VH Decoder for HGP Codes in the Erasure Channel

Nicholas Connolly, Vivien Londe, Anthony Leverrier, Nicolas Delfosse

Contributors

https://arxiv.org/abs/2208.01002

Nicholas Connolly[1], Vivien Londe[2], Anthony Leverrier[3], Nicolas Delfosse[4]

[1] ~~INRIA Paris~~ OIST

[2] Microsoft France

[3] INRIA Paris

[4] ~~Microsoft Quantum~~ IonQ

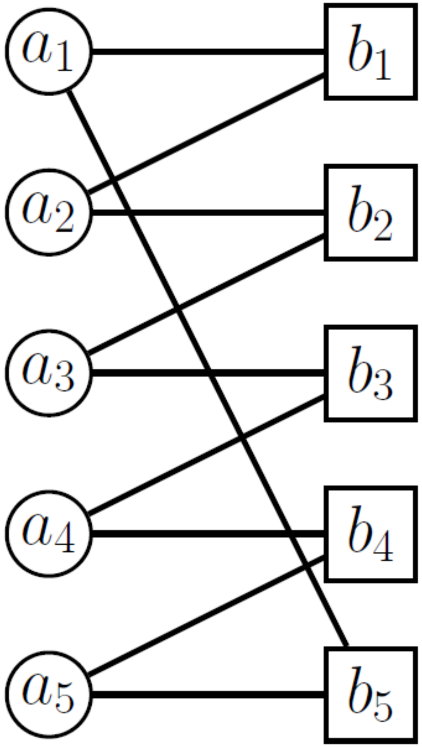
Classical Error Correcting Codes:
A Brief Reminder

Recall: Classical Codes

1. **A classical <u>linear code</u> *C* is a vector space over Z_2.**
   - A code of <u>length</u> n is the kernel of an $r \times n$ <u>parity check matrix</u> H.
   - C has <u>dimension</u> k as s subspace of $\mathbf{Z}_2^n$.
2. **Vectors *x* in *C* are <u>codewords</u>.**
3. ***C* is visualized by its bipartite <u>Tanner graph</u> *T(H)*.**

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \ker(H) = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

$T(H) =$

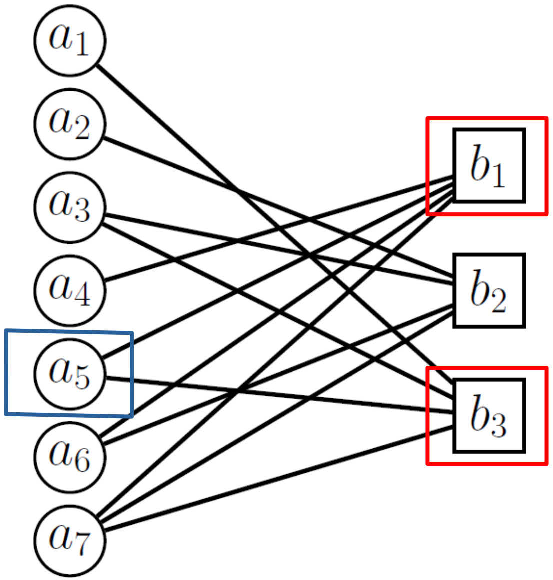


The Tanner Graph of a Code

1. The parity check matrix H defines the bipartite Tanner graph $T(H)$.

- The columns of H define the bit vertices: $A = \{a_1, a_2, \ldots, a_n\}$.
- The rows of H define the check vertices: $B = \{b_1, b_2, \ldots, b_r\}$.
- There exists an edge between a_i and b_j if and only if $H_{i,j} \neq 0$.

$$H = \begin{array}{cc} \begin{array}{ccccccc} & & & & a_5 & & \end{array} & \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} & \begin{array}{c} b_1 \\ \\ b_3 \end{array} \end{array}$$



Recall: Classical Error Correction

$$\underset{x}{\overset{\text{message}}{\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}}} \Rightarrow \underset{x+e=y}{\overset{\text{corruption}}{\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} + \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\\1\\1\\1\end{bmatrix}}} \Rightarrow \underset{s=Hy}{\overset{\text{syndrome}}{\begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix}}}$$

$$\underset{\hat{e}}{\overset{\text{prediction}}{\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}}} \Rightarrow \underset{y+\hat{e}}{\overset{\text{correction}}{\begin{bmatrix}1\\0\\1\\1\\1\end{bmatrix} + \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}}} \Rightarrow \underset{x}{\overset{\text{recovery}}{\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}}}$$

1. Send initial codeword x in C in Z_2^n.
2. Receive corrupted codeword $y = x + e$ in Z_2^n.
3. Make syndrome measurement $s = Hy = He$ in Z_2^r .
4. Decoder predicts an error $\hat{e}$ satisfying $s = H\hat{e}$.
5. Perform error correction $y + \hat{e}$.
6. Recover original codeword if $y + \hat{e} = x$.

Binary Erasure Channel

- **The <u>binary erasure channel</u> erases each bit with probability *p*.**
  - The set of erased bits ε is known.
- **Erasure correction can be achieved using error correction.**
  - Erased bits are assigned random values.

$$\underset{x}{\overset{\text{message}}{\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}}} \Rightarrow \overset{\text{erasure}}{\begin{bmatrix}1\\?\\1\\?\\?\end{bmatrix}} \Rightarrow \underset{y}{\overset{\text{assignment}}{\begin{bmatrix}1\\0\\1\\1\\0\end{bmatrix}}}$$

PART 1.2

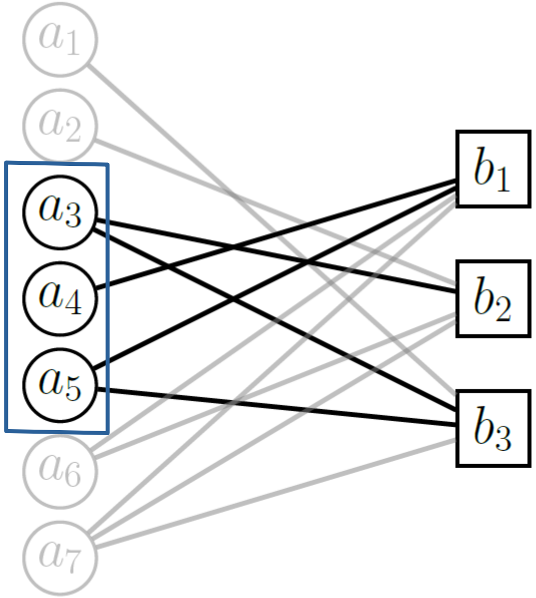
The Peeling Decoder

Erasure-Induced Subgraph of the Tanner Graph

- An erasure ε induces a subgraph of the Tanner graph $T(H)$.
  - **Example**: $\varepsilon = \{a_3, a_4, a_5\}$.
- We can use information about ε to perform correction.
  - Non-erased bits do not have errors.

$$x' = \begin{bmatrix} 1 \\ 1 \\ ? \\ ? \\ ? \\ 1 \\ 0 \end{bmatrix}$$

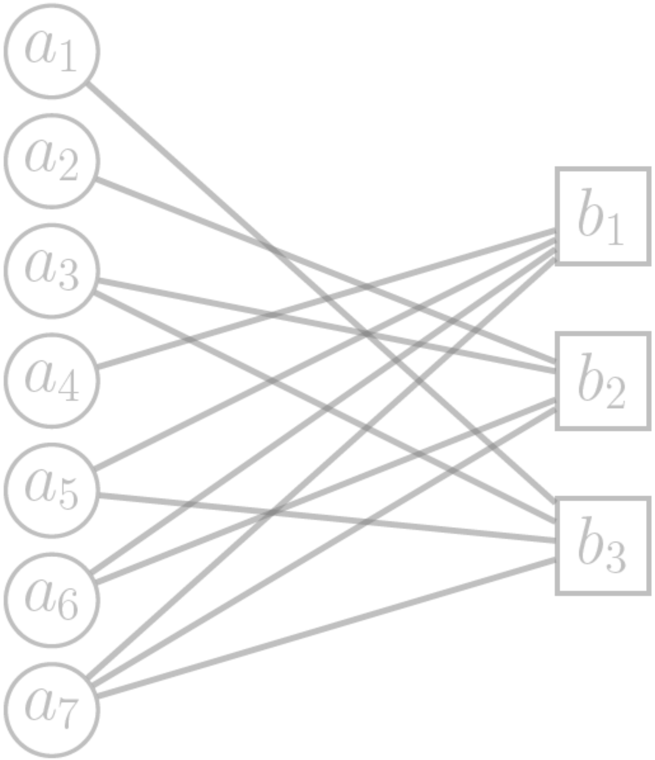
$$H = \begin{array}{c} \begin{array}{ccccccc} & & a_3 & a_4 & a_5 & & \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$$



Algorithm: Peeling Dangling Checks

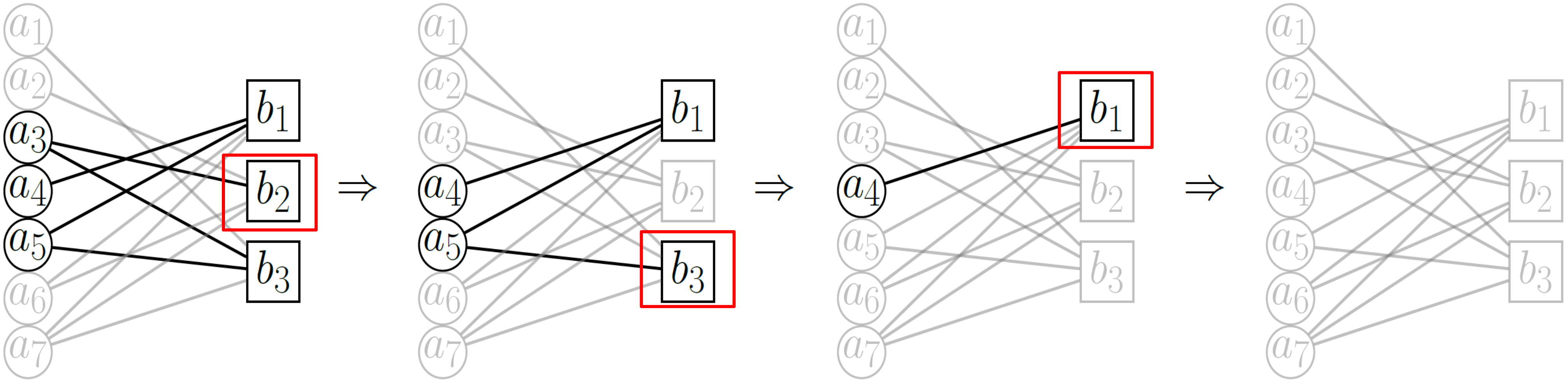
1. Given erasure pattern $\boldsymbol{\varepsilon}$, consider the induced subgraph of *T(H)*.
2. Select a dangling (degree 1) check in this subgraph.
3. Correct the adjacent bit and remove it from $\boldsymbol{\varepsilon}$ (shrinking the subgraph).
4. Algorithm terminates when $\boldsymbol{\varepsilon}$ is empty (or gets stuck in a stopping set).

The complexity of the peeling decoder is linear in the number of bits.



