Trapping Sets of Quantum LDPC Codes

Nithin Raveendran*, and Bane Vasić†
Department of Electrical and Computer Engineering
Center for Quantum Networks
University of Arizona, Tucson, AZ 85721
*nithin@email.arizona.edu, †vasic@ece.arizona.edu

***Abstract*—Iterative decoders for finite length quantum low-density parity-check (QLDPC) codes are attractive because their hardware complexity scales only linearly with the number of physical qubits. However, they are impacted by short cycles, detrimental graphical configurations known as trapping sets (TSs) present in a code graph as well as symmetric degeneracy of errors. These factors significantly degrade the decoder decoding probability performance, and cause so-called error floor. In this paper, we establish a systematic methodology by which one can identify and classify quantum trapping sets (QTSs) according to their topological structure and decoder used. Conventional definition of a TS from classical error correction is generalized to address the syndrome decoding scenario for QLDPC codes. We show that the knowledge of QTSs can be used to design better QLDPC code and decoder. Frame error rate improvements of two orders of magnitude in the error floor regime are demonstrated for some practical finite-length QLDPC codes without requiring any post-processing.**

I. Introduction

Quantum low-density parity check (QLDPC) codes are an important class of quantum error correction (QEC) [1], [2] codes that can realize scalable fault-tolerant quantum computers (FTQCs) with a finite multiplicative overhead [3]. In addition, they have finite asymptotic rates with non-zero fault-tolerant thresholds [4], and support low-complexity iterative decoding. The existing QLDPC code literature primarily focuses on constructing asymptotically good code families with improved minimum distance scaling with the block length and higher code rates, as well as on designing better iterative decoding algorithms [5]–[8]. However, QLDPC codes implemented in practical QEC systems will be of finite length, and will exhibit performance degradation due to failure of iterative decoders to converge to a correct error pattern. This phenomenon specific to finite-length codes is well understood in classical literature, but its analysis and precise mathematical characterization is completely missing in the QEC literature. The convergence failure manifests itself as an *error floor* of the decoding probability of error [9] at low physical error rate levels – an operating regime for large scale FTQCs – and is observed in all state-of-the-art iterative message-passing decoders for QLDPC codes such as belief propagation (BP), min-sum algorithm (MSA) and their variants [10], [11].

A typical approach in QEC literature to reduce the error floor of the above decoding algorithms is to couple them with *ordered statistics decoding (OSD)* and post-processing [10], [12]. However, although exhibiting good performance, this technique is too complex to implement in hardware due to the high complexity of the OSD algorithm [13] which scales cubically with the code dimension (see Eq. 4 in [14]). In contrast, philosophy of our approach and our ultimate goal is to develop message-passing decoders for QLDPC codes which do not require a post-processing step to achieve strong error correction.

Iterative message-passing decoder operates on a Tanner graph which is the graphical representation of a parity check matrix of the underlying code. Error floor is attributed to the presence of specific topologies of sub-graphs in the Tanner graph generically referred to as *trapping sets* (TSs) that are detrimental to iterative decoders. Since a trapping set depends both on the topology of the sub-graph and on the decoder, one must understand key differences of QEC, specifically QLDPC codes and decoding with respect to classical error correction.

The first difference comes as the fact that the stabilizer commutativity/symplectic inner product (SIP) requirement for the parity check matrices introduces additional code construction constraints resulting in unavoidable cycles in the Tanner graph. Furthermore, QLDPC codes are known to be highly degenerate, i.e., their minimum distance is higher than the weight of their stabilizers. From the decoder perspective, this implies that the decoders need to account for degenerate errors which has no equivalent in classical error correction. However, iterative algorithms based on BP are sub-optimal in the presence of cycles and, also, are not capable of correcting all degenerate errors [15], [16]. Another key difference from classical LDPC decoding stems from the inability to directly measure qubits for error correction. Hence, iterative message-passing algorithms used for decoding of QLDPC codes are modified to only make use of the syndrome information to infer the error introduced by the channel. How the classical trapping sets definition accommodates a syndrome-based decoder is not clearly understood. As we show, degenerate errors having no classical analogy introduces new failure configurations unique to the QLDPC codes. The approach presented in this paper accounts for these key differences and their implications.

Failure configurations of QLDPC codes are relatively unknown when compared to the classical trapping set research. One major drawback of BP as pointed out in [17] is that the decoding ability of BP is typically limited by the row weight of the parity-check matrix due to the SIP constraint and identifies pseudo-codeword structures for cycle codes. However, generalization from cycle codes to QLDPC codes is non-trivial.

In this paper, we define quantum trapping sets (QTSs) by investigating into failure configurations for syndrome based iterative message passing algorithms. The quantum trapping set formulation is modified to the syndrome decoding scenario for QLDPC codes considering Pauli X and Z errors separately. We identify QTSs of prominent QLDPC codes and show that the QTSs must be analyzed in conjunction with the particular iterative decoder used along with their location in the Tanner graph [18]. Message update rules and scheduling strategies are also identified that help to decoder escape from such trapping sets improving the error floor performance.

The rest of this paper is organized as follows. In Section II, we introduce QLDPC codes using the stabilizer formalism reviewing some basic notations and then discuss the syndrome decoding problem and classical trapping sets. In Section III, we analyze the different failure configurations, relation between trapping sets and decoder/error correction properties. We also formally define quantum trapping sets and describe the methodology used to identify those specifically for Calderbank, Shor, Steane (CSS) codes [19]. Trapping sets of some classes of CSS codes are analyzed in Section IV. Based on these analyses, we present simulation results which briefly explore two strategies of code and decoder improvement. We explore CSS code constructions without some of the harmful configurations and also compare performance of trapping set-aware decoding strategies in Section V followed by concluding remarks and future research directions in Section VI.

II. Preliminaries

A. Stabilizer Formalism

Stabilizer codes, the quantum analog of classical linear codes, are the most common type of QEC codes considered in both theory and practice. An $[\![n, k, d]\!]$ quantum stabilizer code maps k qubit quantum state $|\phi\rangle$ to an entangled n-qubit codeword $|\psi\rangle$ (a unit vector in the 2^n-dimensional Hilbert space) and is defined as a 2^k-dimensional subspace of the Hilbert space which is a common $+1$ eigenspace of the stabilizer group $\mathcal{S}$. The n-qubit codeword $|\psi\rangle$ is stabilized by all stabilizer elements in $\mathcal{S}$. i.e., $s_j |\psi\rangle = +|\psi\rangle$ for any $s_j \in \mathcal{S}$. We denote a generator set of a given stabilizer group $\mathcal{S}$ by the set $S = \{s_1, s_2, \ldots, s_m\}$. The stabilizer generators form the $m = n - k$ rows of the corresponding stabilizer matrix H_p whose entries are the single-qubit Pauli matrices $\mathrm{I}_2 = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right], \mathrm{X} = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right], \mathrm{Z} = \left[\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right]$, and $\mathrm{Y} = \imath\mathrm{XZ} = \left[\begin{smallmatrix} 0 & -\imath \\ \imath & 0 \end{smallmatrix}\right]$. Kronecker products of n single-qubit Paulis and scalars $\imath^\kappa$, where $\kappa \in \mathbb{Z}_4 = \{0, 1, 2, 3\}$ forms the n-qubit Pauli group $\mathcal{P}_n$, of which the stabilizer group $\mathcal{S}$ is a commutative subgroup that contains only Hermitian Paulis and excludes $-I_n$. The *weight* $w(P)$ of a Pauli operator $P \in \mathcal{P}_n$ is the number of qubits on which it applies a non-identity Pauli matrix. A QLDPC code is a stabilizer code with all stabilizer generators having low weight. Analogous to the classical minimum distance, $[\![n, k, d]\!]$ code have logical operators $L \in \mathcal{P}_n \backslash \mathcal{S}$ that commutes with all s_i having minimum weight d. Like the codeword generators, the logical operators of the code map an n-qubit codeword to another. Logical group $\mathcal{L}$ is generated by k logical X generators: $L_\mathrm{X} = \{lx_1, lx_2, \ldots, lx_k\}$ and k logical Z generators: $L_\mathrm{Z} = \{lz_1, lz_2, \ldots, lz_k\}$ obtained by using either Gottesman's [20] or Wilde's algorithm [21].

The stabilizers commute with each other following the commutativity relation between two n-qubit Pauli operators P and Q defined as follows:

$$P \circ Q := \prod_{j=1}^{n} P_j \circ Q_j, \text{ where } P_j \circ Q_j = \pm 1 \text{ if } P_j Q_j = \pm P_j Q_j.$$

P and Q commutes if $P \circ Q = +1$ and anti-commutes if $P \circ Q = -1$. Every logical generators commute with the stabilizers and Lx_i commutes with every other generators except with $Lz_i \ \forall i \in \{1, k\}$.

