On the Left-Right Lifted Product Quantum LDPC Code

by

Boqing Zhang

Department of Electrical and Computer Engineering
Duke University

Defense Date: March 27, 2024

Approved:

Henry Pfister, Supervisor

Robert Calderbank

Kenneth R Brown

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University
2024

ABSTRACT

On the Left-Right Lifted Product Quantum LDPC Code

by

Boqing Zhang

Department of Electrical and Computer Engineering
Duke University

Defense Date: March 27, 2024

Approved:

Henry Pfister, Supervisor

Robert Calderbank

Kenneth R Brown

An abstract of a thesis submitted in partial fulfillment of the requirements for
the degree of Master of Science in the Department of Electrical and Computer Engineering
in the Graduate School of Duke University
2024

Abstract

Recent advances in both theory and experiment have identified quantum Low-Density Parity-Check (LDPC) codes as a prime candidate for fault-tolerant quantum computation in the near future. This thesis addresses various challenges related to the construction and decoding of quantum LDPC (qLDPC) codes. A major contribution of this research is the creation of a unified family of qLDPC codes called left-right lifted product codes. The proposed approach not only makes the construction process simpler but also brings different qLDPC constructions together into a single framework that greatly improves both understanding and practical use.

This thesis aims to highlight the practical and theoretical advantages of the left-right lifted product construction. As part of this, we present the finite-length distance analysis for the general lifted product code. Moreover, we also extend the finite-length result to asymptotic lengths and provide a conjecture for a new kind of asymptotically good quantum LDPC code that holds promise for more feasible and practical applications.

From a practical standpoint, this work includes the first detailed cycle analysis, specifically focusing on dominant 8-cycles, which significantly impact decoding performance. Our findings indicate that the lifted product construction substantially reduces the number of short 8-cycles compared to the hypergraph product code. Additionally, we establish an upper bound for the girth of the left-right lifted product code, demonstrating that, unlike classical QC-LDPC codes, the presence of 8-cycles is unavoidable, regardless of how the group algebra elements are selected in the base matrices.

Lastly, leveraging newly developed mathematical tools in spectral hypergraph theory, we offer a computable lower bound for the left-expansion property of the target factor graph. We believe that this contribution can be particularly valuable for code design, as the left-expansion property was previously difficult to quantify.

Contents

List of Tables

List of Figures

Acknowledgements

This work was completed in collaboration with my thesis advisor Dr. Henry Pfister of Duke University. First, I would like to thank two of my committee members, Dr. Robert Calderbank and Dr. Kenneth R. Brown. It was their two amazing courses - Introduction to Algebraic Structure and Quantum Error Correction - that brought me to the field of quantum error correction and inspired my research into quantum LDPC codes. Additionally, I would like to thank Dr. Hanwen Yao and Dr. Siyi Yang for being my research mentors and for all the meaningful discussions we have had. Beyond research, I would particularly like to express my gratitude to my advisor Dr. Henry Pfister, for helping me navigate through several challenges and for his support in overcoming personal obstacles. I would like to express my sincerest gratitude to all of them for their guidance and support.

1. Introduction

This chapter aims to provide a concise history and background of Quantum Error Correction, focusing primarily on Quantum Low-Density Parity-Check (LDPC) constructions and various decoding algorithms associated with Quantum LDPC codes in the first section. The second section identifies several prevailing challenges within the field of Quantum LDPC codes and decoding, outlining the thesis's objectives designed to address these issues. The specific contributions of this work are detailed in the third section, summarizing the concrete efforts undertaken to achieve each objective.

1.1 History and Background

Quantum Error Correction (QEC) is a field that addresses the challenge of preserving the integrity of quantum information in the presence of errors due to decoherence and other quantum noise. In QEC, stabilizer codes [1] represent a foundational class of quantum error-correcting codes. These codes are based on the concept of stabilizers, which are operators that "stabilize" a subspace of the quantum system's Hilbert space, ensuring that all states within this subspace are eigenstates of the operators with the same eigenvalue.

CSS (Calderbank-Shor-Steane) construction:

Within the family of stabilizer codes, the CSS (Calderbank-Shor-Steane) codes [2] are constructed from two classical error correcting codes with parity check matrices H_x and H_z, respectively, one for correcting bit-flip errors (X errors) and the other for correcting phase-flip errors (Z errors). Consequently, the CSS construction naturally separates the correction of bit-flip and phase-flip errors, streamlining the error correction process. This separation is achievable because the codes are designed so that the stabilizers for correcting bit-flips (X-stabilizers) commute with those for correcting phase-flips (Z-stabilizers). Mapping to the binary representation, it is equivalent to the equation $H_x \cdot H_z^T = \mathbf{0}$. The number of encoded qubits of CSS code is $k_x + k_z - N$ where N is the number of physical qubits, and k_x, k_z represents the number of encoded qubits for the two classical codes C_x, C_z, respectively.

Additionally, the minimum distance of the stabilizer codes is determined by the minimum weight of the logical operators. For a CSS code, the logical operators can be distinctively categorized as Z and X logical operators, allowing the minimum weight of the CSS code to be defined as the lesser of the Z and X distances, d_z and d_x, respectively. Specifically, the distance of a CSS code, d, is expressed as follows,

$$d_z = \min \mathrm{wt}(L_z), \qquad d_x = \min \mathrm{wt}(L_x), \qquad d = \min(d_z, d_x) \tag{1.1}$$

where $L_z \in Ker(H_x) \backslash Im(H_z)$ and $L_x \in Ker(H_z) \backslash Im(H_x)$. From (1.1), it is apparent that the Z and X distances, d_z and d_x, can exceed the minimum Hamming distances of the classical codes associated with the parity check matrices H_z and H_x. This outcome stems from a unique characteristic of quantum error-correcting codes known as *degeneracy* [1], a property not found in classical coding theory. It results from the fact that error patterns equal to a stabilizer do not change the encoded quantum state and do not need to be corrected.

Quantum Low-Density Parity-Check (LDPC) codes:

Quantum LDPC codes, a subclass of quantum stabilizer codes, originate from their classical analogs and are distinguished by their sparse parity check matrices. LDPC codes [3], both classical and quantum, are characterized by their parity check matrices with a notably low number of non-zero elements - typically, the number of "1"s in each column and row is capped by a constant that does not change with the code size. In the quantum setting, the defining sparsity of quantum LDPC codes enables efficient syndrome measurement. Moreover, the ability to correct many errors combined with a high encoding rate positions high-rate quantum LDPC codes as a strong candidate for fault-tolerant quantum computation, especially for near-term quantum technologies.

The development of quantum LDPC codes has been a significant area of research within quantum error correction. The history began with the renowned Toric code [4], arguably the first quantum LDPC code known for its large error threshold [5]. This code, which has distance scales $O(\sqrt{N})$ and a vanishing rate, received its name from the fact that it is

obtained from the cellulation of the torus. In 2002, Freedman, Meyer, and Luo opted for a more complex manifold than the torus to enhance the distance by a factor $\log^{1/4}(n)$ [6]. Since these quantum LDPC codes are derived from the cellulation of manifolds that can be embedded in a high-dimensional space, they are categorized as topological quantum error correcting codes.

Topological quantum error correcting codes, however, typically have a vanishing encoding rate. In 2009, inspired by the topological product from the Toric code, Tillich and Zemor introduced the *Hypergraph product codes*, which achieves a constant rate, though the distance remains bounded by $O(\sqrt{N})$ [7]. The year 2020 saw major breakthroughs in high-rate quantum LDPC codes. Evra, Kaufmanm, Zemor, and Tessler extend the code distance to $O(\sqrt{N}\log N)$ and $O(\sqrt{N}\log^c N)$, respectively [8][9]. In September of the same year, the *Fiber Bundle codes* [10], presented by Hastings, Haah, and O'Donnell, was the first to break the $O(\sqrt{N})$ boundary, achieving a distance proportional to $n^{0.6}/polylog(n)$. Three months later, Breuckmann and Eberhardt published *Balanced product codes*, offering an explicit construction based on the expander graph to achieve reach similar distance [11]. In December 2020, Panteleev and Kalachev invented the *Lifted product codes* which almost achieves the linear distance $d \propto N/\log(N)$ but with a slightly lower encoding rate $k = \log N$ [12]. Finally, in 2021, Panteleev and Kalachev proved the existence of the asymptotically good quantum LDPC codes through the high-dimensional Cayley expander construction [13]. This construction, combined with the local tensor codes featuring the *Robust Testability* property, has been shown to offer an asymptotically constant encoding rate and distance with high probability [14].

1.2 Challenges and Objectives

Building on the background and history provided, the field of quantum LDPC construction has evolved significantly, making numerous breakthroughs in recent years. Despite these advances, several challenges remain in the construction and decoding of quantum LDPC codes. This section outlines four major problems related to quantum LDPC code

construction, along with some objectives that this thesis aims to achieve.

Challenge 1 (Unifying framework):

The literature on high-rate quantum LDPC (qLDPC) codes presents plenty of seemingly distinct constructions including Hypergraph product construction [7], Balanced product [11], Lifted product [12] as well as the left-right Cayley expander constructions [13] and so on. As can be seen, most of the codes listed above exhibit some product structures. The variety of products in the field adds significant complexity and confusion regarding the product definitions, which will be demonstrated as equivalent in Chapter 2. Additionally, the languages and terminologies used across different constructions vary greatly. Some approaches are described using terms based on rather abstract mathematical concepts such as Sheaves, Fiber Bundles, and Homological Chain Complexes, further complicating accessibility to these code constructions.

Objective 1: Provide a unifying framework and constructions for existing high-rate qLDPC codes using fewer abstract mathematical concepts.

Challenge 2 (Finite-length distance):

In the context of finite-length scenarios, recent literature has begun to discuss finite-length distances for certain specific cases of the Lifted product codes, such as the two-block codes [17]. The code in [17] can be viewed as the simplest form of Lifted product construction, given that the base matrix is just a scalar group algebra element (as discussed in Chapter 2). Despite discussions on finite-length distances for special cases of the Lifted product code, there's an absence of minimum distance analysis and the discussion of the logical codewords for the general lifted product constructions. Furthermore, major advances in code distance have typically occurred in the asymptotic context as discussed in the above section. In such contexts, distance analyses predominantly rely on the quantum Tanner code and the high-dimensional Cayley expander. However, like its classical counter-

part - the expander code - the quantum Tanner construction has limited practical utility, as discussed in Chapter 2. Therefore, the asymptotically good quantum LDPC code based on high-dimensional Cayley expanders, while theoretically significant, may not be as relevant for real-world applications.

Objective 2: Discuss the finite-length minimum distance and the logical codeword structure for the most general left-right Lifted product construction defined in Chapter 2. Additionally, the thesis seeks to propose a simpler approach for constructing asymptotically good quantum LDPC codes that are more likely to be of practical use, without the use of the Cayley expander and the local codes.

Challenge 3 (Cycle analysis):

In applications, designing a code with good practical decoding performance is crucial. State-of-the-art decoding algorithms for high-rate quantum LDPC codes are typically based on the popular message-passing decoder called Belief Propagation [16][19] which has been very successful in classical coding theory and communication. However, the cycle analysis, a critical aspect of the efficacy of the Belief Propagation algorithm, has not received due attention in the context of the lifted product construction.

Objective 3: Carry out comprehensive cycle analysis for the left-right Lifted product code, in both finite lengths and asymptotic scenarios, with a particular focus on the dominant 8-cycles.

Challenge 4 (Computable bound for factor graph expansion):

Expansion is a crucial property that significantly influences many theoretical and practical aspects of a code, as discussed in Chapter 3. In coding theory, the term "expansion" refers to the so-called *left expansion* or *small set expansion* defined in Chapter 3. The best possible computational method to lower bound the left-expansion parameter α (defined in

Section 3.4) is through the graph expansion represented by the Cheeger constant. However, computing the exact Cheeger constant, which quantifies the expansion property, is not feasible [20]. The primary method for estimating the Cheeger constant involves tools from spectral graph theory. For codes, it's important to note that the graphs of interest are the category of factor graphs which is equivalent to hypergraphs. This categorization means that the conventional spectral bounds used for the Cheeger constant are not applicable, posing a considerable challenge for code design and engineering.

Objectives 4: Develop a computable bound for the left expansion property of the factor graph based on the recently developed mathematical tool for hypergraph Cheeger inequality. Perform a rigorous analysis to demonstrate how the Cartesian product of factor graphs (defined in Chapter 2) affects the original left expansion properties.

1.3 Main Contributions

This section provides a detailed overview of the work we have done to address the above challenges. It also highlights the major observations and results that have been achieved in this thesis to fulfill the objectives outlined in Section 1.2.

Contribution 1 (Unifying framework):

In Section 2.1, this thesis introduces a unifying framework aimed at simplifying the complexity inherent in high-rate quantum LDPC code constructions. This gives the most general Lifted product code construction, called left-right lifted product code, that encompasses almost all existing high-rate quantum LDPC codes, such as the Hypergraph Product [7], Lifted Product [12], Balanced Product[11], and the Left-Right Cayley Complex [13][14][15], among others. The left-right lifted product construction is essentially built on top of the lifted product code developed in [12] combined with the *left-right* multiplication for non-Abelian group lifting, which is also used in the balanced product and left-right

Cayley complex constructions [11][14].

This construction is notable not only for its comprehensiveness but also for its practicality in terms of implementation and engineering. This is because for the balanced product construction [11], while it also supports the non-Abelian group symmetry reduction, it is in general infeasible to find those symmetries from product complex in practical implementation. Moreover, in Section 2.2, it can be easily shown that the left-right lifted product is equivalent to the balanced product construction under group lifting structures. Although some people in the community might have already realized the equivalence, we also give a concrete commutative diagram to show the exact connection between these two constructions which is particularly useful for the intuitions and proofs of the theorems presented in Chapter 3.

Contribution 2 (Finite-length distance):

In this thesis, we discuss the finite-length distance for the general left-right lifted product code. As a result, given two parity check matrices with some lifting structure, it is shown that the left-right lifted product could retain both the logical codewords structure and the finite-length distance characteristics akin to those of the hypergraph product construction if it has a special stair-case structure or the group algebra element is carefully designed, yet it requires significantly fewer physical qubits. Their consistency of logical codewords and finite-length distances is fundamentally rooted in both constructions exhibiting the same factor graph product structure (defined in Chapter 2). This finding underscores the advantages in the finite-length distance of the left-right lifted product compared to hypergraph product constructions.

Contribution 3 (Conjecture new asymptotically good quantum LDPC codes):

Building on the result of finite-length distance in Section 3.1, we also consider asymptotic distance properties. If there is a classical "good" code whose distance grows linearly with the lifting size l, rather than with the number of columns and rows $\bar{n}, \bar{m}$ of the base

matrix $\bar{H}$, then it is logically proved that two classical good LDPC codes with lifting structure, the left-right Lifted product will yield quantum LDPC codes with constant rate and distance relative to the number of physical qubits. Towards the end of Section 3.2, after recognizing that the classical expander graph represents a particular instance of such good classical codes, we then conjecture the existence of classical lifted codes based on a non-Abelian group G capable of achieving the required good rate and distance.

Contribution 4 (Cycle analysis):

In Section 3.3, we present a cycle analysis for the left-right lifted product code, with a special emphasis on the dominant 8-cycles caused by the factor graph product structure. We compare the number of dominant 8-cycles in the lifted product code and the number of 8-cycles in the hypergraph product construction, both in finite-length and asymptotic contexts. Our findings indicate that, when utilizing two identical classical codes with lifting structures, the lifted product construction notably decreases the number of short 8-cycles in comparison to the hypergraph product approach.

Contribution 5 (Computable lower bound for the factor graph expansion):

Building on the recent advances concerning the generalized Cheeger constant for uniform hypergraphs [21], we explore the concept of "left expansion" in Section 3.4. This term refers to the type of expansion beneficial for many code properties as well as the efficient decoding algorithms [22]. We connect this concept with the generalized spectral Cheeger inequality tailored for uniform hypergraphs, as detailed in the [21]. Next, a computable lower bound for the left expansion properties of the code is established. Additionally, we have performed a thorough analysis to demonstrate how the Cartesian product of factor graphs adversely impacts the left expansion properties.

