Spatially-Coupled QLDPC Codes

Siyi Yang and Robert Calderbank

Duke Quantum Center, Duke University, Durham, NC 27708, USA

Spatially-coupled (SC) codes is a class of convolutional LDPC codes that has been well investigated in classical coding theory thanks to their high performance and compatibility with low-latency decoders. We describe toric codes as quantum counterparts of classical two-dimensional spatially-coupled (2D-SC) codes, and introduce spatially-coupled quantum LDPC (SC-QLDPC) codes as a generalization. We use the convolutional structure to represent the parity check matrix of a 2D-SC code as a polynomial in two indeterminates, and derive an algebraic condition that is both necessary and sufficient for a 2D-SC code to be a stabilizer code. This algebraic framework facilitates the construction of new code families. While not the focus of this paper, we note that small memory facilitates physical connectivity of qubits, and it enables local encoding and low-latency windowed decoding. In this paper, we use the algebraic framework to optimize short cycles in the Tanner graph of 2D-SC hypergraph product (HGP) codes that arise from short cycles in either component code. While prior work focuses on QLDPC codes with rate less than 1/10, we construct 2D-SC HGP codes with small memories, higher rates (about 1/3), and superior thresholds.

1 Introduction

Quantum computers promise to be more capable of solving certain problems than any classical computer, and the theory of fault-tolerant quantum computation describes how a quantum computer with an arbitrarily low error rate can be built from faulty parts. Quantum error correction is an essential building block, and the class of stabilizer codes has been studied extensively [1–3]. A stabilizer code is the fixed subspace of a commutative subgroup of the Pauli group, and a CSS (Calderbank-Shor-Steane) code is a particular type of stabilizer code where the generators can be separated into strictly X-type and strictly Z-type operators [1, 2].

Geometric locality of stabilizers is important in superconducting qubit platforms, and this design principle has encouraged the development of topological codes. Kitaev [4, 5] introduced toric codes, a class of topological codes defined on a two-dimensional spin lattice. Bravyi et al. [6] modified these codes to allow physical layout of qubits in the Euclidean plane. Surface codes [7–11] are a broad class of topological codes defined by tilings over a closed surface, where holes in the surface correspond to logical operators, and the genus determines code rate. Delfosse and Breuckmann [12–15] introduced hyperbolic and semi-hyperbolic surface codes that use tilings of a closed hyperbolic surface to improve the overhead and logical error rate of toric codes.

Surface codes have very low rates, and quantum LDPC (QLDPC) codes enable higher rates while preserving locality of stabilizers. Tillich and Zemor [16] introduced hypergraph product codes (the hypergraph of two classical LDPC codes), after recognizing the Tanner graph of a surface code as the graph product of two repetition codes. The minimum distance of a hypergraph product code grows with the square root of the block length. Leverrier [17] has developed XYZ codes as a three-dimensional extension of hypergraph product codes. Hypergraph product codes can be viewed as a special case of codes subsequently constructed from more general "products" of component codes.

There has been a great deal of research on constructing codes that achieve a target scaling of minimum distance D and number of encoded bits K with respect to the block length

Siyi Yang: siyi.yang@duke.edu
Robert Calderbank: robert.calderbank@duke.edu

N. Constant rate QLDPC codes with $D \in O(N^{1/2}\log N)$ had been best possible for more than a decade [8, 18], until CSS codes constructed from different homological products of classical codes are brought to our attention. The classical codes are either good random LDPC codes, or expander codes [19] constructed from Cayley graphs of projective linear group or hyperbolic tessellations over closed surfaces [15, 20]. Fibre bundle codes proposed by Hastings, Haah, and O'Donnell [21] are the first codes that improve the scaling exponents to into $K \in O\left(N^{3/5}\right)$ and $D \in O\left(N^{3/5}/\text{polylog}(N)\right)$. Panteleev and Kalachev [22] demonstrated that a scaling of $K \in O(N^{\alpha}\log N)$, $D \in O(N^{1-\alpha/2}/\log N)$ is achievable for arbitrary $0 \leq \alpha < 1$ through the quasi-abelian lifted product codes. Breuckmann and Eberhardt proposed the balanced product codes and improved the scaling exponents to $K \in O(N^{\frac{4}{5}})$ and $D \in O(N^{\frac{3}{5}})$ [23].

Whether constant rates and linear minimum distances could be simultaneously achieved through tensor products of two expander codes defined over non-commutative group rings have remained an open problem [22, 23]. Panteleev and Kalachev completely proved the existence of constant rate codes with linear distance scaling in [24], referred to as expander lifted product codes, obtained from tensor product of two Tanner codes with different local codes defined over the same graph. The local codes are chosen such that their classical product attains good local high-dimensional expansion property, motivated by the idea of high-dimensional expanders proposed in [18]. Concurrently, Leverrier and Zemor [25] introduced a simplification of the lifted product codes from [24] called quantum Tanner codes using a left-right Cayley complex, which can be viewed as the balanced product of Cayley graphs or their double covers.

The recent blossoming of research on QLDPC codes revolves around constructions that guarantee minimum distance. However, classical LDPC codes are not designed to maximize minimum distance, and it does not matter if there is a modest number of low-weight codewords, as long as the noise is statistically unlikely to take the signal in a bad direction. Classical LDPC codes have come to dominate coding practice by focusing on typical errors in graph-based codes rather than on the worst-case errors (minimum distance) in algebraic codes. LDPC codes have succeeded by combining local encoding with iterative decoding. Gallager [26] first suggested using the adjacency matrix of a randomly chosen low-degree bipartite graph as a parity-check matrix. He also introduced belief propagation (BP) decoders, where the task of globally estimating the joint probability of errors (maximum likelihood decoding) is decomposed into parallel local approximations of marginal distributions at each bit (see [27, 28] for a comprehensive analysis of BP decoding).

While BP decoders have been introduced for QLDPC codes [29, 30], subgraphs of the Tanner graph that dominate decoding errors (objects) have not received much attention. Finite length optimization of QLDPC codes have also been overlooked, compared with the rich literature of removing detrimental objects in classical LDPC codes [31–38]. Motivated by the resemblance between toric codes and two dimensional spatially-coupled (SC) codes, we develop SC-QLDPC codes as the quantum analogue of the SC codes in classical world. The SC-QLDPC codes we consider here differ from the non-tail-biting codes introduced by Hagiwara [39]. We consider tail-biting codes to make it easier to increase minimum distance with short component codes. We consider time-invariant codes, since they admit a compact algebraic representation that facilitates systematic optimization. The structure of SC codes is consistent with the locality desired by quantum hardware and it enables systematic optimization with respect to the influence of short cycles. We now highlight our main contributions:

Spatially-coupled QLDPC Codes: We describe toric codes as quantum counterparts of classical two-dimensional spatially-coupled (2D-SC) codes, and introduce spatially-coupled QLDPC codes as a generalization. The parity check matrix of a SC-QLDPC code is assembled into component matrices, which are vertically concatenated into replicas, and these replicas are coupled in a convolutional framework (see [40]) The memory of the code is determined by the number of component matrices in a replica, and small memory simplifies physical connectivity of qubits, enables local encoding, and low-latency windowed decoding (see [41–49], for more information about windowed decoding in the classical world).

Characteristic Functions: We use the language

of two-dimensional convolution to represent the parity check matrix of a 2D-SC code as polynomial $\mathbf{F}(U, V)$ in two indeterminates U, V, where the coefficients are matrices of Pauli matrices. We derive a simple algebraic condition that is necessary and sufficient for a 2D-SC code to be a stabilizer code, and our algebraic framework facilitates the construction of new code families. Characteristic functions are similar in spirit to the Laurent polynomial representation of translation-invariant codes and to the generator polynomials of quantum convolutional codes [50–52]. We observe that the quasi-abelian lifted product codes proposed in [22] can be viewed as high-dimensional spatially-coupled hypergraph product (HGP) codes. Our algebraic framework provides insight into short cycles in the Tanner graph that arise from short cycles in either component codes.

Optimization Framework: Short cycles in Tanner graphs create problems for belief propagation (BP) decoders in both the waterfall and error floor regions (see [31–38] for cycle optimization in the classical world). The fact that quasi-abelian lifted product codes can be viewed as finite-dimensional SC-HGP codes implies that they can be optimized through methods borrowed from classical world with respect to the number of short cycles, and high performance lifted product codes can be constructed even when the lifting exponents are restricted to a small set (corresponding to low memories). As an example, we provide an optimization framework for 2D-SC-HGP codes to minimize the number of short cycles in the Tanner graph that arise from short cycles in either component code, in the same way that generator functions facilitate analysis of the weight distributions of classical convolutional codes. There are short cycles in the Tanner graph that do not involve short cycles in the component codes, and these rigid cycles lead to a fundamental limit on performance (the rigid cycle limit). Simulation results on the quantum depolarizing channel demonstrate that our optimization method provides codes with small memory that approach the rigid cycle limit.

High Rate Quantum LDPC Codes: Research attention has focused on constructing codes that achieve linear scaling of minimum distance and block length and on performance evaluation of low rate codes, such as surface codes. SC-QLDPC codes combine higher rates ($1/3$ rather than $< 1/10$) with high performance.

The rest of the paper is organized as follows. In Section 2, we first introduce classical SC codes, then, after reviewing stabilizer codes, we describe toric codes as 2D-SC codes. In Section 4, we generalize to SC-QLDPC codes, and provide a necessary and sufficient condition for stabilizers to commute. We present two families of SC-QLDPC codes constructed from hypergraph product codes and XYZ codes, and connected the first class with the quasi-cyclic LP codes in [22]. In Section 5, we describe how to enumerate short cycles in the 2D-SC codes constructed from hypergraph product codes, first analyzing cycles of length 4, 6 and 8, then developing an algorithm to minimize the number of these cycles. In Section 6, we compare the performance of our optimized SC-QLDPC codes with that of non-optimized SC-QLDPC codes with the BP decoder described in [29]. In Section 7, we conclude and discuss future research directions.

2 Preliminaries

In this section, we briefly introduce the construction of classical SC-LDPC codes and stabilizer codes. In the remainder of this paper, let $\mathbb{N}$, $\mathbb{N}^*$ represent the nonnegative integers and positive integers, respectively. Let $\mathbb{F}_2$ be the Galois field of order 2. Denote the residue modulo d by $(\cdot)_d$. Denote the all zero matrix and all one matrix of size $m \times n$ by $\mathbf{0}_{m\times n}$, $\mathbf{1}_{m\times n}$, respectively. Let $(\mathbf{A})_{i,j}$ represent the (i, j)-th entry of matrix $\mathbf{A}$. The Kronecker product of matrices is denoted by $\otimes$.

2.1 Classical SC-LDPC Codes

Given a binary matrix $\mathbf{H}^{\mathrm{P}}$, the parity check matrix of a **quasi-cyclic** (QC-)LDPC code is constructed by replacing each element $\left(\mathbf{H}^{\mathrm{P}}\right)_{i,j}$ in $\mathbf{H}^{\mathrm{P}}$ with a $z \times z$ block, where each block is either the zero matrix (if and only if $\left(\mathbf{H}^{\mathrm{P}}\right)_{i,j}=0$) or some power of the circulant matrix defined in (1). The Tanner graph associated with $\mathbf{H}^{\mathrm{P}}$ is referred to as the **protograph** of the QC code, and the QC

code is said to be **lifted from** the protograph.

$$\sigma_z = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{1}$$

An QC-SC code is determined by three $\gamma \times \kappa$ matrices $\mathbf{\Pi}$, $\mathbf{P}$ and $\mathbf{L}$. The **partitioning matrix** $\mathbf{P}$ specifies how the **base matrix** $\mathbf{B}$ is split into $m+1$ component matrices $\mathbf{H}_i^{\mathrm{P}}$ of the protograph. The **lifting matrix** $\mathbf{L}$ specifies how $\mathbf{H}_i^{\mathrm{P}}$ are lifted into component matrices $\mathbf{H}_i$ in the QC-SC code, where $i = 0, \ldots, m$. The entries of $\mathbf{P}$ range from 0 through m and the entries of $\mathbf{L}$ range from 0 through $z-1$. For $1 \leq s \leq \gamma$, $1 \leq t \leq \kappa$, the (s,t)-th element of $\mathbf{H}_i^{\mathrm{P}}$ and the (s,t)-th block of $\mathbf{H}_i$ are specified as follows:

$$\begin{aligned} \left(\mathbf{H}_i^{\mathrm{P}}\right)_{s,t} &= \begin{cases} (\mathbf{H})_{s,t}, & \text{if } (\mathbf{P})_{s,t} = i, \\ 0, & \text{otherwise.} \end{cases}, \text{ and} \\ \mathbf{H}_{i;s,t} &= \begin{cases} \sigma_z^{(\mathbf{L})_{s,t}}, & \text{if } \left(\mathbf{H}_i^{\mathrm{P}}\right)_{s,t} = 1, \\ \mathbf{0}_{z\times z}, & \text{otherwise.} \end{cases} \end{aligned} \tag{2}$$

The component matrices are vertically concatenated into a **replica**, and these replicas are coupled horizontally. SC codes are categorized as **tail-biting** (TB) or **non-tail-biting** (NTB) depending on the form taken by the parity check matrix, as shown in (3). The **memory** of the SC code is m and the number of replicas is referred to as the **coupling length** and is denoted by L.

$$\mathbf{H}_{\mathrm{TB}} = \begin{bmatrix} \mathbf{H}_0 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{H}_m & \cdots & \mathbf{H}_1 \\ \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \cdots & \mathbf{0} & \ddots & \vdots \\ \vdots & \mathbf{H}_1 & \mathbf{H}_0 & \ddots & \ddots & \ddots & \mathbf{H}_m \\ \mathbf{H}_m & \vdots & \ddots & \ddots & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_m & \cdots & \mathbf{H}_1 & \mathbf{H}_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{H}_m & \cdots & \mathbf{H}_1 & \mathbf{H}_0 \end{bmatrix}, \quad \mathbf{H}_{\mathrm{NTB}} = \begin{bmatrix} \mathbf{H}_0 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{H}_1 & \mathbf{H}_0 & \ddots & \vdots \\ \vdots & \mathbf{H}_1 & \ddots & \mathbf{0} \\ \mathbf{H}_m & \vdots & \ddots & \mathbf{H}_0 \\ \mathbf{0} & \mathbf{H}_m & \ddots & \mathbf{H}_1 \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{H}_m \end{bmatrix}. \tag{3}$$

Example 1. *Let $z = 3$, $\gamma = 2$, $\kappa = 3$, $m = 1$, and $L = 4$. The 2×3 base matrix $\mathbf{B}$, partitioning matrix $\mathbf{P}$, and lifting matrix $\mathbf{L}$ are given by:*

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \ \mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \ \text{and } \mathbf{L} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 2 & 1 \end{bmatrix}. \tag{4}$$

The protograph $\mathbf{H}^{\mathrm{P}}$ specified by $\mathbf{B}$ and $\mathbf{P}$ is given by:

$$\mathbf{H}^{\mathrm{P}} = \begin{bmatrix} \mathbf{H}_0^{\mathrm{P}} & \mathbf{0} & \mathbf{0} & \mathbf{H}_1^{\mathrm{P}} \\ \mathbf{H}_1^{\mathrm{P}} & \mathbf{H}_0^{\mathrm{P}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_1^{\mathrm{P}} & \mathbf{H}_0^{\mathrm{P}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_1^{\mathrm{P}} & \mathbf{H}_0^{\mathrm{P}} \end{bmatrix}, \tag{5}$$

where

$$\mathbf{H}_0^{\mathrm{P}} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \ \mathbf{H}_1^{\mathrm{P}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{6}$$

The parity check matrix $\mathbf{H}$ lifted from $\mathbf{H}^{\mathrm{P}}$ with respect to $\mathbf{L}$ is given by:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0 & \mathbf{0} & \mathbf{0} & \mathbf{H}_1 \\ \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_1 & \mathbf{H}_0 \end{bmatrix}, \tag{7}$$

where

$$\begin{aligned} \mathbf{H}_0 &= \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right], \\ \mathbf{H}_1 &= \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]. \end{aligned} \tag{8}$$

2.2 Two-Dimensional SC Codes

A memory m_1 SC code is said to be a **two-dimensional (2D) SC code** if each component matrix $\mathbf{H}_i$, $i = 0, \ldots, m_1$, is itself a memory m_2 SC code with component matrices $\mathbf{H}_{i,j}$, $j = 0, \ldots, m_2$.

Example 2. *Let* $m_1 = 1$ *and* $m_2 = 1$. *Let* $\mathbf{H}_{0,0} = \mathbf{A}$, $\mathbf{H}_{0,1} = \mathbf{B}$, $\mathbf{H}_{1,0} = \mathbf{C}$, *and* $\mathbf{H}_{1,1} = \mathbf{D}$. *The 2D-SC code is given by:*

$$\mathbf{H}_{2D\text{-}SC} = \left[\begin{array}{cccc|cccc|c|cccc} \mathbf{A} & \mathbf{0} & \cdots & \mathbf{B} & & & & & & & \mathbf{C} & \mathbf{0} & \cdots & \mathbf{D} \\ \mathbf{B} & \mathbf{A} & \cdots & \mathbf{0} & & & & & & & \mathbf{D} & \mathbf{C} & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots & & & \mathbf{0} & & & \cdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{B} & \mathbf{A} & & & & & & & \mathbf{0} & \cdots & \mathbf{D} & \mathbf{C} \\ \hline \mathbf{C} & \mathbf{0} & \cdots & \mathbf{D} & \mathbf{A} & \mathbf{0} & \cdots & \mathbf{B} & & & & & & \\ \mathbf{D} & \mathbf{C} & \cdots & \mathbf{0} & \mathbf{B} & \mathbf{A} & \cdots & \mathbf{0} & & \cdots & & & & \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots & & & & \mathbf{0} & & \\ \mathbf{0} & \cdots & \mathbf{D} & \mathbf{C} & \mathbf{0} & \cdots & \mathbf{B} & \mathbf{A} & & & & & & \\ \hline & \vdots & & & & \ddots & & & & \ddots & & \vdots & & \\ \hline & & \mathbf{0} & & & \cdots & & & \mathbf{C} & \mathbf{0} & \cdots & \mathbf{D} & \mathbf{A} & \mathbf{0} & \cdots & \mathbf{B} \\ & & & & & & & & \mathbf{D} & \mathbf{C} & \cdots & \mathbf{0} & \mathbf{B} & \mathbf{A} & \cdots & \mathbf{0} \\ & & & & & & & & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ & & & & & & & & \mathbf{0} & \cdots & \mathbf{D} & \mathbf{C} & \mathbf{0} & \cdots & \mathbf{B} & \mathbf{A} \end{array}\right]. \quad (9)$$

2.3 The Pauli Group

An 2×2 Hermitian matrix can be uniquely expressed as a real linear combination of the four single qubit **Pauli matrices**:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \; X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \\ Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \; Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \quad (10)$$

The matrices satisfy $X^2 = Y^2 = Z^2 = \mathbf{I}_2$, $XY = -YX$, $ZX = -XZ$, and $YZ = -ZY$.

Let $\mathbf{A} \otimes \mathbf{B}$ denote the Kronecker (tensor) product of two matrices $\mathbf{A}$ and $\mathbf{B}$. Let $\mathbf{a} = [a_1, \ldots, a_N]$ and $\mathbf{b} = [b_1, \ldots, b_N]$ be binary vectors. Define

$$D(\mathbf{a}, \mathbf{b}) = X^{a_1} Z^{b_1} \otimes \cdots \otimes X^{a_N} Z^{b_N}, \\ E(\mathbf{a}, \mathbf{b}) = i^{\mathbf{a}\mathbf{b}^{\mathrm{T}}} D(\mathbf{a}, \mathbf{b}). \quad (11)$$

Then,

$$E(\mathbf{a}, \mathbf{b})^2 = i^{2\mathbf{a}\mathbf{b}^{\mathrm{T}}} D(\mathbf{a}, \mathbf{b})^2 \\ = i^{2\mathbf{a}\mathbf{b}^{\mathrm{T}}} \left(i^{2\mathbf{a}\mathbf{b}^{\mathrm{T}}} \mathbf{I}_{2^N} \right) = \mathbf{I}_{2^N}. \quad (12)$$

The N**-qubit Pauli Group** $\mathcal{P}_N$ is given by:

$$\mathcal{P}_N = \{ i^k D(\mathbf{a}, \mathbf{b}) | \mathbf{a}, \mathbf{b} \text{ are binary vectors}, \\ k = 0, 1, 2, 3 \}. \quad (13)$$

We use the Dirac notation $|\cdot\rangle$ to represent the basis states of a single qubit in $\mathbb{C}^2$. The Pauli matrices act on a single qubit as $X|0\rangle = |1\rangle$, $X|1\rangle = |0\rangle$, $Z|0\rangle = |0\rangle$, and $Z|1\rangle = -|1\rangle$. For any $\mathbf{v} = [v_1, \ldots, v_N]$ in $\mathbb{F}_2^N$, we define:

$$|\mathbf{v}\rangle = |v_1\rangle \otimes |v_2\rangle \cdots \otimes |v_N\rangle, \quad (14)$$

the standard basis vector with 1 the position indexed by $\mathbf{v}$ and zeros elsewhere. We may write an arbitrary N qubit quantum state as

$$|\phi\rangle = \sum_{\mathbf{v} \in \mathbb{F}_2^N} \alpha_{\mathbf{v}} |\mathbf{v}\rangle, \quad (15)$$

where $\alpha_{\mathbf{v}} \in \mathbb{C}$ and $\sum_{\mathbf{v} \in \mathbb{F}_2^N} |\alpha_{\mathbf{v}}|^2 = 1$.