B. Stabilizers as binary parity checks

An alternative binary representation maps Pauli matrices to binary tuples as follows: $\mathrm{I}_2 \to (0,0),\ \mathrm{X} \to (1,0),\ \mathrm{Z} \to (0,1),\ \mathrm{Y} \to (1,1)$. More generally, binary representation of an n-qubit Pauli operator P will be a binary vector of length $2n$ of the form $\boldsymbol{p} = (\boldsymbol{p_x}, \boldsymbol{p_z})$, where $\boldsymbol{p_x}$ and $\boldsymbol{p_z}$ are of length n each with ones at positions of X- and Z-Pauli components respectively. Such a mapping aids in construction of quantum stabilizer codes using extensive classical coding literature. The binary representation H_b of the stabilizer matrix of dimension $m \times 2n$ given by

$$H_b = \left[H_\mathrm{X} \mid H_\mathrm{Z}\right], \tag{1}$$

where H_X and H_Z represent binary parity check matrices used for error correction. Each row in H_b denotes a stabilizer generator, and a pair of corresponding columns in H_X and H_Z represent a qubit. Equivalent to the commutativity relation defined for Pauli operators, the stabilizer generators commute with each other based on the *symplectic inner product* (SIP) in their binary representation [22]. Any two rows $\boldsymbol{p} = (\boldsymbol{p_x}, \boldsymbol{p_z})$ and $\boldsymbol{q} = (\boldsymbol{q_x}, \boldsymbol{q_z})$ of $\left[H_\mathrm{X} \mid H_\mathrm{Z}\right]$ must satisfy $\boldsymbol{p} \odot \boldsymbol{q} := \mod(\boldsymbol{p_x}\boldsymbol{q_z}^T + \boldsymbol{p_z}\boldsymbol{b_x}^T, 2) = 0$. This leads to the condition

$$H_\mathrm{X} H_\mathrm{Z}^T + H_\mathrm{Z} H_\mathrm{X}^T = 0, \tag{2}$$

where the right hand side (0) is an $m \times m$ zero matrix, T denotes the transpose of a matrix and operations (addition and multiplication) are done modulo-2. We will refer to Eq. (2) as the SIP constraint.

C. Decoding Problem

For binary decoding, typically considered channel model of a depolarizing channel is isomorphic to two independent binary symmetric channels (BSCs), a simplified model if the correlation between bit and phase flip error is ignored. The BSCs for X and Z errors have a cross-over probability of $2p/3$ [1], decoded using H_Z and H_X, respectively. Let $\boldsymbol{e} = (\boldsymbol{e_x}, \boldsymbol{e_z})$ be the binary representation of a Pauli error acting on the n qubits. The corresponding syndrome is computed as

$$\begin{aligned} \boldsymbol{\sigma} &= [\boldsymbol{\sigma_x},\ \boldsymbol{\sigma_z}] \\ &= [\mod(H_\mathrm{Z}.\boldsymbol{e_x}^\mathrm{T}, 2), \mod(H_\mathrm{X}.\boldsymbol{e_z}^\mathrm{T}, 2)]. \end{aligned}$$

All-zero syndrome ($\boldsymbol{\sigma} = \bar{0}$) indicates that all the stabilizers commute with the error pattern (undetectable error), whereas

non-zero entries/ones in $\boldsymbol{\sigma}$ indicate that some stabilizer generators anti-commute with the error pattern (detectable error). A syndrome based decoder's task is to estimate the error pattern $\hat{\boldsymbol{e}}$ whose syndrome $\hat{\boldsymbol{\sigma}}$ matches with the initial input syndrome $\boldsymbol{\sigma}$. If $\hat{\boldsymbol{\sigma}} = \boldsymbol{\sigma}$, the estimated error pattern $\hat{\boldsymbol{e}}$ is applied to reverse the error $\boldsymbol{e}$ introduced by the channel. Error correction process is successful if $\hat{\boldsymbol{e}} = \boldsymbol{e} \oplus h$, where $h \in \text{rowspace}(H_b)$, i.e., if the code word is recovered up to a stabilizer ($\hat{\boldsymbol{e}} \oplus \boldsymbol{e}$ is a stabilizer, where $\oplus$ denotes pairwise XOR). Error correction fails when the decoder is unable to find an error pattern that matches the syndrome $\boldsymbol{\sigma}$ or when the decoding process results in a logical or miss-correction error. A logical error occurs if $\hat{\boldsymbol{e}} \oplus \boldsymbol{e}$ is a logical operator such that post error correction state is a codeword different from the original codeword. We can detect a logical error if Pauli representation of $\hat{\boldsymbol{e}} \oplus \boldsymbol{e}$ anti-commutes with any of the $2k$ logical generators.

D. Iterative Decoding of CSS codes

Although the syndrome decoding paradigm is applicable to any class of quantum codes, trapping set analysis in this paper is focused on QLDPC families: hypergraph product (HP) codes [7], bicycle codes [1] and generalized bicycle codes [10] representing the CSS class of codes [19]. An attractive property of CSS codes constructed from two classical codes $\mathcal{C}_1$ and $\mathcal{C}_2$, where $\mathcal{C}_2^{\perp} \subseteq \mathcal{C}_1$ is that the parity check matrix can be written in a separable form: $H_b = \begin{bmatrix} H_{\text{X}} & 0 \\ 0 & H_{\text{Z}} \end{bmatrix}$. CSS-QLDPC codes have a sparse matrix H_b with the SIP constraint: $H_{\text{Z}}.H_{\text{X}}^{\text{T}} = 0$.

We can perform error correction for the X and Z errors separately using H_{Z} and H_{X} matrices, respectively. The corresponding input syndromes are obtained as $\boldsymbol{\sigma_x} = \mod(H_{\text{Z}}.\boldsymbol{e_x}^T, 2)$ and $\boldsymbol{\sigma_z} = \mod(H_{\text{X}}.\boldsymbol{e_z}^T, 2)$, respectively. For simplicity going forward, we use H, L and $\boldsymbol{\sigma}$, $\boldsymbol{e}$ for the parity check matrix, logical generator matrix, input syndrome and channel error vector, respectively.

The stabilizer generator matrix/parity check matrix H is the bi-adjacency matrix of a bipartite Tanner graph $G = (V \cup C, E)$, where V represents the set of n qubit/variable nodes (VNs), C is the set of m stabilizer generators/check nodes (CNs) and E is the set of edges between them. CN $c_i \in C$ and VN $v_j \in V$ are neighbors if there is an edge $(v_j, c_i) \in E$ between the nodes, corresponding to the non-zero entry in the parity check matrix $H_{c_i,v_j} = 1$. Diagrammatically, Tanner graphs are drawn with circles representing VNs, squares representing CNs and solid-lines representing the edges. Let us denote the set of CNs connected to a VN v_j by $\mathcal{N}(v_j)$, and $|\mathcal{N}(v_j)|$, where $|\cdot|$ denotes cardinality, is referred to as the degree of the VN v_j. Similarly, we can define the neighbor set and the degree of a CN c_i as $\mathcal{N}(c_i)$ and $|\mathcal{N}(c_i)|$, respectively. A (γ, ρ) QLDPC code have a sparse stabilizer matrix with the variable and stabilizer degree upper-bounded by γ and ρ respectively. For a subset of VNs, say $K \subseteq V$, $\mathcal{N}(K)$ denotes the set of CN neighbors. The induced sub-graph $\mathcal{G}(K)$ is the graph containing the nodes $K \cup \mathcal{N}(K)$ along with the edges $\{(x, y) \in F : x \in K, y \in \mathcal{N}(K)\}$. The girth, g, of the Tanner graph G is the length of the shortest cycle in G. Denote the number of cycles of length g, $g+2, \ldots$ by χ_g, $\chi_{g+2}, \ldots$, respectively. If G has χ_g, $\chi_{g+2}, \ldots$ cycles of length $g, g+2, \ldots$, then the cycle enumerator series $\text{CYC}(x) = \sum_{r \geq 0} \chi_r x^r$ defines the cycle profile of G.

The goal of a syndrome based iterative decoder $\mathcal{D}_s$ is to output an error pattern that matches the input syndrome. This is different from the traditional iterative decoder $\mathcal{D}$ that uses the channel information as initial likelihoods to recover the codeword matching to an all-zero syndrome. Starting from an input syndrome $\boldsymbol{\sigma}$ and an all-zero error vector estimate, $\mathcal{D}_s$ performs a finite number ℓ_{max} of iterations of decoding over the Tanner graph. The messages are passed over the edges of the Tanner graph from check nodes to their neighboring variable nodes and vice-versa at every iteration of message passing decoding. Decoder update rules and message alphabet size can be of varying complexity ranging from the simplest binary message passing algorithms such as Gallager-B [23] to finite alphabet iterative decoders [24], and MSA or BP using floating point messages [1]. Also, schedule of message passing in $\mathcal{D}_s$ can be implemented with a flooding/parallel schedule or a layered/serial schedule. Trapping set analysis presented here is applicable for all such decoder implementations. We discuss a generic syndrome based iterative decoder in Appendix A for completeness. Based on the update rules, $\mathcal{D}_s$ outputs an error vector estimate $\hat{\boldsymbol{e}}^{(\ell)} = (\hat{e}_1^{(\ell)}, \hat{e}_2^{(\ell)}, \ldots, \hat{e}_n^{(\ell)})$ and corresponding output syndrome $\hat{\boldsymbol{\sigma}}^{(\ell)} = (\hat{\sigma}_1^{(\ell)}, \hat{\sigma}_2^{(\ell)}, \ldots, \hat{\sigma}_m^{(\ell)})$. We refer to $\hat{e}_j^{(\ell)}$/$\hat{\sigma}_i^{(\ell)}$ as the value of the variable/check node v_j/c_i at iteration $\ell \leq \ell_{max}$. We conclude that $\mathcal{D}_s$ is successful if the output syndrome $\hat{\boldsymbol{\sigma}}^{(\ell)}$ is equal to the input syndrome $\boldsymbol{\sigma}$ (we also say that syndromes are *matched*). Then, the n-length error pattern $\hat{\boldsymbol{e}}^{(\ell)}$ is decided as the most likely error pattern. The iterative procedure is halted if successfully matched or if ℓ_{max} number of iterations is reached. At the end of iterative decoding, syndrome decoding process is successful if the syndromes are matched. Otherwise, the decoding is said to have failed.