2. Unifying Framework

This subsection provides a unifying framework that simplifies the complexity of constructing high-rate quantum LDPC (qLDPC) codes by demonstrating how various seemingly different constructions can be integrated. This approach shows that nearly all existing high-rate qLDPC codes, including the Hypergraph product [7], Lifted product [12], Balanced product [11], and the left-right Cayley complex constructions [13][14][15], can be encompassed within what we define as the left-right Lifted product (lr-LP) code. This novel code structure builds upon the conventional Lifted product code [12] by incorporating the left-right technique to facilitate non-abelian lifting, enhancing its adaptability.

Following the algebraic definition of the left-right lifted product code, we provide a straightforward proof that this construction meets the stabilizer commutativity constraint, i.e. $H_x \cdot H_z^T = 0$. We also introduce a topological definition of the left-right Lifted product code to demonstrate the clear connection and equivalence between the left-right Lifted product and the Balanced product constructions. Ultimately, we incorporate the left-right Cayley complex construction [13] into this framework, shedding light on its algebraic properties that were previously unexplored in the literature.

2.1 Left-right Lifted product code

The following is the algebraic definition of the left-right Lifted product code. In this thesis, we denote matrices with bars above such as $\bar{H}$ to indicate that they are matrices over group algebra $\mathbb{F}_2[G]$.

Definition 1. *(Algebraic definition of left-right Lifted product code).*
The left-right lifted product code $LP(\bar{H}_1, G, \bar{H}_2)$ is a quantum CSS code with the following $\bar{H}_x, \bar{H}_z$ parity check matrix.

$$\bar{H}_z = \begin{bmatrix} \bar{H}_1^T \otimes \bar{I}_{m2} & \bar{I}_{n1} \otimes \bar{H}_2 \end{bmatrix}, \qquad \bar{H}_x = \begin{bmatrix} \bar{I}_{m1} \otimes \bar{H}_2^T & \bar{H}_1 \otimes \bar{I}_{n2} \end{bmatrix}$$

where $\bar{H}_x, \bar{H}_z$ and $\bar{H}_1, \bar{H}_2$ are matrices over group ring $\mathbb{F}_2[G]$ given arbitrary group G.

From Definition 1, the elements of matrix $\bar{H}_1, \bar{H}_2$ are in group ring $\mathbb{F}_2[G]$, so denote their

elements to be $h^1_{ij} = \sum_{g\in G} \alpha_{g_{ij}} g$ and $h^2_{ij} = \sum_{g\in G} \beta_{g_{ij}} g$, respectively.

Before the proof of the commutativity of X, Z stabilizers, it is useful to have two small concrete examples as follows.

The first example is the $LP(\bar{H}_1, C_3, \bar{H}_2)$ by using the left-right lifted product construction defined above, where C_3 denotes the cyclic group of order 3 and $\bar{H}_1, \bar{H}_2$ are defined in Example 1. It is well known that the group ring $\mathbb{F}_2[C_3]$ has a natural isomorphism to the quotient polynomial ring $\mathbb{F}_2[x]/(x^3-1)$. Using this isomorphism, we assume $h^1_{ij}, h^2_{ij} \in \mathbb{F}_2[x]/(x^3-1), \forall i, j$.

Remark (Transpose of a base matrix): Let $h_{ij} = \sum_{g\in G} \alpha_{g_{ij}} g$ be the group algebra element at the i^{th} row and j^{th} column of $\bar{m} \times \bar{n}$ base matrix $\bar{H}$. The transpose of $\bar{H}$, denoted as $\bar{H}^T$, is of size $\bar{n} \times \bar{m}$. Each element $(h^T)_{ij}$ of base matrix $\bar{H}^T$ is given by,

$$(h^T)_{ij} = (h_{ji})^T = \sum_{g\in G} \alpha_{g_{ji}} g^{-1}$$

Where each group element g of h_{ij} is taken inverse g^{-1} in the base element $(h^T)_{ij}$ because $gg_i = g_j \iff g^{-1}g_j = g_i$, which means that taking inverse transposes its algebraic lifting (regular representation) defined in Definition 4.

Example 1 (Abelian lift):

Let $\bar{H}_1, \bar{H}_2$ for $LP(\bar{H}_1, C_3, \bar{H}_2)$ be defined as follows,

$$\bar{H}_1 = \begin{bmatrix} 1 + x^2 \end{bmatrix} \Rightarrow \bar{H}_1^T = \begin{bmatrix} 1 + x \end{bmatrix}, \quad \bar{H}_2 = \begin{bmatrix} 1 & x \\ x & 1 \end{bmatrix} \Rightarrow \bar{H}_2^T = \begin{bmatrix} 1 & x^2 \\ x^2 & 1 \end{bmatrix}$$

Given $\bar{H}_1, \bar{H}_2$ with $\bar{n}_1 = \bar{m}_1 = 1, \bar{n}_2 = \bar{m}_2 = 2$, by Definition 1, the $\bar{H}_x, \bar{H}_z$ are obtained,

$$\bar{H}_z = \left[\begin{array}{cc|cc} 1+x & 0 & 1 & x \\ 0 & 1+x & x & 1 \end{array}\right], \qquad \bar{H}_x = \left[\begin{array}{cc|cc} 1 & x^2 & 1+x^2 & 0 \\ x^2 & 1 & 0 & 1+x^2 \end{array}\right]$$

Even though the so-called *left-right multiplication* in the Definition 2 has not been defined

yet, the commutativity can still be checked easily since C_3 is an abelian group.

$$\bar{H}_x \cdot \bar{H}_z^T = \begin{bmatrix} 1 & x^2 \\ x^2 & 1 \end{bmatrix} \begin{bmatrix} 1+x^2 & 0 \\ 0 & 1+x^2 \end{bmatrix} + \begin{bmatrix} 1+x^2 & 0 \\ 0 & 1+x^2 \end{bmatrix} \begin{bmatrix} 1 & x^2 \\ x^2 & 1 \end{bmatrix} = 0$$

However, the constraint of $\bar{H}_x \cdot \bar{H}_z^T = 0$ would break if the group G was non-abelian since multiplying the diagonal matrix $\bar{D}$ would not be sufficient for the commutativity of two matrices over the non-commutative ring, i.e. $\bar{A} \cdot \bar{D} \neq \bar{D} \cdot \bar{A}$.

Therefore, it is necessary to add a new trick defined in Definition 2 to make sure that the commutativity constraint still holds even if the group given is non-abelian. Before introducing the trick, first we need the following proposition which is a sufficient condition for the X, Z stabilizers to commute i.e. $\bar{H}_x \cdot \bar{H}_z^T = 0$. While this condition is also mentioned in [12], here we give a brief proof for this proposition.

Proposition 1. *All elements of $\bar{H}_1$ commuting with all elements of $\bar{H}_2$ is a sufficient condition for $LP(\bar{H}_1, G, \bar{H}_2)$ being a valid quantum CSS code i.e. $\bar{H}_x \bar{H}_z^T = 0$.*

Proof. We observe that

$$\bar{H}_x \cdot \bar{H}_z^T = \begin{bmatrix} \bar{I}_{m1} \otimes \bar{H}_2^T & \bar{H}_1 \otimes \bar{I}_{n2} \end{bmatrix} \begin{bmatrix} \bar{H}_1 \otimes \bar{I}_{n2} \\ \bar{I}_{n1} \otimes \bar{H}_2^T \end{bmatrix}$$

$$= (\bar{I}_{m1} \otimes \bar{H}_2^T)(\bar{H}_1 \otimes \bar{I}_{n2}) + (\bar{H}_1 \otimes \bar{I}_{n2})(\bar{I}_{n1} \otimes \bar{H}_2^T) \tag{2.1}$$

By expanding $(\bar{I}_{m1} \otimes \bar{H}_2^T)$ and $(\bar{H}_1 \otimes \bar{I}_{n2})$, the former results in a matrix where entry of $\bar{I}_{m1}$ is replaced with the corresponding $1 \cdot \bar{H}_2^T$ or $0 \cdot \bar{H}_2^T$, the latter results in a matrix where each entry h_{ij}^1 is replaced with a block $h_{ij}^1 \bar{I}_{n2}$. The product involves $\bar{H}_2^T$ multiplied by the diagonal block $h_{ij}^1 \bar{I}_{n2}$, so the result is $\bar{H}_2^T h_{ij}^1 \bar{I}_{n2} = \bar{H}_2^T h_{ij}^1$. Similarly, $(\bar{H}_1 \otimes \bar{I}_{n2})(\bar{I}_{n1} \otimes \bar{H}_2^T)$ reduces to $h_{ij}^1 \bar{H}_2^T$. Therefore, if $\bar{H}_1, \bar{H}_2$ are element-wise commuting, then this is sufficient to make $\bar{H}_2^T \cdot h_{ij}^1 = h_{ij}^1 \cdot \bar{H}_2^T$. □

However, the elements $h_{ij}^1,\ h_{ij}^2$ in general do not commute under group multiplication unless G is an abelian group. To force elements $h_{ij}^1,\ h_{ij}^2$ commute with each other, we need to

redefine the group multiplication to be the *left-right multiplication* given a non-abelian group G, as stated in the Definition 2.

Let g_i, g_j be two elements of given group G. Denote $f(G)$ be the space of all permutation maps from G to itself, and note that $g_i, g_j \in f(G)$ via group action. Hence, we can replace the notation of the group element g with $f_g^l(x) = g \cdot x$ just to emphasize g being a permutation map via left group action, and $f_g^r(x) = x \cdot g$ just to emphasis g being a permutation map via right group action.

Since the matrix elements h_{ij}^1, h_{ij}^2 as well as the elements in $\bar{H}_x, \bar{H}_z$ are all the element in the group algebra $\mathbb{F}_2[G]$, to be consistent, we view the group elements as the element in $\mathbb{F}_2[G]$ as well. Finally, the following defines the left-right multiplication for group elements,

Definition 2. *(Left-right multiplication for group elements).*
Given two permutation maps $f_{g_i}^a, f_{g_j}^b, a, b \in \{l, r\}, a \neq b$ corresponding to the group elements g_i, g_j, where one of them is via left, and the other is via right group action. The left-right multiplication M for group elements is defined as simply the composition of two permutation maps,

$$M : \mathbb{F}_2[G] \times \mathbb{F}_2[G] \to \mathbb{F}_2[G]$$

$$M(f_{g_i}^a, f_{g_j}^b)(x) = f_{g_i}^a \circ f_{g_j}^b(x)$$

where $a, b \in \{l, r\}, a \neq b$.

It is noted that the topological Balanced product defined in [11] applied similar ideas by letting one group action on the left and the other act on the right. The motivation for defining these left-right actions and multiplications is to make the non-commutative group action commutative.

Remark (Left-right trick in mathematics): In fields of mathematics such as Module Theory and Commutative Algebra, intuitively speaking, the left-right group action is a standard trick to make a non-commutative group action commutative.

Except for making sure of the commutativity, there is another reason for defining the left-right multiplication which is to make sense of the important object called base complex (see Definition 8) and its proto-complex (see Definition 6). As it is discussed in Chapter 3, this abstract object played a huge role in proving theorems in Chapter 3. The subtlety is that if we instead applied the left and right group representation as formulated in [13], then the base complex as well as its proto-complex as shown in Diagram 2.6 would no longer be well defined.

Lemma 1 (Left-right multiplication is commutative for elements in group ring)**.** *Given a group G, assume the commutative ring $\mathcal{R}$ to be $\mathbb{F}_2$, denote two elements $h_1 = \sum_{g\in G} \alpha_g g$, $h_2 = \sum_{g\in G} \beta_g g$ are in the group ring $\mathbb{F}_2[G]$. Assume h_1 acts by left group action and h_2 by right group action, so h_1, h_2 is commutative under the left-right multiplication.*

Proof. By Definition 2, we observe that

$$M(f^l_{h_1}, f^r_{h_2})(x) = f^l_{h_1} \circ f^r_{h_2}(x) = h_1 \cdot (x \cdot h_2) = \sum_{g\in G} \alpha_g g \cdot (x \cdot \sum_{g'\in G} \beta_{g'} g')$$

$$= \sum_{g\in G} \alpha_g g \cdot (\sum_{g'\in G} \beta_{g'} x \cdot g') = \sum_{g\in G} \sum_{g'\in G} \alpha_g \beta_{g'} (g \cdot (x \cdot g'))$$

Then, simply by associativity of group multiplications

$$\sum_{g\in G} \sum_{g'\in G} \alpha_g \beta_{g'} (g \cdot (x \cdot g')) = \sum_{g\in G} \sum_{g\in G} \alpha_g \beta_g ((g \cdot x) \cdot g)$$

$$= (\sum_{g\in G} \alpha_g g \cdot x) \cdot \sum_{g\in G} \beta_g g = (h_1 \cdot x) \cdot h_2 = f^r_{h_2} \circ f^l_{h_1}(x) = M(f^r_{h_2}, f^l_{h_1})(x).$$

Thus, this completes the proof

$$M(f^l_{h_1}, f^r_{h_2})(x) = M(f^r_{h_2}, f^l_{h_1})(x)$$

□

So far, the full picture of left-right Lifted product construction has been given, Now, it is useful to see another small concrete example. Unlike Example 1 where the group is abelian, in this case, we describe the simplest non-abelian lifted product construction to demonstrate the use of the left-right trick.

Here, the smallest non-abelian symmetric group S_3 is chosen. By the isomorphism with the dihedral group of order 6, i.e. $D_3 \cong S_3$, where $D_3 = \langle r, f | r^3 = f^2 = e, rf = fr^{-1} \rangle$, we instead use the notation of D_3 to simplify the representation for the group elements.

Example 2 (Non-abelian Lift): The $\bar{H}_1, \bar{H}_2$ for $LP(\bar{H}_1, S_3, \bar{H}_2)$ are as follows,

$$\bar{H}_1 = \begin{bmatrix} (1+r) & rf \end{bmatrix} \Rightarrow \bar{H}_1^T = \begin{bmatrix} (1+r^2) \\ rf \end{bmatrix}, \quad \bar{H}_2 = \bar{H}_2^T = \begin{bmatrix} 1 & f \\ f & 1 \end{bmatrix}$$

Given $\bar{H}_1, \bar{H}_2$ with parameter $\bar{n}_1 = 2, \bar{m}_1 = 1, \bar{n}_2 = \bar{m}_2 = 2$, by Definition 1, the $\bar{H}_x, \bar{H}_z$ are obtained,

$$\bar{H}_z = \left[\begin{array}{cc|cccc} (1+r^2) & 0 & 1 & f & 0 & 0 \\ 0 & (1+r^2) & f & 1 & 0 & 0 \\ rf & 0 & 0 & 0 & 1 & f \\ 0 & rf & 0 & 0 & f & 1 \end{array}\right]$$

$$\bar{H}_x = \left[\begin{array}{cc|cccc} 1 & f & 1+r & 0 & rf & 0 \\ f & 1 & 0 & 1+r & 0 & rf \end{array}\right]$$

Using the left-right multiplication, we are able to see the commutativity also holds for a non-abelian group now,

$$\bar{H}_x \cdot \bar{H}_z^T = \begin{bmatrix} 1 & f \\ f & 1 \end{bmatrix} \cdot \begin{bmatrix} 1+r & 0 & rf & 0 \\ 0 & 1+r & 0 & rf \end{bmatrix} +$$

$$\begin{bmatrix} 1+r & 0 & rf & 0 \\ 0 & 1+r & 0 & rf \end{bmatrix} \cdot \begin{bmatrix} 1 & f & 0 & 0 \\ f & 1 & 0 & 0 \\ 0 & 0 & 1 & f \\ 0 & 0 & f & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1+r & M(f,1+r) & rf & M(f,rf) \\ M(f,1+r) & 1+r & M(f,rf) & rf \end{bmatrix} +$$

$$\begin{bmatrix} 1+r & M(1+r,f) & rf & M(rf,f) \\ M(1+r,f) & 1+r & M(rf,f) & rf \end{bmatrix} = 0$$

By Lemma 2.1, we obtain $M(1+r,f) = M(f,1+r)$ as well as $M(f,rf) = M(rf,f)$, then the above two matrix are the same which implies $\bar{H}_x \cdot \bar{H}_z^T = 0$.

