

Hypergraph-based approximate maximum likelihood decoding for quantum codes under circuit-level noise

Ke Chen¹, Mingzheng Zhu¹, Haishan Song¹, Wei Xie^{1,*}, and Xiang-Yang Li^{1,2}

¹*School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China*

²*Hefei National Laboratory, University of Science and Technology of China, Hefei 230088, China*



(Received 6 August 2025; accepted 12 April 2026; published 20 May 2026)

Maximum likelihood decoding (MLD) achieves optimal accuracy but is computationally infeasible for large-distance codes. Tensor-network-based MLD (TN-MLD) improves efficiency but has so far only been shown to work for small-distance surface codes. We introduce a hypergraph-based approximate MLD (HAML) that enables more efficient decoding of more general codes under circuit-level noise. Unlike TN-MLD, our HAML employs syndrome-guided hyperedge selection and probability-based truncation to reduce redundant computation. We benchmarked our approach on surface codes, color codes, and bivariate-bicycle (BB) codes under a circuit-level noise model. Numerical results demonstrate that HAML can decode BB codes with hundreds of qubits, which MLD and TN-MLD cannot handle. For a surface code with distance and repetition rounds both equal to 3, HAML achieves optimal decoding accuracy while being about $323\times$ faster than MLD and about $551\times$ faster than TN-MLD. HAML also achieves higher accuracy than common decoders—minimum-weight perfect matching, Chromobius, and BP + OSD—in the two smallest parameter settings we considered, reducing logical error rates by an average of 27.8% for surface codes, 58.5% for color codes, and 74.4% for BB codes through degeneracy-aware and precise hypergraph selection.

DOI: [10.1103/ydb8-vd3x](https://doi.org/10.1103/ydb8-vd3x)

I. INTRODUCTION

Quantum computing is fundamentally limited by quantum noise, with state-of-the-art platforms exhibiting physical error rates around 10^{-3} [1,2]. In contrast, fault-tolerant computation requires logical error rates below 10^{-10} [2] for large-scale applications. Quantum error correction (QEC) bridges this gap by detecting and correcting logical errors. Among widely studied quantum error-correcting codes (QECCs) are the surface code [2–4], the color code [5,6], and the bivariate-bicycle (BB) code [7].

In QEC protocols, decoders interpret measured syndromes to infer the underlying logical errors. A practical decoder must balance real-time performance with high accuracy: It must process syndromes faster than their generation rate to avoid computational backlogs and exponential slowdowns [8,9], while maintaining sufficient accuracy to suppress logical error rates efficiently as the code scales. Unlike approaches [10–14] aiming at speed under acceptable accuracy, this work focuses on designing a decoder approaching theoretical optimal accuracy.

Maximum likelihood decoding (MLD) [15] achieves optimal theoretical performance. For each syndrome, it evaluates the probabilities of the logical error cosets consistent with

that syndrome and selects a representative of the most probable equivalence class as the recovery operator. However, MLD's exponential complexity restricts demonstrations under circuit-level noise to the smallest code distances for multiple error-correction rounds [16]. Tensor-network-based MLD (TN-MLD) methods [4,15,17] reformulate the MLD as a tensor-network contraction, enabling approximate contraction techniques. The efficiency of these approximations depends heavily on the network's structure. Under the code-capacity noise model, the surface code forms a two-dimensional (2D) square lattice [15], enabling efficient computation. In the phenomenological noise model, handling measurement errors requires repeated measurements, expanding the network to three dimensions (3D). Finally, circuit-level noise model creates a complex 3D irregular lattice [17], restricting practical decoding to small distances ($d = 3$ or $d = 5$) [4]. Reference [17] proposed an offline precompression that reshapes the network into a regular 3D cubic lattice, demonstrated up to $d = 7$. PLANAR [18] maps MLD to a statistical physics problem, enabling efficient decoding for 2D grids but remaining inapplicable to 3D networks.

Many scalable decoders rely on most likely error (MLE) decoding. This ignores the effect of degeneracy [17,19,20], where different physical errors belong to the same logical equivalence class, thereby potentially degrading decoding performance. This approach is more efficient than MLD, but by neglecting degeneracy, it approximates rather than achieves the theoretically optimal accuracy of MLD [4,16]. In addition, decoding strategies are often tailored to the specific code structure and noise model, with specialized approximations designed accordingly. For the surface code, the minimum-weight perfect matching (MWPM) [2,12,14] performs well

*Contact author: xxieww@ustc.edu.cn

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.

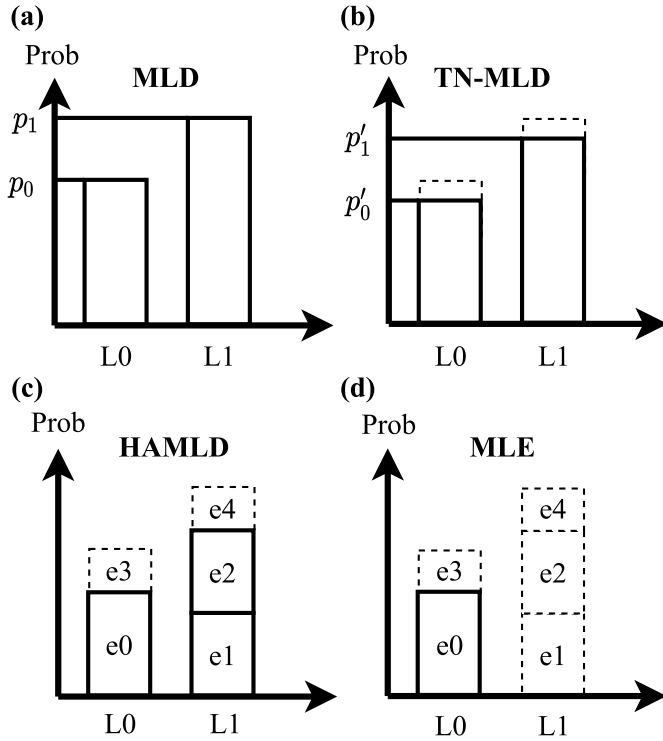


FIG. 1. Example decoding outcomes for a given syndrome, where L0 denotes no logical error and L1 a logical X/Z error: (a) MLD computes exact probabilities and selects L1; (b) TN-MLD approximates probabilities via tensor network contraction and selects L1; (c) HAML D evaluates dominant error patterns from hypergraph contraction and selects L1; and (d) MLE chooses the most probable single error pattern, yielding L0 here. This example is special; in most cases, all decoders produce the same outcome.

under phenomenological noise, where decoding reduces to a graph problem, but its accuracy drops under circuit-level noise due to hypergraph decomposition [21]. Belief propagation (BP), effective for classical LDPC codes, often fails to converge in quantum codes because of short cycles and trapping sets [14,22]. To overcome this, BP is typically combined with ordered statistics decoding (OSD), but the computational cost grows rapidly under circuit-level noise [23]. For BB codes, recent heuristic decoders such as Relay-BP [14] and search-based decoders [24,25] achieve favorable accuracy-efficiency trade-offs, yet fundamentally remain MLE based. By partially accounting for degeneracy, it is possible to significantly reduce the computational complexity and accelerate the decoding process while incurring only minimal loss in decoding accuracy.

We propose hypergraph-based approximate maximum likelihood decoding (HAML D), which performs approximate contraction on hypergraphs built from detector error models under circuit-level noise. As shown in Fig. 1, HAML D differs from TN-MLD in that it contracts only a subset of hyperedges corresponding to the major error patterns generating the observed syndromes, greatly reducing computational overhead. Selecting a subset of hyperedges relies on the syndrome-guided hyperedge selection module, which can employ simple connectivity-based methods or more advanced MLE-inspired search strategies [24,25]. As shown in Fig. 1, by

TABLE I. Comparison of QEC decoders. Acc denotes decoding accuracy, Spd decoding speed, Gen generality across codes and noise, and Deg degeneracy support.

Decoder	Acc	Spd	Gen	Deg
MWPM [21]	Medium	Real-time	✗	✗
Chromobius [35]	Medium	High	✗	✗
BP + OSD [23,36]	Medium	High	✓	✗
Relay-BP [14]	High	Real-time	✓	✗
Search-based [24,25]	High	High	✓	✗
MLD [15,16]	Optimal	Low	✓	✓
TN-MLD [4,15,17]	High	Medium	✗	✓
PLANAR [18]	Optimal	High	✗	✓
HAML D (ours)	High	High	✓	✓

considering multiple error patterns, HAML D partially accounts for degeneracy, allowing it to correctly decode some syndromes that standard MLE would misinterpret. Finally, contraction efficiency is improved by combining optimal contraction ordering [16] with probability-based approximate contraction. As shown in Table I, HAML D is compared with other commonly used decoders.

We evaluate various decoding methods for surface, color, and BB codes under multiround settings ($r = d$). HAML D demonstrates strong generality, scaling to BB codes with hundreds of qubits, where exact MLD or TN-MLD is infeasible. At the two smallest parameter settings we considered, HAML D achieves lower logical error rates than commonly used decoders, reducing them by an average of 27.8% for surface codes, 58.5% for color codes, and 74.4% for BB codes. This improvement stems from both accurate hyperedge selection and explicit handling of degeneracy. We designed sensitivity to analyze the contribution of each module. It highlights two key findings: Greedy ordering substantially reduces decoding overhead, and hyperedge selection is the main determinant of performance—effective at small scales but requiring code-specific optimization for larger codes.

The paper is organized as follows. Section II reviews essential background, including noise models, quantum error-correcting codes, detector error models, and the MLD algorithm. Section III presents the design motivation and architecture of HAML D, while Sec. IV details its methodology. Section V reports and analyzes experimental results, and Sec. VI summarizes the key findings. The Appendixes provide additional information: Appendix A describes the experimental setup; Appendix B discusses the advantages of the decoder in scenarios without data-qubit measurements and its potential applications; Appendix C examines topological properties of decoding hypergraphs and connectivity graphs; and Appendix D outlines parallel contraction strategies.

II. BACKGROUND

A. Noise model

Early studies focusing on the code-capacity noise model assumed that noise occurs on the data qubits prior to the syndrome measurement circuits [26], allowing decoders such as MWPM to operate without requiring detailed noise

models [27]. However, this assumption breaks down in realistic circuits, where every operation can introduce noise.