E. Classical Trapping Sets

A classical trapping set example is shown in Fig. 1 illustrating failure of a classical iterative decoder $\mathcal{D}$ on a small subgraph inside the Tanner graph. Let us consider a simple binary message passing decoder - Gallager-B decoder which performs XOR operation at the check nodes and a majority voting at the variable nodes. More precisely, the outgoing check node message over an edge is computed as the XOR of extrinsic (all incoming messages except the edge for the message is updated) variable node messages. The outgoing variable node message is the majority value among incoming extrinsic check node messages and the channel value. The messages passed over corresponding edges are marked next to the directed arrows in the figure. All-zero transmitted codeword has errors only on three variable nodes (v_2, v_4, v_5 - shaded circles ●) as shown in Fig. 1. The decoder is unable to converge to the all-zero codeword. In fact, the decoder oscillates from error pattern (v_2, v_4, v_5) to (v_1, v_3) and back as its output during the decoding iterations, thus failing to converge.

A classical iterative decoder is said to converge correctly if the decoder output word for any $\ell \leq \ell_{max}$ matches to the

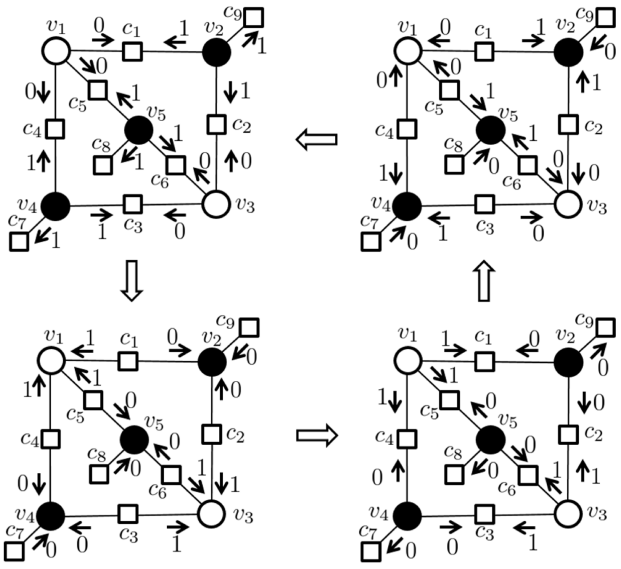
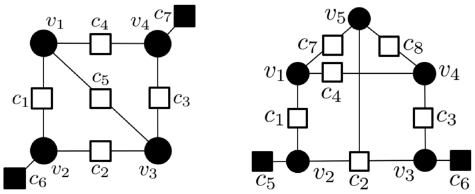


Fig. 1. An illustration of a failure configuration of regular Gallager-B decoder, unable to converge to the all-zero codeword when the input error pattern is a specific weight-three error pattern (v_2, v_4, v_5) (shaded circles ●) among the five VNs in the sub-graph. Figures are marked with the binary messages (next to the arrows indicating the direction of the messages passed) corresponding to the check/variable updates. Upper left figure corresponds to the variable node update at the zero-th iteration. The subsequent CN and VN updates of the decoding process are indicated by the connecting arrows. We assume that the rest of the Tanner graph is correct.



(a) (4,2) TS (b) (5,3) TS

Fig. 2. Graphical (Tanner graph) representations of a $(4, 2)$ TS and a $(5, 3)$ TS. Odd degree/unsatisfied checks are shown using black squares.

transmitted codeword and fail to converge correctly otherwise. A variable node v_j is *eventually correct* if there exists a positive integer I_j such that for all iterations $\ell \geq I_{\mathrm{j}}$, the decoder's estimate of v_j is equal to the transmitted bit value. Then, trapping set is defined as

Definition 1 ([11]): A trapping set $\mathcal{T}$ for an iterative decoder $\mathcal{D}$ is a non-empty set of variable nodes in a Tanner graph G that are not eventually correct. If the sub-graph $\mathcal{G}(\mathcal{T})$ induced by such a set of variable nodes has a VNs and b odd degree CNs, then the trapping set $\mathcal{T}$ is conventionally labeled as an (a, b) trapping set.

Fig. 2 shows examples of TS induced sub-graphs observed in classical LDPC codes. Even at low physical error rate levels, the presence of such small sub-graphs can result in decoding failures resulting in the characteristic error floors in their decoding performance (frame error rate (FER) vs. physical error rate) curves.

Harmfulness of a TS is also closely linked with the decoder through their critical number μ and strength s defined as follows:

Definition 2: Critical number μ of a trapping set $\mathcal{T}$ is the minimal number of variable nodes that have to be initially in error for the decoder to fail to converge.
Let failure inducing set be the set of variable nodes that have to be initially in error for the decoder to fail to converge.

Definition 3: Strength s of a trapping set $\mathcal{T}$ is the number of failure inducing sets of cardinality μ.
Two important assumptions are used in the definition of the critical number and strength of a TS. First one is that the minimum distance of the code is large compared to the size of the TS. The second is an *isolation assumption* [24] which ensures that messages from outside the TS are correct for the TS failure analysis. For example, from Fig. 1, the critical number μ for the $(5, 3)$ TS with Gallager-B decoder is 3 and the number of weight-μ error patterns that fail is $\mathrm{s} = 1$. Note that the decoder also fails to correct weight-4 error patterns and the weight-5 error pattern in the TS. Error floor of the decoder is dominated by the minimum critical number $\mu_{\min}$ and the number of weight-$\mu_{\min}$ failure inducing error patterns. Analytical/semi analytical estimation of error floors of QLDPC codes using critical number and strength of TSs is beyond the scope of this paper and we refer the reader to classical LDPC literature [11], [18].

III. Quantum Trapping Sets

As our focus is on the error floor regime, we are interested in error patterns with small weight, well below the maximum likelihood (ML) error correction capability of the QLDPC code for which the syndrome-based iterative decoder $\mathcal{D}_s$ fails to converge. Such low-weight error patterns are either part of a classical-type TS or a symmetric stabilizer, defined as a quantum trapping set (QTS).

A. Definition of a Quantum Trapping Set

After pre-defined number of iterations, ℓ_{max}, of iterative syndrome decoding, we declare that the decoder $\mathcal{D}_s$ failed for a particular input syndrome/error pattern if the decoder is not able to find an error pattern with a syndrome equal to the input syndrome. More precisely, a decoder failure is said to have occurred if there does not exist $\ell \leq \ell_{max}$ such that $\mathrm{supp}(\hat{\boldsymbol{\sigma}}^{(\ell)} + \boldsymbol{\sigma}) = \emptyset$, where supp denotes the support set (indices of non-zero elements). During iterative decoding, a check node c_i is *eventually satisfied* if there exists a positive integer I_{i} such that for all $\ell \geq I_{\mathrm{i}}$, $\hat{\sigma}_i^{(\ell)} = \sigma_i$. We say that the variable node v_i has *eventually converged* if there exists a positive integer I_{i} such that for all $\ell \geq I_{\mathrm{i}}$, $\hat{e}_i^{(\ell)} = \hat{e}_i^{(\ell-1)}$. Note that the $\hat{e}_i^{(\ell)}$ is not necessarily the correct estimate of error on the i^{th}-variable node. With these definitions, we define quantum TSs as follows:

Definition 4: A trapping set $\mathcal{T}_s$ for a syndrome-based iterative decoder $\mathcal{D}_s$ is a non-empty set of variable nodes in a Tanner graph G that are not eventually converged or are neighbors of the check nodes that are not eventually satisfied

Remark 1: If the sub-graph $\mathcal{G}(\mathcal{T}_s)$ induced by such a set of variable nodes has a VNs and b unsatisfied CNs, then the trapping set $\mathcal{T}_s$ is conventionally labeled as an (a, b) trapping set.