The **symplectic inner product** is $\langle (\mathbf{a}, \mathbf{b}), (\mathbf{c}, \mathbf{d}) \rangle_s = \mathbf{a}\mathbf{b}^{\mathrm{T}} + \mathbf{c}\mathbf{d}^{\mathrm{T}}$. Since $XZ = -ZX$ we have:

$$E(\mathbf{a}, \mathbf{b}) E(\mathbf{c}, \mathbf{d}) = (-1)^{\langle (\mathbf{a}, \mathbf{b}), (\mathbf{c}, \mathbf{d}) \rangle_s} E(\mathbf{c}, \mathbf{d}) E(\mathbf{a}, \mathbf{b}). \quad (16)$$

2.4 Stabilizer Codes

We define a **stabilizer group** S to be a commutative subgroup of the Pauli group $\mathcal{P}_N$, where every group element is Hermitian and no group element is $-I_{2^N}$. We say S has a dimension r if

it can be generated by r independent elements as $\mathcal{S} = \langle \nu_i E(\mathbf{c}_i, \mathbf{d}_i) | i = 1, 2, \ldots, r \rangle$, where $\nu_i = \pm 1$ and $c_i, d_i \in \mathbb{F}_2^N$. Since $\mathcal{S}$ is commutative we must have:

$$\langle (\mathbf{c}_i, \mathbf{d}_i), (\mathbf{c}_j, \mathbf{d}_j) \rangle_s = \mathbf{c}_i \mathbf{d}_j^{\mathrm{T}} + \mathbf{d}_i \mathbf{c}_j^{\mathrm{T}} = 0. \quad (17)$$

Given a stabilizer group $\mathcal{S}$, the corresponding **stabilizer code** is the fixed subspace $\mathcal{V}(\mathcal{S}) = \{|\phi\rangle \in \mathbb{C}^{2N} | g|\phi\rangle = |\phi\rangle \text{ for all } g \in \mathcal{S}\}$. We refer to the subspace $\mathcal{V}(\mathcal{S})$ as an $[[N, K]]$ stabilizer code, because it encodes $K = N - r$ logical qubits into N physical qubits.

3 Toric Codes as SC codes

A **Toric code** is a CSS code defined on a $d \times d$ grid. In Fig. 1, edges represent qubits, edges incident to a face represent X-type generators, and edges incident to a vertex represent Z-type generators. The geometry of the grid implies that any pair of stabilizer generator commutes.

We organize the rows of the parity check matrix $\mathbf{H}$ for the 3×3 toric code so that the structure of $\mathbf{H}$ resembles that of a 2D-SC code, as shown in (20). Given the labels in Fig. 1, column i represents the qubit associated with edge i. Row $2i - 1$ represents the stabilizer generator associated with face i, and row $2i$ represents the generator associated with vertex i. For example, row 15 represents $X_{10} \otimes X_{13} \otimes X_{14} \otimes X_{15}$ and row 10 represents $Z_2 \otimes Z_3 \otimes Z_4 \otimes Z_7$, and face 8.

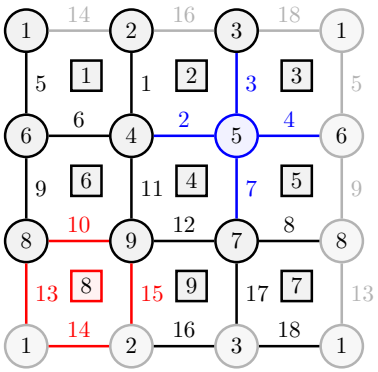


Figure 1: A toric code defines on a 3×3 grid. Vertices are indicated by circles and faces are indicated by squares. Vertices/edges with the same index are glued together to create a torus with 18 edges, 9 vertices, and 9 faces. Vertex 5 specifies the generator $Z_2 \otimes Z_3 \otimes Z_4 \otimes Z_7$ and face 8 specifies generator $X_{10} \otimes X_{13} \otimes X_{14} \otimes X_{15}$.

We have

$$\mathbf{H} = \left[\begin{array}{ccc|ccc|ccc} A & & B & & & & C & & D \\ B & A & & & & & D & C & \\ & B & A & & & & & D & C \\ \hline C & & D & A & & B & & & \\ D & C & & B & A & & & & \\ & D & C & & B & A & & & \\ \hline & & & C & & D & A & & B \\ & & & D & C & & B & A & \\ & & & & D & C & & B & A \end{array}\right], \quad (18)$$

where

$$\mathbf{A} = \begin{bmatrix} X & I \\ I & I \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} X & X \\ Z & I \end{bmatrix}, \ \mathbf{C} = \begin{bmatrix} I & X \\ Z & Z \end{bmatrix}, \ \mathbf{D} = \begin{bmatrix} I & I \\ I & Z \end{bmatrix}. \quad (19)$$

$$\left[\begin{array}{cc|cc|cc|cc|cc|cc} X & I & & X & X & & & & & I & X & & I & I \\ I & I & & Z & I & & & & & I & I & & I & Z \\ \hline X & X & X & I & & & & & & I & I & I & X & \\ Z & I & I & I & & & & & & I & Z & Z & Z & \\ \hline & & X & X & X & I & & & & & & I & I & I & X \\ & & Z & I & I & I & & & & & & I & Z & Z & Z \\ \hline I & X & & I & I & X & & X & X & & & & & & \\ Z & Z & & I & I & Z & & Z & I & & & & & & \\ \hline I & I & I & X & X & & X & X & I & & & & & & \\ I & Z & Z & Z & & & Z & I & I & & & & & & \\ \hline & & I & I & I & Z & & I & I & & & & & & \\ & & I & Z & Z & Z & & I & Z & & & & & & \\ \hline & & & & & & I & X & & I & X & & X & X & \\ & & & & & & Z & Z & & I & I & & Z & I & \\ \hline & & & & & & I & I & X & X & I & X & & & \\ & & & & & & I & Z & Z & I & I & Z & & & \\ \hline & & & & & & & & I & I & X & & I & I & \\ & & & & & & & & I & Z & Z & & Z & Z & \\ \hline & & & & & & & & & & I & I & X & X & I \\ & & & & & & & & & & I & Z & Z & I & I \end{array}\right]. \quad (20)$$

Remark 1. (Extension to $d \times d$ grids) *The face/vertex labels at level i are obtained by adding d to the face/vertex labels at level $i-1$, then cycling to the right. Thus, a face and the vertex at the top left of the face receives the same label l (just as in Fig. 1). To be precise, the (i,j)-th face, $i,j = 1,\ldots,d$, is labelled by $((i-1)d+(j-i)_d+1)$, where $(\cdot)_d$ denotes the residue modulo d.*

The edges at the left and at the bottom of the (i,j)-th face are then labelled consecutively as $2(i-1)d+(j-i-1)_d)+1$ and $2(i-1)d+(j-i-1)_d)+2$. The parity check matrix $\mathbf{H}$ of the $d \times d$ toric code takes the form in (9), where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are given by (19). It has the structure of a 2D-SC code with $(m_1, m_2, L_1, L_2) = (1, 1, d, d)$.

4 Commutative Algebra

We begin by extending the symplectic inner product to matrices for which every entry is either the zero matrix $\mathbf{0}_2$ or a 2×2 Pauli matrix. Given an $m_1 \times n$ matrix $\mathbf{P} = [P_{i,k}]$ and an $m_2 \times n$ matrix $\mathbf{Q} = [Q_{j,k}]$ define $\langle P, Q\rangle_s$ to be the $m_1 \times m_2$ binary matrix given by:

$$\begin{aligned}(\langle \mathbf{P}, \mathbf{Q}\rangle_s)_{i,j} &= \left\langle [\bigotimes_{k:P_{i,k}\neq \mathbf{0}_2} P_{i,k}], [\bigotimes_{k:Q_{i,k}\neq \mathbf{0}_2} Q_{j,k}] \right\rangle_s \\ &= \sum_{k:P_{i,k}\neq \mathbf{0}_2 \text{ and } Q_{i,k}\neq \mathbf{0}_2} \langle P_{i,k}, Q_{j,k}\rangle_s.\end{aligned} \tag{21}$$

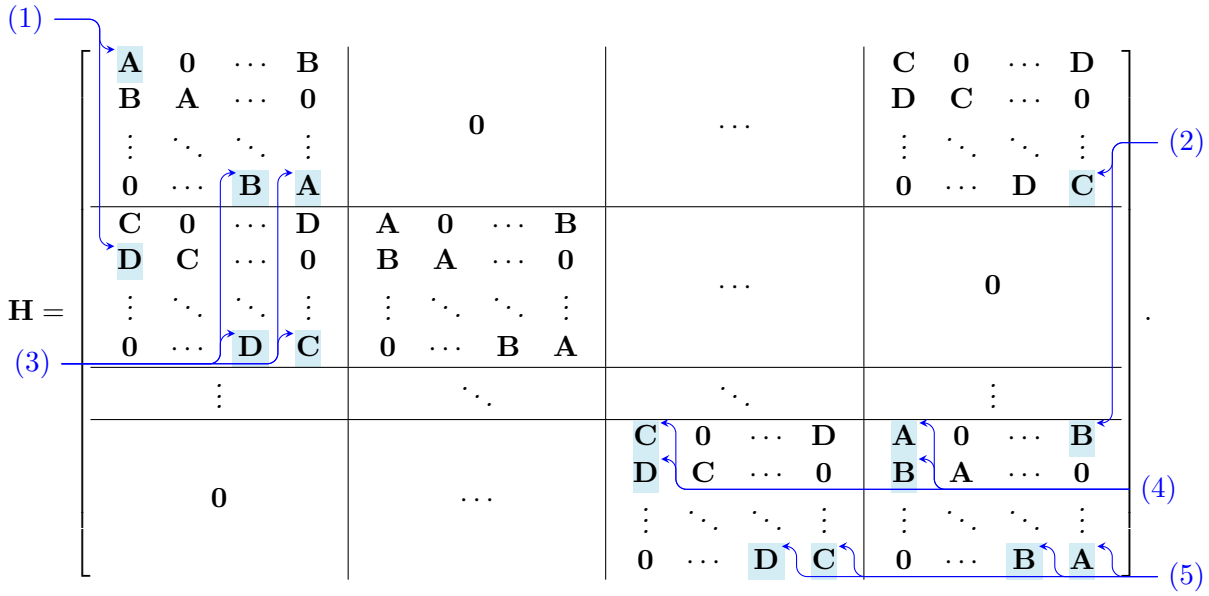
Example 3. *Let $m_1 = 1$, $m_2 = 2$, and $n = 4$. Specify $\mathbf{P} \in \mathcal{P}^{1\times 4}$, $\mathbf{Q} \in \mathcal{P}^{2\times 4}$ as follows:*

$$\begin{aligned}\mathbf{P} &= \begin{bmatrix} X & Y & Z & I \end{bmatrix}, \\ \mathbf{Q} &= \begin{bmatrix} I & X & Z & Y \\ Z & Y & X & X \end{bmatrix}.\end{aligned} \tag{22}$$

Then,

$$\begin{aligned}\langle \mathbf{P}, \mathbf{Q}\rangle_s &= \begin{bmatrix} \langle X,I\rangle_s + \langle Y,X\rangle_s + \langle Z,Z\rangle_s + \langle I,Y\rangle_s & \langle X,Z\rangle_s + \langle Y,Y\rangle_s + \langle Z,X\rangle_s + \langle I,X\rangle_s \end{bmatrix} \\ &= \begin{bmatrix} 0+1+0+0 & 1+0+1+0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix}.\end{aligned} \tag{23}$$

Example 4. ***(Toric codes as 2D-SC codes)*** *We verify that the matrix $\mathbf{H}$ given below (recall (9)) is the parity check matrix of a stabilizer code by checking five symplectic inner products. Every row of $\mathbf{H}$ determines a Pauli matrix, and the requirement that generators commute translates to the requirement that five symplectic inner product vanish. The matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are defined in (19).*



We require

(1) $\langle \mathbf{A}, \mathbf{D} \rangle_s = \mathbf{0}_{2\times 2}$;

(2) $\langle \mathbf{B}, \mathbf{C} \rangle_s = \mathbf{0}_{2\times 2}$;

(3) $\langle \mathbf{A}, \mathbf{C} \rangle_s + \langle \mathbf{B}, \mathbf{D} \rangle_s = \mathbf{0}_{2\times 2}$;

(4) $\langle \mathbf{A}, \mathbf{B} \rangle_s + \langle \mathbf{C}, \mathbf{D} \rangle_s = \mathbf{0}_{2\times 2}$;

(5) $\langle \mathbf{A}, \mathbf{A} \rangle_s + \langle \mathbf{B}, \mathbf{B} \rangle_s + \langle \mathbf{C}, \mathbf{C} \rangle_s + \langle \mathbf{D}, \mathbf{D} \rangle_s = \mathbf{0}_{2\times 2}$.

Remark 2. *More generally, if* $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ *are* $r \times n$ *matrices satisfying the above conditions (with* $\mathbf{0}_{2\times 2}$ *replaced by* $\mathbf{0}_{r\times r}$*), then the 2D-SC code with component matrices* $(\mathbf{H}_{0,0}, \mathbf{H}_{0,1}, \mathbf{H}_{1,0}, \mathbf{H}_{1,1}) = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ *is a stabilizer code.*

Theorem 1. *Let* $\mathcal{P}$ *be the Pauli group on a single qubit, and let* $\mathbf{H}_{i,j} \in \mathcal{P}^{r\times n}$, $i = 0, \ldots, m_1$, $j = 0, \ldots, m_2$, *be the component matrices of a 2D-SC code with parameters* (m_1, m_2, L_1, L_2). *If the following conditions are satisfied:*

$$\sum_{i=0}^{m_1-d_1} \sum_{j=0}^{m_2-d_2} \langle \mathbf{H}_{i,j}, \mathbf{H}_{i+d_1,j+d_2} \rangle = \mathbf{0}_{r\times r}, \quad \forall 0 \le d_1 \le m_1, 0 \le d_2 \le m_2, \tag{24}$$

$$\sum_{i=0}^{m_1-d_1} \sum_{j=0}^{m_2-d_2} \langle \mathbf{H}_{i+d_1,j}, \mathbf{H}_{i,j+d_2} \rangle = \mathbf{0}_{r\times r}, \quad \forall 0 \le d_1 \le m_1, 0 \le d_2 \le m_2, \tag{25}$$

then the 2D-SC code is a stabilizer code.

Remark 3. *When* $r = n = 2$, $m_1 = m_2 = 1$, *and* $L_1 = L_2 = d$, *then the conditions (24) and (25) reduce to conditions (1)-(5) in Example 4.*

Proof. Conditions (24) translates to verifying pairwise orthogonality of blocks of rows, as shown in Fig. 2. Condition (25) translates to a similar visualization, and we omit the details. □

Remark 4. *In the classical world, non-tail-biting (NTB) codes are associated with better thresholds than tail-biting (TB) codes, due to the existence of low degree check nodes at the two ends of the coupling chain (see the discussion of low threshold saturation in SC codes [40, 53, 54]). However, due to the commutativity conditions required by stabilizer codes, in the quantum world, NTB code suffer from low minimum distance.*

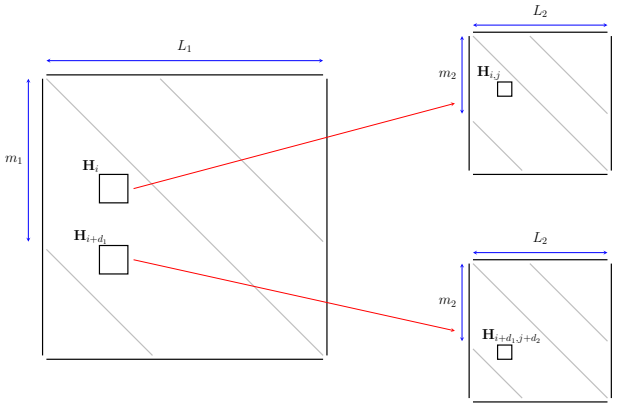


Figure 2: Verifying pairwise orthogonality of blocks of rows in (24).

This is because component matrices must pairwise commute, and the block length of the stabilizer code $\mathcal{C}'$ *generated by the component matrices is small* ($n = N/L$). *Any product of stabilizers in* $\mathcal{C}'$ *gives rise to a non-correctable error pattern in the NTB code.*

4.1 SC Codes on Quantum Hardware

Limitations on qubit connectivity present a significant challenge for implementing QLDPC codes across various quantum hardware platforms. However, the convolutional structure of SC codes allows them to be employed efficiently on diverse quantum architectures.

Superconducting platforms, such as those adopted by companies like IBM and Google, only permit entanglement between neighboring qubits, necessitating extensive swap gates when implementing QLDPC codes with arbitrary connections. IBM has proposed partitioning the Tanner graph into multiple planar subgraphs, where the count of these subgraphs, termed "thickness," influences the implementation efficiency of QLDPC codes [55]. For an SC code with memory parameters (m_1, m_2), the thickness and maximum entanglement distances within each planar subgraph depend on m_1 and m_2. SC codes with short memories thus allow efficient implementations in superconducting architectures.

Neutral atom arrays, such as those developed by QuEra and PASQAL, offer improved QLDPC code implementation through reconfigurable atom arrays (RAAs) [56, 57]. RAAs consist of a fixed atom array using spatial light modulators (SLM) and multiple movable arrays using acoustic-optic deflectors (AOD). The AOD

arrays enable aligned qubit movement along rows and columns, though paths cannot cross within a single AOD array. As qubits in AOD and SLM arrays move in union, any pair within the Rydberg interaction distance can simultaneously form two-qubit gates using a Rydberg laser. By allocating qubits across AOD arrays and arranging their movements, all required two-qubit gates in QLDPC codes can be realized. The cost and fidelity of RAA implementation depend on atom movement overhead, making high parallelism essential. SC codes naturally support this parallelism by distributing ancilla qubits from different replicas across AOD arrays with identical qubit layouts. Given the arrangement of the base block code, the SC code's configuration follows naturally.

For example, Fig. 3 and Fig. 4 show the implementation of edges in the component matrices $\mathbf{H}_{0,0}$ and $\mathbf{H}_{2,1}$, respectively, in an SC code with a base code size of $(9, 4)$, memories $m_1 = m_2 = 2$, and coupling lengths $L_1 = L_2 = 5$. Blue and red dots represent data qubits and ancilla qubits, respectively. The data qubits are arranged by the SLM as a fixed 5×5 grid, with each 3×3 sub-array corresponding to the data qubits in a replica. The ancilla qubits are organized into a large AOD array, which is divided into 3×3 sub-arrays of size 2×2, and 16 smaller AOD arrays, each containing a 2×2 qubit layout, allowing for mobility to facilitate two-qubit gate formation. Qubits within Rydberg distance are enclosed in black ovals.

In Fig.3, qubits within each 2×2 AOD array are positioned at the four corners of their corresponding 3×3 data qubit sub-array, enabling four two-qubit gates in $\mathbf{H}_{0,0}$. In Fig.4, to implement two-qubit gates in $\mathbf{H}_{2,1}$, each AOD array is shifted two grids to the left and one grid up in a rotated layout. Within each 2×2 sub-array, the rows and columns are adjusted so that qubits in the second column align with the top and bottom qubits of the central column in the corresponding 3×3 data qubit sub-array, while qubits in the first column remain separate from all data nodes, thus activating two two-qubit gates in $\mathbf{H}_{2,1}$.

In general, SC codes with small memories support highly parallel implementation on RAAs.

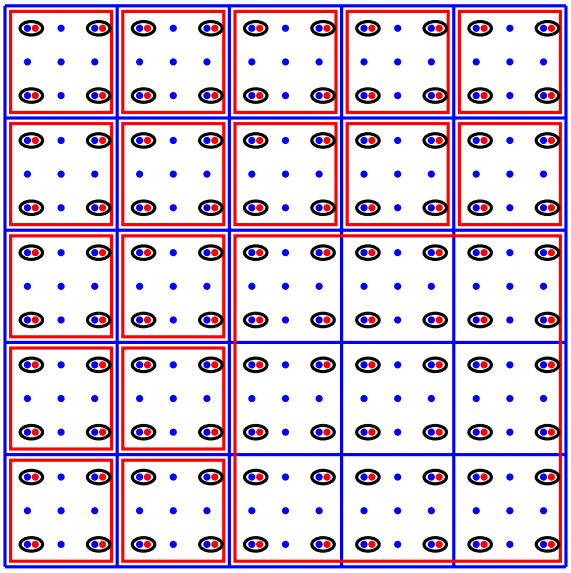


Figure 3: Layout of an SC code with a $(9, 4)$ base code, memories $m_1 = m_2 = 2$, and coupling lengths $L_1 = L_2 = 5$. Blue dots represent data qubits in a fixed spatial light modulator (SLM) array, while red dots denote ancilla qubits in movable acousto-optic deflector (AOD) arrays. Qubit pairs within each black oval are positioned within the Rydberg distance, enabling four two-qubit gates in $\mathbf{H}_{0,0}$ of the corresponding replica via a Rydberg laser.

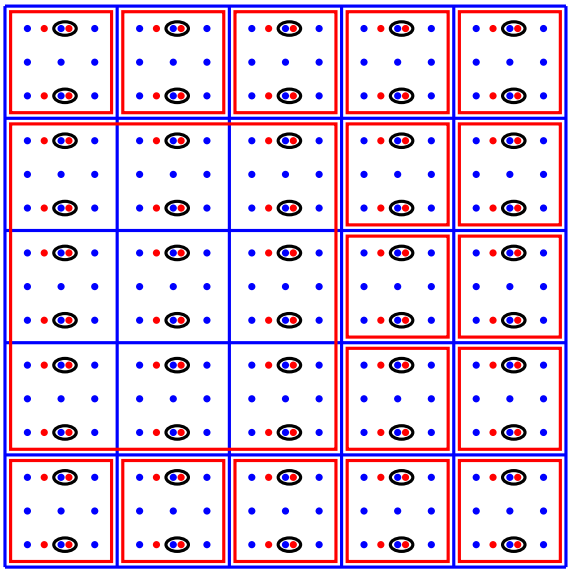


Figure 4: Implementation of gates in $\mathbf{H}_{2,1}$ of the SC code shown in Fig. 3. Each AOD array is shifted two grids left and one grid up in a rotated layout. Qubit pairs enclosed within each black oval are positioned within the Rydberg distance, allowing the formation of four two-qubit gates in $\mathbf{H}_{2,1}$ of the corresponding replica via a Rydberg laser.