Peeling Decoder: Full Example of a Decoder Success

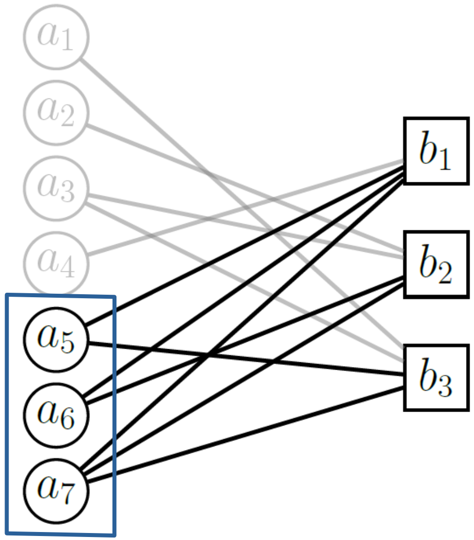
- Decoding success or failure depends <u>only</u> on the erasure-induced subgraph of *T(H)*.
- Success occurs when there exists a sequence of dangling checks that fully “peel” ***ε***.



Stopping Sets for the Peeling Decoder

1. An erasure-induced subgraph of *T(H)* with no dangling checks is a <u>stopping set</u> for the peeling decoder (the decoder fails).
2. Tanner graphs for sparse codes generally have fewer stopping sets.

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} & & & & a_5 & a_6 & a_7 \end{array} \\
H = & \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}
\end{array}
$$



Quantum Code Review:
Hypergraph Product Codes

Review: Families of Quantum Codes

1. **Recall that a <u>quantum code</u> of length N and dimension K is a subspace of a Hilbert space.**
   - Vectors are N-qubit states $|\psi\rangle$ in $\mathbf{C}^N$.
   - Errors on a state $|\psi\rangle$ are described by N-qubit Pauli operators in $P_N = \{I,X,Z,Y\}^{\times N}$.
2. **<u>Stabilizer Codes</u> are the space of states left fixed by a subgroup of the Pauli group P_N.**
3. **<u>CSS Codes</u> are stabilizer codes defined by commuting N-qubit X- and Z-Pauli operators.**
   - X- and Z-Pauli <u>stabilizer generators</u> define the rows of matrices H_X and H_Z (where $H_X H_Z^T = 0$).
   - CSS Z and X error correction is modeled using classical codes $C_X = \ker(H_X)$ and $C_Z = \ker(H_Z)$.
4. **<u>Surface Codes</u> are CSS codes defined from the cellulation of a surface.**
5. **<u>Hypergraph Product Codes</u> are another type of CSS code.**

Review: Pauli Errors for CSS Codes

- Pauli errors X_i and Z_i in P_N can be mapped onto binary strings e_i in Z_2^N.

$$E = X_1 Z_1 X_2 Z_4 \in P_4 \quad \Leftrightarrow \quad e_X = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad e_Z = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

- Pauli error correction for a CSS code can be modeled as classical error correction using H_X or H_Z (handling X and Z errors separately).
- The peeling decoder algorithm can be directly applied to CSS codes.

Hypergraph Product Codes: Definition

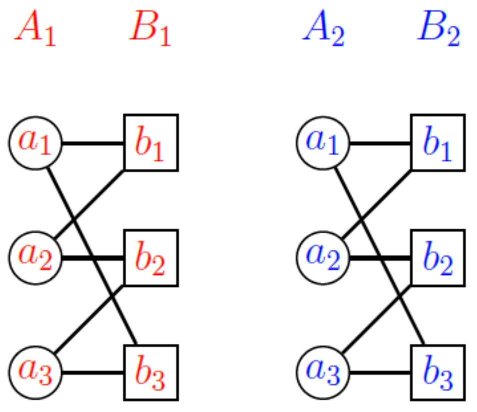
Theorem (Tillich-Zémor):

The Hypergraph Product (HGP) code of two classical codes $C_1 = \ker(H_1)$ and $C_2 = \ker(H_2)$ is the quantum code $C = \mathrm{CSS}(C_X, C_Z)$, where $C_X = \ker(H_X)$ and $C_Z = \ker(H_Z)$ have parity check matrices H_X and H_Z defined from H_1 and H_2 as follows.

$$\begin{aligned} H_X &= \left[\; H_1 \otimes I \;\middle|\; I \otimes H_2^T \;\right] \\ H_Z &= \left[\; I \otimes H_2 \;\middle|\; H_1^T \otimes I \;\right] \end{aligned}$$

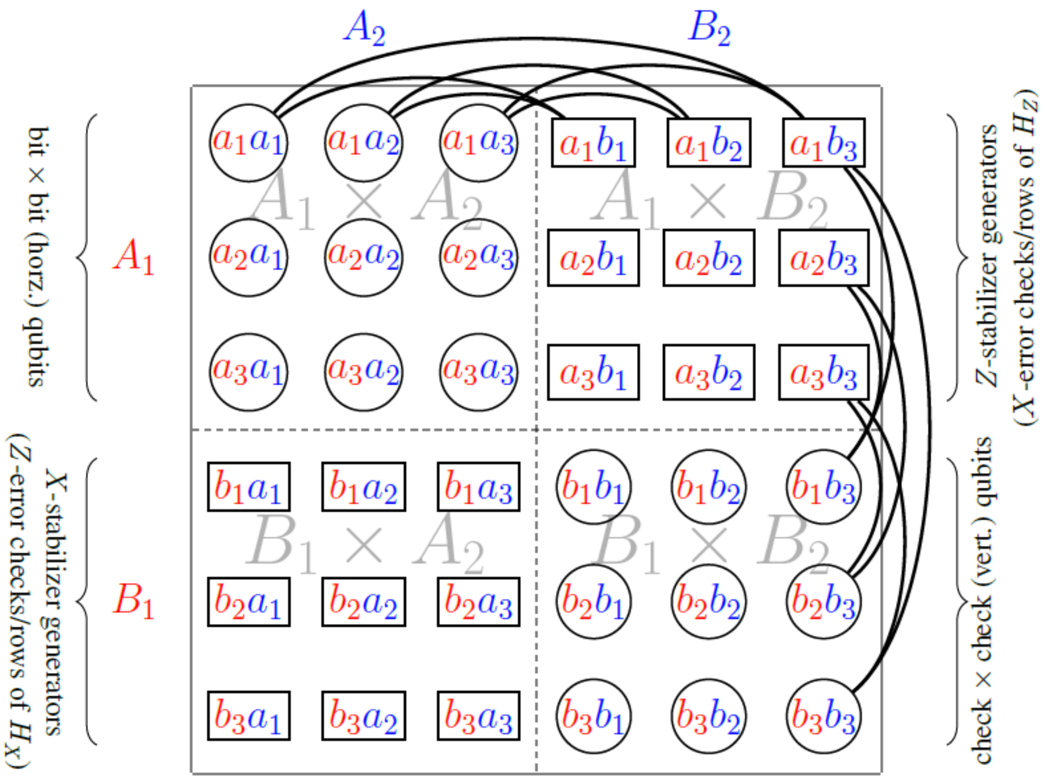
- The matrices have sizes $H_1 = [\, r_1 \times n_1 \,]$, $H_2 = [\, r_2 \times n_2 \,]$, thus $H_X = [\, r_1 n_2 \times (n_1 n_2 + r_1 r_2) \,]$, $H_Z = [\, r_2 n_1 \times (n_1 n_2 + r_1 r_2) \,]$.
- C has length $N = n_1 n_2 + r_1 r_2$ and dimension $K = N - \mathrm{rank}(H_X) - \mathrm{rank}(H_Z)$
- C has minimum distance $\min(d_1, d_2)$, where d_1 and d_2 are the minimum distances of C_1 and C_2.

Hypergraph Product Codes: Tanner Graph Structure



$$T(H_1) \cong T(H_2)$$

$$H_1 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = H_2$$



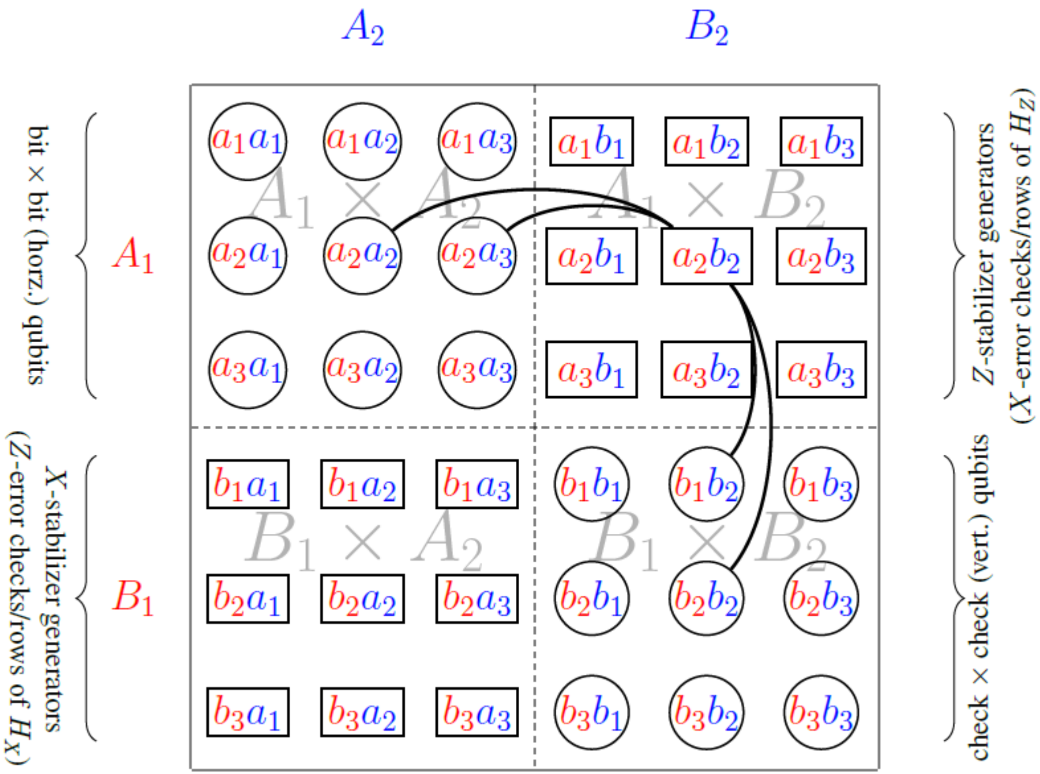
$$H_Z = \begin{bmatrix} 110000000 & 100000100 \\ 011000000 & 010000010 \\ 101000000 & 001000001 \\ 000110000 & 100100000 \\ 000011000 & 010010000 \\ 000101000 & 001001000 \\ 000000110 & 000100100 \\ 000000011 & 000010010 \\ 000000101 & 000001001 \end{bmatrix}$$

$$[\ I \otimes H_2 \mid H_1^T \otimes I\ ]$$

$$H_X = \begin{bmatrix} 100100000 & 101000000 \\ 010010000 & 110000000 \\ 001001000 & 011000000 \\ 000100100 & 000101000 \\ 000010010 & 000110000 \\ 000001001 & 000011000 \\ 100000100 & 000000101 \\ 010000010 & 000000110 \\ 001000001 & 000000011 \end{bmatrix}$$

