On the Efficacy of the Peeling Decoder for the Quantum Expander Code

Jefrin Sharmitha Prabhu[†⋆], Abhinav Vaishya[†⋆], Shobhit Bhatnagar[†], Aryaman Manish Kolhe[^], V. Lalitha[^], P. Vijay Kumar[†]

[†] IISc Bangalore

[^] IIIT Hyderabad

{vaishyaabhinav,jefrinsharmithaprabhu,shobhitb97,aryaman.mk613,lalitha.vadlamani83,pvk1729}@gmail.com

Abstract

The problem of recovering from qubit erasures has recently gained attention as erasures occur in many physical systems such as photonic systems, trapped ions, superconducting qubits and circuit quantum electrodynamics. While several linear-time decoders for error correction are known, their error-correcting capability is limited to half the minimum distance of the code, whereas erasure correction allows one to go beyond this limit. As in the classical case, stopping sets pose a major challenge in designing efficient erasure decoders for quantum LDPC codes. In this paper, we show through simulation, that an attractive alternative here, is the use of quantum expander codes in conjunction with the peeling decoder that has linear complexity. We also discuss additional techniques including small-set-flip decoding, that can be applied following the peeling operation, to improve decoding performance and their associated complexity.

I. Introduction

Quantum information processing promises an exponential speed-up for many classical tasks. However, physical qubits are far more susceptible to noise than classical bits. Quantum error correction is imperative for building a practical quantum computer, and has been an active field of research for the last two decades.

A popular class of quantum error-correcting codes is the class of stabilizer codes developed by Gottesman [1], where the codespace is defined as the simultaneous $+1$-eigenspace of an abelian group of Pauli operators, called the stabilizer group. A practical constraint on the stabilizer operators is that they must have low weight, i.e., they must operate on only a few qubits at a time. Further, each qubit must be operated upon by only a few stabilizer operators. This gives rise to the notion of quantum Low Density Parity Check (LDPC) codes, analogous to the classical case. Gottesman [2] showed that it is possible to build fault-tolerant quantum circuits using such quantum LDPC codes. Efforts to construct good quantum LDPC codes that have both high rate and large minimum distances gave rise to many interesting and innovative ideas [3]–[10]. One such idea is the hypergraph product (HGP) construction by

This research is supported by SERB Grant No. CRG/2021/008479. ⋆ indicates equal contribution.

Tillich and Zémor [11]. The HGP construction takes two classical LDPC codes as inputs and yields a quantum LDPC code. A special case is when the classical codes are expander codes [12]. In this case the resultant quantum code is called a quantum expander code [13].

While quantum expander codes do not simultaneously achieve a constant rate and linear minimum distance (unlike [6], [7], [9]), they are relevant to practice as they permit linear-time decoding [13]–[15].

However, in addition to having good rates and distances, practical quantum error-correcting codes also need to have low decoding complexity. Recently, the problem of recovering from qubit erasures has gained much interest as they occur in many physical systems such as photonic systems [16], [17] (where photon loss, which can be modeled as an erasure, is the dominant source of error) and systems based on neutral atoms [18], trapped ions [19], superconducting qubits [20] and circuit quantum electrodyanamics [21]. Moreover, for the class of HGP codes, it has been shown using a classical Viderman-like approach [22] that the problem of error correction can be converted to that of erasure correction for a certain error regime [23]. Several linear-time decoders for error-correction are known [8]–[10], however their error-correcting capability is limited to half of the minimum distance of the code. Erasure correction allows one to go beyond this limit. As in the classical case, stopping sets pose a major challenge in designing efficient erasure decoders for quantum LDPC codes. Several decoders for erasure-correcting quantum LDPC codes have been proposed in the literature [24]–[32] that deal with stopping sets in different ways.

Our contributions: The principal contribution of the paper is showing through simulation results that make a comparison across quantum LDPC codes, that an attractive option for dealing with erasures, is the use of quantum expander codes in conjunction with the linear-complexity peeling decoder. As in the classical case, stopping sets pose a major challenge in designing efficient erasure decoders for quantum LDPC codes. We discuss additional techniques such as small-set-flip decoding, cluster-based decoding, that are capable of improving decoding performance by handling specific classes of stopping sets. We also discuss their associated complexity. Our simulation results also include limited statistics on erasure patterns that the peeling decoder was unable to recover from.

Organization of the paper: Section II provides background on quantum error correction. The Hypergraph Product Code (HGP) along with the relevant prior literature on decoding the hypergraph product code, are presented in Section III. Our peeling-based decoding algorithm is presented in Section IV. Analysis of our algorithm is presented in Section V. Simulation results are presented in Section VI and the final section, Section VII, draws conclusions.

II. Quantum Error Correction

We will use $\mathbf{0}_n$ to denote the zero vector of length n, and I_n to denote the $(n \times n)$ identity matrix.

A. Classical Error-Correcting Codes

An $[n, k, d_{\text{min}}]$ binary, linear, classical error-correcting code $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_2^n$, that corresponds to the nullspace of a suitably-defined matrix H called the parity-check matrix of the code.

The code can also be defined using a bipartite graph $\mathcal{T}(V \cup C, E)$, alternatively denoted as $\mathcal{T}(H)$, called the Tanner graph. The left vertices V (variable nodes) and the right vertices C (check nodes) correspond to the columns and rows of the parity-check matrix H of size $|C| \times |V|$. An edge exists between $i \in C$ and $j \in V$ if $h_{ij} = 1$; otherwise, $h_{ij} = 0$.

A code is (d_V, d_C)-regular LDPC if its associated Tanner graph is biregular, with variable nodes V having degree d_V and check nodes C having degree d_C. A code is (d_V, d_C)-regular LDPC if its associated Tanner graph is biregular, with variable nodes V having degree d_V and check nodes C having degree d_C. The particular class of biregular, bipartite graphs that we will consider here is the class of expander graphs [12] defined below:

Definition 1. *(Expander graph) Let $\mathcal{T}(V \cup C, E)$ be a bipartite Tanner graph such that $|V| \geq |C|$. The graph $\mathcal{T}$ is said to be (γ_V, δ_V)-left-expanding (resp. (γ_C, δ_C)-right-expanding) for some constants $\gamma_V, \delta_V > 0$ (resp. $\gamma_C, \delta_C > 0$), if for any subset $S \subseteq V$ (resp. $\mathcal{T} \subseteq C$), such that $|S| \leq \gamma_V|V|$ (resp.$|T| \leq \gamma_C|C|$), the neighbourhood $\Gamma(S)$ of S (resp. $\Gamma(T)$ of $\mathcal{T}$) in the graph $\mathcal{T}(V \cup C, E)$ satisfies $|\Gamma(S)| \geq (1 - \delta_V)d_V|S|$ (resp. $|\Gamma(T)| \geq (1 - \delta_C)d_C|T|$). The graph $\mathcal{T}(V \cup C, E)$ is said to be $(\gamma_V, \delta_V, \gamma_C, \delta_C)$-left-right-expanding if it is both (γ_V, δ_V)-left-expanding and (γ_C, δ_C)-right-expanding.*

The expander graphs considered in this paper are left-right-expanding. Classical expander codes, introduced by Sipser and Spielman [12], are defined as codes associated with (γ, δ)-left-expanding (d_V, d_C)-regular Tanner graphs $\mathcal{T}(V \cup C, E)$. The minimum distance d of the expander code associated with Tanner graph $\mathcal{T}(V \cup C, E)$ satisfies the lower bound $d > \gamma|V|$. Viderman proposed in [22], a linear-time erasure decoding algorithm capable of correcting up to $\gamma|V|$ erasures.

Even though explicit construction of good expander graphs is challenging, such graphs can nevertheless be efficiently found using probabilistic techniques [33], [14] as outlined in the theorem below.

Theorem 1. *([14, Theorem 2.3]) Let $\delta_V, \delta_C > 0$ be two constants. For integers $d_V > \frac{1}{\delta_V}$ and $d_C > \frac{1}{\delta_C}$, a graph $\mathcal{T}(V \cup C, E)$ with left-degree bounded by d_V and right-degree bounded by d_C chosen at random according to some distribution is $(\gamma_V, \delta_V, \gamma_C, \delta_C)$-left-right expanding for $\gamma_V, \gamma_C = \Omega(1)$ with high probability.*

B. Quantum Error-Correcting Codes

A quantum error-correcting code encoding k logical qubits into n physical qubits is a 2^k-dimensional subspace of $(\mathbb{C}^2)^{\otimes n}$. Here $\otimes$ denotes tensor product. Gottesman developed the stabilizer formalism for quantum codes in [1], inspired by classical linear codes. The stabilizer formalism serves as a general framework for studying a large class of quantum code. We describe this formalism below.