4.2 Characteristic Function

In Theorem 1, we showed that a 2D-SC code is a stabilizer code if and only if conditions (24) and (25) are satisfied. We now use the language of two-dimensional convolution to translate these conditions into a simple algebraic form that facilitates the construction and analysis of a large class of codes. We begin with the example of the $d \times d$ toric code.

We represent the $d \times d$ toric code as the polynomial $\mathbf{F}(U, V)$ in two indeterminates U, V, with coefficients in $\mathcal{P}^{2\times 2}$ given by

$$\mathbf{F}(U,V) = \mathbf{A} + \mathbf{B}V + \mathbf{C}U + \mathbf{D}UV, \tag{26}$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are given by (19). Formal multiplication gives

$$\begin{aligned}
&\mathbf{F}(U,V)\mathbf{F}(U^{-1},V^{-1})^{\mathrm{T}} \\
=&(\mathbf{A}+\mathbf{B}V+\mathbf{C}U+\mathbf{D}UV) \\
&(\mathbf{A}+\mathbf{B}V^{-1}+\mathbf{C}U^{-1}+\mathbf{D}U^{-1}V^{-1})^{\mathrm{T}} \\
=&(\mathbf{A}\mathbf{A}^{\mathrm{T}}+\mathbf{B}\mathbf{B}^{\mathrm{T}}+\mathbf{C}\mathbf{C}^{\mathrm{T}}+\mathbf{D}\mathbf{D}^{\mathrm{T}}) \\
&+(\mathbf{A}\mathbf{B}^{\mathrm{T}}+\mathbf{C}\mathbf{D}^{\mathrm{T}})V^{-1}+(\mathbf{B}\mathbf{A}^{\mathrm{T}}+\mathbf{D}\mathbf{C}^{\mathrm{T}})V \\
&+(\mathbf{A}\mathbf{C}^{\mathrm{T}}+\mathbf{B}\mathbf{D}^{\mathrm{T}})U^{-1}+(\mathbf{C}\mathbf{A}^{\mathrm{T}}+\mathbf{D}\mathbf{B}^{\mathrm{T}})U \\
&+\mathbf{A}\mathbf{D}^{\mathrm{T}}U^{-1}V^{-1}+\mathbf{D}\mathbf{A}^{\mathrm{T}}UV \\
&+\mathbf{B}\mathbf{C}^{\mathrm{T}}VU^{-1}+\mathbf{C}\mathbf{B}^{\mathrm{T}}UV^{-1}.
\end{aligned} \tag{27}$$

There is a 1–1 correspondence between terms on the right side of (27) and the conditions satisfied by $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ in Example 4, which implies that the $d \times d$ toric code is a stabilizer code.

We now extend the symplectic inner product to polynomials $\mathbf{F}(U, V)$ in two indeterminates U, V with Pauli matrix coefficients. Let $\mathbf{P}(U,V) = \sum_{i,j\in\mathbb{Z}} \mathbf{P}_{i,j}U^iV^j$ with $\mathbf{P}_{i,j} \in \mathcal{P}^{r_1\times n}$ and let $\mathbf{Q}(U,V) = \sum_{i,j\in\mathbb{Z}} \mathbf{Q}_{i,j}U^iV^j$ with $\mathbf{Q}_{i,j} \in \mathcal{P}^{r_2\times n}$.

Definition 1. *The symplectic inner product $\langle \mathbf{P}, \mathbf{Q}\rangle_s$ is the polynomial in two indeterminates $\mathbf{U}, \mathbf{V}$ with coefficients in $\mathbb{F}_2^{r_1\times r_2}$ given by:*

$$\langle \mathbf{P}, \mathbf{Q}\rangle_s = \sum_{k,l\in\mathbb{Z}} \left(\sum_{i,j\in\mathbb{Z}} \langle \mathbf{P}_{i,j}, \mathbf{Q}_{k-i,l-j}\rangle_s \right) U^kV^l. \tag{28}$$

Remark 5. *The five conditions satisfied by $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$, that imply the toric code is a stabilizer code, reduce to the single condition:*

$$\left\langle \mathbf{F}(U,V), \mathbf{F}(U^{-1},V^{-1}) \right\rangle_s = \mathbf{0}_{2\times 2}. \tag{29}$$

Definition 2. *The **characteristic function** of a 2D-SC code is the polynomial*

$$\mathbf{F}(U,V) = \sum_{i=0}^{m_1} \sum_{j=0}^{m_2} \mathbf{H}_{i,j}U^iV^j. \tag{30}$$

Theorem 2. *A 2D-SC code is a stabilizer code if and only if the characteristic function $\mathbf{F}(U, V)$ satisfies:*

$$\left\langle \mathbf{F}(U,V), \mathbf{F}(U^{-1},V^{-1}) \right\rangle_s = \mathbf{0}_{r\times r}. \tag{31}$$

Proof. We rewrite conditions (24) and (25) of Theorem 1 as

$$\sum_{i=\max\{0,-d_1\}}^{\min\{m_1-d_1,m_1\}} \sum_{j=\max\{0,-d_2\}}^{\min\{m_2-d_2,m_2\}} \langle \mathbf{H}_{i,j}, \mathbf{H}_{i+d_1,j+d_2}\rangle = \mathbf{0}_{r\times r}. \tag{32}$$

Now,

$$\begin{aligned}
&\left\langle \mathbf{F}(U,V), \mathbf{F}(U^{-1},V^{-1}) \right\rangle_s \\
=&\left\langle \sum_{i=0}^{m_1}\sum_{j=0}^{m_2} \mathbf{H}_{i,j}U^iV^j, \sum_{i=0}^{m_1}\sum_{j=0}^{m_2} \mathbf{H}_{i,j}U^{-i}V^{-j} \right\rangle_s \\
=&\sum_{d_1=-m_1}^{m_1}\sum_{d_2=-m_2}^{m_2} \left(\sum_{i=\max\{0,-d_1\}}^{\min\{m_1-d_1,m_1\}} \sum_{j=\max\{0,-d_2\}}^{\min\{m_2-d_2,m_2\}} \langle \mathbf{H}_{i,j}, \mathbf{H}_{i+d_1,j+d_2}\rangle_s \right) U^{-d_1}V^{-d_2} \\
=&\mathbf{0}_{r\times r}
\end{aligned} \tag{33}$$

The result follows from Theorem 1. □

Condition (31) in Theorem 2 facilitates the

construction of 2D-SC codes that are stabilizer codes, and we provide two examples below.

$$\mathbf{F}(U,V)=\begin{bmatrix} X(1+U) & Y(1+V) & Z \\ Y(UV) & Z(V) & X(1+UV) \\ Z(U+UV+U^2V^2) & X(V+UV^2+U^2V^2) & Y(U^2V+U) \end{bmatrix}. \tag{34}$$

$$\mathbf{F}(U,V)=\begin{bmatrix} Y(1+UV) & 0 & Z(U+V) & X(1+U) \\ 0 & Z(1+UV) & X(V+UV) & Y(U+V) \\ X(V+UV) & Y(U+V) & 0 & Z(1+UV) \\ Z(U+V) & X(1+U) & Y(1+UV) & 0 \end{bmatrix}. \tag{35}$$

We now explain how characteristic functions facilitate the analysis of minimum distance of 2D-SC codes with coupling lengths L_1 and L_2. We define the characteristic function of a codeword $\mathbf{c}\in\mathcal{P}^{nL_1L_2}$ by

$$\mathbf{c}\left(U,V\right)=\sum_{i=0}^{L_1-1}\sum_{j=0}^{L_2-1}\mathbf{c}_{i,j}U^iV^j, \tag{36}$$

where each $\mathbf{c}_{i,j}$ is a sub-sequence of $\mathbf{c}$ with length n, and $\mathbf{c}=(\mathbf{c}_0,\mathbf{c}_1,\ldots,\mathbf{c}_{L_1-1})$ where $\mathbf{c}_i=(\mathbf{c}_{i,0},\mathbf{c}_{i,1},\ldots,\mathbf{c}_{i,L_2-1})$, $i=0,1,\ldots,L_1-1$. The weight of the codeword (characteristic function) is the number of non-zero terms in $\mathbf{c}(U,V)$.

We also need to define characteristic vectors of vectors in the stabilizer group. Let $\mathbf{s}\in\mathbb{F}_2^{rL_1L_2}$ be an indicator function of the stabilizer $\mathbf{c}=\prod_{i:s_i=1}\mathbf{g}_i$, where $\mathbf{g}_1,\mathbf{g}_2,\ldots,\mathbf{g}_{rL_1L_2}$ are the generators in the parity check matrix of a SC-QLDPC code. Suppose $\mathbf{s}$ is divided into L_1L_2 sub-sequences of length r by $\mathbf{s}=(\mathbf{s}_0,\mathbf{s}_1,\ldots,\mathbf{s}_{L_1-1})$ where $\mathbf{s}_i=(\mathbf{s}_{i,0},\mathbf{s}_{i,1},\ldots,\mathbf{s}_{i,L_2-1})$, $i=0,1,\ldots,L_1-1$. Then the characteristic function of the stabilizer $\mathbf{c}$ can be represented by

$$\mathbf{c}\left(U,V\right)=\mathbf{s}(U,V)\mathbf{F}(U^{-1},V^{-1}), \tag{37}$$

where $U^{L_1}=V^{L_2}=1$ and

$$\mathbf{s}(U,V)=\sum_{i=0}^{L_1-1}\sum_{j=0}^{L_2-1}\mathbf{s}_{i,j}U^iV^j. \tag{38}$$

Theorem 3. *The minimum distance of a 2D-SC code with characteristic function $\mathbf{F}(U,V)$ is the minimum weight of a characteristic function $\mathbf{c}(U,V)$ that satisfies:*

$$\left\langle \mathbf{F}\left(U,V\right),\mathbf{c}\left(U,V\right)\right\rangle=\mathbf{0}_{r\times1},\ \mathbf{c}\left(U,V\right)\notin\left\{\mathbf{s}(U,V)\mathbf{F}(U^{-1},V^{-1})\middle|\mathbf{s}(U,V)\in\mathbb{F}_2\left[U,V\right]/\langle U^{L_1},V^{L_2}\rangle\right\}. \tag{39}$$

Proof. The requirement that a codeword commutes with every stabilizer generator reduces to

$$\textstyle\sum_{k=0}^{m_1}\sum_{l=0}^{m_2}\mathbf{H}_{k,l}\mathbf{c}_{k+i,l+j}=\mathbf{0}_{r\times1},\ \forall0\le i\le L_1-1,0\le j\le L_2-1. \tag{40}$$

Now,

$$\begin{aligned}\left\langle \mathbf{F}\left(U,V\right),\mathbf{c}\left(U,V\right)\right\rangle_s=&\left\langle \textstyle\sum_{i=0}^{m_1}\sum_{j=0}^{m_2}\mathbf{H}_{i,j}U^iV^j,\sum_{i=0}^{L_1-1}\sum_{j=0}^{L_2-1}\mathbf{c}_{i,j}U^iV^j\right\rangle_s\\ =&\textstyle\sum_{i=0}^{L_1-1}\sum_{j=0}^{L_2-1}\left(\sum_{k=0}^{m_1}\sum_{l=0}^{m_2}\mathbf{H}_{k,l}\mathbf{c}_{i-k,j-l}\right)U^iV^j\\ =&\mathbf{0}_{r\times1},\end{aligned} \tag{41}$$

and the theorem follows from (41). □

Example 5. *The characteristic function of the $d \times d$ toric code is given by*

$$\mathbf{F}\left(U,V\right) = \begin{bmatrix} X(1+V) & X(V+U) \\ Z(V+U) & Z(1+V)U \end{bmatrix}. \quad (42)$$

The characteristic function $\mathbf{c}\left(U,V\right)$ of a codeword satisfies

$$\begin{aligned} \left\langle \mathbf{F}\left(U,V\right), \mathbf{c}\left(U,V\right) \right\rangle_s &= \left\langle \mathbf{F}\left(U,V\right), \begin{bmatrix} Xf_X + Yf_Y + Zf_Z & Xg_X + Yg_Y + Zg_Z \end{bmatrix} \right\rangle_s \\ &= \begin{bmatrix} (1+V)(f_Y+f_Z) + (V+U)(g_Y+g_Z) \\ (V+U)(f_X+f_Y) + (1+V)U(g_X+g_Y) \end{bmatrix} = \mathbf{0}_{r\times 1}, \end{aligned} \quad (43)$$

where f_X, f_Y, f_Z and g_X, g_Y, g_Z are polynomials in indeterminates U, V with binary coefficients.

We notice that $(f_X, f_Y, f_Z) = (UV, U, V)$, $(g_X, g_Y, g_Z) = (U, V, 1)$ satisfy (43) and is thus a codeword with characteristic function $\mathbf{c}(U,V) = [XUV + YU + ZV, XU + YV + Z]$.

Set $\mathbf{s}(U,V) = [UV, UV]$, and observe that $\mathbf{c}(U,V) = \mathbf{s}(U,V)\mathbf{F}\left(U^{-1}, V^{-1}\right)$. It follows that the associated codeword is a stabilizer.

Given a polynomial $\mathbf{F}(U,V)$ with coefficients in $\mathcal{P}^{r\times n}$, let m_1, m_2 respectively be the maximal degrees of the indeterminates U, V. We replace U^sV^t in $\mathbf{F}(U,V)$ by $U^{m_1-s}V^{m_2-t}$ to obtain the **complementary polynomial** $\bar{\mathbf{F}}(U,V)$. We observe

$$\bar{\mathbf{F}}(U,V) = U^{m_1}V^{m_2}\mathbf{F}(U^{-1}, V^{-1}). \quad (44)$$

Example 6. (2D-SC HGP Codes) *Given $\mathbf{A}(U,V) \in \mathbb{F}_2^{r_1\times n_1}[U,V]$ and $\mathbf{B}(U,V) \in \mathbb{F}_2^{r_2\times n_2}[U,V]$, define:*

$$\mathbf{F}(U,V) = \begin{bmatrix} X(\mathbf{I}_{n_2\times n_2} \otimes \mathbf{A}(U,V)) & X(\bar{\mathbf{B}}(U,V)^T \otimes \mathbf{I}_{r_1\times r_1}) \\ Z(\mathbf{B}(U,V) \otimes \mathbf{I}_{n_1\times n_1}) & Z(\mathbf{I}_{r_2\times r_2} \otimes \bar{\mathbf{A}}(U,V)^T) \end{bmatrix}. \quad (45)$$

We now show that the 2D-SC hypergraph product code is a stabilizer code by verifying that

$$\left\langle \mathbf{F}(U,V), \mathbf{F}(U^{-1}, V^{-1}) \right\rangle_s = \mathbf{0}_{(r_1n_2+r_2n_1)\times(r_1n_2+r_2n_1)}. \quad (46)$$

We have

$$\left\langle \mathbf{F}(U,V), \mathbf{F}(U^{-1}, V^{-1}) \right\rangle_s = \left[\begin{array}{c|c} \mathbf{0}_{r_1n_2\times r_1n_2} & \begin{matrix} \mathbf{B}(U^{-1},V^{-1})^T \otimes \mathbf{A}(U,V) \\ +\bar{\mathbf{B}}(U,V)^T \otimes \bar{\mathbf{A}}(U^{-1},V^{-1}) \end{matrix} \\ \hline \begin{matrix} \mathbf{B}(U,V) \otimes \mathbf{A}(U^{-1},V^{-1})^T \\ +\bar{\mathbf{B}}(U^{-1},V^{-1}) \otimes \bar{\mathbf{A}}(U,V)^T \end{matrix} & \mathbf{0}_{r_2n_1\times r_2n_1} \end{array}\right].$$

Given a polynomial $\mathbf{F}(U,V)$ with coefficients in $\mathcal{P}^{r\times n}$, let m_1, m_2 respectively be the maximal degrees of the indeterminates U, V. We replace U^sV^t in $\mathbf{F}(U,V)$ by $U^{m_1-s}V^{m_2-t}$ to obtain the **complementary polynomial** $\bar{\mathbf{F}}(U,V)$. We observe

$$\bar{\mathbf{F}}(U,V) = U^{m_1}V^{m_2}\mathbf{F}(U^{-1}, V^{-1}). \quad (47)$$

Example 7. (2D-SC HGP Codes) *Given* $\mathbf{A}(U,V) \in \mathbb{F}_2^{r_1\times n_1}[U,V]$ *and* $\mathbf{B}(U,V) \in \mathbb{F}_2^{r_2\times n_2}[U,V]$*, define:*

$$\mathbf{F}(U,V) = \begin{bmatrix} X(\mathbf{I}_{n_2\times n_2}\otimes \mathbf{A}(U,V)) & X(\bar{\mathbf{B}}(U,V)^T\otimes \mathbf{I}_{r_1\times r_1}) \\ Z(\mathbf{B}(U,V)\otimes \mathbf{I}_{n_1\times n_1}) & Z(\mathbf{I}_{r_2\times r_2}\otimes \bar{\mathbf{A}}(U,V)^T) \end{bmatrix}. \tag{48}$$

We now show that the 2D-SC hypergraph product code is a stabilizer code by verifying that $\left\langle \mathbf{F}(U,V), \mathbf{F}(U^{-1},V^{-1})\right\rangle_s = \mathbf{0}_{(r_1n_2+r_2n_1)\times(r_1n_2+r_2n_1)}$. We have

$$\left\langle \mathbf{F}(U,V), \mathbf{F}(U^{-1},V^{-1})\right\rangle_s = \left[\begin{array}{c|c} \mathbf{0}_{r_1n_2\times r_1n_2} & \begin{array}{c}\mathbf{B}(U^{-1},V^{-1})^T\otimes \mathbf{A}(U,V) \\ +\bar{\mathbf{B}}(U,V)^T\otimes \bar{\mathbf{A}}(U^{-1},V^{-1})\end{array} \\ \hline \begin{array}{c}\mathbf{B}(U,V)\otimes \mathbf{A}(U^{-1},V^{-1})^T \\ +\bar{\mathbf{B}}(U^{-1},V^{-1})\otimes \bar{\mathbf{A}}(U,V)^T\end{array} & \mathbf{0}_{r_2n_1\times r_2n_1} \end{array}\right].$$

It follows from (47) that the off diagonal matrices vanish. At the bottom left we obtain

$$\mathbf{B}(U,V)\otimes \mathbf{A}(U^{-1},V^{-1})^{\mathrm{T}} + U^{-m_1}V^{-m_2}\mathbf{B}(U,V)\otimes \left(U^{m_1}V^{m_2}\mathbf{A}(U^{-1},V^{-1})^{\mathrm{T}}\right) = \mathbf{0}_{r_2n_1\times n_1r_2}. \tag{49}$$

At the top right we obtain

$$\mathbf{B}(U^{-1},V^{-1})^{\mathrm{T}}\otimes \mathbf{A}(U,V) + U^{m_1}V^{m_2}\mathbf{B}(U^{-1},V^{-1})^{\mathrm{T}}\otimes \left(U^{-m_1}V^{-m_2}\mathbf{A}(U,V)\right) = \mathbf{0}_{r_1n_2\times r_2n_1}. \tag{50}$$

Remark 6. ***(Quasi-abelian lifted product codes are finite-dimensional SC-HGP codes)*** *Observe that the aforementioned construction resembles the representation of lifted product codes. Using the fact that every finitely generated abelian group is a direct product of cyclic groups, quasi-abelian lifted product (LP) codes proposed in [22] can be viewed as finite-dimensional SC-HGP codes. This means that it is possible to efficiently construct high-performance quasi-abelian LP codes through cycle optimization methods developed for classical SC-LDPC codes. Section 5 provides a framework for the 2D case.*

Construction 1. ***(2D-SC XYZ Codes)*** *Given* $\mathbf{A}(U,V) \in \mathbb{F}_2^{r_1\times n_1}[U,V]$, $\mathbf{B}(U,V) \in \mathbb{F}_2^{r_2\times n_2}[U,V]$, *and* $\mathbf{C}(U,V) \in \mathbb{F}_2^{r_3\times n_3}[U,V]$*, define:*

$$\mathbf{F}(U,V) = \begin{bmatrix} X(\mathbf{A}\otimes \mathbf{I}_{n_2}\otimes \mathbf{I}_{n_3}) & I_{r_1n_2n_3\times n_1r_2r_3} & Z(\mathbf{I}_{r_1}\otimes \mathbf{I}_{n_2}\otimes \bar{\mathbf{C}}^T) & Y(\mathbf{I}_{r_1}\otimes \bar{\mathbf{B}}^T\otimes \mathbf{I}_{n_3}) \\ Y(\mathbf{I}_{n_1}\otimes \mathbf{B}\otimes \mathbf{I}_{n_3}) & Z(\mathbf{I}_{n_1}\otimes \mathbf{I}_{r_2}\otimes \bar{\mathbf{C}}^T) & I_{n_1r_2n_3\times r_1n_2r_3} & X(\bar{\mathbf{A}}^T\otimes \mathbf{I}_{r_2}\otimes \mathbf{I}_{n_3}) \\ Z(\mathbf{I}_{n_1}\otimes \mathbf{I}_{n_2}\otimes \mathbf{C}) & Y(\mathbf{I}_{n_1}\otimes \bar{\mathbf{B}}^T\otimes \mathbf{I}_{r_3}) & X(\bar{\mathbf{A}}^T\otimes \mathbf{I}_{n_2}\otimes \mathbf{I}_{r_3}) & I_{n_1n_2r_3\times r_1r_2n_3} \\ I_{r_1r_2r_3\times n_1n_2n_3} & X(\mathbf{A}\otimes \mathbf{I}_{r_2}\otimes \mathbf{I}_{r_3}) & Y(\mathbf{I}_{r_1}\otimes \mathbf{B}\otimes \mathbf{I}_{r_3}) & Z(\mathbf{I}_{r_1}\otimes \mathbf{I}_{r_2}\otimes \mathbf{C}) \end{bmatrix}. \tag{51}$$

The 2D-SC XYZ code obtained from $\mathbf{F}(U,V)$ *is a stabilizer code.*

Example 8. *Let* $e,g,y \in \mathbb{N}^*$*, with* $\sigma = \sigma_z$ *as defined in (1).* $\mathbf{F}(U,V)$ *below is a lifting of the*

characteristic function in (34).

$$\mathbf{F}(U,V) = \begin{bmatrix} X(\sigma^{e+y}+\sigma^{2e-g}U) & Y(\sigma^{e}+\sigma^{e}V) & Z\sigma^{e} \\ Y(\sigma^{e}UV) & Z(\sigma^{g}V) & X(\sigma^{g}+\sigma^{e-y}UV) \\ Z(\sigma^{g+y}U+\sigma^{g+y}UV+\sigma^{e}U^2V^2) & X(\sigma^{2g-e+y}V+\sigma^{g}UV^2+\sigma^{e-y}U^2V^2) & Y(\sigma^{e-y}U^2V+\sigma^{g}U) \end{bmatrix}. \tag{52}$$

Remark 7. ***(SC-Generalized Bicycle Codes as 1D-SC-QLDPC Codes)*** *Note that we can also specify characteristic functions of SC-QLDPC codes with arbitrary dimensions by changing the number of indeterminates (which should be equal to the dimension). Several codes in the literature could be categorized as 1D-SC-QLDPC codes, including Generalized Bicycle (GB) codes. The characteristic function of a GB code has the general form:*

$$\mathbf{F}(U) = \begin{bmatrix} Xa(U) & Xb(U) \\ ZU^{L}b(U^{-1}) & ZU^{L}a(U^{-1}) \end{bmatrix}. \tag{53}$$

In particular, the characteristic function of the $[[126, 28]]$ *GB code in [58] is specified as follows with* $m = 62$ *and* $L = 63$*:*

$$\begin{aligned} \mathbf{F}(U) &= \begin{bmatrix} X(1+U+U^{14}+U^{16}+U^{22}) & X(1+U^{3}+U^{13}+U^{20}+U^{42}) \\ Z(1+U^{63-3}+U^{63-13}+U^{63-20}+U^{63-42}) & Z(1+U^{63-1}+U^{63-14}+U^{63-16}+U^{63-22}) \end{bmatrix} \\ &= \begin{bmatrix} X(1+U+U^{14}+U^{16}+U^{22}) & X(1+U^{3}+U^{13}+U^{20}+U^{42}) \\ Z(1+U^{60}+U^{50}+U^{43}+U^{21}) & Z(1+U^{62}+U^{49}+U^{47}+U^{41}) \end{bmatrix}. \end{aligned} \tag{54}$$

5 Code Optimization

In this section, we develop a framework for optimizing the SC-HGP codes introduced in Section 4.2. We start from the alternating sum condition that specifies when a closed path in the base matrix/protograph gives rise to a closed path/cycle in the protograph/Tanner graph. We then develop an optimization algorithm that efficiently reduces the number of short cycles.