$$[\ H_1 \otimes I \mid I \otimes H_2^T\ ]$$

Hypergraph Product Codes: *Z*-type Stabilizer Generators



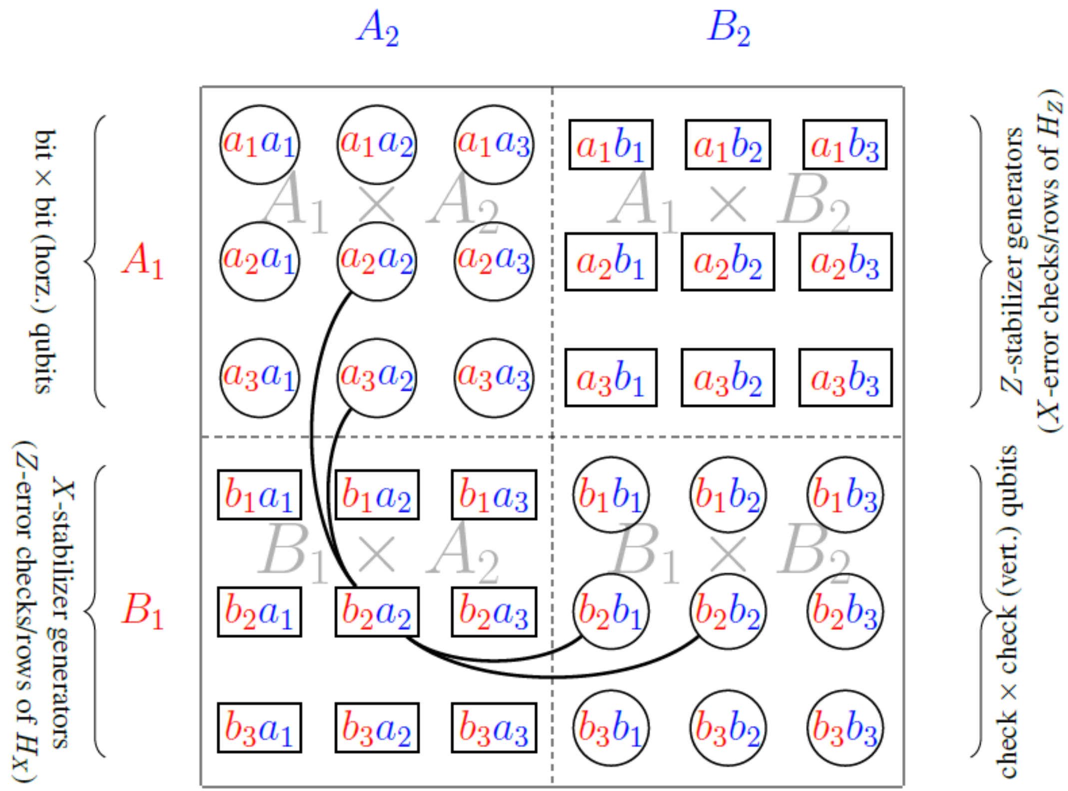
$$
\begin{array}{cc}
 & \qquad a_2a_2 \;\; a_2a_3 \qquad\qquad b_1b_2 \;\; b_2b_2 \\
H_Z = & \begin{bmatrix}
110000000 & 100000100 \\
011000000 & 010000010 \\
101000000 & 001000001 \\
000110000 & 100100000 \\
\boxed{000011000} & \boxed{010010000} \\
000101000 & 001001000 \\
000000110 & 000100100 \\
000000011 & 000010010 \\
000000101 & 000001001
\end{bmatrix} \quad a_2b_2
\end{array}
$$

$$\left[\; I \otimes H_2 \;\middle|\; H_1^T \otimes I \;\right]$$

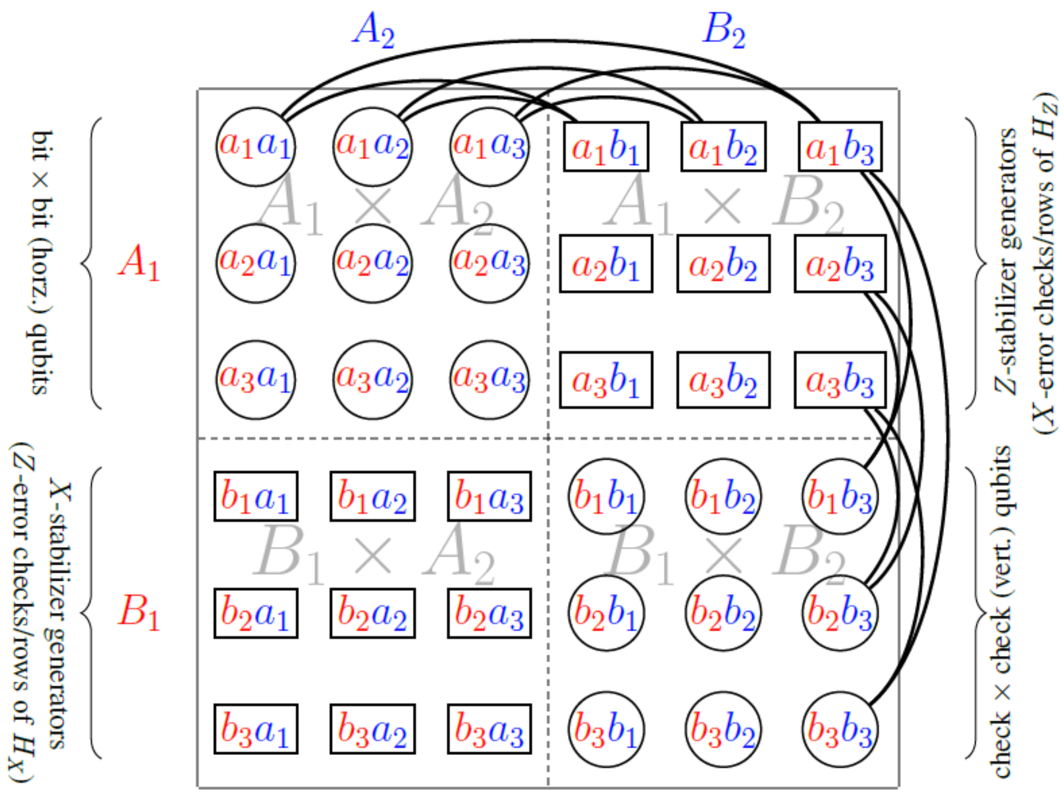
Hypergraph Product Codes: X-type Stabilizer Generators

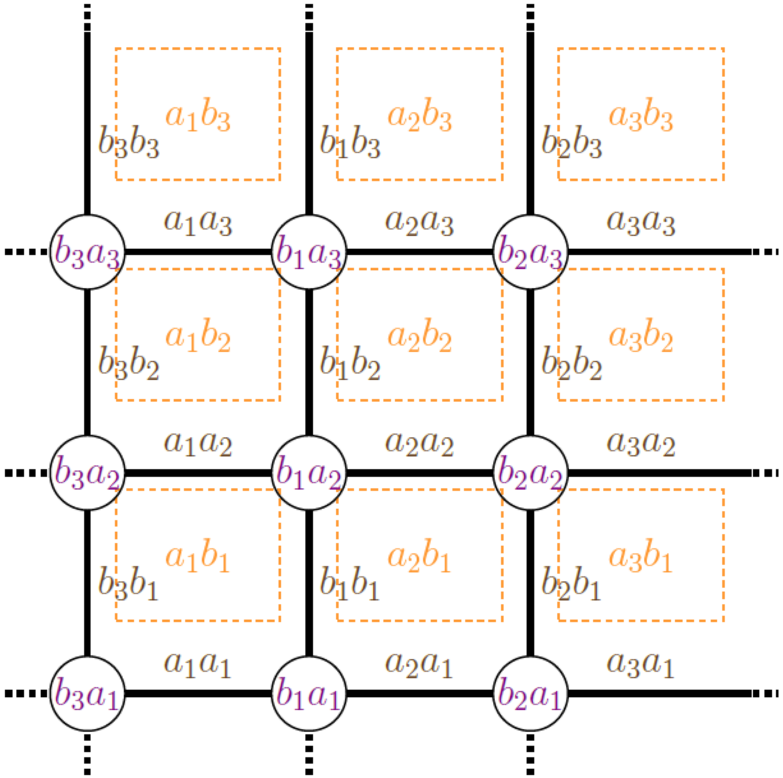
$$
\begin{array}{ccc}
 & a_2a_2 \quad a_3a_2 & b_2b_1 \quad b_2b_2 & \\
H_X = & \left[\begin{array}{cc}
100100000 & 101000000 \\
010010000 & 110000000 \\
001001000 & 011000000 \\
000100100 & 000101000 \\
\boxed{000010010} & \boxed{000110000} \\
000001001 & 000011000 \\
100000100 & 000000101 \\
010000010 & 000000110 \\
001000001 & 000000011
\end{array}\right] & \\
\end{array}
\quad b_2a_2
$$