The above concrete examples should give the intuition that the commutativity constraint has been generalized to the case of non-abelian group lifting, as formally stated in the Theorem 1.

Theorem 1. *The left-right lifted product (lr-LP) code in the Definition 1 is a valid quantum CSS code.*

Proof. From Lemma 2.1, we see that $f_{h_1}^l$ commute with $f_{h_2}^r$ under left-right multiplication. Therefore, Proposition 1 holds and implies that $\bar{H}_x \cdot \bar{H}_z^T = 0$. □

So far, the formal algebraic definition of the lr-LP code has been given, it is time to make several remarks that show almost all high-rate quantum LDPC codes can be incorporated into the given construction.

Remark (Hypergraph product code [7] and abelian LP code [12] are lr-LP codes): It is obvious that when the group is the trivial group $\{e\}$, the element of $\bar{H}_1, \bar{H}_2$ will reduce to the element in $\mathbb{F}_2$, the left-right LP code will then reduce to the hypergraph product code.

It is also easy to see when the given group is abelian, there is no difference between the left group action and the right group action. Through the lens of representation theory, it is equivalent to saying that the left representation and the right representation are the same for elements in the abelian group. Hence, the left-right LP gives the same construction as the abelian LP code in [12][16].

Remark (Quantum bicycle code [16] and two-block group algebra code [17] are lr-LP codes): The two-block group algebra code has the following parity check matrix based on its definition [17],

$$H_x = \big[A, B\big], \quad H_z = \big[B^T, A^T\big]$$

where A, B are $l \times l$ square commuting matrices. If we map this code back to the base matrix of the given lr-LP construction, then we see that $\bar{H}_1, \bar{H}_2$ are simply scalar group algebra elements denoted by $\bar{H}_1 = b, \bar{H}_2 = a^T$, with $a, b \in \mathbb{F}_2[G]$. Therefore, the $\bar{H}_x, \bar{H}_z$ are given by,

$$\bar{H}_x = \big[e \otimes a, b \otimes e\big] = \big[a, b\big], \quad \bar{H}_z = \big[b^T \otimes e, \ a^T \otimes e\big] = \big[b^T, a^T\big]$$

It follows that such a two-block group algebra code can be seen as the simplest lr-LP code with the base matrix $\bar{H}_1, \bar{H}_2$ reduced to a single scalar group algebra element.

If we further restrict the group G to be abelian, then above two-block group algebra code will reduced to the quantum bicycle code [16][18].

Theorem 1 verifies the validity of the left-right lifted product code of Definition 1. However, just like the definition of the two-block group algebra code given in [17], we still need to map the base code to a real quantum CSS code and lift the base parity check matrices over the group algebra $(\bar{H}_x, \bar{H}_z)$ to parity check matrices over the binary field (H_x, H_z). Hence, for this reason, the following definition of algebraic lifting as well as the left and right representations are given.

Definition 3. *(Left and right representations).*
Given a group G, its regular representation is a representation where G acts on a vector space V constructed from G itself with a basis corresponding bijectively to the elements of G. That is, if $G = \{g_1, g_2, ..., g_n\}$, then V has a basis $\{e_{g_1}, e_{g_2}, ..., e_{g_n}\}$, where each e_g is a basis vector corresponding to the element $g \in G$. Then for each $g \in G$, its left (resp. right) representation $\lambda(g)$ (resp. $\rho(g)$) is linear transformation acting on basis vectors by permuting them according to the left (resp. right) group action.

$$\lambda(g)(e_h) = e_{g \cdot h}, \quad \rho(g)(e_h) = e_{h \cdot g}, \qquad \forall h \in G$$

where $\lambda(g)$ is also called the left-regular representation of g in traditional Representation Theory [24].

This definition of left and right representations is also given in [13]. However, unlike the terminologies used in [13], we are not calling $\rho(g)$ the right-regular representation in this thesis. This is because, in representation theory, the right-regular representation is instead defined as $\rho(g)(e_h) = e_{h \cdot g^{-1}}$ to guarantee the group isomorphism property of representation. Hence, this thesis calls the linear transformations in Definition 3 the left and right representations in order to avoid the potential confusion of terminologies.

Given the left and right representations, we can define the algebraic lifting as below.

Definition 4. *(Algebraic lifting).*
Given a group G with group order $|G| = l$, the left and right G-lift are the maps from an element in group algebra to its left and right representation defined in Definition 3, respectively. To be consistent with the corresponding notations and terminologies in algebraic topology [23], we denote left and right G-lift as π_l^{-1}, π_r^{-1}, respectively.

Left G-lift π_l^{-1}:

$$\mathbb{F}_2[G] \longrightarrow \mathbb{F}_2^{l \times l}, \quad \pi_l^{-1} : \sum_{g \in G} \alpha_g g \mapsto \sum_{g \in G} \alpha_g \lambda(g)$$

Right G-lift π_r^{-1}:

$$\mathbb{F}_2[G] \longrightarrow \mathbb{F}_2^{l \times l}, \quad \pi_r^{-1} : \sum_{g \in G} \alpha_g g \mapsto \sum_{g \in G} \alpha_g \rho(g)$$

where $\lambda(g), \rho(g)$ are the left, right group representation of element $g \in G$, respectively, as defined in Definition 3.

After the definition of algebraic lifting, the codes, $LP(\bar{H}_1, C_3, \bar{H}_2)$ and $LP(\bar{H}_1, D_3, \bar{H}_2)$ in Example 1 and 2 can finally be mapped into the form of quantum CSS code with H_x, H_z being lifted into the binary field.

Definition 5. *(Algebraic projection (covering)).*

The algebraic projection or algebraic covering map π in this context is defined as the inverse map of the lifting map, which maps a square matrix in $F^{l\times l}$ into the corresponding element in group algebra $\mathbb{F}_2[G]$.

Since the base matrices $\bar{H}_1, \bar{H}_2$ always have a lifting structure under a group G for our lr-LP code construction, one can check that the map of algebraic lifting defined in Definition 4 is bijective (the map is an isomorphism). Hence, the inverse map always exists and the algebraic projection is well-defined.

Remark (Clarification of the definitions used in this thesis): It is worthwhile to discuss the concepts and terminologies defined above. Firstly, the terms "lifting" and "covering" are borrowed from algebraic topology [23]. However, their exact meanings and definitions in this thesis slightly differ from those found in a standard algebraic topology textbook. The formal definitions in such textbooks are often obscure and unnecessary for our purposes. This is partly because our codes are purely combinatorial objects, so we should be less concerned (if not completely unconcerned) about the topological continuity of the map. Secondly, the formal definitions—and many others in modern pure mathematics—are defined "categorically". This means that usually, the maps (morphisms) and objects are formally described by their relationships to other mathematical objects and by how they interact with each other using commutative diagrams. Although this categorical approach to defining concepts is rigorous and elegant, it can add unnecessary abstraction and confusion for readers. This violates one of the objectives of this thesis, which is to use less abstract terms to increase accessibility.

Furthermore, there is a subtle distinction in what the terms 'lifting' and 'covering' really mean in this thesis compared to their formal definitions in algebraic topology. Traditionally in algebraic topology, the covering space, represented here by binary matrices H with a lifting structure is first defined. The covering map π, in algebraic topology, is surjective

but not injective as π maps multiple distinctive points into a single point. What this means in our context is that the map π should have projected the binary matrix H onto a proto-matrix $\bar{P}$ as defined below instead of a base matrix $\bar{H}$.

Definition 6. *(Proto-matrix).*

Given a $\bar{m} \times \bar{n}$ base matrix $\bar{H}$ with elements in the group algebra, the corresponding proto-matrix $\bar{P}$ is defined as the $\bar{m} \times \bar{n}$ binary matrix with element "1" assigned to the entries (i, j) where h_{ij} is a non-zero group algebra element. The name "proto-matrix" is also consistent with the terminology used in the classical literature on quasi-cyclic code [25].

Example 3 below is a concrete example in order to clarify the subtle connections between the concepts - "covering" and "lifting" - in this thesis and the corresponding definitions in algebraic topology.

Example 3 (Algebraic projection (covering)):

This example is based on Example 1, and the group chosen here is the cyclic group of order 3, i.e. $G = C_3$.

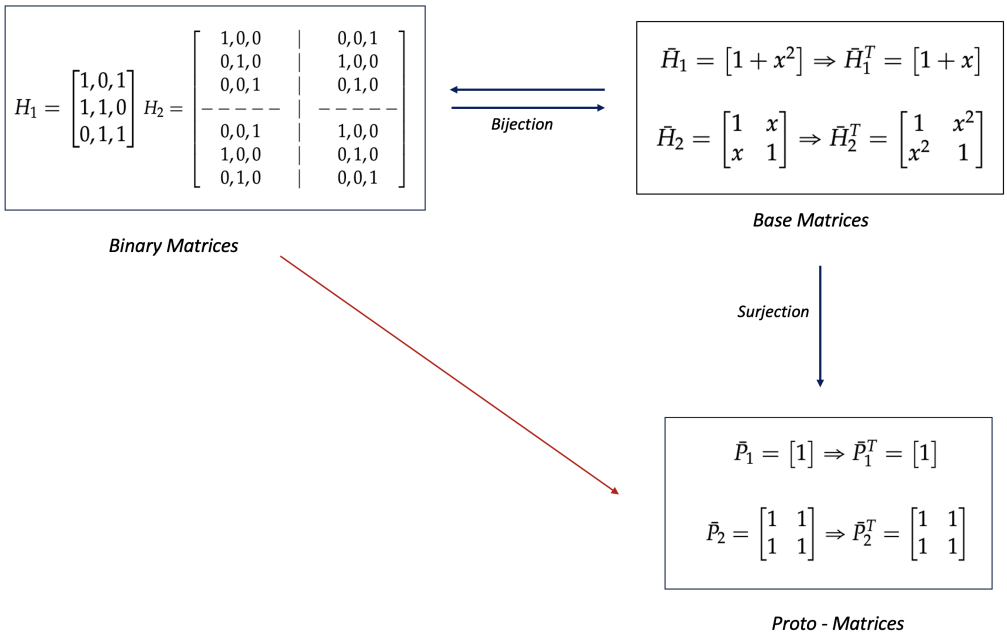

Figure 2.1: The picture of algebraic projection and covering.

From Figure 2.1, the map marked by the red arrow from binary matrices to proto-matrices is surjective, which is basically what the covering map means in algebraic topology. The inverse lifting map from proto-matrices to binary matrices will not be unique.

However, it is noted that for left-right lifted product code construction, we always enforce some lifting structures by defining the base matrices in advance. Hence, there is a unique lifting map from proto-matrices to binary matrices by mapping each element in $\bar{P}$ to the corresponding defined group algebra element h_{ij} and then mapping back to the binary matrices by algebraic lifting defined in Definition 4.

Therefore, for our code construction, the covering and lifting maps are unique and mutually inverse if the base matrices are given. This is why in this thesis, the algebraic covering and lifting maps are defined between binary matrices and base matrices instead of proto-matrices. As discussed above, the definitions of algebraic lifting and covering are still fundamentally equivalent to the original definitions in algebraic topology, but we need this discussion to clarify the subtleties. The same logic also applies to the topological covering and lifting defined in Section 2.2.

2.2 Connection to the Balanced Product Construction

To prove the equivalence and show the explicit connections between the left-right lifted product and the balanced product constructions, the topological perspective (factor graph and factor complex) of the lr-LP code should also be defined correspondingly.

There are several motivations for the topological aspects of the qLDPC code. One is from classical coding theory, every code can be represented by a factor graph (also known as combinatorial topological spaces in [26]) which is particularly useful for decoding analysis and graph-based code design [27]. Another obvious motivation is due to the essential product structure for constructing high-rate qLDPC code, which originated from the product in algebraic topology and is more intuitive to see the structure by viewing the code topologically as a graph and complex.

For this thesis, the topological illustration is particularly necessary for the demonstration of the equivalence between the Balanced product code and the, yet to been seen, left-right Cayley complex. It is partly because the original paper [11][13][14][15] is mostly written in topological and homological languages. Additionally, the interaction between algebraic and topological formalism will be utilized to prove a few new theorems in Chapter 3. We believe their proofs would be far more complicated to prove if one only uses one of the mathematical languages.

Firstly, we define a Cartesian product of two factor graphs which can be seen as a generalization of the Cartesian product of two classical graphs [28]. The factor graph Cartesian product is an essential component for the topological Lifted product construction.

Definition 7. *(Cartesian product of two factor graphs).*
Given two factor graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{F}_1, \mathcal{E}_1)$ *with* $|\mathcal{F}_1| = m_1, |\mathcal{V}_1| = n_1$ *and* $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{F}_2, \mathcal{E}_2)$ *with* $|\mathcal{F}_2| = m_2, \mathcal{V}_2 = n_2$*, the Cartesian product of two graphs* $\mathcal{G}_1 \square \mathcal{G}_2$ *is also a factor graph* $(\mathcal{V}, \mathcal{F}, \mathcal{E})$ *with,*

$$\mathcal{V} = \Big\{ \mathcal{V}_1 \times \mathcal{F}_2 \cup \mathcal{F}_1 \times \mathcal{V}_2 \Big\}, \quad \mathcal{F} = \Big\{ \mathcal{F}_1 \times \mathcal{F}_2 \Big\}$$

$$\mathcal{E} = \left\{ \left\{ (v_1 \times f_2,\ \ f_1 \times f_2) : (v_1, f_1) \in \mathcal{E}_1, f_2 \in \mathcal{F}_2 \right\} \cup \left\{ (f_1 \times v_2,\ \ f_1 \times f_2) : (v_2, f_2) \in \mathcal{E}_2, f_1 \in \mathcal{F}_1 \right\} \right\}.$$

where the product structure results the factor graph $\mathcal{G}_1 \square \mathcal{G}_2$ *has* $m_1 \cdot m_2$ *factor nodes and* $(n_1 m_2 + m_1 n_2)$ *variable nodes, i.e.* $|\mathcal{F}| = m_1 m_2,\ |\mathcal{V}| = n_1 m_2 + m_1 n_2$.

Remark (Equivalence to the Cartesian Product of Hypergraphs):
It is noted that if we instead view the check node $\mathcal{F}$ as vertices and the variable node $\mathcal{V}$ as (hyper)edges, every factor graph then has a one-to-one correspondence to a hypergraph, which is another common code representation, especially for the topological codes such as Toric code [4]. If applying the hypergraph representation, the above Cartesian product of factor graphs in Definition 7 will be equivalent to the definition of the hypergraphs Cartesian product defined in [29]. The equivalence is also consistent with the hypergraph product in [7].

Example 3 (Cartesian product of factor graphs):
Figure 2.2 gives a simple example of the Cartesian product of two factor graphs. As shown in Figure 2.2, the number of variable nodes and check nodes are $n_1 = 4, m_1 = 2$, $n_2 = 2, m_2 = 3$, respectively. It is easy to check that the result factor graph $\mathcal{G}_1 \square \mathcal{G}_2$ has $m_1 \cdot m_2 = 6$ factor nodes and $n_1 m_2 + n_2 m_1 = 4 \times 3 + 2 \times 2 = 16$ variable nodes.

So far, the topological aspect of the product construction has been introduced. In the next definition, we state the other important ingredient of lr-LP construction - lifting, from the topological point of view as well. The definitions of the base factor graph (see Definition 8) and the topological lifting of factor graphs (see Definition 9) are both inspired by the definition of the voltage graph and its derived graph in [30]. Hence, the following definitions can be considered to generalize the voltage graph to the factor graph category.