The Pauli matrices are defined as

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \ X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \ Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

and $Y = iXZ$. The Pauli group on n qubits, $\mathcal{P}_n$, is the set of all Pauli operators acting on n qubits, with usual matrix multiplication as the group operation:

$\mathcal{P}_n = \{\omega P_1 \otimes P_2 \otimes \cdots \otimes P_n : P_i \in \{I_2, X, Z, Y\} \ \forall \ i, \omega = \{\pm 1, \pm i\}\}.$

The operator $P = X^{a_1}Z^{b_1} \otimes X^{a_2}Z^{b_2} \otimes \cdots \otimes X^{a_n}Z^{b_n}$ can be represented by the vector

$$p = (a_1 \ a_2 \cdots a_n \mid b_1 \ b_2 \cdots b_n) \in \mathbb{F}_2^{2n}.$$

This representation of an operator as a binary vector is called the *symplectic representation*.

A stabilizer group, $\mathcal{S}$, is an abelian subgroup of the Pauli group such that $-I_{2^n} \notin \mathcal{S}$. A quantum stabilizer code $Q_{\mathcal{S}}$ is defined as the simultaneous $+1$ eigenspace of all the operators in $\mathcal{S}$, i.e.,

$$Q_{\mathcal{S}} = \{|\psi\rangle : S\,|\psi\rangle = |\psi\rangle \ \ \forall \ S \in \mathcal{S}\},$$

where we have used Dirac's bra-ket notation. The weight of a Pauli operator $P = \omega^l P_1 \otimes P_2 \otimes \cdots \otimes P_n$ is the number of indices $j \in [1, n]$ such that $P_j \neq I_2$. For a quantum stabilizer code $Q_{\mathcal{S}}$ defined via a stabilizer group $\mathcal{S}$, the minimum distance, $d_{\min}$ is defined as the minimum weight of all Pauli operators in $N(\mathcal{S}) \setminus \mathcal{S}$, where $N(\mathcal{S})$ is the normalizer of $\mathcal{S}$ in $\mathcal{P}_n$. This stabilizer code $Q_{\mathcal{S}}$ is denoted as a $[[n, k, d_{\min}]]$ code.

C. Quantum Erasure Channel

The quantum erasure channel is a noise model in which each qubit is lost or erased independently with some probability. This loss can be detected, and the density matrix of the qubit, ρ, is replaced with a maximally mixed state, $\frac{I}{2}$. The maximally mixed state can be expressed as $\frac{I}{2} = \frac{1}{4}(\rho + X\rho X^\dagger + Y\rho Y^\dagger + Z\rho Z^\dagger)$, implying that each erased qubit can be interpreted as being acted upon by the Pauli operators I, X, Y, and Z with equal probability. In this manner, erasure correction is effectively converted into correction of errors with known locations, denoted by a vector ε. Replacing each qubit in the support of ε with a maximally mixed state is equivalent to the encoded state being subjected to a uniform Pauli error E supported on the erasure locations, i.e., $\text{supp}(E) \subseteq \text{supp}(\varepsilon)$.

D. CSS Codes

Calderbank-Shor-Steane (CSS) codes are a sub-class of stabilizer codes where each stabilizer generator is a tensor product of either X-type or Z-type Pauli matrices [34], [35], and are one of the most well-studied classes of quantum codes. A CSS code is defined by two binary classical codes, $\mathcal{C}_X = \ker(H_X)$ and $\mathcal{C}_Z = \ker(H_Z)$, such that $\mathcal{C}_Z^\perp \subseteq \mathcal{C}_X$ (or equivalently, $\mathcal{C}_X^\perp \subseteq \mathcal{C}_Z$). Here, H_X of size $r_X \times n$ and H_Z of size $r_Z \times n$ represent the parity-check matrices of $\mathcal{C}_X$ and $\mathcal{C}_Z$, respectively. The basis of dual code ${\mathcal{C}_X}^\perp$ (resp. ${\mathcal{C}_Z}^\perp$) corresponds to the X-type (resp. Z-type) stabilizer generators and is used to correct Z-type (resp. X-type) errors.

The quantum code CSS$(\mathcal{C}_X, \mathcal{C}_Z)$ is a stabilizer code with stabilizer generators determined by the following check matrix:

$$H = \begin{pmatrix} H_X & 0 \\ 0 & H_Z \end{pmatrix}$$

The dimension of the CSS code is given by $k = \dim(\mathcal{C}_X/\mathcal{C}_Z^\perp) = \dim(\mathcal{C}_Z/\mathcal{C}_X^\perp) = \dim(\mathcal{C}_X) + \dim(\mathcal{C}_Z) - n$. The minimum distance of CSS code is given by, $d_{\min} = \min\{d_X, d_Z\}$, where, $d_X := \min\{w_H(E) \mid E \in \mathcal{C}_X \backslash \mathcal{C}_Z^\perp\}$ and $d_Z := \min\{w_H(E) \mid E \in \mathcal{C}_Z \backslash \mathcal{C}_X^\perp\}$, where $w_H(\cdot)$ denotes Hamming weight. The CSS code CSS$(\mathcal{C}_X, \mathcal{C}_Z)$ is an $[[n, k, d_{\min}]]$ code.

The error operator E, is symplectically represented by the binary $2n$-tuple (E_X, E_Z). The syndrome corresponding to (E_X, E_Z) is given by $\sigma = (\sigma_X, \sigma_Z)$, where $\sigma_Z = H_X E_Z^T$ (resp. $\sigma_X = H_Z E_X^T$) is obtained by measuring X-type (resp. Z-type) stabilizer generators and is used for correcting Z-type (resp. X-type) errors independently.

Error correction for CSS codes can be performed independently for X-errors and Z-errors. Specifically, by considering the syndrome $\sigma = (\sigma_X, \mathbf{0}_{r_z})$ (respectively, $\sigma = (\mathbf{0}_{r_x}, \sigma_Z)$), X-errors (respectively, Z-errors) can be corrected separately. In this paper, the description of the algorithm presented in Section IV is restricted to only X-type errors E, using the syndrome σ obtained by measuring the Z-type stabilizer generators.

Quantum LDPC codes are a class of stabilizer codes where each stabilizer generator acts on a constant number of qubits, and each qubit is acted upon by a constant number of stabilizer generators. When CSS codes are constructed with the classical parity-check matrices H_X and H_Z being sparse, they correspond to quantum LDPC codes. Precisely, these matrices have a constant row weight and a constant column weight, which characterizes the LDPC property.

III. Hypergraph Product Codes

The HGP construction by Tillich and Zémor [11] provides a framework for constructing quantum CSS codes from two classical codes.

Let $\mathcal{C}$ be a classical code with parity-check matrix H. The transpose code $\mathcal{C}^T$ is obtained as the null-space of H^T. Informally, $\mathcal{C}^T$ is the code obtained by reversing the roles of check and variable nodes in the Tanner graph of $\mathcal{C}$.

Let $\mathcal{C}_1$ be an $[n_1, k_1, d_1]$ classical code with parity-check matrix H_1 of size $r_1 \times n_1$, and $\mathcal{C}_2$ be an $[n_2, k_2, d_2]$ classical code with parity-check matrix H_2 of size $r_2 \times n_2$.

The product code $\mathcal{C}_1 \otimes \mathcal{C}_2$ is the classical code with a parity-check matrix defined as:

$$H = \begin{pmatrix} I_{n_1} \otimes H_2 \\ H_1 \otimes I_{n_2} \end{pmatrix}$$

The hypergraph product of two classical codes $\mathcal{C}_1$ and $\mathcal{C}_2$, denoted as $\mathrm{HGP}(\mathcal{C}_1, \mathcal{C}_2)$, is defined as the quantum CSS code, $\mathrm{CSS}(\mathcal{C}_X, \mathcal{C}_Z)$, where, $\mathcal{C}_X^T = \mathcal{C}_1 \otimes \mathcal{C}_2^T$, and $\mathcal{C}_Z^T = \mathcal{C}_1^T \otimes \mathcal{C}_2$. The check matrix of $\mathrm{HGP}(\mathcal{C}_1, \mathcal{C}_2)$ is given by

$$H = \begin{pmatrix} H_X & 0 \\ 0 & H_Z \end{pmatrix}$$

where $H_X = \begin{pmatrix} I_{n_1} \otimes H_2 & H_1^T \otimes I_{r_2} \end{pmatrix}$ and $H_Z = \begin{pmatrix} H_1 \otimes I_{n_2} & I_{r_1} \otimes H_2^T \end{pmatrix}$. It can be verified that $H_X H_Z^T = 0$, satisfying the CSS property. Furthermore, if $\mathcal{C}_1$ and $\mathcal{C}_2$ are classical LDPC codes, then $\mathrm{HGP}(\mathcal{C}_1, \mathcal{C}_2)$ is a quantum LDPC code.