5.1 Cycle Conditions

In classical coding theory, short cycles in Tanner graphs create problems for belief propagation (BP) decoders in both the waterfall and error floor regions. Reducing the number of cycles in the Tanner graph is a major goal in optimizing LDPC codes, but efficiently reducing the number of cycles in general LDPC codes is not an easy task. Progressive edge growth (PEG) is an algorithm that efficiently improves the **girth** (the minimum length of a cycle) of the Tanner graph. However, girth is not the only metric that determines performance since multiplicities of cycles are also important. For example, a code with girth 6 than contains 1000 cycles-6 can perform worse than a code with 1 cycle-4 and 2 cycles-6 if the channel is good enough. The convolutional structure of SC codes implies that the number of cycles can be represented by the base matrix, partitioning matrix, and the lifting matrix in a compact way, which facilitates enumeration and optimization of short cycles.

In particular, each cycle in the Tanner graph is lifted from a closed path in the protograph, and the latter arises from another closed path in the base matrix. Closed paths are cycles that allow repetitive nodes, they are also referred to as **cycle candidates**. Whether a cycle candidate in the base matrix/protograph gives rise to a cycle in the protograph/Tanner graph is determined by the alternating sum of entries along the cycle in the partitioning/lifting matrix.

Example 9. *The matrix* $\mathbf{H}$ *given in (55) specifies a 1D-TB code with* $m = 2$ *and* $L = 4$*. The base matrix* $\mathbf{H}_0 + \mathbf{H}_1 + \mathbf{H}_2$ *is the* 3×4 *matrix*

with each entry equal to 1. *Entries* $0, 1, 2$ *in the partitioning matrix* $\mathbf{P}$ *give rise to a* 1 *at the same location in the component matrices* $\mathbf{H}_0, \mathbf{H}_1$, *and* $\mathbf{H}_2$, *respectively. We highlight cycle candidates of lengths* 4, 6, *and* 8 *(also referred to as cycles-*4, *cycles-*6, *and cycles-*8 *candidates in the rest of the paper) in blue, red, and green respectively.*

$$\mathbf{H} = \left[\begin{array}{cccc|cccc|cccc|cccc}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
\hline
0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{array}\right], \quad \mathbf{P} = \left[\begin{array}{cccc} 0 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{array}\right]. \tag{55}$$

The alternating sums of the blue, red, and green cycle candidates are

$$\begin{aligned} &2 - 1 + 0 - 1 = 0, \\ &0 - 0 + 2 - 1 + 0 - 1 = 0, \\ &0 - 1 + 1 - 0 + 1 - 2 + 2 - 1 = 0. \end{aligned}$$

All give rise to cycles in the Tanner graph, and each cycle candidate gives rise to four cycles.

Lemma 1 provides a necessary and sufficient condition for a cycle candidate in the base matrix to give rise to a cycle in the protograph (see [59, 60] for more details). Since we have not constrained the SC-QLDPC codes to be quasi-cyclic, the protograph is exactly the Tanner graph.

Lemma 1. ***(alternating sum condition)*** *Let* $g \geq 2$, *and let* $(j_1, i_1, j_2, i_2, \ldots, j_g, i_g)$ *be a cycle-*$2g$ *candidate* C *in the base matrix, where* $j_{g+1} = j_1$ *and* (i_k, j_k), (i_k, j_{k+1}), $1 \leq k \leq g$, *are nodes of* C *in the partitioning matrix* $\mathbf{P}$. *Then* C *becomes a cycle-*$2g$ *in the protograph if and only if*

$$\sum_{k=1}^{g} \Big((\mathbf{P})_{i_k, j_k} - (\mathbf{P})_{i_k, j_{k+1}} \Big) = 0 \tag{56}$$

for NTB codes, or

$$\sum_{k=1}^{g} \Big((\mathbf{P})_{i_k, j_k} - (\mathbf{P})_{i_k, j_{k+1}} \Big) = 0 \pmod{L} \tag{57}$$

for TB codes.

The k-th term in the alternating sum is the horizontal displacement of node $2k$ with respect to node $(2k-1)$ (interpreted $\bmod L$ for TB codes) on C. It therefore follows that when C is a closed path, the alternating sum is zero. For a TB 2D-SC code, the alternating sum of the terms $(\mathbf{P})_{i_k, j_k} - (\mathbf{P})_{i_k, j_{k+1}}$ is required to be $(0, 0)$ $\bmod (L_1, L_2)$.

5.2 Optimization of SC-QLDPC Codes

In this section, we focus on optimization of cycles of length $4, 6, 8$ in 2D-SC-HGP codes (essentially quasi-abelian lifted product codes on a group that is a product of two cyclic groups of sizes L_1 and L_2) introduced in Section 4. Recall that the characteristic function $\mathbf{F}(U, V)$ is given by

$$\mathbf{F}(U, V) = \begin{bmatrix} X(\mathbf{I}_{n_2 \times n_2} \otimes \mathbf{A}(U, V)) & X(\bar{\mathbf{B}}(U, V)^T \otimes \mathbf{I}_{r_1 \times r_1}) \\ Z(\mathbf{B}(U, V) \otimes \mathbf{I}_{n_1 \times n_1}) & Z(\mathbf{I}_{r_2 \times r_2} \otimes \bar{\mathbf{A}}(U, V)^T) \end{bmatrix}, \tag{58}$$

where $\mathbf{A}(U,V) \in \mathbb{F}_2^{r_1 \times n_1}[U,V]$ and $\mathbf{B}(U,V) \in \mathbb{F}_2^{r_2 \times n_2}[U,V]$.

We now illustrate how to derive the partitioning matrix $\mathbf{P}$ from a characteristic function $\mathbf{F}(U,V)$ of the form (58). Note that $\mathbf{F}(U,V)$ is completely determined by $\mathbf{A}(U,V)$ and $\mathbf{B}(U,V)$ since $\bar{\mathbf{A}}(U,V)$ and $\bar{\mathbf{B}}(U,V)$ are simply $\mathbf{A}(U,V)$ and $\mathbf{B}(U,V)$, with U^iV^j replaced by $U^{m_1-i}V^{m_2-j}$. For simplicity, we consider an example where the (s,t)-th entry of $\mathbf{F}(U,V)$ is a monomial $F_{s,t}U^iV^j$. The (s,t)-th entry of the partitioning matrix $\mathbf{P}$ is then (i,j), and $\mathbf{P}$ is completely determined by two partitioning matrices $\mathbf{P}_a$ and $\mathbf{P}_b$ corresponding to $\mathbf{A}(U,V)$ and $\mathbf{B}(U,V)$ respectively ($\mathbf{P}$ is essentially the hypergraph product of $\mathbf{P}_a$ and $\mathbf{P}_b$).

Example 10. *Here $r_1 = r_2 = 2$, $n_1 = n_2 = 3$, with base matrices*

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

The characteristic functions are given by

$$\mathbf{A}(U,V) = \begin{bmatrix} U^{a_1}V^{a_1'} & 0 & U^{a_2}V^{a_2'} \\ U^{a_3}V^{a_3'} & U^{a_4}V^{a_4'} & 0 \end{bmatrix},$$

$$\mathbf{B}(U,V) = \begin{bmatrix} U^{b_1}V^{b_1'} & U^{b_2}V^{b_2'} & 0 \\ 0 & U^{b_3}V^{b_3'} & U^{b_4}V^{b_4'} \end{bmatrix}.$$

The two partitioning matrices $\mathbf{P}_a$ and $\mathbf{P}_b$ are given by

$$\mathbf{P}_a = \begin{bmatrix} (a_1,a_1') & & (a_2,a_2') \\ (a_3,a_3') & (a_4,a_4') & \end{bmatrix},$$

$$\mathbf{P}_b = \begin{bmatrix} (b_1,b_1') & (b_2,b_2') & \\ & (b_3,b_3') & (b_4,b_4') \end{bmatrix}.$$

The projection of the partitioning matrix $\mathbf{P}$ onto the first dimension is then given by (59). Variable nodes (VNs) are labelled with (j_1,j_2) or (i_1,i_2), check nodes (CNs) are labelled with (i_1,j_2) or (j_1,i_2). The Kronecker product structure is evident – three copies of $\mathbf{P}_a$ along the diagonal of the top left block, and each entry of $\mathbf{P}_b$ is replaced by a 3×3 diagonal matrix. We distinguish four types of node/edge corresponding to the four blocks in (58):

1. *(j_1,j_2)–(i_1,j_2), where i_1–j_1 is an edge in $\mathbf{A}$, corresponding to entries in the top left block $\mathbf{I}_{n_2\times n_2} \otimes \mathbf{A}(U,V)$;*
2. *(j_1,j_2)–(j_1,i_2), where i_2–j_2 is an edge in $\mathbf{B}$, corresponding to entries in the bottom left block $\mathbf{B}(U,V) \otimes \mathbf{I}_{n_1\times n_1}$;*
3. *(i_1,i_2)–(i_1,j_2), where i_1–j_1 is an edge in $\mathbf{B}$, corresponding to entries in the top right block $\bar{\mathbf{B}}(U,V)^T \otimes \mathbf{I}_{r_1\times r_1}$;*
4. *(i_1,i_2)–(j_1,i_2), where i_1–j_1 is an edge in $\mathbf{A}$, corresponding to entries in the bottom right block $\mathbf{I}_{r_2\times r_2} \otimes \bar{\mathbf{A}}(U,V)^T$.*

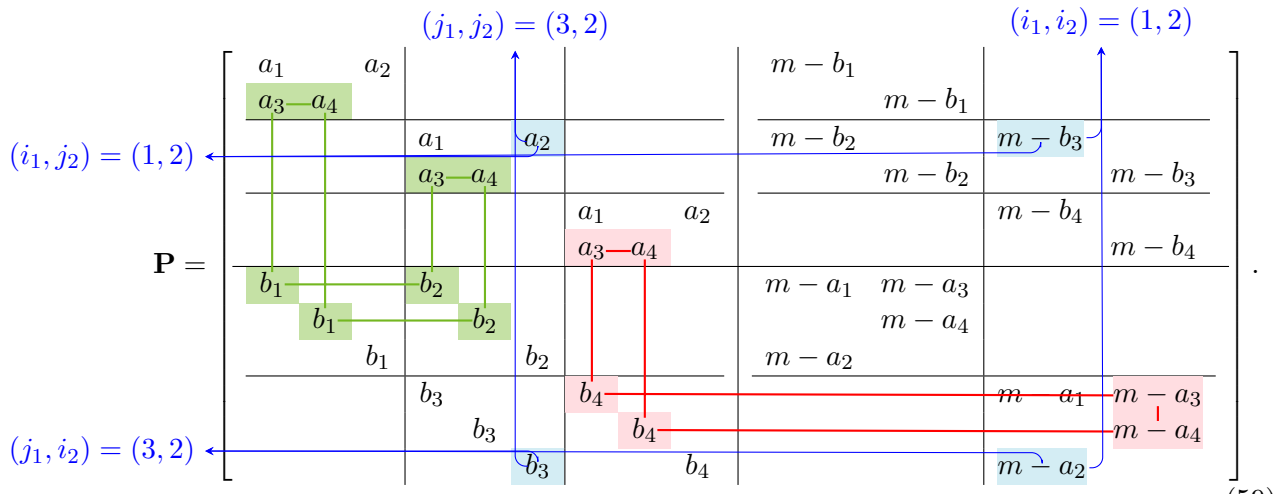
The Kronecker product structure makes it possible to label the entries of the partitioning matrix $\mathbf{P}$. For example, the type (2) blue shaded entry b_3 is in the second row and second column of $\mathbf{P}_b$, so $i_2 = j_2 = 2$. It is the third diagonal entry so $j_1 = 3$. Similarly, the type (1) blue shaded entry a_2 is in the second copy of $\mathbf{P}_a$ so $j_2 = 2$. It is in the entry in row 1 and column 3 of $\mathbf{P}_a$ so $i_1 = 1$ and $j_1 = 3$.

We now enumerate cycles-4, cycles-6, and cycles-8 in SC-HGP codes. According to Lemma 1, a cycle-$2g$ candidate $x_1, x_2, \ldots, x_{2g}$ becomes a cycle-$2g$ in the SC-HGP code if $\sum_{i=1}^{2g}(-1)^i x_i = 0 \mod L$.

We highlighted three cycle candidates of lengths 4 (blue), 6 (red), and 8 (green) in (59), respectively. Recall the alternating sum conditions for each of them giving rise to a cycle in the SC-HGP codes: the blue cycle needs $a_2 - (m - b_3) + (m - a_2) - b_3 = 0 \mod L$, the red cycle needs $a_3 - a_4 + b_4 - (m - a_4) + (m - a_3) - b_4 = 0 \mod L$, and the green cycle needs $a_3 - a_4 + b_1 - b_2 + a_4 - a_3 + b_2 - b_1 = 0 \mod L$. Notice that parameters in the alternating sums for highlighted cycle candidates cancel so the cycle conditions are always satisfied regardless of the exact assignment of $\mathbf{P}_a$ and $\mathbf{P}_b$ once $\mathbf{A}$ and $\mathbf{B}$ are fixed. As an result, the multiplicities of these cycles are only dependent on the base matrices $\mathbf{A}$ and $\mathbf{B}$: we call cycles with this property **rigid cycles**. For example, each pair of nonzero entries in $\mathbf{A}$ and $\mathbf{B}$ results in L rigid cycles-4 in the SC-HGP code, thus the total number of rigid cycles-4 is $4 \times 4 \times L = 16L$.

Similarly, we refer to the cycles whose multiplicities are dependent on the choice of $\mathbf{P}_a$ and $\mathbf{P}_b$ as **flexible cycles**. Example 11 shows some examples of flexible cycle-6 and cycles-8 candidates in an SC-HGP code. Then, our objective is to find $\mathbf{P}_a$ and $\mathbf{P}_b$ that minimize the total number

of flexible cycles in the resultant SC-HGP code, given the structure of an underlying HGP base matrix and the SC parameters, i.e., given $\mathbf{A}$, $\mathbf{B}$, m_1, m_2, L_1, and L_2. In Example 11, we show how flexible cycles can either be a single cycle candidate from $\mathbf{A}$ or $\mathbf{B}$, or be decomposed into a pair of node/edge/cycle candidate from $\mathbf{A}$ and $\mathbf{B}$.



(59)

Example 11. *Fig. 5 shows how flexible cycle (candidates) in SC-HGP codes can be decomposed into components in* $\mathbf{P}_a$ *and* $\mathbf{P}_b$.

We calculate alternating sums of the orange and green cycle-6 candidates in Fig. 5(a) to obtain

$$\begin{aligned} &a_1 - b_3 + (m - a_2) - (m - a_4) + b_3 - a_3 \\ =&a_1 + a_4 - a_2 - a_3, \text{ and} \\ &a_4 - a_3 + (m - b_3) - (m - a_1) + b_3 - a_2 \\ =&a_1 + a_4 - a_2 - a_3. \end{aligned}$$

A flexible cycle-6 arise from these cycle candidates can be decomposed into a cycle-4 candidate in $\mathbf{P}_a$ *that satisfies the alternating sum condition* $a_1 + a_4 - a_2 - a_3 = 0 \mod L$ *along with an entry* b_3 *in* $\mathbf{P}_b$.

Let $s_a = a_1 - a_2 + a_4 - a_3$ *and* $s_b = b_1 - b_2 + b_4 - b_3$. *We calculate alternating sums of the red and blue cycle-8 candidates in Fig. 5(a) to obtain*

$$\begin{aligned} &a_1 - (m - b_1) + (m - b_3) - (m - b_4) + (m - a_3) \\ &- (m - a_4) + (m - b_2) - a_2 = s_a + s_b, \text{ and} \\ &a_1 - (m - b_2) + (m - a_3) - (m - a_4) + (m - b_4) \\ &- (m - b_3)) + (m - b_1) - a_2 = s_a - s_b. \end{aligned}$$

The condition of at least one of the cycle candidates determines a cycle-8 in the SC-HGP code is

$$|a_1 + a_4 - a_2 - a_3| = |b_1 + b_4 - b_2 - b_3| = S, \quad (60)$$

and if $S = 0$, *both paths determine cycles-8. The associated cycles-8 can be decomposed into a pair of cycles-4 candidate in* $\mathbf{P}_a$ *and* $\mathbf{P}_b$ *with identical or opposite alternating sums.*

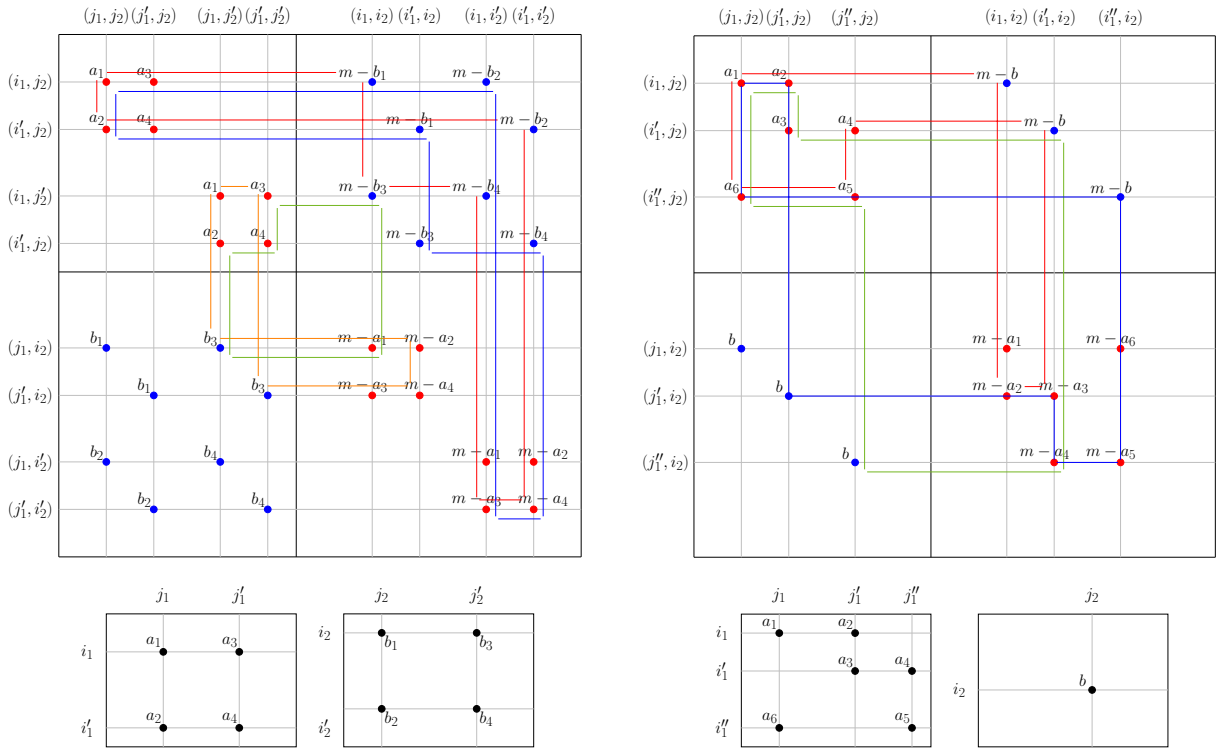
We calculate the alternating sum of the three cycle-8 candidates in Fig. 5(b) to obtain

$$\begin{aligned} &a_1 - a_2 + a_3 - (m - b) + (m - a_4) - b \\ &+ a_5 - a_6 = (a_1 - a_2 + a_3 - a_4 + a_5 - a_6). \end{aligned}$$

A flexible cycles-8 arise from these cycle candidates can be decomposed into a cycle-6 candidate in $\mathbf{P}_a$ *that satisfies the alternating sum condition* $a_1 - a_2 + a_3 - a_4 + a_5 - a_6 = 0 \mod L$ *along with an entry* b *in* $\mathbf{P}_b$.

