recent work [17,19,20] has used a different strategy to decode transversal Clifford gates in the surface code. We refer to this decoder as the *hierarchical* matching decoder (it has also been called a sequential or ordered matching decoder). In this section, we explain why such an approach is *not* fault-tolerant in general, hence justifying our design of the LOM decoder. It is worth noting that we do not rule out the possibility that these issues with a hierarchical matching decoder could be overcome; the goal here is simply to point out what issues need to be solved.

We describe the hierarchical matching decoder in the simplified setting of nonconditional Clifford circuit under the basic error model, as this is sufficient to break its fault-tolerance. The hierarchical matching decoder works by splitting the decoding graph $\mathcal{G}$ into a number of *track* subgraphs $T_\ell = (V_\ell, E_\ell)$ for $\ell = 1, 2, \ldots, L$, each of which is a graph and is hence matchable. The idea is to decode each track T_ℓ sequentially, starting at $\ell = 1$, and committing to a subset of hyperedges in $\mathcal{G}$ at each step. More precisely, we define each track by specifying a vertex subset $V_\ell \subseteq \mathcal{V}$. This vertex subset must satisfy the following criterion:

Criterion 1 (Hierarchical Track Vertex Assignment).

(1) Every vertex $v \in \mathcal{V}$ is contained in exactly one track vertex subset V_ℓ,
(2) For every edge $e \in \mathcal{H}$ with weight $|e| = 2$, both endpoints v, v' of e are contained in the same track $v, v' \in T_\ell$,
(3) For every hyperedge $h \in \mathcal{H}$ with weight $|h| = 3$, there exists ℓ, ℓ' such that $\ell < \ell'$, $|h \cap V_\ell| = 2$, and $|h \cap V_{\ell'}| = 1$.

With Criterion 1, the tracks are defined as

Definition 2 (Hierarchical Tracks). For a set of vertex subsets $\{V_\ell\}_{\ell=1,\ldots,L}$ satisfying Criterion 1, the track subgraph T_ℓ is given by $T_\ell = (V_\ell, E_\ell)$ where the track edges

are given by

$$E_\ell = \{h \cap V_\ell : h \in \mathcal{H}, |h \cap V_\ell| \leq 2\}. \tag{3}$$

We also define the hyperedge subset H_ℓ as the set of hyperedges that correspond to the edges in E_ℓ.

With a set of tracks defined by Definition 2, the hierarchical decoder works by building a correction to the full hypergraph $\mathcal{G}$ "hierarchically"; that is, by decoding each track T_ℓ sequentially, starting at $\ell = 1$, and committing a subset of hyperedges in $\mathcal{G}$ at each step. A simple example is shown in Fig. 8. By the end of the algorithm we will have built a correction $\vec{h} \subseteq \mathcal{H}$ of the full hypergraph. Note that Criterion 1(3) guarantees that the first track T_1 is always matchable. Therefore, we perform matching on T_1,

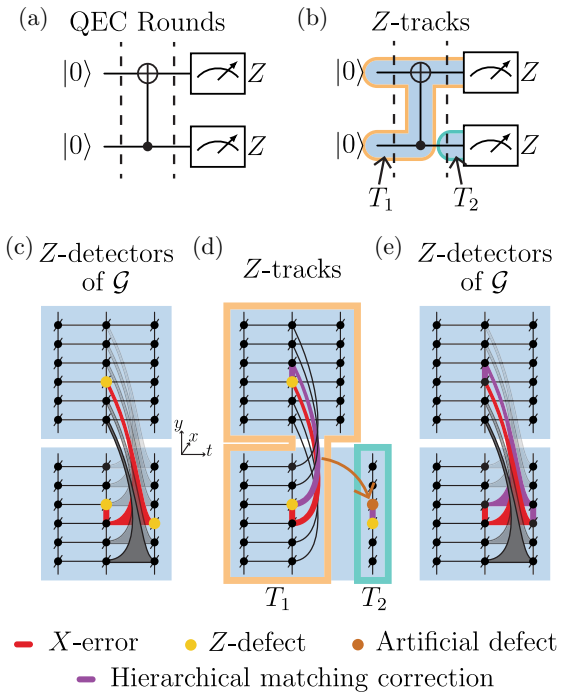
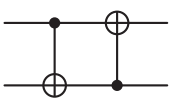


FIG. 8. An example of successful correction by a hierarchical matching decoder. (a) The circuit that we decode. For this example, we only consider X-errors and the Z-detectors of the decoding graph. (b) The circuit locations corresponding to the tracks T_1 and T_2. (c) The Z-detectors of the full decoding hypergraph $\mathcal{G}$, with an example weight-2 X-error pattern. Again, only a slice of the decoding hypergraph is shown for simplicity. (d) The decoding hypergraph is split into the tracks T_1 and T_2. After matching is run on T_1, the decoder makes the correction in purple, which is not exactly the same as the error pattern that occurred. When the correction from T_1 is committed to in the full hypergraph, this introduces an "artificial" defect in T_2. This is then matched in T_2 with the leftover original defect from $\mathcal{G}$. (e) The corrections from T_1 and T_2 are lifted to a correction in the full hypergraph $\mathcal{G}$. Here we can see that the combination of the error and the correction give a space-time stabilizer, similar to Fig. 1 from Supplemental Material [37], but this time in the bulk of the decoding graph.

returning a track correction $\vec{e}_1 \subseteq E_1$. We then commit this correction to the full hypergraph $\mathcal{G}$; that is, for every hyperedge $h_1 \in H_1$ we commit to its inclusion in the correction $\vec{h} \subseteq \mathcal{H}$ depending on whether the corresponding edge is in the track correction $\vec{e}_1$. Then, we update the remaining defects in the graph, flipping them if they are the endpoint of a hyperedge in the correction $\vec{h}$. By definition, this means that there will be no more defects in the first track, but the update could introduce or remove defects in the later tracks (we call these "artificial" defects). We then continue to the second track T_2. Now, note that all hyperedges h with $|h \cap V_2| = 1$ must satisfy $|h \cap V_1| = 2$ and therefore must already be committed to. We can then decode the real and artificial defects in T_2 using matching on the subgraph with vertices V_2 and edges E_2. Repeating this process for all ℓ provides a correction to the hypergraph $\mathcal{G}$, from which the logical action can be inferred.

At this point it is useful to compare the hierarchical matching decoder to the LOM decoder defined in Sec. III. Indeed, the two decoders are closely related: the region of the hypergraph $\mathcal{G}$ decoded in each iteration of the single-LOM decoder always forms a valid choice for the *first* track T_1 of a hierarchical matching decoder because the matchable subgraph G_O for a logical operator O is a valid first track choice. The difference is that the hierarchical matching decoder then attempts to use this correction from T_1 as a starting point for a correction to the full hypergraph $\mathcal{G}$, while the LOM decoder does not use the result of T_1 at all when decoding other regions of the hypergraph. In fact, the reverse is also (almost) true: any valid choice of first track T_1 corresponds to the matchable subgraph G_O corresponding to the backpropagation of some operator O, although this operator may not be an actual observable (for example, the operator to be backpropagated could have an X operator immediately preceding Z-measurement). This point can be seen by noting that after fixing a choice of which time-like edges immediately following a logical gate $\overline{L}$ are included in T_1, there is only one valid choice of time-like edges immediately preceding $\overline{L}$ that can be included in T_1. By inspection of Fig. 2, this unique valid choice corresponds to the backpropagation of the corresponding logical Pauli operator through $\overline{L}$.

We note that some previous work [17,20] only considered situations where the first track only traverses a single logical qubit; this is however not sufficient for decoding arbitrary Clifford circuits. Take, for example, a pair of alternating CNOT gates anywhere in a circuit



(4)

Here it is not possible to choose a track T_1 that traverses only a single logical qubit. It is therefore necessary to consider more general choices of tracks [42], and this brings in

the possibility of the tracks containing loops that we now discuss.

1. Time-like loops

The most serious problem with the hierarchical decoder that we have found concerns the presence of time-like loops. Examples of time-like loops are shown in Fig. 9(a)(i–iii) in an alternating-CNOT experiment. There are three choices of first Z-track shown in Fig. 9(a) corresponding to the three single-LOM instances one can run in the circuit, all of which contain a time-like loop.

More precisely, we say that there is a loop in a track T_ℓ if there exists a set of time-like edges in E_ℓ such that together they do not trigger any defects in T_ℓ. If a nonfinal track T_ℓ with $\ell < L$ contains such a loop, then it can produce artificial defects in later tracks that do not appear in pairs, causing the hierarchical decoder to perform high-weight corrections to match the defects to spatial boundaries. In the wrong configuration, this means that constant-weight errors can lead to logical errors in the later tracks when decoded.

The presence of loops depends on both the circuit and the choice of tracks within the circuit. In the example in Fig. 9, a weight-3 error in the $d = 7$ surface code leads to a logical error, although in this circuit the error is of constant weight and would cause a logical error no matter the distance. Similar examples can also be constructed in other nontrivial circuits by, for example, including a T-injection between two of the CNOT gates. In general, the larger the loop, the larger the weight of the error required to cause a logical error. Any solution to using hierarchical matching will have to address the decoding of loops—perhaps by avoiding them using circuit identities or clever track design, or else by decoding them in some other way—in order to be fault-tolerant.

We briefly remark that time-like loops are not the *only* impediment to fault-tolerance with a hierarchical decoder: low-weight errors arising from fragile time boundaries and time-like snakes similar to what we will discuss later in

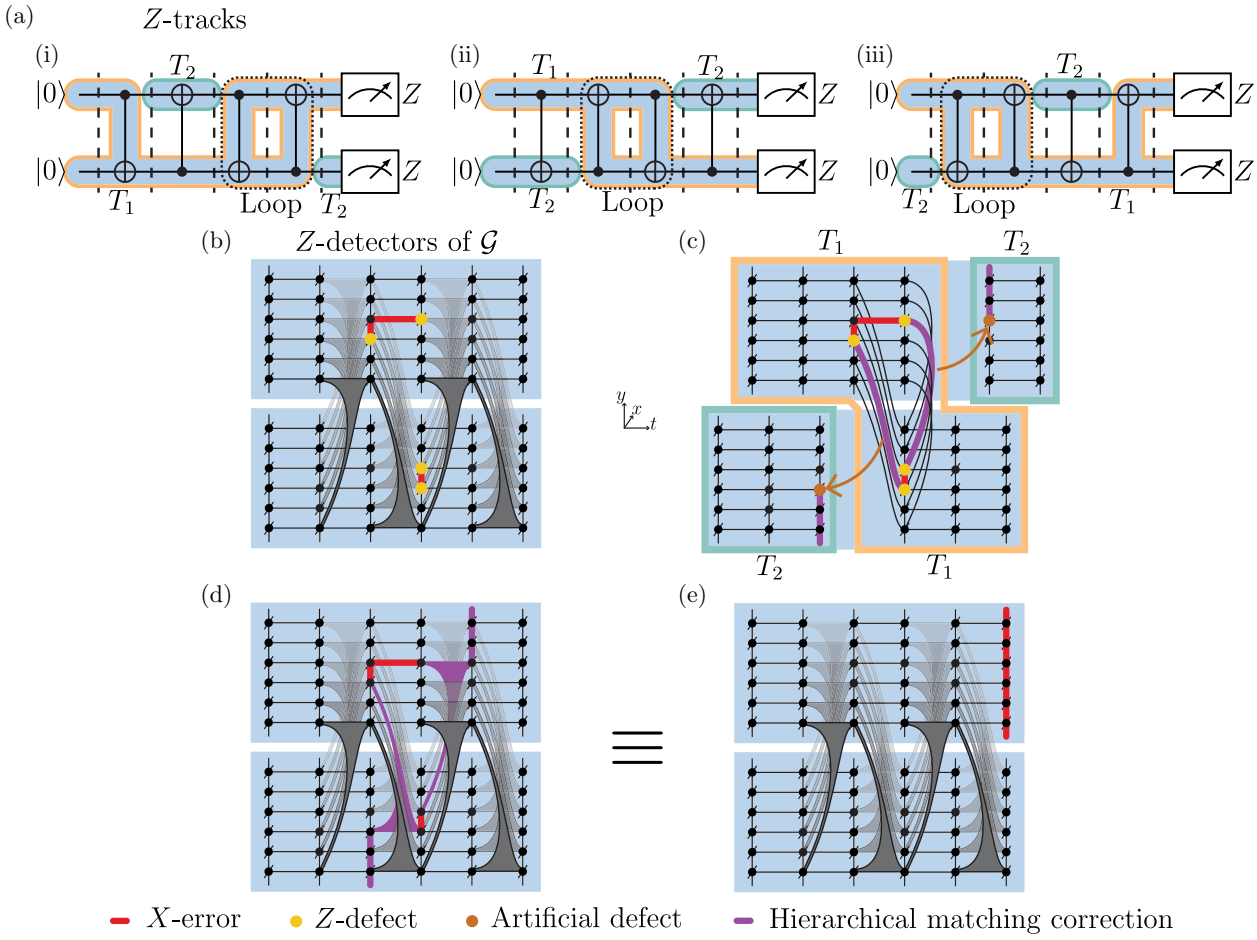


FIG. 9. An example of a failure of hierarchical matching due to the presence of a loop in the first track. (a) The circuit consists of four alternating CNOT gates. Looking only at the Z-stabilizers, all three possible choices of tracks (i–iii) contain a loop in the first track. (b)–(e) For the choice of tracks in (a)(ii), a weight-3 physical error in the $d = 7$ surface code resulting in a logical X-error immediately before the first measurement.

Sec. V can also occur. An example of the issue of fragile time boundaries is shown in Fig. 21 in Appendix A.

IV. NUMERICAL SIMULATIONS

A. Experiments performed

Here we describe the numerical simulations conducted to characterize the performance of our logical observable decoding strategy on sequences of transversal Clifford gates. We use Stim [24] to perform the stabilizer simulations and PyMatching [43] for the MWPM optimization. We have written a wrapper package in Python that builds the encoded circuit for each experiment under the specified noise model, available in Ref. [44], and the decoder software can be found in Ref. [45].

The physical implementation of the fold-transversal $\overline{H}$ and $\overline{S}$ gates is shown in Fig. 1. For the QEC round of the surface code, we use the parallel schedule of Ref. [46] with an ancilla reset after each measurement, see the circuits in Sec. V in the Supplemental Material [37]. We have used a longer schedule than the "standard" one [28,41] to use the same primitive gate set—namely CZ and $R_Y(\pi/2)$ gates—as the current superconducting qubit and neutral atom platforms [47]. The detectors (or checks) correspond to the vertices in the hypergraph and are constructed by tracking the code stabilizers through the fold-transversal operations [10], according to what we call the pre-gate frame, see Sec. II C and Sec. II in the Supplemental Material [37].

The logical circuits used to benchmark the decoder are divided into two classes. Both classes correspond to Clifford circuits without conditional gates because Stim is a stabilizer simulator and it does not have out-of-the-box support for conditional gates (except for Pauli gates, which can be tracked in software). The experiments only include reliable observables because all fragile ones are ignored by LOM decoding, see Sec. III B 2.

The circuits in the first class focus on a single type of gate $G \in \{I, S, H, \text{CNOT}\}$ and are depicted in Fig. 10(a). For each experiment, the considered logical gate is repeated $d+1$ times on an unrotated surface code of distance d, and each time it is followed by one QEC round. This class of circuits is denoted as *repeated-gate experiments*. In the case $G = I$, these circuits correspond to memory experiments, as realized in recent experiments [27,48]. Such similarity allows us to extract a "threshold" for each transversal gate and compare it to the idling or memory experiment.

The input states to the circuits are $|0\rangle^{\otimes n}$ (Z-basis experiment) and $|+\rangle^{\otimes n}$ (X-basis experiment), with $n = 2d^2 - 2d + 1$ for a distance d code, followed by a single QEC round. Since the distance d will be odd, $G^{d+1} = I$, so the ideal final state will be the same as the initial state. We measure the final logical state in the corresponding logical basis, by measuring all data qubits in such basis. Figure 10(a) shows the experiments done in the Z-basis, although we simulate both Z- and X-bases.

1. Arbitrary two-qubit Clifford circuits

The second class of experiments aims to benchmark our decoder on one- and two-qubit circuits built using fold-transversal Clifford gates. The motivation for running this test is to verify that the decoder's performance does not depend on any particular structure of the decoding hypergraph and that our decoder should be useful for experiments running logical randomized benchmarking on one or two logical qubits [49,50].

Logical randomized benchmarking would use random sequences of Clifford gates $C_m \circ C_{m-1} \circ \ldots \circ C_1$ followed by their inverse $(C_m \circ \ldots \circ C_1)^{-1}$, and compile each C_i

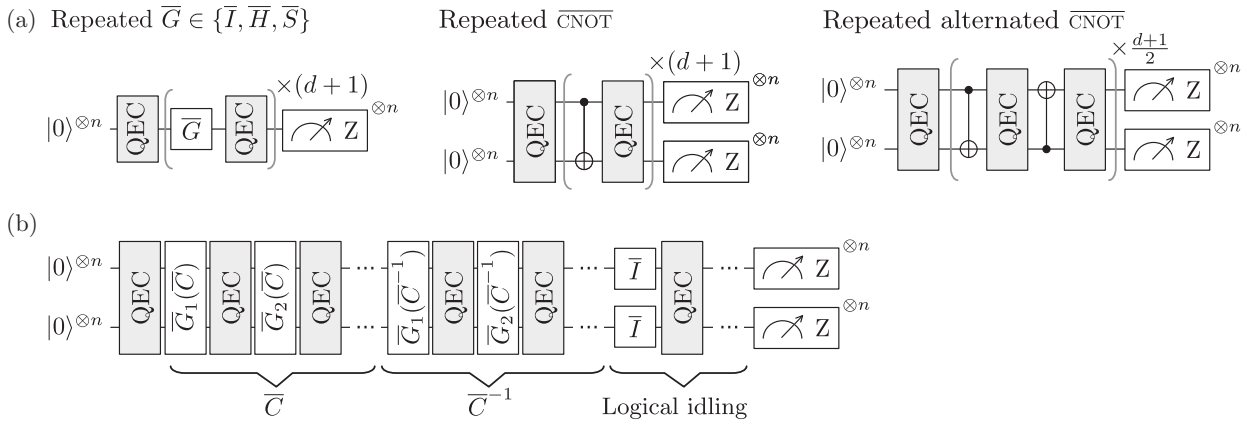


FIG. 10. Logical circuits used to benchmark the decoder in the Z-basis: (a) repeated-gate experiments, (b) two-qubit Clifford experiments. The QEC box corresponds to a single QEC round. The gates $G_i(C) \in \{I, H, S, \text{CNOT}\}$ for $i = 1, 2, \ldots$ are the compilation of the Clifford gate C. The logical idling at the end of the circuit in (b) is used to fix the length to be independent of the Clifford C. The physical implementation of each block is described in Sec. V in the Supplemental Material [37].

into H, S, and CNOT. Instead, here we consider a Clifford gate C and run $C^{-1} \circ C$ where we compile C into H, S, and CNOT, and separately compile C^{-1} into H, S, and CNOT. In principle, one could do this for the whole two-qubit Clifford group, $C \in \mathcal{C}_2$, but since $|\mathcal{C}_2| = 11\,520$, it would be needlessly computationally expensive.