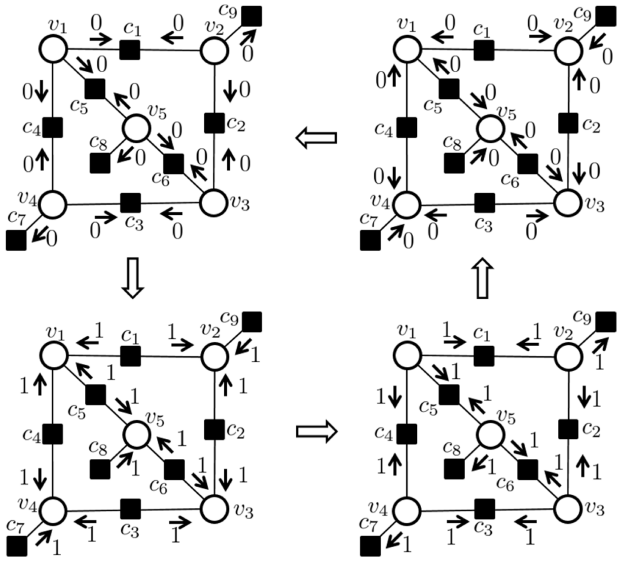
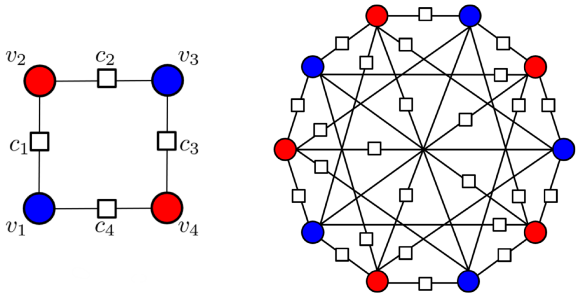


Fig. 3. An illustration of a failure configuration of syndrome-based Gallager-B decoder [23]. The shaded squares ■ represent the anti-commuting stabilizers/checks. Syndrome iterative decoder starts message passing with an all zero error pattern trying to find the true error pattern that matches with the all-one syndrome but is unable to converge successfully. The messages passed within the sub-graph in consecutive iterations oscillates showing that the decoder is trapped.



(a) (b)

Fig. 4. The Tanner graph representations of the $(4,0)$ and $(10,0)$ symmetric stabilizers with ● and ● representing the disjoint sets of variable nodes of the stabilizer.

The QTSs similar to the TSs in classical LDPC codes have exactly the same definition as the first criterion, and we refer to them as *classical-type* trapping sets. The second class of trapping sets are specifically the harmful degenerate errors observed within the stabilizers classified as *symmetric stabilizer trapping sets*. We will see that in such trapping sets, even though the variable nodes eventually converge to some error pattern, there exist check nodes that are not eventually satisfied. The definitions and assumptions for critical number and strength of the QTS remain the same as for the classical trapping set.

In the next two paragraphs we give examples of these two classes of trapping sets. We assume that $\mathcal{D}_s$ is the well-known Gallager-B decoding algorithm. This assumption is made mostly for pedagogical reasons, but also because some trapping sets of Gallager-B are also trapping sets of other decoders such as BP or MSA.

1) Classical-type trapping set: We will first show in an illustration why classical-type TSs as shown in Fig. 2 are also failure configurations of syndrome decoders. The same error pattern of the (5,3) TS given in Fig. 1 is redrawn for a syndrome based Gallager-B decoder in Fig. 3. The syndrome input is all-one vector, indicated by the black squares and the decoder starts from the all-zero error pattern (no error). The outgoing check node message over an edge is computed as the XOR of extrinsic variable node messages and the syndrome input value at the check node, and the variable node message computation remains the same as in regular Gallager-B decoder. The messages passed over corresponding edges are marked next to the directed arrows. Note that even though the messages passed in Fig. 3 are different from that in Fig. 1 of the regular Gallager-B decoder, the syndrome decoder is also unable to converge, and its output oscillates from the all-zero error pattern to errors in v_2, v_4, and v_5 and back. Hence, the $(5,3)$ TS in Fig. 3 is classified as a QTS for the syndrome decoder as well.

In addition to classical-type TSs, iterative decoders on QLDPC codes fail for specific degenerate errors. Our quantum trapping set Definition 4 captures such failure configurations as well. This distinctive difference from classical codes deserves further analysis in the next section.

2) Symmetric stabilizer trapping set: Recall the quantum decoding problem in Section II-C, wherein the decoder needs to identify any recovery operator such that $\hat{\boldsymbol{e}} \oplus \boldsymbol{e} = \text{rowspace}(H_b)$. This is in contrast to the classical decoding problem where an exact match of error $\hat{\boldsymbol{e}} = \boldsymbol{e}$ is required. In quantum decoding, we say error vectors $\boldsymbol{e}$ and $\boldsymbol{f}$ are *degenerate errors* if $\boldsymbol{e} \oplus \boldsymbol{f}$ is a stabilizer, which makes it equivalent to output any one of the degenerate errors as the candidate error pattern for matching the syndrome. However, in QLDPC codes whose minimum distance is higher than their stabilizer weight, some degenerate errors can be detrimental to iterative decoding. A symmetric topology of the stabilizer sub-graph that contains degenerate error patterns $\boldsymbol{e}$ and $\boldsymbol{f}$ of equal weight will result in a decoding failure. We will see more examples of such decoder failure when the iterative decoder attempts to converge to error patterns $\boldsymbol{e}$ and $\boldsymbol{f}$ simultaneously, thus not matching the input syndrome. This failure can be attributed to the symmetry of the both the stabilizer and the decoder message update rules. Hence, such errors are referred to as symmetric degenerate errors and corresponding sets of variable nodes as symmetric stabilizer trapping sets or just symmetric stabilizers, for short. Although degenerate errors are typically classified as harmless for quantum decoding, from the above discussion it follows that some (not all) degenerate error patterns in a symmetric stabilizer are harmful for iterative decoders.

Definition 5: A symmetric stabilizer is a stabilizer with the set of variable/qubit nodes, whose induced sub-graph has no odd-degree check nodes, and that can be partitioned into an even number of disjoint subsets, so that: (a) sub-graphs induced by these subsets of variable nodes are isomorphic, and (b) each subset has the same set of odd degree check node neighbors in its induced sub-graph.

Example 1: Consider the Fig. 4(b) with the stabilizer sub-

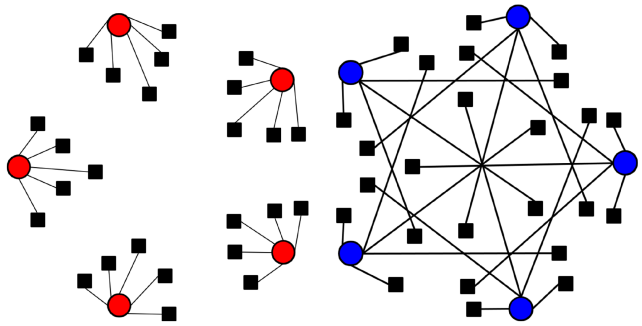


Fig. 5. The induced sub-graphs from ● and ● variable nodes of a $(10, 0)$ symmetric stabilizer trapping set. The sub-graphs 5(a) and 5(b) are isomorphic and have the same odd-degree checks represented using dark squares ■.

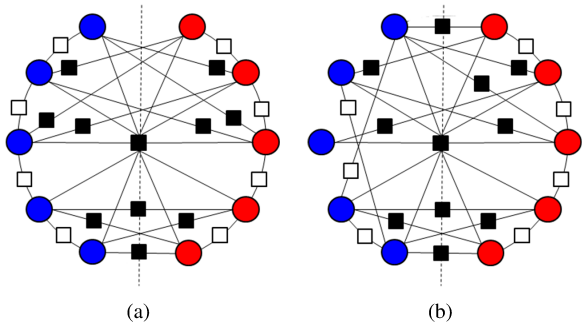


Fig. 6. Degenerate errors $\boldsymbol{e}$ and $\boldsymbol{f}$ located on ● and ● variable nodes, respectively in the symmetric stabilizer in 6(a) result in an iterative decoder failure. Introducing asymmetry during the QLDPC code design can lead to decoder success taking advantage of degeneracy of the QLDPC codes. As an example, for the sub-graph in 6(b), a BP decoder is able to match to the syndrome (dark squares represent unsatisfied checks) correctly with the red error pattern.

graph induced by ten variable nodes that are partitioned into two disjoint sets with the coloring ● and ●. The induced sub-graphs from ● and ● variable nodes are shown in the Fig. 5. The sub-graphs in Fig. 5(a) and Fig. 5(b) are isomorphic and have the same odd-degree checks represented using dark squares ■. Hence, the stabilizer shown in Fig. 4(b) satisfies the definition of a symmetric stabilizer.

Remark 2: The symmetric stabilizer shown in Fig. 4 (b) is present in generalized bicycle codes given in [10]. Surface codes are highly degenerate, and symmetric stabilizers, for example as shown in Fig. 4(a), are ubiquitous in them.

Now, we discuss how degenerate errors within the symmetric stabilizer are harmful for iterative decoders. As a non-trivial example of a symmetric degenerate error, let the error pattern $\boldsymbol{e}$ be located on the ● variable nodes in Fig. 6(a). They result in unsatisfied check shown as ■. Note, however, that the sub-graph is symmetric with respect to the vertical axis, and therefore each erroneous node has a ● twin. The set of all ● twins form an alternative error pattern $\boldsymbol{f}$. The existing iterative decoders fail as they simultaneously attempt to converge to both these error patterns. It is not difficult to see that such "ambiguity" happens for all decoders for which: (a) the check and message update rules are symmetric functions in incoming messages, and (b) in the same iteration all variable/check nodes in the graph apply in parallel the same variable/check update function, respectively. For example, during the iterations of the Gallager-B decoder, every unsatisfied CN ■ sends binary message, one back to the VNs. Because of the symmetry, the VNs in both $\boldsymbol{e}$ and $\boldsymbol{f}$ receive exactly the same messages, thus converging to $\boldsymbol{e} \oplus \boldsymbol{f}$, the symmetric stabilizer.