Let $\mathcal{T}(H_1) := \mathcal{T}(V_1 \cup C_1, E_1)$ be the Tanner graph for the classical code $\mathcal{C}_1$ and $\mathcal{T}(H_2) := \mathcal{T}(V_2 \cup C_2, E_2)$ be the Tanner graph for the classical code $\mathcal{C}_2$. The Tanner graph of $\mathrm{HGP}(\mathcal{C}_1, \mathcal{C}_2)$ is a 4-partite graph $\mathcal{T}((V_1 \times V_2) \cup (C_1 \times C_2) \cup (V_1 \times C_2) \cup (C_1 \times V_2), E)$.

Here, the qubits are indexed by the set $V_Q := (V_1 \times V_2) \cup (C_1 \times C_2)$. The X-stabilizer generators are indexed by the set $C_X := C_1 \times V_2$, and the Z-stabilizer generators are indexed by the set $C_Z := V_1 \times C_2$.

Two nodes $u_1, u_2 \in V_Q \times (C_X \cup C_Z)$ are connected if: the first coordinate of u_1 and u_2 is the same, and the second coordinate is connected in $\mathcal{T}(H_2)$, or the second coordinate of u_1 and u_2 is the same, and the first coordinate is connected in $\mathcal{T}(H_1)$. Fig. 1 illustrates this construction.

If $\mathcal{C}_1$ and $\mathcal{C}_2$ are classical LDPC codes, then $\mathrm{HGP}(\mathcal{C}_1, \mathcal{C}_2)$ is a quantum LDPC code.

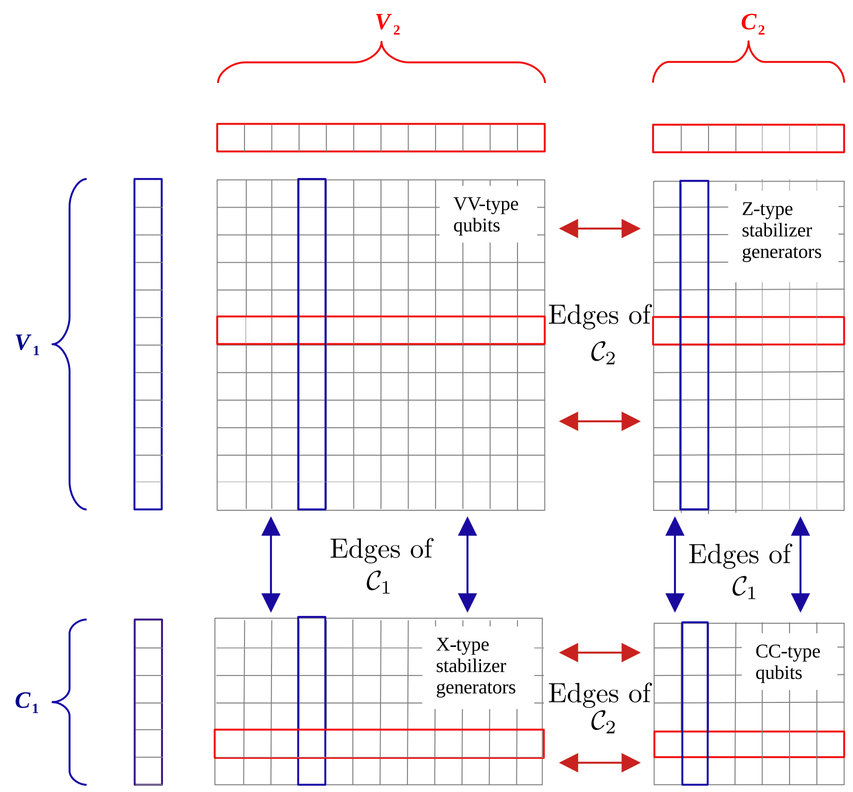


Fig. 1: The Tanner graph for the HGP code constructed from classical codes $\mathcal{C}_1$ with Tanner graph $\mathcal{T}((V_1 \cup C_1), E_1)$ and $\mathcal{C}_2$ with Tanner graph $\mathcal{T}((V_2 \cup C_2), E_2)$.

A. *Quantum Expander Codes*

The hypergraph product construction of Tillich and Zémor, applied to classical expander codes, results in a class of quantum LDPC codes with constant rate and minimum distance of order $\Theta(\sqrt{n})$, where n is the length of the resulting quantum code. Such codes are referred to as quantum expander codes.

Let $\mathcal{T}(V \cup C, E)$ be a biregular $(\gamma_V, \delta_V, \gamma_C, \delta_C)$-left-right-expander graph with left degree d_V and right degree d_C, such that $d_V \leq d_C$. Consider the classical expander code, $\mathcal{C}$, associated with the Tanner graph $\mathcal{T}(H)$, with minimum distance $d_{\min}$ and parity-check matrix H of size $|C| \times |V|$.

The hypergraph product of $\mathcal{C}$ with itself results in a quantum expander code with parity-check matrices:

$$H_X = \left(I_{|V|} \otimes H \;\; H^T \otimes I_{|C|} \right), H_Z = \left(H \otimes I_{|V|} \;\; I_{|C|} \otimes H^T \right).$$

The parameters of this code are

$$[[N = |V|^2 + |C|^2,\; k \geq (|V| - |C|)^2,\; \geq \min\{d_{\min}, d_{\min}^T\}]],$$

where $d_{\min}^T$ is the minimum distance of the transpose code $\mathcal{C}^T$. Further, this quantum code is LDPC with generators of weight $(d_V + d_C)$, and each qubit is checked by at most $(2\max(d_V, d_C))$ generators.

We denote by V^2 (resp. C^2), the $V \times V$ type (resp. $C \times C$ type) qubits. The X-stabilizer generators are indexed by the set $C_X := C \times V$ of size R_X, and Z-stabilizer generators are indexed by the set $C_Z := V \times C$ of size R_Z. Denote by $\mathcal{T}(H_Z)$ (resp. $\mathcal{T}(H_X)$ the Tanner graph with the set of variable nodes $V^2 \cup C^2$ and the set of check nodes C_Z (resp. C_X).

B. Pruned Peeling Decoder

The pruned peeling erasure decoder for quantum CSS codes proposed by Connolly et al. in [36] is inspired by the classical peeling decoder described in [37], [33]. Consider a classical LDPC code with Tanner graph $\mathcal{T}(H)$. Given an erasure vector ε, two terms are defined: *dangling check* and *dangling bit*. A dangling check is a check node that is incident to a single erased bit and this erased bit is referred to as a dangling bit.

The classical peeling decoder for classical LDPC codes is an iterative algorithm that uses the syndrome value at the dangling check to determine the erased value at the corresponding dangling bit and then modifies the syndrome accordingly. Using this updated syndrome, the process repeats until the erasures are fully corrected or peeling is not possible further.

A stopping set for $\mathcal{T}(H)$ is defined as a subset of bits that does not contain a dangling bit. Thus, if a stopping set lies within the erasure location, it cannot be corrected using the classical peeling decoder, resulting in decoder failure.

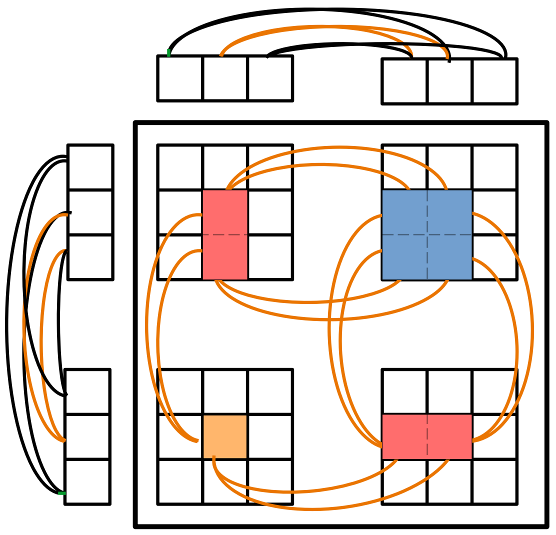
Now, consider decoding the X-type errors using the syndrome obtained by measuring Z-type stabilizer generators. This problem can be cast as a classical decoding problem of a linear code with Tanner graph $\mathcal{T}(H_Z)$, where H_Z is the parity-check matrix whose rows correspond to Z-type stabilizer generators. The classical peeling decoder can then be applied to a quantum CSS code directly; however, it tends to fail more frequently due to an increased number of stopping sets. This increase occurs because the support of an X-type generator forms a stopping set for the Tanner graph $\mathcal{T}(H_Z)$. This is due to the fact that the symplectic representation of the X-type stabilizers are codewords of the linear code $\ker(H_Z)$. Such stopping sets, induced by stabilizers, are referred to as stabilizer stopping sets.