The phenomenological model [3,28,29] accounts for errors on ancilla qubits (i.e., faults in syndrome measurement circuits) and requires multiple rounds of syndrome measurements. However, it still assumes errors occur independently on each qubit, making it quite different from the noise in real devices.

Circuit-level models [4,30,31] are currently the most accurate and widely adopted error models. They account for noise introduced by every individual operation, including gates, idle steps, and measurements. The specific circuit-level noise model used in this study is detailed in Appendix A 1.

B. Quantum error-correcting code

In this work, we adopt the standard $[[n, k, d]]$ notation to describe QECCs, where n denotes the number of data qubits (excluding ancilla qubits), k is the number of encoded logical qubits, and d is the code distance.

To construct and analyze syndrome measurement circuits, it is necessary to utilize the concept of stabilizers [32]. All QECCs considered in this work are stabilizer codes, which define a 2^k -dimensional code space stabilized by an abelian subgroup $\mathcal{S}$ of the n -qubit Pauli group Π^n , excluding the element $-I^{\otimes n}$. The stabilizer group is generated by $n - k$ independent, commuting Pauli operators, referred to as stabilizer generators:

$$\{g_1, g_2, \dots, g_{n-k} \mid g_i \in \mathcal{S}\}, \quad (1)$$

such that any code state $|\psi\rangle$ satisfies

$$g_i |\psi\rangle = +1 \cdot |\psi\rangle, \quad \forall i = 1, \dots, n - k. \quad (2)$$

These generators play a role analogous to the parity-check matrix in classical error-correcting codes, facilitating error detection and correction.

Since direct measurement of multiqubit stabilizer generators is infeasible in quantum hardware, syndrome measurement circuits employ ancilla qubits: They are first initialized in the $|0\rangle$ state, then entangled with data qubits via cnot or cz gates according to the stabilizer structure, and finally measured to obtain the outcomes. These outcomes are classically processed to yield syndromes, which are then used to identify and correct errors in the system.

C. Detector error model

The detector error model (DEM) is a data structure that includes a Tanner graph and information about the probabilities of error occurrences. In classical coding theory, a Tanner graph is a bipartite graph that connects parity-check measurements (syndromes s) to errors e through the relation $s = He$, where H denotes the parity-check matrix. In circuit-level noise models, the DEM is obtained using stim [30] via Pauli frame tracking.

In the DEM, detectors (D) are binary variables that record syndrome information, typically derived from measurements of ancilla qubits. Similarly, logical observables (L) are binary variables indicating whether a logical qubit has experienced a logical error. Logical errors, classified as bit flip or phase flip,

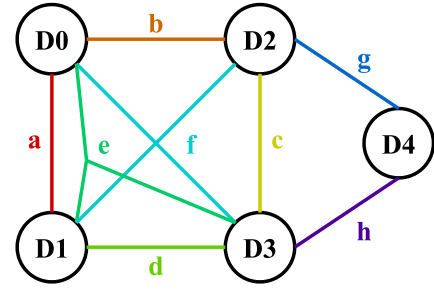


FIG. 2. A hypergraph representation of a DEM.

are detected via logical Z or X measurements, implemented by measuring the relevant data qubits in the Z or X basis and combining outcomes to determine whether a logical error has occurred. An error mechanism is a set of physical errors that produce the same flips in all detectors and logical observables, occurring with probability p . An error pattern is the set of mechanisms that occur in a single shot, with its probability given by the product of p for the mechanisms that occur and $1 - p$ for those that do not.

For a QECC with $m = |s|$ detectors, the syndrome is represented as a binary vector:

$$s = (D_0, D_1, \dots, D_{|s|-1}), \quad (3)$$

where each bit indicates whether the corresponding detector flipped. The calculation of $|s|$ for surface codes, color codes, and BB codes, as well as the logical error rate after decoding from the syndrome and logical observables, can be found in Appendix A 2. The number of flipped bits in a syndrome defines its hamming weight (HW) [13].

Additionally, the DEM can be naturally formulated as a hypergraph $H = (V, E)$, where V denotes the set of nodes—comprising all detectors [33] and logical observables—and E is the set of hyperedges, each representing an error mechanism. A hyperedge e connects the subset of nodes flipped by the corresponding error mechanism (i.e., detectors and logical observables), and its weight w_e is given by the probability p of that error mechanism. Figure 2 depicts a hypergraph with five detectors (D_0 – D_4) and eight error mechanisms (a – h); for instance, error f flips detectors D_0 – D_3 , while the logical observable L_0 is excluded from the analysis due to its noncontractibility. A more detailed and realistic description of the hypergraph structure can be found in Appendix C.

D. Maximum likelihood decoder

Early work [15] introduced MLD and TN-MLD under code-capacity noise models, where the tensor-network topology follows the code graph, with one tensor per physical qubit, increasing linearly with the number of measurement rounds.

When extended to circuit-level noise models [16], the decoding problem can still be formulated as a tensor-network contraction. However, even for special surface codes, the corresponding tensor networks become highly irregular three-dimensional graphs. Moreover, the number of tensors scales with the sum of detectors and error mechanisms in the detector error model (i.e., the number of nodes plus hyperedges), far exceeding the number of physical qubits multiplied by the

number of repeated rounds (approximately equal to the number of nodes). As a result, decoding becomes infeasible even at small code distances. Only by introducing additional network compression techniques [17] can these methods be executed, and even then only up to code distance seven. This limitation is even more severe when attempting to extend MLD-based approaches to color codes or BB codes.

This paper focuses on the online MLD strategy [34] based on hypergraph contraction. The core task is to compute the probability distribution $\Pr[s, l]$, where $s \in \mathbb{Z}_2^{|V|}$ represents a syndrome of length $|V|$, and $l \in \mathbb{Z}_2^k$ indicates whether logical errors have occurred on the k logical qubits.

By contracting hyperedges in the hypergraph, the probability distribution $\Pr[s, l]$ can be computed to perform maximum likelihood decoding. The initial distribution is set as $\Pr[0^{|V|}, 0^k] = 1$. Contracting a hyperedge e updates the distribution by accounting for both its occurrence and nonoccurrence, after which e is removed from the hypergraph. The updated distribution after contracting a hyperedge e is given by

$$\Pr_{\text{new}}[s, l] \leftarrow [(1 - p_e) \times \Pr[s, l], p_e \times \Pr[s \oplus s_e, l \oplus l_e]],$$

where p_e represents the probability of hyperedge e occurring, and s_e and l_e represent the flipped detectors and the logical state flip induced by the error mechanism associated with hyperedge e , respectively.

When all hyperedges have been contracted, and the syndrome measurement result s^* is known, the most likely logical error l^* can be determined based on the final probability distribution $\Pr[s, l]$:

$$l^* = \arg \max_l \Pr[s^*, l]. \quad (4)$$

In the actual hypergraph contraction, we perform node contractions sequentially, contracting the uncontracted hyperedges connected to each node. Since s^* is known, after contracting all hyperedges associated with each detector D_j , lossless pruning of the probability distribution can be performed based on the observed value s_j^* , thereby reducing computational complexity.

III. THE HAML D OVERVIEW

Existing circuit-level MLD methods [16] contract all hyperedges without approximation in a fixed detector-based order, resulting in extremely high computational costs. Although recent studies have explored approximate tensor-network contractions [4, 17], involving optimization of contraction order and approximate tensor-network methods, they still cannot be applied to large-scale surface codes or BB codes.

Tensor-network approximation methods may not be optimal for decoding, since they focus on minimizing global tensor errors, whereas decoding only requires identifying the most probable error configuration. By formulating the process in terms of hypergraph contractions, we have uncovered approximation and acceleration strategies that are comparable to, or even surpass, existing approaches:

(1) Contracting only the hyperedges strongly correlated with flipped syndrome bits introduces a very small error. As

TABLE II. Sampled 10^5 instances. Average and maximum syndrome HW and the ratio (in %) of average HW to syndrome length for different $[[n, k, d]]$ codes and rounds r .

$[[n, k, d]]$	r	Length	Average HW	Max HW	Ratio (%)
$[[72, 12, 6]]$	1	72	2.452	21	3.41
$[[72, 12, 6]]$	6	432	24.924	72	5.77
$[[144, 12, 12]]$	1	144	4.879	27	3.39
$[[144, 12, 12]]$	12	1728	100.920	194	5.84

shown in Table II, most syndromes have low HW, so only a small subset of hyperedges typically needs to be contracted.

(2) Optimizing the contraction order is a lossless acceleration technique in tensor-network contraction and proves effective for the MLD problem as well. However, it requires a redefinition of the contraction cost.

(3) Many low-probability terms in the distribution are unlikely to be selected in the final decoding. Discarding these terms can significantly accelerate the decoding process without compromising accuracy.

Based on this, we propose the HAML D decoder, shown in Fig. 3, which has three main parts:

(1) Syndrome-guided hyperedge selection: A priority measure is defined to select a subset of high-probability hyperedges relevant to the syndromes, constructing a simplified subhypergraph. The size of this subset can be tuned to achieve a trade-off between accuracy and computational efficiency.

(2) Greedy contraction order search: By redefining the contraction cost, we develop a fast greedy strategy to determine the optimal contraction sequence of nodes.

(3) Approximate hypergraph contraction: During contraction, probability-based truncation retains only the high-probability terms while discarding negligible ones. The truncation parameter can be tuned to balance accuracy and computational efficiency.

While hyperedge selection and approximation may introduce limited accuracy loss, their overall impact on decoding performance is minimal. The next section presents the detailed design of our method.

IV. THE HAML D METHOD

This section presents the design of the HAML D method, illustrated in Fig. 3. The method consists of three modules: (1) syndrome-guided hyperedge selection; (2) greedy contraction order search; and (3) approximate hypergraph contraction.

A. Syndrome-guided hyperedge selection

The hyperedge selection consists of two steps. First, we represent the DEM as a hypergraph. We transform the DEM into a hypergraph where each detector is a node and each error mechanism forms a hyperedge weighted by its probability.