Instead, as logical Paulis are tracked in software, we consider the two-qubit Clifford group modulo Paulis, i.e., $C \in \mathrm{Sp}(4, \mathbb{F}_2)$ [51] which contains 720 elements. In order to further save time, we remove unnecessary SWAP gates from C, as follows. We iterate through all $C \in \mathrm{Sp}(4, \mathbb{F}_2)$ to create a list, and let Stim compile each gate C into H, S and CNOT using Gaussian elimination [24,52]. Then, if some $C' = \mathrm{SWAP} \cdot C$ where C is already on the list, we remove C' from the list, only keeping 360 Clifford circuits in total. For each C on the shortened list, we compute C^{-1} and compile this again into H, S, and CNOT. The maximum CNOT–H–S depth of the compilation of $C^{-1} \circ C$ is $\ell = 14$. For a fair comparison, if the depth falls short of 14, it is increased to 14 by adding logical idling gates. For most of the circuits $C^{-1} \circ C$, the compilation of C and C^{-1} into H, S, and CNOT is such that C^{-1} does not undo each H, S, or CNOT in the order in which they are applied in C, ensuring that the gate sequence and hence the decoding hypergraph does not have a special structure.

Finally, for each choice of Clifford C, we have two associated circuits: one in the Z-basis and one in the X-basis. In the discussion of the numerical results, we index the top (bottom) logical qubit as 1 (2).

We want to note that, although we could have used the deep logical circuits from Ref. [9] (as done in Refs. [25, 33]), their logical measurements are applied as noiseless multiqubit Pauli product measurements, which are different from standard transversal X- and Z-basis measurements. Due to the importance of open-time boundaries in the LOM decoder, we have preferred to use our arbitrary two-qubit Clifford circuits for benchmarking our decoder.

B. Decoding phenomenological and circuit-level noise

Quantum device noise is more complex than the basic error model defined in Sec. II C; in particular, realistic errors can trigger more than three detectors, leading to new hyperedges. In this section, we show how these new hyperedges can be preprocessed before running the LOM decoder. Unavoidably, the logical performance is diminished because correlation information is discarded when decomposing these new hyperedges into multiple edges in order to run a matching decoder. However, these hyperedges can be decomposed into a constant number of edges and therefore do not pose a fundamental fault-tolerance problem, unlike the weight-3 time-like hyperedges in Sec. II C. We use two noise models in our numerical simulations, phenomenological depolarizing noise and SI1000 circuit-level depolarizing noise, defined as follows. In the phenomenological depolarizing noise model:

(1) one applies a Pauli error drawn from $\{X, Y, Z\}$ with probability $p/3$ to each data qubit independently, before a logical gate, before a logical idling step, and before each single QEC round,
(2) one applies an X error with probability p before a physical qubit Z measurement.

The reason to both insert Pauli noise before the logical gate and after the logical gate (i.e., directly before the QEC round) is that these Pauli errors have different effects in the decoding hypergraph, see Sec. II in the Supplemental Material [37]. We choose depolarizing noise in the phenomenological setting instead of independent X and Z errors [12], since it is symmetric with respect to X, Y, Z and our logical Clifford gates transform these Paulis into each other.

We use the SI1000 circuit-level noise model [53] to benchmark our decoder in conditions similar to future experimental realizations of logical circuits. This model is inspired by the superconducting qubit error probabilities, with two-qubit gate infidelities ten times higher than single-qubit gates and five times lower than measurement errors. Although the superconducting qubit platform does not currently have the long-range connectivity required for fold-transversal gates, the mentioned infidelity factors are also similar for neutral atoms, with infidelities of $\sim$0.03% for single-qubit gates, $\sim$0.5% for two-qubit gates, and 1%–5% for measurement operations [54–57]. Therefore, we believe this noise model is a good proxy for near-term realizations of logical experiments with transversal gates.

A natural method of preprocessing the new hyperedges follows the standard procedure for memory experiments [24]. It decomposes the new hyperedges in terms of errors from the basic error model and updates the error probabilities of these basic errors accordingly. We note that there always exists a decomposition into $O(1)$ basic errors for any error in our phenomenological and circuit-level noise models since any circuit-level error can be propagated to a set of $O(1)$ basic errors. Since the LOM decoder can correct any error of weight less than $d/2$, it follows that the LOM decoder can decode any error of weight less than d/c with $c \geq 2$, with the constant c determined by the decomposition of circuit-level errors into basic errors.

In particular, we have applied Algorithm 3 from Ref. [23] to the decoding subgraph to obtain the hyperedge decompositions, although other methods are available [24]. An alternative method is to simply remove the new hyperedges from the decoding hypergraph. It is a reasonable option if the probability update from the hyperedge decomposition is small, or if it is similar for all edges so it leads to just an overall weight rescaling not impacting the matching preference. We have compared this option with

the standard method for the repeated-gate experiments and see a relative difference of $<10\%$ in the logical error probability. In our numerical results, we are then using this alternative method for its simplicity.

1. Circuit distance of the repeated-gate experiments

Prior to decoding any experiments, as a sanity check, we have also computed the circuit distance d_{circ} for the entire repeated-gate experiment circuits for $d = 3$ and $d = 5$, assuming circuit-level noise, by solving the equivalent MaxSAT problem of finding the lowest-weight undetectable logical error, see Sec. IV in the Supplemental Material [37] and Ref. [58]. We numerically found that $d_{\text{circ}} = d$ suggesting that a minimum-weight decoder would correct up to $< d/2$ errors, hence the circuits implementing the logical gates are fault-tolerant. The $d = 7$ computation involves more than 1600 variables and 1700 conditions and was stopped because it did not finish even after more than 20 h. Since we are computing the circuit distance of *the entire circuit*, the decoding volume of the repeated-gate experiments grows as $\Theta(d^3)$, making this problem computationally challenging. Similarly, for the two-qubit experiments, the decoding volume is large because the considered circuits are deep, thus computing their circuit distance is computationally expensive.

In addition to the circuit distance, we have also computed the lowest-weight undetectable error pattern—we call it $\vec{e}_{\text{min}}$—which leads to a decoding failure in the repeated-gate experiments *when decoding with the single-*LOM *decoder*. Note that this error can have weight less than d since we are now using a circuit-level error model instead of the basic error model, and because the single-LOM decoder discards syndrome information since it only considers the decoding subgraphs G_O. As mentioned, the circuit distance of the LOM decoder can be reduced by a constant factor, since we know that the LOM decoder can correct any basic error of weight less than $d/2$ and any circuit-level error can be expressed as $O(1)$ basic errors. For $d = 3, 5, 7$, with phenomenological noise, we have numerically obtained $|\vec{e}_{\text{min}}| = 3, 5, 7$, respectively, across all repeated-gate experiments, verifying that our decoder is circuit-distance preserving. For circuit-level noise and $d = 3, 5$, we get $|\vec{e}_{\text{min}}| = 3, 5$ except for the repeated-$\overline{S}$ experiment in the X-basis where $|\vec{e}_{\text{min}}| = 2, 4$. The weight of this error $\vec{e}_{\text{min}}$ is one lower due to a "hook" error and its error propagation from the long-range CZs in the physical implementation of the $\overline{S}$ gate, see Fig. 2 from Supplemental Material [37]. We discuss the effect of this distance reduction in the numerics.

2. Logical failures

We consider a logical failure to have occurred whenever any of the error-corrected logical outcomes from the experiment do not match the prepared logical input state ($+1$ eigenstate of logical Z or X). The logical error probability, $\overline{P}$, is the probability of having a logical decoding failure. We estimate $\overline{P}$ using Monte Carlo sampling, and we stop when reaching 5×10^4 logical decoding failures or 5×10^9 samples, whichever is achieved first. The uncertainty in the logical error probability is calculated using the Wilson score interval with a 95% confidence level [59,60], which is more reliable at $\overline{P} \to 0$ than using a normal approximation or Wald interval [61]. The threshold values have been estimated using the method from Ref. [62], which is robust when the logical error probabilities around the threshold are close to their saturation value of $1 - 0.5^k$ with k being the number of logical qubits in the experiment.

C. Numerical failure of splitting-hyperedge decoding

To test the performance of the splitting hyperedge decoder discussed in Sec. II C 1, we run the repeated-gate experiments for $G \in \{I, S, \text{CNOT}\}$ using the fold-transversal implementation of these gates and the $d = 3, 5, 7$ unrotated surface codes, under circuit-level depolarizing noise. We decode these numerical experiments in both pre-gate and post-gate frames using a splitting-hyperedge MWPM decoder, in which the hyperedges in the decoding graph are attempted to be decomposed into edges using Stim with

```
stim.Circuit.detector_error_model(
   decompose_errors=True,
   ignore_decomposition_failures=True,
)
```

The logical error probability is shown in Fig. 11. We have also used the "splitting-hyperedge" methods from Ref. [23] and obtained similar logical performance.

As discussed in Sec. II C, this decoding strategy is not fault-tolerant because there exist weight-2 error patterns that lead to a logical error when decoded, for any code distance. Such bad performance is observed in Fig. 11, with the logical error probability scaling as p for small p for distances 3, 5, and 7, and saturating for large p at $\overline{P} \approx 1/2$ for experiments on a single logical qubit and $\overline{P} \approx 3/4$ for experiments on two logical qubits. Note that this is worse than the $O(p^2)$ scaling predicted in Sec. II C due to the presence of circuit-level noise. This $O(p)$ scaling is observed for both pre-gate and post-gate frames. Therefore, the splitting-hyperedge MWPM decoder cannot handle the hyperedges present when performing only 1 QEC round after each logical gate and has motivated us to develop the decoding method in this paper to decode across transversal gates.

We emphasize that if logical gates are separated by $\Theta(d)$ QEC rounds, it could be possible to decode each gate in an individual decoding window using the decoding algorithms from Ref. [20] for the $\overline{S}$ gate and from Refs. [17–19] for the $\overline{\text{CNOT}}$ gate. Or, alternatively, Ref. [9] shows how

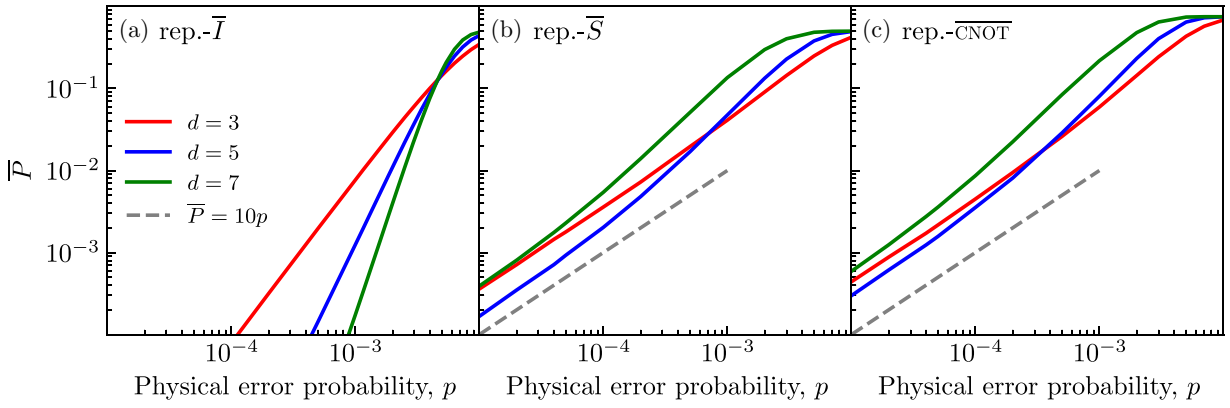


FIG. 11. Logical error probability of the "splitting-hyperedge" MWPM decoder as a function of the physical error probability for some repeated-gate experiments, showing a scaling of $\overline{P}$ with p and no evidence of a threshold. The experiments are run on unrotated surface codes with circuit-level noise, in the $\overline{Z}$-basis, using the pre-gate frame. The results for the post-gate frame are statistically identical. To decode, we use Stim to decompose hyperedges into edges and then run PyMatching, which we call a splitting-hyperedge MWPM decoder. The 95% confidence intervals are shown as shaded regions but they are too small to be visible.

the decoding hypergraph can be modified in this situation so that it can be decoded with a splitting-hyperedge MWPM decoder. However, these decoding methods slow down the logical circuit execution by a factor $\Theta(d)$, thus not exploiting the advantage of fast logic that comes from performing transversal gates.

D. Decoding across logical two-qubit Clifford gates

This section contains the numerical results of our LOM decoder for the two classes of experiments. The circuits have been decoded using both the pre-gate and post-gate detector frames. We have found that the logical error probabilities are statistically identical for both frames in all considered scenarios, thus the discussion and figures only report the results in the pre-gate frame. The reason for such similar performance is that the decoding hypergraphs of both frames differ neither in the number nor in the weight of the hyperedges, but only in which vertices participate in each hyperedge, i.e., they both have weight-3 as in Fig. 2, but just differ in which three vertices form a hyperedge.

We have not implemented the windowed-LOM decoders which will be defined in Sec. V and we leave their numerical benchmarking as future work.

1. Repeated-gate experiments

The repeated-gate experiments correspond to memory experiments with logical gates instead of idling between QEC rounds. Such a similarity allows us to benchmark our decoder on each logical gate individually by comparison to the memory, repeated-$\overline{I}$, experiment. Our decoder for the nontrivial repeated-gate experiments obtains a very similar performance to the repeated-$\overline{I}$ experiments, see Fig. 12 for both phenomenological and circuit-level noise (and Fig. 4 from Supplemental Material [37] for additional results). We observe threshold behavior and overall good logical error suppression. We also run the same experiments with 2 and 3 QEC rounds between the $d+1$ logical gates, but we see worse performance (similar to results in Ref. [9]): this is not surprising as the total sequence of physical gates is simply longer in these experiments with more interspersed QEC rounds.

The experiments with worse performance are the repeated-$\overline{S}$ in the $\overline{X}$-basis and the two repeated-$\overline{\text{CNOT}}$ experiments. Regarding the repeated-$\overline{S}$ experiment, a contribution to the difference between the $\overline{X}$- and $\overline{Z}$-bases is that the number of circuit-level errors that the LOM decoder can correct is smaller by 1 in the former, as described in Sec. IV in the Supplemental Material [37]. Another factor is the difference in the probability of a logical Pauli error that can flip the measurement outcome for both bases. For surface codes, the minimum weight of Pauli $\overline{Y}$ is larger than that of Pauli $\overline{X}$ and $\overline{Z}$ ($2d-1 > d$), hence it is more probable to have $\overline{X}$ or $\overline{Z}$ errors than $\overline{Y}$ errors. For the repeated-$\overline{S}$ experiment in the $\overline{Z}$-basis, the observable to protect throughout the circuit is always $\overline{Z}$, thus we are susceptible to only $\overline{X}$ and $\overline{Y}$ errors. However, in the $\overline{X}$-basis, the $\overline{S}$ gates make the observable change from Pauli $\overline{X}$ to Pauli $\overline{Y}$ and vice versa. Since $\overline{Y}$ is susceptible to both $\overline{X}$ and $\overline{Z}$ errors, this leads to a worse performance when carrying out the experiment in the $\overline{X}$-basis. In the other experiments, the logical gates do not map $\overline{X}$ nor $\overline{Z}$ to $\overline{Y}$, thus this effect is not present, see Sec. VI in the Supplemental Material [37].

Regarding the repeated-gate experiments with $\overline{\text{CNOT}}$ gates, there are two observables to decode: $\overline{Z}_1$ and $\overline{Z}_2$ for the $\overline{Z}$-basis experiment and $\overline{X}_1$ and $\overline{X}_2$ for the $\overline{X}$-basis experiment. Due to the Pauli propagation through the $\overline{\text{CNOT}}$ gates, one of the two observables will have support on both logical qubits at some point in the circuit. This observable is going to be susceptible to logical Pauli errors happening

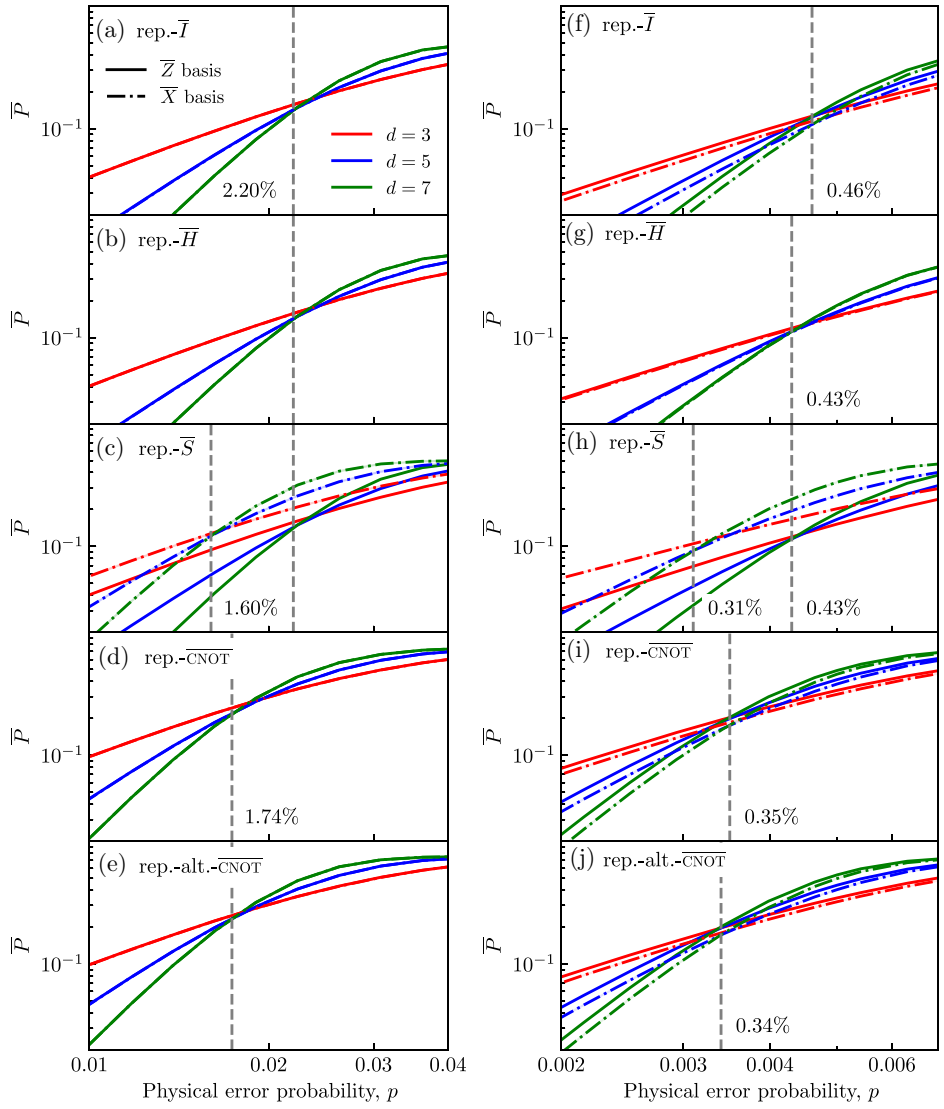


FIG. 12. Logical error probability of the LOM decoder as a function of the physical error probability for the repeated-gate experiments under (a)–(e) phenomenological depolarizing noise and (f)–(j) circuit-level noise for the pre-gate frame. In the experiments, each logical operation is followed by one QEC round, thus $d + 2$ rounds are performed, see Fig. 10(a). The vertical dashed gray lines indicate the crossing of $\overline{P}$ for the $d = 5$ and 7 surface codes. The 95% confidence intervals are shown as shaded regions but they are too small to be visible.

on both logical qubits, compared to just one of them for the memory experiment. Therefore, the repeated-gate experiments with $\overline{\text{CNOT}}$ gates perform worse than the memory experiment. The probability of incorrectly inferring each one of the individual observables is consistent with these arguments and shown in Fig. 13.