The idea of the pruned peeling decoder is to iteratively correct all dangling bits using the syndrome at the dangling check. For the remaining errors that are not corrected (as they are part of a stabilizer stopping set), the approach is to identify a stabilizer generator S supported entirely within the erasure locations and then arbitrarily remove a qubit from this support. This works because the error E and the error ES act identically on this qubit. By breaking the stopping sets in this manner, the decoder can correct more erasures.

The pruned peeling decoder can be applied to any quantum CSS code; however, when applied to hypergraph product codes, it does not show significant improvement over the classical peeling decoder due to the nature of hypergraph stopping sets, as explained next.

C. Stopping Sets for Hypergraph Product Codes

Consider a hypergraph product code $\mathrm{HGP}(\mathcal{C}, \mathcal{C})$ following the notations from Section III-A. Let $\mathcal{T}(H)$ be the Tanner graph of $\mathcal{C}$, and $\mathcal{T}(H^T)$ be the Tanner graph of the transpose code $\mathcal{C}^T$. There are two types of stopping



🟧 Chosen X-type stabilizer generator

🟥 Stabilizer stopping sets

🟦 Z-type checks affected by the stopping sets

Fig. 2: Stabilizer stopping set for the HGP code constructed from two classical 3-bit repetition code.

sets for the Tanner graph $\mathcal{T}(H_Z)$:

1. *Stabilizer stopping sets*: These types of stopping sets are induced by the X-type stabilizer generators, as explained previously. See Fig. 2 for an example.

2. *Vertical and horizontal stopping sets*: These types of stopping sets arise from the stopping sets of the Tanner graphs $\mathcal{T}(H)$ and $\mathcal{T}(H^T)$. If S_V is a stopping set for $\mathcal{T}(H)$, then for all $v \in V$, $\{v\} \times S_V$ is a stopping set for the Tanner graph $\mathcal{T}(H_Z)$. These stopping sets $\{v\} \times S_V$ are referred to as horizontal stopping sets. Similarly, if S_C is a stopping set for $\mathcal{T}(H^T)$, then for all $c \in C$, $S_C \times \{c\}$ is a stopping set for the Tanner graph $\mathcal{T}(H_Z)$. These stopping sets $S_C \times \{c\}$ are referred to as vertical stopping sets. Fig. 3 illustrates this.

The pruned peeling decoder, when applied to hypergraph product codes, can successfully break out of some stabilizer stopping sets. However, this decoder fails when the erasure pattern lies within vertical or horizontal stopping sets. Furthermore, each horizontal (resp. vertical) stopping set in $\mathcal{T}(H)$ (resp. $\mathcal{T}(H^T)$) results in $\sqrt{N}$ copies of the corresponding stopping set in $\mathcal{T}(H_Z)$. Consequently, vertical and horizontal stopping sets account for the majority of failures of the pruned peeling decoder. As a result, pruned peeling decoder provides only a slight improvement over the classical peeling decoder for hypergraph product codes.

The vertical-horizontal (VH) decoder, discussed next, effectively addresses the vertical and horizontal stopping sets.

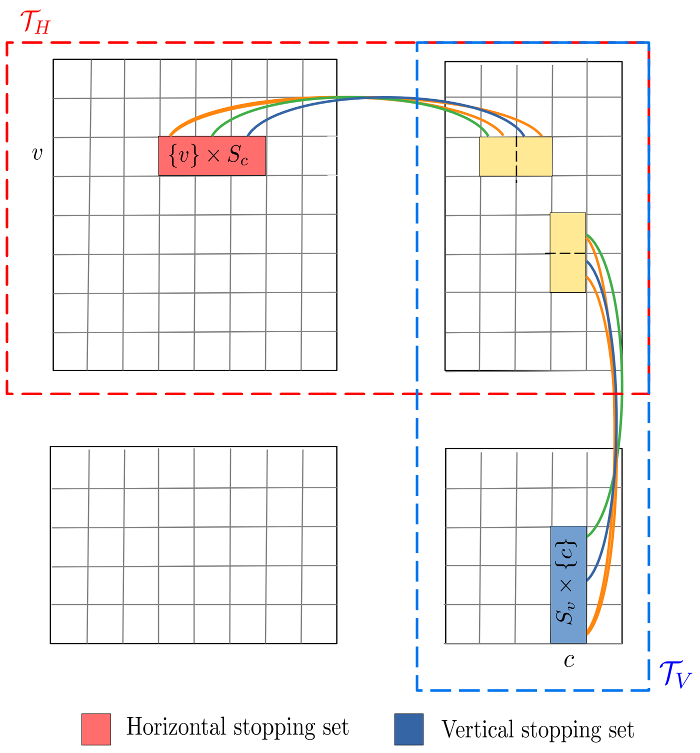


Fig. 3: Horizontal and vertical stopping sets for the HGP code.

D. *Vertical-Horizontal (VH) Decoder for Hypergraph Product Codes*

After observing that the pruned peeling decoder fails due to horizontal and vertical stopping sets, Connolly et al. [36] proposed the VH erasure decoder for hypergraph product codes. This decoder leverages the product structure of hypergraph product codes to decompose the erasures remaining after pruned peeling into vertical and horizontal clusters, which are then solved sequentially using classical Gaussian decoder.

For the description of the VH erasure decoder, consider the hypergraph product of the classical code $\mathcal{C} = \ker(H)$, having a Tanner graph $(V \cup C, E)$, with itself. The Tanner graph of HGP$(\mathcal{C}, \mathcal{C})$ is given by $\mathcal{T}(V_Q \cup (C_X \cup C_Z), E)$. Let $\mathcal{T}_V$ (resp. $\mathcal{T}_H$) be the subgraph induced by nodes $((V \cup C) \times C)$ (resp. $(V \times (V \cup C))$). The subgraph $\mathcal{T}_V$ (resp. $\mathcal{T}_H$) contains the vertical (resp. horizontal) edges of $\mathcal{T}(V_Q \cup (C_X \cup C_Z), E)$.

Given an erasure vector ε, let $V(\varepsilon)$ be the set of erased qubits and the checks connected to these qubits. A *vertical cluster* (resp. *horizontal cluster*) contains a subset of $V(\varepsilon)$ that forms a connected component in the graph $\mathcal{T}_V$ (resp. $\mathcal{T}_H$).

From the structure of $\mathcal{T}(V_Q \cup (C_X \cup C_Z), E)$, no two vertical clusters (resp. horizontal clusters) can intersect with each other. A vertical cluster and a horizontal cluster can intersect at most in one check, which is referred to as a *connecting check*. A check internal to a single cluster that is not a connecting check is called an *internal check*.

A VH graph for ε is a bipartite graph with vertices representing the horizontal and vertical clusters. An edge exists between two vertices in this graph if the corresponding clusters intersect at a connecting check.

A cluster that does not contain any connecting checks, i.e., it only has internal checks, is called an *isolated cluster*. These are the standalone vertices of the VH graph that do not have any edges. A cluster that contains exactly one connecting check, along with other internal checks, is called a *dangling cluster*. These correspond to the vertices of the VH graph that are connected to exactly one other vertex. A cluster containing more than one connecting check, in other words, one that is neither isolated nor dangling, is called a *non-dangling cluster*. A dangling cluster is further classified as either *free* or *frozen* according the following definition [36].

Definition 2. *If there exists an error pattern within a dangling cluster that has a trivial syndrome at the internal checks and syndrome* 1 *at the connecting check, then it is a free check. Otherwise, the check is called frozen.*

Equivalently, a dangling check is *frozen* if, and only if, for all error patterns within a dangling cluster consistent with the original syndrome at the internal checks, the syndrome at the connecting check is either always 0 or always 1. This is because if there were two errors, both consistent with the syndrome at the internal checks, but having different values at the connecting check, then the syndrome corresponding to the sum of these two errors is 0 on all the internal checks and 1 at the connecting check, implying that the connecting check is free. When the check is classified as free or frozen, we call the corresponding dangling cluster as free or frozen as well.

The VH decoder begins by classifying each cluster as isolated, dangling or non-dangling. Then, the dangling clusters are classified as free or frozen. An isolated cluster is corrected independently of other clusters, as it is not connected to any other cluster. Gaussian elimination is employed to correct such clusters. For a frozen dangling cluster, although it has a connecting check, this connecting check is frozen, meaning that any correction has the same effect on the connecting check. Therefore, a frozen dangling cluster is corrected in the same manner as an isolated cluster. Correcting a free dangling cluster is postponed until the end of the procedure. During the intermediate steps, this cluster, along with its connecting check, is removed from the Tanner graph $\mathcal{T}(H_Z)$ while the remaining erasures are corrected. After correcting the other erasures and updating the syndrome, a free dangling cluster is corrected using the updated syndrome within that cluster.

This is an iterative algorithm that iterates over all the clusters sequentially. It corrects isolated clusters and frozen dangling clusters independently of other clusters. Once corrected, these clusters are removed from the VH graph, potentially making the remaining clusters correctable.