Then, we select hyperedges correlated with flipped nodes from the hypergraph to construct a subhypergraph. For a given set of flipped nodes, we propose a beam search method to select relevant hyperedges, proceeding in five main steps:

(1) Initialization: Maintain a beam set $\mathcal{B} = \{(\mathcal{D}, E_{\text{cur}}, \pi_{\text{sum}}, w_{\text{sum}})\}$, where each beam state contains

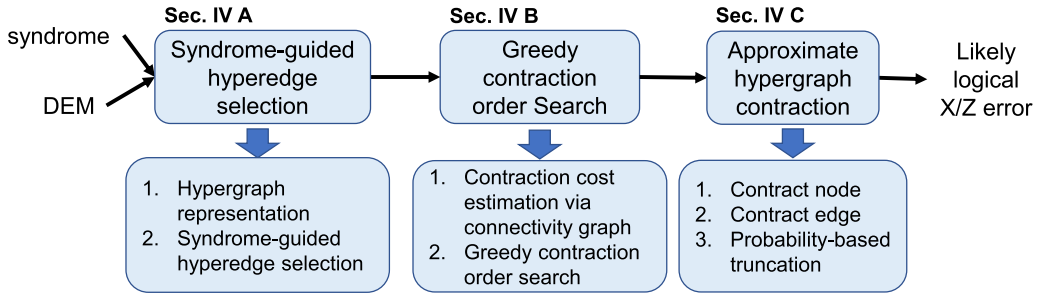


FIG. 3. Under the circuit-level noise model, the figure illustrates the decoding flow of HAML, consisting of three core modules: syndrome-guided hyperedge selection, greedy contraction order search, and approximate hypergraph contractions. The lower part further outlines the suboperations of each module, detailed in the corresponding sections.

the current flipped node set $\mathcal{D}$, the selected hyperedge set E_{cur} , the accumulated priority π_{sum} , and the accumulated log-likelihood weight w_{sum} .

(2) Heuristic hyperedge selection: For each beam $B_i \in \mathcal{B}$, collect candidate hyperedges connected to $\mathcal{D}_i$. Compute each candidate's priority $\pi(\mathcal{D}_i, e)$ and log-likelihood $-\ln w_e$, then sort candidates in descending order by both values. Select the top- $k_{\text{hyperedge}}$ hyperedges to generate $k_{\text{hyperedge}}$ beam states, updating

$$2\mathcal{D}'_i = \mathcal{D}_i \oplus e, \quad E'_{\text{cur}} = E_{\text{cur}} \cup \{e\}, \\ \pi'_{\text{sum}} = \pi_{\text{sum}} + \pi(\mathcal{D}_i, e), \quad w'_{\text{sum}} = w_{\text{sum}} - \ln w_e.$$

(3) Deduplication and pruning: Remove duplicate states based on E_{cur} and retain only the top- k_{beam} beams according to π_{sum} and w_{sum} .

(4) Termination: Repeat steps 2 and 3 until either the maximum search depth d_{max} is reached, the candidate hyperedge set is empty, or enough beams have empty flipped node sets $\mathcal{D}$.

(5) Subhypergraph construction: Collect the E_{cur} sets from beams whose flipped node sets $\mathcal{D}$ are empty and merge them to form the final subhypergraph.

Specifically, we define the priority metric $\pi(\mathcal{D}, e)$ as

$$\pi(\mathcal{D}, e) = |e \cap \mathcal{D}| - (|e| - |e \cap \mathcal{D}|), \quad (5)$$

where $|e \cap \mathcal{D}|$ counts flipped nodes in e , and $|e| - |e \cap \mathcal{D}|$ counts unflipped nodes. Larger π indicates stronger correlation with $\mathcal{D}$ and higher decoding priority.

The algorithm's main cost is computing and sorting heuristic scores of candidate hyperedges. Sorting $|E_{\text{cand}}|$ candidates per beam requires $O(|E_{\text{cand}}| \ln |E_{\text{cand}}|)$ time. Since $|E_{\text{cand}}| \leq O(nr)$, where n is the number of qubits and r the repetition rounds, the worst-case complexity is $O(d_{\text{max}} k_{\text{beam}} nr \ln(nr))$, where d_{max} and k_{beam} are search hyperparameters, as defined earlier.

In our experiments, we tested various hyperedge selection schemes. For example, the simplest is a static connectivity-based approach, with fixed $\mathcal{D}$ and priorities π computed once for all hyperedges. This scheme often fails to find a solution, performing poorly for multi-round BB codes. To analyze this, we use the metric recall, where E_{error} denotes the true error hyperedges generating the current syndrome s , and E' the set

selected by the algorithm. Recall is defined as

$$\text{recall} = \frac{|E_{\text{error}} \cap E'|}{|E_{\text{error}}|}. \quad (6)$$

Due to degeneracy, many instances have a recall less than 1, yet can still be correctly decoded. To make recall more relevant for analyzing logical error rates, we define logical recall (L-Recall). We extract the nodes corresponding to logical observables from both E_{error} and E' , compute the xor within each set, and assign an L-Recall of 1 if the xors match, and 0 otherwise. In Sec. VB2, we provide a detailed discussion of the performance of our current beam search hyperedge selection method.

B. Greedy contraction order search

Previous work [16] used the detector sequence as the contraction order. Drawing on tensor-network contraction strategies [37–39], we first present a cost estimation for hypergraph contraction, and then propose a greedy search method for determining the contraction order.

When describing the contraction order search method, we first define the contraction cost of a detector node. The contraction cost of the t th node is defined as $C_{D_t} = \text{Size}(t)$, where $\text{Size}(t)$ denotes the size of the resulting probability distribution after contraction. The following explains why this definition is used.

Figure 4 illustrates the node contraction process, which consists of two repeated steps: (1) contracting hyperedges connected to the node and (2) merging identical terms. As shown in Ref. [16], a lossless pruning step can be applied after contracting a detector node, guided by its syndrome value D_t .

As illustrated in Fig. 4, let $\text{Size}(t-1)$ and $\text{Size}(t)$ denote the sizes of the probability distribution before and after contracting the t th node, respectively. Let $\text{Count}(D_t)$ represents the number of hyperedges contracted at step t . The upper bound on the node-contraction cost at this step is given by

$$\max(\text{Size}(t-1), \text{Size}(t)) \times \text{Count}(D_t). \quad (7)$$

Pruning based on D_t can reduce the distribution size by at most half, i.e., $\text{Size}(t) \geq \text{Size}(t-1)/2$. Therefore, the upper bound simplifies to $O(2 \times \text{Count}(D_t) \times \text{Size}(t))$. Since $\text{Count}(D_t)$ is typically a constant of bounded size and does not grow unbounded with d or r , the contraction cost can be further simplified to $O(\text{Size}(t))$.

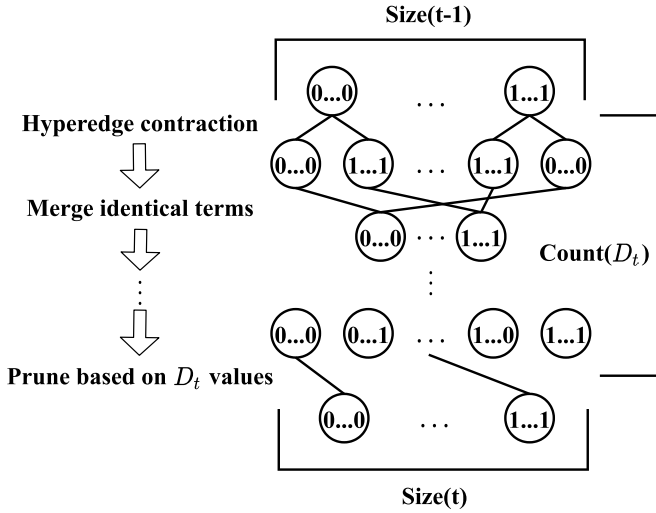


FIG. 4. The maximum contraction cost for the t th detector node is approximately proportional to $\max(\text{Size}(t-1), \text{Size}(t)) \times \text{Count}(D_t)$, where $\max(\text{Size}(t-1), \text{Size}(t))$ denotes the larger of the input and output distribution sizes, and $\text{Count}(D_t)$ indicates the number of hyperedges contracted at step t .

Next, we present a greedy strategy to find the node-contraction order that minimizes the contraction cost. Accurately estimating the distribution size $\text{Size}(t)$ after the t th contraction is challenging. To efficiently approximate this cost, we introduce the connectivity graph to compute the contraction cost.

The connectivity graph is a simplified version of the hypergraph, retaining the same set of nodes. If two nodes are connected by the same hyperedge in the original hypergraph, there is an edge between them in the connectivity graph. Figure 5 illustrates the connectivity graph derived from the hypergraph in Fig. 2.

The contraction width W_t at step t is defined as the size of the neighbor set $\mathcal{N}$ of the searched nodes $\mathcal{S}$, with $\mathcal{N}$ excluding nodes in $\mathcal{S}$. These neighbors are the free variables in the probability distribution, while nodes in $\mathcal{S}$ have been pruned according to the observed syndrome values. The maximum

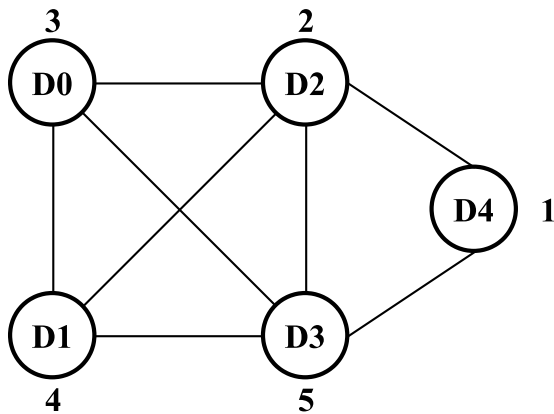


FIG. 5. An example of a connectivity graph and the greedy contraction order. The numeric labels next to the nodes indicate the contraction sequence determined by the greedy search algorithm.

ALGORITHM 1. Greedy contraction order search.