$$\left[\; H_1 \otimes I \;\middle|\; I \otimes H_2^T \; \right]$$



Aside: Toric Code HGP Picture versus Lattice Picture

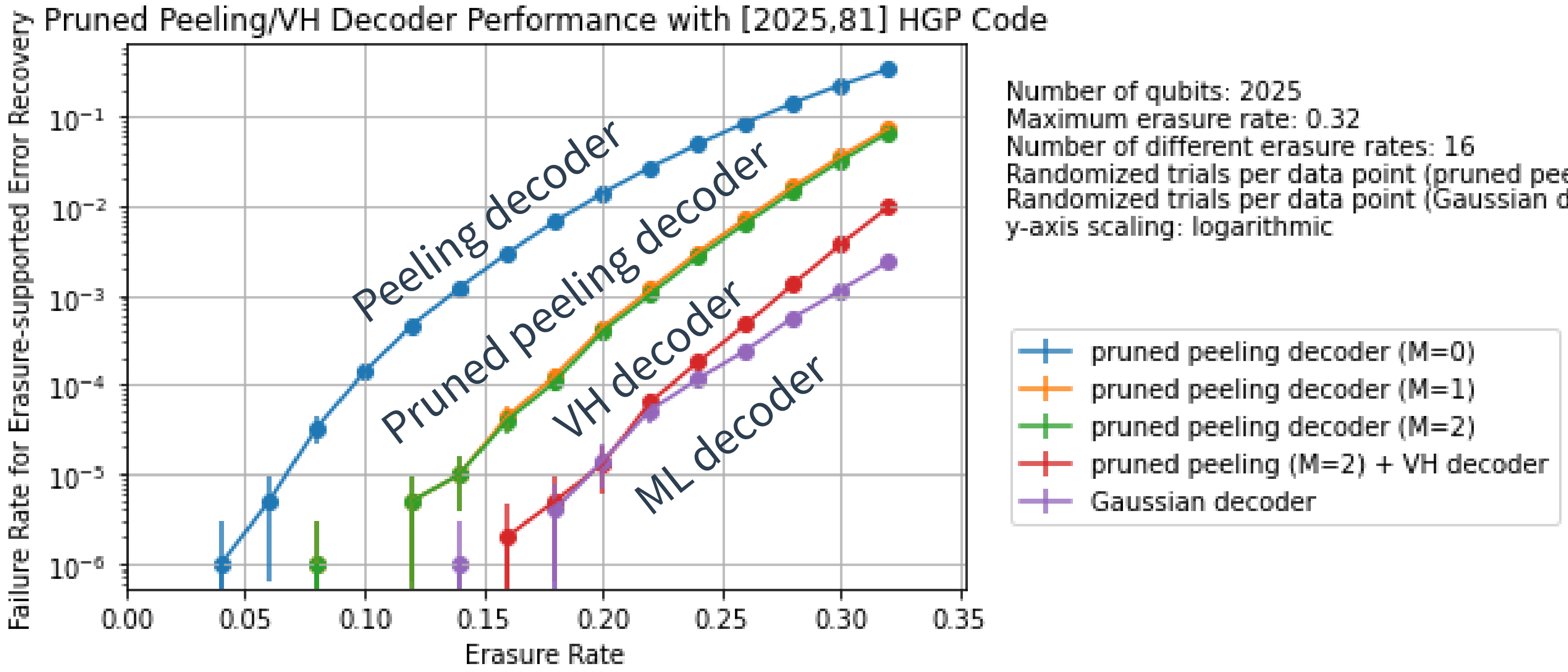




PART 2.1

Generalized Peeling Decoder for HGP Codes

Comparison of Performance with Gaussian (ML) Decoder

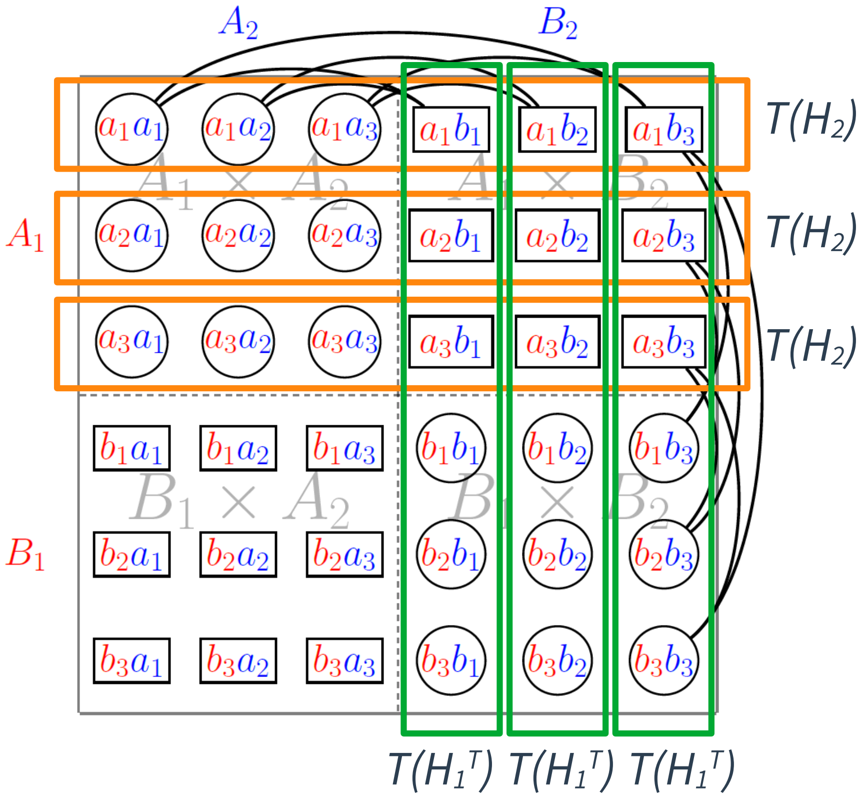


Number of qubits: 2025
Maximum erasure rate: 0.32
Number of different erasure rates: 16
Randomized trials per data point (pruned peeling and VH decoder): 10^6
Randomized trials per data point (Gaussian decoder): 10^6
y-axis scaling: logarithmic

Classical Subgraphs in HGP Tanner Graph

- The Tanner graph $T(H_Z)$ contains copies of $T(H_2)$ and $T(H_1^T)$ as subgraphs.
  - Horizontal copies of $T(H_2)$.
  - Vertical copies of $T(H_1^T)$.
- Stopping sets for $T(H_2)$ and $T(H_1^T)$ lift to stopping sets of $T(H_Z)$ on a row or column.
- These are horizontal and vertical <u>classical stopping sets</u> for $T(H_Z)$ in the HGP code.

$$H_Z = \left[\ I \otimes \boxed{H_2} \ \middle|\ \boxed{H_1^T} \otimes I\ \right]$$

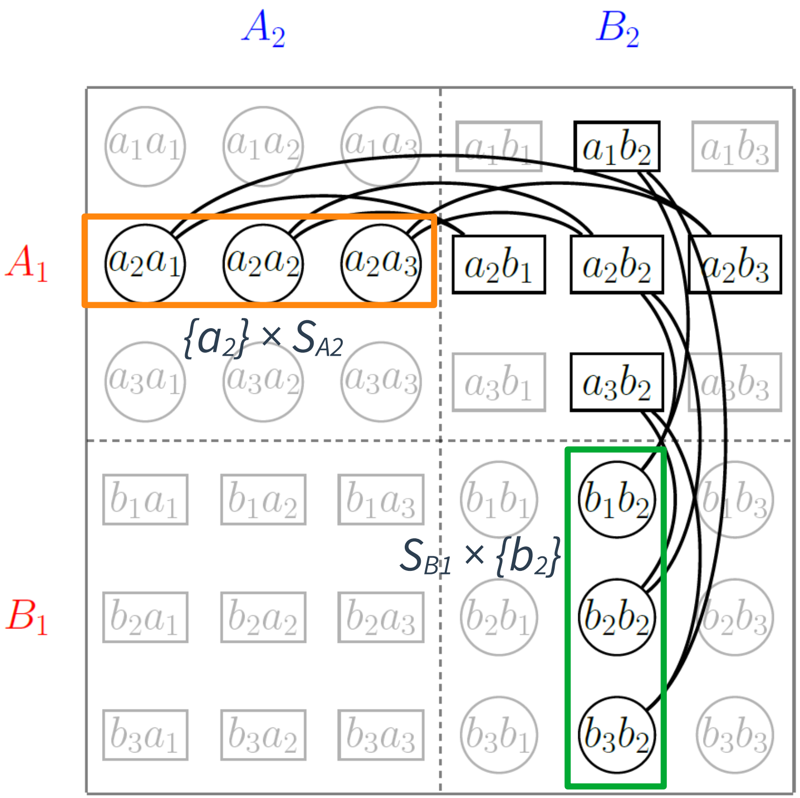


Formalizing Horizontal and Vertical Stopping Sets

- The HGP Tanner graph $T(H_Z)$ is the product of two bipartite classical Tanner graphs.
  - $T(H_1) = (A_1$ u $B_1, E_1)$
  - $T(H_2) = (A_2$ u $B_2, E_2)$
- Classical stopping sets in $T(H_Z)$ can be decomposed into classical components.
  - Horizontal: $\{a_i\} \times S_{A2}$ in $A_1 \times A_2$

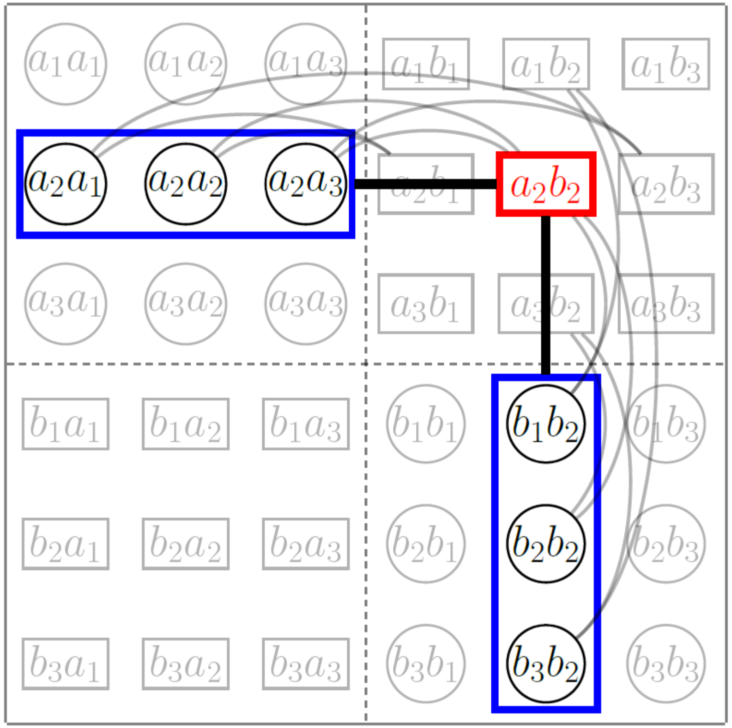
    S_{A2} is a stopping set of $T(H_2)$
  - Vertical: $S_{B1} \times \{b_j\}$ in $B_1 \times B_2$

    S_{B1} is a stopping set of $T(H_1^T)$



The Vertical-Horizontal (VH) Graph

- **Given an erasure pattern ε, define the vertical-horizontal graph as follows.**
  - Vertices are clusters of erased qubits in the same connected component and row/column of $T(H_Z)$.
  - There exists an edge between clusters if there exists a check in $T(H_Z)$ adjacent to a qubit in each.
- **The VH graph is closely related to the erasure-induced subgraph of $T(H_Z)$.**
  - Any two clusters share at most one check (edge).
  - There does not exist an edge between two clusters of the same type (horizontal or vertical).



Considerations for the VH Decoder

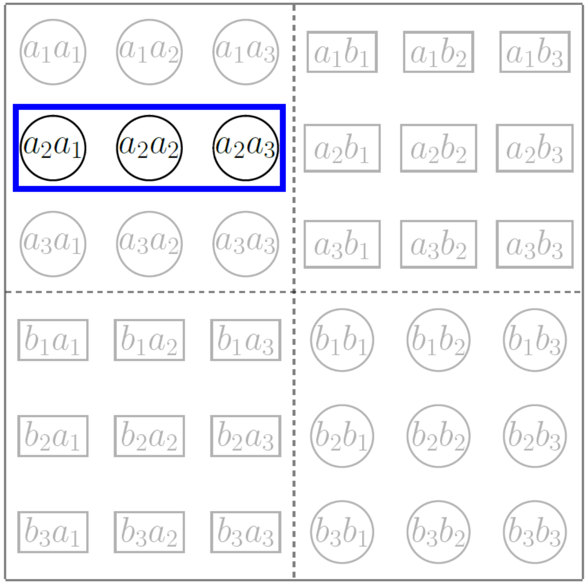
OBSERVATIONS

- The ML decoder uses Gaussian elimination, which is slow (cubic complexity).
- The (pruned) peeling decoder is faster, but performs poorly for HGP codes.
- Numerically, we see many small classical stopping sets in the VH graph.

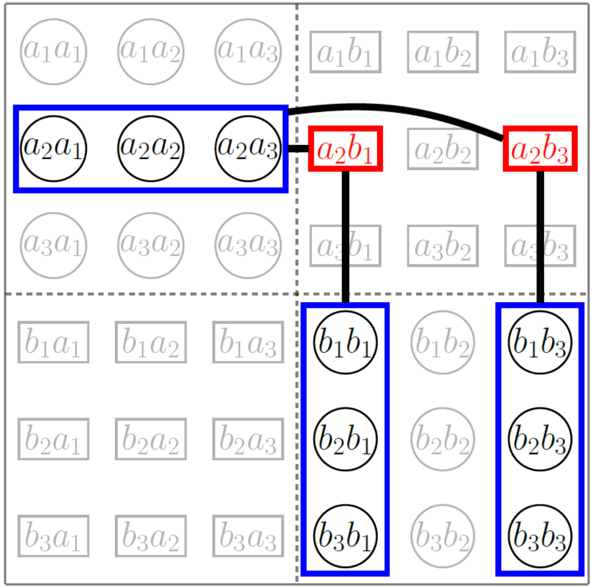
STRATEGY

- Design a decoder combing best parts of pruned peeling and ML decoders.
- Peel until stuck, then apply Gaussian elimination on classical stopping sets.
  - VH clusters have size $O(\sqrt{N})$, so Gaussian decoder contributes $O(N^{1.5})$ per cluster.
  - Number of clusters only grows as $O(\sqrt{N})$.