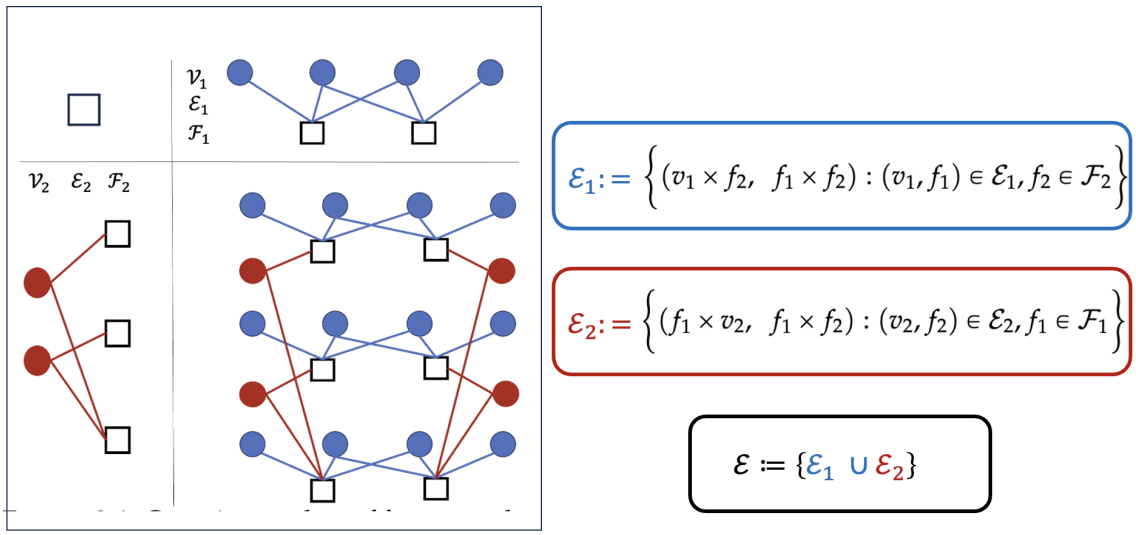

Figure 2.2: Cartesian product of factor graphs.

Additionally, it is the Cartesian product of the base factor graphs that corresponds to the algebraic definition of lr-LP code in (1). Similar to the notations of matrices over group algebra, all the base graphs in this thesis have bars above such as $\bar{\mathcal{G}}$ to distinguish them from the classical graphs.

Definition 8. *(Base factor graph and Base factor complex).*
Given a group G, a base factor graph $\bar{\mathcal{G}} := (\bar{\mathcal{V}}, \bar{\mathcal{F}}, \bar{\mathcal{E}}, \alpha)$ is a factor graph with directed edges that are labeled invertible by elements in a group algebra $\mathbb{F}_2[G]$. The assignment function $\alpha : \bar{\mathcal{E}} \to \mathbb{F}_2[G]$ labels each directed edge of $\bar{\mathcal{G}}$ with an element $\sum_{g \in G} \alpha_g g$, where $\alpha_g \in \mathbb{F}_2$ is edge dependent. A base factor complex $\bar{\mathcal{G}} := (\alpha_z, \bar{\mathcal{F}}_z, \bar{\mathcal{E}}_z, \bar{\mathcal{V}}, \bar{\mathcal{E}}_z, \bar{\mathcal{F}}_x, \alpha_x)$ is a union of two base factor graphs $\mathcal{G}_z = (\bar{\mathcal{V}}, \bar{\mathcal{F}}_z, \bar{\mathcal{E}}_z, \alpha_z)$ and $\mathcal{G}_x = (\bar{\mathcal{V}}, \bar{\mathcal{F}}_x, \bar{\mathcal{E}}_x, \alpha_x)$ defined on the same set of variable nodes $\bar{\mathcal{V}}$.

Definition 9. *(Topological lifting and topological projection).*
Given a group G, the topological lifting π^{-1} of a base factor graph $(\bar{\mathcal{G}}, \alpha)$ is the factor graph $\mathcal{G}$ whose variable nodes $\mathcal{V}$ and factor nodes $\mathcal{F}$ are $\mathcal{V} = \bar{V} \times |G|, \mathcal{F} = \bar{F} \times |G|$, respectively. The edge set $\mathcal{E} = \bar{\mathcal{E}} \times |G|$ is constructed by generating one edge $(\bar{e}, g) \in \bar{\mathcal{E}}$ for each $\bar{e} = (\bar{v}, \bar{w}) \in \bar{\mathcal{E}}$ and $g \in G$. This edge connects $(\bar{v}, g)$ to $(\bar{w}, \alpha(\bar{e}) \cdot g)$ for the left lifting π_l^{-1}, and to $(\bar{w}, g \cdot \alpha(\bar{e}))$ for right lifting π_r^{-1}.

Like algebraic projection, the topological projection(topological covering) π is the inverse of topological lifting, i.e. $(\mathcal{G}, \alpha) \to \bar{\mathcal{G}}$.

Example 4 (Topological lifting of factor graphs):

The following give two concrete examples of topological lift. One is the abelian cyclic group C_3-lift shown on the left side, the other is a non-abelian symmetric group D_3-lift.

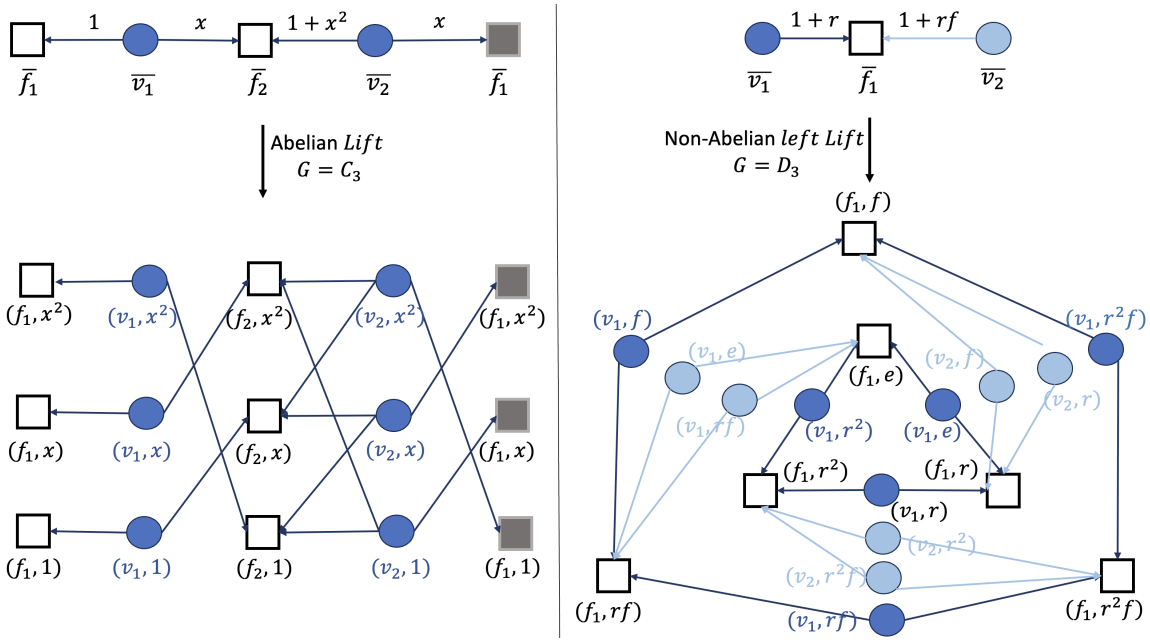

Figure 2.3: Topological abelian and non-abelian lift of base factor graphs.

As shown in Figure 2.3, the left cyclic lift is chosen because it is studied extensively in classical quasi-cyclic code which will have a topological picture exactly like the lifting picture shown on the left [31]. On the right, the base graph for non-abelian group lifting is chosen because we aim to show the notion of Cayley graph $Cay(G, A)$ in group theory can also be obtained by the lifting process. Indeed, the figure on the right is the Cayley graph of the Dihedral group D_3 of order 6 given the two generators a rotation $g_1 = r$ and a reflection $g_2 = rf$, except for a minor subtle point which is discussed below.

Technical Subtleties: (Directed edges and Double Cover)

As given in the Definition 8 and 9, the base factor graph and its lifting have to be directed graph in order to specify the direction of the group action. To make a directed graph undirected as a normal factor graph in coding theory, each directed edge e must have its own inverse $e^{'}$ which is another directed edge going the opposite direction. This implies that its assignment function $\alpha^{'} = \alpha^{-1}$ the inverse of the group element being assigned g^{-1}. Additionally, if the assignment function instead assign a group algebra element $\sum_{g\in G} \alpha_g g$ to an edge e, the opposite edge $e^{'}$ should assign the inverse element as $\sum_{g\in G} \alpha_g g^{-1}$ instead of the inverse of the group algebra element $(\sum_{g\in G} \alpha_g g)^{-1}$.

Another technical subtle point relates to the group element of order two which is also pointed out in [13]. As shown on the left of Figure 2.3, the lifting graph is almost the Cayley graph except for generator rf(light blue nods and edges). There are supposed to be three edges in the correct Cayley graph since $rf = rf^{-1}$ while in the lifting graph, we have six edges. Therefore, the group element of order two is problematic, and the paper [13][15] instead uses the double cover of the base graph to make it exactly the same as the double cover of the correct Cayley graph.

So far, we have all the ingredients for the topological definition of the left-right Lifted product code. It is worth mentioning that the topological product definition shown below is motivated by the topological product of cell complexes [32] in algebraic topology, and this thesis generalizes it to the factor complex (two-dimensional partially ordered set [13]). It is also worth noting that the definition is similar to the definition of the Tanner graph of the Hypergraph product code [29], except we take a product of two base factor graphs.

Definition 10. *(Topological Product of base factor graphs).*
Given two base factor graphs $\bar{\mathcal{G}}_1 = (\bar{\mathcal{V}}_1, \bar{\mathcal{F}}_1, \bar{\mathcal{E}}_1)$ and $\bar{\mathcal{G}}_2 = (\bar{\mathcal{V}}_2, \bar{\mathcal{F}}_2, \bar{\mathcal{E}}_2)$, the topological product of $\bar{\mathcal{G}}_1$ and $\bar{\mathcal{G}}_2$ is the base factor complex (see Definition 8) denoted by $\bar{\mathcal{G}}_1 \times \bar{\mathcal{G}}_2$ with two base factor graphs as components $\bar{\mathcal{G}}_x, \bar{\mathcal{G}}_z$,

$$\bar{\mathcal{G}}_z = \bar{\mathcal{G}}_1 \square \bar{\mathcal{G}}_2, \quad \bar{\mathcal{G}}_x = \bar{\mathcal{G}}_1^* \square \bar{\mathcal{G}}_2^*$$

where $\mathcal{G}^*$ is the variable-factor dual of the factor graph $\mathcal{G}$ by swapping the variable nodes and the factor nodes, i.e. $\mathcal{V}(\mathcal{G}^*) = \mathcal{F}(\mathcal{G}), \mathcal{F}(\mathcal{G}^*) = \mathcal{V}(\mathcal{G}), \mathcal{E}(\mathcal{G}^*) = \mathcal{E}(\mathcal{G})$.

Finally, the left-right Lifted product code in Definition 1 can be also defined topologically.

Definition 11. *(Topological definition of the left-right lifted product code).*
The left-right Lifted product code is a base factor complex $\bar{\mathcal{G}} = \bar{\mathcal{G}}_1^* \times \bar{\mathcal{G}}_2$ obtained by taking the topological product (see Definition 10) of two base factor graphs $\mathcal{G}_1^*, \mathcal{G}_2$.

$$\mathcal{G}_z = \mathcal{G}_1^* \ \square \ \mathcal{G}_2, \quad \mathcal{G}_x = \mathcal{G}_1 \ \square \ \mathcal{G}_2^*$$

Lemma 2. *The topological definition of the left-right lifted product construction in Definition 11 is consistent with the algebraic definition defined in Definition 1. By consistent, it means that the parity check matrices $\bar{H}_x, \bar{H}_z$ given in Definition 1 equal to the two adjacency matrices between $\bar{\mathcal{V}}, \bar{\mathcal{F}}_x$ and $\bar{\mathcal{V}}, \bar{\mathcal{F}}_z$, respectively, in the base complex $\bar{\mathcal{G}}$.*

Proof. Firstly, using Definition 11, the edge set $\bar{\mathcal{E}}_z$ in the product graph $\bar{\mathcal{G}}_z = \bar{\mathcal{G}}_1^* \square \bar{\mathcal{G}}_2$ is

$$\mathcal{E}_z = \left\{ \left\{ (f_1 \times f_2, v_1 \times f_2) : (v_1, f_1) \in \mathcal{E}_1, f_2 \in \mathcal{F}_2 \right\} \cup \left\{ (v_1 \times v_2, v_1 \times f_2) : (v_2, f_2) \in \mathcal{E}_2, v_1 \in \mathcal{V}_1 \right\} \right\},$$

given the fact that parity check matrix is simply the adjacency matrix of the factor graph, we obtain the parity check matrix $\bar{H}_z = \begin{bmatrix} \bar{H}_1^T \otimes \bar{I}_{m2} & \bar{I}_{n1} \otimes \bar{H}_2 \end{bmatrix}$. Similarly, for the other component $\bar{\mathcal{G}}_x = \bar{\mathcal{G}}_1 \square \bar{\mathcal{G}}_2^*$, the resultant edges are

$$\mathcal{E}_x = \left\{ \left\{ (f_1 \times f_2, f_1 \times v_2) : (v_2, f_2) \in \mathcal{E}_2, v_1 \in \mathcal{V}_1 \right\} \cup \left\{ (v_1 \times v_2, f_1 \times v_2) : (v_1, f_1) \in \mathcal{E}_1, v_2 \in \mathcal{V}_2 \right\} \right\},$$

and so the base parity check matrix for X-stabilizers can be obtained in a similar way as $\bar{H}_x = \begin{bmatrix} \bar{I}_{m1} \otimes \bar{H}_2^T & \bar{H}_1 \otimes \bar{I}_{n2} \end{bmatrix}$. Therefore, we find that the topological Cartesian product equals the algebraic one given in Definition in 1.

Next, we need to prove that the algebraic lift exactly corresponds to the topological lift. Without loss of generality, assume it is the left lift π_l^{-1}. From Definition 9 (topological lift), each edge $\bar{e} = (\bar{v}, \bar{w}), \bar{v} \in \bar{\mathcal{V}}, \bar{w} \in \bar{\mathcal{F}}$ is lifted to $|G|$ copies (assume the element being assigned $\alpha(\bar{e}) = a, a \in G$ for simplicity) connecting $e = \pi_l^{-1}(\bar{e}, \alpha(\bar{e})) =$

$\pi_l^{-1}\Big((\bar{v},\bar{w}),\alpha(\bar{e})\Big) = \Big((\bar{v},g),(\bar{w},\alpha(\bar{e})\cdot g)\Big), \forall g \in G$. Mapping to the algebraic representation, the edge $\bar{e} = (\bar{v},\bar{w})$ and its assignment $\alpha(\bar{e})$ correspond to the matrix element $h_{ij} = \alpha(\bar{e})$ of the base parity check matrix $\bar{H}$, where i,j are the row and column indices by which the base variable node $\bar{v}$ and factor node $\bar{w}$ are labeled, respectively. From the definition of algebraic lifting (see Definition 4), the group element $h_{ij} = \alpha(\bar{e})$ is lifted to the $|G| \times |G|$ permutation matrix which is given by the left representation $\lambda(\alpha(\bar{e}))$. By definition of (3), each basis e_g is mapped to the basis $e_{\alpha(\bar{e})\cdot g}$, which implies that there is an edge connecting $(\bar{v},g) \in \mathcal{V}$ and $(\bar{w},\alpha(\bar{e})\cdot g) \in \mathcal{F}$ in graph $\mathcal{G}$ after lifting.