Based on the Definition 4, the set of VNs involved in the symmetric stabilizer form a QTS and the sub-graph in Fig. 6 (a) is a (10,0) TS by convention. We can prove as in the following lemma pertaining to the general case.

Lemma 1: A symmetric stabilizer is an $(a, b = 0)$ trapping set, and a is even.

Proof Let the cardinality of the set of VNs of the stabilizer be a. By definition, the induced sub-graph having no odd-degree check nodes implies that $b = 0$. Also, according to the symmetric stabilizer definition, the disjoint VN sets that partitions the stabilizer must have the same odd degree check node neighbor set. This implies that there can only be even number of such disjoint sets which further implies that the parameter a is even. □

When there are more than a pair (an even number greater than two) of disjoint sets of VNs, the symmetric stabilizer can be split into smaller symmetric stabilizers.

From the discussion earlier, it follows that symmetric stabilizers are trapping sets not only for the syndrome BP decoder, but for many other iterative decoders, such as bit-flipping, Gallager-B and MSA with different critical number and strength. Harmfulness of symmetric stabilizers associated with decoders is distinct from classical-type trapping sets, as summarized in the following theorem.

Lemma 2: For an $(a, 0)$ symmetric stabilizer TS with any iterative decoder with a critical number $a/2$, no error pattern on more than $a/2$ nodes of the symmetric stabilizer is a trapping set.

Proof Consider an $(a, 0)$ symmetric stabilizer with critical number $a/2$ for a syndrome decoder $\mathcal{D}_s$. By the definition of the critical number, any error pattern of weight smaller than the critical number $a/2$ with support on the symmetric stabilizer is corrected by the decoder $\mathcal{D}_s$. Error patterns of weight larger than the critical number $a/2$ with support on the symmetric stabilizer are decoded correctly, converging to their respective low-weight degenerate error pattern. □

In Fig. 4 (b), if a syndrome decoder $\mathcal{D}_s$ is able to correct all error patterns of weight smaller than five, it can also correct error patterns of weight six and more by converging to their respective low-weight degenerate error patterns.

The strength of a $(a, 0)$ symmetric stabilizer TS with critical number $a/2$ is given by the twice the number of possible partitions into two disjoint subsets of VNs that satisfy the symmetric stabilizer definition. Each of such partition (distinct by their unsatisfied syndromes) contributes two error patterns each to the decoder failure in the TS.

B. Searching for Quantum Trapping Sets

Using the definition of a QTS, one can search for small sub-graphs in the Tanner graph of the QLDPC code to identify

and enumerate the QTSs. There are efficient algorithms for TS search widely used in classical literature [25], [26] to identify sub-graphs that are (a, b) TSs. Such techniques are utilized in the search for classical-type TSs. Note that there can be more than one non-isomorphic sub-graphs with the same (a, b) parameters. For example, a $(5, 3)$ TS can have non-isomorphic sub-graphs as in Fig. 2(b) and Fig. 3. Observe that they all have different combinations of short cycles of length six, eight and ten. Enumeration of cycles and their combinations also allows to find harmful classical-type TSs in the QLDPC code. Unlike these classical-type TSs, the search for symmetric stabilizer TSs requires a different approach of finding low-weight codeword sub-graphs [27] with additional symmetry constraints. In the case of CSS codes, the H_{Z} even-weight stabilizer generators are examples of symmetric stabilizer TSs for iterative decoding over the Tanner graph of H_{X} matrix and vice-versa. After obtaining the list of relevant QTSs, we can perform decoder simulation with an iterative decoder $\mathcal{D}_s$ to verify their relative harmfulness. In the next section, we find and enumerate QTSs in some prominent QLDPC code families presented in the literature. We also provide the harmfulness analysis of "dominant" QTSs present in these code families.

IV. Trapping Set Analysis of CSS codes

A myriad of QLDPC code families have been proposed over the years. They include the CSS-based constructions (bicycle codes [1], hypergraph product (HP) codes [7] and their generalizations [10], expander codes [8]), non-CSS based QLDPC codes [28], [29] and quaternary QLDPC codes [30]. In this section, we analyze trapping sets of CSS based QLDPC codes, the generalized bicycle codes and HP codes, in particular. Similar analysis may be extended to the general class of stabilizer codes.

A. Generalized bicycle codes

Bicycle codes [1] were generalized by Kovalev and Pryadko in [31] as follows: Consider two binary $n/2 \times n/2$ matrices A and B that commute ($AB = BA$). Let

$$H_{\mathrm{X}} = [A, B] \text{ and } H_{\mathrm{Z}} = [B^T, A^T].$$

The SIP condition is clearly satisfied by definition, and in [31], A and B are chosen as binary circulant matrices so that they commute. Bicycle codes are dual containing CSS codes where $B = A^T$. Compared to the HP codes, these codes generally have a wider range of parameters; in particular, they can have a higher rate while preserving the estimated error threshold [31]. In [10], Panteleev and Kalachev use binary polynomials over rings to define the circulant matrices for constructing $[[n, k]]$ family of generalized bicycle codes. The choice of circulant matrices determines the properties of both the classical-type TSs and the symmetric stabilizers in the code. Hence, the observations on QTSs in the example we discuss next can be generalized to the code family.

Example 2: For the purpose of illustration in our TS analysis, we chose the $A1[[254, 28]]$ code, where the circulant size is 127, $a(x) = 1 + x^{15} + x^{20} + x^{28} + x^{66}$ and $b(x) = 1 + x^{58} + x^{59} + x^{100} + x^{121}$ as given in Appendix B in [10]. The girth of the Tanner graph is six, CN degree $\rho = 10$ and VN degree $\gamma = 5$.

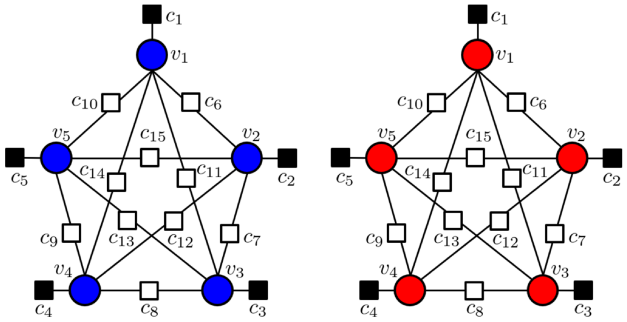


Fig. 7. A $(5, 5)$ TS with 5 variable nodes and 5 odd degree check nodes (the shaded squares represent the odd-degree checks). The degree of every variable node is 5. The blue and red shaded circles for the variable nodes indicate their relative position in the H_{X} matrix, from A and B, respectively.

1) Classical-type trapping sets: Based on our QTS definition, we search for QTSs of small size (upto $a = 5$) present in the $A1$ code in Ex. 2. As noted in [10], based on the circulant matrices in the $A1$ code, it does not have (a, b) trapping sets with $b \leq 5$. The $(5, 5)$ TS is the most harmful small sub-graph present in the Tanner graph making BP based iterative decoders to fail for low weight error patterns. The critical number and strength of this TS are determined by the specific decoder used and neighborhood of the $(5, 5)$ TS. Fig. 7 shows the dense $(5, 5)$ TS present in both the circulant matrices A and B. From their cyclic property, we can locate 127 (equal to the circulant size) isomorphic $(5, 5)$ TSs in each of them. In the Fig. 7, blue and red shaded circles for the VNs indicate their relative position in the H_{X} matrix, from A and B respectively.

The $(5, 5)$ trapping set in Fig. 7 has five variable nodes. Every variable nodes have exactly the same VN degree $\rho = 5$ and one odd-degree check node neighbor (black squares). The number of small cycles within the trapping set and their symmetry makes this a hard configuration to decode. For simple binary decoders like syndrome based Gallager-B, any weight three or more error patterns will result in a failure inducing set. Hence, the critical number for the $(5, 5)$ TS with the Gallager-B algorithm is $\mu = 3$. For stronger decoders such as BP and MSA decoder, the behavior is more complex and interesting. Any weight-5 error pattern in the TS in the circulant matrix A indicated by blue qubits results in a failure, whereas similar error patterns in the TSs in the circulant matrix B (indicated by the red qubits) are decoded correctly. Such a behavior is typically attributed to the neighborhood of the TS in the Tanner graph. Whether a TS is in fact harmful depends not just on the graphical configuration, but also on the neighborhood and the decoder. This "true" behavior of all such quantum TSs can be systematically analyzed and characterized by extending the sub-graph expansion-contraction algorithm [18] developed for classical LDPC codes to quantum codes, and it is left for future work.

2) Symmetric stabilizer trapping sets: Failures of syndrome-based iterative decoding on generalized bicycle code also consist of the degenerate error patterns in symmetric stabilizers discussed in the Section III-A2. An example of a

pair of symmetric degenerate error patterns of weight five in the Ex. 2 code is shown in Fig. 5(a) and Fig. 5(b). Together, they form a $(10, 0)$ TS shown in Fig. 4(b) referred as a symmetric stabilizer. Interestingly, the blue and red shaded circles indicates the variable nodes relative position as before in case of the $(5, 5)$ TS coloring. Also, these error patterns induce isomorphic sub-graphs - trees without any cycles, quite distinct from the error patterns in classical-type TSs which are usually composed of one or more cycles. Using the cyclic property of the circulant matrices in the code, we can easily locate 127 isomorphic symmetric stabilizers present in the Tanner graph of H_{X}, and similarly for H_{Z}.