We note that the repeated-$\overline{I}$ and $\overline{\text{CNOT}}$ experiments in the $\overline{Z}$-basis have slightly lower performance than in those in the $\overline{X}$-basis under circuit-level noise, see Figs. 12(f) and 12(i)–12(j). Although one might expect the opposite due to the extra layer of H gates during the initialization of $|+\rangle^{\otimes n}$ (see the circuits in Sec. V in the Supplemental Material [37]). References [63,64] explain this discrepancy with the order of the CZs during the QEC round which affects $\overline{X}$ differently than $\overline{Z}$. This is not observed in the repeated-$\overline{H}$ experiment as it symmetrizes $\overline{X}$ and $\overline{Z}$.

2. Preferential treatment

We show numerical evidence for the recommendation mentioned at the end of Sec. III B 1 about how to build the generating set of final observables. In particular, we mentioned that, if the final outcome of an algorithm is some XOR of final measurements, it is advisable to include this as a (reliable) observable in the generating set to minimize its logical error rate. We consider the case of the repeated-alternated-$\overline{\text{CNOT}}$ experiment in the $\overline{Z}$ basis (whose initial logical state is $|\overline{00}\rangle$) under

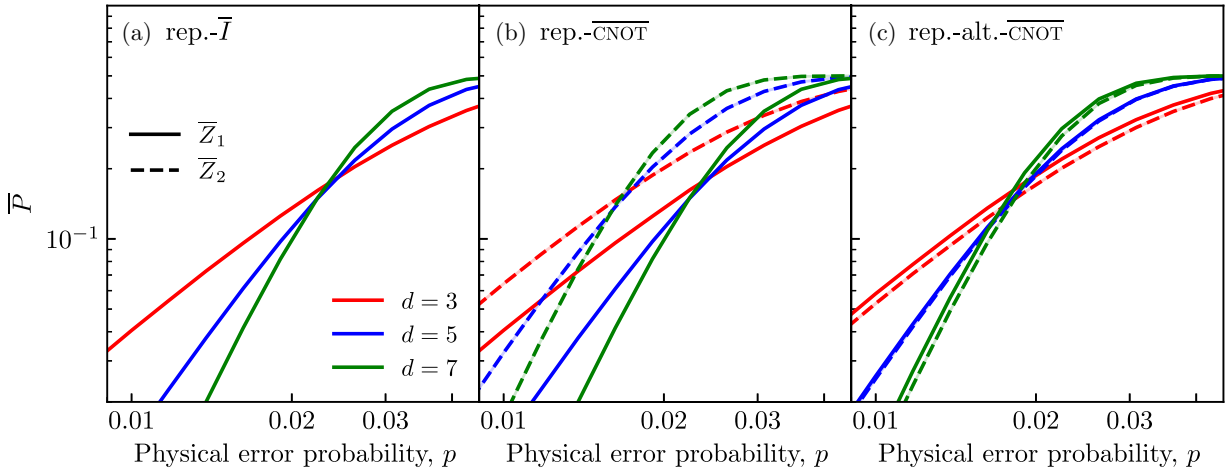


FIG. 13. Focusing on single logical observables for the two-qubit repeated-gate experiments under phenomenological noise. As one may expect, the performance of $\overline{Z}_1$ in the repeated-$\overline{\text{CNOT}}$ experiment in (b) matches the performance of the repeated-$\overline{I}$ experiment in (a) while $\overline{Z}_2$, being sensitive to logical X errors on both qubits, performs worse. For the alternating-CNOT experiment, the logical error rates of $\overline{Z}_1$ and $\overline{Z}_2$ are symmetrized. The 95% confidence intervals are shown as shaded regions but they are too small to be visible.

phenomenological depolarizing noise and use two methods for obtaining the error-corrected $\overline{Z}_1\overline{Z}_2$ outcome, see Fig. 14. The first one corresponds to running a single-LOM decoder for the $\overline{Z}_1\overline{Z}_2$ observable, while in the second one the $\overline{Z}_1$ and $\overline{Z}_2$ observables are decoded independently with single-LOM decoders and then the error-corrected $\overline{Z}_1\overline{Z}_2$ outcome is obtained by taking their XOR. The first method has a slightly better performance.

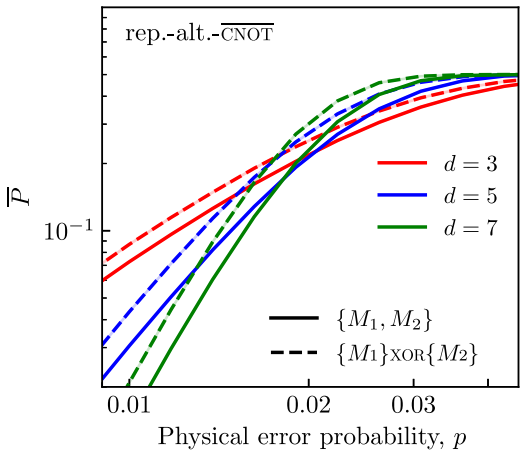


FIG. 14. Performance of the LOM decoder for the $\overline{Z}_1\overline{Z}_2$ observable of the repeated-alternated-$\overline{\text{CNOT}}$ when using two different methods of obtaining the logical correction. The "$\{M_1, M_2\}$" method corresponds to running a single-LOM decoder for the $\overline{Z}_1\overline{Z}_2$ observable. The "$\{M_1\}$XOR$\{M_2\}$" method corresponds to running single-LOM decoders for the $\overline{Z}_1$ and $\overline{Z}_2$ observables separately and then obtaining the error-corrected $\overline{Z}_1\overline{Z}_2$ outcome by taking their XOR. The circuit is run with phenomenological depolarizing noise. The 95% confidence intervals are shown as shaded regions but they are too small to be visible.

3. Arbitrary two-qubit Clifford experiments

In this section, we show that our decoder maintains its performance when decoding all two-qubit Clifford experiments under phenomenological and circuit-level depolarizing noise. In Fig. 15, we report the average (solid line) logical error probabilities across all circuits, as well as the best- and worst-performing ones (shaded region). For phenomenological noise, we observe the same scaling as for the repeated-gate experiments (i.e., $p^{\lceil (d+1)/2 \rceil}$), while for circuit-level noise, the average and worst cases seem to scale as $p^{\lceil (d+1)/2 \rceil - 1}$ due to the presence of $\overline{S}$ gates for which the circuit distance can be reduced by 1 as shown in Sec. IV in the Supplemental Material [37]. The exponential suppression of the physical errors shows that our decoder can decode any two-qubit Clifford unitary and does not exploit any particular structure of the decoding hypergraph.

The crossings of the average $\overline{P}$ [Figs. 15(a) and 15(c)] are hard to see because of the large depth of the two-qubit Clifford experiments. The logical error probability is very close to $\overline{P}_{\max} = 3/4$ at the crossings, which has also been reported by Ref. [9] for these types of (deep) logical circuits. We compute the thresholds using the method described in Ref. [62], which has been designed for this situation of $\overline{P}$ saturating to its maximum close to the crossings. The threshold values for the average $\overline{P}$ across all circuits under phenomenological noise are 2.9576(2)% and 2.8212(2)% for the $\overline{Z}$ and $\overline{X}$ basis (respectively), while for circuit-level noise they are 0.5036(1)% and 0.4963(1)% (respectively). We note that the crossing between $d = 5$ and 7 is at a higher physical error probability than the repeated-$\overline{I}$ (memory) experiment for both noise models: these results are not comparable as in the repeated-gate experiments the depth of the logical circuit increases with

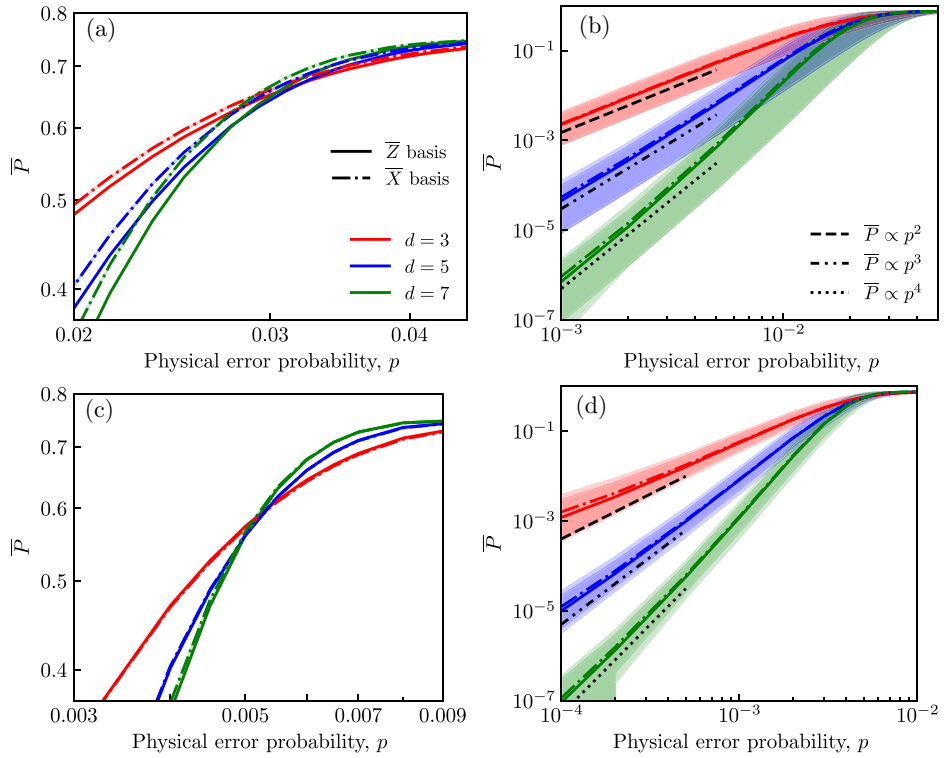
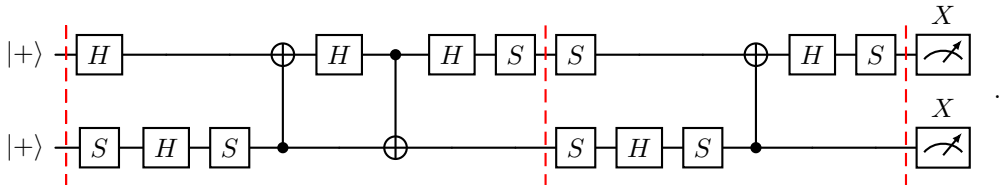


FIG. 15. Logical error probability of the LOM decoder as a function of the physical error probability for the two-qubit Clifford experiments in the pre-gate frame under (a) and (b) phenomenological and (c),(d) circuit-level noise. The solid (respectively, dashed) line corresponds to the average over all experiments in the $\overline{Z}$- (respectively, $\overline{X}$-) basis. The shaded regions in (a) and (c) are the 95% confidence intervals, but they are too small to be visible. The shaded regions in (b) and (d) show the best and worst performance across all circuits.

$d + 1$ while for the two-qubit Clifford experiments it is simply fixed at 14.

Apart from the average logical error probability, we are also interested in the performance difference between the best and worst circuits. We focus the discussion on phenomenological noise because the location and number of noise channels are the same for all two-qubit Clifford experiments, ensuring a fair comparison across circuits. As expected, the best-performing circuit corresponds to the memory experiment while the worst one is



(5)

In this latter circuit, the observables are propagated to both qubits and transformed into Pauli $\overline{Y}$, making them more prone to errors, as discussed in Sec. IV D 1. The factor $\overline{P}_{\text{worst}}/\overline{P}_{\text{best}}$ at a fixed physical probability grows with the code distance, with values 5, 12, and 22 for $d = 3$, 5, and 7, respectively, at $p = 2 \times t10^{-3}$. At such a low physical error probability, $\overline{P}$ is dominated by $C \cdot p^{\lceil (d+1)/2 \rceil}$. The entropic prefactor C depends on the number of lowest-weight error patterns leading to a logical failure. The larger the size of the observing region, the faster C grows with the distance. Therefore, the factor $\overline{P}_{\text{worst}}/\overline{P}_{\text{best}} \sim C_{\text{worst}}/C_{\text{best}}$ increases with the distance at a fixed physical probability.

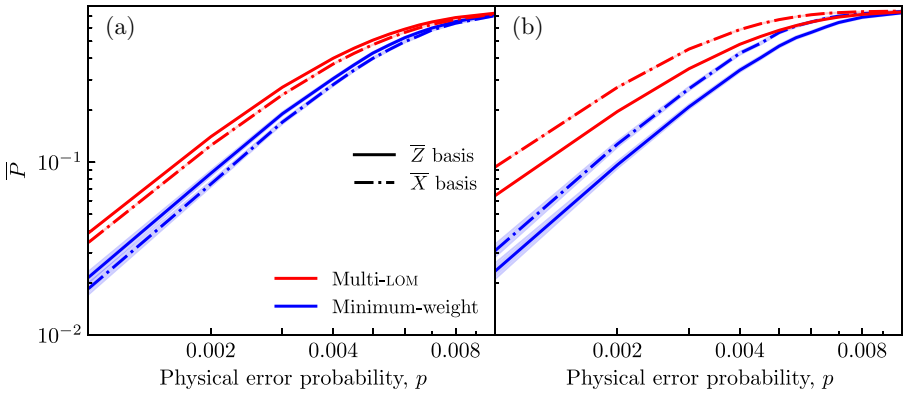


FIG. 16. Performance comparison between LOM and (approximate) minimum-weight decoding for (a) the two-qubit memory experiment and (b) circuit [Eq. (5)] for the distance-3 unrotated surface code under circuit-level noise. The solid (dashed) line corresponds to the experiment in the $\overline{Z}$- (respectively, $\overline{X}$-) basis. The shaded regions show the confidence interval with 95% confidence level. Note that for the LOM decoder, the interval is too small to be visible.

4. Comparison with minimum-weight decoding

We compare the performance of our decoder to minimum-weight decoding, which we implemented in Ref. [65] by solving the equivalent mixed-integer (linear) programming problem [9,66] with the Gurobi solver [67]. Because minimum-weight decoding is computationally expensive, we only decode the best- and worst-performing two-qubit Clifford circuit, that is, a two-qubit memory experiment and circuit [Eq. (5)], respectively, under circuit-level noise, see Fig. 16. For the memory experiment, both decoders perform similarly with just a factor of two difference because minimum-weight decoding does not have to decompose the hyperedges and can process them directly. For the worst-performing circuit, the performance difference increases at lower physical error probabilities due to the worse scaling of the LOM decoder in the presence of $\overline{S}$ gates. We expect the performance of the LOM decoder to be closer to minimum-weight decoding when running belief propagation (BP) as a prestep to update the decoding hypergraph probabilities.

5. Deep Clifford circuits

This section contains a benchmark of the LOM decoder applied to deep Clifford circuits [9], giving a direct numerical comparison with decoding via the Ghost Protocol in Ref. [25]. The goal of this section is to show that one can easily enhance the performance of the LOM decoder by taking into account correlations (e.g., due to Y errors) by running BP in the *full hypergraph* as a preprocessing step to optimize error probabilities in the LOM decoding graphs. The Ghost Protocol decoder already tries to account for these correlations directly. In our BP+LOM decoder, if the belief propagation does not converge after n_{BP} iterations, then LOM is executed using the updated error probabilities from the last belief-propagation iteration.

A preparation step involves initializing four surface code logical qubits in $|\overline{+}\rangle^{\otimes 4}$ followed by 32 logical gate layers, where each layer is composed of random transversal $\{H, X, Y, Z\}$ gates followed by random transversal CNOTs, see Ref. [9] for more information. One then

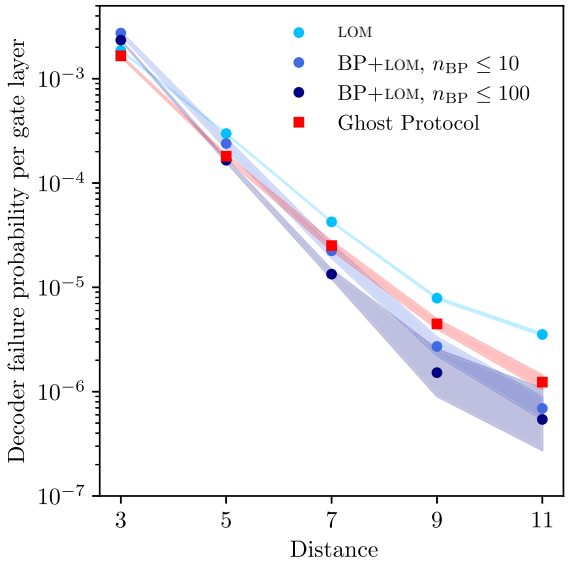


FIG. 17. Decoder failure probability per gate layer of the LOM, BP+LOM, and Ghost Protocol [25] decoders for the deep Clifford circuit with a single QEC round per gate layer and noise model from Ref. [25]. The circuit has 32 gate layers, where each layer is composed of random transversal $\{H, X, Y, Z\}$ gates followed by random transversal CNOTs on each qubit. All distances have been simulated with the same noise strength $p = 10^{-3}$. The parameter n_{BP} in BP+LOM corresponds to the maximum number of belief-propagation iterations that are run. The shaded regions show the confidence interval with 95% confidence level. Note that, for $n_{\mathrm{BP}} = 100$ at distances 9 and 11, the confidence intervals are quite large as only $\sim$10 decoding failures have been sampled due to longer decoding runtimes.

decodes the multi-Pauli (logical) stabilizers of the corresponding four (logical) qubit state, which should ideally have outcome $+1$. We consider a decoding failure to have occurred whenever any of the error-corrected logical outcomes from the circuit do not match the ideal outcome $+1$. The noise model used in the simulations of these deep Clifford circuits is the one from Ref. [25], with the same physical error rate $p = 10^{-3}$ across all code distances, and the random circuit that is implemented is identical to the one in Ref. [25].

The performance of the LOM, BP+LOM, and Ghost Protocol decoders on deep Clifford circuits with a single QEC round per gate layer is shown in Fig. 17. As expected,BP+LOM is more accurate than LOM due to BP taking into account correlations, with its performance increasing with the number of iterations. In particular, very few belief-propagation iterations can reduce the decoder failure probability by an order of magnitude at high distances, showing that little correlation information can heavily improve the LOM performance. Finding a faster and more efficient method for this purpose is an open question and beyond the scope of this work.

V. WINDOWED LOGICAL OBSERVABLE MATCHING DECODING

The LOM decoder discussed so far is not a scalable approach to decoding because each instance of the single-LOM decoder is independent and will therefore need to decode the circuit all the way back to the initial time step. To tackle this problem we introduce a *sliding window* version of the LOM decoder that we call windowed-LOM decoder for short. This works by splitting the circuit into sliding windows in time, with each window decoded by splitting it into multiple single-LOM decoding instances and committing to a part of the correction that they make. This way, later iterations of the single-LOM decoder can use the previously committed corrections to reduce the space-time volume of the decoding subgraph.

We wish for the windowed-LOM decoder to satisfy three properties. First, that it is computationally efficient to run, growing at most polynomially in the distance d of the code independent of the number of logical qubits and the depth of the circuit. Second, that it is able to correct any basic error pattern with weight $<d/2$. And finally, we wish for this to happen while maintaining the ability to perform fast resets, gates, and measurements. Unfortunately, we are unable to give a construction where all these properties are satisfied simultaneously. Instead, we provide a number of versions of the windowed-LOM decoder, each of which has different computational complexity, fault-tolerance, and requirements on the circuit; these are summarized in Table I.

We begin pedagogically with the simplest version of the windowed-LOM decoder that we call the "basic" windowed-LOM decoder. In this setting, we make some fairly strict assumptions about the circuit; namely, that resets are not fast, but slow, using $\Omega(d)$ rounds of QEC to reach the "quiescent" state before starting to interact with other logical qubits. With this, the basic windowed-LOM decoder is computationally efficient, but it cannot always correct up to $d/2$ errors.