Input: Connectivity graph $G = (V, E)$
Output: Order $O_{\text{order}} = (D^{(1)}, \dots, D^{(|V|)})$
Initialize $O_{\text{order}} \leftarrow ()$;
Initialize $\mathcal{S} \leftarrow \emptyset$;
while $|\mathcal{S}| < |V|$ **do**
 if No neighbor of $\mathcal{S}$ exists in $V \setminus \mathcal{S}$ **then**
 Get $D \in V \setminus \mathcal{S}$ with smallest degree and index;
 Add D to $\mathcal{S}$;
 Append D to O_{order} ;
 else
 Let $\mathcal{N}$ be neighbors of $\mathcal{S}$ in $V \setminus \mathcal{S}$;
 foreach $D_i \in \mathcal{N}$ **do**
 Estimate contraction width W_i of D_i ;
 Compute contraction cost $C_i \leftarrow 2^{W_i}$;
 end
 Select $D^* = \arg \min_{D_i \in \mathcal{N}} C_i$;
 Add D^* to $\mathcal{S}$;
 Append D^* to O_{order} ;
 end
end
return O_{order} ;

size of the contracted distribution is thus bounded by 2^{W_t} . Note that this estimate is an overestimate, as it reflects a locally optimal solution found during the search rather than the global optimum.

The greedy search algorithm for determining the contraction order is presented in Algorithm 1. At each step, the neighbor set $\mathcal{N}$ serves as the pool of candidate nodes for contraction. The algorithm computes the contraction cost for each candidate and selects the node with the lowest cost for the next contraction. If $\mathcal{N}$ is empty, it chooses the node with the smallest degree; if multiple nodes share the smallest degree, the one with the lowest index is selected.

The complexity analysis of the greedy contraction order search algorithm is as follows: At each contraction step, the algorithm selects a node from the candidate set. In practical decoding tasks, the node degree in the connectivity graph is typically constant and does not grow with the code distance d or the number of syndrome measurement rounds r , so the size of the candidate set can be considered constant. Therefore, the time complexity of traversing candidate nodes and their neighbors at each step is $O(1)$. Since the total number of contraction steps is proportional to the number of nodes in the graph, it is approximately $O(rd^2)$.

In the graph shown in Fig. 5, the node with the smallest degree, D_4 , is contracted first. Next, either D_2 or D_3 can be chosen; since both have a contraction width of 2 and contracting either yields the neighboring nodes D_0 and D_1 , the node with the smaller index, D_2 , is selected. The process continues in this manner until the contraction order D_4, D_2, D_0, D_1, D_3 is established.

C. Approximate hypergraph contraction

This section provides a detailed introduction to the approximate hypergraph contraction module in HAML D. The algorithmic procedure is presented in Algorithm 2, which

ALGORITHM 2. Approximate contraction.

Input: $s \in \{0, 1\}^m$: syndrome (m is node count)
 H : hypergraph
 O_{order} : node contraction order ($D_1, \dots, D_m$)
 Pr : initial probability distribution $\{0^{m+k} : 1\}$
 S : approximation strategy with:

- type $\in \{\text{top}, \text{threshold}\}$
- param α : truncation parameter
- stage $\in \{\text{node}, \text{hyperedge}\}$

Output: Approximated distribution Pr^*

Procedure ContractNode($\text{Pr}, H, O_{\text{order}}, S$)

```

foreach  $D_t \in O_{\text{order}}$  do
  foreach  $e \in \text{NodeToContractEdges}(H, D_t)$  do
     $(\text{Pr}, H) \leftarrow \text{ContractEdge}(\text{Pr}, H, e, S)$ ;
  end
   $\text{Pr} \leftarrow \text{Prune}(\text{Pr}, D_t)$ ;
  if  $S.\text{stage} = \text{node}$  then
     $\text{Pr} \leftarrow \text{ProbabilityBasedTruncation}(\text{Pr}, S)$ ;
  end
end
return  $\text{Pr}^*$ ;

```

Function ContractEdge(Pr, H, e, S)

```

 $p \leftarrow H.\text{EdgeWeight}(e)$ ;
 $H \leftarrow H \setminus \{e\}$ ;
 $\text{Pr}_{\text{new}} \leftarrow \emptyset$ ;
for  $(s, q) \in \text{Pr}$  do
   $\text{Pr}_{\text{new}}[s \oplus e] \leftarrow \text{Pr}_{\text{new}}[s \oplus e] + q \cdot p$ ;
   $\text{Pr}_{\text{new}}[s] \leftarrow \text{Pr}_{\text{new}}[s] + q \cdot (1 - p)$ ;
end
if  $S.\text{stage} = \text{hyperedge}$  then
   $\text{Pr}_{\text{new}} \leftarrow \text{ProbabilityBasedTruncation}(\text{Pr}_{\text{new}}, S)$ ;
end
return  $(\text{Pr}_{\text{new}}, H)$ ;

```

Function ProbabilityBasedTruncation(Pr, S)

```

if  $S.\text{type} = \text{top}$  then
  return Top  $\alpha$  elements by probability in  $\text{Pr}$ ;
end
else if  $S.\text{type} = \text{threshold}$  then
  return  $\{(s, q) \in \text{Pr} \mid q \geq \alpha\}$ ;
end

```

comprises three operations: node contraction, hyperedge contraction, and probability-based truncation.

The module takes as input the syndrome s , a hypergraph H (or a subhypergraph), the node contraction order O_{order} , the initial probability distribution Pr , and a specified approximate contraction strategy S . It outputs an approximate distribution Pr^* that estimates the probabilities of different X/Z logical errors conditioned on the given syndrome. The logical error with the highest probability in Pr^* is selected as the most likely decoding outcome.

In the MLD contraction process, all nodes are contracted by sequentially contracting hyperedges connected to each node, with each node contracted only once. Accordingly, our main function is ContractNode.

The ContractNode function outlines five main steps: (1) traverse the nodes to be contracted in the specified order O_{order} ; (2) call the NodeToContractEdges function to obtain the uncontracted hyperedges connected to the current node;

(3) contract the selected hyperedges using the ContractEdge function; (4) perform lossless pruning based on the node's value in the syndrome; and (5) if the contraction strategy at this stage is set to node, invoke the ProbabilityBasedTruncation function.

The ContractEdge function describes the detailed process of contracting a hyperedge and includes the following key steps: (1) obtain the weight of the hyperedge to be contracted, remove it from the hypergraph, and initialize a probability distribution; (2) perform the contraction by updating all entries in the current distribution, considering whether the error mechanism corresponding to the hyperedge occurs or not, which doubles the distribution size; (3) merge identical entries to handle degeneracy; and (4) if the contraction strategy stage is set to hyperedge, invoke the ProbabilityBasedTruncation function.

The probability-based truncation function is the core concept underlying approximate hypergraph contraction. It takes a probability distribution as input and outputs an approximated distribution. Using a truncation parameter α , it retains only a certain number of the highest-probability terms.

To provide flexibility in practice, we define three parameters: type, param (α), and stage. The type specifies the approximation strategy and can be either top or threshold. The top strategy keeps the top α terms with the highest probabilities, while the threshold strategy removes all terms with probabilities below α . The stage indicates when probability-based truncation is applied, typically after node contraction or hyperedge contraction. Node-level approximation offers higher accuracy but may struggle to control computational cost effectively.

In subsequent applications, for BB codes, the combination of type = top and stage = hyperedge performs best, as it better controls overall computation. For surface codes, both hyperedge- and node-level stages perform well. Overall, the top method is preferred over threshold, since the latter's truncation parameter is hard to determine.

The complexity analysis of approximate hypergraph contraction is as follows: In exact MLD, the computational complexity grows exponentially [16]. In contrast, using an approximate strategy with the hyperedge stage and top type results in a worst-case time complexity of $O(N_{\text{max}} \times \alpha)$, where α is the approximation parameter and N_{max} is the maximum number of contracted hyperedges. The space complexity under hypergraph approximate contraction is $O(\alpha)$. It is important to note that N_{max} and α are not fixed or maximum values; they typically scale with $O(rd^2)$. Therefore, for large-scale cases, the approximate contraction complexity generally approaches $O(r^2d^4)$.

V. EVALUATIONS RESULT

This section provides a systematic analysis, based on experimental results, of the regimes in which HAML D outperforms existing decoders, as well as its limitations. In Sec. V A, we show that (1) HAML D achieves faster decoding than MLD and scales to large surface, color, and BB codes; (2) compared with widely used decoders such as MWPM, Chromobius, and belief propagation with ordered statistics decoding (BP