The Gaussian decoder is used to determine whether a connecting check is frozen or free, as well as for correction within a cluster. The computational complexity of this process is $O(n^3)$ for a cluster of size $O(n)$.

This algorithm fails if the VH graph contains a cluster-cycle arising from stabilizer stopping sets. To address this, the authors modify the algorithm by removing free checks from all clusters, not just from free dangling clusters. This modification helps eliminate some cycles but introduces additional complexity, as it requires classifying multiple checks per cluster as frozen or free, rather than just a single check as in the case of dangling clusters. Consequently, the modified algorithm is slower than the VH decoder and has complexity exceeding $O(n^3)$ for a cluster of size $O(n)$. However, this modification still does not account for all the stabilizer stopping sets. Therefore, despite this

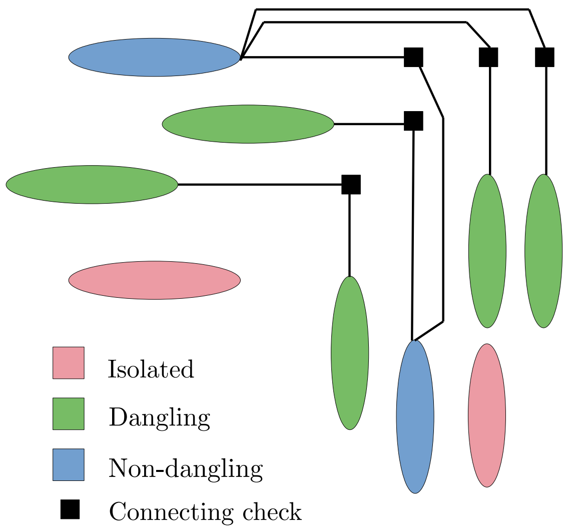


Fig. 4: Illustration of different types of clusters.

modification, the algorithm fails if a cycle still persists in the VH graph.

IV. A Linear-time erasure decoder for quantum expander codes

Our algorithm can be divided into two phases. In the first phase, we apply the classical peeling decoder to the Tanner graph $\mathcal{T}(H_Z)$. That is, we iteratively correct all dangling bits using the syndrome value at the corresponding dangling checks. The second phase, as given in Algorithm 4, begins by decomposing the remaining erasures into clusters, as defined and described in Section III-D.

Fig. 4 illustrates this.

Given a cluster κ, if it is dangling, we first classify it as either free or frozen. If the cluster cannot be classified at this stage, it will remain untouched for now and will be revisited at a later point. The classification procedure is described in detail in Section V-B.

If the cluster κ is isolated or frozen dangling, we correct it using the classical linear-time erasure decoding algorithm [22], given in Algorithm 1. Classical erasure decoding can be used here, as each horizontal (resp. vertical) isolated or frozen dangling cluster belongs to only a single classical Tanner graph, $\mathcal{T}(H)$ (resp. $\mathcal{T}(H^T)$), corresponding to the classical expander code. This is because such clusters do not have a connecting check in the case of an isolated cluster, or have a frozen connecting check in the case of a frozen dangling cluster.

The notation used in Algorithm 1 is as follows: V_κ denotes the set of variable nodes and C_κ denotes the set of check nodes within the cluster κ, respectively. For a variable node, $v \in V_\kappa$, $\Gamma(v)$ is a vector of length R_Z supported on the neighborhood of v and e_v is the weight-one binary vector of length N supported on the v^{th} variable node. Also, σ_κ denotes the syndrome vector of length R_Z restricted to the check nodes in the cluster, and $\sigma_{\kappa,u}$ indicates the syndrome value at the u^{th} check node.

Algorithm 1 begins by initializing a set Unique, which contains all the check nodes in C_κ that are connected to only one variable node $v \in V_\kappa$. We will refer to such checks as unique checks. While there exists a variable node v that is connected to a unique check u, we check the syndrome value, $\sigma_{\kappa,u}$, at this particular check. If $\sigma_{\kappa,u} = 1$, then we add an error vector supported at the v^{th} node to the estimate $\hat{E}_i$ and update the syndrome accordingly. Then we remove the variable node v from V_κ. Next, we check if any of the check nodes in the neighborhood of v have become unique checks. If they have become unique checks, they are added to the set Unique. Then the process repeats.

Algorithm 1 Algorithm to decode from erasures [22]

Input: For a dangling cluster κ, set of erasure locations V_κ, set of checks C_κ, syndrome $\sigma_\kappa \in \mathbb{F}_2^{Rz}$
Output: Estimate of error $\hat{E}_\kappa$

```
1: procedure PEEL
2:     $\hat{E}_0 = \mathbf{0}_N;\ \sigma_0 = \sigma_\kappa;\ i = 0;\ \text{unpeeled} = |V_\kappa|$
3:     $\text{Unique} = \{u \in C_\kappa \ : \ |V_\kappa \cap \text{supp}(\Gamma(u))| = 1\}$
4:     while $\exists\ v \in V_\kappa$ and $u \in \text{Unique}$ such that $\text{supp}(\Gamma(u)) \cap V_\kappa = v$ do
5:         if $\sigma_{\kappa,u} = 1$ then
6:             $\hat{E}_{i+1} = \hat{E}_i \oplus e_v$
7:             $\sigma_{i+1} = \sigma_i \oplus \Gamma(v)$
8:         end if
9:         $V_\kappa = V_\kappa \backslash \{v\}$
10:        for all $u' \in \text{supp}(\Gamma(v))$ do
11:            if $|\text{supp}(\Gamma(u')) \cap V_\kappa| = 1$ then
12:                $\text{Unique} = \text{Unique} \cup \{u'\}$
13:            else
14:                $\text{Unique} = \text{Unique} \backslash \{u'\}$
15:            end if
16:        end for
17:        $i = i + 1, \text{unpeeled} = \text{unpeeled} - 1$
18:    end while
19:    return $\hat{E}_i, \sigma_i, \text{unpeeled}$
20: end procedure
```

The estimated error within this cluster, $\hat{E}_{\text{PEEL}}$, is then added to $\hat{E}$, and $\sigma_{\text{PEEL}} \oplus \sigma_\kappa$ is added to σ . Finally, V_κ is removed from erasure ε.

In the case of a free dangling cluster, κ, the cluster and its free check are removed from the Tanner graph $\mathcal{T}(H_Z)$, added to a stack, and the correction is delayed until all correctable frozen dangling and isolated clusters are corrected. Note that each such cluster becomes essentially an isolated cluster later, so we use Algorithm 1 to correct it.

The process of correcting clusters independently can be viewed as peeling each cluster from the VH graph. After iteratively correcting these clusters, the remaining errors will be the ones which couldn't be peeled during any step in the algorithm and those that are part of cluster-cycles in the VH graph arising from the stabilizer stopping sets. These remaining erasures are then corrected using the SSF erasure-decoding algorithm, as described in Algorithm 3.

The SSF algorithm, proposed by Leverrier et al. in [13], is the quantum analogue of the bit-flip error-decoding algorithm introduced by Gallager for classical LDPC codes in [38] and analyzed by Sipser and Spielman for classical expander codes in [12].

The SSF decoder works similar to the bit-flip decoder by dividing its execution into several rounds, where the algorithm attempts to reduce the syndrome weight in each round. However, unlike the bit-flip decoder, where the syndrome weight can be reduced by flipping a single bit, the SSF decoder requires flipping a set of qubits F in order to reduce the syndrome weight. Here, by flipping a qubit, we mean that the qubit undergoes an X-type error.

Consider correcting X-type errors using the syndrome obtained by measuring Z-type stabilizer generators. The authors in [13] define a term *small set*, which is a subset of an X-type generator's support, and the set of all small sets is denoted by $\mathcal{F}$. The set F, which when flipped reduces the syndrome weight, is then chosen from $\mathcal{F}$. We refer the readers to Chapter 4 of [39] for more details.

SSF for errors can be modified to work for erasures or, more specifically, errors with known locations by defining small sets the following way. A set $F \subseteq V_Q$ is a small set if and only if it is included in the support of an X-type generator as well as belongs to the erasure set. SSF is analysed for erasures in Section V-C.

Finally, Algorithm 4 returns a Failure if the residual syndrome is not zero; otherwise, it returns the estimated error $\hat{E}$. The analysis of our algorithm is provided in Section V.