Thus, the resultant edge obtained by algebraic lifting is verified by topological lifting. This completes the proof. □

Example: (Topological picture of $LP(\bar{H}_1, C_3, \bar{H}_2)$ in example 1)

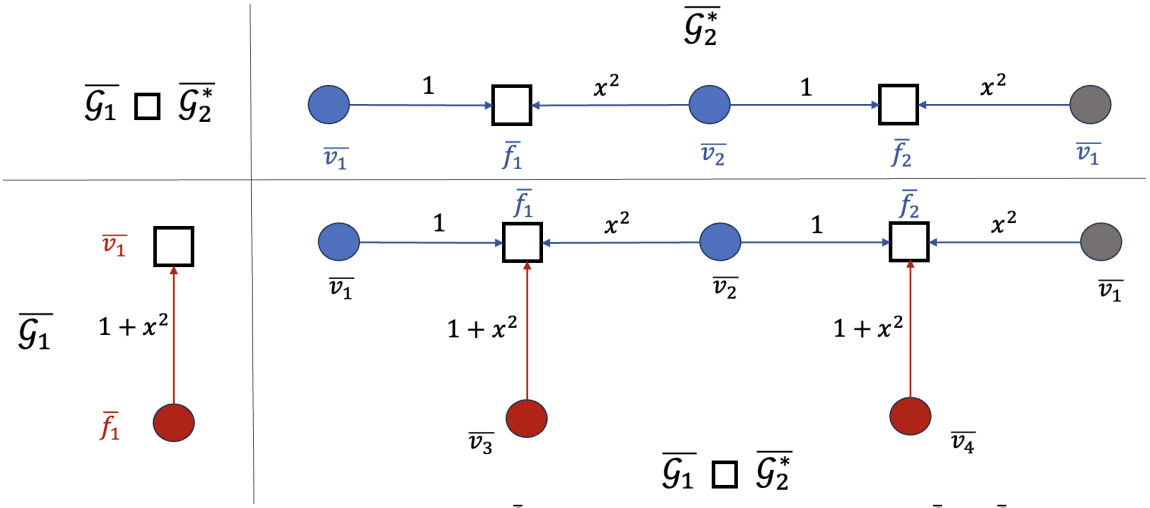

Figure 2.4: Base factor graph $\bar{\mathcal{G}}_x$ in the product base complex $\bar{\mathcal{G}}_1 \times \bar{\mathcal{G}}_2$.

To prove the equivalence of Balanced product code, we have to clarify the notation, especially since the paper uses advantage topological terminologies. However, it should be emphasized that the level of abstraction and the mathematical rigor in [11] is, in some sense, unnecessary. This is because, for our code, everything can be perfectly defined as combinatorial, and there is no concern for the topological continuity. Thus, the formal

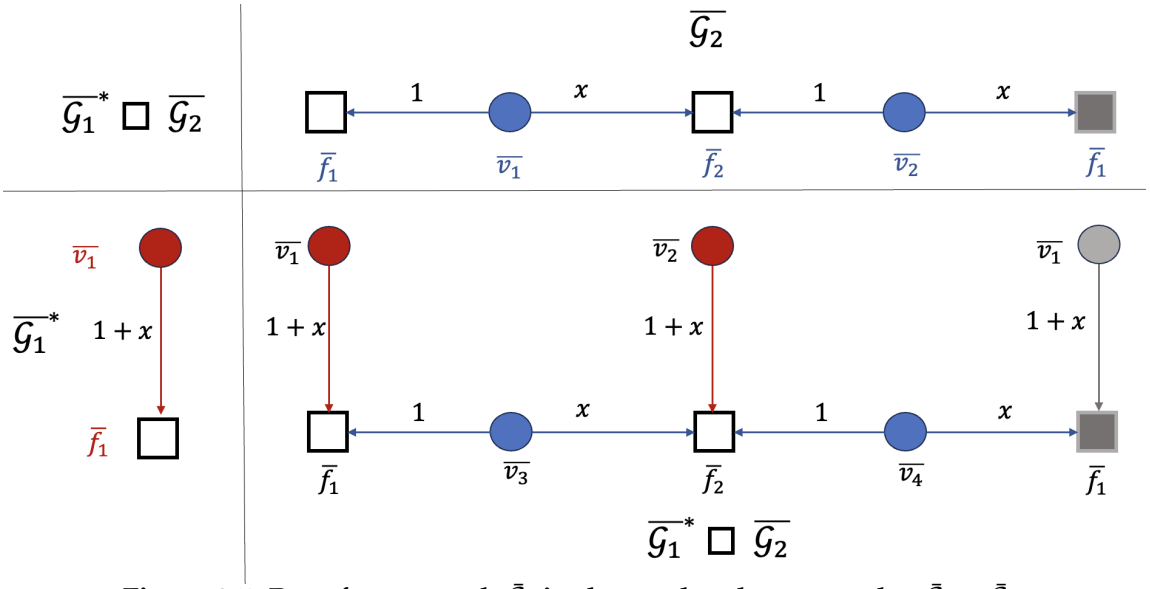

Figure 2.5: Base factor graph $\bar{\mathcal{G}}_z$ in the product base complex $\bar{\mathcal{G}}_1 \times \bar{\mathcal{G}}_2$.

mathematical definition in algebraic topology and Fiber Bundle theory might not be the best in our combinatorial setting. However, in order to demystify the connection with Balanced product construction, it is still necessary to discuss their notation and state the equivalence with ours for the convenience of the final proof.

Remark: (Consistent notation and terminologies in [11])

Even though we define the projection map π as the inverse of lifting, it can also be viewed as the quotient map where the group being quotient out is the group being lifted G. It is easy to check the group action is always free, and the group orbits $\mathcal{O}$ are simply the $\{(v_i, g) \cup (f_j, g)) : g \in G, i \in [n], j \in [m]\}$. Thus, the covering map is simply the quotient map $\mathcal{G}/\mathcal{O}$ used in [11]. Since the orbit can be exactly represented as the group G, people in the field of principal Fiber Bundle use the notion $\mathcal{G}/G$ to represent the base graph $\bar{\mathcal{G}}$.

$$\pi^l : \mathcal{G} \mapsto \bar{\mathcal{G}} = \mathcal{G}/{}^l\mathcal{O} = \mathcal{G}/{}^l G$$

$$\pi^r : \mathcal{G} \mapsto \bar{\mathcal{G}} = \mathcal{G}/\mathcal{O}^r = \mathcal{G}/G^r$$

Now, it is time to demonstrate the exact connection between the balanced product construction in [11] and the left-right lifted product code and prove the exact equivalence under group lifting.

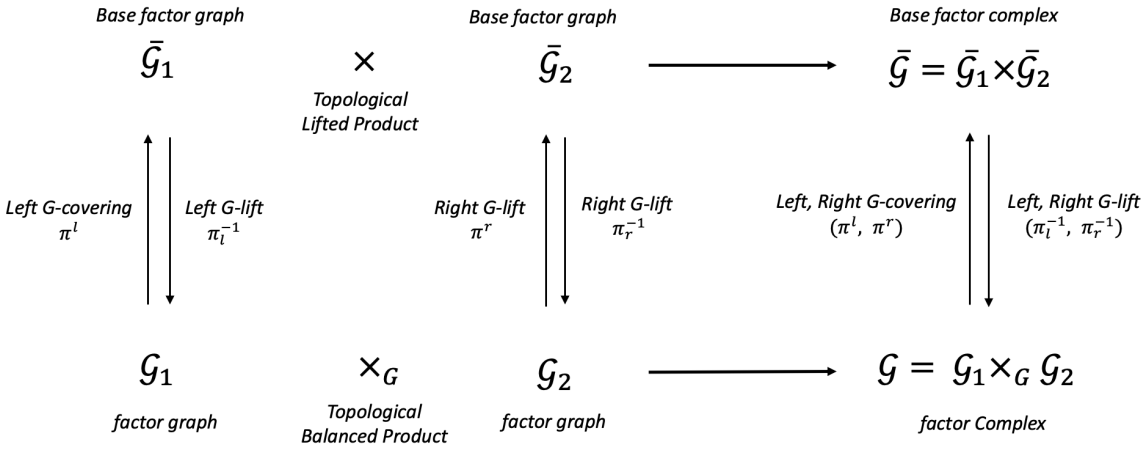

Figure 2.6: Diagram for topological Lifted and Balanced product construction.

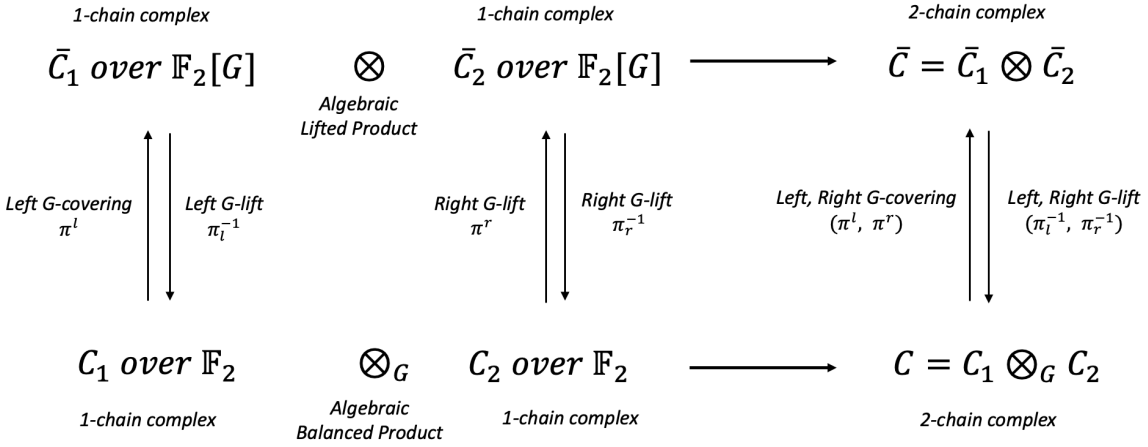

Figure 2.7: Diagram for algebraic Lifted and Balanced product construction.

Theorem 2. *Diagram 2.6 is a commutative diagram, and consequently, the balanced product code construction is equivalent to the left-right lifted product code construction.*

Proof. Starting from two factor graphs $\mathcal{G}_1, \mathcal{G}_2$,

$$\pi^l : \mathcal{G}_1 \mapsto \bar{\mathcal{G}}_1 = \mathcal{G}_1 / {}^l G, \qquad \pi^r : \mathcal{G}_2 \mapsto \bar{\mathcal{G}}_2 = \mathcal{G}_2 / G^r$$

Take a topological product of two base factor graphs,

$$LP(\bar{\mathcal{G}}_1, G, \bar{\mathcal{G}}_2) = \bar{\mathcal{G}}_1 \times \bar{\mathcal{G}}_2 = \mathcal{G}_1 / {}^l G \times \mathcal{G}_2 / G^r$$

Take a left-right G lift,

$$\pi_{l,r}^{-1} : \mathcal{G}_1 / {}^l G \times \mathcal{G}_2 / G^r \;\; \mapsto \;\; \mathcal{G}_1 / {}^l G \times {}^l G^r \times \mathcal{G}_2 / G^r = (\mathcal{G}_1 \times \mathcal{G}_2) / G$$

One can easily check, the factor complex $(\mathcal{G}_1 \times \mathcal{G}_2)/G$ constructed in this way has a group action g such that $(g_1, g_2) \cdot g = (g \cdot g_1, g_2 \cdot g)$, which satisfies the definition of Balanced product code defined in [11].

□

Example: (The commutative diagram of a concrete code)

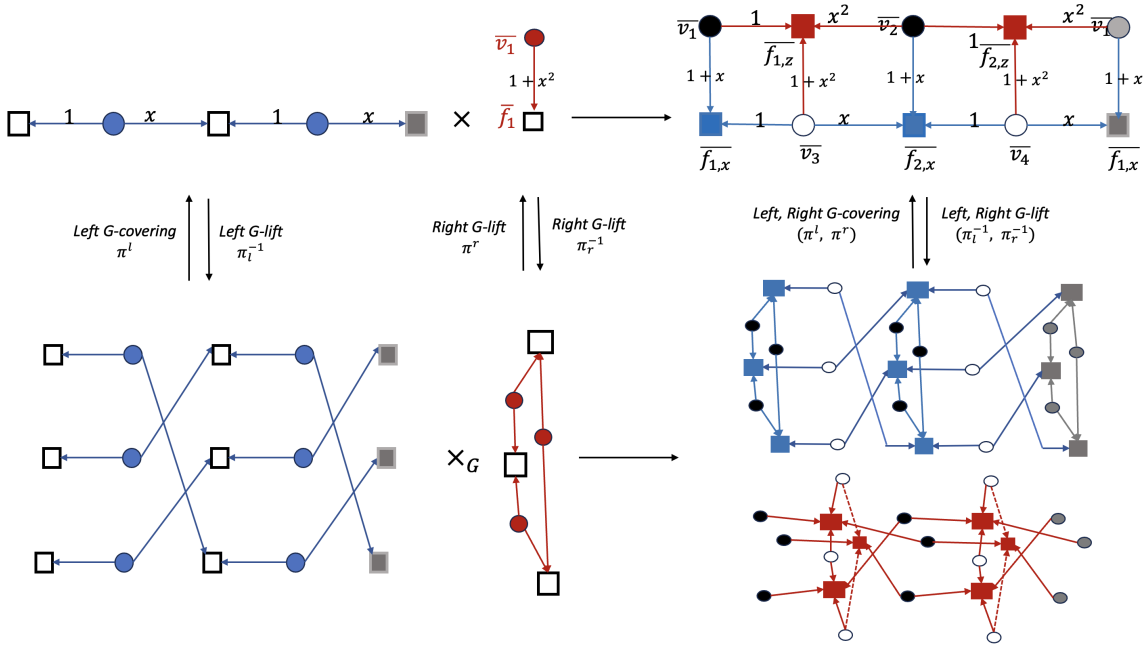

Figure 2.8: The commutative diagram of the code in example 1.

2.3 Connection to the Left-right Cayley Complex

So far, we have formally proved the equivalence between the Balanced product and left-right Lifted product construction, and demonstrate the exact connection between these two constructions. There is one step left for the big unification. Thus, in this section, we finally incorporate the last left-right Cayley expander construction into our framework.

First, the original definition of the left-right Cayley complex from [14][15] is given below. Following the Definition 12, we prove the Cayley complex being lr-LP construction by specifying its base parity check matrices $\bar{H}_1$ and $\bar{H}_2$, and show that the left-right Lifted product of such two base matrices exactly leads to a topological object satisfying the Cayley complex definition.

Definition 12. *(Left-right Cayley complex [14][15]).*
Given two generator sets A, B such that $A = A^{-1}, B = B^{-1}$ and A, B satisfies the Total No-Conjugacy condition(TNC) [14] i.e. $\forall a \in A, b \in B, g \in G, ag \neq gb$. A left-right Cayley complex X is made up of vertices, $A-$edges, $B-$edges, and squares. The vertex set is G, the $A-$edges are pairs of vertices of the form $\{g, ag\}$ for $g \in G$ and $B-$edges are of the form $\{g, gb\}$ for $g \in G, a \in A, b \in B$. A square is a set of group elements of the form $\{g, ag, gb, agb\}$.

Theorem 3. *The left-right Cayley complex $Cay(A, G, B)$ is nothing but the left-right lifted product code $LP(\bar{H}_1, G, \bar{H}_2)$ with,*

$$\bar{H}_1 = \begin{bmatrix}(1+g_{a1}) & (1+g_{a2}) & ,...., & (1+g_{a\bar{n}_1})\end{bmatrix} = \begin{bmatrix}..., (1+g_{ai}), ...\end{bmatrix}$$

$$\bar{H}_2^T = \begin{bmatrix}(1+g_{b1}) & (1+g_{b2}) & ,...., & (1+g_{b\bar{n}_2})\end{bmatrix} = \begin{bmatrix}..., (1+g_{bj}), ...\end{bmatrix}$$

where here for simplicity we assume there is no degree 2 element in the set A and B, so there is no need for the technical condition of making it double cover.

Proof. Since there is only one row in $\bar{H}_1, \bar{H}_2$, the topological product base graph $\bar{\mathcal{G}}_1 \square \bar{\mathcal{G}}_2^*$ is the simplest base factor graph with only one factor node $\bar{f}_1 \times \bar{f}_2$ connected to the $(\bar{n}_1 + \bar{n}_2)$ variable nodes each of which has edge assigned by $(1 + g_{ai}), (a + g_{bj}), i \in [\bar{n}_1], j \in [\bar{n}_2]$,

respectively. By the definition of the topological lifting π^{-1}, we obtain the fact that for any factor node $(f, g) \in \mathcal{F}$, it is connected with the factor nodes of the form $(f,\ g_{ai} \cdot g)$ through the variable node (v_a, g), and connected with the factor nodes $(f,\quad g \cdot g_{bj})$ through the variable node (v_b, g). As $(g_{ai} \cdot (g \cdot g_{bj})) = ((g_{ai} \cdot g) \cdot g_{bj})$, the four factor nodes $\{(f, g),\ (f,\ g_{ai} \cdot g),\ (f,\quad g \cdot g_{bj}),\ (f,\ g_{ai} \cdot g \cdot g_{bj})\}$must form a square through the four variable nodes $(v_{ai}, g), (v_{bj}, g), (v_{ai},\ g_{ai} \cdot g), (v_{bj},\ g \cdot g_{bj})$.