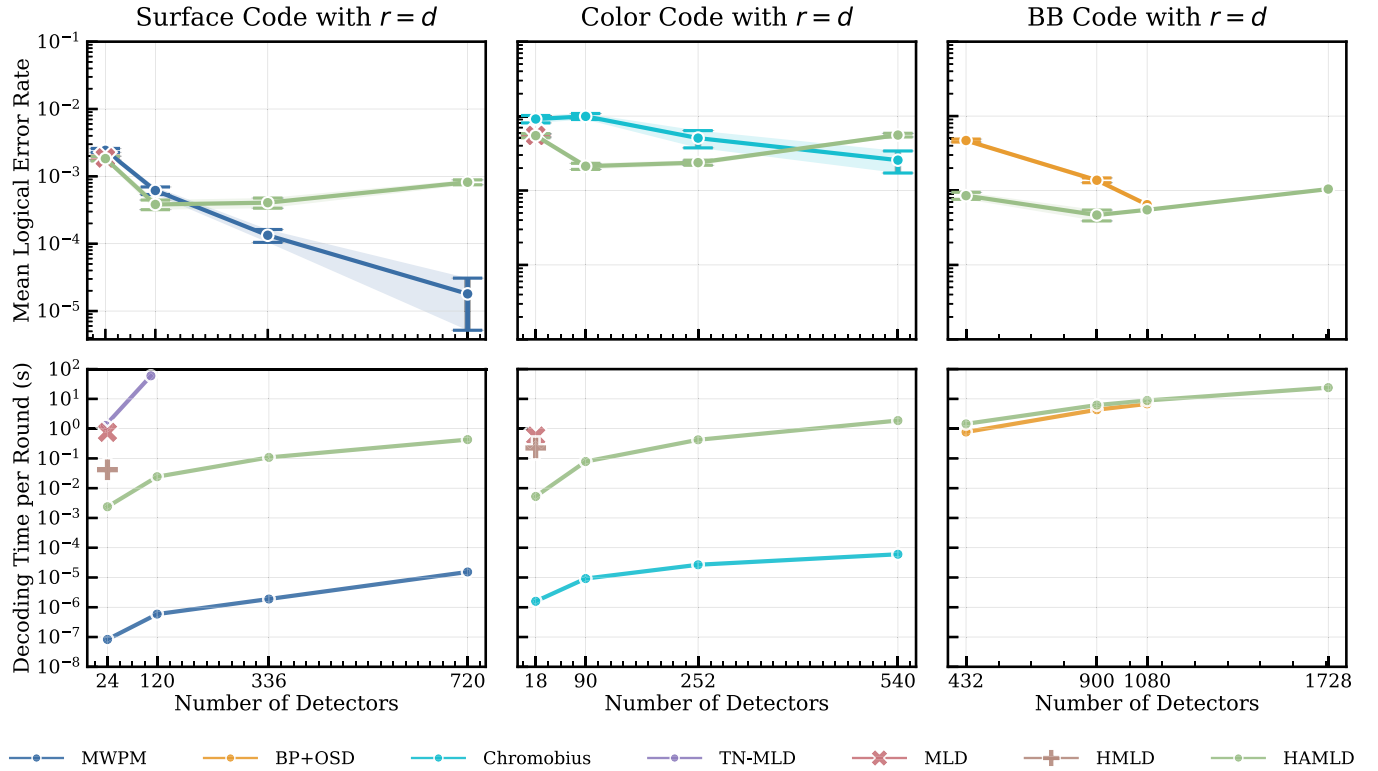


FIG. 6. Example of logical error rates and run-times for decoding under circuit-level noise with $r = d$. Run-time for MLD, hypergraph-based MLD (HMLD) (using only the improved contraction ordering), and TN-MLD is evaluated using 100 randomly sampled syndromes, while all other results are based on 10^5 samples; see Appendix A for details. HAMLd is significantly faster than MLD and TN-MLD, scaling to larger code distances while maintaining comparable accuracy at small distances. It outperforms code-specific decoders in the small-distance regime through explicit degeneracy handling and precise hyperedge selection. At larger distances, its logical error rate increases due to fixed hyperparameters and the difficulty of detecting long-chain errors (see Secs. V A and V B 2). Data points requiring more than 3 days of run-time are not shown.

+ OSD), HAMLd offers higher decoding accuracy at small code distances.

In Sec. V B, we further find that (1) greedy contraction ordering significantly reduces computational cost; (2) the current hyperedge search strategy in HAMLd is efficient and accurate for small instances of arbitrary codes, but requires further optimization at larger scales; and (3) the more accurate the hyperedge selection, the simpler the required approximate contraction.

A. Overall performance of HAMLd

This section evaluates the accuracy and run-time of the HAMLd method across different error-correcting codes, as shown in Fig. 6.

Using the SI1000 noise model with $p = 10^{-3}$, HAMLd sets the hypergraph selection parameters to $[k_{\text{beam}}, k_{\text{hyperedge}}] = [40, 4]$ and $d_{\text{max}} = 60$ to balance accuracy and speed. To enhance decoding performance, it employs a multilevel hyperedge selection strategy: In the first stage, a beam search with 40 beams per step and four hyperedges per beam is applied, doubling these parameters if no valid solution is found. If the search still fails, the method reverts to connectivity-based hyperedge selection for approximate contraction, with the maximum number of hyperedges set

to the number of detectors. As shown in Sec. V B 2, most syndromes are resolved successfully in the first stage.

In the specific sampling instances considered, HAMLd achieves optimal decoding accuracy while performing faster decoding—up to $323\times$ faster than MLD and $551\times$ faster than TN-MLD for a distance-3 surface code—and also scales efficiently to larger surface, color, and BB codes. It furthermore attains higher accuracy than common and specialized decoders in small-scale settings, thanks to degeneracy-aware subhypergraph handling and precise beam-search-based hyperedge selection, reducing logical error rates by 27.8% for surface codes, 58.5% for color codes, and 74.4% for BB codes.

At larger scales, however, performance lags behind some specialized decoders, primarily because the beam search hyperedge selection is not fully optimized and the search parameters are fixed (see Sec. V B 2).

To quantify statistical uncertainty, we computed the averaged X/Z logical error rates and their standard deviations at several small-scale representative points. For the surface code, HAMLd achieved $(1.86 \pm 0.16) \times 10^{-3}$ and $(3.70 \pm 0.58) \times 10^{-4}$ at $(d = 3, r = 3)$ and $(d = 5, r = 5)$, compared with $(2.45 \pm 0.14) \times 10^{-3}$ and $(5.9 \pm 0.8) \times 10^{-4}$ for MWPM. For the color code, HAMLd reached $(5.46 \pm 0.24) \times 10^{-3}$ and $(2.06 \pm 0.15) \times 10^{-3}$ at $d = 3$ and $d = 5$, versus $(9.08 \pm 0.30) \times 10^{-3}$ and $(9.99 \pm 0.32) \times 10^{-3}$ for

TABLE III. Maximum contraction widths and computational costs for different contraction orders in the X basis of surface codes.

d	r	Width (MLD/HMLD)	Cost (MLD/HMLD)
3	1	6/3	$1.43 \times 10^2 / 3.50 \times 10^1$
	3	12/8	$1.99 \times 10^4 / 1.96 \times 10^3$
5	1	15/6	$9.90 \times 10^4 / 8.23 \times 10^2$
	5	35/25	$6.55 \times 10^{11} / 6.98 \times 10^8$
7	1	28/8	$8.12 \times 10^8 / 7.93 \times 10^3$
	7	70/50	$4.54 \times 10^{22} / 4.04 \times 10^{16}$

Chromobius. On the BB code [[72, 12, 6]], HMLD's rate was $(8.54 \pm 0.99) \times 10^{-4}$. In all cases, standard deviations were only 5%–15% of the means, indicating well-controlled statistical fluctuations and confirming HMLD's superior accuracy over baseline decoders in small-scale systems.

B. Sensitivity

We evaluate the three modules and their parameter settings in terms of decoding accuracy and speed, yielding three key insights: (1) HMLD's greedy ordering substantially reduces decoding overhead relative to MLD, particularly for multi-round surface codes. (2) Beam search hypergraph selection accurately identifies syndrome-related errors in small-scale cases, while highlighting the effect of degeneracy on decoding performance. (3) With effective hyperedge selection, approximate contraction becomes unnecessary; for codes with sufficient local structure, such as surface codes, decoding can rely solely on approximate contraction.

1. Contraction order

As shown in Table III, we evaluate the decoding overhead of HMLD with a greedy contraction order compared to conventional MLD with a detector-index-based order, focusing on maximum contraction width and estimated contraction cost.

The results, obtained under the X-basis using the SI1000 noise model with noise strength $p = 10^{-3}$, cover surface codes with code distances $d = 3, 5, 7$, for both single-round and multiround ($r = d$) settings.

In both single-round and multiround cases, HMLD clearly outperforms MLD by reducing the maximum contraction width and computational cost. Without any approximations, decoding a single syndrome at $d = r = 5$ can take 10^5 – 10^6 times longer than at $d = r = 3$. For example, as shown in Fig. 6, decoding a single syndrome at $d = r = 3$ already takes nearly 1 s, making repeated decoding of 10^5 syndromes impractical.

Although HMLD significantly accelerates decoding compared to MLD, its run-time still grows exponentially. For large quantum error-correcting codes, both contraction width and computational cost remain prohibitively high, making approximations or exploiting the sparsity of syndromes in the hypergraph essential.

2. Hyperedge selection

We analyze the effectiveness of the hyperedge selection module by verifying whether it accurately covers the hyper-

TABLE IV. We sampled 10^5 error patterns at $p = 10^{-3}$ with $r = d$. For each pattern, we report the maximum and average HW, which counts the number of errors in the mechanism, and the ratio gives the average HW as a percentage of the pattern length.

Code	d	Length	Average HW	Max HW	Ratio (%)
Surface	3	221	0.326	5	0.147
Surface	9	12 707	9.857	25	0.078
Color	3	144	0.342	5	0.238
Color	9	1728	10.167	25	0.588
BB	6	16 200	8.110	22	0.050
BB	12	67 824	31.711	57	0.047

edges corresponding to true error mechanisms. To do this, we use stim's detector error model sampler, which generates syndromes and logical observables along with the ground-truth error mechanisms for each case.

Statistical analysis of error patterns shows that the number of hyperedges per syndrome is much smaller than the total, supporting the feasibility of our hypergraph selection strategy. As Table IV (from 10^5 instances) indicates, both the average and maximum counts constitute only a tiny fraction of the total. This sparsity is expected to grow as hardware noise decreases. Therefore, the main challenge in decoding is efficiently identifying the most probable hyperedge set—or its equivalence class—for a given syndrome within the large hypergraph.

We evaluate various hyperedge selection parameters using the L-Recall metric from Sec. IV A. “No Recall” counts cases where L-Recall is below 1, and “No Search” counts cases where the search fails to find an empty flipped set. As Table V shows, the method performs well at small code distances but experiences search failures at larger distances, reflecting the difficulty of accurately identifying long error chains with beam search. Balancing search width and computational cost, developing more robust decoders for large-scale, complex error patterns remains an important direction for future work.

For the surface and color codes, we systematically compare logical error rates with and without accounting for degeneracy. When degeneracy is considered, up to four best solutions identified via hypergraph search are retained, and their corresponding hyperedges are used to construct the subhypergraph. As shown in Table VI, within the explored parameter regime,

TABLE V. Surface-code performance with hyperedge selection parameters $[k_{\text{beam}}, k_{\text{hyperedge}}]$. “No Recall” and “No Search” indicate the number of times L-Recall is below 1 and an empty flipped set was not found, respectively.

Size	Param	L-Recall	No L-Recall	No Search
$d = 3$	[20, 2]	0.998 01	199	0
	[40, 4]	0.998 01	199	0
	[80, 8]	0.998 01	199	0
$d = 9$	[20, 2]	0.997 37	263	3
	[40, 4]	0.999 07	93	2
	[80, 8]	0.999 56	44	0

TABLE VI. Logical error rates of surface codes and color codes with and without degeneracy (Deg) under the HAML D decoder.