Let $n^A_{2g,i,j}$ and $n^B_{2g,i,j}$ denote the number of cycle-$2g$ candidates, $g = 2, 3, 4$, with alternating sums i, j (modulo L) respectively in the partitioning matrices $\mathbf{P}_a$, $\mathbf{P}_b$. Let n^A_{2g} and n^B_{2g} denote the total number of cycle-$2g$ candidates in $\mathbf{P}_a$ and $\mathbf{P}_b$, $g = 2, 3, 4$, respectively. Let L_1, L_2 be the coupling lengths of the SC-HGP codes. Each cycle-$2g$ candidate that satisfies the alternating sum condition gives rise to $L = L_1 L_2$ flexible cycles-$2g$ in the resulting SC-HGP code. We define N_{2g} to be the total number of flexible cycles-$2g$ divided by L.

Lemma 2. *Let* e_A, e_B *denote the number of non-zero entries in* $\mathbf{A}$, $\mathbf{B}$, *respectively. Suppose that*



(a) Cycles-4 induced flexible cycle candidates. (b) Cycles-6 induced flexible cycle candidates.

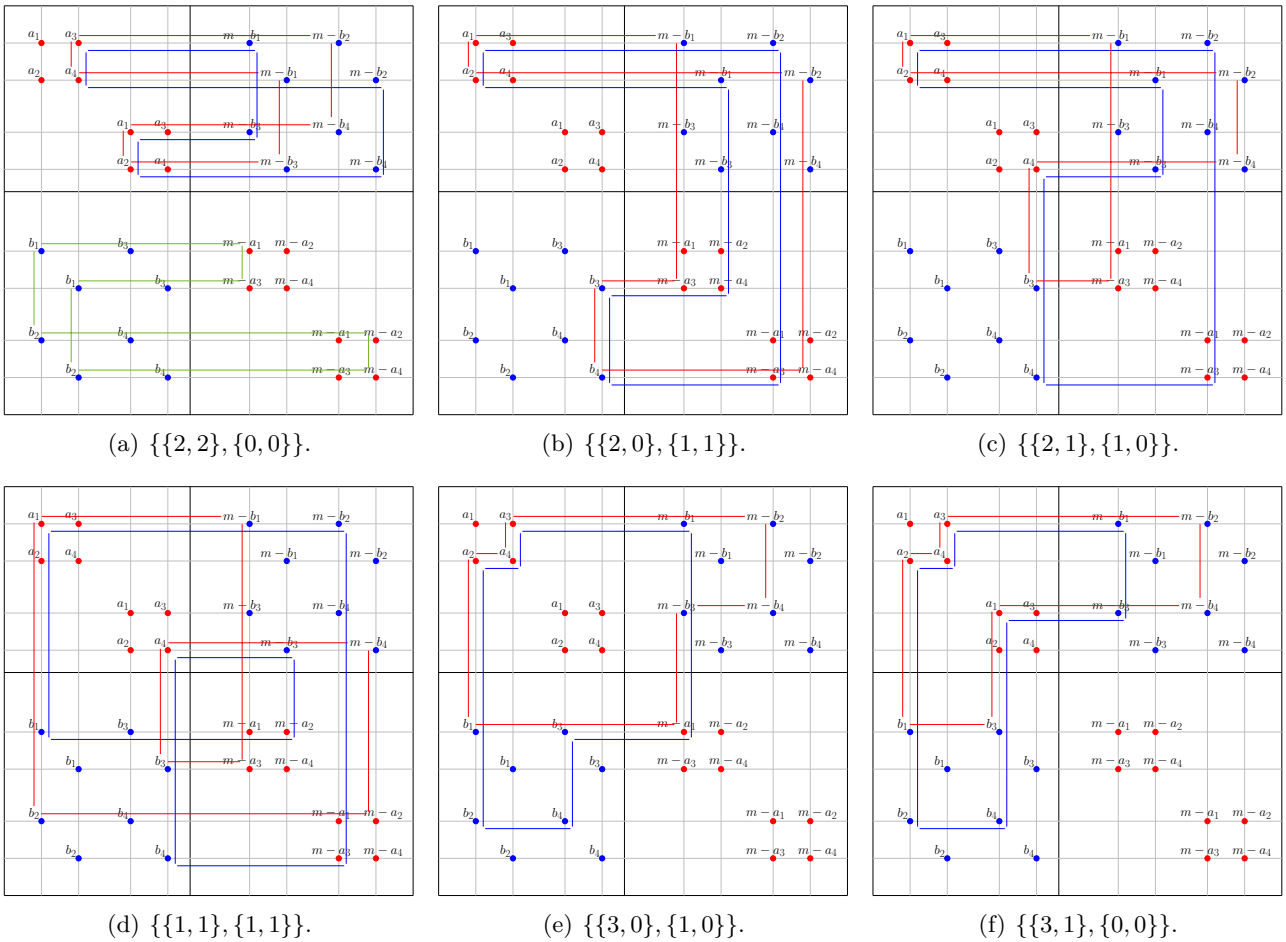
Figure 5: Examples show how flexible cycle candidates of SC-HGP codes can be decomposed into two different components from $\mathbf{P}_a$ and $\mathbf{P}_b$. The matrices in the bottom panels of (a) and (b) are $\mathbf{P}_a$ and $\mathbf{P}_b$, and those in the top panels are $\mathbf{P}$. Nodes from copies of $\mathbf{A}$ and $\mathbf{B}$ are highlighted by red nodes and blue nodes, respectively. The number at each node is the value in its partitioning matrix $\mathbf{P}_a$ or $\mathbf{P}_b$. The red and blue cycles in (a) shows two flexible cycles-8 candidates where each of them is resulting from a pair of cycles-4 candidates from $\mathbf{P}_a$ and $\mathbf{P}_b$. Let $s_a = a_1 - a_2 + a_4 - a_3$, $s_b = b_1 - b_2 + b_4 - b_3$, the alternating sums associated with the blue and red cycle candidates are $s_a - s_b$ and $s_a + s_b$, respectively. The orange and green cycles in (a) shows two flexible cycles-6 candidates where each of them is resulting from a cycle-4 candidate from $\mathbf{P}_a$ and node b_3 from $\mathbf{P}_b$. The alternating sums of both of them are s_a. (b) shows three flexible cycles-8 candidates where each of them is resulting from a cycle-6 candidate from $\mathbf{P}_a$ and an entry b from $\mathbf{P}_b$. The alternating sums of all three cases are $a_1 - a_2 + a_3 - a_4 + a_5 - a_6$.

for any cycle-4 candidate in $\mathbf{P}_a$ and $\mathbf{P}_b$, the alternating sum is non-zero. Then,

$$\begin{aligned} &N_6 = n^A_{6,0,0}(n_2 + r_2) + n^B_{6,0,0}(n_1 + r_1), \\ &N_8 \\ =&n^A_{8,0,0}(n_2 + r_2) + n^B_{8,0,0}(n_1 + r_1) + 30(n^A_{6,0,0}e_B \\ &+ n^B_{6,0,0}e_A) + 124\Big(2n^A_{4,0,0}n^B_{4,0,0} + \\ &\sum_{(i,j)\neq(0,0)} (n^A_{4,i,j} + n^A_{4,-i,-j})(n^B_{4,i,j} + n^B_{4,-i,-j})\Big). \end{aligned} \tag{61}$$

Proof. The first term of each equation corresponds to the number of cycles associated with a cycle candidate with zero alternating sum in $\mathbf{P}_a$ or $\mathbf{P}_b$. A cycle candidate in $\mathbf{A}$ appears n_2 times in $\mathbf{I}_{n_2\times n_2} \otimes \mathbf{A}$ and r_2 times in $\mathbf{I}_{r_2\times r_2} \otimes \mathbf{A}^{\mathrm{T}}$, hence n_2+r_2 times in total. Similarly, a cycle candidate in $\mathbf{B}$ appears $n_1 + r_1$ times in total.

Our initial assumption excludes cycles-6 determined by a cycle-4 candidate in $\mathbf{P}_a$ ($\mathbf{P}_b$) with zero alternating sum and a nonzero entry in $\mathbf{P}_b$ ($\mathbf{P}_a$). This assumption also excludes all cycles-8 determined by a cycle-4 candidate in $\mathbf{P}_a$ ($\mathbf{P}_b$) with zero alternating sum and a node or an edge in $\mathbf{P}_b$ ($\mathbf{P}_a$), since the alternating sum of such a cycle candidate reduces to the alternating sum of the associated cycle-4 candidate, which is assumed to be nonzero. The green cycle in Fig. 6(a) is an example of such a cycle-8 candidate, its alternating sum is $b_1 - (m - a_1) + (m - a_3) - b_1 + b_2 - (m - a_4) + (m - a_2) - b_2 = a_1 + a_4 - a_2 - a_3$. Notice that parameters associated with the edge b_1–b_2 are all cancelled out and only the alternating sum of the cycle-4 candidate a_1–a_2–a_3–a_4 remains in the alternating sum, which is nonzero



(a) $\{\{2,2\},\{0,0\}\}$. (b) $\{\{2,0\},\{1,1\}\}$. (c) $\{\{2,1\},\{1,0\}\}$.

(d) $\{\{1,1\},\{1,1\}\}$. (e) $\{\{3,0\},\{1,0\}\}$. (f) $\{\{3,1\},\{0,0\}\}$.

Figure 6: Different partitioning patterns of cycle-8 candidates resulting from a pair of cycle-4 candidates in $\mathbf{P}_a$ and $\mathbf{P}_b$. The **partitioning pattern** is represented by $\{\{a,b\},\{c,d\}\}$, where each one of a,b,c,d corresponds to the number of nodes in each group of $\{a_1,a_2,a_3,a_4\}$ that form a cycle-4 candidate. The outer bracket separates those correspond to nodes in the top left panel (i.e., $\mathbf{I}_{n_2\times n_2}\otimes\mathbf{A}$) and those correspond to nodes in the bottom right panel (i.e., within $\mathbf{I}_{r_2\times r_2}\otimes\mathbf{A}^{\mathrm{T}}$). For example, in (a), all the nodes associated with a_i, $i=1,\ldots,4$, are in the top left panel, while each group of $\{a_1,a_2,a_3,a_4\}$ in this panel contains 2 nodes of the cycle-8 candidate. Therefore, the partitioning pattern is $\{\{2,2\},\{0,0\}\}$. Similarly, the partitioning pattern of cycles in Fig. 5(a) is $\{\{2,0\},\{2,0\}\}$. The green cycle in (a) is also a cycle-8 candidate, but the alternating sum is not dependent on b_i, $i=1,\ldots,4$, thus these cycles can be easily removed by removing cycle-4 candidates in $\mathbf{P}_a$ with an zero alternating sum.

under the assumption.

We now count cycles-8 determined by a cycle-6 candidate in $\mathbf{P}_a$ ($\mathbf{P}_b$) and a node in $\mathbf{P}_b$ ($\mathbf{P}_a$). The alternating sum of the cycle-6 candidate is zero. The case where the cycle-6 candidate lies in $\mathbf{P}_a$ is illustrated in Fig. 5(b). The different colors of the three cycle-8 candidates correspond to different distributions (i,j) of the six nodes associated with a_i, $i=1,\ldots,6$ among the top left panel and the bottom right panel. Blue corresponds to the distribution $(3,3)$, red to $(4,2)$, and green to $(4,1)$. The distributions $(2,4)$ and $(1,5)$ are not illustrated. Nodes in each panel should be consecutive in the original cycle-6 candidate, so there 6 possibilities for each of the 5 distributions.

Next we count the number of cycles-8 determined by a pair of cycle-4 candidates with identical or opposite alternating sums, one from each partitioning matrix. There are seven different types (shown in Fig. 5(a) and Fig. 6) distinguished by the distribution of nodes associated with a_1,a_2,a_3,a_4, among the four copies of the cycle-4 candidate a_1–a_2–a_4–a_3 in the partitioning matrix. Types are labelled by a **partitioning pattern** $\{\{a,b\},\{c,d\}\}$ where a,b indicate the number of nodes in the two copies of a_1,a_2,a_3,a_4 in the top left panel, and c, d indicate the number of nodes in the two copies of a_1,a_2,a_3,a_4 in the bottom right panel.

1. Fig. 5(a), $\{\{2,0\},\{2,0\}\}$. There are 2×2 ways to choose the copies, after which there are 4 ways to separate a_1, a_2, a_3, a_4. Once the red nodes are chosen, the blue nodes are determined. Therefore, there are $2\times 2\times 4 = 16$ such cycles.

2. Fig. 6(a), $\{\{2,2\},\{0,0\}\}$. There are 2 ways to choose the panel, after which there are 4 ways to separate a_1, a_2, a_3, a_4. Therefore, there are $2\times 4 = 8$ such cycles.

3. Fig. 6(b), $\{\{2,0\},\{1,1\}\}$. Two nodes are clustered in a group from one panel, while each group in the other panel contains one node. There are 4 ways to choose the group containing 2 red nodes and 4 ways to choose the 2 nodes in this group, after which the other two red nodes and all blue nodes are determined (note that only one of the red cycle and the blue cycle can lead to a cycle-8). Therefore, there are $4\times 4 = 16$ such cycles.

4. Fig. 6(c), $\{\{2,1\},\{1,0\}\}$. Two nodes are clustered in a group from one panel, while two groups from different panels contain one node. There are 4 ways to choose the group containing two red nodes, and 4 ways to choose the 2 nodes in this group, after which there are 2 ways to choose the red node in the other group from the current panel. After these choices, the remaining red node and all blue nodes are determined (again, only one of the red cycle and the blue cycle can lead to a cycle-8). Therefore, there are $4\times 4\times 2 = 32$ such cycles.

5. Fig. 6(d), $\{\{1,1\},\{1,1\}\}$. Each group has exactly one red node. There are 4 ways to choose the red node in the top left group, after which all other nodes are determined. Therefore, there are 4 such cycles.

6. Fig. 6(e), $\{\{3,0\},\{1,0\}\}$. Two groups from different panels have 3 red nodes and 1 red node, respectively. There are 4 ways to choose the group containing 3 red nodes and 4 ways to choose the three nodes, after which there are 2 ways to choose the remaining red node. Therefore, there are $4\times 4\times 3 = 32$ such cycles.

7. Fig. 6(f), $\{\{3,1\},\{0,0\}\}$. Two groups from the same panel have 3 and 1 red nodes, respectively. There are 4 ways to choose the group containing the 3 red nodes, and 4 ways to choose the three nodes, after which the remaining nodes are determined. Therefore, there are $4\times 4 = 16$ such cycles.

Each pair of cycle-4 candidates from different partitioning matrices ($\mathbf{P}_a$ and $\mathbf{P}_b$) with identical or opposite alternating sums contributes $16+8+16+32+4+32+16 = 124$ (248 if both alternating sums are zero since each red/blue cycles pair contributes to 2 cycles-8 in this case) cycles-8, thus completes the derivation of N_8. □

We now optimize the partitioning matrix by extending the probabilistic framework developed in [38] to SC-HGP codes. We suppose that each entry in the partitioning matrix $\mathbf{P}_a$ ($\mathbf{P}_b$) is a random variable with probability distribution $\mathbb{P}\left[X^A = (i,j)\right] = p^A_{i,j}$ ($\mathbb{P}\left[X^B = (i,j)\right] = p^B_{i,j}$). We associate the probability distributions for entries of $\mathbf{P}_a$ and $\mathbf{P}_b$ with the **characteristic polynomials**

$$\begin{aligned} f^A(S,T) &= \sum_{i,j=0}^{m_1,m_2} p^A_{i,j} S^i T^j \text{ and} \\ f^B(S,T) &= \sum_{i,j=0}^{m_1,m_2} p^B_{i,j} S^i T^j. \end{aligned} \tag{62}$$

Starting from Theorem 4, we now derive Theorem 4 which expresses the expected number of N_6 (N_8) of flexible cycles-6 (cycles-8) in terms of the characteristic polynomials $f^A(S,T)$ and $f^B(S,T)$. This makes it possible to apply gradient descent (**the gradient descent (GRADE)-algorithmic optimization (AO) method**) to jointly optimize the probability distributions p^A and p^B. The initial distribution is obtained by randomly assigning $\lfloor e_A p^A_{i,j} \rfloor$ or $\lceil e_A p^A_{i,j} \rceil$ ($\lfloor e_B p^B_{i,j} \rfloor$ or $\lceil e_B p^B_{i,j} \rceil$) edges in $\mathbf{A}$ ($\mathbf{B}$) to the (i,j)-th component matrix, where e_A, e_B denote the number of non-zero entries in $\mathbf{A}$, $\mathbf{B}$, respectively.

Theorem 4. *Given a polynomial $f(S,T)$, let $[\cdot]_{i,j}$ denote the coefficient of S^iT^j. Then,*

$$\begin{aligned}
\mathbb{E}_{p^A,p^B}[N_6] =& n_6^A(n_2+r_2)\Big[\big(f^A(S,T)f^A(S^{-1},T^{-1})\big)^3\Big]_{0,0} + n_6^B(n_1+r_1)\Big[\big(f^B(S,T)f^B(S^{-1},T^{-1})\big)^3\Big]_{0,0},\\
\mathbb{E}_{p^A,p^B}[N_8] =& n_8^A(n_2+r_2)\Big[\big(f^A(S,T)f^A(S^{-1},T^{-1})\big)^4\Big]_{0,0} + n_8^B(n_1+r_1)\Big[\big(f^B(S,T)f^B(S^{-1},T^{-})\big)^4\Big]_{0,0}\\
&+30\bigg(n_6^A e_B\Big[\big(f^A(S,T)f^A(S^{-1},T^{-1})\big)^3\Big]_{0,0} + n_6^B e_A\Big[\big(f^B(S,T)f^B(S^{-1},T^{-1})\big)^3\Big]_{0,0}\bigg)\\
&+248n_4^An_4^B\Big[\big(f^A(S,T)f^A(S^{-1},T^{-1})\big)^2\big(f^B(S,T)f^B(S^{-1},T^{-1})\big)^2\Big]_{0,0}.
\end{aligned} \tag{63}$$

Proof. Let $p^A_{2g,i,j}$ ($p^B_{2g,i,j}$) represent the probability that a cycle-$2g$ candidate in $\mathbf{P}_a$ ($\mathbf{P}_b$) has alternating sum (i,j). It follows from [38] that

$$p^A_{2g,i,j} = p^A_{2g,-i,-j} = \Big[\big(f^A(S,T)f^A(S^{-1},T^{-1})\big)^g\Big]_{i,j},\ p^B_{2g,i,j} = p^B_{2g,-i,-j} = \Big[\big(f^B(S,T)f^B(S^{-1},T^{-1})\big)^g\Big]_{i,j}. \tag{64}$$

Since entries in the partitioning matrix are independent, it follows from linearity of expectation that $\mathbb{E}_{p^A}[n^A_{2g,i,j}] = n^A_{2g}p^A_{2g,i,j}$ and $\mathbb{E}_{p^B}[n^B_{2g,i,j}] = n^B_{2g}p^B_{2g,i,j}$. Therefore,

$$\begin{aligned}
\mathbb{E}_{p^A,p^B}[N_6] =& n_6^Ap^A_{6,0,0}(n_2+r_2) + n_6^Bp^B_{6,0,0}(n_1+r_1),\\
\mathbb{E}_{p^A,p^B}[N_8] =& n_8^Ap^A_{8,0,0}(n_2+r_2) + n_8^Bp^B_{8,0,0}(n_1+r_1) + 30(n_6^Ap^A_{6,0}e_B + n_6^Bp^B_{6,0,0}e_A)\\
&+124n_4^An_4^B\Big(2p^A_{4,0,0}p^B_{4,0,0} + \sum_{(i,j)\neq(0,0)}(p^A_{4,i,j}+p^A_{4,-i,-j})(p^B_{4,i,j}+p^B_{4,-i,-j})\Big)\\
=& n_8^Ap^A_{8,0}(n_2+r_2) + n_8^Bp^B_{8,0,0}(n_1+r_1) + 30(n_6^Ap^A_{6,0}e_B + n_6^Bp^B_{6,0,0}e_A)\\
&+248n_4^An_4^B\sum_{(i,j)\neq(0,0)}p^A_{4,i,j}p^B_{4,-i,-j}.
\end{aligned} \tag{65}$$

The result follows from substituting (64) and (65) in Lemma 2. □

Example 12. *Let $\mathbf{1}_{r\times n}$ be the $r\times n$ matrix with every entry equal to 1. We now describe the GRADE algorithm for the special case where $\mathbf{A} = \mathbf{1}_{r_1\times n_1}$ and $\mathbf{B} = \mathbf{1}_{r_2\times n_2}$. It follows from [38] that for $g = 2,3,4$, the number n_{2g} of closed paths of length $2g$ in $\mathbf{1}_{r\times n}$ is given by*

$$\begin{aligned}
n_4 =& \binom{r}{2}\binom{n}{2},\ n_6 = 6\binom{r}{3}\binom{n}{3},\\
n_8 =& \frac{1}{2}\binom{r}{2}\binom{n}{2} + 18\binom{r}{3}\binom{n}{3} + 72\binom{r}{4}\binom{n}{4}\\
&+3\bigg(\binom{r}{2}\binom{n}{3} + \binom{r}{3}\binom{n}{2}\bigg) + 6\bigg(\binom{r}{2}\binom{n}{4}\\
&+\binom{r}{4}\binom{n}{2}\bigg) + 36\bigg(\binom{r}{3}\binom{n}{4} + \binom{r}{4}\binom{n}{3}\bigg).
\end{aligned} \tag{66}$$