Algorithm 2 Dangling Cluster Classification

Input: For a dangling cluster κ, set of erasure locations V_κ, set of checks C_κ

Output: Either Free or Unclassified

```
1: procedure CLASSIFY
2:     Set the syndrome ($\sigma_\kappa^{\text{new}}$) at the internal checks in $C_\kappa$ as 0 and at the connecting check as 1
3:     $\hat{E}_{\text{PEEL}}$, $\sigma_{\text{PEEL}}$, unpeeled = PEEL($V_\kappa, \sigma_\kappa^{\text{new}}$)
4:     if unpeeled $\neq 0$ then
5:         return Unclassified          ▷ Procedure PEEL did not terminate successfully
6:     end if
7:     if $\sigma_{\text{PEEL}|C_\kappa} = 0$ then          ▷ If the syndrome restricted to the check $C_\kappa$ is trivial
8:         return Free
9:     end if
10:    return Frozen
11: end procedure
```

Algorithm 3 Small-Set-Flip (SSF) Algorithm [39]

Input: An erasure vector $\varepsilon \in \mathbb{F}_2^N$, a syndrome $\sigma \in \mathbb{F}_2^{R_Z}$
Output: Estimate of error $\hat{E}$

```
1: procedure Small-Set-Flip
2:     $\hat{E}_0 = \mathbf{0}_N;\ \sigma_0 = \sigma_\kappa;\ i = 0$
3:     while $\exists F$ such that $\mathrm{supp}(F) \in \mathcal{F} : |\sigma_i| - |\sigma_i \oplus \sigma(F)| \geq \beta d_C |\mathrm{supp}(F)|$ do
4:         $\mathrm{argmax}_{F \in \mathcal{F}} \frac{|\sigma_i| - |\sigma_i \oplus \sigma(F)|}{|F|}$
5:         $\hat{E}_{i+1} = \hat{E}_i \oplus F_i$
6:         $\sigma_{i+1} = \sigma_i \oplus \sigma(F_i)$
7:         $i = i + 1$
8:     end while
9:     return $\hat{E}_i, \sigma_i$
10: end procedure
```

Remark 1. *Algorithm 4 (in the current form) may result in an infinite loop if the condition on line 8 is always true after some point, due to classification failure. To avoid this, we can keep a variable $\hat{E}_i$ for the i^{th} iteration and check if the error in the current iteration is the same as in the previous one. If it is, then we break the while loop. For simplicity, we have not added this in the algorithm.*

V. Analysis of Algorithm 4

In this section, we provide the proof of correctness of Algorithm 4.

A. Analysis of Algorithm 1

Theorem 2. *Consider a classical expander code associated with a bipartite Tanner graph $\mathcal{T}(V \cup C, E)$, which is (γ, δ)-left expanding for constants $\gamma, \delta > 0$. Any error $E \subseteq \varepsilon$, where ε is the set of erasure locations such that $|\varepsilon| \leq \gamma|V|$, can be corrected by Algorithm 1 in linear time.*

Proof. We refer the readers to Section 4.2 in Ref. [22] for the proof.

□

In Algorithm 4, we use Algorithm 1 to correct an isolated or frozen dangling cluster κ. If $|V_\kappa| \leq \Gamma_V|V|$ (resp. $|V_\kappa| \leq \Gamma_C|C|$) for a horizontal (resp. vertical) cluster, then the erasures in this cluster are corrected completely according to Theorem 2.

B. Analysis of Algorithm 2

After applying the classical peeling decoder to the Tanner graph $\mathcal{T}(H_Z)$, none of the internal checks for any cluster would be dangling. It follows that if there is a dangling check in a cluster, then it has to be the connecting

Algorithm 4 Linear-time Erasure Decoder for Quantum Expander Codes

Input: An erasure vector $\varepsilon \in \mathbb{F}_2^N$, a syndrome $\sigma \in \mathbb{F}_2^{R_Z}$
Output: Either Failure or an X-type error $\hat{E} \in \{I, X\}^N$ such that $\sigma(\hat{E}) = \sigma$ and $\mathrm{supp}(\hat{E}) \subseteq \mathrm{supp}(\varepsilon)$

```
1: procedure LINEAR-TIME ERASURE DECODER
2:     $\hat{E} = \mathbf{0}_N$                                  ▷ $\hat{E}$: error estimate, initialized as a zero vector
3:     $E_{\text{res}} = \varepsilon$                              ▷ $E_{\text{res}}$: set of residual erasures
4:     $L$ = []                                                  ▷ $L$: an empty stack
5:     while there exists an isolated or a dangling cluster $\kappa$ do
6:         if $\kappa$ is dangling then
7:             output = CLASSIFY($V'_\kappa$)                     ▷ $V'_\kappa$: a copy of $V_\kappa$
8:             if output = Unclassified then
9:                 continue                                      ▷ Go to line 5
10:            end if
11:        end if
12:        if $\kappa$ is Isolated or Frozen then
13:            $\hat{E}_{\text{PEEL}}, \sigma_{\text{PEEL}}$ = PEEL($V_\kappa, \sigma_\kappa$)
14:            $\hat{E} = \hat{E} \oplus \hat{E}_{\text{PEEL}},\ E_{\text{res}} = E_{\text{res}} \oplus \hat{E}_{\text{PEEL}},\ \sigma = \sigma \oplus (\sigma_{\text{PEEL}} \oplus \sigma_\kappa)$
15:        else                                                  ▷ When $\kappa$ (and the connecting check) is free
16:            Remove the free connecting check $c$ of $\kappa$ from the Tanner Graph $\mathcal{T}(H_Z)$
17:            Add the pair $(\kappa, c)$ to the stack $L$
18:            Set $E_{\text{res}_{|V_\kappa}} = \mathbf{0}_{|V_\kappa|}$          ▷ $E_{\text{res}_{|V_\kappa}}$: residual errors restricted to $V_\kappa$
19:        end if
20:    end while
21:    while the stack $L$ is non-empty do
22:        Pop a pair $(\kappa, c)$ from the stack $L$
23:        Add the check node $c$ to the Tanner Graph $\mathcal{T}(H_Z)$
24:        $\hat{E}_{\text{PEEL}}, \sigma_{\text{PEEL}}$ = PEEL($V_\kappa, \sigma_\kappa$)
25:        $\hat{E} = \hat{E} \oplus \hat{E}_{\text{PEEL}},\ E_{\text{res}} = E_{\text{res}} \oplus \hat{E}_{\text{PEEL}},\ \sigma = \sigma \oplus (\sigma_{\text{PEEL}} \oplus \sigma_\kappa)$
26:    end while
27:    $\hat{E}_{\text{SSF}}, \sigma_{\text{SSF}}$ = SMALL-SET-FLIP($E_{\text{res}}, \sigma$)
28:    $\hat{E} = \hat{E} \oplus \hat{E}_{\text{SSF}},\ E_{\text{res}} = E_{\text{res}} \oplus \hat{E}_{\text{SSF}}$
29:    if $\sigma_{\text{SSF}} \neq \mathbf{0}_{R_Z}$ then
30:        return Failure
31:    end if
32:    return $\hat{E}$
33: end procedure
```

check. Thus, peeling for any dangling cluster either for classification or for erasure-correction must begin with the connecting check.

For classification, we fix the (virtual) syndrome at the internal checks as 0 and at the connecting check as 1 and use PEEL to assign values to variable nodes consistent with this syndrome. If the procedure does not terminate, some of the variables nodes are left unassigned. In this case, the PEEL procedure has failed to classify.

Since the procedure PEEL runs in linear-time, classification is also done in linear-time.

C. Analysis of Algorithm 3

The notation followed in this analysis is: Let $\mathcal{T}(V \cup C, E)$ be a (d_V, d_C)-biregular $(\gamma_V, \delta_V, \gamma_C, \delta_C)$-left-right-expanding graph. Let $\mathcal{C}$ be the classical expander code associated with $\mathcal{T}$. The quantum expander code $\mathcal{C}_{\mathcal{T}} = \mathrm{HGP}(\mathcal{C}, \mathcal{C})$ is obtained by taking the hypergraph product of $\mathcal{C}$ with itself.

Define r and β by:

$$r := \frac{d_V}{d_C} = \frac{|C|}{|V|}, \quad \beta := \frac{r}{2}\left[1 - 4\left(\delta_V + \delta_C + (\delta_C - \delta_V)^2\right)\right].$$

The decoding strategy of SSF, given in Algorithm 3, is to decrease the syndrome by flipping a set of qubits called a small set. Let the set of all small sets be given by

$$\mathcal{F} := \{F \subseteq \Gamma_X(g) \cap \varepsilon : g \in C_X\}.$$

Here, $\Gamma_X(g)$ represents the support of the X-type stabilizer generator g, and ε denotes the set of erasures. However, in Algorithm 3, the small set, the set of erasures, and the support of stabilizer generators are treated as vectors supported appropriately.

For correcting erasures, the idea is to go through the set of all small sets $\mathcal{F}$ and check whether there exists a small set $F \in \mathcal{F}$ that would decrease the syndrome. If such an F exists, we flip F and then proceed to find the next small set. In Algorithm 3 a small set F is flipped if and only if it decreases the syndrome bt atleast $\beta d_C|F|$. The analysis of the SSF decoder for erasures follows a similar analysis as done for errors in [14]. Given the syndrome σ, Algorithm 3 provides an estimate of the error $\hat{E}$, supported on the erasure locations, which satisfies the syndrome.