TABLE I. Summary of the windowed-LOM decoder variants introduced in this section. Here, "efficient" means that the decoder runs in polynomial time in d; the asterisk here is to indicate that the basic windowed-LOM decoder is not efficient for some unphysical circuits. "Fault-tolerant" means that the decoder can always correct basic errors of weight less than $d/2$. All the variants can handle fast gates and T-injections, and the last column indicates whether they can also handle fast resets and measurements.

Decoder	Efficient	Fault-tolerant	Resets and measurements
Basic	Yes*	No	Slow
Sec. V B	Sec. V B 1	Sec. V D	
Two-step	No	No	Fast
Sec. V C	Sec. V C 1	Sec. V D	
Basic +	Yes*	Conjectured	Slow
short-cut edges	Sec. V B 1	Appendix B 3	
Two-step +	No	Conjectured	Synchronized
short-cut edges	Sec. V C 1	Appendix B 3	

We solve the issue of slow resets in Sec. V C by introducing a second step to the windowed-LOM decoder to give what we call the *two-step* windowed-LOM decoder. However, the second step is not computationally efficient in general, and the two-step windowed-LOM decoder suffers from the same issues of fault-tolerance as the basic windowed-LOM decoder.

In both the basic and two-step windowed-LOM decoders, the same two modifications need to be made for it to be able to correct $d/2$ basic errors. First, it requires the "synchronization" of resets and measurements to occur just before or after each so-called window boundary, respectively. Notably, this synchronization is *not* necessary for the measurements occurring in T-injections which can occur at any time in the circuit, but it does mean that algorithms with lots of other resets and measurements (such as running parity-check circuits of some code using logical surface-code qubits) will either be slowed down or will require additional idling logical qubits. And second, it requires the addition of extra edges in each single-LOM decoder that we call "short-cut" edges. With these modifications, we conjecture that the windowed-LOM decoder can always correct any number of basic errors $<d/2$.

This section is structured as follows. First, we briefly review the general structure of sliding window decoders, before explaining how the basic windowed-LOM decoder works. Then, we discuss the problems with the basic

windowed-LOM decoder and their respective solutions, first discussing how fast resets compromise the computational efficiency of the windowed-LOM decoder, and then why the basic windowed-LOM decoder cannot correct up to $d/2$ basic errors. We leave a thorough optimization of each variant of the windowed-LOM decoder to future work.

A. Sliding window decoders

We review how the sliding-window matching decoder works for a surface code memory experiment [12], as shown in Fig. 18 (or see Fig. 2 in Ref. [30]). Each window consists of a *commit* region followed by a *buffer* region, with both regions having width $> d/2$ in time. Each window has a backward time boundary at the start of the commit region, and a forward time boundary at the end of the buffer region. The backward time boundary of each window is closed—i.e., it has no time-like edges connected to the boundary node—because we know the past, while the forward time boundary is open—it has time-like edges connected to the boundary node, similar to a fragile time boundary, as we do not know the future. We call the boundary between the commit and the buffer regions the *center* of the window, which coincides with the backward time boundary of the next window. We label the times associated with the backward time boundary, the center of the window, and the forward time boundary t_{prev}, t_{center}, and t_{current} (respectively).

The decoder is then run sequentially on overlapping windows to build a correction in the full hypergraph. Each time we commit to a correction, we update the defects in the decoding graph that still need to be corrected by flipping the vertices (defect $\leftrightarrow$ no-defect) that are the endpoints of edges in the committed correction. This guarantees that after committing to the correction, there are no more defects in the commit region of the window. This procedure can introduce new defects in the rest of the graph that we call *artificial* defects which will be seen and decoded by the next window.

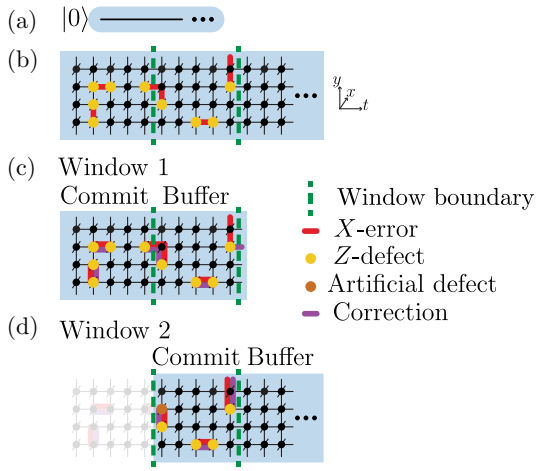


FIG. 18. The operation of a sliding window matching decoder on a Z-basis memory experiment, shown in (a), in the $d = 5$ surface code. (b) The Z-detectors in the first rounds of the full decoding graph with an arbitrarily chosen set of X-errors. The decoding problem is split into overlapping windows, the first two of which are shown in (c) and (d), including their respective buffer and commit regions and the correction that they apply. Note that only the correction in the commit region of window 1 is committed to, giving rise to an artificial defect in window 2.

B. The basic windowed-LOM decoder

We now define the basic windowed-LOM decoder for a circuit $\mathcal{C}$ with fast gates and measurements, but slow resets. In particular, we assume that each reset is followed by $\Omega(d)$ rounds of QEC to allow it to reach the "quiescent" state. During this $\Omega(d)$ delay, before the windowed-LOM decoder is run, a round of normal sliding-window decoding is performed that commits to the correction in the vicinity of the fragile time boundary. As a result, we assume that there are no open time boundaries arising from resets in the windowed-LOM decoder, allowing us to safely decode all observables regardless of whether they are fragile or not. Remember that we had to handle fragile observables separately in Sec. III B 2 for the LOM decoder, thus making reliable observables potentially be of high-weight, supported on many logical qubits.

The job of a windowed decoder is threefold:

(1) To infer the artificial defects in the following window. This only requires fixing the correction to the time-like edges *at* the time t_{center}, but not before.
(2) To update the "logical Pauli frame" of the computation up to time t_{center}; that is, for each logical Pauli operator, we must commit to whether an error has occurred before time t_{center} in its observing region.
(3) To commit to all the logical measurement outcomes of the circuit in order to perform any gates that are conditioned on these outcomes. This must be done immediately to allow for fast T-injections.

To execute these three jobs, we structure the windows as follows. Just like in the memory experiment, in the windowed-LOM decoder we define a set of window boundaries separated by $>d/2$ rounds of QEC. As we execute the circuit, we run a window of the windowed-LOM decoder each time we reach a window boundary, with the buffer and commit regions defined in the same way as in Sec. V A.

In particular, the basic windowed-LOM decoder must commit to the time-like edges and the logical Pauli frame. To do this, we run $2k$ independent instances of the single-LOM decoder, where k is the number of qubits that are active at the center of the window, see Fig. 19. Each single-LOM decoder is obtained by propagating a single-logical-qubit $\overline{X}$ or $\overline{Z}$ operator at t_{center} *forward and backward* through the window. Once a correction is obtained, we

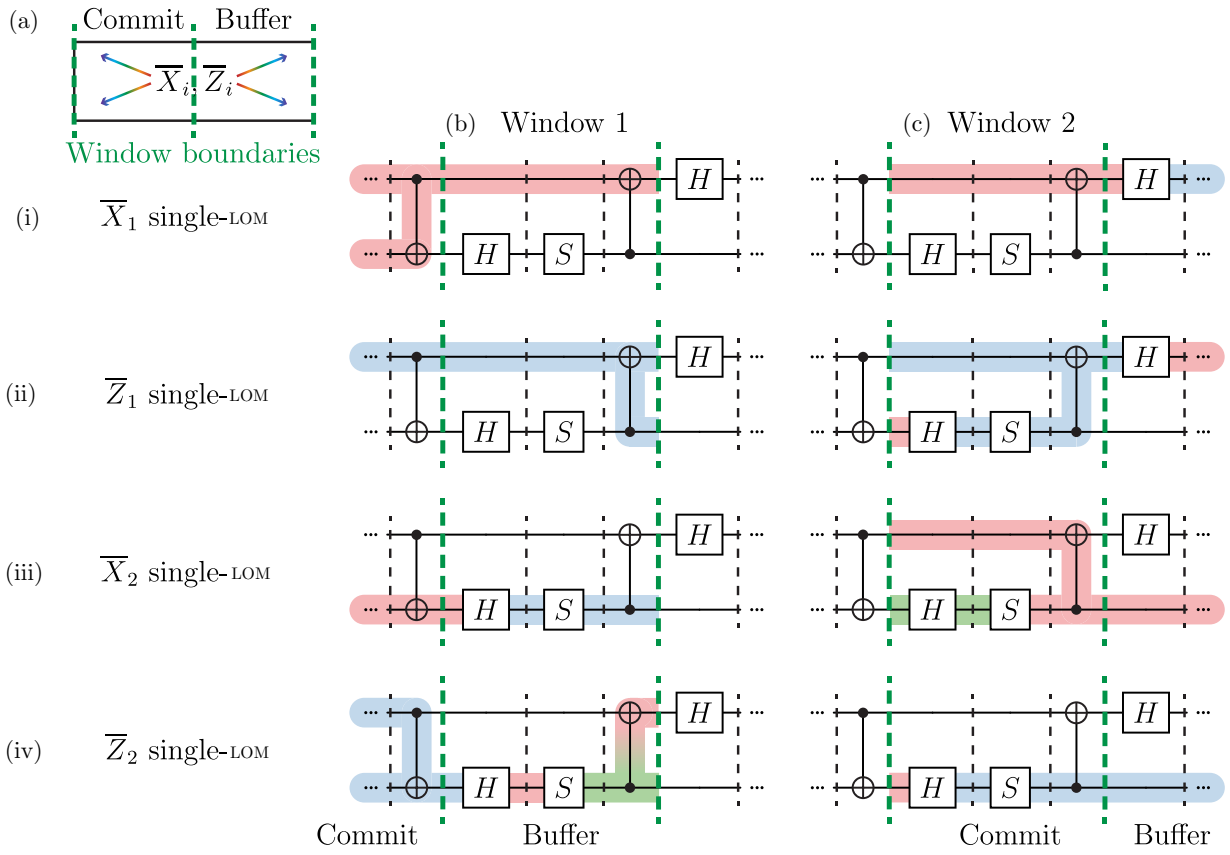


FIG. 19. (a) A visual representation of the windowed-LOM decoder, including both the buffer and commit regions. In each window, a single-LOM decoding instance is run on the track obtained by forward- and backward-propagating a single-logical-qubit $\overline{X}$ or $\overline{Z}$ from the center of the window. (b) and (c) A detailed example of the single-LOM tracks in two windows of the windowed-LOM decoder implemented on the $d = 3$ surface code, where red, green, and blue regions represent X, Y, and Z Pauli support, respectively.

commit to the time-like edges *at* the time t_{center}, but not before. This is well-defined because at each time t_{center}, each time-like edge is, by definition, contained in only one single-LOM. Moreover, we can use the single-LOM decoder to update the logical Pauli frame of the observable. That is, based on whether the correction intersects an even or odd number of times with the observing edge set in the commit region of the current window, we decide whether or not the logical Pauli operator has flipped between t_{prev} and t_{center}. From this, we can determine whether the logical Pauli operator has been flipped through the *whole* circuit up until $t = t_{\text{center}}$ by looking at whether the (backpropagation of) the Pauli operator was flipped in any of the previous windows. It is important that each single-LOM has support only on a single-logical-qubit $\overline{X}$ or $\overline{Z}$ operator at the center of the window to ensure fault-tolerance, as will be discussed in Sec. V D.

However, in addition to running windows of decoders, we immediately run additional single-LOM decoders for every logical measurement that occurs. This is important because the outcome may condition a Clifford gate. To do this, we simply run a single-LOM decoder on the measurement back to the last point in the circuit where the correction has been committed to. To put this in context, the most-recent window boundary that we have passed represents the forward time boundary of the most-recently decoded window, and the correction has been committed up to the second-most-recent window boundary. Note, of course, that because we have slow resets, even measurements that are fragile can be decoded in this way, because there are no more fragile time boundaries from the slow resets as described above. We then use this single-LOM decoder to commit to the measurement outcome, without committing to any edges in the correction. This is advantageous in the presence of T gates, where the subsequent conditional S gate can thus be performed without delay. The windowed-LOM decoder therefore does not require any additional ancilla qubits to perform fast T gates, in contrast to the windowed minimum-weight decoder considered in Ref. [11].

1. *Efficiency of the basic windowed-LOM decoder*

The basic windowed-LOM decoder is highly parallelizable but is limited by the maximum time it takes to decode a single logical observable. The maximum space-time

decoding volume of each single-LOM decoder is bounded by the number of logical qubits that a single-logical-qubit Pauli operator propagates to in the forward- or backward direction within the window. To quantify this, we define $f(t)$ as the maximum number of qubits that a single-qubit Pauli operator can spread to in a circuit of depth t. Each single-LOM propagates from a single qubit through a circuit of depth $t = \Theta(d)$. If there are no limitations on the circuit, then $f(t) = 2^t$, and the volume of the decoding graph grows exponentially in d, as $O(d^3 2^{\Theta(d)})$. Note that for scrambling circuits, which could be Clifford circuits, such spread is maximized [68].

However, in any physical implementation of the circuit, we must have $f(t) = O(t^D)$ for some integer D. This is true even in platforms without fixed connectivity such as neutral atoms because there is a limit to the distance that an atom can travel in preparation for a CNOT gate before it accumulates enough noise that a QEC round needs to be performed. In such physical cases, the volume of the decoding graph grows only polynomially as $O(d^{3+D})$.

We remark that just because the decoder is efficient does *not* mean that it avoids the decoder backlog problem as described in Ref. [69], which instead would require some kind of parallel windowed decoder [30].

C. The two-step windowed-LOM decoder

In Sec. V B, we argued that the basic windowed-LOM decoder is computationally efficient when the circuit contains only slow resets, whenever the circuit is implemented on a physical device. It is worth noting that slow resets may not be much of a hindrance for many applications. For example, if the desired algorithm is written in terms of Clifford and T gates with resets and measurements only occurring at the start and end of the algorithm, respectively, then slow resets and synchronized measurements only *add* an $O(d)$ time cost to the start and the end of the algorithm. This cost is likely to be small compared to the time it takes to execute the gates in the circuit.

Nevertheless, in this section we explain how to modify the windowed-LOM decoder to handle fast resets using a second step in each window, and why this comes at the cost of the efficiency of the decoder. The resulting two-step windowed-LOM decoder can decode circuits with fast resets, measurements, and gates, but is computationally inefficient in general. We leave solving this efficiency issue as an open problem.

The key issue that the two-step windowed-LOM decoder needs to solve is how to handle fragile observables. Recall that the basic windowed-LOM decoder avoids this problem because the low resets mean that even fragile observables have no open time boundaries on resets, and therefore can be safely decoded. Now, with fast resets, we must adapt the techniques developed in Sec. III B 2 to work for the windowed-LOM decoder. In particular, we need to be careful about decoding any observables that are *commit-region fragile*: that is, the observable anticommutes with a reset occurring in the commit region between t_{prev} and t_{center}. Indeed, if the observable anticommutes with a reset that occurs before the backward time boundary t_{prev}, then the corresponding single-LOM will not contain the open time boundary caused by that reset and it can therefore be decoded normally. We will refer to observables that are not commit-region fragile as commit-region reliable.

Dealing with commit-region fragile observables is complicated because of the multifaced nature of the windowed-LOM decoder: each window needs to commit both to the time-like edges at t_{center} and to the logical Pauli frame flips in the commit region (we will return to the third job of the windowed-LOM decoder—committing to logical measurement outcomes—later). Importantly, the single-LOM decoding instances that are used to commit to the time-like edges must *only* have support on a single-logical-qubit $\overline{X}$ or $\overline{Z}$ operator at the center of the window. This is necessary to provide a unique assignment of the time-like errors, and to avoid time-like loops as we will discuss in Sec. V D. On the other hand, the single-LOM decoding instances that are used to commit to the Pauli frame flips cannot be commit-region fragile. These two conditions cannot be satisfied simultaneously in general, necessitating the use of two steps in the decoder as we now explain.

In the first step of the two-step windowed-LOM decoder, we decode observables that commit to the time-like edges at t_{center} that have support only on a single-logical-qubit $\overline{X}$ or $\overline{Z}$ at t_{center}. This step is the same as in the basic windowed-LOM decoder, except for the fact that we do not commit to the logical Pauli frame flips in the commit region. Then, in the second step, we decode observables that commit to the logical Pauli frame flips, in particular by picking an (independent) generating set of commit-region reliable observables. For each generating commit-region reliable observable, we use a single-LOM defined by the backpropagation of that observable that spans *only* the commit region of the window; that is, from $t = t_{\text{prev}}$ to $t_{\text{center}} - 1$. This single-LOM has closed boundaries on both the forward and backward time boundaries, where the artificial defects on the forward time boundary are inferred from the committed time edges in the first step of the windowed-LOM decoder. The outcome of this single-LOM decoder can then be used to update the logical Pauli frame in a way that is consistent with the time-like edges committed to in the first step. An independent generating set of commit-region fragile observables can then be assigned random values based on a coin toss to complete the logical Pauli frame assignment.

This procedure can be simplified somewhat if some of the observables from the first step are already commit-region reliable, or can be made commit-region reliable by combining with measurements that occur in the commit region. In this case, the logical Pauli frame can already be

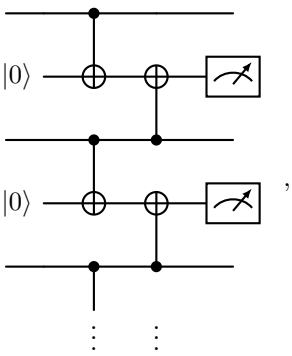
committed to in the first step, and these observables do not need to be decoded again in the second step.

Finally, we return to the issue of the decoding of logical *measurements* in the presence of fast resets. Recall that we decode these logical measurements immediately after they occur so we can apply any gates that are conditioned on the measurement outcome. As a result, the single-LOM spans from t_{prev} (the time up to which the most-recent window has committed to) to the time of the measurement; in this context we call this the commit region of the single-LOM, although recall that it only commits to the logical Pauli frame flips of the measured observable and not to any time-like edges. Similarly to in the LOM decoder, we need to take into account the possibility that the observable is commit-region fragile but "correlated" with another observable that is commit-region fragile.

To handle this, we follow a similar procedure to the one in Sec. III B 2. In particular, each time a measurement M occurs in the circuit, we first check whether the observable $O = \{M\}$ is commit-region fragile or not. If O is commit-region reliable, then we simply decode it using the single-LOM decoder and use the previously committed logical Pauli frame to obtain the outcome O. If O instead is commit-region fragile, we check whether there exists a product with another, earlier or simultaneous observable O' that is also commit-region fragile, such that OO' is commit-region reliable. If no such O' exists, we uniformly sample an outcome for O and continue. If such an O' does exist, then we decode OO' and use the outcome of O' and the previously committed logical Pauli frame to infer the outcome of O.

1. Efficiency of the two-step windowed-LOM decoder

Now that we have described the action of the two-step windowed-LOM decoder, we explain why the decoder is not efficient in general. The issue is with the second-step single-LOMs that are obtained by the backpropagation of a multilogical-qubit Pauli operator. As foreshadowed in Sec. III B 2, there is no general upper bound on the weight of this operator, even for low-depth local circuits. Take, for example, the following repetition-code style circuit



, (6)

which is of constant depth and local in 1D. If we have a window boundary immediately after these logical measurements, the lowest-weight reliable observable that contains $\overline{X}_1$ is the product of *all* remaining $\overline{X}$ operators. The volume of the corresponding second-step single-LOM therefore scales with the total number of logical qubits in the circuit. On the other hand, it may possible that in certain circuits, such as those obtained by concatenating a surface code with another QEC code, that the weight of these reliable observables can be bounded. We leave it as an open problem how to solve this efficiency issue with the two-step windowed-LOM decoder.