Code	d	Detectors	Rounds	LER (Deg)	LER (No Deg)
Surface	3	24	3	0.002 035	0.002 085
	5	120	5	0.000 380	0.000 390
	7	336	7	0.000 500	0.000 505
	9	720	9	0.000 595	0.000 595
Color	3	18	3	0.005 360	0.005 525
	5	90	5	0.002 365	0.002 475
	7	252	7	0.002 305	0.002 350
	9	540	9	0.005 510	0.005 715

incorporating degeneracy consistently lowers the logical error rate, demonstrating its benefit at the considered code sizes.

Notably, under the current experimental settings, considering degeneracy has little effect on the noise threshold or logical error rate for the BB code and the $d = 9$ surface code. For the BB code, this is because the optimal solutions of the MLD and MLE problems coincide for most syndromes. For the $d = 9$ surface code, the limited improvement arises mainly from insufficient hyperedge search accuracy, which prevents the decoder from reliably capturing degenerate solutions.

3. Approximation contraction

This section briefly discusses approximate contraction. When hyperedges can be selected with sufficient accuracy, the computational speedup and accuracy loss from approximate contraction are minimal. As shown in Table IV, for a $d = 3$ surface-code hypergraph with 221 hyperedges, only 5 hyperedges need to be searched without considering degeneracy, making approximate contraction nearly unnecessary.

Approximate contraction in MLD-based schemes also faces a key challenge: The error introduced by approximation is difficult to quantify, and parameter choices are often heuristic with little theoretical justification. Therefore, we argue that concentrating complexity control on hyperedge selection is more effective than optimizing contraction approximation.

Nonetheless, approximate contraction retains theoretical value. Analysis of the topology in Appendix C shows that it performs well on hypergraphs with locality and low average degree. This also suggests that approximate contraction alone is insufficient to extend MLD methods to high-degree quantum low-density parity-check (qLDPC) codes. In the special setting without data-qubit measurements discussed in Appendix B, a simple combination of hyperedge selection and approximate contraction achieves improved performance compared with MWPM and BP + OSD.

VI. CONCLUSION AND DISCUSSION

We propose HAML D, a hypergraph-based approximate maximum likelihood decoding method applicable to various quantum error-correcting codes under circuit-level noise models. We provide a demonstration that, compared with MLD-based decoders [4, 15, 17] (such as MLD and TN-MLD), HAML D can handle larger codes and a wider range of code types. Moreover, at small code sizes, HAML D achieves higher

decoding accuracy than commonly used decoders such as MWPM [21], Chromobius [35], and BP + OSD [23].

The main advantage of HAML D over TN-MLD lies in its syndrome-guided hyperedge selection, which enables it to contract only a small fraction of the hyperedges—far fewer than the total—particularly under low-noise conditions. In contrast, TN-MLD must contract the full tensor network with sufficiently large truncation parameters to ensure correctness. This difference explains HAML D's superior efficiency compared to MLD and TN-MLD. Furthermore, HAML D's explicit consideration of degeneracy provides a key advantage over the MLE decoder.

For large code distances, HAML D does not outperform existing MLE methods, mainly because the current beam-search-based hyperedge selection cannot reliably capture long error chains, leaving room for further optimization. Key open questions include how to optimize hyperedge selection for specific codes and noise models, when degeneracy is relevant, how to extend HAML D for real-time decoding, and how it performs in computational or fault-tolerant stabilizer measurement scenarios. These remain important directions for future research.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant No. 62102388) and the Innovation Program for Quantum Science and Technology (Grant No. 2021ZD0302901). This work was partially supported by USTC Kunpeng & Ascend Center of Excellence.

DATA AVAILABILITY

The data that support the findings of this article are openly available on GitHub [40].

APPENDIX A: EVALUATION SETUP

The detailed experimental setup includes the basic decoding problem configuration, evaluation metrics, compared decoding methods, and the software and hardware environments used in our simulations.

1. Basic configuration of the decoding problem

In our experiments, we consider a quantum memory decoding task, where no logical operations are applied. We focus on three widely studied quantum error-correcting codes: the surface code, the color code, and the BB code. Their syndrome measurement circuits follow established implementations: ZXXZ for the surface code [2], the scheme in Ref. [24] for the color code, and those in Refs. [7, 24] for the BB code.

The noise model follows the SI1000 (superconducting inspired, with a 1000-ns round duration) circuit-level model [31], which is inspired by superconducting hardware and assumes a 1000-ns round duration. The noise strength is controlled by a single physical error rate parameter p , with gate-specific depolarizing rates as follows:

(1) CZ or CNOT gates: two-qubit depolarizing noise with rate p .

- (2) Single-qubit Clifford gates: single-qubit depolarizing noise with rate $p/10$.
- (3) Initialization and measurement in the Z basis: $2p$ and $5p$, respectively.
- (4) Idle and resonator idle operations: $p/10$ and $2p$, respectively.

2. Metrics

To standardize the decoding task across different codes and scenarios (e.g., single- or multi-round), we use the total number of detectors as a common metric. For the surface code, there are $(d^2 - 1) \times r$ detectors and one logical observable. For the color code, the number of detectors is $\frac{3}{4}(d^2 - 1) \times r$, with one logical observable. For the BB code, there are $n \times r$ detectors and k logical observables.

Logical error rate and fault-tolerance threshold are key metrics for evaluating decoder performance [2,4]. The logical error rate can be estimated via Monte Carlo sampling, denoted as R_{sample} . By performing n_{sample} samples, we obtain n_{sample} sets of syndromes and logical observables, where the logical observable indicates a logical error: 0 for no noise and 1 for a logical Z/X error. We predict whether a logical error occurs from the syndrome and xor it with the actual logical observable. If the result is 1, a logical error remains after correction. Let m be the number of such cases. Then, the logical error rate is $R_{\text{sample}} = \frac{m}{n_{\text{sample}}}$.

Decoding speed refers to the time required to decode a single syndrome [12]. In our experiment, we decoded n_{syndrome} syndromes with a total decoding time of T . Therefore, the average decoding time per syndrome is given by $T_{\text{per syndrome}} = \frac{T}{n_{\text{syndrome}}}$.

3. Baseline decoding methods

In the experiments, we compared several commonly used decoding methods to demonstrate the advantages of the HMLD under circuit-level noise (see Table I).

MWPM [21], Chromobius [35], and BP + OSD [23] were implemented using publicly available open-source code. Except for BP + OSD, where the number of iterations is set to 100, all other parameters are kept at their default values.

Our MLD implementation follows the approach described in Ref. [16]. For TN-MLD, the implementation in Ref. [4] is not publicly available and lacks key details, making reproduction challenging. The open-source code in Ref. [17] can handle small-scale surface codes, but for larger codes, even after precompression, the run-time remains excessively long and memory requirements are prohibitive. Since this method cannot directly handle multi-logical-qubit decoding, experiments on BB codes are not feasible. Moreover, TN-MLD cannot be directly applied to the color code due to its requirement for special topological structures. The TN-MLD parameter settings are kept consistent with those described in Ref. [17].

To accurately estimate decoding fidelity, most experiments were repeated 20 times (10 for X measurements and 10 for Z measurements), with 10^5 shots sampled per run. Some experiments are highly time consuming and may take several weeks to complete—for instance, BB codes with 1080 and

1728 detectors, as well as the MLD, HMLD, and TN-MLD decoders. For these cases, we report results generated from a single run taking not more than 3 days.

To evaluate decoding speed, we considered the decoding time per syndrome. For each experiment, the total run-time was divided by the number of syndromes to obtain the average decoding time, and repeated experiments were averaged. For slower decoders—such as MLD, HMLD, and TN-MLD—100 randomly selected runs were performed for each case. TN-MLD is implemented in Julia, while all other decoding methods are implemented in C++ and accessed via Python.

4. Simulation setup

Software configuration for decoding simulation experiments. We use stim [30] to construct noisy quantum circuits. The proposed decoding method is implemented in Python 3.10 and runs on the Ubuntu 22.04.3 LTS operating system. To accelerate logical error rate testing, we use Python's open-source multiprocessing library to run parallel processing across different syndromes.

Hardware configuration for decoding simulation experiments. The platform environment adopted in this experiment is equipped with an AMD EPYC 9654 processor, which has 96 cores, 192 threads per CPU, and a main frequency of 2.4 GHz, and is equipped with 512 GB of DDR5-4800 MT/s ECC memory.

APPENDIX B: ANALYSIS OF NON-FAULT-TOLERANT SCENARIOS WITHOUT DATA-QUBIT INFORMATION

We evaluate HMLD in a setting without a final data-qubit measurement, where decoding relies solely on syndromes extracted from stabilizer measurements. This setup serves as a useful analytical tool for assessing whether an error-correction circuit can effectively suppress the noise it introduces. In practical quantum error correction, however, at least one round of noiseless or sufficiently low-noise stabilizer measurements is typically required to ensure a well-defined fault-tolerance threshold. This is because, although each round can correct noise accumulated from previous rounds, it cannot fully eliminate the noise generated by the error-correction circuit itself during execution (see Chap. 12 of Ref. [41]).

Although practical scenarios without data-qubit measurements are unclear, this setting is useful for studying the fault-tolerant implementation of stabilizer measurements. As noted in Chap. 12 of Ref. [41], commonly used stabilizer measurement schemes are not strictly fault-tolerant, making a rigorous fault-tolerance threshold difficult to define. However, Shor-, Steane-, and Knill-type schemes are theoretically fault-tolerant, as also discussed in Ref. [41].