Following the references [13] and [14], we define the notion of critical generators. An X-type generator is termed a critical generator if its support contains a set of qubits that can be flipped to reduce the syndrome.

Definition 3. *A generator $g \in C_X$ is said to be a critical generator for $E \subseteq \varepsilon$ if:*

$$\Gamma_X(g) = \Gamma_1 \uplus \overline{\Gamma}_1 \uplus \Gamma_2 \uplus \overline{\Gamma}_2$$

where

- $\Gamma_X(g) \cap V^2 = \Gamma_1 \uplus \overline{\Gamma}_1$ *and* $\Gamma_X(g) \cap C^2 = \Gamma_2 \uplus \overline{\Gamma}_2$*;*
- $\Gamma_1, \Gamma_2 \subseteq E$ *and* $\overline{\Gamma}_1, \overline{\Gamma}_2 \subseteq \varepsilon \setminus E$
- *for all* $v_1 \in \Gamma_1$*,* $v_2 \in \Gamma_2$*,* $\overline{v}_1 \in \overline{\Gamma}_1$ *and* $\overline{v}_2 \in \overline{\Gamma}_2$*:*
  - $E \cap [\Gamma_Z[\Gamma_Z(v_1) \cap \Gamma_Z(v_2)] = \{v_1, v_2\}$
  - $E \cap [\Gamma_Z[\Gamma_Z(\overline{v}_1) \cap \Gamma_Z(\overline{v}_2)] = \phi$

  - $E \cap [\Gamma_Z[\Gamma_Z(v_1) \cap \Gamma_Z(\overline{v}_2)] = \{v_1\}$
  - $E \cap [\Gamma_Z[\Gamma_Z(\overline{v}_1) \cap \Gamma_Z(v_2)] = \{v_2\}$
- $\Gamma_1 \cup \Gamma_2 \neq \phi$.

Lemma 1. *([14, Lemma B.2]) For an error $E \subseteq V_Q$ such that $0 < |E| \leq \min(\gamma_V|V|, \gamma_C|C|)$, there exists a critical generator for E.*

Lemma 2. *([14, Lemma B.3]) Let $E \subseteq V_Q$ be a error such that $0 < |E| \leq r\,\min(\gamma_V|V|, \gamma_C|C|)$ then there exists an error $F \subseteq \mathcal{F}$ with $|\sigma(E)| - |\sigma(E \oplus F)| \geq \beta d_C|F|$.*

Theorem 3. *Let $\mathcal{T}(V \cup C, E)$ be a (d_V, d_C)-biregular $(\gamma_V, \delta_V, \gamma_C, \delta_C)$-left-right-expanding graph such that $\delta_V, \delta_C < 1/8$. For a quantum expander code $\mathcal{C}_\mathcal{T}$, the small-set-flip decoder runs in linear time in the code length $n = |V|^2 + |C|^2$ and decodes any erasure pattern ε where*

$$|\varepsilon| \leq r\min(\gamma_V|V|, \gamma_C|C|).$$

Proof. We run Algorithm 3 on the input E_0, and let $\hat{E}_f$ denote the output. Define

$$E_f = E_0 \oplus \hat{E}_f = E_0 \oplus \left(\bigoplus_{k=0}^{f-1} F_k \right).$$

We also define $U := E_0 \cup F_0 \cup \cdots \cup F_{f-1}$, referred to as the *execution support*, where $f \in \mathbb{N}$ represents the number of iterations. This set includes all qubits that were in error at some point during the algorithm's execution.

By definition, $F_i \subseteq \varepsilon \ \forall\ i$ and $E_0 \subseteq \varepsilon$, which implies $U \subseteq \varepsilon$. Consequently, $|U| \leq |\varepsilon|$, leading to

$$|U| \leq r\min(\gamma_V|V|, \gamma_C|C|).$$

Thus,

$$|E_f| = |E_0 \oplus \hat{E}_f| \leq |U| \leq r\min(\gamma_V|V|, \gamma_C|C|).$$

To prove Theorem 3, it is sufficient to show that $E_f = 0$. Assume, for the sake of contradiction, that $E_f \neq 0$. Then, by the hypothesis of Lemma 2, there exists $F \in \mathcal{F}$ such that $|\sigma(E_f)| - |\sigma(E_f \oplus F)| \geq \beta d_C|F|$. However, since E_f is the output, the while condition in Algorithm 3 is not satisfied for $\sigma_f = \sigma(E_0 \oplus \hat{E}_f)$, leading to a contradiction.

□

In Algorithm 4, we use Algorithm 3 to correct the errors which were not corrected in the previous iterations of peeling. From Theorem 3, if the number of such errors is up to $\frac{d_v}{d_c}\min(\gamma_V|V|, \gamma_C|C|)$, which is a fraction of the minimum distance of the quantum expander code, then they will be corrected completely by SSF.

VI. Simulation Results

In Fig. 5, we present simulation results for the decoding performance of $[[1525, 25]]$, $[[6100, 100]]$ and $[[8784, 144]]$ quantum expander codes under the peeling decoder. The plots are generated by performing 10^4 trials for each erasure rate.

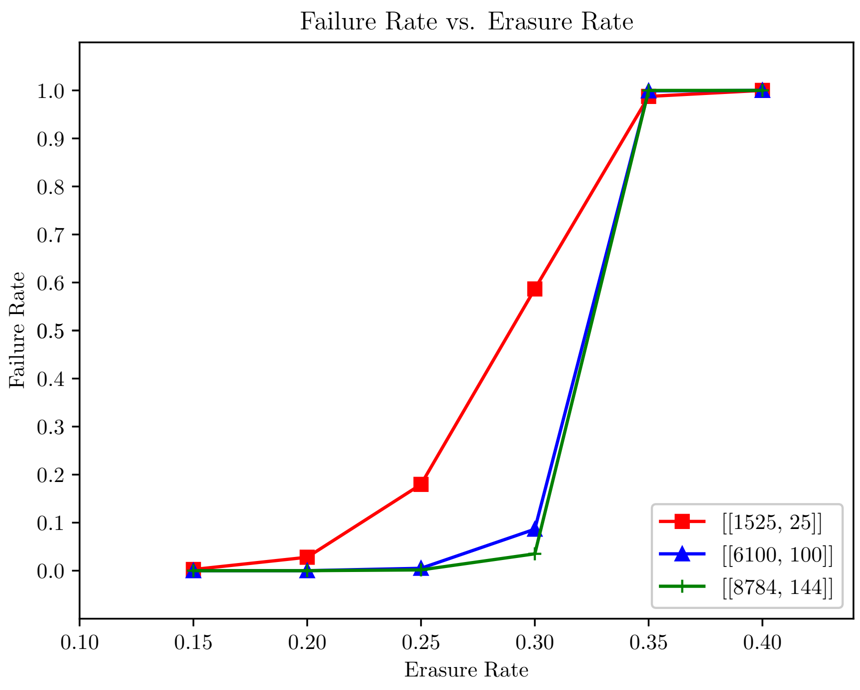


Fig. 5: Performance of $(5,6)$-biregular quantum expander codes using the peeling decoder.

As Fig. 5 shows, the failure rate decreases with the length of the quantum expander code for the same erasure rate. This is in contrast to the simulation results in [36] where increasing the length of the (general) HGP codes does not improve the performance of the peeling decoder, which stays constant at around 10^{-2} for an erasure rate of 0.2. Further, for the same erasure rate, the $[[1600, 64]]$ HGP code decoded using the VH decoder in [36] can tolerate a failure rate of approximately 10^{-3}, but at the cost of increased complexity. However, as Fig. 5 shows, a longer quantum expander code can be instead employed to achieve a comparable performance under peeling, which is linear.

In Fig. 6, we plot the maximum and minimum weights of the residual error after peeling. At erasure rate 0.3, we found that the maximum weight of a residual error was 35. However, the failure rate is 0.08 (see Fig. 5). In many cases, we found that the residual error was correctable by the modified SSF (with known locations). A more detailed analysis of the probability of SSF succeeding is an interesting direction for future work.

We also observed that the peeling procedure inside clusters was rarely successful because after the initial peeling, the connecting checks in the clusters had degree > 1 inside the cluster.