D. Issues with fault-tolerance

The basic and two-step windowed-LOM decoders both suffer from a number of issues relating to its fault-tolerance that we now address in turn. In particular, each issue has the potential to cause the windowed-LOM decoder to be unable to correctly decode an error of weight $<d/2$. Intuitively, these arise because we are using the outcomes of a single-LOM decoder to infer artificial defects that are then seen by other single-LOM decoders, and we bump into issues that are thus similar in nature to what we may encounter for a hierarchical decoder. However, for both the basic and two-step windowed-LOM decoders, we are able to solve each of these problems with some appropriate modifications, and we *conjecture* that with these modifications the windowed-LOM decoder is able to correct $<d/2$ basic errors. We note that this discussion is *not* the same as asking whether the windowed-LOM decoder does, or does not, have a threshold either with or without the modifications, which we leave as open problems.

1. Time-like loops

First, we explain that the windowed-LOM decoder does *not* fail due to the presence of time-like loops, unlike the hierarchical decoder Sec. III C 1. It is true that time-like loops may be present in each single-LOM track, *and* it is also true that edges from the single-LOM are used to infer artificial defects in the next window. However, by definition, in any single-LOM that is used to commit to time-like edges, any such time-like loop cannot traverse the center of the window from which the artificial defects are inferred, since at this point each single-LOM only has support on a single logical qubit $\overline{X}_i$ or $\overline{Z}_i$. Time-like loops therefore do not lead to constant-weight logical decoding errors in the windowed-LOM decoder. This is important because the remaining issues that we discuss below are not as serious as that of time-like loops: the smallest number of basic errors required to cause a logical error in the windowed-LOM decoder increases with d, even if it is less than $d/2$.

2. Fragile time boundaries

In contrast, the presence of fragile time boundaries can cause an issue for the windowed-LOM decoder because they can cause the time-like edges to be incorrectly inferred, as shown in Fig. 20. Because we propagate operators forward *and* backward in the windowed-LOM decoder, this problem can even arise in the basic windowed-LOM decoder even though resets are slow due to fragile time boundaries on *measurements*. In the worst case, this can allow errors of weight $d/4 + O(1)$ to cause a logical error in the windowed-LOM decoder. Note that this is an entirely separate issue to decoding fragile *observables* (which are not present in the basic windowed-LOM decoder due to the slow resets) because this problem relates to inferring the committed time-like edges instead of the committed logical Pauli frame.

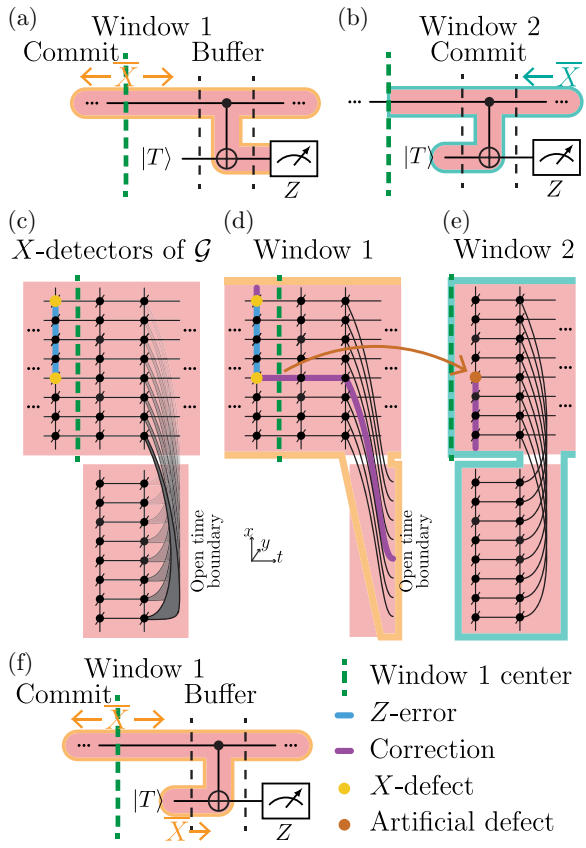


FIG. 20. (a) An example of a fragile time boundary in the forward direction, in which a logical X operator at the window boundary propagates forward to the logical Z measurement that it anticommutes with. Since the window boundary of the windowed-LOM decoder is too close (less than $d/2$ time-edges away) to the logical Z measurement, a low-weight error can subsequently cause a logical error in a single-LOM decoder in window 2, shown in (b). (c) An example of a weight-4 error in the $d = 9$ surface code that will cause a logical error. In general the weight of the error is $d/4 + O(1)$. (d) The correction that the single-LOM decoder shown in (a) makes involves matching to the fragile time boundary, causing an artificial defect in window 2. (e) The correction of this artificial defect in a single-LOM in window 2 that leads to a logical error. Note that in window 2 we backpropagate the logical X and therefore have no support on this fragile-time boundary. (f) A modification to the single-LOM in (a) avoids this issue by "peeling" the single-LOM off the fragile time boundary and onto the closed time boundary arising from the fault-tolerant state preparation of $|\overline{T}\rangle$.

Solving this for the windowed-LOM decoder is simple: we simply need to ensure that every committed time edge is $>d/2$ time-like edges away from any fragile boundary in the single-LOM. One way to guarantee this is by synchronizing all resets to occur only at the first time step after each window boundary, and all measurements at the last time step before each window boundary. That way, all resets and measurements will be $>d/2$ away from the center of every single-LOM that contains it. Of course, this is not necessary for the slow resets in the basic windowed-LOM decoder.

However, it is possible in some cases to be more flexible than this, in particular in the case of T-injection circuits. When a single-LOM involves propagating an $\overline{X}$ operator forward through the T-injection circuit, it can have support on the fragile time boundary of the Z-measurement, as shown in Fig. 20(a). This can lead to errors of weight $d/4 + O(1)$ causing a logical error, as shown in Figs. 20(b)–20(e). The solution to this is to include an $\overline{X}$ operator arising from the $|T\rangle$ state preparation in the observable that we are forward-propagating through the circuit, as shown in Fig. 20(f). This way, the single-LOM is "peeled" off the fragile time boundary, instead depending on the nonfragile time boundary resulting from the fault-tolerant preparation of the $|\overline{T}\rangle$ state. This peeling-off idea however does not work for arbitrary mid-circuit measurements, necessitating the synchronization of the remaining measurements and resets.

3. Time-like snakes

The final issue that we discuss is the circuit-dependent occurrence of time-like *snakes*. Intuitively, the issue arises because a low-weight spatial error in one window can be decoded multiple times by different single-LOMs, each of which leaves behind a pair of defects for the following window to decode—we call these *leftover* defects as they may contain both artificial defects, which are due to the committed correction, as well as real defects that have not yet been corrected by a committed correction.

If all these leftover defects appear in the decoding graph of the same single-LOM in the second window, then the minimum-weight correction is no longer guaranteed to be low-weight. It is therefore possible that an error of weight $<d/2$ causes a logical decoding error. Actually realizing such a bad example also relies upon the pairs of leftover defects being "far away" in the second window; that is, the shortest number of time-like edges required to travel from

one pair of leftover defects to another is long. For this to happen, the circuit needs to be such that there is a "snake" in the structure of the single-LOM. We discuss these issues in more detail in Appendix B, with two explicit examples. In Appendix B 1 we show a relatively small example in which the weight of the physical error scales as $2d/5 + 6$, although the smallest-distance example where the physical error is of weight strictly less than $d/2$ is only $d = 65$. Meanwhile, in Appendix B 2, we show an asymptotically worse example where the physical error weight scales sublinearly with d. Both of these examples motivate a solution to the problem of snakes.

Our solution to address the issue of snakes is to modify the decoding subgraph in each single-LOM by adding "short-cut" edges between vertices that represent different logical qubits (defined precisely in Appendix B 3). Intuitively, the short-cut edges remove the problem of snakes because there is always a short time-like edge that links pairs of leftover defects. Moreover, each short-cut edge always connects vertices with the same spatial coordinates, and therefore does not inadvertently create other low-weight logical errors. We conjecture that the short-cut edges, together with the above "synchronization" of resets and measurements, are enough to guarantee that the windowed-LOM decoder can correct all basic error patterns of weight $<d/2$. It is worth noting, however, that these short-cut edges may reduce performance in practice, particularly if the issue of time-like snakes does not appear much in real circuits. We leave theoretical and numerical investigations of thresholds to future work.

VI. DISCUSSION

In this work we have made substantial progress toward an efficient, fault-tolerant matching-based decoder for algorithms implemented in the rotated surface code with fast transversal Clifford gates, fast T-injections, fast resets and fast measurements. The LOM decoder constitutes a matching-based decoder that can correct all basic error patterns of weight $<d/2$, and numerically performs well under repeated- and random-Clifford gate circuits. The basic windowed-LOM decoder adds to this by being efficient, but comes at the cost of slowing down the resets and synchronizing the non-T-injection measurements in the circuit. Although the basic and/or two-step windowed-LOM decoders may perform well in practice, we still leave open the fundamental question: is it possible to design an efficient matching-based decoder that does not require these slow resets?

Apart from this general open question, there are also a number of other possible directions to further study (windowed) LOM decoders.

We have shown that the logical performance of the LOM decoder is improved by first running a belief-propagation algorithm on the full decoding hypergraph before executing the LOM decoder. BP updates the weights of all hyperedges taking into account the correlations as it uses the *full* decoding hypergraph, such as correlations from Y errors. However, there could be faster and more efficient methods than BP for providing correlation information to the LOM decoder.

It is natural to consider whether the LOM decoder applies when more than one $\overline{S}$ and/or $\overline{\text{CNOT}}$ gate is performed between QEC rounds. In this more general setting, it is possible for weight-4 hyperedges to appear in the decoding hypergraph $\mathcal{G}$ that can remain weight-4 in the decoding sub(hyper)graph G_O for an observable O, even for the basic error model, hence not reducing the problem to matching, as we discuss in Sec. III from Supplemental Material [37].

Although we have focused on the unrotated surface code, the LOM decoder can be also applied to rotated surface and color codes (with an appropriate matching-based color code decoder, see, e.g., Ref. [70]). An interesting question would be to determine the minimum number of circuit-level errors that cause a logical error for the LOM decoder applied to these codes, to see if the exact circuit distance is achieved or not. Recent work [20] has shown a more efficient implementation of the $\overline{S}$ gate for the rotated surface code: the logical gate is implemented inside the syndrome extraction circuit, which changes the structure of the decoding hypergraph. Further work is thus required to numerically decode this implementation with the LOM decoder.

Multiple future directions are necessary also to evaluate the practicality of the windowed-LOM decoder. It is likely that one can extend the windowed-LOM decoders to parallel windowed-LOM decoders to improve the speed of real-time decoding and to provide a solution to the backlog decoding problem [30,31,71]. Numerical simulations are also needed to study the existence of a threshold and to compare the performance of the windowed decoder variants. Moreover, we have considered T-state preparation as a black box in this work; investigating the interplay of the windowed-LOM decoder with an actual magic state distillation protocol is important future work.

On a fundamental level, the fault-tolerance issue of time-like snakes that we uncovered for the windowed-LOM decoder, and for which we proposed the short-cut edge solution, has a somewhat fundamental character which could also come up for other fast-logic decoding strategies and different codes. The issue arises because multiple independent observers (i.e., decoders) find and commit to causes (i.e., corrections) for events (i.e., defects) in overlapping bounded regions of space-time. The observers pass on information about these commitments to a following observer (i.e., leaving leftover defects), with the following observer seeing a subsequent space-time region. Since the commitments of the previous observers were independent, and due to the causal structure in the subsequent

space-time region, the next observer, given the previous commitments it has to accept, may deduce a cause that is less likely than the actual cause, leading to logical decoherence. More information thus needs to be passed from the independent multiple observers to the next observers about "what could have happened before" and our proposed short-cut edge solution is a blanket expression of this. Phrased in this way, it is clear that the issue is not about the nature of the defects or edges versus hyperedges, but of a more fundamental character, related to independent space-time bounded observers. However, notably, decoding strategies in which independent observers commit to causes in nonoverlapping regions of space-time do not suffer from this issue: for example, a windowed parallel hyperedge decoder where space-time is split up into nonoverlapping commit regions, that is followed by another parallel decoding step which completes the set of commitments to a full commitment [30]. Since our aim is to construct a general windowed efficient *matching* decoder for fast logic, we are led (or perhaps even forced) into using independent observers which commit to causes in overlapping space-time regions.

ACKNOWLEDGMENTS

This work is supported by QuTech NWO funding 2020-2028—Part I "Fundamental Research," Project No. 601.QT.001-1, financed by the Dutch Research Council (NWO) and the OpenSuperQPlus100 Project (No. 101113946) of the EU Flagship on Quantum Technology (HORIZON-CL4-2022-QUANTUM-01-SGA). We thank Boris Varbanov and Zherui Wang for initial discussions on decoding fold-transversal gates for the surface code. We thank Timo Hillmann, Alex Kubica, Mark Turner, and the authors of Ref. [33] for interesting discussions. We acknowledge the use of the DelftBlue supercomputer [72] for decoding the experiments.

M.S.P. and M.H.S. contributed equally to this work, with M.S.P. focusing on numerics and M.H.S. focusing on theory, and B.M.T. provided motivation, writing, and supervision.

DATA AVAILABILITY

The data that support the findings of this article are openly available [44,45,73].

APPENDIX A: HIERARCHICAL MATCHING DECODING WITH FRAGILE TIME BOUNDARIES

The hierarchical matching decoder defined in Sec. III C can run into issues when decoding around fragile time boundaries that are *completely separate* to the issues discussed with the LOM-decoder and fragile observables in Sec. III B 2. The issue in the hierarchical matching decoder occurs when the fragile time boundary is *not* decoded in the last track T_L. When this is the case, one can construct errors of weight $d/4 + O(1)$ that cause a logical error on an adjacent logical qubit. This is shown in Fig. 21 for a weight-4 error in the $d = 9$ surface code. Intuitively, this happens because a single defect is created close to the fragile time boundary in the full hypergraph $\mathcal{G}$ but is decoded only in a later track that cannot be matched to the fragile time boundary, causing a high-weight spatial correction

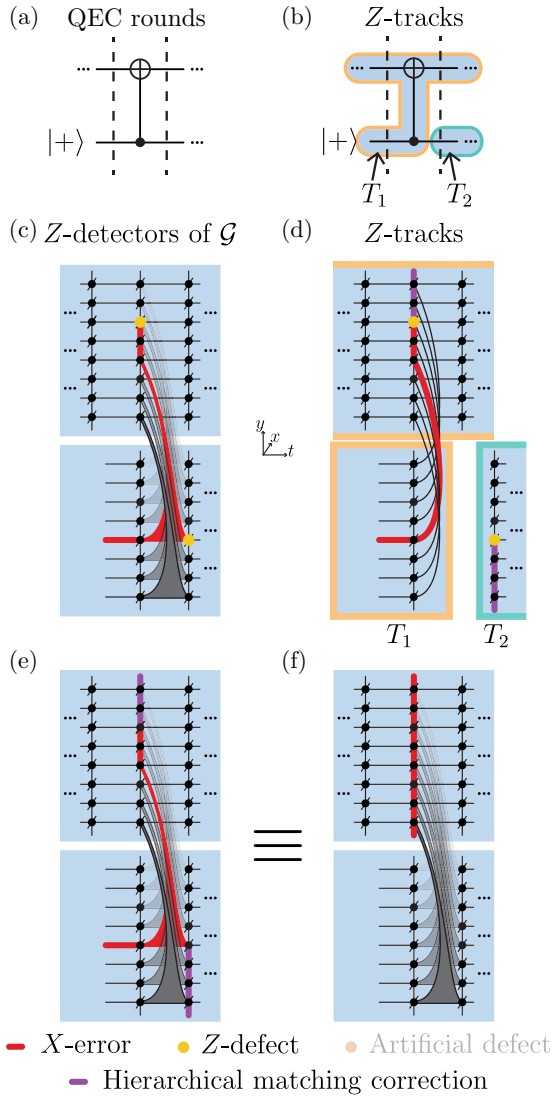


FIG. 21. An example of a failure of hierarchical matching due to the presence of a fragile time boundary in the first track. (a)–(e) The circuit, tracks, error, action of the hierarchical matching decoder, and the final correction for the error, respectively, similar as in Fig. 8. The error + correction in (e) can be multiplied by space-time stabilizers to the error string in (f), which corresponds to a logical X-error on the top qubit in (a). The physical error here has weight $4 < 9/2 = d/2$. Note also that although we have suggested that the circuit extends in both time directions in the figure, this example still works if, say, the first qubit is initialized in $|0\rangle$.

instead. This problem has a simple solution: simply ensure that all fragile time boundaries are contained in the last track T_L. Indeed, when the fragile time boundary in Fig. 21 is included in track T_2 instead of T_1, the hierarchical matching decoder indeed corrects the error Fig. 21(c) without a logical error.

APPENDIX B: TIME-LIKE SNAKES IN THE WINDOWED-LOM DECODER

In this appendix we go into detail about the issue of time-like snakes, short-cut edges, and the conjecture that the windowed-LOM decoder can correct up to $d/2$ basic errors. The discussion applies both to the one-step and two-step windowed-LOM decoders. We begin in Appendix B 1 by showing a relatively simple example of how a time-like "snake" can allow a low-weight error [still of weight $\Theta(d)$] to lead to a logical decoding error. Then, in Appendix B 2 we show a more elaborate example where the weight of the error grows sublinearly in d, further motivating a need to solve this issue. We do so in Appendix B 3, where we describe more precisely the "short-cut edges" that we use to break-up the snake. The short-cut edges only connect vertices with the same spatial coordinates. Intuitively, these work by ensuring that any pair of defects within the same single-LOM can be matched by a string of edges of weight depending only on the space-time coordinates of the defects, and not on the rest of the logical structure of the circuit, as shown in Lemma B2. The short-cut edges thus change the space-time metric, making it more logically trivial like in a memory experiment.

Of course, this discussion does not immediately imply the existence of a threshold. Indeed, the windowed-LOM decoder without short-cut edges may still have a threshold even if it is not able to correct some errors scaling sublinearly in d; and the windowed-LOM decoder *with* short-cut edges does not necessarily have a threshold even though it is able to correct all basic errors with weight $<d/2$. It is therefore an open question whether a threshold exists for the windowed-LOM decoder in both variants, and which variant performs better in practical scenarios.

1. Simple example of a time-like snake

We begin by showing our relatively simple example of how a time-like "snake" can allow a low-weight error to lead to a logical decoding error. In particular, in Figs. 22–25 we show how an error of weight $2d/5+6$ can lead to a logical error in the windowed-LOM decoder. In this example there are multiple minimum-weight corrections that each single-LOM decoder can make, one such choice in each single-LOM leads to the logical decoding error [74]. For simplicity, we moreover only consider X errors that are spread using CNOT gates, which requires seven logical qubits in this example; although nothing precludes the possibility that similar structures could be constructed with fewer logical qubits by making use of both X- and Z-errors spread using S gates.