Remark 3: The input syndrome (dark squares representing odd-degree checks) in both Fig. 5(a) and Fig. 5(b) is not matched correctly using iterative decoder using a parallel or flooding schedule. Breaking such TSs requires use asymmetric update of variable node decisions such as used in a layered/serial decoder. Since such symmetric trapping sets have clear distinction of red and blue nodes with respect to the cyclic matrices A and B, we can identify the layered decoder schedule that can break such trapping sets.

B. Hypergraph product codes

HP codes by Tillich and Zemor [7] and their improvements by Kovalev and Pryadko [32] are constructed by taking Kronecker product (denoted as $\otimes$) of two classical LDPC codes. Using two classical parity check matrices H_1 and H_2 of dimensions $m_1 \times n_1$ and $m_2 \times n_2$ respectively, we have $H_{\mathrm{X}} = \left[H_1 \otimes \mathrm{I}_{\mathrm{n}_2} \mid \mathrm{I}_{\mathrm{m}_1} \otimes H_2^T \right]$ and $H_{\mathrm{Z}} = \left[\mathrm{I}_{\mathrm{n}_1} \otimes H_2 \mid H_1^T \otimes \mathrm{I}_{\mathrm{m}_2} \right]$.

Example 3: For our trapping set analysis, we use the example of a $[[900, 36, 10]]$ HP code given in [12] using a symmetric Kronecker product of a single $(n = 24, k = 6, d = 10)$ classical code.

The classical LDPC code determines the HP code properties and its trapping sets. As we analyze the cycle profile of the Tanner graph of the classical code used in Ex. 3, we observe that there are 54 cycles of length six (6-cycles) and 160 8-cycles.

TABLE I
HPG CODE PARAMETERS AND NUMBER OF CYCLES

Code	n	m	g	χ_g	χ_{g+2}
[24,6,10]	24	18	6	54	160
[[900,36,10]]	900	432	6	2268	14496

These cycles appear in the HP codes, multiplying according to the size of the classical parity check matrix $(m = 18, n = 24)$ as given in Table I. For example, 54 six cycles in the classical code gives rise to $(54 \times 24) + (54 \times 18) = 2268$ 6-cycles in both H_{X} and H_{Z} matrix of the $[[900, 36, 10]]$ code. This behavior is consistent across the symmetric HP code family. Our TS search procedure identified thirty $(4, 2)$ trapping sets and ten $(5, 1)$ trapping sets in the classical code. Also, there are two non-isomorphic topologies of $(5, 3)$ TSs: one hundred and seventy $(5, 3)$ TSs whose induced graph has a six, eight, and ten-cycle, and fifteen $(5, 3)$ TSs having three eight cycles. All these trapping sets manifest themselves in the HP code with their count scaling as in the case of small

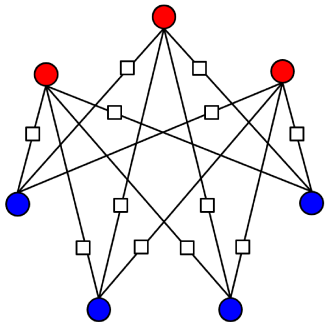


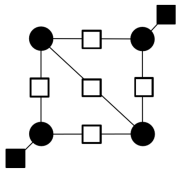
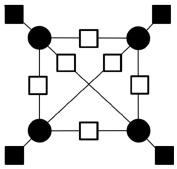
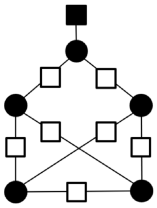
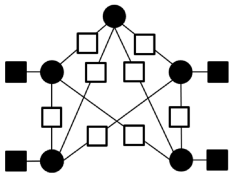
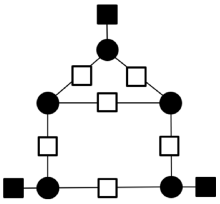
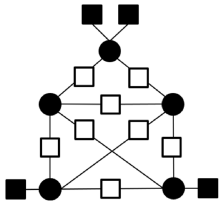
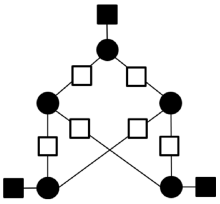
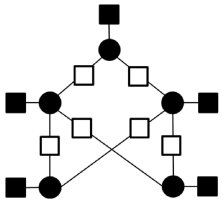
Fig. 8. A stabilizer sub-graph in the $[[900, 36, 10]]$ HP code [12] is not symmetric. Note that the red and blue variable nodes have four and three check node neighbors, respectively. Thus, a BP decoder converges to the red variable nodes as its output exploiting the asymmetry in the stabilizer.

cycles. In Table II, we enumerate all the smallest QTSs (with $a \leq 5, b \leq a$) present in the $[[900, 36, 10]]$ HP code having no CN with degree > 2 in their induced sub-graphs. These values for a and b are chosen as such classical-type TSs are typically the most harmful for iterative decoders. The QTS enumeration of H_{X} and H_{Z} for symmetric HP codes is the same. Also, the cycle enumerator series $\mathrm{CYC}(x) = \sum_{r \geq 0} \chi_r x^r$ for each QTS sub-graph in the parameters column in the Table indicates the number of small cycles present, which we refer to as its cycle profile. Observe that the $(5, 3)$ QTS with $\gamma = 3$ has two non-isomorphic topologies with different cycle profiles. Another interesting observation specific to these HP codes having VNs with degree 3 and 4 is the presence of $(5, 3)$ and $(5, 5)$ QTSs with the same cycle profile - $\mathrm{CYC}(x) = 3x^8$. The $(5, 5)$ QTSs are indeed the result of the Kronecker product in the HP codes. The QTSs in Table II are the main reason for poor iterative decoding performance of such family of codes.

We observe that the node degree of the classical parity check Tanner graph influences the symmetric property of the stabilizers of the HP code. In Ex. 3, since the classical parity check code chosen has $\gamma = 3$ and $\rho = 4$ in its Tanner graph, the variable nodes of the HP code have VN degrees 3 and 4. For the stabilizer in Fig. 8, the VNs with degree $\gamma = 4$ are shown as ● and those with $\gamma = 3$ as ●. The stabilizer is not symmetric according to the Definition 5. Suppose the input syndrome corresponds to all the check nodes in the sub-graph in error, then an iterative BP or MSA decoder (based on the update rule) will be able to successfully converge to the red error pattern. This happens as every ● VN uses messages from four CNs compared to three CNs for the ● VNs in the decoding process to successfully converge. Note that the above statement depends on the update rule chosen. For example, the red and blue error patterns in the stabilizer are indeed failure configurations for a simple binary decoding algorithm like Gallager-B algorithm. The above observation emphasizes the importance of the decoder in characterizing the harmfulness of QTSs [18]. In the subsequent section showing applications of TS analysis, we will use the example of different decoding schedule that "breaks" such symmetry of the stabilizer.

[1]In Table II, the parameters-(a,b), $\mathrm{CYC}(x)$, and Count are listed row-wise under the column header-Parameters for each QTS.

TABLE II
QTS ENUMERATION IN $H_{\mathrm{X}}/H_{\mathrm{Z}}$ OF $[[900, 36, 10]]$ HP CODE [12]

Quantum TS	Parameters [1] (a,b) CYC(x) Count	Quantum TS	Parameters (a,b) CYC(x) Count
	(4,2) $2x^6 + x^8$ 720		(4,4) $4x^6 + 3x^8$ 72
	(5,1) $2x^6 + 3x^8 + 2x^{10}$ 240		(5,4) $4x^6 + 5x^8 + 4x^{10}$ 36
	(5,3) $x^6 + x^8 + x^{10}$ 4080		(5,4) $5x^6 + 5x^8 + 2x^{10}$ 90
	(5,3) $3x^8$ 360		(5,5) $3x^8$ 5184

V. USING THE QTS TO DESIGN BETTER QLDPC CODES AND BETTER DECODERS

In this section, we explain practical importance of QTS analysis by providing two approaches for finite length QLDPC code/decoder design with QTS knowledge.

A. Improved Code Design

While previous research has observed the issue of symmetric degenerate errors [15], [16], there has been no effort to fully characterize them. Identifying symmetric stabilizers, particularly of low weight in the QLDPC code is an important step in quantifying the effect of degenerate errors on iterative decoding. Removal of symmetry in low-weight stabilizers present in the Tanner graph, especially during the QLDPC code design significantly reduces the number of instances of iterative decoder failure. For example, during the row removal step in the bicycle code [1] we carefully modify the original bicycle code to obtain codes wherein the stabilizer is asymmetric as in Fig. 6(b). The BP decoder will able to match to the syndrome (dark squares represent unsatisfied checks) correctly with the red error pattern, in contrast to the symmetric stabilizer in Fig. 6(a). Similarly, in the case of HP codes, careful removal of TSs in constituent classical LDPC codes helps to further optimize rate and minimum distance properties. Such code design improvements lead to performance gains especially in the error floor region for QLDPC codes. Here, we show an example of such an improved HP code design with quasi-cyclic (QC) LDPC [33] code for the constituent classical LDPC code.