Finally, by simply viewing the variable node v_{ai}, v_{bj} as A - edges and B-edges and factor node as vertices, we can see the left-right lifted product construction $LP(\bar{H}_1, G, \bar{H}_2)$ exactly gives the left-right Cayley complex as the definition shown in [14][15]. □

Corollary 1. *The Left-right Cayley complex construction is the limit of the Left-right LP construction in the sense that all information is stored in the fiber while trivial information is stored in the base, topologically speaking.*

Proof. It is shown in theorem 3 that the corresponding base parity check matrices $\bar{H}_1, \bar{H}_2$ only contain a single row. Topologically, it means that the base graph $\mathcal{G}_1, \mathcal{G}_2$ is reduced to a single factor node (not considering double cover). In this sense, the base graph is as much reduced as possible. Thus, it is why we argue that the base graph does not store the information of the code.

Additionally, it is well known that the expander code has a distance that grows linearly with the number of variable nodes. The critical point is to realize that what goes to infinity is the lifting group size l whereas the generators (corresponds to the number of columns) are shown to be constant [14]. Therefore, in this sense, we say all the information of the code is stored in the lifting group algebra elements. □

This theorem also brings the algebraic perspective for this combinatorial object, which gives us a way to implement the asymptotically good code practically [13]. However, it turns out that the good quantum Tanner code built on top of the left-right Cayley complex should be the "no-go" code for practical uses. The reasons are stated below.

Remark (Quantum Tanner code with linear distance [15] is infeasible to construct): Two obstacles are making the quantum Tanner construction less promising in practice. First, observe that each column of H_1, H_2 after algebraic lifting will have exactly two "1"s, so the topological picture $\mathcal{G}_1, \mathcal{G}_2$ of parity check matrices H_1, H_2 can be both reduced from the factor graph(hypergraph) to the classical graph category. In this case, H_1, H_2 is equal to the incidence matrix ∂ in classical graph theory [34]. However, it is well known that code from a classical graph will never have good parameters. Although we can add local code to amplify the distance, such construction is rather limited and so far, there are no explicit local codes with Robust Testability.

The second problem is because of the infeasible number of qubits used for constructing such codes. As stated in [14], the smallest possible group is the PSL_{17} which has 2448 group elements. This is even without considering the local code. Even in this case, the number of physical qubits used will be $N = l \cdot (\bar{n}^2 + \bar{m}^2) = 2448 \times (18^2 + 1) \approx 1 \times 10^6$, which is certainly not practical.

3. Advantages of the lr-LP construction

In this section, we demonstrate both the theoretical and practical advantages of the left-right Lifted product qLDPC code by examining three important non-trivial code properties, which are 1. the minimum distance of the quantum LDPC code, 2. the dominant short cycles in the factor graph, and 3. the factor graph expansion properties.

In section 3.1, we argue that under some special conditions, the finite-length distance of the left-right Lifted product qLDPC code $\pi^{-1}(LP(\bar{H}_1, G, \bar{H}_2))$ could achieve $\min(d_1, d_2)$, where d_1, d_2 are the classical distance of the codes associated with the parity check matrices $\pi^{-1}(\bar{H}_1)$ and $\pi^{-1}(\bar{H}_2)$, respectively. It is noted that while the distance is the same as the distance using the hypergraph product construction [29], the number of physical qubits N is reduced by the lifting size l compared to the HGP code. This demonstrates that given two classical codes, the left-right Lifted product code achieves the same minimum distance as the HGP construction but with much fewer physical qubits used! Hence, the lr-LP construction could give a new good quantum LDPC code given the assumption that we have two classical good LDPC codes with lifting structure and simultaneously satisfy the technical condition given in section 3.1. Moreover, motivated by the finite-length distance increase of the classical quasi-cyclic LDPC code, we conjecture that the classical non-Abelian lifting LDPC code is very likely to have both a constant rate and linear distance.

In section 3.3, we proved that given two classical codes, the number of short cycles of lr-LP codes, which is an essential obstacle for the message-passing decoder [35], is again much smaller than the number of short cycles of Hypergraph product code. This implies that the lr-LP construction probably has better decoding performance over HGP code as well. Combined with the result of the finite-length distance, it can also be concluded that both the finite-length distance and portion of short 8-cycles are preserved under the symmetry reduction, which is essentially the process from Hypergraph product to left-right Lifted product construction.

Finally, in section 3.4, based on the recent development of spectral hypergraph theory, we developed a computational lower bound for the so-called *left-expansion* property of

the targeted factor graph(code), especially for the factor graph with product structure. Furthermore, we also conduct a rigorous analysis to show how the topological product degrades the original left-expansion properties.

3.1 Finite-length distance of lr-LP qLDPC code

In this subsection, we show that given two classical codes with lifting structure, the left-right lifted product code under proper technical conditions can universally achieve the same minimum distance as hypergraph product code but with much fewer physical qubits used. Not surprisingly, the equivalence of the distances is the direct consequence of the topological product structure. The key difference is that for lr-LP code, we should work on the base complex and base matrices, which may need a few more definitions and technical conditions for the theorem to hold.

Before the proof of the theorem, we need to define the base weight $\bar{w}$ for the base matrix $\bar{H}$.

It is noticed that each base matrix is associated with a binary proto-matrix, which can also be considered as the parity check matrix of a code $\bar{C}$. Hence, we will use the distance of the dual code of $\bar{C}$ to define the base weight as follows.

Definition 13 (Base weight)**.** *Given a base matrix $\bar{H}$, assume the corresponding proto-matrix is full rank. Then, the base weight $\bar{w}$ of the $\bar{H}$ is defined to be the dual distance $d^{\perp}$ of the code $\bar{C}$ given the proto-matrix as the parity check matrix.*

The base weight $\bar{w}$ is a critical parameter for proving the logical codewords and the minimum distance of the lr-LP code. The next definition states that if the parity check matrix has a lifting(covering) structure, the classical logical codeword L can also be represented as a base vector with elements in group algebra, i.e. $\bar{L} = \pi(L)$.

Definition 14 (Base logical codeword)**.** *Given a $\bar{m} \times \bar{n}$ base parity check matrix $\bar{H}$ with a lifting structure under group G, the base logical codeword $\bar{L}$ is simply $1 \times \bar{n}$ vector with group algebra*

element $\sum_{i=0}^{|G|-1} \alpha_i g_i$, $g_i \in G$ *such that* $\bar{L} \cdot \bar{H}^T = \mathbf{0}$.

Lemma 3. *After lifting, the lifted binary logical codeword* $L = \pi^{-1}(\bar{L})$ *is indeed the actual logical codeword. This means,*

$$\bar{L} \cdot \bar{H}^T = 0 \iff \pi^{-1}(\bar{L}) \cdot \pi^{-1}(\bar{H}^T) = 0$$

Proof. Since the block matrix multiplications give the same result as matrix multiplication and the lifting map π^{-1} is an isomorphism, we have $\bar{L} \cdot \bar{H}^T = \pi^{-1}(\bar{L} \cdot \bar{H}^T) = \pi^{-1}(\bar{L}) \cdot \pi^{-1}(\bar{H}^T) = 0$. □

Example: (Base matrix and base logical codeword) Given the base parity check matrix $\bar{H}$ and base logical codeword $\bar{L}$ under group C_3, defined as follows,

$$\bar{H} = \begin{bmatrix} 1 & x \\ x & 1 \end{bmatrix}, \qquad \bar{L} = \begin{bmatrix} (1+x+x^2) & (1+x+x^2) \end{bmatrix}$$

$$\bar{L} \cdot \bar{H}^T = [(1+x+x^2) + x(1+x+x^2),\ \ x(1+x+x^2) + (1+x+x^2)]$$

$$= [(1+x+x^2) + (x+x^2+1),\ \ (x+x^2+1) + (1+x+x^2)] = \mathbf{0}$$

After lifting, the binary logical codeword becomes,

$$L = \pi(\bar{L}) = \pi\Big([(1+x+x^2), (1+x+x^2)]\Big) = [1,1,1,1,1,1]$$

One can also check this is the only logical codeword.

An essential definition of outer block weight and inner block weight, and the related lemma is necessary for the proof which are stated as follows,

Definition 15 (Inner block weight and outer block weight)**.** *Given a row vector* $\underline{v}$*, and divide it into* l_a *number of blocks, each block has length* l_b*. The inner block weight is defined as the minimum non-zero weight of each sub-inner-block of the vector at positions inner-block(i) over all* $i, i < l_a$*, where inner-block(i)* $:= \Big\{ n \in \mathbb{N} |\ \ i \cdot l_b \leqslant n < (i+1) \cdot l_b \Big\}$.

On the other hand, the outer block weight of vector $\underline{v}$ is defined as the minimum non-zero weight of the sub-outer-block of the vector at positions outer-block(i) over all $i, i < l_b$, where outer-block(i) $:= \left\{ n \in \mathbb{N} |\ \ n = i + a \cdot l_b, \forall a < l_a, a \in \mathbb{N} \right\}$.

Example: (Inner and outer block weight) We give an example of the inner and outer block weight of the vector given below, let $l_a = 3, l_b = 4$.

$$\left[0,1,1,1,\quad 0,1,1,1,\quad 0,0,0,1\right]$$

Then the inner block weight is 1 which is the weight of the third inner block. The outer block weight is 0 which is the weight of the first outer block.

Now we prove the following lemma which simply connects the base weight $\bar{w}$ of $\bar{H}$ defined in Definition 13 and the inner block(outer block) weight of the row vector in the row space of the associated proto-matrix.

For the notation and terminology, denote the two vectors with the minimum inner block (base) weight and outer block (base) weight in the row space of $\bar{H}$ to be $\underline{v}_{in}$ and $\underline{v}_{out}$, respectively. Here, we abuse notation, and call the inner block and outer block (base) weight of a matrix to refer to the inner block weight of a row vector $\underline{v}_{in}$ and the outer block weight $\underline{v}_{out}$, respectively.

Next, we will use the outer and inner blocks to give a different proof of minimum distance for the hypergraph product code. Then, we analyze and discuss in what situation the argument holds true for the general lifted product construction.

Lemma 4. *Given the weight of the matrix H being w, then the outer block base weight of $(H \otimes \bar{I})$ is at least w, and the inner block base weight of $(I \otimes H)$ is also at least w. Additionally, the base weights of both $(H \otimes I)$ and $(I \otimes H)$ are at least w as well.*

Proof. Because of the tensor product structure, the linear combination of the rows h in $(I \otimes H)$ and $(H \otimes I)$ can be decomposed as follows,

For $(I \otimes H)$:

$$h = \sum_{i}^{n-1} \delta_i \otimes \sum_{j}^{m-1} \alpha_{i,j} h_j = \delta_0 \otimes \sum_{j}^{m-1} \alpha_{0,j} h_j + ... + \delta_{n-1} \otimes \sum_{j}^{m-1} \alpha_{(n-1),j} h_j$$

By definition 13, the row vector $\delta_i \otimes \sum_j^{m-1} \alpha_{i,j} h_j, \forall i \in [n]$ has weight at least w. Since the resultant vectors do not overlap for different i, the base weight of $(I \otimes H)$ is certainly greater or equal to each term which is at least w. It is obvious to see that the inner weight is at least w as well from definition 13 since each inner block $\delta_i \otimes H$ has weight either zero or greater than w.

For $(H \otimes I)$:

$$h = \sum_{j}^{m-1} \alpha_{i,j} h_j \otimes \sum_{i}^{n-1} \delta_i = \sum_{j}^{m-1} \alpha_{0,j} h_j \otimes \delta_0 + ... + \sum_{j}^{m-1} \alpha_{n-1,j} h_j \otimes \delta_{n-1}$$

Similarly, by definition 13, the row vector $\sum_j^{m-1} \alpha_{i,j} h_j \otimes \delta_i, \forall i \in [n]$ has weight at least w. Since for different i, the result vectors do not overlap, the minimum weight of $(H \otimes I)$ is certainly greater than or equal to each term which is at least $\bar{w}$. It is again obvious to see that the outer weight is at least $\bar{w}$ as well from definition 15 because each outer block $H \otimes \delta_i$ has weight either zero or greater than w.

□

Finally, we have all the definitions and ingredients for the proof of the theorem 4 shown below which uses a different approach from [7] to prove the minimum distance of HGP code by analyzing the structure of the logical codeword.

Theorem 4 (Logical codewords of Hypergraph Product code)**.** *Given the two base parity check matrices H_1, H_2 with the base weight w_1, w_2, and for simplicity, assume its element are in the group G. Then the following L_z, L_x are the Z, X logical codewords of the left-right lifted product code $LP(H_1, G, H_2)$, respectively. The key condition is $|K_z| < w_2$ and $|K_x| < w_1$.*

$$L_z = \begin{bmatrix} \mathbf{0}_{m_1 m_2} & | L_1 \otimes \sum_{k_z \in K_z} \delta_k \end{bmatrix}, \qquad L_x = \begin{bmatrix} \mathbf{0}_{m_1 m_2} & | \sum_{k_x \in K_x} \delta_k \otimes L_2 \end{bmatrix}$$

where the set $K_z = \{k_z : k_z \in \mathbb{N}, k_z < \bar{n}_2\}$ *and* $K_x = \{k_x : k_x \in \mathbb{N}, k_x < \bar{n}_1\}$*, and* δ_k *is the discrete delta function(row vector) with "1" only in position k.*

Proof. The whole proof is divided into the two sub-proofs of the two claims as follows.

Claim 1: the above base logical operators are not logical errors, i.e. $L_z, L_x \notin Err(C)$.

By the definition of the base logical operator, we simply multiply the Z(X)-logical operator $L_z(L_x)$ with X(Z)-stabilizer generators $H_x(H_z)$.

$$L_z H_x^T = \mathbf{0} + L_1 H_1^T \otimes \sum_{k \in K} \delta_k I_{n2} = \mathbf{0} + \mathbf{0} \otimes \sum_{k \in K} \delta_k I_{n2} = \mathbf{0}$$

$$L_x H_z^T = \mathbf{0} + \sum_{k \in K} \delta_k I_{n1} \otimes L_2 H_2^T = \mathbf{0} + \sum_{k \in K} \delta_k I_{n1} \otimes \mathbf{0} = \mathbf{0}$$

Since the above operators commute with all stabilizers, they are either in the stabilizer group $S(\mathcal{Q})$ or belong to the logical operator $L(\mathcal{Q})$.

Claim 2: the logical operators L_z, L_x *are not in the stabilizers, i.e.* $L_z, L_x \notin S(\mathcal{Q})$.

We prove this by contradiction. For L_z, suppose $L_z \in S_z(\mathcal{Q})$, then there would be a $H_z = \sum_j^{m-1} \alpha_j h_j$ where h_j is a row vector in H_z (a Z-stabilizer generator), and where $\alpha_j \in \mathbb{F}_2[G]$. Then one of the necessary conditions for the equality is that H_z must have the same inner weight with L_z. Firstly, it is clear that the inner weight of L_z is chosen to be strictly less than the weight of the H_2, i.e. $|K_z| < w_2$. However, from lemma 4, we get the inner block weight of $H_z = I_{n1} \otimes H_2$ is at least w_2.

A similar proof strategy also applies to L_x. For L_x, suppose there is a X-stabilizer $H_x = \sum_j^{m-1} \alpha_j h_j$ where h_j become a row vector(X-stabilizers generator) in H_x and $\alpha_j \in \mathbb{F}_2[G]$. L_x and H_x having the same outer weight is certainly a necessary condition for equality. The outer weight of L_x is $|K_x|$ which is chosen to be strictly less than w_1. However, from lemma 4, the outer block weight of $H_x = H_1 \otimes I_{n2}$ is at least w_1.