Hence

$$\begin{aligned}
w_1 &= \frac{n_6}{n_4} = \frac{2}{3}(r-2)(n-2),\\
w_2 &= \frac{n_8}{n_4} = \frac{1}{2}(r^2-3r+3)(n^2-3n+3),\\
w_3 &= \frac{n+r}{n_4} = \frac{4(n+r)}{r(r-1)n(n-1)},\\
w_4 &= \frac{e}{n_4} = \frac{4}{(r-1)(n-1)}.
\end{aligned} \tag{67}$$

We recall the expressions for N_6, N_8 provided

in Lemma 2, and we minimize the objective function $N = wN_6 + N_8$.

$$\begin{aligned}\frac{\mathbb{E}_{p^A,p^B}[N]}{n_4^A n_4^B} =& w\left(w_1^A w_3^B\left[\left(f^A(S,T)f^A(S^{-1},T^{-1})\right)^3\right]_{0,0} + w_1^B w_3^A\left[\left(f^B(S,T)f^B(S^{-1},T^{-1})\right)^3\right]_{0,0}\right)\\ &+\left(w_2^A w_3^B\left[\left(f^A(S,T)f^A(S^{-1},T^{-1})\right)^4\right]_{0,0} + w_2^B w_3^A\left[\left(f^B(S,T)f^B(S^{-1},T^{-})\right)^4\right]_{0,0}\right)\\ &+30\left(w_1^A w_4^B\left[\left(f^A(S,T)f^A(S^{-1},T^{-1})\right)^3\right]_{0,0} + w_1^B w_4^A\left[\left(f^B(S,T)f^B(S^{-1},T^{-1})\right)^3\right]_{0,0}\right)\\ &+248\left[\left(f^A(S,T)f^A(S^{-1},T^{-1})\right)^2\left(f^B(S,T)f^B(S^{-1},T^{-1})\right)^2\right]_{0,0},\end{aligned} \tag{68}$$

where w_i^A *and* w_i^B *are obtained by replacing* r, n *in (67) with* r_1, n_1 *and* r_2, n_2*, respectively.*

We consider the objective function defined below, and apply gradient descent algorithm to find a locally optimal pair p^A, p^B.

$$\begin{aligned}F(p^A,p^B) =& c_3^A\left[\left(f^A\bar{f}^A\right)^3\right]_{0,0} + c_3^B\left[\left(f^B\bar{f}^B\right)^3\right]_{0,0}\\ &+c_4^A\left[\left(f^A\bar{f}^A\right)^4\right]_{0,0} + c_4^B\left[\left(f^B\bar{f}^B\right)^4\right]_{0,0}\\ &+c\left[\left(f^A\bar{f}^A\right)^2\left(f^B\bar{f}^B\right)^2\right]_{0,0},\end{aligned} \tag{69}$$

where $c_3^A = n_6^A(wn_2 + wr_2 + 30e_B)/(n_4^A n_4^B)$, $c_3^B = n_6^B(wn_1 + wr_1 + 30e_A)/(n_4^A n_4^B)$, $c_4^A = n_8^A(n_2 + r_2)/(n_4^A n_4^B)$, $c_4^B = n_8^B(n_1 + r_1)/(n_4^A n_4^B)$, and $c = 248$[1]. We abbreviate polynomials $f^A(S,T)$, $f^A(S^{-1},T^{-1})$ by f^A, $\bar{f}^A$, respectively, and $f^B(S,T)$, $f^B(S^{-1},T^{-1})$ by f^B, $\bar{f}^B$, respectively.

Given that densities sum up to 0, we define the Lagrangian by

$$\begin{aligned}&L(p^A,p^B)\\ =&F(p^A,p^B) + c_A\left(1 - \textstyle\sum_{i=0}^{m_1}\sum_{j=0}^{m_2} p_{i,j}^A\right)\\ &+c_B\left(1 - \textstyle\sum_{i=0}^{m_1}\sum_{j=0}^{m_2} p_{i,j}^B\right).\end{aligned} \tag{70}$$

Then, the gradient of (70) is obtained by

$$\begin{aligned}\left(\nabla_{p^A}L(p^A,p^B)\right)_{i,j} &= \left(\nabla_{p^A}F(p^A,p^B)\right)_{i,j} - c_A,\\ \left(\nabla_{p^B}L(p^A,p^B)\right)_{i,j} &= \left(\nabla_{p^B}F(p^A,p^B)\right)_{i,j} - c_B,\end{aligned} \tag{71}$$

[1]In the special case where every entry of $\mathbf{A}$ and $\mathbf{B}$ is equal to 1, $c_3^A = ww_1^A w_3^B + 30w_1^A w_4^B$, $c_3^B = ww_1^B w_3^A + 30w_1^B w_4^A$, $c_4^A = w_2^A w_3^B$, $c_4^B = w_2^B w_3^A$, and $c = 248$.

where

$$\begin{aligned}&\left(\nabla_{p^A}F(p^A,p^B)\right)_{i,j}\\ =&6c_3^A\left[\left(f^A\bar{f}^A\right)^2 f^A\right]_{i,j} + 8c_4^A\left[\left(f^A\bar{f}^A\right)^3 f^A\right]_{i,j}\\ &+4c\left[\left(f^A\bar{f}^A\right)f^A\left(f^B\bar{f}^B\right)^2\right]_{i,j}, \text{ and}\\ &\left(\nabla_{p^B}F(p^A,p^B)\right)_{i,j}\\ =&6c_3^B\left[\left(f^B\bar{f}^B\right)^2 f^B\right]_{i,j} + 8c_4^B\left[\left(f^B\bar{f}^B\right)^3 f^B\right]_{i,j}\\ &+4c\left[\left(f^A\bar{f}^A\right)^2\left(f^B\bar{f}^B\right)f^B\right]_{i,j}.\end{aligned} \tag{72}$$

The updating law for p^A is $(p^A)_{i,j}^{(\ell+1)} = (p^A)_{i,j}^{(\ell)} + \alpha\left(\left(\nabla_{p^A}F(p^A,p^B)\right)_{i,j} - c_A\right)$, for some step size $\alpha \in \mathbb{R}$. To preserve the constraints on probability densities, we let c_A be the average of all elements in $\nabla_{p^A}F(p^A,p^B)$ at each step. We refer to this process as the **GRADE** step and details are shown in Algorithm .1.

We initialize partitioning matrices $\mathbf{P}_a$ and $\mathbf{P}_b$ with empirical distributions close to the optimal pair p^A and p^B. We then apply a greedy algorithm (the AO step described in Algorithm .2) that removes short cycles through replacing some entry with a value that reduces the weighted number of short cycles, and terminating when no such improvement is possible. Note that to reduce complexity, we fix $\mathbf{P}_a$ ($\mathbf{P}_b$) and locally optimize $\mathbf{P}_b$ ($\mathbf{P}_a$) alternatively.

Algorithm .1: Gradient-Descent Distributor (GRADE) for Cycle Optimization

Input:

$\mathbf{A}$, $\mathbf{B}$: component matrices of the HGP base matrix;

w: weight of each cycle-6 candidate (assuming that of a cycle-8 is 1);

ϵ, α: accuracy and step size of gradient descent;

Output:

p^A, p^B: locally optimal edge distributions over $\{0,1,\ldots,m_1\}\times\{0,1,\ldots,m_2\}$;

v_{prev}, v_{cur}: the value of the objective function in (69) at the previous and the current iterations;

g_A, g_B: the gradient of the objective function with respect to p^A and p^B, respectively;

r_1, n_1, r_2, n_2: the sizes of $\mathbf{A}$ and $\mathbf{B}$;

e_A, e_B: the number of nonzero elements in $\mathbf{A}$ and $\mathbf{B}$;

n_6^A, n_6^B (n_8^A, n_8^B): the number of length 6 (8) closed paths in $\mathbf{A}$ and $\mathbf{B}$;

Function `initialize()`
- $c_3^A \leftarrow n_6^A(wn_2+wr_2+30e_B)/(n_4^An_4^B)$;
- $c_3^B \leftarrow n_6^B(wn_1+wr_1+30e_A)/(n_4^An_4^B)$;
- $c_4^A \leftarrow n_8^A(n_2+r_2)/(n_4^An_4^B)$;
- $c_4^B \leftarrow n_8^B(n_1+r_1)/(n_4^An_4^B)$;
- $c \leftarrow 248$;
- $v_{prev}, v_{cur} \leftarrow 1$;
- $p^A, p^B \leftarrow \frac{1}{(m_1+1)(m_2+1)}\mathbf{1}_{(m_1+1)\times(m_2+1)}$;
- $g^A, g^B, f^A, \bar{f}^A, f^B, \bar{f}^B \leftarrow \mathbf{0}_{(m_1+1)\times(m_2+1)}$;

While $|v_{prev}-v_{cur}|>\epsilon$ **do**
- obtain $F(p^A,p^B)$ by (69);
- $\nabla_{p^A}F(p^A,p^B)$, $\nabla_{p^B}F(p^A,p^B)$ by (72);
- $v_{prev} \leftarrow v_{cur}$;
- $v_{cur} \leftarrow F(p^A,p^B)$;
- $g^A \leftarrow \nabla_{p^A}F(p^A,p^B)$;
- $g^B \leftarrow \nabla_{p^B}F(p^A,p^B)$;
- $g^A \leftarrow g^A - \mathrm{mean}(g^A)$;
- $g^B \leftarrow g^B - \mathrm{mean}(g^B)$;
- **if** $v_{prev} > v_{cur}$ **then**
  - $\alpha \leftarrow \alpha/2$;
- $p^A \leftarrow p^A - \alpha g^A/||g^A||$;
- $p^B \leftarrow p^B - \alpha g^B/||g^B||$;

return p^A, p^B

Algorithm .2: Cycle-Based GRADE-A Optimizer (AO)

Input:

$\mathbf{A}$, $\mathbf{B}$: Component matrices of the HGP base matrix;

r_1, n_1, r_2, n_2, e_A, e_B: Parameters of $\mathbf{A}$ and $\mathbf{B}$ in Algorithm .1;

m_1, m_2, L_1, L_2: Memories and coupling lengths of the 2D-SC ensemble;

$\mathbf{E}^A$, $\mathbf{E}^B$: Edge lists of $\mathbf{A}$ and $\mathbf{B}$;

p^A, p^B: Edge distributions from Algorithm .1;

w: Weight of each cycle-6 assuming cycle-8 has weight 1;

d_1, d_2: Bounds on search space size;

Output:

$\mathbf{P}_a$, $\mathbf{P}_b$: Locally optimal partitioning matrices for $\mathbf{A}$ and $\mathbf{B}$;

$\mathbf{d}$: Counting vector indicating distance to the initial matrix;

Obtain the lists $\mathcal{L}_{2g}^A$, $\mathcal{L}_{2g}^B$, $g=2,3,4$, of cycle-$2g$ candidates in the base matrix;

Find $(\mathbf{u}^A,\mathbf{u}^B) = \arg\min_{\mathbf{x},\mathbf{y}} ||\frac{1}{e_A}\mathbf{x}-p^A||_2 + ||\frac{1}{e_B}\mathbf{y}-p^B||_2$ where $\|\mathbf{x}\|_1=e_A$, $\|\mathbf{y}\|_1=e_B$;

foreach $0\le i\le m_1$, $0\le j\le m_2$ **do**
- Randomly assign $\mathbf{u}^A[i][j]$ ($\mathbf{u}^B[i][j]$) (i,j)'s to $\mathbf{P}_a$ ($\mathbf{P}_b$);

Initialize $\mathbf{d}\leftarrow\mathbf{0}$ and `noptimal` $\leftarrow$ `False`;

foreach $1\le e_a\le e_A$, $1\le e_b\le e_B$ **do**
- Extract (i_a,j_a) and (i_b,j_b) from $\mathbf{E}^A[e_a]$ and $\mathbf{E}^B[e_b]$;
- Compute alternating sums of cycles in $\mathcal{L}_{2g}$ and count cycle statistics;
- Compute n_4, n_6, and n_8 using (61);
- **foreach** $(v_a,v_b)\in\{0,1,\ldots,m_1\}\times\{0,1,\ldots,m_2\}$ **do**
  - $\mathbf{d}'\leftarrow\mathbf{d}$; $\mathbf{d}'[v_a]\leftarrow\mathbf{d}'[v_a]+1$;
  - **if** $\|\mathbf{d}'\|_1\le d_1$ *and* $\|\mathbf{d}'\|_\infty\le d_2$ **then**
    - Temporarily update $\mathbf{P}_a$ and $\mathbf{P}_b$;
    - Recompute cycle statistics n_4', n_6', n_8', and $n'=wn_6'+n_8'$;
    - **if** $n_4'<n_4$ *or* $n'<n$ **then**
      - `noptimal` $\leftarrow$ `True`;
      - Update n_4, n, $\mathbf{d}$, $\mathbf{P}_a$, $\mathbf{P}_b$;

if *noptimal* **then**
- **goto** step 6;

return $\mathbf{P}_a, \mathbf{P}_b$;

6 Simulation Results

In Section 6.1, we construct SC-HGP codes using the methods described in Section 5.2, and in Section 6.2 we describe a BP decoder for these QLDPC codes. In Section 6.4 we present simulation results which demonstrate the value of optimizing the number of short cycles.

6.1 Constructions

Section 4.2 described how an SC-HGP code is specified by a base matrix $\mathbf{A}$ of size $r_1 \times n_1$, a base matrix $\mathbf{B}$ of size $r_2 \times n_2$, and by the coupling parameters m_1, m_2, L_1, and L_2. The parity check matrix of the HGP base code has $r = r_1 n_2 + r_2 n_1$ rows and $n = r_1 r_2 + n_1 n_2$ columns, thus it encodes at least $k = n - r = (n_1 - r_1)(n_2 - r_2)$ logical qubits. Thus, we obtain a SC-HGP code that encodes at least $K = kL_1L_2 = (n_1 - r_1)(n_2 - r_2)L_1L_2$ logical qubits as $N = (r_1 r_2 + n_1 n_2)L_1L_2$ physical qubits.

We use Algorithm .1 and Algorithm .2 to generate two families of SC-HGP codes. The first family contains seven $[[7300, 2500]]$ codes with $L_1 = L_2 = 10$ and $(r_1, r_2, n_1, n_2) = (3, 3, 8, 8)$, and the second family contains seven $[[5800, 1600]]$ codes with $L_1 = L_2 = 10$ and $(r_1, r_2, n_1, n_2) = (3, 3, 7, 7)$. Table 1 and Table 2 list the number of short cycles (divided by L_1L_2).

Within each family, Code 2 results from drawing partitioning matrices from the uniform distribution, and Code 4 results from drawing partitioning matrices from the locally optimal distribution generated by the GRADE algorithm (Algorithm .1), Codes 1 and 3, respectively, result from applying the AO algorithm (Algorithm .2) to the partitioning matrices that define Codes 2 and 4. Codes 1-4 all have memory $m_1 = m_2 = 2$, Code 5 has memory $m_1 = m_2 = 1$, Code 6 has memory $(m_1, m_2) = (1, 2)$, and Code 7 has memory $m_1 = m_2 = 3$. The partitioning matrices that define Codes 5-7 are generated by GRADE-AO (Algorithms 1 and 2 in combination). Partitioning matrices are specified by equations in the text, and it is these equation numbers that appear in Table 1 and Table 2[2]. Note that $>$ exists in the table since the counting in (61) assumes no existence of cycles-4 after partitioning, while in Codes 2 and 4, there are cycles-4. Therefore, the actual numbers of cycles-8 in Codes 2 and 4 are larger than numbers counted by (61).

Table 1: Information of $[[7300, \geq 2500]]$ codes

Common parameters	$(r_1, n_1, r_2, n_2, L_1, L_2) = (3, 8, 3, 8, 10, 10)$						
Codes	Code 1	Code 2	Code 3	Code 4	Code 5	Code 6	Code 7
Minimum Distances ($\leq$)	10	6	12	8	6	8	20
Partitioning Matrices	(76)	(77)	(78)	(79)	(80)	(81)	(82)
Distributions	Uniform		GRADE				
Optimized by AO	✓	×	✓	×	✓		
(m_1, m_2)	$(2, 2)$				$(1, 1)$	$(1, 2)$	$(3, 3)$
Number of cycles 4	0	110	0	66	0	0	0
Number of cycles 6	11	264	11	143	583	198	0
Number of cycles 8	5113	>48142	5131	>23120	64585	23266	1144

[2]Note that in a SC-QLDPC code with memories (m_1, m_2), any d in $\mathbf{P}_a$ and $\mathbf{P}_b$ is a value from $\{0, 1, \ldots, (m_1+1)(m_2+1)-1\}$, where d refers to $(i, j) = (\lfloor d/(m_2+1) \rfloor, d \mod (m_2+1))$.

Table 2: Information of $[[5800, \geq 1600]]$ codes

Common parameters	$(r_1, n_1, r_2, n_2, L_1, L_2) = (3, 7, 3, 7, 10, 10)$						
Codes	Code 1	Code 2	Code 3	Code 4	Code 5	Code 6	Code 7
Minimum Distances ($\leq$)	10	10	14	10	8	6	16
Partitioning Matrices	(83)	(84)	(85)	(86)	(87)	(88)	(89)
Distributions	Uniform		GRADE				
Optimized by AO	✓	×	✓	×	✓		
(m_1, m_2)	$(2, 2)$				$(1, 1)$	$(1, 2)$	$(3, 3)$
Number of cycles 4	0	30	0	40	0	0	0
Number of cycles 6	0	150	0	140	320	70	0
Number of cycles 8	2316	>17804	2466	>18710	30424	8594	380

6.2 BP Decoders

The belief propagation (BP) decoder is a probabilistic decoder widely used for decoding classical LDPC codes. In algebraic decoders, the decoder fails if the number of errors is at least the minimum distance of the code. However, if different error patterns lead to identical syndromes and their probabilities are different, then it is possible to do better. A maximum likelihood (ML) decoder chooses the error pattern with the highest probability given the obtained syndromes. ML decoder performance is optimal, but can be of high complexity in large codes. The BP decoder is a low complexity approximation of the ML decoder that efficiently estimates the marginal distribution of each symbol through iterations of local updates of messages between VNs and CNs. BP decoders are known to be optimal on acyclic graphs. In a cyclic graph with large girth (the minimum number of nodes on a cycle), the local computing graph of the BP algorithm is close to a tree and can thus still perform quite well.

The BP decoder used for QLDPC codes is shown in Figure 7. Likelihoods of binary symbols in the classical LDPC decoder are replaced by likelihoods of Pauli matrices, so that CN to VN messages become vectors

$$L_{c_j \to v_i} = \left(\log \frac{\mathbb{P}\left[v_i = X | c_j\right]}{\mathbb{P}\left[v_i = I | c_j\right]}, \log \frac{\mathbb{P}\left[v_i = Y | c_j\right]}{\mathbb{P}\left[v_i = I | c_j\right]}, \log \frac{\mathbb{P}\left[v_i = Z | c_j\right]}{\mathbb{P}\left[v_i = I | c_j\right]} \right). \tag{73}$$

The VN-to-CN messages are values denoted by

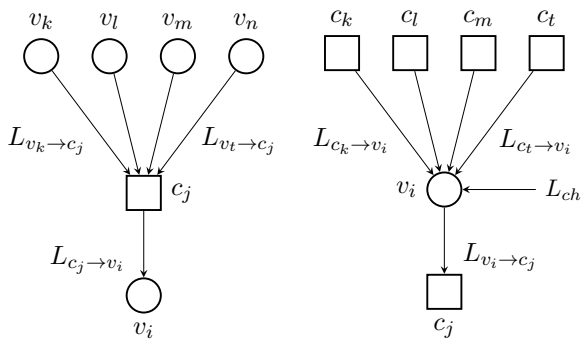
$$L_{v_i \to c_j} = \log \frac{\mathbb{P}\left[\langle v_i, S_{i,j} \rangle_s = 0\right]}{\mathbb{P}\left[\langle v_i, S_{i,j} \rangle_s = 1\right]}. \tag{74}$$

The channel log-likelihood-ratios (LLRs) are defined by

$$L_i^{ch} = \left(\log \frac{\mathbb{P}\left[v_i = X\right]}{\mathbb{P}\left[v_i = I\right]}, \log \frac{\mathbb{P}\left[v_i = Y\right]}{\mathbb{P}\left[v_i = I\right]}, \log \frac{\mathbb{P}\left[v_i = Z\right]}{\mathbb{P}\left[v_i = I\right]} \right).$$

We introduce functions $f : \mathbb{R}^3 \times \mathcal{P} \to \mathbb{R}$ and $g : \mathbb{R} \times \mathcal{P} \to \mathbb{R}^3$ that convert between likelihoods on binary symbols and likelihoods on Pauli matrices:

$$f(l; S) = \begin{cases} \log(1 + e^{l_1}) - \log(e^{l_2} + e^{l_3}), & S = X, \\ \log(1 + e^{l_2}) - \log(e^{l_3} + e^{l_1}), & S = Y, \\ \log(1 + e^{l_3}) - \log(e^{l_1} + e^{l_2}), & S = Z, \end{cases}$$



(a) CN-to-VN update. (b) VN-to-CN update.

Figure 7: Update laws in the QLDPC decoder.

and

$$g(l;S) = \begin{cases} (0,-l,-l), & S=X, \\ (0,-l,0), & S=Y, \\ (-l,-l,0), & S=Z. \end{cases}$$

The update laws are specified by

$$L_{v_i \to c_j} = f(L'_{v_i \to c_j}; S_{i,j}),$$

where $L'_{v_i \to c_j} = L_i^{ch} + \sum_{k \in \mathcal{N}_i \setminus \{j\}} L_{c_k \to v_i}$; and

$$L_{c_j \to v_i} = g(L'_{c_j \to v_i}; S_{i,j}),$$

where $L_{c_j \to v_i} = g(L'_{c_j \to v_i}; S_{i,j})$, and

$$L'_{c_j \to v_i} = \prod_{k \in \mathcal{N}_j \setminus \{i\}} \operatorname{sgn}(L_{v_k \to c_j}) \phi\Big(\sum_{k \in \mathcal{N}_j \setminus \{i\}} \phi(|L_{v_k \to c_j}|) \Big),$$

and $\phi(x) = -\log(\tanh(x/2))$.