1) Improved HP codes without harmful TSs: One of the disadvantages of QLDPC codes is that the random code construction makes the stabilizers highly non-local, requiring arbitrary qubit-qubit inter-connectivity to perform check operations. Using QC LDPC code brings structure to the constituent codes and flexibility in improving finite length QLDPC codes along with efficient implementation of decoders. Instead of the random codes which is only optimized for girth $g = 6$, we construct QC $[40, 10, 12]$ code with girth 8 and making sure small trapping sets present in the random code are not present. The QC code with circulant size $Q = 10$ is free of harmful trapping sets and constructed by progressively building the Tanner graph. Fig. 9 shows improved decoding performance (flooding BP decoder with $\ell_{max} = 100$ iterations) in the error floor regime for the newly constructed HP code.

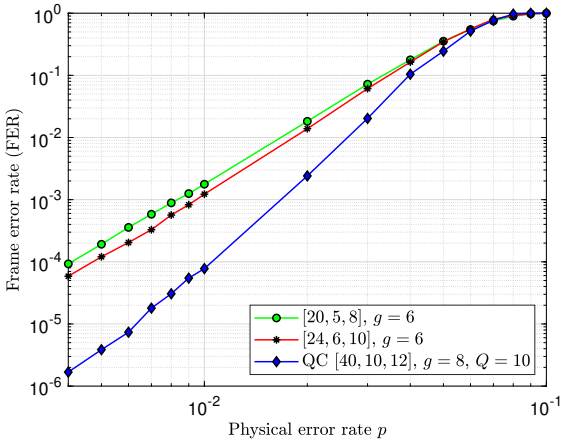


Fig. 9. Figure shows the FER performance comparison for the symmetric HP codes constructed using random constituent codes ([20,5,8] code and [24,6,10] code in [12]) with the HP code constructed using a trapping set aware QC [40,10,12] code.

B. Novel Decoder Design

An alternative or complementary approach is to devise iterative decoders that do not fail for the error patterns in the QTSs identified for the QLDPC code. This approach prevalent in classical LDPC decoders (finite alphabet iterative decoding (FAID) algorithms such as in [24]) do not ignore the topology of the TSs while devising decoder update rules. Breaking symmetry of messages by using non-linear message update rules leads to orders of magnitude decoding error performance improvements [24]. For QLDPC iterative decoders, the typically used parallel/flooding message update schedule (in the same iteration all variable/check nodes in the Tanner graph apply in parallel the same variable/check update function, respectively) attributes to decoders' failure on symmetric degenerate errors. We devise decoder strategy that corrects these errors by taking into account the topologies of the symmetric stabilizers in the code. Specifically, we show that an MSA decoder with sequential message update schedule (layered decoder as in classical literature [34]) that uses the knowledge of location of the symmetric stabilizers in the code as well as other harmful trapping sets improves the error floor decoding performance. As an intuitive example, suppose the symmetric stabilizer of weight 6 has support on variable nodes $v_1, \ldots, v_6$ with symmetric degenerate error patterns: e_1, e_2, e_3 and e_4, e_5, e_6. A layered decoder with the update order: starting with VN update of v_1, v_2, v_3, followed by the check node updates, and then VN update of v_4, v_5, v_6 converges to the correct error pattern without getting trapped. Since the schedule order is with respect to the variable nodes corresponding to the columns of the H matrix, we refer to such schedule as column-layered. In addition to fast decoder convergence in terms of the number of iterations [34], [35], column-layered decoders break some harmful TSs in classical LDPC codes [36]. In the following section V-B1, we compare the two schedules: flooding MSA and layered MSA decoders for the chosen QLDPC code.

1) Layered Decoding to break QTSs: We employ a specific layered decoding schedule to break the symmetric stabilizers in the $A1[[254, 28]]$ code in Fig. 10 using a column layered schedule. The layered schedule employed here is based on the circulant-size of the cyclic matrices A and B. The symmetric trapping sets have clear distinction of red and blue nodes with respect to these cyclic matrices giving a straight forward update order: $v_1, \ldots, v_{127}$ followed by $v_{128}, \ldots, v_{254}$. The column-layered decoder (MSA with $\ell_{max} = 20$ iterations) is able to decode all the symmetric stabilizer TSs and numerous classical-type TSs correctly leading to two orders of magnitude improvement in the error floor regime (low physical error rates) compared to the flooding MSA decoder.

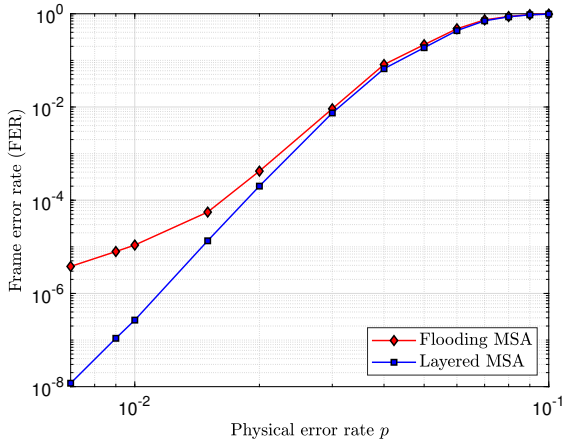


Fig. 10. Figure shows the FER performance comparison for the $A1[[254, 28]]$ code using the min-sum algorithm (MSA) for two different schedules: flooding/parallel and layered schedule. The layered schedule is able to decode all the symmetric stabilizer TSs and numerous classical-type TSs correctly leading to two orders of magnitude improvement in the error floor regime (low physical error rates).

Remark 4: This alternative approach is attractive when the QLDPC code is fixed and modifying it is not an option (due to technology or system-level constraints). Clearly, joint code and decoder design would guarantee further decoding performance improvement.

VI. Summary and Future Work

In this paper, we identified and classified quantum trapping sets using their definition adapted from the classical error correction to address the syndrome decoding scenario for QLDPC codes. The knowledge of QTSs is shown to significantly improve stabilizer code/decoder designs and also decoder performance in the error floor regimes of practical finite-length QLDPC codes. Analysis of failure configurations of the QLDPC codes, which are in fact generalization of the surface codes will have near-future implications in surface code designs and their decoders.

In future work, we will analyze finite length performances of recently proposed QLDPC codes that break the $\sqrt{n}$ growing minimum distance barrier [37] based on their QTS enumeration. We will establish the parent-child relationship between the harmful sub-graphs and determine their relative harmfulness. Understanding the effect of neighborhood of the

Tanner graph with respect to the decoder used is not easy, but important to understand the actual harmful error patterns. In future work, we plan to modify the expansion-contraction method [18] to QLDPC codes to obtain the exact set of most harmful configurations that should be avoided in the Tanner graph of QLDPC codes. Enumeration of symmetric stabilizers in QLDPC codes is also an important step towards exploiting degeneracy to the decoder's advantage. Approaches used in classical literature for structured QLDPC code constructions such as efficient low-weight codeword search are promising in this direction. In addition, extension of QTS definition to consider X and Z type errors together (correlated errors) and non-CSS stabilizer codes in general will set up the framework to study and explore non-binary quantum trapping sets.

Appendix A
Iterative Decoding Algorithm

A syndrome-based iterative decoder $\mathcal{D}_s$ is a 6-tuple $\mathcal{D}_s = (\mathcal{M}, \mathcal{Y}, \zeta, \Phi, \Psi, \hat{\Phi})$, where $\mathcal{M}$ is the message alphabets, $\mathcal{Y}$ is the same a-priori channel value chosen for all variable nodes, Φ,Ψ are the update functions used in variable and check nodes, and $\hat{\Phi}$ is the decision function, and ζ is the check value alphabet (for syndrome) with $\boldsymbol{\sigma}$ and $\hat{\boldsymbol{\sigma}}$ as the input and output syndromes respectively. The alphabets $\mathcal{M}$ and $\mathcal{Y}$ depend on a decoder type and quantum channel model.

Messages passed in an iterative decoder can be of floating point precision (floating point BP and MSA) or quantized to fixed number of levels for practical implementation. For a quantized decoder with Z levels, the message alphabet $\mathcal{M}$ consists of $Z = 2z + 1$ levels to which the message values are confined to. The message alphabet is defined as follows: $\mathcal{M} = \{-B_z, \ldots, -B_1, 0, B_1, \ldots, B_z\}$, where $B_i \in \mathbb{Z}+$ (positive integers) and $B_i > B_j$ for any $i > j$. The sign of a message $m \in \mathcal{M}$ can be interpreted as the error estimate of the variable node for which m is being passed to or from (positive for zero and negative for one), and the magnitude as a measure of how reliable the error estimate is. For BSC, the initial channel value for variable node v_i is set as $y_i = +Y$ mapping $0 \rightarrow Y$ according to the assumption of zero error pattern. The variable node message from v_i is initialized to $\Phi(y_i, \mathbf{0})$, and in each iteration updated according to the rules Φ and Ψ.

The messages passed over the edges of the Tanner graph (say, at ℓ-th iteration-iteration will be indicated as superscript when required) are denoted as follows: $\mu_{c_i \rightarrow v_j}$ and $\nu_{v_j \rightarrow c_i}$ denote a message from check node c_i to variable node v_j and vice-versa respectively.