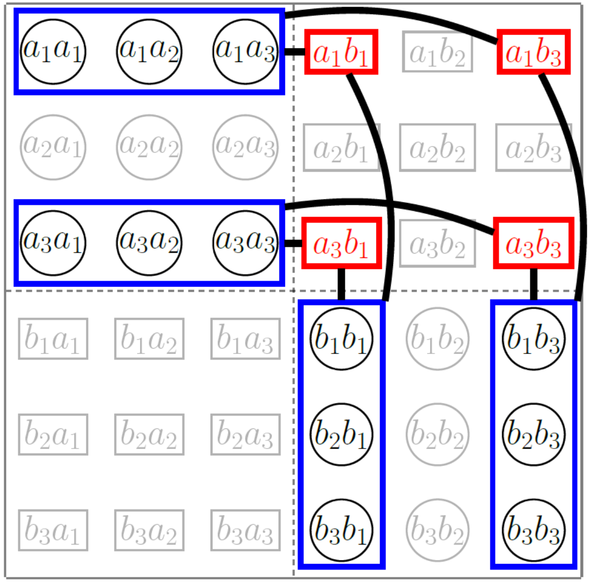
Types of Cluster Configurations in the VH Graph



Isolated Cluster



Cluster Tree



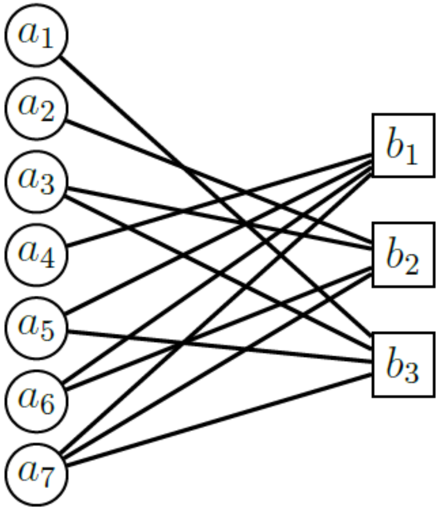
Cluster Cycle

Example: Failure of the Naive VH Decoder

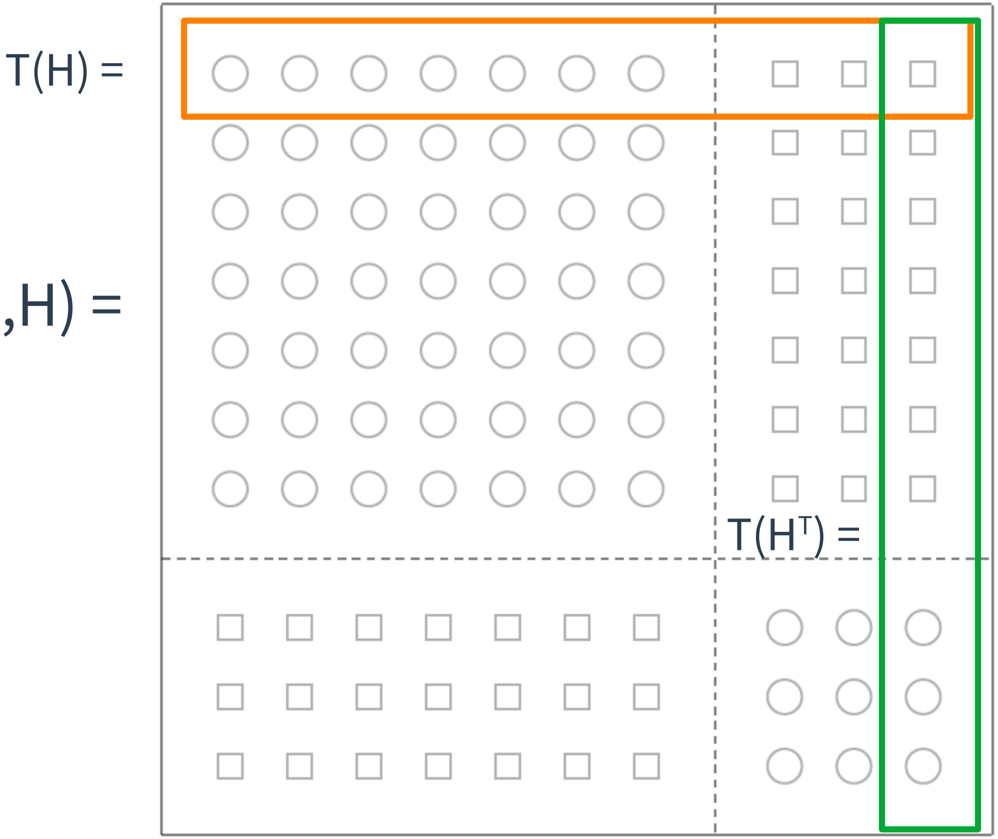
Visualizing the HGP Code from the [7,3] Hamming Code