In this example we consider two windows of decoding, where the artificial defects left behind from the first window of decoding cause a logical Pauli flip in one of the single-LOMs in the second window. There are three features of the circuit and error pattern that make this possible:

(1) The "proliferation" of defects: multiple single-LOMs in the first window *see* the same set of Δy spatial errors, i.e., meaning the defects of these errors

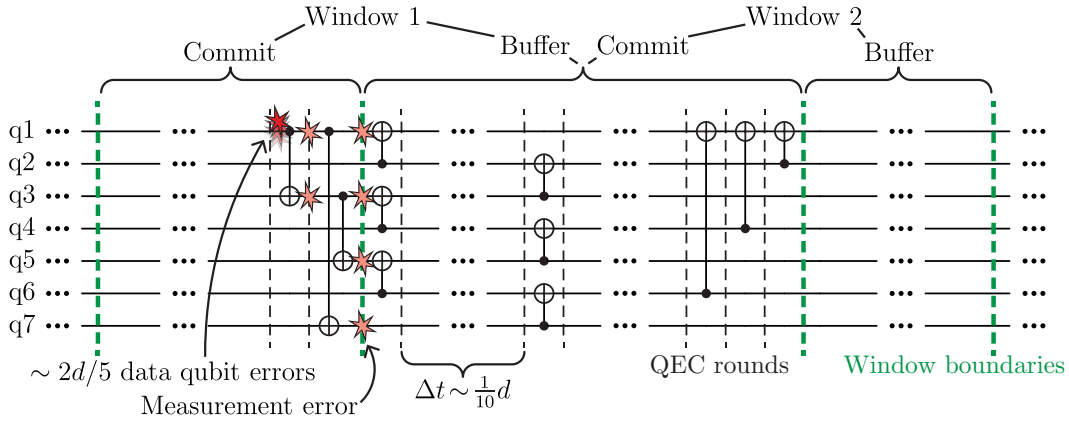


FIG. 22. A circuit $\overline{C}$ with seven logical qubits, each encoded in the unrotated surface code of distance d. An error of weight $\sim 2d/5$ can cause a logical error when decoded by the windowed-LOM decoder, in the absence of short-cut edges. The error occurs at seven locations in the circuit: the first set of errors consists of $\sim 2d/5$ X-errors on data qubits immediately before the first CNOT gate, while the remaining errors are single measurement errors on Z-stabilizer measurements during a QEC round. No other physical errors occur. The precise configuration of the X errors is shown in Fig. 23. The error will be decoded in two windows (1 and 2) of width $>d$, whose commit and buffer regions are labeled above the circuit. We will show in Figs. 23–25 that a logical error will be inferred for the $\overline{Z}_1$ operator at the end of the circuit.

occur in the decoding subgraphs of these single-LOMs, each leaving behind a pair of artificial defects. This means that in the second window, a correction of weight $>\Delta y$ is required to match the observed defects.

(2) A time-like "snake," i.e., the logical structure of the circuit in the second window implies that the proliferation of defects is seen by a next single-LOM decoder. This decoder has the opportunity to match these defects in novel ways, in particular partially using time-like edges (measurement errors). Specifically, we require that any time-like correction that is matching between different pairs of artificial defects (left by different previous decoders) are also long (but not *too* long, as discussed below).

(3) A "descending staircase" pattern of defects in the second window, i.e., some of the artificial defects are left behind at different spatial coordinates, causing a pattern of defects where the bad correction is always as short as the shortest good correction.

We show the structure of the circuit in Fig. 22 and the placement of the errors in the decoding hypergraph in Fig. 23. Figure 24 shows how the proliferation of defects occurs through each of the single-LOMs in the first window. Note also that each single-LOM applies a different correction that will create the descending staircase pattern of defects in the next window. Figure 25 then shows the snake in the second window. The key feature of the snake is that the shortest time-like path between pairs of leftover defects in the single-LOM shown in Figs. 25(a) and 25(b) is $\Theta(d)$. To see why this is necessary, we must consider the possible good and bad corrections that the single-LOM may make as shown in Fig. 25(c). If the snake is too short, then $\Delta\tilde{t} = 6\Delta t + 9$ will be small, meaning that good corrections 2, 3, and 4 will be shorter than the bad correction. Meanwhile if the snake is too *long* and $\Delta\tilde{t}$ too large, then good correction 1 will be shorter than the bad correction. Therefore only when $\Delta\tilde{t}$ is intermediate in size—roughly $3d/5$—can the bad correction be shorter in length than all the good corrections.

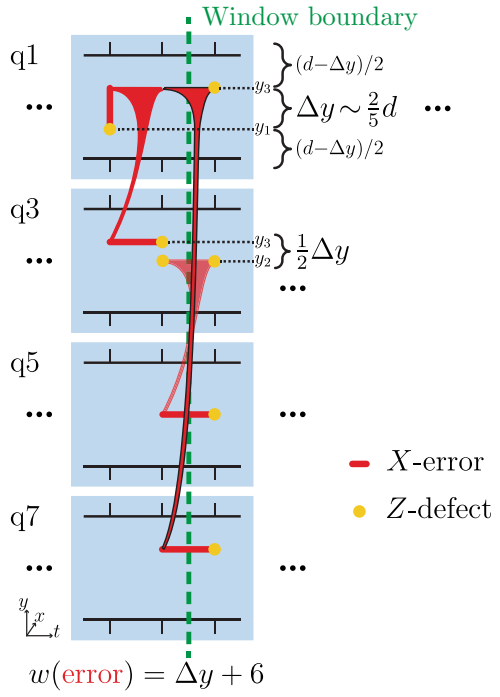


FIG. 23. A simplified representation of the decoding hypergraph $\mathcal{G}$ showing the X-error pattern of weight $2d/5 + 6$ in the circuit in Fig. 22 that leads to a logical error when decoded by the windowed-LOM decoder. Only the slice of the circuit in Fig. 22 where the physical errors occur is shown in the figure. For ease of visualization the space and time coordinates are not to scale; each interval in time represents just one time-like edge.

It is an interesting question to determine the smallest error of weight $<d/2$ that can cause a logical decoding error. For the example shown in Figs. 22–25, this is $d = 65$, which works with the parameters $\Delta y = 26$ and $\Delta t = 5$ such that the error has weight $\Delta y + 6 = 32 < 65/2$ and all the corrections in Fig. 25(c) have weight 78. It is therefore unlikely that this particular example will cause practical problems in the surface code, and it may even be the case that below some minimum distance it is impossible for there to be a snake that causes such a problem. However, we have no proof that more elaborate examples do not exist for smaller distances, and it is therefore desirable to find a solution to this problem which can be tested on practical examples later.

2. An asymptotically worse example

To construct an example where the weight of the error which causes a logical error scales slower than $\Theta(d)$, we use the more elaborate circuit and set of errors shown in Figs. 26–28, which represent a generalization and modification of the smaller circuit analyzed above, in Appendix B 1. The three features of the example in Appendix B 1—the proliferation of defects, the time-like snake, and the descending staircase of defects—are still present. Just like in Appendix B 1, there are two windows in the circuit that are relevant, where the artificial defects from the first window of decoding cause a logical error in the second window. The weight of the physical error depends on the function $f(t)$ that determines the maximum spreading of a Pauli operator in a circuit of depth t as discussed in Sec. V. In the unrealistic worst-case where there are no spatial restrictions on the gates that can be implemented with $f(t) = 2^t$, we find that the weight of the physical error which creates a logical error can scale as slowly as $\Theta\left(\log_2(d)\right)$; however, in realistic practical scenarios where $f(t) = \Theta\left(t^D\right)$ for some integer D, the weight of the physical error instead can scale as $\Theta\left(d^{1/(D+1)}\right)$.

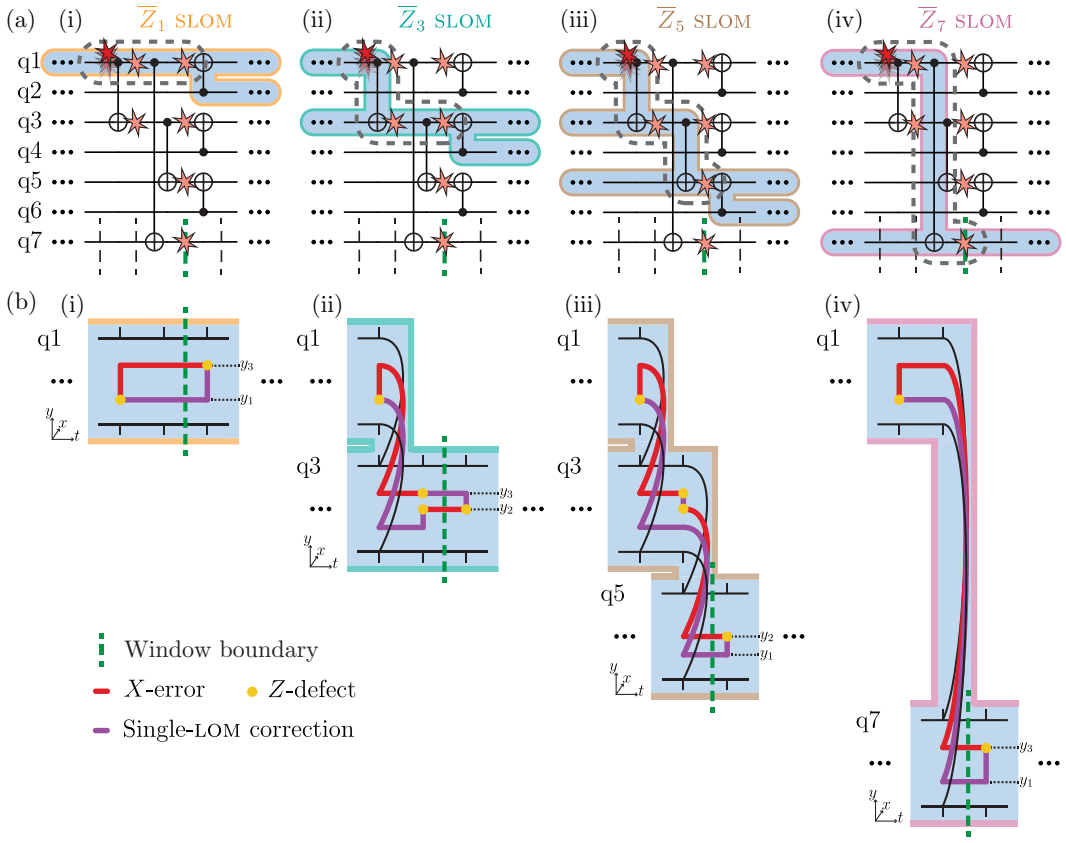


FIG. 24. The first window of decoding the error from Figs. 22 and 23. We are only interested in four of the iterations of the single-LOM (abbreviated SLOM in the figure), corresponding to the $\overline{Z}$ operators on logical qubits (i) 1, (ii) 3, (iii) 5, and (iv) 7 in the center of the window. (a) The propagation of each single-LOM through the circuit. (b) The decoding subgraph of each single-LOM, focusing only on the regions containing physical errors, as shown by the gray dotted regions in (a). In order to create a logical error in the second window, the correction in each single-LOM decoder must follow the pattern shown in purple. Each correction is one of multiple minimum-weight corrections in each single-LOM.

The circuit itself is shown in Fig. 26 and involves nine *blocks* of logical qubits, each of which contains M or $M+1$ logical qubits. Here M is a parameter that will scale as some function of d that we will determine later. The location of the spatial errors and the corrections made by the single-LOMs in the first window are shown in Fig. 27, and the correction that causes a logical error in the second window is shown in Fig. 28.

To understand how this works, we explain how this circuit enables the proliferation of defects, a time-like snake, and a descending staircase of artificial defects to cause a logical error. The proliferation of defects occurs in the first window of decoding, where the spatial errors on the first qubit in, for example, blocks 1 and 7 are seen by all $M+1$ single-LOMs derived from $\overline{Z}$-operators in block 1. Key to this is the formal circuit $U^{(X)}_{1\to N}$, which we define for any integer N as any Clifford circuit such that

$$U^{(X)}_{1\to N}\overline{X}_1 U^{(X)\dagger}_{1\to N} = \overline{X}^{\otimes N}. \tag{B1}$$

We likewise define $U^{(X)}_{N\to 1} = U^{(X)\dagger}_{1\to N}$. Importantly, any $\overline{Z}_n$ operator with $n = 1,\ldots,N$ anticommutes with $X^{\otimes N}$, and therefore $U^{(X)}_{1\to N}\overline{Z}_n U^{(X)\dagger}_{1\to N}$ anticommutes with $\overline{X}_1$. In block 1 with $N = M+1$, this means that every single-LOM derived from logical observable $\overline{Z}_n$ in block 1 at the center of the first window includes $\overline{Z}_1$ prior to the $U^{(X)}_{1\to M+1}$ gate, and thus *sees* the defects produced by the spatial errors on the first logical qubit in block 1. The same argument applied forward means that every such single-LOM also contains the Z-detectors for $\overline{Z}_1$ at the end of block 7, as shown in Fig. 27(a). The depth of the circuit $U^{(X)}_{1\to N}$ depends on $f(t)$ and, as we will see later, will govern the minimum weight of the physical error that is possible in this example.

Each of the $M+1$ single-LOMs in block 1 thus sees the same set of spatial errors, which is shown in Fig. 27(b). Of course, each single-LOM also has support on other parts of the circuit, but since no errors occur in those locations they are not relevant for us. Of particular importance is the number of QEC rounds between the two sets of spatial errors Δt_{err}, shown in Fig. 27(a), which depends

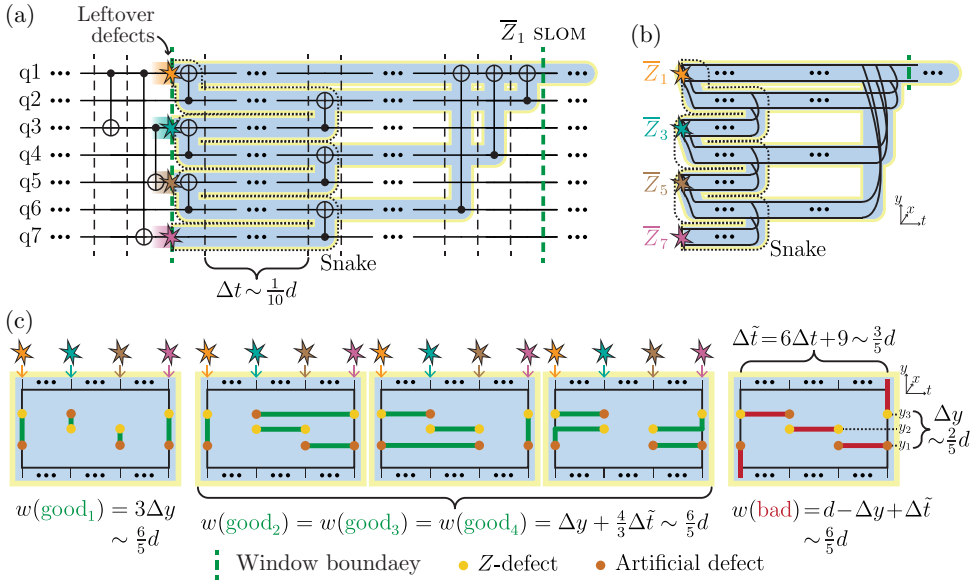


FIG. 25. The second window of decoding the errors from Figs. 22 and 23 along with the leftover defects from the first window of decoding in Fig. 24. We are only interested in the single-LOM corresponding to $\overline{Z}_1$ in the center of the window. (a) The propagation of the $\overline{Z}_1$ single-LOM through the circuit, with the locations of the leftover defects from the four single-LOMs from the first window shown. (b) The decoding subgraph of the single-LOM. Focusing only on the "snake" part of the subgraph and flattening it out gives the representation in (c), which shows five possible corrections to the leftover defects in the single-LOM. The temporal length $\Delta\tilde{t}$ is the length of the snake; note that it includes a constant offset (nine in this example) due to the turns of the snake. The first four corrections shown are good because they correspond to corrections in the same logical class as the physical error that occurred, while the last one is bad because it causes a logical error equivalent to an $\overline{X}_1$ error in the middle of the second window. All five corrections have weight $\sim 6d/5$, despite being caused by a physical error of weight only $\sim 2d/5$.

on the operator-spreading structure of the circuit $U^{(X)}_{1\to M+1}$ and hence on $f(t)$. By inspection of Fig. 2, the number of time edges separating the two spatial errors in any given single-LOM may be larger than the number of QEC rounds separating them, but is always bounded by $\Delta t_{\text{err}} \le \Delta t_{\text{SLOM}} \le 2\Delta t_{\text{err}}$. The error shown in Fig. 27(b) has weight $2\Delta y$, while the correction has weight $2\Delta t_{\text{SLOM}}$. In order for the indicated correction (in purple) to be chosen in each single-LOM, we therefore require $2\Delta y \ge 2\Delta t_{\text{SLOM}}$ for each single-LOM in block 1. Alternatively, a simpler sufficient condition is $\Delta y \ge 2\Delta t_{\text{err}}$. The same construction is repeated for the single-LOMs in blocks 3 and 5 as shown in Figs. 27(c)–27(d), albeit with the errors happening at different spatial coordinates. In total, the construction uses an error of weight $4\Delta y$, and a sufficient condition for the purple correction to be chosen is

$$\Delta y \ge 4\Delta t_{\text{err}}, \tag{B2}$$

where the constant factor of 4 instead of 2 arises because the height of the loops in Figs. 27(c)–27(d) is only $\Delta y/2$.

The next ingredient that is required is the presence of a time-like snake in a single-LOM in the following window, which we show in Fig. 28(a). The most fool-proof way of constructing such a snake is to consider a single-LOM that is in the second step of the two-step windowed-LOM decoder. The single-LOM is propagated backward from a multilogical-qubit Pauli operator at the center of window 2 given by a product of $\overline{Z}$ operators on the first qubit of the first block and all qubits in blocks 2, 4, and 6, as shown in Fig. 28(a). Such a second-step single-LOM can arise when dealing with the fragile observables arising from fast resets and/or measurements in the commit region of window 2 (not shown in explicitly Figs. 26–28). We suspect that examples could also be constructed in the basic windowed-LOM decoder where all single-logical observables are reliable, by including a unitary gate $H^{\otimes(3M+1)}U^{(X)}_{3M+1\to 1}H^{\otimes(3M+1)}$ on the first qubit of block 1 and all qubits in blocks 2, 4, and 6 immediately before the center of window 2. In this case, however, it is more complicated to prove Lemma B1, even though we suspect the lemma would still be true.

The snake, shown in Fig. 28(a), wriggles its way through the different blocks of the circuit, from the first qubit in blocks 1, 2, 3, 4, 5, and then 6, and then back to the second qubit in blocks 1, 2, 3, and so on, until it terminates on the last qubit of block 1. Moreover, the single-LOM sees a pair of leftover defects for each of the single-LOMs in Fig. 27, which are equally spaced throughout the snake region. Importantly, the shortest time-like matching between defects in leftover defects is through the snake and therefore scales with $\Delta t'$ and M.