We evaluate the performance of the HMLD method across different error-correction codes and repetition rounds, as shown in Fig. 7. The circuit construction is straightforward. Relative to standard circuit-level error-correction circuits, we modify only the final data-qubit measurements by removing measurement noise and using the outcomes solely to define logical observables rather than detectors. This effectively leaves the data qubits unmeasured, enabling a clearer evaluation of the logical error rate after the circuit and decoding. Our

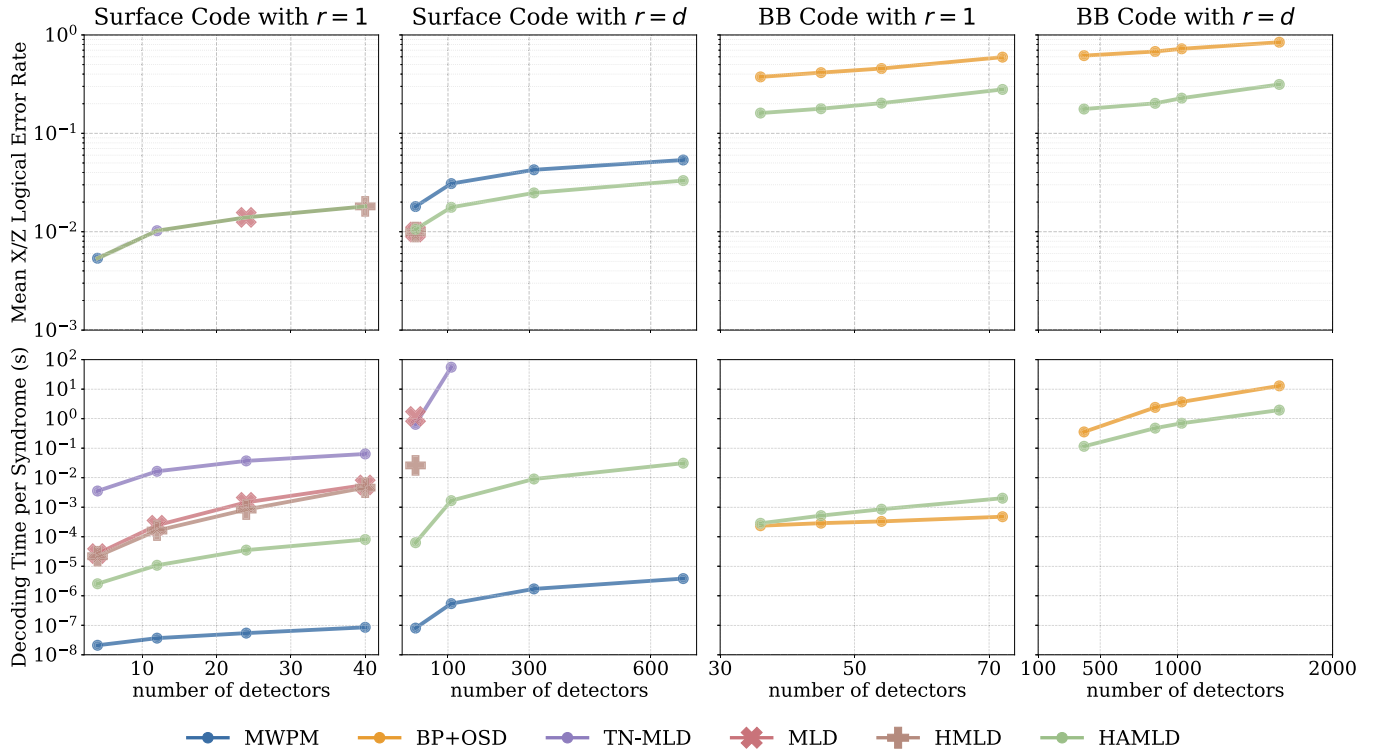


FIG. 7. Performance analysis of different decoders in non-fault-tolerant scenarios without data-qubit information. For the hyperedge selection, we adopt the simplest static, priority-based scheme. In the experimental setup, aside from the hyperedge selection method and decoding the stim circuits without measuring the data qubits, all other parameters remain identical to those specified in Appendix A.

results show that no fault-tolerance threshold appears in this setting, indicating that the circuit cannot suppress the noise generated during its own execution.

Although the above setting is not strictly fault-tolerant, it still provides a valuable benchmark for analysis. As shown in Fig. 7, the proposed decoder demonstrates significantly better performance than existing methods. Specifically, HMLD achieves lower logical error rates across multiple scenarios, including multiround surface codes, single-round BB codes, and multiround BB codes, outperforming both MWPM and BP + OSD overall. Compared with MWPM and BP + OSD, HMLD achieves average improvements of up to $1.70\times$, $2.26\times$, and $3.18\times$ across the different code types and experimental configurations.

Fault-tolerant stabilizer measurement schemes typically require the preparation of special quantum states and impose more complex and stringent constraints on qubit connectivity. These aspects are not explored in depth in this work. Evaluating the practical performance of fault-tolerant stabilizer measurement under a circuit-level noise model will be left for future research.

APPENDIX C: DECODING GRAPH TOPOLOGY: HYPERGRAPH AND CONNECTIVITY GRAPH

In hypergraph contraction, the structure of the hypergraph significantly influences the efficiency of the contraction process. In this Appendix, we introduce the hypergraph and connectivity graph representations corresponding to the DEM, constructed both from ancilla qubits alone and incorporating

data qubits. Hypergraphs are generally too complex for direct visualization; thus, we primarily conduct quantitative analyses on properties such as the number of nodes, number of hyperedges, maximum node degree, and maximum hyperedge size. In contrast, connectivity graphs offer a more tractable framework for analyzing the computational cost associated with the contraction process.

In practical quantum error-correction systems, both the maximum hyperedge size and the maximum node degree in the connectivity graph are inherently bounded. For example, in surface codes, the maximum hyperedge size is 4, whereas in BB codes, it can reach up to 9. Similarly, the maximum node degree in surface codes is 24, while it increases to 105 in a $[[144, 12, 12]]$ BB code. These bounds arise from the local nature of physical noise processes and the construction of the DEM, which records only changes in syndrome outcomes between adjacent rounds.

As detailed in Table VII, for the surface code with $r = 1$, each hyperedge connects at most two nodes, effectively reducing the hypergraph to a standard graph. When $r = d$, hyperedges can connect up to four nodes, resulting in a genuine hypergraph structure. In contrast, as shown in Table VIII, the decoding task for BB codes corresponds to a hypergraph regardless of whether $r = 1$ or $r = d$.

Figure 8 illustrates the marked differences in connectivity graphs for the surface code and BB code at $r = 1$. The surface code exhibits an approximately planar, gridlike topology, whereas the BB code demonstrates a much more complex and nonlocal structure with no apparent regularity. At $r = d$, the surface code evolves into a more intricate three-dimensional

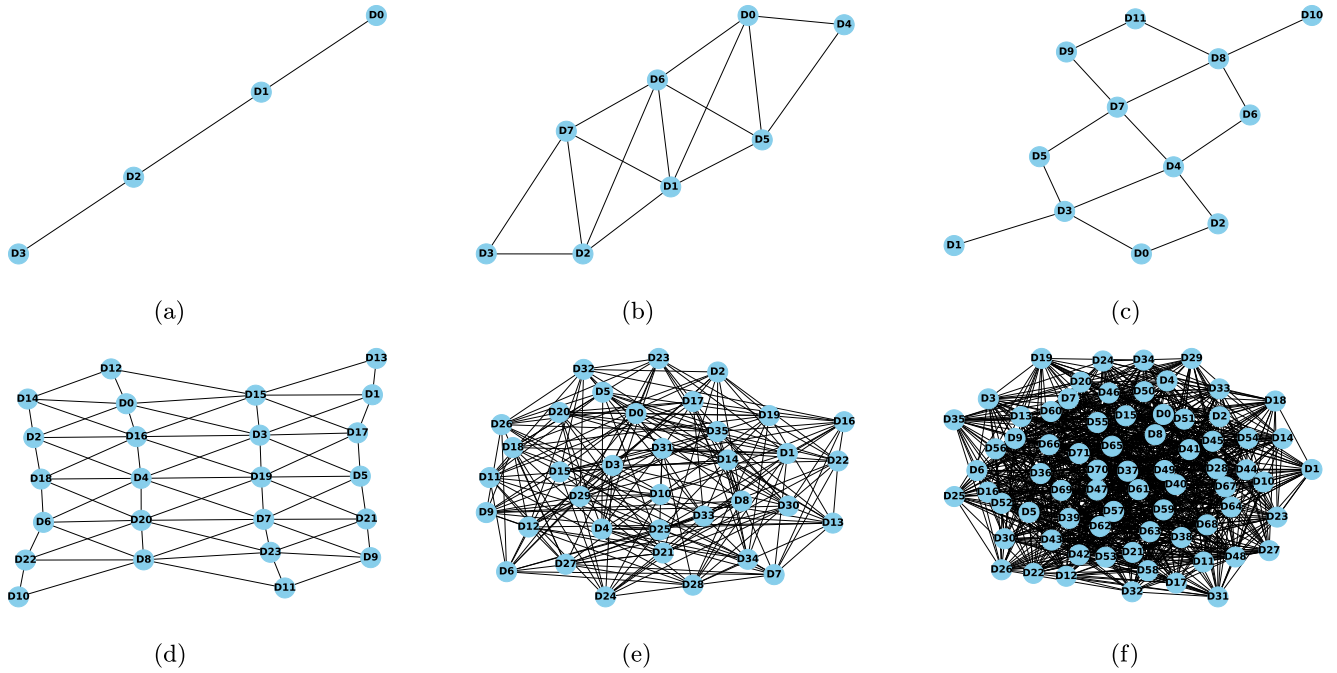


FIG. 8. The connectivity graph structures of various quantum error-correcting code configurations are presented: Panels (a)–(d) depict surface codes, while panels (e)–(f) illustrate BB codes. The terms “with data” and “no data” refer to whether measurement information from data qubits is utilized for syndrome extraction. Notably, surface codes exhibit distinctive structural features, whereas BB codes lack such clear patterns. This distinction helps explain why TN-MLD methods may perform effectively for surface codes but face challenges in BB code decoding. (a) $d = 3, r = 1$, no data; (b) $d = 3, r = 1$, with data; (c) $d = 5, r = 1$, no data; (d) $d = 5, r = 1$, with data; (e) $[72, 12, 6], r = 1$, no data; (f) $[72, 12, 6], r = 1$, with data.

TABLE VII. Hypergraph properties for surface codes. d : code distances; r : rounds; Data incl.: includes data qubits; No. of Nodes: nodes; No. of Hyperedges: hyperedges; Max size: largest hyperedge size.

d	r	Data incl.	No. of Nodes	No. of Hyperedges	Max size
3	1	✓	8	23	2
3	3	✓	24	219	4
5	1	✓	24	77	2
5	5	✓	120	1677	4
3	1		4	10	2
3	3		20	196	4
5	1		12	31	2
5	5		108	1599	4

TABLE VIII. Hypergraph properties for BB codes. $[n, k, d]$: code parameters; r : rounds; Data incl.: includes data qubits; No. of Nodes: nodes; No. of Hyperedges: hyperedges; Max size: largest hyperedge size.