6.3 Degenerate Errors

Assume the information sent over the channel is $\mathbf{x}$, and the decoder receives a noisy version $\mathbf{y}$, making an estimation of $\mathbf{x}$ as $\tilde{\mathbf{x}}$. One major difference between quantum error correction (QEC) codes and classical error correction codes (ECC) is as follows: in classical ECC, any $\tilde{\mathbf{x}} \neq \mathbf{x}$ leads to a decoding error; whereas in quantum ECC, $\tilde{\mathbf{x}}$ can still be a correct estimate as long as $\mathbf{x}$ and $\tilde{\mathbf{x}}$ lie in the same coset of the stabilizer group of the QEC code. Thus, in quantum ECC, there is always a potential gain from adjusting $\tilde{\mathbf{x}}$ towards a codeword.

When a BP decoder fails, the estimated vector $\tilde{\mathbf{x}}$ is typically a near-codeword, close to the original $\mathbf{x}$. Due to the presence of degenerate errors in QEC codes, it is often advantageous to push $\tilde{\mathbf{x}}$ toward a nearby codeword $\mathbf{x}'$. Decoding remains successful as long as the difference $(\mathbf{x}' - \mathbf{x})$ lies within the stabilizer group. In highly degenerate codes, algorithms that efficiently push the BP decoder's estimation back into the codespace can significantly reduce the error floor.

Several works have explored post-processing steps to achieve this goal. The Ordered Statistics Decoder (OSD) has been the most widely applied approach [58, 61]. OSD is a post-processing algorithm that runs after BP, and its core idea is to identify a subset S of variable nodes (VNs) that contains a sufficient number of linearly independent columns. This subset enables the estimate to be calculated by inverting a matrix using the syndrome. The subset S is typically chosen from the least reliable VNs, with reliability measured by the log-likelihood ratios (LLRs) produced by the BP decoder.

The Stabilizer Inactivation (SI) decoder [62] addresses cases where an X (Z) stabilizer generator $\mathbf{g}$ has exactly half of its variable nodes (VNs) contained in an X (Z) error $\tilde{\mathbf{x}}$. In such scenarios, $(\tilde{\mathbf{x}} + \mathbf{g})$ and $\tilde{\mathbf{x}}$ represent essentially the same error, but $(\tilde{\mathbf{x}} + \mathbf{g})$ may be a better representative of the coset than $\tilde{\mathbf{x}}$, even though BP chooses $\tilde{\mathbf{x}}$. This situation is more likely when the probabilities of $(\tilde{\mathbf{x}} + \mathbf{g})$ and $\tilde{\mathbf{x}}$ are very close. The SI decoder measures the "closeness" between $(\tilde{\mathbf{x}} + \mathbf{g})$ and $\tilde{\mathbf{x}}$ by summing the absolute values of the log-likelihood ratios (LLRs) of all VNs in the support of $\mathbf{g}$. SI recursively spans $\mathbf{g}$ and identifies the one with the smallest measure, such that flipping the support of $\mathbf{g}$ and rerunning BP may lead to a valid codeword.

Belief Propagation with Guided Decimation (BPGD) takes a different approach to sampling error patterns [63]. Rather than decoding toward the most probable codeword, BPGD aims to sample an error pattern based on the posterior probabilities. This is achieved by sequentially fixing the values of error bits according to their marginals, as estimated by BP. BPGD has demonstrated competitive performance compared to the widely used BP-OSD under both bit-flip noise and depolarizing noise.

In this manuscript, we employ the BP-OSD decoder [64]. However, running the BP-OSD decoder directly on QLDPC codes with lengths exceeding 5000 becomes computationally pro-

hibitive due to its non-linear complexity, making it too slow to gather data points from the error floor region. To address this, instead of simulating with BP-OSD directly, we adopt a two-step approach as follows:

1. We first run the original BP small-set-flip (SSF) decoder—which does not require additional BP iterations [65]. The reason for choosing BP-SSF decoder will be explained later. These steps yield performance curves that are only 0.3-0.6 orders of magnitude away from the BP-OSD curves. We then collect a sufficient number of errors (typically 200-400) that result in decoding failures from the BP-SSF decoder, allowing us to obtain an initial FER p_e^{SSF}.

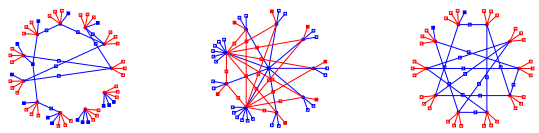
2. We then pass only these errors to BP-OSD to further evaluate the number of unresolved decoding failures. From this, we obtain the ratio of unresolved decoding failures out of the total number of errors, denoted as α_e^{OSD}. Consequently, the real logical error rate under BP-OSD is given by $p_e^{\mathrm{SSF}}\alpha_e^{\mathrm{OSD}}$.

To illustrate the reason for applying BP-SSF decoder, we analyzed the error vectors collected from the non-binary BP decoder. We observed that these error vectors can be divided into three major types, as shown in Fig. 8:

Type 1: All X (Z) errors where some variable nodes (VNs) are connected to more unsatisfied check nodes (CNs) than satisfied CNs. These errors are likely to occur due to slow convergence, often resulting from the existence of very short cycles. By increasing the number of iterations to a very large number—though this approach is inefficient—the errors are likely to decrease. Bit flip (BF) can easily remove these errors.

Type 2: All X (Z) errors where a subset of VNs is completely contained in or has a large overlap with the support of a Z stabilizer, with the size of this subset exceeding half the weight of the stabilizer. Flipping the stabilizer is an efficient method to address these errors.

Type 3: All X (Z) errors that are neither Type I nor Type II, which correspond to classical absorbing sets. These errors represent true absorbing sets or trapping sets that are unsolvable by the BP decoder and require more advanced post-processing steps, as proposed in [58, 62, 63].



(a) Type 1 errors. (b) Type 2 errors. (c) Type 3 errors.

Figure 8: Typical errors obtained through non-binary BP decoder.

Based on the observations above, running the following post-processing steps recursively effectively improves the outcomes of the BP decoder with low complexity, which is exactly the BP-SSF decoder described in [65]:

Step 1: We perform bit flip (BF) on the estimated error. In particular, we check VNs more than half of their neighbors are unsatisfied CNs, and flip one VN that is adjacent to the largest number of unsatisfied CNs. We perform this step recursively until no VNs could be flipped.

Step 2: We perform stabilizer flip if no VNs is flipped in Step 1. Specifically, if there still exist a set S of unsatisfied $X(Z)$ stabilizers (CNs), we check if there also exist any $Z(X)$ stabilizer that more than half of its neighbors are contained in S. This is motivated by the stabilizer inactivation decoder proposed in [62].

Step 3: We perform a shortest path search on the errors if no variable nodes (VNs) are flipped in Steps 1 and 2. Specifically, we identify a shortest path that connects at least two unsatisfied check nodes (CNs) and flip all the VNs along this path. To prevent getting trapped in periodic states, we restrict the SP search to include only those VNs that have been visited at most once.

With the aforementioned SSF post-processing steps, we collect all the errors that lead to decoding failures. We then pass those errors through the BP-OSD software [64] and record the portion

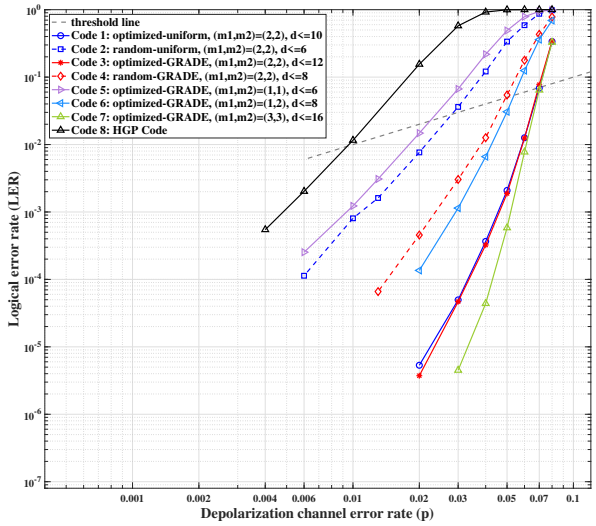


Figure 9: The Logical Error Rate (LER) for $[[7300, \geq 2500]]$ quantum low-density parity-check (QLDPC) codes, using the belief propagation with ordered statistics decoding (BP-OSD) decoder, is presented. Codes 1-7 are spatially coupled (SC) QLDPC codes, while Code 8 is a hypergraph product (HGP) code constructed from the hypergraph of a $[30, 80]$ quasi-cyclic LDPC code with itself.

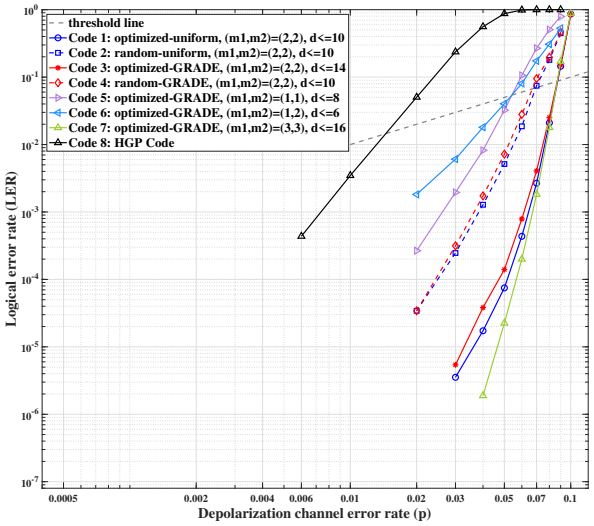


Figure 10: The Logical Error Rate (LER) for $[[5800, \geq 1600]]$ quantum low-density parity-check (QLDPC) codes, using the belief propagation with ordered statistics decoding (BP-OSD) decoder, is presented. Codes 1-7 are spatially coupled (SC) QLDPC codes, while Code 8 is a hypergraph product (HGP) code constructed from the hypergraph of a $[30, 70]$ quasi-cyclic LDPC code with itself.

of errors that still remains in uncorrectable under BP-OSD.

6.4 Numerical Results

We consider a depolarization channel where the probability of a physical qubit being correct is $1-p$, and the probability of an X, Y, or Z error is equal to $p/3$. Fig. 9 and Fig. 10 show the result of applying the decoder described in Section 6.2 to the Codes 1 through 7 specified in Section 6.1.

We conducted simulations in parallel on a high-performance computing cluster. The job indices served as seeds for the standard Mersenne Twister random number generator (mt19937 in C++), while Pauli errors were generated using the built-in discrete distribution function. The BP decoder estimates the error vector at the end of each iteration, checking the check node (CN) conditions, and halting when all CN conditions are satisfied or the maximum number of iterations reaches 500.

If, after 500 iterations, not all CN conditions are satisfied, the estimated errors undergo the SSF postprocessing step outlined in Section 6.3. Decoding is considered failed if the CN conditions remain unsatisfied or if they are satisfied but the output does not belong to the stabilizer group. Since SC-HGP codes are CSS codes, checking containment within the stabilizer group is straightforward after applying offline Gaussian elimination to reduce the X- and Z-components of the parity-check matrix to systematic form. We collect a sufficient number of errors that result in decoding failures in the BP-SSF decoder, and record the error rate as p_e^{SSF}.

We then pass only the aforementioned errors to the BP-OSD algorithm to further assess the number of unresolved decoding failures [58, 61]. The ratio of unresolved decoding failures to the total number of errors is denoted as α_e^{OSD}. Consequently, the actual logical error rate under BP-OSD is given by $p_e^{\mathrm{SSF}}\alpha_e^{\mathrm{OSD}}$. The BP-OSD software was obtained from [64], with $w = 2$ and the number of iterations also set to 500.

Fig. 9 shows FER plots for the $[[7300, 2500]]$ codes specified in Table 1, and Fig. 10 shows FER plots for the $[[5800, 1600]]$ codes specified in Table 2. Dashed lines represent codes (2 and 4) constructed from random partitioning matrices, solid lines represent codes optimized by AO, and the color of the line represents a choice of random/optimized code. In addition, we compare our constructed codes with the $[[7300, 2500]]$ and $[[5800, 1600]]$ hypergraph product (HGP) codes. These HGP codes are derived from two identical

[30, 80] and [30, 70] binary random LDPC codes, respectively. The parity-check matrices of these random LDPC codes are modified from array-based quasi-cyclic codes, lifted from all-one matrices of sizes 3×7 and 3×8, ensuring the elimination of cycles-4 and the minimization of cycles-6 in the lifted codes. The lifting matrices are specified in (90). The performance curves for these codes are shown in Fig. 9 and Fig. 10, where they are labeled as Code 8.

We make the following observations:

6.4.1 Reducing the number of short cycles leads to significant gains in the performance of SC-HGP codes

We attribute the gain of Code 1 over Code 2, and the gain of Code 3 over Code 4, to the reduction in the number of short cycles documented in Table 1 and Table 2.

Remark 8. *(Detrimental objects) There is more to improving finite-length performance than reducing the number of cycles. First, the influence of a cycle depends on its neighborhood, so some cycles are more detrimental than others. Second, objects such as absorbing sets and stopping sets may be more detrimental than cycles. We attribute the performance gains reported in this paper to the fact that short cycles are the basic building blocks of these more general detrimental objects. In future work we might learn error vectors from decoding failures, and expurgate them from the Tanner graph.*

6.4.2 High performance is possible with limited memory

Figs. 9 and 10 illustrate that increasing memory from $(1,1)$ / $(1,2)$ to $(2,2)$ / $(3,3)$ improves the FER by an order of magnitude. However, the gain associated with increasing memory from $(2,2)$ to $(3,3)$ is marginal (the FER curves cross in Fig. 9 and they overlap in Fig. 10). As the memory increases, the number of rigid cycles dominates the number of flexible cycles, hence reducing the number of flexible cycles has a marginal impact on performance.

Remark 9. *In Fig. 10, the FER curve of the memory* $(1,2)$ *code crosses that of the memory* $(1,1)$ *code, despite the fact that neither is close to the rigid cycle limit. It is rare that cycle optimization produces a code with small minimum distance, but that is the case here, since at least* $2/3$ *of the errors are convergent errors.*

6.4.3 Rigid cycle limit

We also observe that while GRADE does consistently lead to better random partitioning matrices than matrices sampled from uniform distributions (i.e., Code 4 always perform better than Code 2), the AO-optimized codes (Code 1 and 3) almost overlap in both figures, which is also consistent with the fact that their number of cycles are also close. The reason is still due to the rigid cycle limit.

Remark 10. *(Rigid Cycle Limit) Consider to what extent we are able to reduce the number of rigid cycles by optimizing the matrices* $\mathbf{A}$ *and* $\mathbf{B}$ *in the SC-HGP code construction. Suppose that the* i*-th VN is connected to* $d_i = d_i^X + d_i^Y + d_i^Z$ *CNs, where* d_i^X, d_i^Y, *and* d_i^Z *respectively denote the number of* X*-type,* Y*-type and* Z*-type connections. Since short cycles have the greatest impact on performance, we focus on cycles-4 to show how the distribution of degrees* d_i *limits the extent to which the number of rigid cycles can be reduced. Commutativity of the stabilizer group implies that the* i'*th VN is contained in at least* $(d_i^X d_i^Y + d_i^Y d_i^Z + d_i^Z d_i^X)$ *CNs. Hence the total number of cycles-4 is*

$$\frac{1}{2}\sum_{i=1}^{N}(d_i^X d_i^Y + d_i^Y d_i^Z + d_i^Z d_i^X), \tag{75}$$

where N *is the number of VNs. Observe that for CSS codes, (75) reduces to* $\frac{1}{2}\sum_{i=1}^{N} d_i^Z d_i^X$. *It is necessary to optimize degree distributions in order to reduce the number (and influence) of rigid cycles.*

6.4.4 Rigid cycles limit the extent to which GRADE and AO can improve performance

GRADE consistently provides random partitioning matrices that improve upon those sampled from a uniform distribution (for example, Code 4 is better than Code 2). Further optimization (AO) is less consequential when the number of cycles is close to the rigid cycle limit (for example, Codes 1 and 3 almost overlap). However, note that $\mathbf{A}$ and $\mathbf{B}$ we used here are pretty small

and the HGP base matrix is a sparse one. Higher memories might be needed for larger/denser base matrices.

6.4.5 Performance is not a simple function of minimum distances

Figs. 9 and 10 illustrate that SC-HGP codes with smaller minimum distances do not necessarily perform worse than those with larger minimum distances. The fact that Code 1 in Fig. 10, with a minimum distance upper bounded by 10, shows close and even slightly better performance compared to Code 3, which has a minimum distance upper bounded by 14, demonstrates that the minimum distance is not the only factor influencing QLDPC code performance. There are two reasons for this:

1. Even when using a maximum likelihood decoder, the error rate depends on the weight distribution of the codewords. Specifically, the multiplicities of the minimum weight codewords are significant, rather than just the minimum distance alone.

2. A codeword is also an absorbing set and contains a large number of concentrated cycles. Reducing cycles can strategically lower the number of small-weight codewords.

6.4.6 Comparisons with Prior Work

Consider the memory $(2, 2)$ and $(3, 3)$ codes featured in Figs. 9 and 10 with rates as high as 0.3425 (2500/7800) and 0.2759 (1600/5800) respectively. The rate 0.3425 codes have thresholds around 7.0%, and the rate 0.2759 codes have thresholds around 8.7%. The HGP codes appearing in [58] have much lower rates (less than 1/10) and inferior thresholds. We also compared SC-HGP codes with HGP codes of the same size, as demonstrated by the performance curves of Code 8 in each figure. It is important to note that we removed all cycles-4 and minimized the number of cycles-6 in the component matrices using a greedy algorithm to prevent potential performance degradation and ensure a fair comparison. We observe that the HGP codes exhibit inferior performance compared to the SC-HGP codes.

Remark 11. *(Generalized Bicycle (GB) Codes) The* $[[126, 28]]$ *GB code appearing in [58] is a short code with high rate and excellent performance (slightly worse than our rate* $[[7300, 2500]]$ *codes). It is possible that increasing the length will improve performance. Nevertheless, we have noted that GB codes can be categorized as special classes of 1D-SC-QLDPC codes with high memory. This example illustrates the importance of a well-chosen base matrix.*

7 Conclusion and Future Work

We have described toric codes as quantum counterparts of classical two-dimensional spatially coupled (2D-SC) codes, and introduced spatially-coupled (SC) QLDPC codes as a generalization. We have used the framework of two-dimensional convolution to represent the parity check matrix of a 2D-SC code as a polynomial in two indeterminates. Our representation leads to a simple algebraic condition that is necessary and sufficient for a 2D-SC code to be a stabilizer code. It also leads to formulae relating the number of short cycles of different lengths that arise from short cycles in either component code. We have described how to use these formulae to optimize SC-hypergraph product (HGP) codes by extending methods used to optimize classical 2D-SC codes. Our simulation results show that reducing the number of short cycles leads to significant gains in the performance of SC-HGP codes. We have shown that high performance is possible with limited memory.

Future work includes implementation of windowed decoders, modification of the decoder to efficiently address the bottleneck that results from rigid cycles and development of new SC-QLDPC codes that employ more general base.

Acknowledgement

We thank Dr. Victor Albert and Dr. Anthony Leverrier for making us aware of related literature. We thank Dr. Pavel Panteleev for specifying the relation between quasi-abelian lifted product codes and SC-HGP codes, and for bringing the more general lifted product codes defined over non-abelian groups to our attention. We thank Dr. Nithin Raveendran for providing us with estimations of minimum distances of the codes. We also thank Xinyu (Norah) Tan and Vahid Nourozi

for their suggestions on improving the presentation of the paper. This work was supported in part by the National Science Foundation through QLCI-CI: Institute for Robust Quantum Simulation, and through Grant CCF-2106213.

Data Availability

The optimization software can be found at [66].

References

[1] A. Robert Calderbank, Eric M. Rains, Peter W. Shor, and Neil J. A. Sloane. "Quantum error correction and orthogonal geometry". Physical Review Letters **78**, 405–409 (1997).

[2] A. Robert Calderbank, Eric M. Rains, Peter W. Shor, and Neil J. A. Sloane. "Quantum error correction via codes over GF(4)". IEEE Transactions on Information Theory **44**, 1369–1387 (1998).

[3] Daniel Gottesman. "Stabilizer codes and quantum error correction". PhD thesis. California Institute of Technology. (1997).

[4] A Yu Kitaev. "Quantum computations: algorithms and error correction". Russian Mathematical Surveys **52**, 1191 (1997).

[5] A Yu Kitaev. "Fault-tolerant quantum computation by anyons". Annals of physics **303**, 2–30 (2003).

[6] Sergey B Bravyi and A Yu Kitaev. "Quantum codes on a lattice with boundary" (1998). arXiv:9811052.

[7] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. "Topological quantum memory". Journal of Mathematical Physics **43**, 4452–4505 (2002).

[8] Michael H Freedman, David A Meyer, and Feng Luo. "Z2-systolic freedom and quantum codes". Chapter 12, pages 303–338. Chapman and Hall/CRC. (2002).

[9] Sergey Bravyi, David Poulin, and Barbara Terhal. "Tradeoffs for reliable quantum information storage in 2d systems". Physical review letters **104**, 050503 (2010).

[10] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. "Surface codes: Towards practical large-scale quantum computation". Physical Review A **86**, 032324 (2012).

[11] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter. "Surface code quantum computing by lattice surgery". New Journal of Physics **14**, 123011 (2012).

[12] Nicolas Delfosse and Gilles Zémor. "Quantum erasure-correcting codes and percolation on regular tilings of the hyperbolic plane". In 2010 IEEE Information Theory Workshop. Pages 1–5. IEEE (2010).

[13] Barbara M Terhal. "Quantum error correction for quantum memories". Reviews of Modern Physics **87**, 307 (2015).