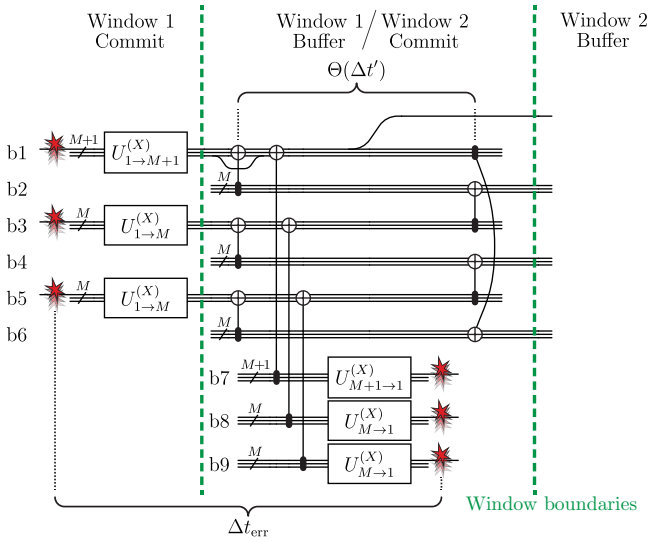


FIG. 26. A circuit $\overline{\mathcal{C}}$ with $9M+2$ logical qubits in which an error which scales less than linearly in d can cause a logical error when decoded by the windowed-LOM decoder. We represent the logical qubits in nine blocks of M or $M+1$ qubits labeled b1–b9. CNOT gates between the blocks represent independent logical CNOTs between the mth qubit in the first and the mth qubit in the second block. The U-gates are defined in Eq. (B1). Note that we also split off individual logical qubits from block 1 in some of the CNOTs, either from the top of the block or from the bottom. X-data-qubit-errors occur in six circuit locations, the coordinates of these errors are shown in detail in Fig. 27.

Finally, the descending staircase of defects can be seen in the flattened representation of the single-LOM shown in Fig. 28(b), with two example good corrections and one example bad correction to this pattern of defects. We choose $\Delta t' = \Delta y/2$ as indicated in Fig. 28(b). This way, the two good corrections shown in Fig. 28(b) have the same weight. In fact, the structure of the descending staircase allows us to prove that at least one of three corrections shown in Fig. 28(b) is a minimum-weight correction to the defects.

Lemma B1. For the descending staircase of defects in Fig. 28(b) with $\Delta t' = \frac{1}{2}\Delta y$ and $\Delta y < d/2$, every correction to the defects by the multi-Pauli single-LOM at the center of window 2 in Fig. 28(a) has weight w lower-bounded as

$$w \geq \min\left((4M+2)\Delta y/2, d + (3M-2)\Delta y/2\right). \quad \text{(B3)}$$

Proof. We will structure this proof by considering the number of times n_{bdy} the correction matches to the boundary (either the top or bottom spatial boundary). All other edges in the matching must be between the remaining $6M+2-n_{\text{bdy}}$ nonboundary vertices, which is only possible if n_{bdy} is even. The nonboundary matching graph of these nonboundary vertices $G_{\text{match}} = (V_{\text{match}}, E_{\text{match}})$, which contains only the vertices that correspond to (artificial) defects, is shown in Fig. 28(c). Note that the number of vertices in G_{match} is $|V_{\text{match}}| = 6M+2$.

We begin by considering the case where $n_{\text{bdy}} \neq 0$. Since the closest defect to the boundary is $(d-\Delta y)/2$, the weight of each edge in the matching connected to the boundary is at least $(d-\Delta y)/2$. Moreover, each edge in G_{match} (i.e., every edge that is *not* connected to the boundary) has weight at least $\frac{1}{2}\Delta y$. Therefore, by counting the

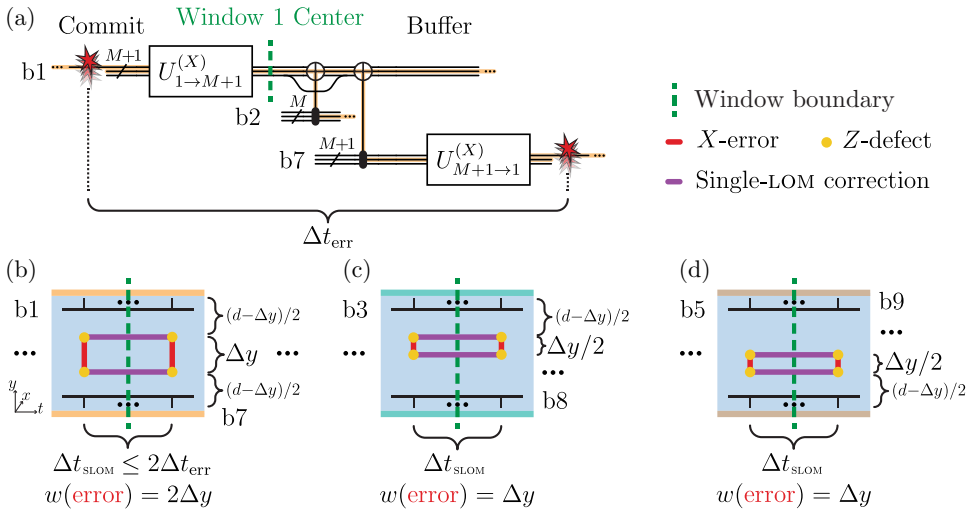


FIG. 27. The first window of decoding the error from Fig. 26. Each $\overline{Z}_i$ single-LOM with i representing a qubit in blocks 1, 3, or 5 are relevant for our example. (a) The propagation of an arbitrary block-1-single-LOM for $\overline{Z}_n$, $n = 1, \ldots, M+1$ through the circuit. Similar structures are followed for the other single-LOMs in this window. (b)–(d) The decoding subgraphs of single-LOMs in blocks 1, 3, and 5 (respectively), along with the errors and corrections in each single-LOM. The artificial defects from each single-LOM appear in the second window and cause a logical error in decoding, as shown in Fig. 28.

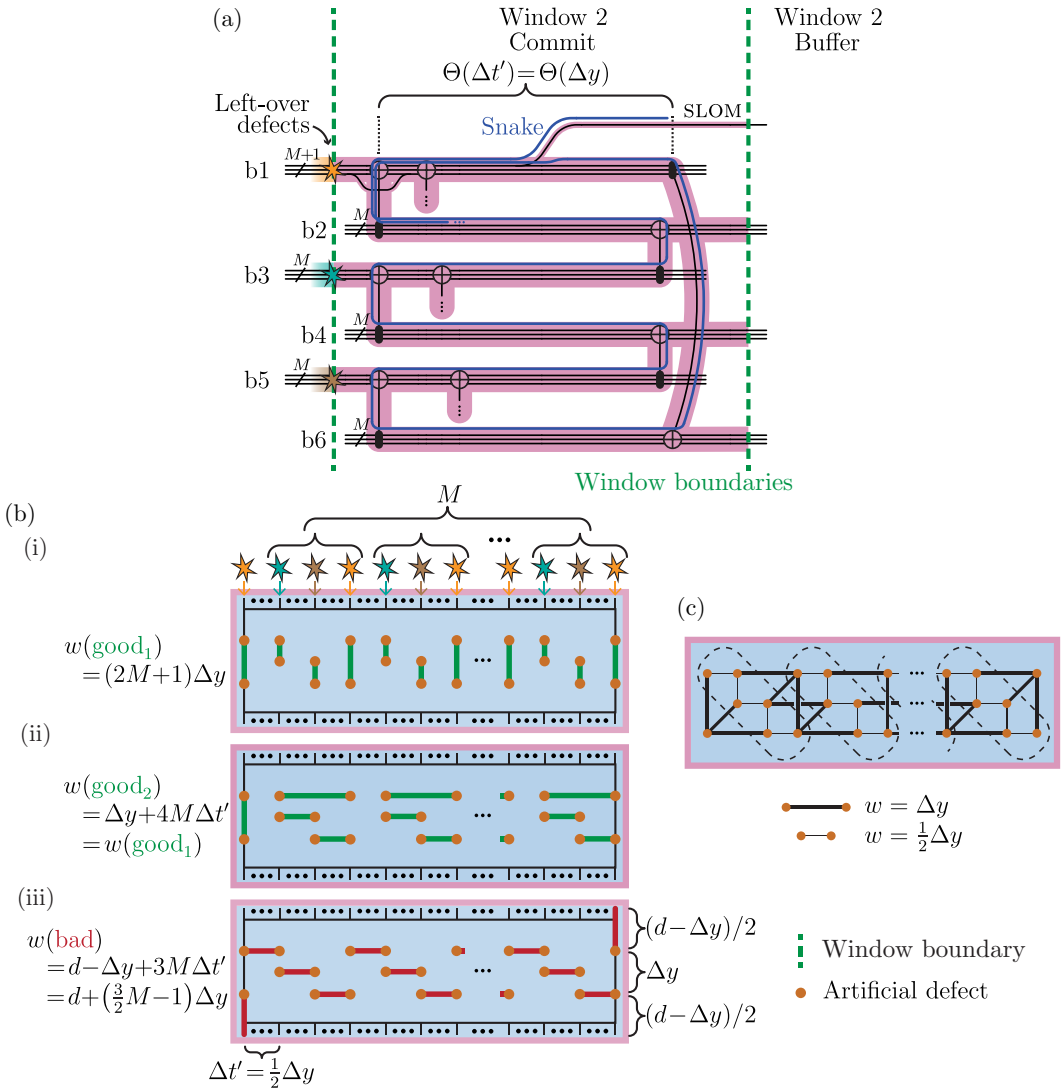


FIG. 28. The second window of decoding the leftover defects from the errors and corrections in Figs. 26 and 27. We are only interested in the single-LOM corresponding to $\overline{Z}_1$, shown shaded in (a). We also show in dark blue the structure of the snake, which wriggles from the first qubit in block b1, then to the first qubit in each of the blocks in turn down to b6, before returning to the second qubit in block b1, and so on. On blocks b1, b3, and b5 artificial defects are created at different spatial locations according to Figs. 27(b)–27(d). (b) Some good and bad corrections. (c) The matching graph of the artificial defects, with edges to the boundaries removed, used in the arguments in Lemma B1. The weight of each edge is indicated by the thickness of the line and follows the Manhattan distance. The dashed ovals indicate each "descending staircase" of defects, as used in Lemma B1.

number of edges that are and are not connected to the boundary we have

$$w \geq n_{\text{bdy}} d/2 + \left(3M + 1 - \frac{3}{2} n_{\text{bdy}}\right) \Delta y/2. \tag{B4}$$

If $n_{\text{bdy}} \geq 2$, we can use $\Delta y < d/2$ to write

$$w \geq (n_{\text{bdy}} - 2)d/2 + \left(3M + 1 - \frac{3}{2}(n_{\text{bdy}} - 2)\right) \Delta y/2 + \left(d - \frac{3}{2}\Delta y\right)$$

$$\geq (n_{\text{bdy}} - 2)d/2 + \left(3M + 1 - \frac{3}{2}(n_{\text{bdy}} - 2)\right) \Delta y/2, \tag{B5a}$$

and therefore we can substitute $n_{\text{bdy}} = 2$ into Eq. (B4), giving

$$w \geq d + (3M - 2)\Delta y/2, \quad \text{if } n_{\text{bdy}} \geq 2. \tag{B5b}$$

Now, if $n_{\text{bdy}} = 0$, then the entire correction must use only the edges shown in Fig. 28(c). Consider each "descending staircase" of vertices $V_{\text{stair},m} \subset V_{\text{match}}$ for $m = 1, \ldots, M$, shown in the dotted regions. Edges of weight

$\Delta y/2$ connect pairs of vertices in the same staircase $V_{\text{stair},m}$; to match vertices in different staircases requires a "thick" edge of weight Δy. For each $V_{\text{stair},m}$, there are thick edges to the left, connecting it with $V_{\text{stair},m-1}$, and thick edges to the right connecting it with $V_{\text{stair},m+1}$. Since there are an even number of vertices in each staircase, the number of thick edges to the left and right that are included in the matching must be either both even, or both odd. Moreover, looking at the first descending staircase $V_{\text{stair},1}$, in order for the matching to match the bottom-left vertex there must be an odd number of thick edges to the left that are included in the matching. This means that there must be an odd number of thick edges between every pair of staircases in G_{match}. Therefore, there are at least $M+1$ thick edges in the matching. Since from before there are at least of $3M+1$ edges in the matching, we have

$$w \geq (M+1)\Delta y + 2M\Delta y/2 = (2M+1)\Delta y, \quad \text{if } n_{\text{bdy}} = 0. \tag{B6}$$

Combining Eqs. (B6) and (B5b) proves the result. ■

With these three ingredients in place, we can now determine the minimum asymptotic scaling of the weight of the physical error that causes the logical error shown in Figs. 26–28. From Lemma B1, the bad correction in Fig. 28(b)(iii) will be a minimum-weight correction to the error syndrome if

$$d \leq (M+4)\Delta y/2, \tag{B7}$$

or, in asymptotic terms, we require $M\Delta y = \Omega(d)$. Moreover, from the corrections in the first window, we require

$$\Delta y \geq 4\Delta t_{\text{err}}. \tag{B2a}$$

However, Δt_{err} is lower-bounded by the depth of the $U^{(X)}_{1\to N}$ circuits. In particular, in Sec. V, we defined the function $f(t)$ as the maximum number of qubits that a single-qubit Pauli operator can be propagated to in a circuit of depth t. In general, $f(t) = 2^t$; however, in a physical setting, locality and efficiency requirements will limit this to $f(t) = \text{poly}(t)$ for some polynomial in t. Therefore, the depth of $U^{(X)}_{1\to N}$ is lower-bounded by $f^{-1}(N)$. Therefore there exist circuits $U^{(X)}_{1\to M}$ such that Δt_{err} scales as $O\left(f^{-1}(M)\right)$.

First, consider the worst-case unphysical situation where $f(t) = 2^t$. In this case we can set

$$\begin{aligned} M &= \Theta(d), \\ \Delta y &= \Theta(\Delta t_{\text{err}}) = \Theta\left(f^{-1}(M)\right) = \Theta\left(\log_2(d)\right), \end{aligned} \tag{B8}$$

such that the weight of the error is $\Theta\left(\log_2(d)\right)$ while the scaling still satisfies $M\Delta y = \Omega(d)$. In the more physical case where $f(t) = \Theta(t^D)$ for some dimension D, we can likewise set

$$\begin{aligned} M &= \Theta(d^{D/(D+1)}), \\ \Delta y &= \Theta(\Delta t_{\text{err}}) = \Theta\left(f^{-1}(M)\right) = \Theta\left(d^{1/(D+1)}\right). \end{aligned} \tag{B9}$$

In both cases, the weight of the error Δy which can cause a logical error as in Fig. 28, scales sublinearly with d.

3. Short-cut edges

We provide a precise definition of the short-cut edges in either the basic or two-step windowed-LOM decoders for which we introduce the following notation. We write the full decoding hypergraph as $\mathcal{G} = (\mathcal{V}, \mathcal{H})$, and the decoding subgraph for a single-LOM as $G = (V, E)$. Moreover, every basic error $h \in \mathcal{H}$ that overlaps with the single-LOM can be projected down to an edge $e \in E$. A subset of edges $s \subseteq E$ has a *vertex boundary* $\partial s \subseteq V$ that includes all vertices that are the endpoints of an odd number of edges in s. We write $w(s) = |s|$ for the weight (i.e., size) of the edge subset s. If a nonempty subset of edges has no more than two endpoints we call it an edge *string*.

Recall that each detector $d(t, j, x, y)$ is labeled with a set of coordinates as in Table I from Supplemental Material [37]. Here, $t \in \mathbb{Z}$ represents the time of the detector, j represents the logical qubit and x and y are spatial coordinates. X-detectors have spatial coordinates satisfying $x \in \mathbb{Z}$ and $y \in \mathbb{Z} + \frac{1}{2}$ while for Z-detectors we have $x \in \mathbb{Z} + \frac{1}{2}$ and $y \in \mathbb{Z}$, with $0 \leq x, y \leq d-1$. In the statements below we will find it useful to have a different labeling of vertices $v(t, k, z, \tilde{z})$, given by $v(t, 2j-1, x, y+1/2) \equiv d(t, j, x, y)$ for X-detectors and $v(t, 2j, y, x+1/2) \equiv d(t, j, x, y)$ for Z-detectors. This way, $z, \tilde{z} \in \{0, 1, \ldots, d-1\}$ regardless of whether the vertex represents an X- or Z-detector.

Now, consider any single-LOM instance in the execution of the windowed-LOM decoder. In particular, the single-LOM decoders runs MWPM on a decoding subgraph $G = (V, E)$. We can moreover assume without loss of generality that G is connected; after all, if G is *not* connected, then we can decode it in two independent single-LOMs. For convenience, in each single-LOM we introduce a set of *boundary* vertices V_{bdy}. In particular, for each edge in the single-LOM that is connected to the boundary vertex v_{bdy} we define a separate boundary vertex that has its own unique set of $(t, k, z, \tilde{z})$ coordinates. We define the set of boundary vertices as V_{bdy}, which itself is partitioned into three subsets: V_{top}, V_{bottom}, and V_{time} representing the top spatial boundary ($\tilde{z} = d$), bottom spatial boundary ($\tilde{z} = 0$), and open time boundaries, respectively. To obtain a valid correction, the correction subset of hyperedges $c \subseteq \mathcal{H}$ must have vertex boundary ∂c that matches the observed defects everywhere except on the boundary vertices V_{bdy}. For the purposes of running MWPM one needs to connect all these boundary vertices to each other by weight-zero

edges; however, for the purposes of our proof, we instead equivalently assume that there are no edges between these vertices. In this picture, the boundary vertices are the only vertices of degree one in the decoding graph.

With this nomenclature, all of the standard edges in the decoding subgraph, assuming basic errors, fall in one of the four categories:

(1) z-type edges between a pair of vertices $\{v(t,k,z,\tilde{z}), v(t,k,z\pm 1,\tilde{z})\}$;
(2) $\tilde{z}$-type edges between $\{v(t,k,z,\tilde{z}), v(t,k,z,\tilde{z}\pm 1)\}$;
(3) t-type edges between $\{v(t,k,z,\tilde{z}), v(t+1,k',z,\tilde{z})\}$ for some pair of indices k,k' that may or may not be equal; or
(4) k-type edges between $\{v(t,k,z,\tilde{z}), v(t,k',z,\tilde{z})\}$ for $k\neq k'$.

Note that by definition, z-type, $\tilde{z}$-type, and k-type edges connect vertices whose coordinates differ only in a single coordinate (the z, $\tilde{z}$, and k coordinates, respectively), while t-type edges contain vertices that can differ in both their t- and k-coordinate values. We will also occasionally refer to the z- and $\tilde{z}$-type edges collectively as *space-like* edges, and the t- and k-type edges as *time-like* edges. The observing edge set in any given single-LOM contains only $\tilde{z}$-type edges between a pair of vertices $\{v(t,k,z,d-1), v(t,k,z,d)\}$, where the latter vertex is on the top boundary $v(t,k,z,d)\in V_{\text{top}}$ and the time coordinate t is in the commit region of the single-LOM.

With this, we define the short-cut edges that we add to each single-LOM during the execution of the circuit as follows:

Definition B1 (Short-cut edges).
(1) *(k-type short-cut edges:)* for each $t,z,\tilde{z}$, add edges $\{v(t,k,z,\tilde{z}), v(t,k',z,\tilde{z})\}$ for all k,k' that are in the single-LOM at time t and such that neither vertex is a boundary vertex (i.e., $v(t,k,z,\tilde{z}), v(t,k',z,\tilde{z})\notin V_{\text{bdy}}$), and
(2) *(t-type short-cut edges:)* for each $t,z,\tilde{z}$, add edges $\{v(t,k,z,\tilde{z}), v(t+1,k',z,\tilde{z})\}$ for every k that is in the single-LOM at time t and for every k' that is in the single-LOM at time $t+1$ and such that neither vertex is a boundary vertex.

In words, this means that for every single-LOM, there is a (real or short-cut) k-type edge between *every* pair of vertices in the single-LOM that has the same time and space coordinates, even if they differ by logical qubit and/or the Pauli type of the detector; *and* there is a (real or short-cut) t-type edge between every pair of vertices that has the same space coordinates and time coordinates that differ by one.