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

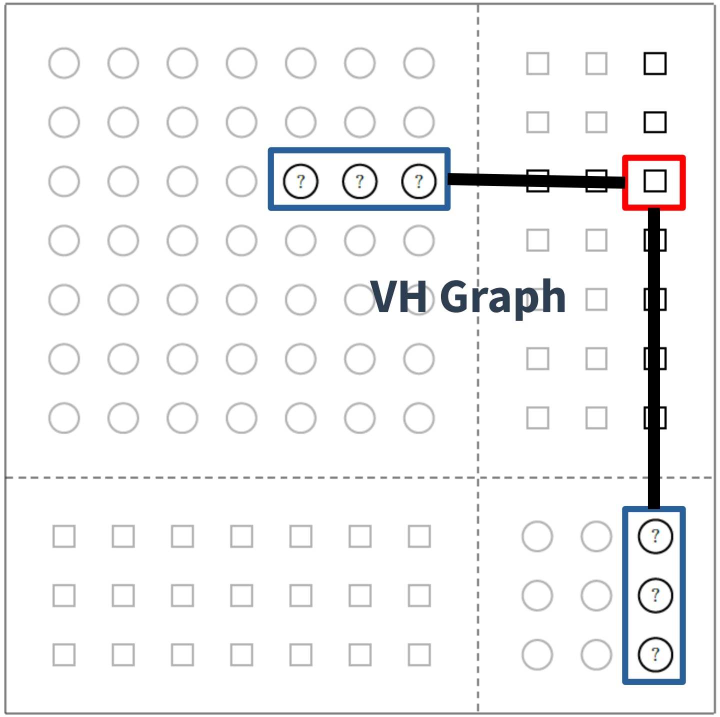
$T(H) =$



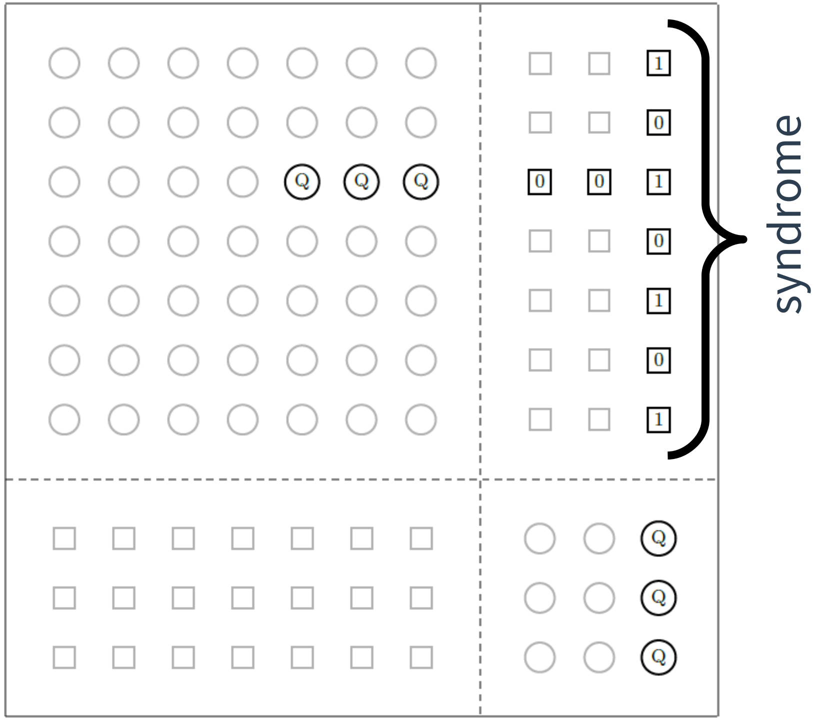
HGP(H,H) =



An Erasure Pattern and Random Erasure-Supported Error

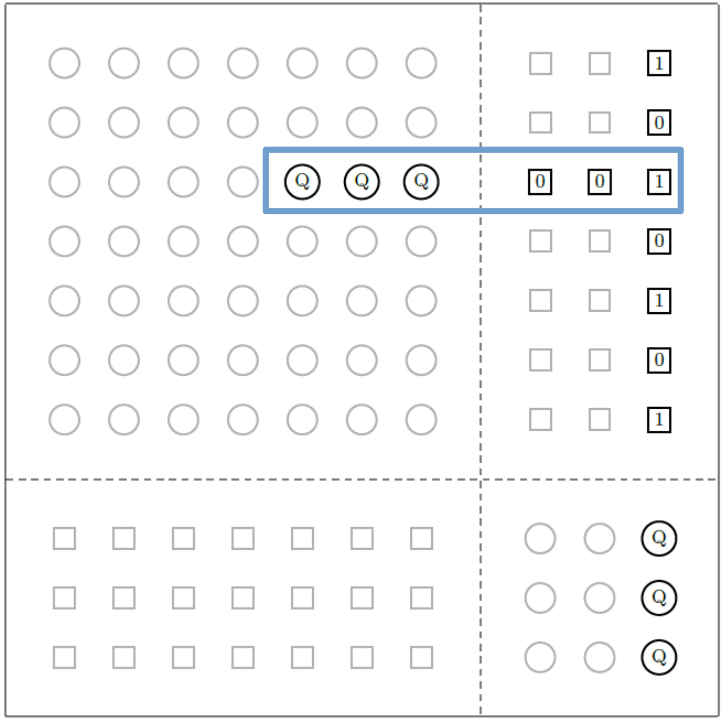


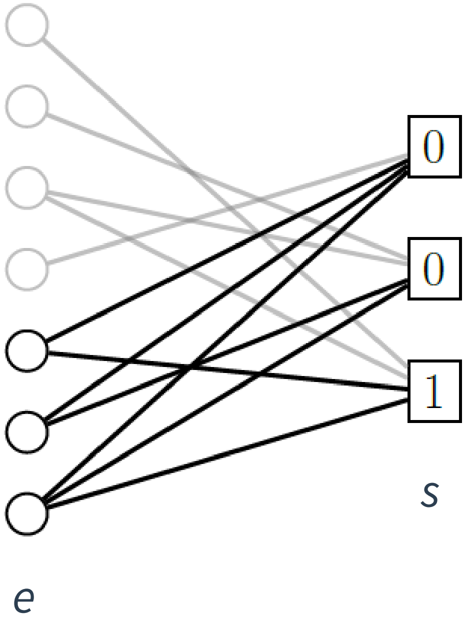
Erasure Pattern



Random Erasure-Supported Error

Naive VH Decoder: Gaussian Elimination on First Cluster



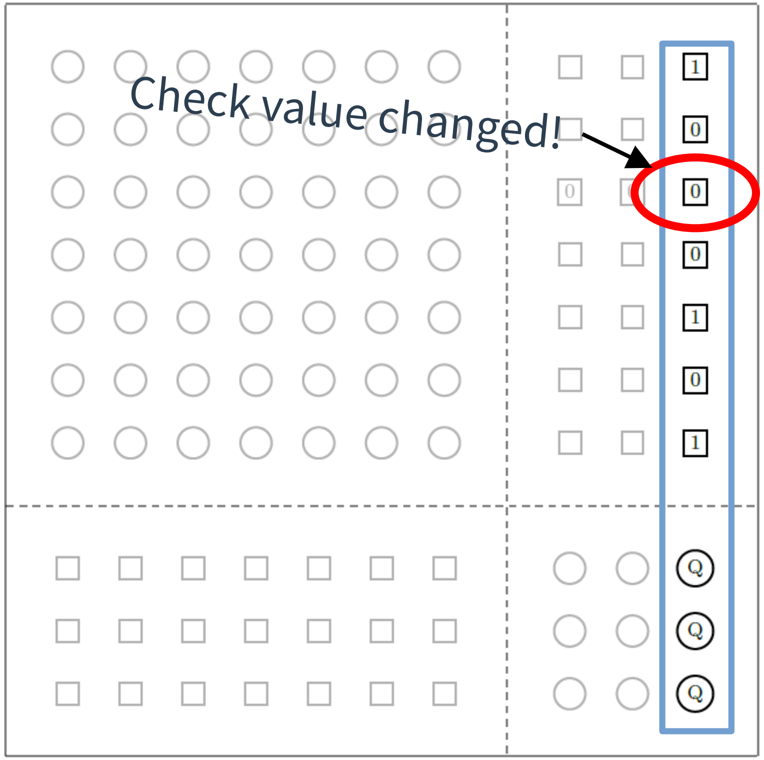


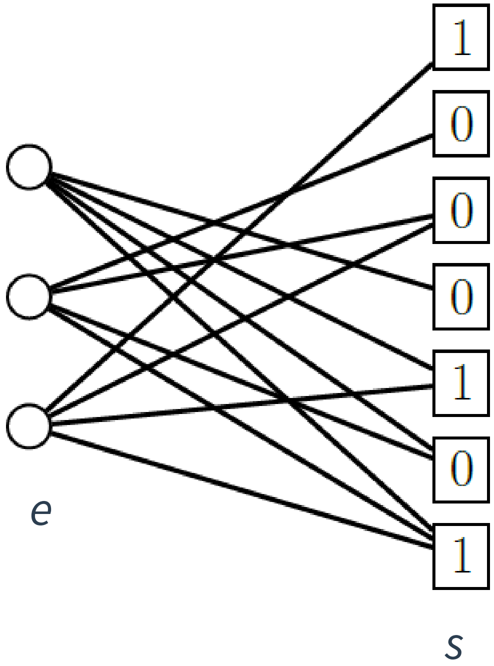
$$H = \begin{array}{c} \begin{array}{ccccccc} & & & & & H' & \end{array} \\ \left[\begin{array}{cccc|ccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right] \end{array}$$

Solve: $H'e = s$

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}}_{H'} \underbrace{\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}}_{e} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{s}$$

Naive VH Decoder: Gaussian Elimination on Second Cluster





$$H^T = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$H^T e = s$$

Problem: there is no solution!

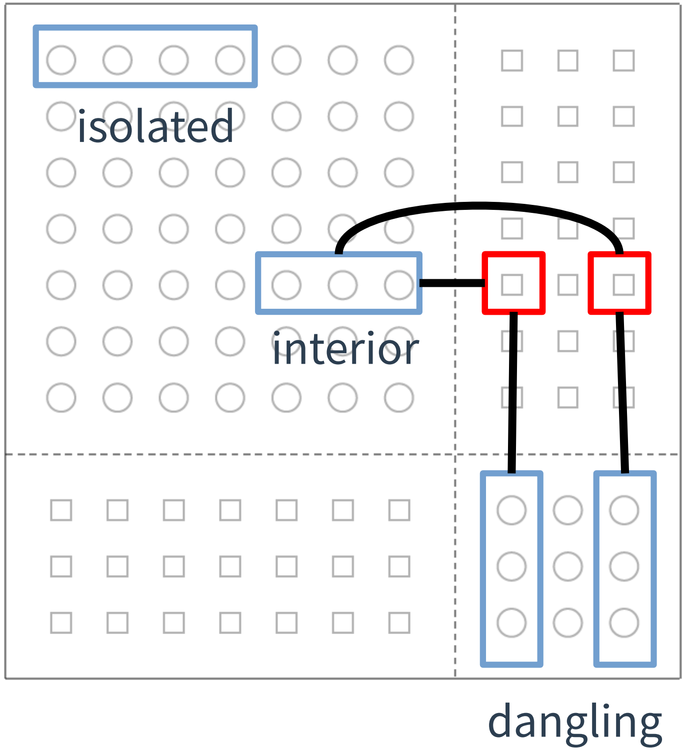
Fixing the Naive VH Decoder

PROBLEM

- Connected VH clusters share a syndrome node.
- Local cluster solutions might be incompatible.

MODIFICATION

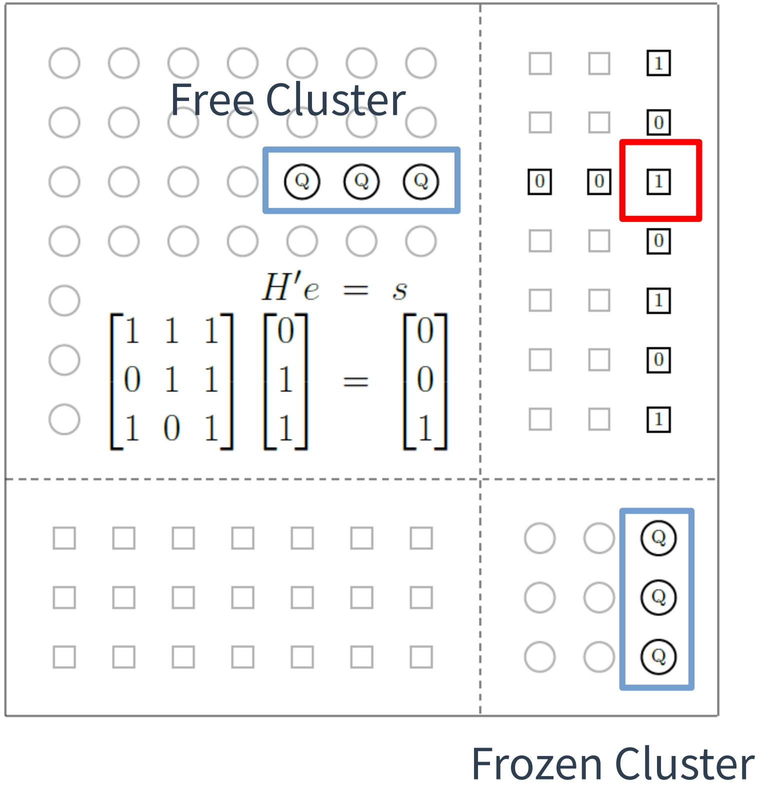
- Distinguish between cluster types in VH Graph.
  - Isolated Clusters
  - Interior Clusters
  - Dangling Clusters (*Free* versus *Frozen*)
- Solve clusters in order*.



The VH Decoder

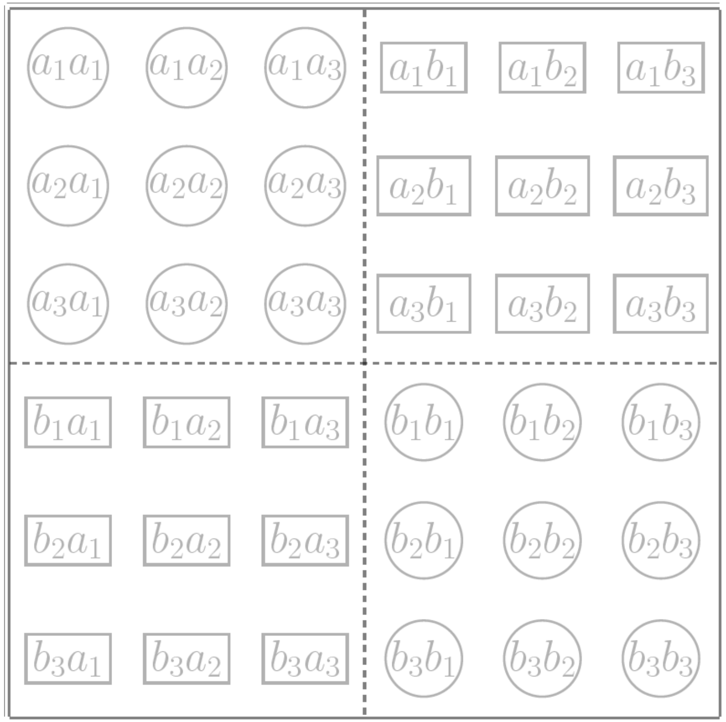
A Technical Condition on Dangling Clusters: *Free* vs. *Frozen*

- **CONNECTING CHECKS**
  - A cluster is <u>free</u> if it contains a vector with (weight 1) syndrome on the connecting check.
  - Otherwise, a cluster is <u>frozen</u>.
- **FREE CLUSTERS**
  - Enables flipping connecting check's value.
  - Can safely be solved after other clusters.
- **FROZEN CLUSTERS**
  - Solvable like isolated clusters (all solutions have same effect on connecting check).



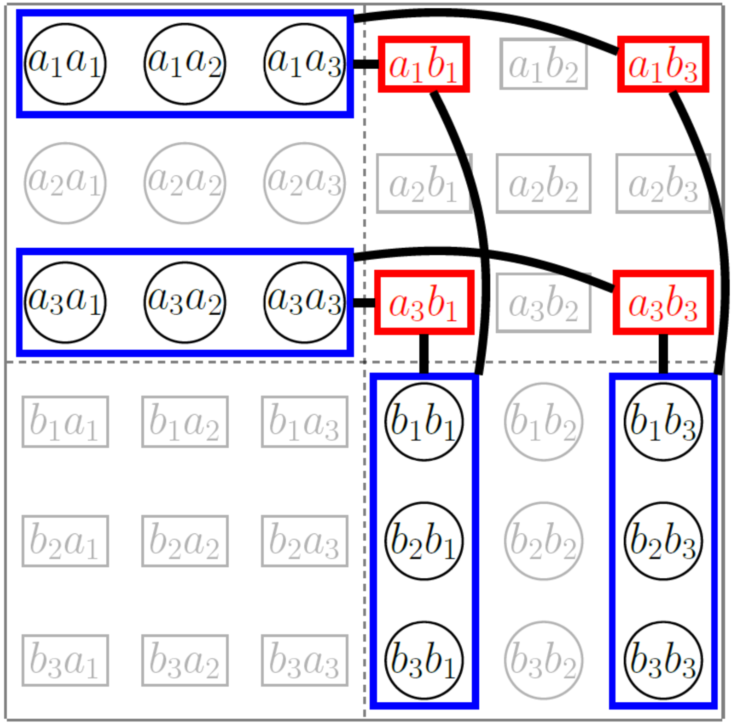
Subroutine: Solving VH Graph Clusters in Sequence

1. Identify isolated, interior, and dangling clusters in the VH graph.
2. Solve and remove all isolated clusters.
3. Within the remaining VH graph, identify all dangling clusters as *free* or *frozen*.
4. Solve and remove all frozen clusters.
5. Remove all free dangling clusters from the VH graph, but wait until end to solve these.
6. Repeat above steps until VH graph contains no isolated nor dangling clusters.
7. Solve prior removed free clusters (in reverse).