VII. Conclusion

The problem of recovering from qubit erasures has recently gained attention as erasures occur in many physical quantum systems. In this paper, it was shown through simulation results that compare across quantum LDPC codes, that an attractive option to be considered, is the use of quantum expander codes in conjunction with the linear-complexity peeling decoder. As in the classical case, stopping sets pose a major challenge in designing efficient

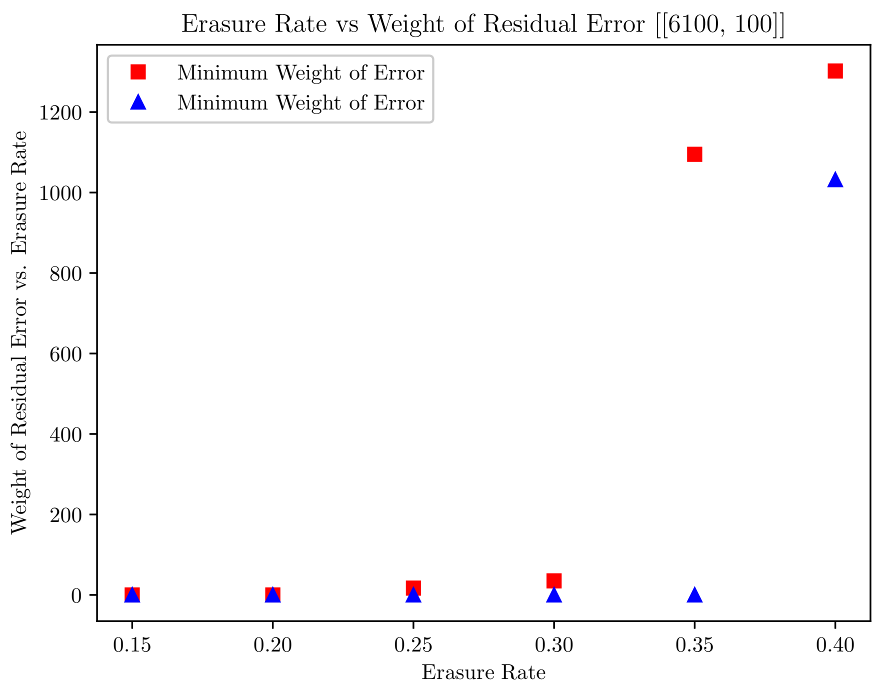


Fig. 6: Maximum and minimum weights of the residual error after peeling.

erasure decoders for quantum LDPC codes. The linear-time decoding algorithm presented here employed small-set-flip decoding as well as cluster-based decoding to handle stopping sets. However, it was our finding through simulation, that these additional techniques improve only in small measure, upon the performance of the peeling decoder. Our simulation results also include limited statistical data compiled on erasure patterns that the peeling decoder was unable to recover from.

REFERENCES

[1] D. Gottesman, *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

[2] D. Gottesman, "Fault-tolerant quantum computation with constant overhead," *arXiv preprint arXiv:1310.2984*, 2013.

[3] M. B. Hastings, J. Haah, and R. O'Donnell, "Fiber bundle codes: breaking the n 1/2 polylog (n) barrier for quantum ldpc codes," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1276–1288, 2021.

[4] N. P. Breuckmann and J. N. Eberhardt, "Balanced product quantum codes," *IEEE Transactions on Information Theory*, vol. 67, no. 10, pp. 6653–6674, 2021.

[5] P. Panteleev and G. Kalachev, "Quantum ldpc codes with almost linear minimum distance," *IEEE Transactions on Information Theory*, vol. 68, no. 1, pp. 213–229, 2021.

[6] P. Panteleev and G. Kalachev, "Asymptotically good quantum and locally testable classical ldpc codes," in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 375–388, 2022.

[7] A. Leverrier and G. Zémor, "Quantum tanner codes," in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 872–883, IEEE, 2022.

[8] A. Leverrier and G. Zémor, "Decoding quantum tanner codes," *IEEE Transactions on Information Theory*, vol. 69, no. 8, pp. 5100–5115, 2023.

[9] I. Dinur, M.-H. Hsieh, T.-C. Lin, and T. Vidick, "Good quantum ldpc codes with linear time decoders," in *Proceedings of the 55th annual ACM symposium on theory of computing*, pp. 905–918, 2023.

[10] S. Gu, C. A. Pattison, and E. Tang, "An efficient decoder for a linear distance quantum ldpc code," in *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pp. 919–932, 2023.

[11] J.-P. Tillich and G. Zémor, "Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Transactions on Information Theory*, vol. 60, no. 2, pp. 1193–1202, 2013.

[12] M. Sipser and D. A. Spielman, "Expander codes," *IEEE transactions on Information Theory*, vol. 42, no. 6, pp. 1710–1722, 1996.

[13] A. Leverrier, J.-P. Tillich, and G. Zémor, "Quantum expander codes," in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pp. 810–824, IEEE, 2015.

[14] O. Fawzi, A. Grospellier, and A. Leverrier, "Efficient decoding of random errors for quantum expander codes," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 521–534, 2018.

[15] O. Fawzi, A. Grospellier, and A. Leverrier, "Constant overhead quantum fault tolerance with quantum expander codes," *Communications of the ACM*, vol. 64, no. 1, pp. 106–114, 2020.

[16] E. Knill, R. Laflamme, and G. J. Milburn, "A scheme for efficient quantum computation with linear optics," *nature*, vol. 409, no. 6816, pp. 46–52, 2001.

[17] S. Bartolucci, P. Birchall, H. Bombin, H. Cable, C. Dawson, M. Gimeno-Segovia, E. Johnston, K. Kieling, N. Nickerson, M. Pant, *et al.*, "Fusion-based quantum computation," *Nature Communications*, vol. 14, no. 1, p. 912, 2023.

[18] Y. Wu, S. Kolkowitz, S. Puri, and J. D. Thompson, "Erasure conversion for fault-tolerant quantum computing in alkaline earth Rydberg atom arrays," *Nature communications*, vol. 13, no. 1, p. 4657, 2022.

[19] M. Kang, W. C. Campbell, and K. R. Brown, "Quantum error correction with metastable states of trapped ions using erasure conversion," *PRX Quantum*, vol. 4, no. 2, p. 020358, 2023.

[20] A. Kubica, A. Haim, Y. Vaknin, H. Levine, F. Brandão, and A. Retzker, "Erasure qubits: Overcoming the T_1 limit in superconducting circuits," *Physical Review X*, vol. 13, no. 4, p. 041022, 2023.

[21] T. Tsunoda, J. D. Teoh, W. D. Kalfus, S. J. de Graaf, B. J. Chapman, J. C. Curtis, N. Thakur, S. M. Girvin, and R. J. Schoelkopf, "Error-detectable bosonic entangling gates with a noisy ancilla," *PRX Quantum*, vol. 4, no. 2, p. 020354, 2023.

[22] M. Viderman, "Linear-time decoding of regular expander codes," *ACM Transactions on Computation Theory (TOCT)*, vol. 5, no. 3, pp. 1–25, 2013.

[23] A. Krishna, I. Livni Navon, and M. Wootters, "Viderman's algorithm for quantum ldpc codes," in *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2481–2507, SIAM, 2024.

[24] N. Delfosse and G. Zémor, "Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel," *Physical Review Research*, vol. 2, no. 3, p. 033042, 2020.

[25] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *Quantum*, vol. 5, p. 595, 2021.

[26] N. Delfosse, V. Londe, and M. E. Beverland, "Toward a union-find decoder for quantum ldpc codes," *IEEE Transactions on Information Theory*, vol. 68, no. 5, pp. 3187–3199, 2022.

[27] S. Lee, M. Mhalla, and V. Savin, "Trimming decoding of color codes over the quantum erasure channel," in *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 1886–1890, IEEE, 2020.

[28] H. M. Solanki and P. K. Sarvepalli, "Decoding topological subsystem color codes over the erasure channel using gauge fixing," *IEEE Transactions on Communications*, vol. 71, no. 7, pp. 4181–4192, 2023.

[29] H. M. Solanki and P. K. Sarvepalli, "Correcting erasures with topological subsystem color codes," in *2020 IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2021.

[30] H. Yao, M. Gökduman, and H. D. Pfister, "Cluster decomposition for improved erasure decoding of quantum ldpc codes," *arXiv preprint arXiv:2412.08817*, 2024.

[31] K.-Y. Kuo and Y. Ouyang, "Degenerate quantum erasure decoding," *arXiv preprint arXiv:2411.13509*, 2024.

[32] M. Gökduman, H. Yao, and H. D. Pfister, "Erasure decoding for quantum ldpc codes via belief propagation with guided decimation," in *2024 60th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1–8, IEEE, 2024.

[33] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE transactions on information theory*, vol. 47, no. 2, pp. 638–656, 2001.

[34] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, no. 2, p. 1098, 1996.

[35] A. M. Steane, "Error correcting codes in quantum theory," *Physical Review Letters*, vol. 77, no. 5, p. 793, 1996.

[36] N. Connolly, V. Londe, A. Leverrier, and N. Delfosse, "Fast erasure decoder for hypergraph product codes," *Quantum*, vol. 8, p. 1450, 2024.

[37] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.

[38] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.

[39] A. Grospellier, *Constant time decoding of quantum expander codes and application to fault-tolerant quantum computation*. PhD thesis, Sorbonne Université, 2019.