The short-cut edges are important when running MWPM on the single-LOM, since without them low-weight errors can cause logical decoding errors as discussed in Appendix B. However, the short-cut edges do not directly change what is committed to in an individual single-LOM. For one, since none of the short-cut edges have any spatial component, they are not in the observing edge set and therefore cannot directly flip the inference of a logical outcome. And second, they also do not directly lead to any artificial defects. To understand this second point, note that any pair $\{v_1,v_2\}$ of vertices that are the endpoints of a short-cut edge can always be matched using regular t-type and/or k-type edges. Whenever the correction uses a short-cut edge, we can therefore interpret the short-cut edge instead as a matching with regular edges. However, this matching is designed to be of lower weight than when regular edges are used, hence they represent short-cuts. Moreover, due to the structure of any first-step single-LOM, this matching can never traverse the center of the single-LOM window, so it therefore does not directly change the artificial defects in the following window.

The short-cut edges are necessary for the following.

Conjecture B1. Assuming a basic error model, a basic or two-step windowed-LOM decoder defined in Sec. V and adapted to include the short-cut edges, applied to circuits defined in Sec. II B with synchronized resets and measurements and implemented in the unrotated surface code, can correct any basic error of weight $<d/2$.

Intuitively, the reason we believe this works is because short-cut edges remove the possibility of snakes occurring in any given single-LOM. In particular, recall that in Appendices B 1 and B 2, we explained that for the counterexamples to work, the snake had to have a particular length in order for the bad correction to be shorter than the good corrections. If the snake was too short, or too long, then at least one of the good corrections would be shorter than the bad correction. The short-cut edges work by essentially making sure the snake is as small as possible, effectively removing it from the single-LOM. For example, in Figs. 25 and 28, the different pairs of leftover defects are separated by a set of $\Theta(d)$ time-like edges in the single-LOM in the second window. The k-type short-cut edges here act to reduce this time-like distance to 1, so that good corrections 2, 3, and 4 in Fig. 25(c), and good correction 2 in Fig. 28(b), are always the minimum-weight correction. On the other hand, one might worry that the short-cut edges might "help" cause a logical error that is of weight $<d/2$. However, this does not happen because *none* of the short-cut edges have any spatial component; that is, they always connect vertices with the same z and $\tilde{z}$ coordinates. We hope to include a proof of Conjecture B1 either in a later version of this manuscript or in a future publication.

One helpful lemma that highlights the importance of the short-cut edges is the following:

Lemma B2. For any decoding subgraph $G=(V,E)$ for a single-LOM, and for any pair of vertices $v(t,k,z,\tilde{z})$,

$v(t',k',z',\tilde{z}') \in V \setminus V_{\text{bdy}}$, the minimum-weight edge string s with vertex boundary $\partial s = \{v(t,k,z,\tilde{z}), v(t',k',z',\tilde{z}')\}$ is of weight

$$w(s) = |z - z'| + |\tilde{z} - \tilde{z}'| + |t - t'| + \delta_{t,t'}(1 - \delta_{k,k'}). \tag{B10}$$

Moreover, the shortest edge string s with vertex boundary $\partial s = \{v(t,k,z,\tilde{z}), v_{\text{top}}\}$ for some $v_{\text{top}} \in V_{\text{top}}$ has weight $w(s) = d - \tilde{z}$, and the shortest edge string s with $\partial s = \{v(t,k,z,\tilde{z}), v_{\text{bottom}}\}$ for some $v_{\text{bottom}} \in V_{\text{bottom}}$ has $w(s) = \tilde{z}$.

Importantly, the weight Eq. (B10) does *not* depend on the logical structure of the single-LOM. Therefore, if the same pair of defects arise in two different single-LOMs, they can be matched with a correction of the same weight in each single-LOM.

Proof. We prove Lemma B2 as follows. We begin with the edge string between a pair of vertices. If $t \neq t'$ then assume without loss of generality $t < t'$. Since every (real or short-cut) edge has weight 1 and increments at most one coordinate $z, \tilde{z}$ or t by 1, it is clear that every such edge string has weight lower-bounded by $|z - z'| + |\tilde{z} - \tilde{z}'| + |t - t'|$. We can construct such a string beginning with a string of t-type edges $s_{\text{time}} = \{v(t_{\text{int}}, k(t_{\text{int}}), z, \tilde{z}), v(t_{\text{int}} + 1, k(t_{\text{int}} + 1), z, \tilde{z})\}$ for $t_{\text{int}} = t, t+1, \ldots, t'-1$. This must have the property that $k(t) = k$, $k(t') = k'$, and $k(t_{\text{int}})$ is in the single-LOM at time t_{int}—such a choice of $k(t)$ is always possible since there is at least one value of k in the single-LOM at every time step. This string of t-type edges s_{time} is then composed with the space-like edge string s_{space} from $v(t',k',z,\tilde{z})$ to $v(t',k',z',\tilde{z}')$ which is of weight $|z - z'| + |\tilde{z} - \tilde{z}'|$, so that in total $w(s) = w(s_{\text{time}}) + w(s_{\text{space}}) = |z - z'| + |\tilde{z} - \tilde{z}'| + |t - t'|$.

If $t = t'$ and $k = k'$ then the statement is clear simply by constructing a space-like string. If $k \neq k'$ then clearly the minimum weight of such a string must be at least $|z - z'| + |\tilde{z} - \tilde{z}'| + 1$, since we must use at least one t-type or k-type edge to move from k to k'. We can construct such an edge string s with the short-cut edge $\{v(t,k,z,\tilde{z}), v(t,k',z,\tilde{z})\}$ and the space-like edge string from $v(t,k',z,\tilde{z})$ to $v(t,k',z',\tilde{z}')$ which is of weight $|z - z'| + |\tilde{z} - \tilde{z}'|$, so that in total $w(s) = |z - z'| + |\tilde{z} - \tilde{z}'| + 1$.

The minimum-weight strings connecting a vertex with the top or bottom spatial boundary with the correct weight are clear since we can simply construct a space-like string that does not contain any t- or k-type edges. ■

[1] B. Zeng, A. Cross, and I. L. Chuang, Transversality versus universality for additive quantum codes, IEEE Trans. Inf. Theory **57**, 6272 (2011).

[2] B. Eastin and E. Knill, Restrictions on transversal encoded quantum gate sets, Phys. Rev. Lett. **102**, 110502 (2009).

[3] H. Bombin and M. A. Martin-Delgado, Topological quantum distillation, Phys. Rev. Lett. **97**, 180501 (2006).

[4] A. Kubica, B. Yoshida, and F. Pastawski, Unfolding the color code, New J. Phys. **17**, 083026 (2015).

[5] J. E. Moussa, Transversal Clifford gates on folded surface codes, Phys. Rev. A **94**, 042316 (2016).

[6] N. P. Breuckmann and S. Burton, Fold-transversal Clifford gates for quantum codes, Quantum **8**, 1372 (2024).

[7] D. Bluvstein *et al.*, Logical quantum processor based on reconfigurable atom arrays, Nature **626**, 58 (2024).

[8] Specs of Quantinuum Model H2 Chip (2025), https://www.quantinuum.com/products-solutions/quantinuum-systems/system-model-h2.

[9] M. Cain, C. Zhao, H. Zhou, N. Meister, J. P. B. Ataides, A. Jaffe, D. Bluvstein, and M. D. Lukin, Correlated decoding of logical algorithms with transversal gates, Phys. Rev. Lett. **133**, 240602 (2024).

[10] H. Zhou, C. Zhao, M. Cain, D. Bluvstein, N. Maskara, C. Duckering, H.-Y. Hu, S.-T. Wang, A. Kubica, and M. D. Lukin, Low-overhead transversal fault tolerance for universal quantum computation, Nature **646**, 303 (2025).

[11] J. Zhang, Z.-Y. Chen, J.-N. Li, T.-H. Wei, H.-Y. Liu, X.-N. Zhuang, Q.-S. Li, Y.-C. Wu, and G.-P. Guo, Scalable constant-time logical gates for large-scale quantum computation using window-based correlated decoding, arXiv:2410.16963 [quant-ph].

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

[13] E. T. Campbell, A theory of single-shot error correction for adversarial noise, Quantum Sci. Technol. **4**, 025006 (2019).

[14] Note that this does not contradict the known existence of hierarchical cellular automaton decoders for the surface code which use local feedback corrections based on noisy parity check data, see Ref. [75] and references therein.

[15] M. McEwen, D. Bacon, and C. Gidney, Relaxing hardware requirements for surface code circuits using time-dynamics, Quantum **7**, 1172 (2023).

[16] N. Sundaresan, T. J. Yoder, Y. Kim, M. Li, E. H. Chen, G. Harper, T. Thorbeck, A. W. Cross, A. D. Córcoles, and M. Takita, Demonstrating multi-round subsystem quantum error correction using matching and maximum likelihood decoders, Nat. Commun. **14**, 2852 (2023).

[17] K. Sahay, Y. Lin, S. Huang, K. R. Brown, and S. Puri, Error correction of transversal CNOT gates for scalable surface code computation, PRX Quantum **6**, 020326 (2025).

[18] K. H. Wan, M. Webber, A. G. Fowler, and W. K. Hensinger, An iterative transversal CNOT decoder, arXiv:2407.20976 [quant-ph].

[19] A. Guernut and C. Vuillot, Fault-tolerant constant-depth Clifford gates on toric codes, arXiv:2411.18287 [quant-ph].

[20] Z.-H. Chen, M.-C. Chen, C.-Y. Lu, and J.-W. Pan, Transversal logical Clifford gates on rotated surface codes with reconfigurable neutral atom arrays, arXiv:2412.01391 [quant-ph].

[21] M. E. Beverland, A. Kubica, and K. M. Svore, Cost of universality: A comparative study of the overhead of state distillation and code switching with color codes, PRX Quantum **2**, 020341 (2021).

[22] N. Delfosse and N. H. Nickerson, Almost-linear time decoding algorithm for topological codes, Quantum **5**, 595 (2021).
[23] N. Delfosse, A. Paetznick, J. Haah, and M. B. Hastings, Splitting decoders for correcting hypergraph faults, arXiv:2309.15354 [quant-ph].
[24] C. Gidney, Stim: A fast stabilizer circuit simulator, Quantum **5**, 497 (2021).
[25] M. L. Turner, E. T. Campbell, O. Crawford, N. I. Gillespie, and J. Camps, Scalable decoding protocols for fast transversal logic in the surface code, arXiv:2505.23567 [quant-ph].
[26] L. Caune, L. Skoric, N. S. Blunt, A. Ruban, J. McDaniel, J. A. Valery, A. D. Patterson, A. V. Gramolin, J. Majaniemi, K. M. Barnes *et al.*, Demonstrating real-time and low-latency quantum error correction with superconducting qubits, arXiv:2410.05202.
[27] Google Quantum AI and Collaborators, Quantum error correction below the surface code threshold, Nature **638**, 920 (2024).
[28] Y. Tomita and K. M. Svore, Low-distance surface codes under realistic quantum noise, Phys. Rev. A **90**, 062320 (2014).
[29] T. E. O'Brien, B. Tarasinski, and L. DiCarlo, Density-matrix simulation of small surface codes under current and projected experimental noise, npj Quantum Inf. **3**, 39 (2017).
[30] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, and E. T. Campbell, Parallel window decoding enables scalable fault tolerant quantum computation, Nat. Commun. **14**, 7040 (2023).
[31] X. Tan, F. Zhang, R. Chao, Y. Shi, and J. Chen, Scalable surface-code decoders with parallelization in time, PRX Quantum **4**, 040344 (2023).
[32] J. Viszlai, J. D. Chadwick, S. Joshi, G. S. Ravi, Y. Li, and F. T. Chong, Predictive window decoding for fault-tolerant quantum programs, arXiv:2412.05115 [quant-ph].
[33] M. Cain, D. Bluvstein, C. Zhao, S. Gu, N. Maskara, M. Kalinowski, A. A. Geim, A. Kubica, M. D. Lukin, and H. Zhou, Fast correlated decoding of transversal logical algorithms, arXiv:2505.13587 [quant-ph].
[34] D. Bacon, S. T. Flammia, A. W. Harrow, and J. Shi, Sparse quantum codes from quantum circuits, IEEE Trans. Inf. Theory **63**, 2464 (2017).
[35] N. Delfosse and A. Paetznick, Spacetime codes of Clifford circuits, arXiv:2304.05943 [quant-ph].
[36] D. Gottesman, Opportunities and challenges in fault-tolerant quantum computation, arXiv:2210.15844 [quant-ph].
[37] See Supplemental Material at http://link.aps.org/supplemental/10.1103/sk5y-25b1 for details.
[38] For brevity we use the term *hyperedge* to refer to hyperedges of any weight, meaning vertex support—including weight two—while we reserve the term *edge* to only refer to weight-2 hyperedges.
[39] Observing edge sets have been previously depicted in 3D decoding hypergraphs as logical sheets, see Refs. [26,76].
[40] A different definition of the observing edge set, for example, one that includes space-like edges in the bulk of the surface code rather than just on the spatial boundary, can increase the minimum weight of the space-time stabilizer (say to three), but it will always be of constant weight.
[41] 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).
[42] One can show that it is necessary in some circuits to have more than $L = 2$ tracks, and that it is sufficient for arbitrary circuits to have $L = 2k$ tracks where k is the number of logical qubits in the circuit, leaving aside issues with decoding near fragile time-boundaries which can be solved. Because the focus here is to show that hierarchical matching does *not* work in general, we do not provide a proof of these statements.
[43] O. Higgott and C. Gidney, Sparse blossom: Correcting a million errors per core second with minimum-weight matching, Quantum **9**, 1600 (2025).
[44] M. Serra-Peralta, surface-sim on GitHub (2024), https://github.com/MarcSerraPeralta/surface-sim.
[45] M. Serra-Peralta, lomatching on GitHub (2024), https://github.com/MarcSerraPeralta/lomatching.
[46] R. Versluis, S. Poletto, N. Khammassi, B. Tarasinski, N. Haider, D. J. Michalak, A. Bruno, K. Bertels, and L. DiCarlo, Scalable Quantum Circuit and Control for a Superconducting Surface Code, Phys. Rev. Appl. **8**, 034021 (2017).
[47] K. Wintersperger, F. Dommert, T. Ehmer, A. Hoursanov, J. Klepsch, W. Mauerer, G. Reuber, T. Strohm, M. Yin, and S. Luber, Neutral atom quantum computing hardware: Performance and end-user perspective, EPJ Quantum Technol. **10**, 32 (2023).
[48] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. J. Norris, C. K. Andersen, M. Müller, A. Blais, C. Eichler, and A. Wallraff, Realizing repeated quantum error correction in a distance-three surface code, Nature **605**, 669 (2022).
[49] J. Combes, C. Granade, C. Ferrie, and S. T. Flammia, Logical randomized benchmarking, arXiv:1702.03688 [quant-ph].
[50] N. Lacroix, A. Bourassa, F. J. H. Heras, L. M. Zhang, J. Bausch, A. W. Senior, T. Edlich, N. Shutty, V. Sivak, A. Bengtsson *et al.*, Scaling and logic in the colour code on a superconducting quantum processor, Nature **645**, 614 (2025).
[51] A. R. Calderbank, E. M. Rains, P. M. Shor, and N. J. A. Sloane, Quantum error correction via codes over GF(4), IEEE Trans. Inf. Theory **44**, 1369 (1998).
[52] Note that we do not use the decomposition from Ref. [77] as the hypergraphs would have a particular structure across all circuits and thus this would ruin the purpose of the two-qubit Clifford experiments.
[53] C. Gidney, M. Newman, A. Fowler, and M. Broughton, A fault-tolerant honeycomb memory, Quantum **5**, 605 (2021).
[54] S. J. Evered, D. Bluvstein, M. Kalinowski, S. Ebadi, T. Manovitz, H. Zhou, S. H. Li, A. A. Geim, T. T. Wang, N. Maskara *et al.*, High-fidelity parallel entangling gates on a neutral-atom quantum computer, Nature **622**, 268 (2023).
[55] P. S. Rodriguez, J. M. Robinson, P. N. Jepsen, Z. He, C. Duckering, C. Zhao, K.-H. Wu, J. Campo, K. Bagnall, M.

Kwon *et al.*, Experimental demonstration of logical magic state distillation, Nature **645**, 620 (2025).

[56] T. M. Graham, L. Phuttitarn, R. Chinnarasu, Y. Song, C. Poole, K. Jooya, J. Scott, A. Scott, P. Eichler, and M. Saffman, Midcircuit Measurements on a Single-Species Neutral Alkali Atom Quantum Processor, Phys. Rev. X **13**, 041051 (2023).

[57] E. Deist, Y.-H. Lu, J. Ho, M. K. Pasha, J. Zeiher, Z. Yan, and D. M. Stamper-Kurn, Mid-Circuit Cavity Measurement in a Neutral Atom Array, Phys. Rev. Lett. **129**, 203602 (2022).

[58] M. Serra-Peralta, qec-util on GitHub (2024), https://github.com/MarcSerraPeralta/qec-util.

[59] E. B. Wilson, Probable inference, the law of succession, and statistical inference, J. Am. Stat. Assoc. **22**, 209 (1927).

[60] S. Heußen, D. Winter, M. Rispler, and M. Müller, Dynamical subset sampling of quantum error-correcting protocols, Phys. Rev. Res. **6**, 013177 (2024).

[61] L. D. Brown, T. T. Cai, and A. DasGupta, Interval estimation for a binomial proportion, Stat. Sci. **16**, 101 (2001).

[62] T. Hillmann, G. Dauphinais, I. Tzitrin, and M. Vasmer, Single-shot and measurement-based quantum error correction via fault complexes, Phys. Rev. A **112**, L040401 (2025).

[63] Z. Wang, Surface code error correction during logical operations, Master's thesis, TU Delft, 2023.

[64] A. R. O'Rourke and S. Devitt, Compare the pair: Rotated versus unrotated surface codes at equal logical error rates, Phys. Rev. Res. **7**, 033074 (2025).

[65] M. Serra-Peralta, mle-decoder on GitHub (2025), https://github.com/MarcSerraPeralta/mle-decoder.

[66] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes, arXiv:1108.5738 [quant-ph].

[67] Gurobi Optimization, Gurobi Optimizer Reference Manual (2024), https://www.gurobi.com.

[68] A. Nahum, S. Vijay, and J. Haah, Operator Spreading in Random Unitary Circuits, Phys. Rev. X **8**, 021014 (2018).

[69] B. M. Terhal, Quantum error correction for quantum memories, Rev. Mod. Phys. **87**, 307 (2015).

[70] S.-H. Lee, A. Li, and S. D. Bartlett, Color code decoder with improved scaling for correcting circuit-level noise, Quantum **9**, 1609 (2025).

[71] N. Gillett, Parallel and low-latency streaming decoding of quantum error correction codes (2025), APS March Meeting 2025. https://summit.aps.org/events/MAR-B16/13.

[72] Delft High Performance Computing Centre (DHPC), DelftBlue Documentation (2023), https://doc.dhpc.tudelft.nl/delftblue/, accessed: 2023-05-15.

[73] M. Serra-Peralta, data-lom-decoders on GitHub (2025), https://github.com/MarcSerraPeralta/data-lom-decoders.

[74] Stronger examples where the *unique* minimum-weight correction in each single-LOM leads to a logical error, can also be constructed, but just require another $O(1)$ physical errors.

[75] S. Balasubramanian, M. Davydova, and E. Lake, A local automaton for the 2D toric code, arXiv:2412.19803 [quant-ph].

[76] C. Chamberland and E. T. Campbell, Universal quantum computing with twist-free and temporally encoded lattice surgery, PRX Quantum **3**, 010331 (2022).

[77] D. Grier and L. Schaeffer, The classification of Clifford gates over qubits, Quantum **6**, 734 (2022).