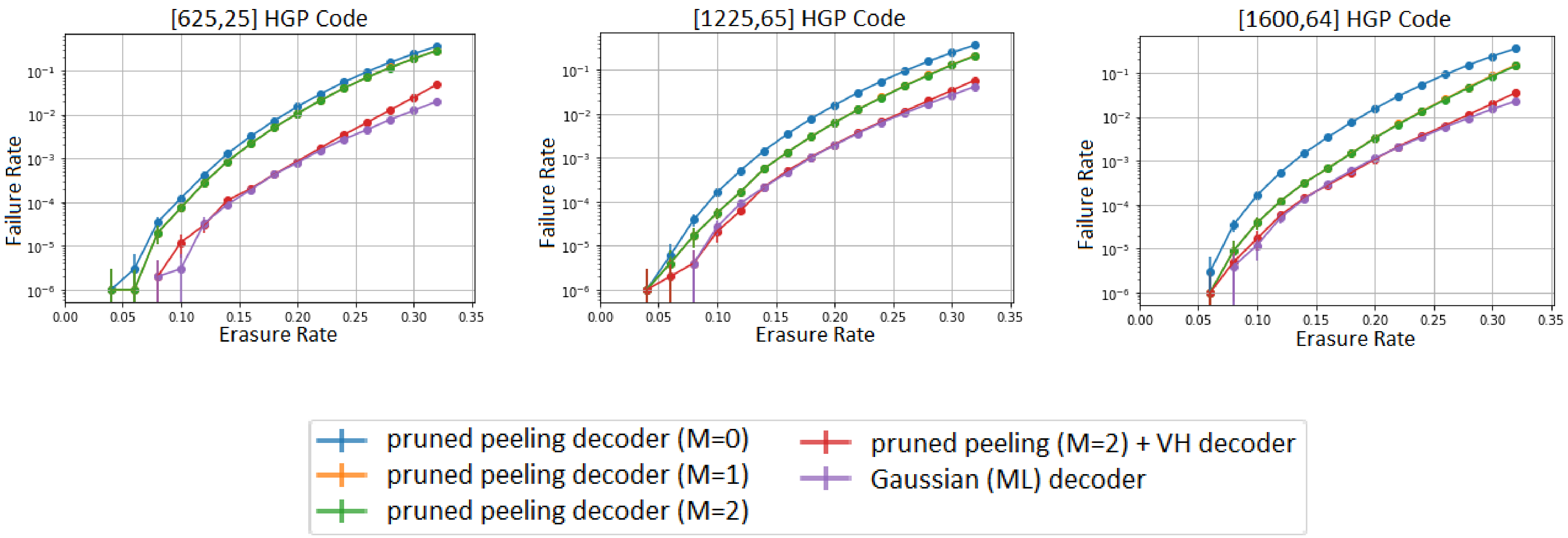


Full Algorithm: Vertical-Horizontal (VH) Decoder

1. **Given erasure pattern ε, apply the pruned-peeling decoder until stuck in a stopping set.**
2. **Compute the VH-graph of ε.**
3. **Apply the preceding VH graph subroutine.**
    1. If there exist <u>isolated clusters</u>, solve using Guassian elimination, then lift solution.
    2. If there exist <u>dangling clusters</u>, identify *free* and *frozen*, solve in sequence, then lift solution.
4. **Repeat these steps until the VH graph is either empty or stuck in a VH stopping set.**
    - For example, a cycle of clusters in the VH graph.



Performance Comparisons for Other Examples of HGP Codes



Computational Complexity of the combined Decoder

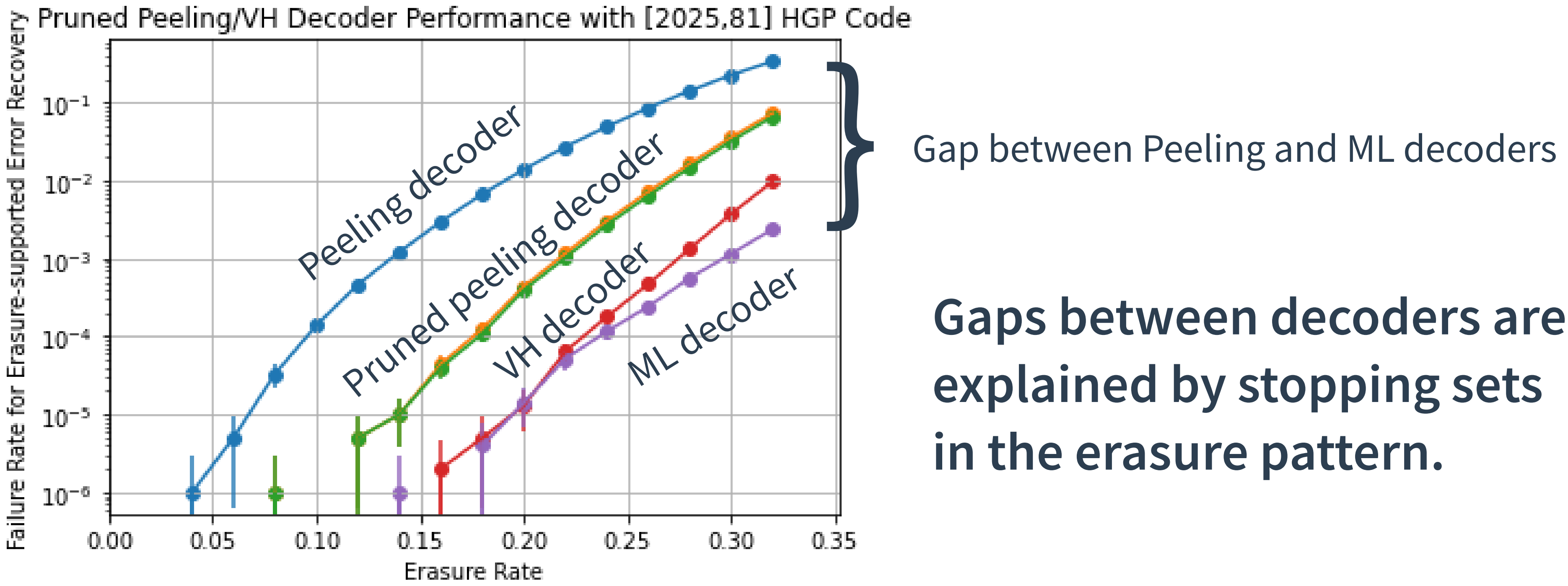
The computational complexity of combined pruned peeling and VH decoders is dominated by Gaussian elimination applied to clusters.

- Clusters in the VH graph have size $O(\sqrt{N})$, where N is the HGP code length.
- On a single cluster, cubic-complexity Gaussian decoder contributes $O(N^{1.5})$.
- The number of possible clusters grows as $O(\sqrt{N})$.
- Across all clusters, the VH-decoder has complexity $O(N^2)$.
- With a probabilistic implementation of the Gaussian decoder, this can be further reduced to $O(N^{1.5})$ in total.

PART 3.1 (time allowing)

BONUS: The Pruned Peeling Decoder

Numerical Comparison of Decoder Performance

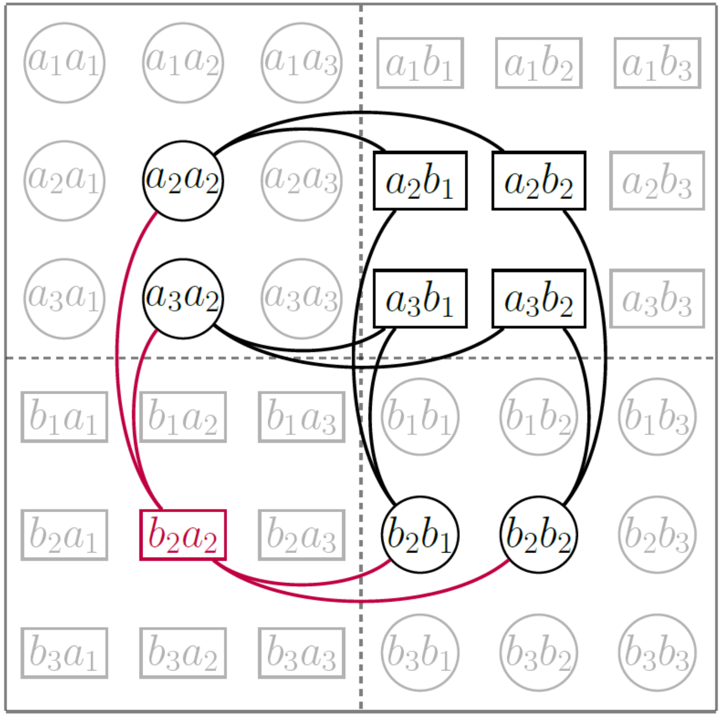


Gaps between decoders are explained by stopping sets in the erasure pattern.

Visualizing Stabilizer Support in the HGP Tanner Graph

a_2a_2 $\quad a_3a_2 \quad b_2b_1 \quad b_2b_2$

$$H_X = \begin{array}{cl} \begin{bmatrix} 100100000 & 101000000 \\ 010010000 & 110000000 \\ 001001000 & 011000000 \\ 000100100 & 000101000 \\ 000010010 & 000110000 \\ 000001001 & 000011000 \\ 100000100 & 000000101 \\ 010000010 & 000000110 \\ 001000001 & 000000011 \end{bmatrix} & \begin{array}{l} \\ \\ \\ \\ b_2a_2 \\ \\ \\ \\ \\ \end{array} \end{array}$$



X-stabilizer generators (rows of H_X)

a_2a_2 $\quad a_3a_2 \quad b_2b_1 \quad b_2b_2$

$$H_Z = \begin{array}{cl} \begin{bmatrix} 110000000 & 100000100 \\ 011000000 & 010000010 \\ 101000000 & 001000001 \\ 000110000 & 100100000 \\ 000011000 & 010010000 \\ 000101000 & 001001000 \\ 000000110 & 000100100 \\ 000000011 & 000010010 \\ 000000101 & 000001001 \end{bmatrix} & \begin{array}{l} \\ \\ \\ a_2b_1 \\ a_2b_2 \\ \\ a_3b_1 \\ a_3b_2 \\ \\ \end{array} \end{array}$$

Stabilizer Stopping Sets for the Peeling Decoder

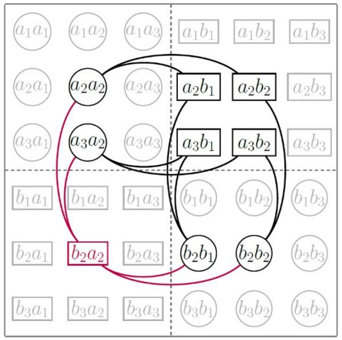
The qubit support of an *X*-type stabilizer is a stopping set for the Tanner graph $T(H_Z)$.

PROOF

- Each X-type stabilizer commutes with Z-type stabilizer generators by construction $(H_Z H_X^T = 0)$.
- The binary representation of an X-stabilizer is a codeword for the classical code $C = \ker(H_Z)$.
- Each row of H_Z (Z generator) is adjacent to an even number of qubits in the support of the X-stabilizer.
- The subgraph induced by this support contains no degree 1 checks (hence, it is a stopping set).