Additionally, since we assume that the element in the matrix just be element in the group $g \in G$, this condition guarantees that each non-zero element in the row vector h_j cannot convert back to "0" unless $\alpha_j = 0$ which set the whole vector to be zero vector. Therefore, with this condition, the weight argument indeed contradicts the assumption that $L_z^2 \in S_z(\mathcal{Q}), L_x^2 \in S_x(\mathcal{Q})$. □

Corollary 2 (Unique logical codewords)**.** *For K_z, K_x such that $|K_z| < \bar{w}_2/2$, $|K_x| < \bar{w}_1/2$, the logical codewords L_z, L_x are unique which means they are not equivalent up to stabilizers.*

Proof. Since the weight of the logical operator is less than half of the corresponding outer block weight or inner block weight, the sum of the two logical operators can't be in the stabilizer group as the weight does not match. □

The next corollary is to prove the finite-length distance of the lr-LP code given the technical condition that the two parity check matrices H_1, H_2 have the weights $\bar{w}_1 \geqslant 2$ for H_1 and $\bar{w}_2 \geqslant 2$ for H_2, and the matrices are full rank. Then we have the following finite-length distance. It is noted that the following distance proof is different from the conventional proof in [7], the advantage of the following matrix base proof is that it will reveal more relations about the logical codeword and minimum distance so that we can easily extend it to the lifted product case.

Corollary 3 (Finite length distance of HGP code from examining logical codeword)**.** *Given two parity check matrices H_1, H_2 with the weights $w_1 \geqslant 2$ for H_1 and $w_2 \geqslant 2$ for H_2, The distance of the HGP code $HGP(H_1, H_2)$ are* $\min(d_1, d_2)$*. where d_1, d_2 are the classical distance of the codes associated with H_1, H_2.*

Proof. The whole proof is divided into the following three sub-proofs of the three claims.
Claim 1: The following are the two logical operators in $S_z(\mathcal{Q}), S_x(\mathcal{Q})$, respectively.

$$L_z = \begin{bmatrix} \mathbf{0}_{m_1 m_2} & |L_1 \otimes \delta_{k_z}\rangle \end{bmatrix}, \qquad L_x = \begin{bmatrix} \mathbf{0}_{m_1 m_2} & |\delta_{k_x} \otimes L_2\rangle \end{bmatrix}$$

From theorem 4, since both the set K_z, K_x has only one element, i.e. $|K_z| = |K_x| = 1 < 2$, the above are already proved to be the Z and X logical operators given the weights are all

greater than 1.

Claim 2: They are the only two candidates for the logical operators with the minimum weight.
Firstly, we need to show that adding stabilizers will not decrease the weight of the L_z, L_x. The weights are given $w_1, w_2 \geqslant 2$ for H_1, H_2. Therefore, adding Z, X base stabilizers $h = \sum_j^{m-1} \alpha_j h_j$ to the Z, X logical operators L_z, L_x will never decrease their weights.

Secondly, we need to prove that there are no other logical operators in different cosets ($L_x^{'} \neq L_x + h_x,\ L_z^{'} \neq L_z + h_z$) which have smaller weight than L_x and L_z. To prove this, observe that the only potential operators have the following form.

$$L_z = \left[\delta_{k_z} \otimes L_2^* \quad | \quad \mathbf{0}_{n_1 n_2} \right], \qquad L_x = \left[L_1^* \otimes \delta_{k_x} \quad | \quad \mathbf{0}_{n_1 n_2} \right]$$

However, we assume that the proto-matrices of H_1, H_2 are full rank and each element in the matrix is a group element which is also full rank in its algebraic lift. There is no non-zero base dual logical operators for L_1^* and L_2^*. Therefore, for each logical codeword, adding stabilizers will never decrease its weight.

□

Discussion of the minimum distance for lifted product code.
As proved in the corollary 3, the minimum distance proof of the hypergraph product code stems from the fact that the weight of the stabilizer is at least 2 so that when we attempt to cancel the weight of the logical codeword, we must add at least one on another block. Thus, the weight of the logical codeword will never be decreased. This result has an analogy for the lifted product case where the base logical codewords are shown below,

$$\bar{L}_z^1 = \left[\quad \mathbf{0}_{m_1 m_2} \quad | \bar{L}_1 \otimes \textstyle\sum_{k_{z1} \in K_{z1}} \delta_k \right], \qquad \bar{L}_x^1 = \left[\quad \mathbf{0}_{m_1 m_2} \quad | \textstyle\sum_{k_{x1} \in K_{x1}} \delta_k \otimes \bar{L}_2 \right]$$

This is almost the same for the hypergraph product case except that the element is no longer binary but in the group ring $\mathbb{F}_2[G]$. The natural question to ask is if the base weight of $\bar{H}_1, \bar{H}_2$ is no less than 2, can we use the same reasoning the prove the minimum distance bound for the lifted product construction? Unfortunately, the answer is no because in this

case, when we consider the linear combination of base stabilizers, the coefficient is no longer 0 or 1 but also in group algebra. Therefore, there is no direct result for the minimum distance bound of the general lifted product code. However, if we have a staircase-like structure with an open loop, one can show that the base weight of base parity check matrices $\bar{H}_x, \bar{H}_z$ must be greater than 2. In this case, we can use same reasoning to prove the minimum distance of lifted product code is at least $\min(d_1, d_2)$ same as the minimum distance of hypergraph product code.

Therefore, for the lifted product code to achieve the same minimum distance, one must add a special structure of base matrices or carefully choose the group algebra element enabling the base weight to be no less than two which is the critical technical condition.

Finite-length distance advantages over HGP code [29].

Given two classical codes H_1, H_2 with distance d_1, d_2, assume it satisfies the technical condition in corollary 3, the left-right lifted product construction $LP(\bar{H}_1, G, \bar{H}_2)$ achieves the same distance as the hypergraph product code $HGP(H_1, H_2)$. So far, people may think that the lr-LP construction does not offer distance advantages. However, it is important to realize that the number of physical qubits for lifted product code is $N = l \cdot (\bar{n}_1\bar{n}_2 + \bar{m}_1\bar{m}_2) = \frac{1}{l} \cdot (N_1N_2 + M_1M_2)$ while the number of physical qubits of hypergraph product code is $(N_1N_2 + M_1M_2)$. Therefore, we can conclude that the lr-LP construction achieves the same distance as the HGP construction, but with only $1/l$ portion of the physical qubits used. This is a formal demonstration of the claim that the power of the lifted product construction comes from the symmetry reduction.

3.2 Asymptotic distance and new good qLDPC codes

The following corollary naturally extends the finite-length distance in corollary 3 to the asymptotic distance result.

Corollary 4 (Asymptotic distance scale of lr-LP code)**.** *Given two good classical LDPC codes with base matrices $\bar{H}_1$ and $\bar{H}_2$ satisfying the technical condition stated in 4, the left-right lifted product construction $LP(\bar{H}_1, G, \bar{H}_2)$ has distance scales with $O(N)$.*

Proof. The whole proof is divided into three sub-proofs of the following three claims.

Claim 1: The asymptotic increase of the code size is contributed by the lifting size l rather than the base block size $\bar{n}_1$, i.e. $l = O(N_1), \bar{n}_1 = O(1)$.

It is important to mention that for the classical LDPC code with lifting structure(such as quasi-cyclic code [31]), the code size can be decomposed as $N_1 = l \cdot \bar{n}_1$. In the asymptotic context, to preserve the LDPC properties, what goes to infinity is the lifting size l rather than the base block size $\bar{n}_1$. This is because the base matrix is directly accountable for the row and column weight of the code which should be constant scale $O(1)$ by the definition of LDPC code. Hence, we can conclude that the asymptotic increase of the code size is contributed by the lifting size l instead of n, i.e. $l = O(N_1), \bar{n}_1 = O(1)$.

Claim 2: The classical distance scales linearly with lifting size l, i.e. $d = O(N) = O(l)$.

Since we assume that we already have two good classical codes corresponding to the base parity check matrix $\bar{H}_1, \bar{H}_2$ with their distance scales linearly with code sizes, i.e. $d_1 = O(N_1)$ and $d_2 = O(N_2)$. Also, the proof of the first claim implies that the distance also scales linearly with the lifting size l, i.e. $d_1 = O(N_1) = O(l)$ and $d_2 = O(N_2) = O(l)$.

Claim 3: The lr-LP construction brings a new good quantum LDPC code $LP(\bar{H}_1, G, \bar{H}_2)$.

For simplicity, we assume the H_1, H_2 are full rank, by corollary 3, we obtain the distance of the code $LP(\bar{H}_1, G, \bar{H}_2)$ is $\min(d_1, d_2)$. The code size of the quantum code is

$N = (\bar{n}_1\bar{n}_2 + \bar{m}_1\bar{m}_2) \cdot l = (N_1N_2 + M_1M_2)/l$. Since the lifting size l scales linearly with the classical code size $l = O(N_1) = O(N_2)$ from the proof of claim 2. Thus, we have the distance $d = O(N_1) = O(N_2) = O(l)$, and the code size N of $LP(\bar{H}_1, G, \bar{H}_2)$ scales linearly with lifting size as well, i.e. $N = O(N_1) = O(N_2) = O(l)$. Therefore, we prove the linear scale of the distance for the quantum LDPC code, i.e. $d = O(N)$. □

Corollary 13 generalizes the quantum Cayley expander [14].

A natural but important question after the corollary 4 is "Does such good classical code with lifting structure already exist?". The answer is yes, but not good enough for practical construction as discussed in section 2.3. The well-known example is indeed the classical Tanner code which is a very special case of such good classical code with lifting structures. Not surprisingly, the lr-LP construction with two classical expanders is indeed the well-known left-right Cayley complex as stated in the theorem 3.

Advantages over the quantum Tanner code [13][15].

The expander construction is not very useful for practical construction as discussed in section 2.3. We should emphasize again that the main reason is that we restrict ourselves to the classical graph theory so that all the good mathematical tools from graph theory(mainly the spectral graph theory) can then be utilized to prove the asymptotic distance bound. However, constructing good code(factor graph) from the classical graph is rather limited, and is unlikely to work well in practice.

However, our construction is not limited to the expander constructions since the conjecture states that under mild technical conditions, for any good classical LDPC code with a lifting structure, the left-right lifted product code will always yield good quantum LDPC code. Therefore, there is no reason to stick to the graph category, and we are free to design code in the category of the factor graph(hypergraph), which almost certainly yields better finite-length code parameters than using the expander code construction.

From QC LDPC code to the classical non-Abelian lifting code.

The simple but popular classical LDPC code with a lifting structure is quasi-cyclic LDPC code. It is then natural to ask "Can the QC LDPC construction yield asymptotically good code?". Unfortunately, the answer is no. [25] states that the QC LDPC code has a minimum distance bounded above by the number of rows in the base matrix $\bar{m}$, i.e. $d \leqslant (\bar{m}+1)!$. This implies that the classical distance d of QC LDPC code cannot grow with the lifting size l. The consequence is that the classical quasi-cyclic LDPC code cannot have asymptotic good distance. Thus, it is impossible to take the left-right lifted product construction under the cyclic group to get the corresponding quantum LDPC code that has an asymptotic good distance. In [12], this upper bound is also generalized to the case of Abelian group lifting.

However, there is no such limit if the group being lifted is non-Abelian as the classical expander graph is just the case. It is well known that there does not exist an expander graph under Abelian lift, but for some non-Abelian groups there even exists explicit construction for the expander graph [14]. Therefore, it is almost certainly true that the good classical code with a lifting structure exists if we instead consider the non-Abelian lifted structure. In this sense, the asymptotic distance result in corollary 4 can also be considered as a motivation for the study of the classical non-Abelian lifting LDPC code.

The above discussion leads to the following conjecture for the classical asymptotically good non-Abelian lifting code.

Conjecture 1. *There exists a non-Abelian group G such that for a class of randomly selected(randomly select elements from group algebra $\mathbb{F}_2[G]$) $\bar{m} \times \bar{n}$ base matrices $\{\bar{H}\}$ with $\bar{m}, \bar{n}$ bounded above, the class of the code associated with the lifted parity check matrices $\{H\}$ has asymptotically good distance with high probability as the lifting size $l \to \infty$.*

There is already plenty of evidence for the above conjecture. The evidence includes that a random LDPC code has a linear distance with high probability proved in [3], and the random lifting almost yields a Ramanujan graph. Since we are standing directly in the Hy-

pergraph category, presumably the random lifting of the factor graph should give a better expansion as the factor node degree can be greater than two. The only restriction we have to enforce is the group structure for the left-right Lifted product, which means the random element chosen has to come from the same group algebra $\mathbb{F}_2[G]$. However, it is realized that the explicit expander construction also uses the non-Abelian group lifting, so the condition should not be problematic.

Concrete examples of the class of finite-length lr-LP codes.

It is also worth mentioning that even though the distance does not scale linearly for abelian lifting, the cyclic lifted product code could still become a perfect candidate for short to medium finite-length qLDPC code due to its construction simplicity and the fact that it has already been well-studied in classical coding theory. Indeed, in [16], a few finite-length qLDPC codes obtained by taking the lifted product of two classical QC LDPC codes are given.

In [25], several concrete and small (3,4)-regular QC LDPC codes are given as follows with different lifting sizes l.

$$H_1 = \begin{bmatrix} x & x^2 & x^4 & x^8 \\ x^5 & x^{10} & x^{20} & x^9 \\ x^{25} & x^9 & x^7 & x^{14} \end{bmatrix}, \quad H_2 = \begin{bmatrix} x^1 & x^2 & 0 & 0 & x^4 & 0 & x^8 & 0 \\ x^2 & x^0 & 0 & 0 & 0 & x^4 & 0 & x^8 \\ x^5 & 0 & x^9 & 0 & x^{10} & x^{20} & 0 & 0 \\ 0 & x^5 & 0 & x^9 & x^{20} & x^{10} & 0 & 0 \\ 0 & 0 & x^{25} & x^{19} & 0 & 0 & x^7 & x^{14} \\ 0 & 0 & x^{19} & x^{25} & 0 & 0 & x^{14} & x^7 \end{bmatrix}$$

Based on the result in [25], the distances given different lifting size l are shown below,

Table 3.1: Finite-length distance

l	1	27	28	29	31
d	2	14	16	18	24

We should notice that the proto-matrix of $\bar{H}_1$ is not full rank which does not satisfy the technical condition for the finite-length distance. However, that technical condition can be further relaxed, and we are going to look for a weaker condition in future work.

3.3 Girth and dominant cycles

In this subsection, we show another advantage of the lr-LP qLDPC code. Specifically, given two classical codes with lifting structure, the number of short 8-cycles by lr-LP qLDPC construction is significantly reduced compared to the hypergraph product construction. This also can be considered as one of the evidence of the practical advantages of using lr-LP construction over hypergraph product construction.

Assume two factor graphs $\mathcal{G}_1, \mathcal{G}_2$ represent two classical codes C_1, C_2 with code parameters $[N_1, M_1, d_1]$ and $[N_2, M_2, d_2]$. Those two codes have the lifting structure with lifting size l under a group G. For simplicity, we also assume that both base factor graphs $\bar{\mathcal{G}}_1 = \mathcal{G}_1/G,\ \bar{\mathcal{G}}_2 = \mathcal{G}_2/G$ have girth greater than 8, and the group algebra being assigned only has single term(reduced to the group element). It is noted that this girth assumption can be greatly relaxed, it is assumed just to simplify the equation and the analysis.

Theorem 5 (Dominant 8-cycles in the lr-LP qLPDC code)**.** *Given the above condition, the factor graph $\mathcal{G}_z$ and $\mathcal{G}_x$ obtained by left-right lifted product construction $\pi^{-1}\left(\bar{\mathcal{G}}_1 \times \bar{\mathcal{G}}_2\right)$ both have girth 8. The number of dominant 8-cycles for the two factor graphs $\mathcal{G}_z, \mathcal{G}_x$ of the left-right lifted product code are,*

$$\#\ \textit{8-cycles}\ \ (G_z) = l \cdot \sum_{i=0}^{\bar{n}_2-1} \sum_{j=0}^{\bar{m}_1-1} \binom{\bar{d}_{c_{1j}}}{2} \binom{\bar{d}_{v_{2i}}}{2}.$$

$$\#\ \textit{8-cycles}\ \ (G_x) = l \cdot \sum_{i=0}^{\bar{n}_1-1} \sum_{j=0}^{\bar{m}_2-1} \binom{\bar{d}_{c_{2j}}}{2} \binom{\bar{d}_{v_{1i}}}{2}.$$

where v_{1i}, c_{1j} and v_{2i}, c_{2j} refer to the i^{th} variable node and the j^{th} check node of the base graph $\bar{\mathcal{G}}_1, \bar{\mathcal{G}}_2$, respectively.