$[[n, k, d]]$	r	Data incl.	No. of Nodes	No. of Hyperedges	Max size
$[72, 12, 6]$	1	✓	72	432	6
$[72, 12, 6]$	6	✓	432	16164	9
$[144, 12, 12]$	1	✓	144	864	6
$[144, 12, 12]$	12	✓	1728	67752	9
$[72, 12, 6]$	1		36	332	3
$[72, 12, 6]$	6		396	15932	9
$[144, 12, 12]$	1		72	617	3
$[144, 12, 12]$	12		1656	67366	9

architecture, while the BB code’s connectivity graph becomes even more complex and less localized.

APPENDIX D: PARALLELIZATION: SLICE-BASED PARALLELISM AND PARALLEL WINDOWS

Early studies focused on lossless acceleration using parallel tensor-network strategies [38], centering on two key ideas: designing an efficient contraction order and implementing a slice-based parallelization scheme. The effectiveness of a well-designed contraction order is demonstrated in the experimental results.

Slice-based strategies can be employed in tensor-network contraction to enable parallel computation; however, such methods generally yield significant improvements only under specific network topologies [38,39]. As shown in Fig. 8, our

TABLE IX. Comparison of contraction widths and time costs for HMLD with and without slicing. For $r = 1$, the number of slice hyperedges is 8; for $r = d$, it is 32.

r	d	Width (order/(slice + order))	Cost (order/(slice + order))
1	3	3/1	35/15
	5	6/6	823/377
	7	8/8	7927/5647
d	3	8/7	$1.96 \times 10^3 / 767$
	5	25/24	$6.98 \times 10^8 / 3.01 \times 10^8$
	7	50/48	$4.04 \times 10^{16} / 2.30 \times 10^{16}$

network structure is not suitable for efficient slicing, and slicing causes an exponential increase in the number of parallel tasks, limiting its practicality.

Table IX presents the contraction widths and costs for individual tasks under slice-based parallelization. It is important to note that the total computational cost across all parallel kernels scales exponentially with the number of slices, which poses a significant challenge to achieving effective acceleration through slicing. As a result, slicing introduces greater overall computational overhead.

Additionally, inspired by the multi-round parallel decoding strategies in Refs. [8,12], we attempted to divide an r -round decoding task into multiple subtasks and merge their results.

However, due to the sparsity of syndrome flips, the subhypergraphs generated by the approximation are relatively small, resulting in limited parallel speedup. Furthermore, the overhead associated with task initialization and result aggregation often outweighs the potential benefits. Parallelization yields slight improvements only in rare cases where a large number of flips remain after approximation, but such instances are statistically negligible.

As noise levels decrease and models become more precise, we expect that ignoring minor correlations will enable more efficient parallel decoding. Leveraging field-programmable gate arrays or other efficient hardware architectures for parallel computation is highly beneficial for real-time decoding.

-
- [1] D. Gao *et al.*, Establishing a new benchmark in quantum computational advantage with 105-qubit Zuchongzhi 3.0 processor, *Phys. Rev. Lett.* **134**, 090601 (2025).
 - [2] Google Quantum AI and Collaborators, Quantum error correction below the surface code threshold, *Nature (London)* **638**, 920 (2024).
 - [3] 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).
 - [4] R. Acharya, I. Aleiner, R. Allen, T. I. Andersen, and M. Ansmann, Suppressing quantum errors by scaling a surface code logical qubit, *Nature (London)* **614**, 676 (2023).
 - [5] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes, [arXiv:1108.5738](https://arxiv.org/abs/1108.5738).
 - [6] N. Lacroix, A. Bourassa, F. J. Heras, L. M. Zhang, J. Bausch, A. W. Senior, T. Edlich, N. Shutty, V. Sivak, A. Bengtsson, *et al.*, Scaling and logic in the color code on a superconducting quantum processor, *Nature (London)* **645**, 614 (2025).
 - [7] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, High-threshold and low-overhead fault-tolerant quantum memory, *Nature (London)* **627**, 778 (2024).
 - [8] 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).
 - [9] X. Tan, F. Zhang, R. Chao, Y. Shi, and J. Chen, Scalable surface-code decoders with parallelization in time, *PRX Quantum* **4**, 040344 (2023).
 - [10] P. Das, A. Locharla, and C. Jones, Lilliput: A lightweight low-latency lookup-table decoder for near-term quantum error correction, in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Association for Computing Machinery, New York, 2022), pp. 541–553.
 - [11] G. S. Ravi, J. M. Baker, A. Fayyazi, S. F. Lin, A. Javadi-Abhari, M. Pedram, and F. T. Chong, Better than worst-case decoding for quantum error correction, in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Association for Computing Machinery, New York, 2023), Vol. 2, pp. 88–102.
 - [12] Y. Wu and L. Zhong, Fusion blossom: Fast MWPM decoders for QEC, in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, New York, 2023), Vol. 1, pp. 928–938.
 - [13] N. Alavisamani, S. Vittal, R. Ayanzadeh, P. Das, and M. Qureshi, Promatch: Extending the reach of real-time quantum error correction with adaptive predecoding, in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Association for Computing Machinery, New York, 2024), Vol. 3, pp. 818–833.
 - [14] T. Müller, T. Alexander, M. E. Beverland, M. Bühler, B. R. Johnson, T. Maurer, and D. Vandeth, Improved belief propagation is sufficient for real-time decoding of quantum memory, [arXiv:2506.01779](https://arxiv.org/abs/2506.01779).
 - [15] S. Bravyi, M. Suchara, and A. Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, *Phys. Rev. A* **90**, 032326 (2014).
 - [16] N. Sundaresan, T. J. Yoder, Y. Kim, M. Li, and E. H. Chen, Demonstrating multi-round subsystem quantum error correction using matching and maximum likelihood decoders, *Nat. Commun.* **14**, 2852 (2023).
 - [17] C. Piveteau, C. T. Chubb, and J. M. Renes, Tensor-network decoding beyond 2D, *PRX Quantum* **5**, 040303 (2024).
 - [18] H. Cao, S. Zhao, D. Feng, Z. Shen, H. Yan, T. Su, W. Sun, H. Xu, F. Pan, H. Yu, and P. Zhang, Exact decoding of quantum error-correcting codes, *Phys. Rev. Lett.* **134**, 190603 (2025).
 - [19] P. Fuentes, J. E. Martinez, P. M. Crespo, and J. Garcia-Frías, Degeneracy and its impact on the decoding of sparse quantum codes, *IEEE Access* **9**, 89093 (2021).
 - [20] K.-Y. Kuo and C.-Y. Lai, Exploiting degeneracy in belief propagation decoding of quantum codes, *npj Quantum Inf.* **8**, 111 (2022).
 - [21] O. Higgott, Pymatching: A Python package for decoding quantum codes with minimum-weight perfect matching, *ACM Trans. Quantum Comput.* **3**, 1 (2022).
 - [22] N. Raveendran and B. Vasić, Trapping sets of quantum LDPC codes, *Quantum* **5**, 562 (2021).
 - [23] O. Higgott, stimposd: A package for decoding Stim circuits using belief propagation with ordered statistics decoding, GitHub (2024), available at: <https://github.com/oscarhiggott/stimbposd>.
 - [24] L. A. Beni, O. Higgott, and N. Shutty, Tesseract: A search-based decoder for quantum error correction, [arXiv:2503.10988](https://arxiv.org/abs/2503.10988).
 - [25] K. R. Ott, B. Hetényi, and M. E. Beverland, Decision-tree decoders for general quantum LDPC codes, [arXiv:2502.16408](https://arxiv.org/abs/2502.16408).

- [26] In recent literature, the term “syndrome measurement circuit” is most commonly used, though some works also refer to it as a “syndrome extraction circuit” or simply a QECC circuit.
- [27] A. G. Fowler, Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $\mathcal{O}(1)$ parallel time, *Quantum Info. Comput.* **15**, 145 (2015).
- [28] A. deMarti iOlius, J. E. Martinez, P. Fuentes, P. M. Crespo, and J. Garcia-Frias, Performance of surface codes in realistic quantum hardware, *Phys. Rev. A* **106**, 062428 (2022).
- [29] A. iOlius, P. Fuentes, R. Orús, P. M. Crespo, and J. E. Martinez, Decoding algorithms for surface codes, *Quantum* **8**, 1498 (2024).
- [30] C. Gidney, Stim: A fast stabilizer circuit simulator, *Quantum* **5**, 497 (2021).
- [31] C. Gidney, M. Newman, A. Fowler, and M. Broughton, A fault-tolerant honeycomb memory, *Quantum* **5**, 605 (2021).
- [32] D. Gottesman, *Stabilizer Codes and Quantum Error Correction* (California Institute of Technology, Pasadena, California, 1997).
- [33] In practical implementations of MLD, contractions are typically performed sequentially over individual detectors. Therefore, we assume $|V| = |D|$ in the following discussion.
- [34] The MLD referred to in this paper specifically denotes online MLD, as actual decoding only requires decoding based on the measured syndrome, without the need to compute decoding results for all possible syndromes.
- [35] C. Gidney and C. Jones, New circuits and an open source decoder for the color code, [arXiv:2312.08813](https://arxiv.org/abs/2312.08813).
- [36] J. Roffe, D. R. White, S. Burton, and E. Campbell, Decoding across the quantum low-density parity-check code landscape, *Phys. Rev. Res.* **2**, 043423 (2020).
- [37] J. C. Bridgeman and C. T. Chubb, Hand-waving and interpretive dance: An introductory course on tensor networks, *J. Phys. A: Math. Theor.* **50**, 223001 (2017).
- [38] C. Huang, F. Zhang, M. Newman, X. Ni, and D. Ding, Efficient parallelization of tensor network contraction for simulating quantum computation, *Nat. Comput. Sci.* **1**, 578 (2021).
- [39] J. Gray and S. Kourtis, Hyper-optimized tensor network contraction, *Quantum* **5**, 410 (2021).
- [40] K. Chen, HAMLID (hypergraph-based approximate maximum likelihood decoding), GitHub (2025), available at: <https://github.com/kechen666/hamld>.
- [41] D. Gottesman, Surviving as a Quantum Computer in a Classical World (2024), <https://www.cs.umd.edu/class/spring2024/cmssc858G/>.