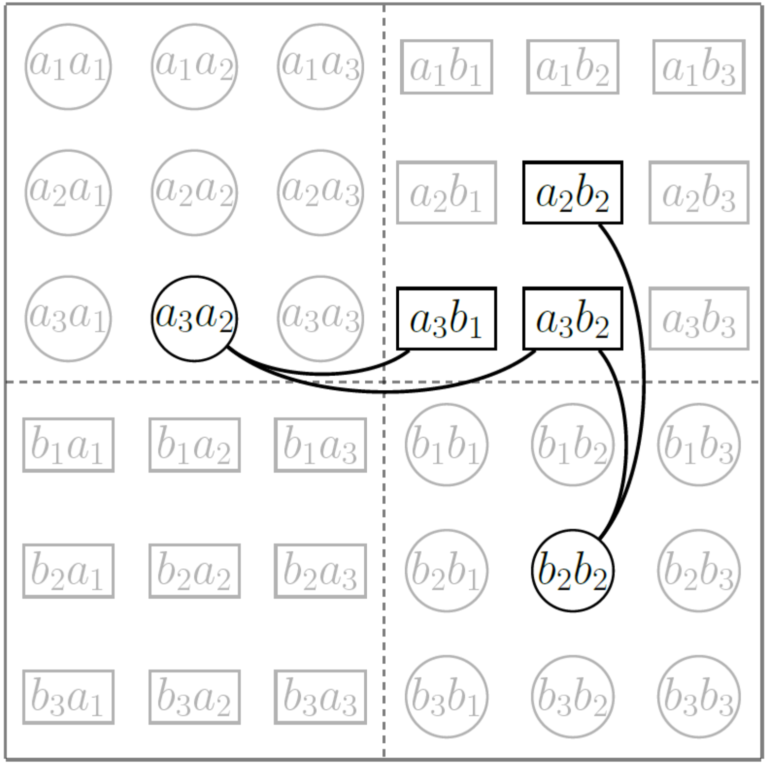
$$H_X = \begin{bmatrix} 100100000 & 101000000 \\ 010010000 & 110000000 \\ 001001000 & 011000000 \\ 000100100 & 000101000 \\ \boxed{000010010 \quad 000110000} \\ 000001001 & 000011000 \\ 100000100 & 000000101 \\ 010000010 & 000000110 \\ 001000001 & 000000011 \end{bmatrix} \quad e_X \qquad e_X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$$

$$H_Z e_X = \begin{bmatrix} 110000000 & 100000100 \\ 011000000 & 010000010 \\ 101000000 & 001000001 \\ 000110000 & 100100000 \\ 000011000 & 010010000 \\ 000101000 & 001001000 \\ 000000110 & 000100100 \\ 000000011 & 000010010 \\ 000000101 & 000001001 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



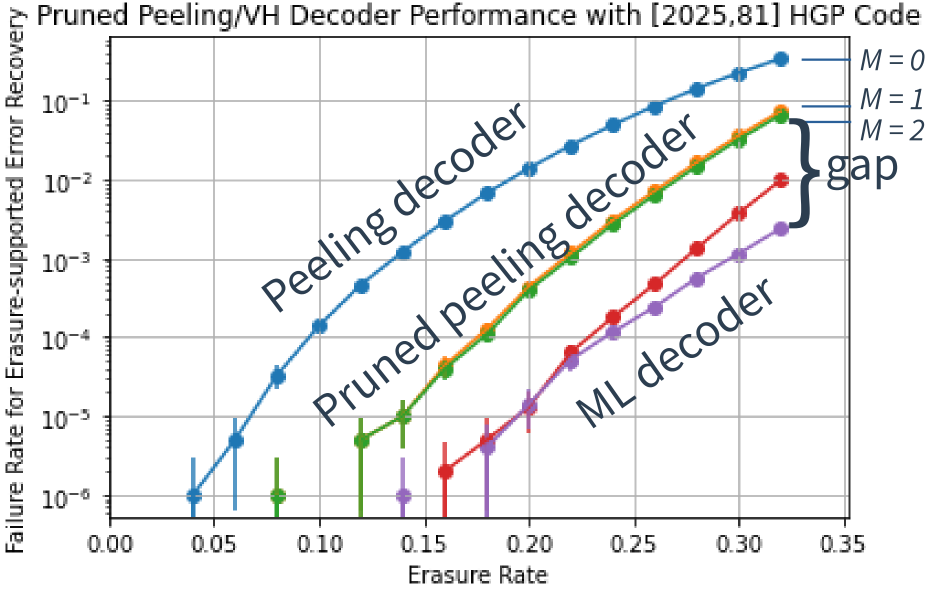
Algorithm: Pruned Peeling Decoder

1. Given erasure pattern ε, apply the standard peeling decoder until stuck.
2. Check whether ε contains the qubit-support *S* of a stabilizer.
   - If so, then *S* is a stabilizer stopping set.
3. Break *S* by removing some qubit from the erasure ε, shrinking the subgraph.
   - This is possible since errors are corrected up to multiplication by a stabilizer.
4. Continue with the peeling decoder.

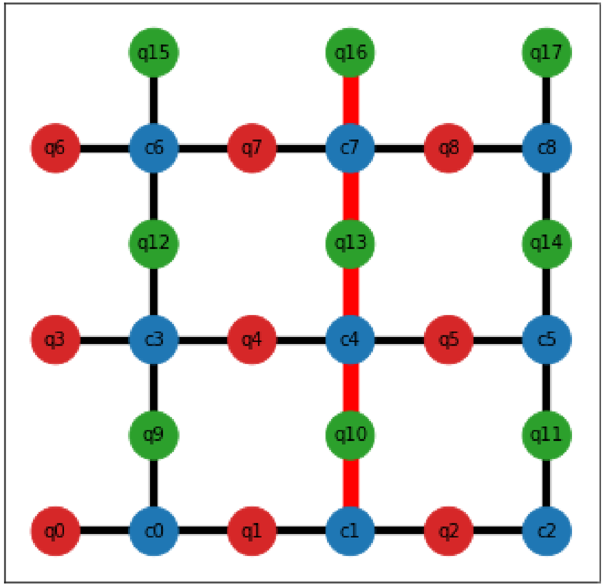


Restrictions on Searching for Stabilizer Stopping Sets

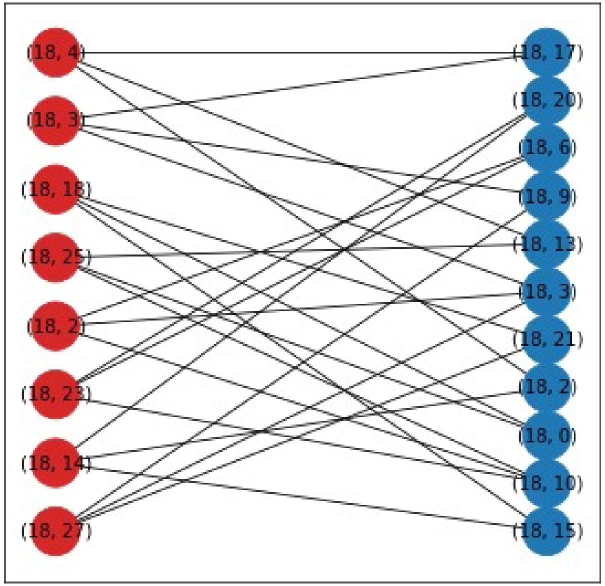
1. **Any product of *M* stabilizer generators defines a valid X-stabilizer.**
   - Naive peeling corresponds to $M = 0$.
   - Using only single X-stabilizer generators corresponds to $M = 1$ (the rows of H_X).
   - It is not easy to search for arbitrary products of stabilizer generators with large values of M.
2. **Numerically, we see almost no performance increase for large *M*.**
   - The gap is negligible between $M = 1$ and $M = 2$.
   - We only consider up to $M = 2$ in simulations.



Example: Subgraph of Pruned Peeling Decoder Stopping Sets



Subgraph for a stopping set of the 3 × 3 toric code shown on the standard lattice.

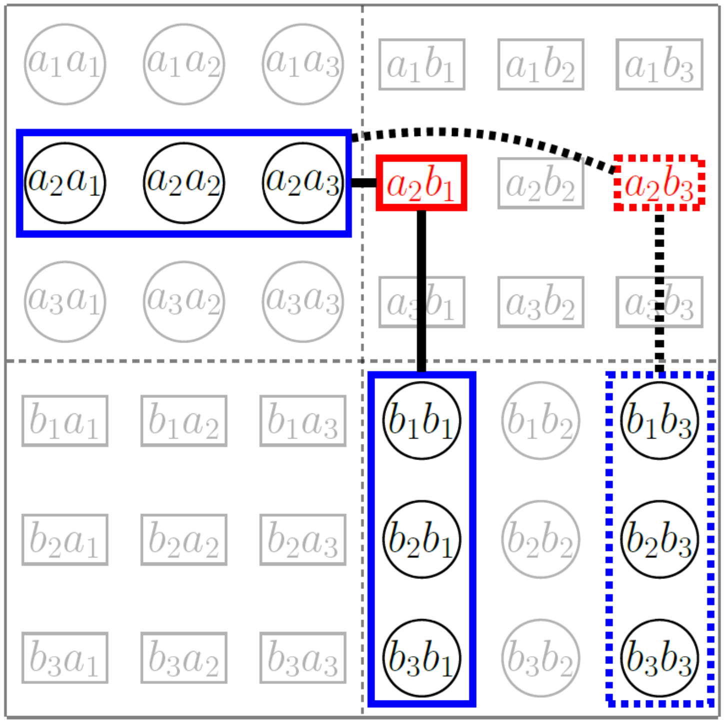


Example of the subgraph induced by a stopping set for a 1600-qubit HGP code.

Thank you!

Peeling Dangling Clusters in Sequence

1. An edge between two clusters in the VH graph defines a <u>connecting check</u> in $T(H_Z)$.
2. There are two possibilities for dangling clusters.
   - A connecting check is <u>free</u> if it is the (weight 1) syndrome of a vector in this dangling cluster.
   - Otherwise, the connecting check is <u>frozen</u>.
3. <u>Frozen dangling clusters</u> can be solved like isolated clusters and removed from the graph.
   - Solutions have the same contribution to this check.
4. <u>Free dangling clusters</u> can be removed from the VH graph and solved after the other clusters.
   - A cluster solution exists independent of this check.



Relative Size and Quantity of Classical Stopping Sets

$$H_1 = [r_1 \times \boxed{n_1}] \qquad\qquad H_X = [r_1 n_2 \times \boxed{(n_1 n_2 + r_1 r_2)}]$$
$$H_2 = [r_2 \times \boxed{n_2}] \quad \Rightarrow \quad H_Z = [r_2 n_1 \times \boxed{(n_1 n_2 + r_1 r_2)}]$$

Classical code lengths n_1 and n_2

HGP code length N

1. The sizes of H_1 and H_2 determine the length N of the HGP code.
2. Assuming that $n_1 \approx n_2 \approx r_1 \approx r_2$, the classical codes $C_1 = \ker(H_1)$ and $C_2 = \ker(H_2)$ have length $n_1 = O(\sqrt{N}) = n_2$ when compared with the length of the HGP code.
3. For each classical stopping set of $T(H_2)$ and $T(H_1^T)$, the Tanner graph $T(H_Z)$ contains on the order of $O(\sqrt{N})$ horizontal and vertical stopping sets.

Further Generalizing the Pruned Peeling Decoder

1. OBSERVATION
   - Numerically, the majority of Pruned Peeling Decoder stopping sets are classical.
2. INTUITION
   - The maximum likelihood decoder uses cubic complexity Gaussian elimination, which is too slow; but can it be applied efficiently to smaller classical stopping sets?
3. CONSIDERATIONS
   - If there exist multiple classical stopping sets, how do they interact with each other?
   - Are classical stopping set solutions always consistent with the HGP solution?
   - In combination with peeling, can these stopping sets always be eliminated?

PART 5

Performance of the Pruned Peeling and VH Decoders