Check node message $\mu^{(\ell)}_{c_i \rightarrow v_j} = \Psi(\mathbf{n}^{(\ell-1)}, \sigma_i)$, where $\mathbf{n} = \nu_{\mathcal{N}(c_i) \backslash v_j \rightarrow c_i}$ denote all incoming variable node messages to the check node c_i except from the variable node v_j. Note that Ψ is a symmetric function, i.e., any permutation of the function variables leaves the function unchanged. Variable node message is updated as $\nu^{(\ell)}_{v_j \rightarrow c_i} = \Phi(y_j, \mathbf{m}^{(\ell)})$, where $\mathbf{m} = \mu_{\mathcal{N}(v_j) \backslash c_i \rightarrow v_j}$ denote all incoming check node messages to the variable node v_j except the message from the check node c_i.

The decision function on v_j is computed using all messages incoming to v_j denoted by $\mathbf{l} = \mu_{\mathcal{N}(v_j) \rightarrow v_j}$. The decision function $\lambda^{(\ell)}_j = \hat{\Phi}(\mathbf{l}^{(\ell)}, y_j)$ decides the error bit $\hat{E}_j$ based on the sign using an indicator function as $\hat{e}^{(\ell)}_j = \mathbb{1}_{\lambda^{(\ell)}_j < 0}$. Output syndrome value for i^{th} check node in the ℓ-th iteration $\hat{\sigma}^{(\ell)}_i = \sum_{k \in \mathcal{N}(c_i)} \hat{e}_k$ modulo-2. A check node c_i is matched only if $\hat{\sigma}_i = \sigma_i$. If all syndromes are matched, we say that iterative decoder $\mathcal{D}_s$ successfully decoded to output the error pattern $\hat{\mathbf{e}}$.

The order of message passing in the Tanner graph is generally referred to as the updating schedule. Message passing follows a parallel/flooding schedule where Ψ at all CNs are updated simultaneously followed by updating Φ at all VNs simultaneously. In contrast, a row (column) layered schedule performs sequential update of messages in an order. An iteration of row (column) layered decoder proceeds by computing a check node (variable node) update function in the sequence followed by computing neighboring variable node (check node) function till all check nodes (variable nodes) are updated. Decision function $\hat{\Phi}$ computed at each layer accelerates the decoder convergence significantly.

Acknowledgment

We would like to thank David Declercq, Leonid Pryadko, and Saikat Guha for helpful discussions and insights. This work is funded by the NSF under grants SaTC-1813401, CCF-1855879, ECCS/CCSS-2027844 and NSF-ERC 1941583.

References

[1] D. MacKay, G. Mitchison, and P. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Trans. on Inform. Theory*, vol. 50, no. 10, pp. 2315–2330, Oct. 2004.

[2] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, pp. R2493–R2496, Oct. 1995.

[3] D. Gottesman, "Fault-tolerant quantum computation with constant overhead," *Quantum Inform. and Computation*, vol. 14, no. 15–16, pp. 1338–1372, Nov. 2014.

[4] A. A. Kovalev and L. P. Pryadko, "Fault tolerance of quantum low-density parity check codes with sublinear distance scaling," *Phys. Rev. A*, vol. 87, p. 020304, Feb. 2013.

[5] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "The road from classical to quantum codes: A hashing bound approaching design procedure," *IEEE Access*, vol. 3, pp. 146–176, 2015.

[6] ——, "Fifteen years of quantum LDPC coding and improved decoding strategies," *IEEE Access*, vol. 3, pp. 2492–2519, 2015.

[7] J.-P. Tillich and G. Zemor, "Quantum LDPC codes with positive rate and minimum distance proportional to $n^{1/2}$," *Proc. IEEE Intl. Symp. on Inform. Theory*, pp. 799–803, Jul. 2009.

[8] A. Leverrier, J. Tillich, and G. Zémor, "Quantum expander codes," in *Proc. IEEE 56th Ann. Symp. on Foundations of Computer Science*, Berkeley, CA, USA, Oct. 2015, pp. 810–824.

[9] T. J. Richardson, "Error floors of LDPC codes," in *Proc. 41st Ann. Allerton Conf. Commun., Contr. and Comp.*, Monticello, IL, USA, Sept. 2003, pp. 1426–1435.

[10] P. Panteleev and G. Kalachev, "Degenerate quantum LDPC codes with good finite length performance," *arXiv:1904.02703*, 2019.

[11] B. Vasić, D. Nguyen, and S. K. Chilappagari, "Chapter 6 - failures and error floors of iterative decoders," in *Channel Coding: Theory, Algorithms, and Applications: Academic Press Library in Mobile and Wireless Commun.* Oxford: Academic Press, 2014, pp. 299–341.

[12] J. Roffe, D. R. White, S. Burton, and E. T. Campbell, "Decoding across the quantum LDPC code landscape," *arXiv:2005.07016*, 2020.

[13] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. on Inform. Theory*, vol. 41, pp. 1379 – 1396, 10 1995.

[14] M. Baldi, N. Maturo, E. Paolini, and F. Chiaraluce, "On the use of ordered statistics decoders for low-density parity-check codes in space telecommand links," *EURASIP J. Wirel. Commun. Netw.*, vol. 2016, no. 272, pp. 1– 15, 2016.

[15] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *Quantum Inform. and Computation*, vol. 8, no. 10, pp. 987–1000, Nov. 2008.

[16] A. Rigby, J. C. Olivier, and P. Jarvis, "Modified belief propagation decoders for quantum low-density parity-check codes," *Phys. Rev. A*, vol. 100, p. 012330, Jul. 2019. [Online]. Available: http://dx.doi.org/10.1103/PhysRevA.100.012330

[17] J. X. Li and P. O. Vontobel, "Pseudocodeword-based decoding of quantum stabilizer codes," *arXiv:1903.01202*, 2019.

[18] N. Raveendran, D. Declercq, and B. Vasić, "A sub-graph expansion-contraction method for error floor computation," *IEEE Trans. on Commun.*, vol. 68, no. 7, pp. 3984–3995, 2020.

[19] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A*, vol. 54, pp. 1098–1105, Aug. 1996.

[20] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, 1997.

[21] M. M. Wilde, "Logical operators of quantum codes," *Phys. Rev. A*, vol. 79, p. 062322, Jun 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.79.062322

[22] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. New York, NY, USA: Cambridge University Press, 2011.

[23] N. Raveendran, P. J. Nadkarni, S. S. Garani, and B. Vasić, "Stochastic resonance decoding for quantum LDPC codes," in *Proc. IEEE Intl. Conf. on Commun.*, May 2017, pp. 1–6.

[24] S. K. Planjery, D. Declercq, L. Danjean, and B. Vasić, "Finite alphabet iterative decoders, Part I: Decoding beyond belief propagation on the binary symmetric channel," *IEEE Trans. on Commun.*, vol. 61, no. 10, pp. 4033–4045, Nov. 2013.

[25] M. Karimi and A. Banihashemi, "Efficient algorithm for finding dominant trapping sets of LDPC codes," *IEEE Trans. Inform. Theory*, vol. 58, no. 11, pp. 6942–6958, Nov. 2012.

[26] D. V. Nguyen, S. Chilappagari, M. Marcellin, and B. Vasić, "On the construction of structured LDPC codes free of small trapping sets," *IEEE Trans. Inform. Theory*, vol. 58, no. 4, pp. 2280–2302, Apr. 2012.

[27] S. M. Khatami, L. Danjean, D. V. Nguyen, and B. Vasić, "An efficient exhaustive low-weight codeword search for structured LDPC codes," in *Proc. Inform. Theory and Applications Workshop*, San Diego, CA, USA, Feb. 10–15 2013, pp. 1 – 10.

[28] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Construction of quantum LDPC codes from classical row-circulant QC-LDPCs," *IEEE Commun. Letters*, vol. 20, no. 1, pp. 9–12, Jan. 2016.

[29] M. Hagiwara and H. Imai, "Quantum quasi-cyclic LDPC codes," in *Proc. IEEE Intl. Symp. on Inform. Theory*, Jun. 2007, pp. 806–810.

[30] Y. Xie and J. Yuan, "Reliable quantum LDPC codes over GF(4)," in *Proc. IEEE Globecom Workshops*, Dec. 2016, pp. 1–5.

[31] A. A. Kovalev and L. P. Pryadko, "Quantum kronecker sum-product low-density parity-check codes with finite rate," *Phys. Rev. A*, vol. 88, p. 012311, Jul 2013. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.88.012311

[32] A. Kovalev and L. Pryadko, "Improved quantum hypergraph-product LDPC codes," in *Proc. IEEE Intl. Symp. on Inform. Theory*, Jul. 2012, pp. 348–352.

[33] M. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. on Inform. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.

[34] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proc. IEEE Workshop on Signal Processing Systems*, 2004, pp. 107–112.

[35] E. Sharon, S. Litsyn, and J. Goldberger, "Efficient serial message-passing schedules for LDPC decoding," *IEEE Trans. on Inform. Theory*, vol. 53, no. 11, pp. 4076–4091, 2007.

[36] N. Raveendran and B. Vasic, "Trapping set analysis of horizontal layered decoder," in *Proc. IEEE Intl. Conf. Commun.*, Kansas City, MO, USA, May 2018, pp. 1–6.

[37] P. Panteleev and G. Kalachev, "Quantum LDPC codes with almost linear minimum distance," *arXiv:2012.04068*, 2020.