[14] Nicolas Delfosse, Pavithran Iyer, and David Poulin. "Generalized surface codes and packing of logical qubits" (2016). arXiv:1606.07116.

[15] Nikolas P Breuckmann, Christophe Vuillot, Earl Campbell, Anirudh Krishna, and Barbara M Terhal. "Hyperbolic and semi-hyperbolic surface codes for quantum storage". Quantum Science and Technology **2**, 035007 (2017).

[16] Jean-Pierre Tillich and Gilles Zémor. "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength". IEEE Transactions on Information Theory **60**, 1193–1202 (2014).

[17] Anthony Leverrier, Simon Apers, and Christophe Vuillot. "Quantum XYZ product codes". Quantum **6**, 766 (2022).

[18] Shai Evra, Tali Kaufman, and Gilles Zémor. "Decodable quantum LDPC codes beyond the n distance barrier using high-dimensional expanders". SIAM Journal on ComputingPages 276–316 (2022).

[19] M. Sipser and D.A. Spielman. "Expander codes". IEEE Transactions on Information Theory **42**, 1710–1722 (1996).

[20] Gilles Zémor. "On cayley graphs, surface codes, and the limits of homological coding for quantum error correction". In Coding and Cryptology: Second International Workshop, IWCC 2009, Zhangjiajie, China, June 1-5, 2009. Proceedings 2. Pages 259–273. Springer (2009).

[21] Matthew B Hastings, Jeongwan Haah, and Ryan 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. Pages 1276–1288. (2021).

[22] Pavel Panteleev and Gleb Kalachev. "Quantum LDPC codes with almost linear minimum distance". IEEE Transactions on Information Theory **68**, 213–229 (2022).

[23] Nikolas P Breuckmann and Jens N Eberhardt. "Balanced product quantum codes". IEEE Transactions on Information Theory **67**, 6653–6674 (2021).

[24] Pavel Panteleev and Gleb Kalachev. "Asymptotically good quantum and locally testable classical LDPC codes". In Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing. Pages 375—-388. (2022).

[25] A. Leverrier and G. Zemor. "Quantum tanner codes". In 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS). Pages 872–883. (2022).

[26] Robert Gallager. "Low-density parity-check codes". IRE Transactions on information theory **8**, 21–28 (1962).

[27] Thomas J Richardson, Mohammad Amin Shokrollahi, and Rüdiger L Urbanke. "Design of capacity-approaching irregular low-density parity-check codes". IEEE transactions on information theory **47**, 619–637 (2001).

[28] Thomas J Richardson and Rüdiger L Urbanke. "The capacity of low-density parity-check codes under message-passing decoding". IEEE Transactions on information theory **47**, 599–618 (2001).

[29] David Poulin and Yeojin Chung. "On the iterative decoding of sparse quantum codes". Quantum Information & Computation **8**, 987–1000 (2008).

[30] Daniel A Lidar and Todd A Brun. "Quantum error correction". Cambridge university press. (2013).

[31] D. G. M. Mitchell, M. Lentmaier, and D. J. Costello. "Spatially coupled LDPC codes constructed from protographs". IEEE Trans. Information Theory **61**, 4866–4889 (2015).

[32] David G. M. Mitchell and Eirik Rosnes. "Edge spreading design of high rate array-based SC-LDPC codes". In 2017 IEEE International Symposium on Information Theory (ISIT). Pages 2940–2944. (2017).

[33] Allison Beemer, Salman Habib, Christine A. Kelley, and Joerg Kliewer. "A generalized algebraic approach to optimizing SC-LDPC codes". In 2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton). Pages 672–679. (2017).

[34] Allison Beemer. "Design and analysis of graph-based codes using algebraic lifts and decoding networks". PhD thesis. The University of Nebraska-Lincoln. (2018). url: https://digitalcommons.unl.edu/mathstudent/87/.

[35] Ahmed Hareedy, Chinmayi Lanka, and Lara Dolecek. "A general non-binary LDPC code optimization framework suitable for dense flash memory and magnetic storage". IEEE Journal on Selected Areas in Communications **34**, 2402–2415 (2016).

[36] A. Hareedy, H. Esfahanizadeh, and L. Dolecek. "High performance non-binary spatially-coupled codes for flash memories". In 2017 IEEE Information Theory Workshop (ITW). Pages 229–233. (2017).

[37] Ahmed Hareedy, Ruiyi Wu, and Lara Dolecek. "A channel-aware combinatorial approach to design high performance spatially-coupled codes". IEEE Trans. Information Theory **66**, 4834–4852 (2020).

[38] Siyi Yang, Ahmed Hareedy, Robert Calderbank, and Lara Dolecek. "Breaking the computational bottleneck: Probabilistic optimization of high-memory spatially-coupled codes". IEEE Transactions on Information Theory **69**, 886–909 (2023).

[39] Manabu Hagiwara, Kenta Kasai, Hideki Imai, and Kohichi Sakaniwa. "Spatially coupled quasi-cyclic quantum LDPC codes". In 2011 IEEE International Symposium on Information Theory Proceedings. Pages 638–642. (2011).

[40] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov. "Iterative decoding threshold analysis for LDPC convolutional codes". IEEE Trans. Information Theory **56**, 5274–5289 (2010).

[41] Aravind R Iyengar, Marco Papaleo, Paul H Siegel, Jack Keil Wolf, Alessandro Vanelli-Coralli, and Giovanni E Corazza. "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels". IEEE Transactions on Information Theory **58**, 2303–2320 (2011).

[42] Aravind R Iyengar, Paul H Siegel, Rüdiger L Urbanke, and Jack Keil Wolf. "Windowed decoding of spatially coupled codes". IEEE

transactions on Information Theory **59**, 2277–2292 (2012).

[43] Najeeb Ul Hassan, Ali E Pusane, Michael Lentmaier, Gerhard P Fettweis, and Daniel J Costello. "Non-uniform window decoding schedules for spatially coupled LDPC codes". IEEE Transactions on Communications **65**, 501–510 (2016).

[44] Lai Wei, David GM Mitchell, Thomas E Fuja, and Daniel J Costello. "Design of spatially coupled LDPC codes over GF(q) for windowed decoding". IEEE Transactions on Information Theory **62**, 4781–4800 (2016).

[45] Min Zhu, David GM Mitchell, Michael Lentmaier, Daniel J Costello, and Baoming Bai. "Braided convolutional codes with sliding window decoding". IEEE Transactions on Communications **65**, 3645–3658 (2017).

[46] Kevin Klaiber, Sebastian Cammerer, Laurent Schmalen, and Stephan ten Brink. "Avoiding burst-like error patterns in windowed decoding of spatially coupled LDPC codes". In 2018 IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC). Pages 1–5. IEEE (2018).

[47] Homa Esfahanizadeh, Lev Tauz, and Lara Dolecek. "Multi-dimensional spatially-coupled code design: Enhancing the cycle properties". IEEE Transactions on Communications **68**, 2653–2666 (2020).

[48] Lev Tauz, Homa Esfahanizadeh, and Lara Dolecek. "Non-uniform windowed decoding for multi-dimensional spatially-coupled LDPC codes". In 2020 IEEE International Symposium on Information Theory (ISIT). Pages 485–490. IEEE (2020).

[49] Eshed Ram and Yuval Cassuto. "On the decoding performance of spatially coupled LDPC codes with sub-block access". IEEE Transactions on Information Theory **68**, 3700–3718 (2022).

[50] Jeongwan Haah. "Commuting pauli hamiltonians as maps between free modules". Communications in Mathematical Physics **324**, 351–399 (2013).

[51] Jeongwan Haah. "Algebraic methods for quantum codes on lattices". Revista colombiana de matematicas **50**, 299–349 (2016).

[52] Harold Ollivier and Jean-Pierre Tillich. "Description of a quantum convolutional code". Physical Review Letters **91**, 177902 (2003).

[53] S. Kudekar, T. J. Richardson, and R. L. Urbanke. "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC". IEEE Trans. Information Theory **57**, 803–834 (2011).

[54] S. Kumar, A. J. Young, N. Macris, and H. D. Pfister. "Threshold saturation for spatially coupled LDPC and LDGM codes on BMS channels". IEEE Trans. Information Theory **60**, 7389–7415 (2014).

[55] Sergey Bravyi, Andrew W Cross, Jay M Gambetta, Dmitri Maslov, Patrick Rall, and Theodore J Yoder. "High-threshold and low-overhead fault-tolerant quantum memory". Nature **627**, 778–782 (2024).

[56] Qian Xu, J Pablo Bonilla Ataides, Christopher A Pattison, Nithin Raveendran, Dolev Bluvstein, Jonathan Wurtz, Bane Vasić, Mikhail D Lukin, Liang Jiang, and Hengyun Zhou. "Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays". Nature PhysicsPages 1–7 (2024).

[57] Hanrui Wang, Pengyu Liu, Daniel Bochen Tan, Yilian Liu, Jiaqi Gu, David Z Pan, Jason Cong, Umut A Acar, and Song Han. "Atomique: A quantum compiler for reconfigurable neutral atom arrays". In 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA). Pages 293–309. (2024).

[58] Pavel Panteleev and Gleb Kalachev. "Degenerate quantum LDPC codes with good finite length performance". Quantum **5**, 585 (2021).

[59] M. Battaglioni, A. Tasdighi, G. Cancellieri, F. Chiaraluce, and M. Baldi. "Design and analysis of time-invariant SC-LDPC convolutional codes with small constraint length". IEEE Trans. Communications **66**, 918–931 (2018).

[60] M. P. C. Fossorier. "Quasicyclic low-density parity-check codes from circulant permutation matrices". IEEE Trans. Information Theory **50**, 1788–1793 (2004).

[61] Joschka Roffe, David R. White, Simon Burton, and Earl Campbell. "Decoding across the quantum low-density parity-check code landscape". Physical Review Research**2** (2020).

[62] Julien Du Crest, Mehdi Mhalla, and

Valentin Savin. "Stabilizer inactivation for message-passing decoding of quantum LDPC codes". In 2022 IEEE Information Theory Workshop (ITW). Pages 488–493. IEEE (2022).

[63] Hanwen Yao, Waleed Abu Laban, Christian Häger, Alexandre Graell i Amat, and Henry D Pfister. "Belief propagation decoding of quantum ldpc codes with guided decimation". In 2024 IEEE International Symposium on Information Theory (ISIT). Pages 2478–2483. IEEE (2024).

[64] Joschka Roffe (2022). code: https://pypi.org/project/ldpc/.

[65] Antoine Grospellier, Lucien Grouès, Anirudh Krishna, and Anthony Leverrier. "Combining hard and soft decoders for hypergraph product codes". Quantum **5**, 432 (2021).

[66] Siyi Yang (2025). code: SiyiPriscaYang/Spatially-Coupled-QLDPC-Codes.

A Proof of Example 4

Proof.

$$\langle \mathbf{A}, \mathbf{D} \rangle = \left\langle \begin{bmatrix} X & I \\ I & I \end{bmatrix}, \begin{bmatrix} I & I \\ I & Z \end{bmatrix} \right\rangle = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix};$$
$$\langle \mathbf{B}, \mathbf{C} \rangle = \left\langle \begin{bmatrix} X & X \\ Z & I \end{bmatrix}, \begin{bmatrix} I & X \\ Z & Z \end{bmatrix} \right\rangle = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix};$$

$$\begin{aligned} \langle \mathbf{A}, \mathbf{C} \rangle + \langle \mathbf{B}, \mathbf{D} \rangle &= \left\langle \begin{bmatrix} X & I \\ I & I \end{bmatrix}, \begin{bmatrix} I & X \\ Z & Z \end{bmatrix} \right\rangle + \left\langle \begin{bmatrix} X & X \\ Z & I \end{bmatrix}, \begin{bmatrix} I & I \\ I & Z \end{bmatrix} \right\rangle \\ &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}; \end{aligned}$$

$$\begin{aligned} \langle \mathbf{A}, \mathbf{B} \rangle + \langle \mathbf{C}, \mathbf{D} \rangle &= \left\langle \begin{bmatrix} X & I \\ I & I \end{bmatrix}, \begin{bmatrix} X & X \\ Z & I \end{bmatrix} \right\rangle + \left\langle \begin{bmatrix} I & X \\ Z & Z \end{bmatrix}, \begin{bmatrix} I & I \\ I & Z \end{bmatrix} \right\rangle \\ &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}; \end{aligned}$$

$$\begin{aligned} &\langle \mathbf{A}, \mathbf{A} \rangle + \langle \mathbf{B}, \mathbf{B} \rangle + \langle \mathbf{C}, \mathbf{C} \rangle + \langle \mathbf{D}, \mathbf{D} \rangle = \langle \mathbf{B}, \mathbf{B} \rangle + \langle \mathbf{C}, \mathbf{C} \rangle \\ =& \left\langle \begin{bmatrix} X & X \\ Z & I \end{bmatrix}, \begin{bmatrix} X & X \\ Z & I \end{bmatrix} \right\rangle + \left\langle \begin{bmatrix} I & X \\ Z & Z \end{bmatrix}, \begin{bmatrix} I & X \\ Z & Z \end{bmatrix} \right\rangle \\ =& \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \end{aligned}$$

□

B Constructions

$$\mathbf{P}_a = \begin{bmatrix} 2 & 1 & 3 & 8 & 4 & 8 & 3 & 3 \\ 2 & 0 & 6 & 1 & 6 & 6 & 2 & 5 \\ 6 & 8 & 2 & 0 & 4 & 1 & 5 & 7 \end{bmatrix}, \ \mathbf{P}_b = \begin{bmatrix} 2 & 2 & 6 & 5 & 6 & 3 & 1 & 0 \\ 7 & 6 & 2 & 0 & 0 & 4 & 3 & 8 \\ 6 & 0 & 0 & 7 & 5 & 8 & 5 & 3 \end{bmatrix}. \tag{76}$$

$$\mathbf{P}_a = \begin{bmatrix} 2 & 3 & 5 & 3 & 4 & 0 & 7 & 0 \\ 3 & 6 & 1 & 6 & 6 & 0 & 2 & 8 \\ 4 & 8 & 2 & 7 & 8 & 5 & 5 & 1 \end{bmatrix}, \ \mathbf{P}_b = \begin{bmatrix} 3 & 3 & 6 & 6 & 8 & 3 & 2 & 5 \\ 1 & 4 & 2 & 0 & 0 & 4 & 7 & 8 \\ 5 & 1 & 0 & 7 & 5 & 8 & 6 & 2 \end{bmatrix}. \tag{77}$$

$$\mathbf{P}_a = \begin{bmatrix} 8 & 1 & 2 & 2 & 6 & 8 & 6 & 7 \\ 0 & 2 & 4 & 5 & 3 & 7 & 1 & 5 \\ 8 & 6 & 8 & 6 & 2 & 2 & 1 & 0 \end{bmatrix}, \ \mathbf{P}_b = \begin{bmatrix} 8 & 8 & 2 & 6 & 6 & 0 & 3 & 0 \\ 7 & 6 & 6 & 2 & 1 & 8 & 2 & 5 \\ 3 & 1 & 8 & 5 & 2 & 3 & 8 & 7 \end{bmatrix}. \tag{78}$$

$$\mathbf{P}_a = \begin{bmatrix} 0 & 1 & 7 & 2 & 5 & 8 & 6 & 8 \\ 0 & 2 & 4 & 8 & 3 & 6 & 0 & 5 \\ 3 & 6 & 8 & 6 & 2 & 2 & 1 & 0 \end{bmatrix}, \ \mathbf{P}_b = \begin{bmatrix} 0 & 5 & 1 & 6 & 3 & 3 & 2 & 0 \\ 6 & 6 & 8 & 0 & 1 & 8 & 2 & 5 \\ 0 & 4 & 8 & 8 & 2 & 6 & 2 & 7 \end{bmatrix}. \tag{79}$$

$$\mathbf{P}_a = \begin{bmatrix} 1 & 3 & 0 & 2 & 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 1 & 1 & 2 & 2 & 2 \\ 2 & 0 & 0 & 0 & 2 & 3 & 0 & 1 \end{bmatrix}, \ \mathbf{P}_b = \begin{bmatrix} 1 & 2 & 1 & 3 & 0 & 2 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 2 & 2 \\ 2 & 0 & 3 & 0 & 3 & 1 & 0 & 1 \end{bmatrix}. \tag{80}$$

$$\mathbf{P}_a = \begin{bmatrix} 2 & 4 & 3 & 1 & 2 & 3 & 2 & 3 \\ 3 & 3 & 0 & 5 & 4 & 5 & 0 & 1 \\ 2 & 0 & 2 & 3 & 0 & 0 & 3 & 5 \end{bmatrix}, \ \mathbf{P}_b = \begin{bmatrix} 5 & 1 & 3 & 0 & 2 & 0 & 3 & 5 \\ 0 & 0 & 1 & 5 & 3 & 2 & 2 & 3 \\ 1 & 5 & 5 & 1 & 3 & 5 & 3 & 2 \end{bmatrix}. \tag{81}$$

$$\mathbf{P}_a = \begin{bmatrix} 14 & 12 & 3 & 3 & 9 & 11 & 0 & 2 \\ 15 & 6 & 8 & 12 & 1 & 14 & 0 & 7 \\ 4 & 0 & 15 & 10 & 15 & 3 & 11 & 12 \end{bmatrix}, \mathbf{P}_b = \begin{bmatrix} 14 & 9 & 5 & 5 & 12 & 0 & 7 & 3 \\ 12 & 3 & 6 & 2 & 1 & 15 & 0 & 8 \\ 0 & 1 & 15 & 14 & 7 & 2 & 4 & 15 \end{bmatrix}. \quad (82)$$

$$\mathbf{P}_a = \begin{bmatrix} 2 & 5 & 6 & 8 & 0 & 6 & 5 \\ 6 & 7 & 2 & 6 & 5 & 0 & 6 \\ 7 & 0 & 2 & 1 & 7 & 8 & 5 \end{bmatrix}, \mathbf{P}_b = \begin{bmatrix} 2 & 1 & 3 & 2 & 0 & 7 & 6 \\ 1 & 8 & 8 & 6 & 8 & 0 & 5 \\ 6 & 6 & 2 & 3 & 0 & 5 & 1 \end{bmatrix}. \quad (83)$$

$$\mathbf{P}_a = \begin{bmatrix} 8 & 3 & 1 & 7 & 7 & 2 & 1 \\ 2 & 4 & 8 & 1 & 6 & 0 & 4 \\ 7 & 5 & 5 & 4 & 3 & 6 & 0 \end{bmatrix}, \mathbf{P}_b = \begin{bmatrix} 5 & 0 & 4 & 8 & 1 & 6 & 1 \\ 6 & 8 & 1 & 4 & 5 & 4 & 3 \\ 3 & 7 & 2 & 7 & 7 & 0 & 2 \end{bmatrix}. \quad (84)$$

$$\mathbf{P}_a = \begin{bmatrix} 6 & 2 & 5 & 1 & 6 & 0 & 8 \\ 4 & 3 & 8 & 0 & 1 & 2 & 2 \\ 2 & 6 & 0 & 8 & 1 & 4 & 1 \end{bmatrix}, \mathbf{P}_b = \begin{bmatrix} 6 & 1 & 0 & 6 & 7 & 5 & 2 \\ 8 & 3 & 0 & 2 & 2 & 4 & 6 \\ 1 & 8 & 3 & 6 & 8 & 6 & 0 \end{bmatrix}. \quad (85)$$

$$\mathbf{P}_a = \begin{bmatrix} 6 & 2 & 3 & 0 & 6 & 7 & 0 \\ 2 & 6 & 8 & 5 & 6 & 8 & 7 \\ 2 & 0 & 0 & 8 & 1 & 4 & 1 \end{bmatrix}, \mathbf{P}_b = \begin{bmatrix} 2 & 5 & 7 & 6 & 7 & 2 & 2 \\ 8 & 0 & 0 & 1 & 6 & 4 & 6 \\ 1 & 8 & 3 & 6 & 8 & 0 & 0 \end{bmatrix}. \quad (86)$$

$$\mathbf{P}_a = \begin{bmatrix} 0 & 3 & 1 & 3 & 2 & 1 & 2 \\ 1 & 0 & 2 & 3 & 1 & 3 & 0 \\ 2 & 3 & 0 & 0 & 3 & 2 & 1 \end{bmatrix}, \mathbf{P}_b = \begin{bmatrix} 1 & 3 & 3 & 0 & 0 & 2 & 1 \\ 2 & 1 & 0 & 2 & 3 & 3 & 1 \\ 3 & 0 & 2 & 1 & 3 & 1 & 2 \end{bmatrix}. \quad (87)$$

$$\mathbf{P}_a = \begin{bmatrix} 0 & 2 & 0 & 3 & 2 & 5 & 3 \\ 5 & 0 & 3 & 4 & 1 & 2 & 5 \\ 0 & 4 & 2 & 2 & 3 & 3 & 1 \end{bmatrix}, \mathbf{P}_b = \begin{bmatrix} 2 & 4 & 5 & 2 & 0 & 5 & 0 \\ 3 & 2 & 2 & 4 & 2 & 3 & 5 \\ 4 & 3 & 0 & 0 & 5 & 2 & 4 \end{bmatrix}. \quad (88)$$

$$\mathbf{P}_a = \begin{bmatrix} 4 & 10 & 3 & 12 & 13 & 1 & 5 \\ 7 & 8 & 8 & 11 & 7 & 11 & 3 \\ 1 & 6 & 12 & 3 & 2 & 0 & 15 \end{bmatrix}, \mathbf{P}_b = \begin{bmatrix} 4 & 11 & 13 & 12 & 7 & 3 & 12 \\ 3 & 13 & 15 & 10 & 6 & 1 & 0 \\ 4 & 1 & 5 & 7 & 14 & 15 & 3 \end{bmatrix}. \quad (89)$$

$$\mathbf{L}_1 = \begin{bmatrix} 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 2 & 4 & 6 & 8 & 1 & 3 \end{bmatrix}, \mathbf{L}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 2 & 4 & 6 & 9 & 1 & 3 & 5 \end{bmatrix}. \quad (90)$$