The following lemma is critical for the proof of theorem 5, which simply counts the number of 8-cycles in the Cartesian product of two-factor graphs.

Lemma 5 (Cycles in the Cartesian product of factor graphs)**.** *Given two factor graphs with the number of variable node n_1, n_2. For simplicity, assume the girth of both factor graphs is larger than 8. Denote the i^{th} variable degree of factor graph $\mathcal{G}_1$ to be $d_{v_{1i}}$, and $d_{v_{2i}}$ for factor graph $\mathcal{G}_2$. Then, the Cartesian product of these two factor graphs(defined in 7) $\mathcal{G} = \mathcal{G}_1 \square \mathcal{G}_2$ has 8-cycles,*

$$\#\ \textit{8-cycles}\ \ (G) = \sum_{i=0}^{n_2-1} \sum_{j=0}^{n_1-1} \binom{d_{v_{2i}}}{2} \binom{d_{v_{1j}}}{2}.$$

Proof. The whole proof proceeds with the sub-proofs of the two claims as follows,

Claim 1: The number of 8-cycles in $\sum_{i=0}^{n_2-1} \sum_{j=0}^{n_1-1} \binom{d_{v_{2i}}}{2}\binom{d_{v_{1j}}}{2}$ if $\mathcal{G}_1, \mathcal{G}_2$ are tree factor graphs.

Suppose $\mathcal{G}_1, \mathcal{G}_2$ are tree factor graphs which means we ignore the existing cycles in $\mathcal{G}_1, \mathcal{G}_2$ at first. Then, by the definition of factor graph Cartesian product in 7, every pair of edges in $\mathcal{E}_1, \mathcal{E}_2$ connected to a single variable node $v_i \in \mathcal{V}_1, v_j \in \mathcal{V}_2$ form a 8-cycles in the product graph $\mathcal{G}$. Thus, $\mathcal{G}_1 \square \mathcal{G}_2$ has at least $\sum_{i=0}^{n_2-1} \sum_{j=0}^{n_1-1} \binom{d_{v_{2i}}}{2}\binom{d_{v_{1j}}}{2}$ number of 8-cycles.

Claim 2: The cycles in $\mathcal{G}_1, \mathcal{G}_2$ do not split into sub-cycles in the $\mathcal{G}_1 \ \square\ \mathcal{G}_2$.

Without loss of generality, suppose the cycle in $\mathcal{G}_1$ splits(same for $\mathcal{G}_2$) into small cycles in $\mathcal{G}_1 \square \mathcal{G}_2$, then there would exist two new edges e connecting to the factor node $f_i \in \mathcal{F}_1$ and $f_j \in \mathcal{F}_1$ through a new variable node v. Based on the definition 7, there are two possibilities for the new variable nodes - one is that the new variable node v is from $\mathcal{V}_1 \times \mathcal{F}_2$, the other is that the variable node v is from $\mathcal{F}_1 \times \mathcal{V}_2$. Since the edge set $\mathcal{E}$ can only connect to two nodes with same index from definition 7(i.e. either (v_1, f_1) or (v_2, f_2), v can only be in the set $\mathcal{V}_1 \times \mathcal{F}_2$, then consequently e can only in the set $\mathcal{E}_1 \times \mathcal{F}_2$ which contradicts the fact that $e \notin \mathcal{E}_1$.

Therefore, we prove that the number of the dominant 8-cycle of the product graph $\mathcal{G}_1 \ \square\ \mathcal{G}_2$ is exactly $\sum_{i=0}^{n_2-1} \sum_{j=0}^{n_1-1} \binom{d_{v_{2i}}}{2}\binom{d_{v_{1j}}}{2}$ and the girth is 8 as well.

□

Now, we are ready to prove the theorem 5. Again, for simplicity, assume two base factor graphs $\mathcal{G}_1, \mathcal{G}_2$ have girth larger than eight, and the group algebra element assigned to each edge has only one term. The second technical condition is just to make sure there are no double edges in the base factor graph.

It is also worth mentioning that the two above technical conditions are made just to simplify the proof and analysis, but both can be relaxed in real practical construction.

Proof. The whole proof is divided into the following two sub-proofs.

Claim 1: The base factor graph $\bar{\mathcal{G}}_z$ and $\bar{\mathcal{G}}_z$ have 8-cycles $\sum_{i=0}^{\bar{n}_2-1}\sum_{j=0}^{\bar{m}_1-1}\binom{\bar{d}_{c_{1j}}}{2}\binom{\bar{d}_{v_{2i}}}{2}$ and $\sum_{i=0}^{\bar{n}_1-1}\sum_{j=0}^{\bar{m}_2-1}\binom{\bar{d}_{c_{2j}}}{2}\binom{\bar{d}_{v_{1i}}}{2}$, respectively.

From the topological definition of lr-LP code 11, we know that the two base factor graph $\bar{\mathcal{G}}_z, \bar{\mathcal{G}}_z$ comes from the Cartesian product of factor graphs as stated in the definition 7, More specifically, $\bar{\mathcal{G}}_z = \bar{\mathcal{G}}_1^* \square \bar{\mathcal{G}}_2$ and the other base factor graph $\mathcal{G}_x = \bar{\mathcal{G}}_1 \square \bar{\mathcal{G}}_2^*$. Thus, the lemma 5 can be utilized, and it implies that the number of 8-cycles in $\bar{\mathcal{G}}_z$, $\bar{\mathcal{G}}_x$ are exactly the given number in claim 1.

Claim 2: After lifting, the 8-cycles in the factor graph $\mathcal{G}_z, \mathcal{G}_x$ are exact l times of the 8-cycles in the base graph $\bar{\mathcal{G}}_z, \bar{\mathcal{G}}_x$.

To see what lift affects the number of 8-cycles, we switch back to the matrix languages. Based on the previous result of cycle design and analysis in classical QC LDPC code [38][39], we know that for each 8-cycle in the base graph, the number of 8-cycles being lifted is at most l. For cyclic lift, the cycle length being lifted is related to the permutation shift [39] as the equation shown on the left, which can also be easily generalized to the

non-Abelian case as the equation shown on the right.

$$\textit{Cyclic: } P_w = \sum_{i=0}^{7} (-1)^i p_{i+1} \mod l \;\Rightarrow\; \textit{Any group: } P_w = \prod_{i=0}^{7} g_{i+1}^{(-1)^i} \mod l$$

Since the graph product has a very special structure - every edge exactly appears twice to form 8-cycles. Hence, the alternating sum above is all canceled out, and the $P_w = e$. This implies that the 8-cycles in the base graph are inevitable in the lifted factor graph. Therefore, the number of 8-cycles in the factor graph is exactly l times larger than the 8-cycles in the base graph.

□

Corollary 5. *Theorem 5 works for general factor graphs. If we assume the corresponding two base factor graphs $\bar{\mathcal{G}}_1, \bar{\mathcal{G}}_2$ are $(\bar{d}_{v_1}, \bar{d}_{c_1})$, and $(\bar{d}_{v_2}, \bar{d}_{c_2})$-regular, then the theorem 5 can be simplified as follows,*

$$\#\textit{ 8-cycles } (G_z) = l \cdot \binom{\bar{d}_{c_1}}{2} \binom{\bar{d}_{v_2}}{2} \bar{m}_1 \bar{n}_2.$$

$$\#\textit{ 8-cycles } (G_x) = l \cdot \binom{\bar{d}_{v_1}}{2} \binom{\bar{d}_{c_2}}{2} \bar{n}_1 \bar{m}_2.$$

Corollary 6. *Asymptotically, the number of dominant 8-cycle for hypergraph product construction grows $\Theta(N_1^2)$ while for the left-right lifted product construction, the dominant 8-cycle grows $\Theta(N_1)$.*

Proof. Following from theorem 5, it is known that if the lifting size $l = 1$ the lr-P construction is reduced to the hypergraph construction as discussed in section 2.1. Thus, for LDPC code, the number of 8-cycles will become $c \cdot M_1 N_2$ and $c \cdot N_1 M_2$, where c is a constant, for factor graph G_z and G_x. Asymptotically, as both N_1, M_1, N_2, M_2 grow in the same order, the number of 8-cycles for hypergraph product construction scales $\Theta(N_1^2)$.

For lifted product code, since $N_1 = \bar{n}_1 \cdot l$, $N_2 = \bar{n}_2 \cdot l$ and we know $\bar{n}_1, \bar{n}_2$ are constant $\Theta(1)$, the lifting size l scales linearly $\Theta(N_1)$. Hence, by theorem 5, the number of dominant 8-cycles is $c \cdot l$ for both factor graphs G_z and G_x, which means the number of 8-cycle scales with $\Theta(N_1)$. □

This implies that given the same classical codes with lifting structure, the lifted product construction gives the quadratic advantage in terms of the dominant 8-cycles over the hypergraph product construction.

The portion of the 8-cycles is preserved under symmetry reduction.

It should be clear that even though the number of dominant 8-cycles is significantly reduced by using lifted product code compared to the hypergraph product construction. The ratio of the number of 8-cycles over the number of physical qubits N is the same for both the left-right lifted product and the hypergraph product constructions. Since the number of qubits for hypergraph product code is $N = N_1N_2 + M_1M_2$, both constructions have the number of 8-cycles grows with $\Theta(N)$.

However, like the case of the code distance, although the lifted product significantly reduces the number of physical qubits by symmetry reduction, the code distance and the number of dominant short 8-cycles are both preserved. In other words, the symmetry reduction has no negative effect on the two properties, and in this sense, the physical qubits being quotiented out are purely redundant.

Implication for message-passing decoding.

It is well known that the short cycle, especially the large number of dominant short cycles can significantly affect the decoding performance under a message-passing decoder such as the Belief Propagation decoder. Additionally, it is stated in [35] that the short 4-cycle is inevitable in the X-Z factor graph as there must be an even overlap of every X and Z stabilizers. Additionally, for practical good decoding performance, the girth

should be at least 6 [40]. The negative result may imply that it might not give us the real decoding advantages of using Belief propagation on X-Z factor graphs [19] without further complicating the algorithm complexity.

However, the good news is that for independent X or Z factor graphs, we can get rid of any short cycles below 8. The only issue is that the product structure leads to a large number of 8-cycles which may still be problematic. Nevertheless, so far, the left-right lifted product construction has the least number of 8-cycles among all high-rate qLDPC constructions which also makes it a perfect candidate for the practical construction and decoding of the qLDPC code.

3.4 Factor graph expansion

In this subsection, another important code properties - the expansion properties - are rigorously examined for the lr-LP construction. It is well known that high expansion is a necessary condition for obtaining many theoretical and practical good parameters of the code including the code asymptotic distance [41], efficient low-complexity decoder [22], and the single-shot syndrome measurement [42][43].

However, as introduced in section 2.3, there are a few problems related to the expansion properties of the code. The first one is the NP-hardness of computing the exact Cheeger constant which is the general quantity for defining the expansion properties [20]. Although it is well known that the spectral graph theory could be utilized to bound the Cheeger constant, it cannot be directly applied to the general code as the general factor graphs associated with codes are almost always sitting in the hypergraph category. Also, it has been proved that the code in the classical graph category cannot have good parameters [44]. Furthermore, the left-right Cayley expander as discussed in section 2.3 is not practical code construction.

Therefore, in this subsection, we aim to provide a computational tool for the expansion and connectivity of the factor graph based on the generalized Cheeger constant developed recently in [21]. It is noted that such generalized Cheeger constant is developed for uniform hypergraph [21], which corresponds to the factor graph with regular check degree. Moreover, built on top of the newly developed mathematical tool of the spectral hypergraph theory, we give a hypergraph spectral analysis for the expansion properties of any high-rate qLDPC codes with product structure. In this case, there is no reason to restrict ourselves to the graph category to construct the expander graph first and add local codes. Thus, this new tool should enable us to examine and construct high-expansion code with more flexibility than the conventional classical and quantum Tanner construction [45][15].

Definition 16 (Generalized Cheeger Constant). *[21] Given a k-uniform hypergraph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*, the generalized Cheeger constant is defined as follows,*

$$h(S) := \min_{\emptyset \neq S \subsetneq V} \frac{E(S, \bar{S})}{\min\{vol(S), vol(\bar{S})\}} = \min_{\emptyset \neq S \subsetneq V} \frac{\sum_{r=1}^{k-1} |E_r(S)| r(k-r)}{\min\{vol(S), vol(\bar{S})\}}$$

where the $E_r(S)$ is defined in the definition 2.7 in [21], and $vol(S)$ denote the sum of vertex degrees in S. One can also check that the quantity $\sum_{r=1}^{k-1} |E_r(S)| r(k-r)$ counts exactly the pairwise connections between set S and $\bar{S}$.

As the mathematical tool was originally developed for the oriented hypergraph in [21], we need to first map the important definitions to the language of the undirected factor graphs for later analysis. The conventional Spectral graph theory only cares about the vertex adjacency matrix and Laplacian.

However, it is important to realize that for a factor graph(Hypergraph), we should define so-called both variable Laplacian and check Laplacian since hyperedges(variable nodes) can have degrees larger than two for a hypergraph(factor graph). Moreover, based on the topological definition of the lifted product construction, the variable(check) Laplacian of the dual graph is the check(variable) Laplacian of the original graph, which plays a role in computing the expansion properties for the factor graph $\mathcal{G}_x$ and $\mathcal{G}_z$.

Definition 17 (Variable Cheeger constant and Check Cheeger constant)**.** *For a (d_v, d_c)-regular factor graph $\mathcal{G} = (\mathcal{V}, \mathcal{F}, \mathcal{E})$ with n variable nodes and m factor nodes, the corresponding variable and check node Cheeger constant h_v and h_c of $\mathcal{G}$ are,*

$$h_v(V) := \min_{\emptyset \neq V \subsetneq \mathcal{V}} \frac{E(V, \bar{V})}{d_v \min\{|V|, |\bar{V}|\}}, \quad h_c(V) := \min_{\emptyset \neq F \subsetneq \mathcal{F}} \frac{E(F, \bar{F})}{d_c \min\{|F|, |\bar{F}|\}}$$

Definition 18 (Normalized variable Laplacian and check Laplacian)**.** *See definition 2.5 in [21]. Given a factor graph $\mathcal{G} = (\mathcal{V}, \mathcal{F}, \mathcal{E})$, the degree matrix of factor graph $\mathcal{G}$ is the $n \times n$ diagonal matrix for variable nodes and is $m \times m$ diagonal matrix for factor nodes.*

$$D_v(\mathcal{G}) = diag\Big(d_{v_1}, d_{v_2}, ..., d_{v_n}\Big), \quad D_c(\mathcal{G}) = diag\Big(d_{c_1}, d_{c_2}, ..., d_{c_n}\Big)$$

The variable adjacency matrix of factor graph $\mathcal{G}$ is the $n \times n$ matrix $A_v(\mathcal{G}) := A_{ij}$, where $A_{ii} = 0$ for each $i \in [n]$ and for $i \neq j$, $A_{ij} := \left\{ |a| : f_a \in \{\partial v_i \cap \partial v_j\} \right\}$ which is equivalent to the number of connected edges in $\mathcal{E}$ between variable node v_i and v_j. The same definition goes for check adjacency matrix $A_c(\mathcal{G})$ simply by swapping variable nodes and factor nodes appearing in the above definition of the variable adjacency matrix.

Finally, the normalized variable Laplacian $L_v(\mathcal{G})$ is the $n \times n$ matrix, and the normalized check Laplacian $L_c(\mathcal{G})$ is the $m \times m$ matrix instead,

$$L_v(\mathcal{G}) := I_n + D_v^{-1} A_v, \quad L_c(\mathcal{G}) := I_m + D_c^{-1} A_c,$$

where in classical graph theory, L usually is named signless normalized Laplacian. In graph theory, the more commonly used un-normalized Laplacian is defined $\mathcal{L} := D - A$ instead.

Lemma 6 (Relation between the eigenvalues of L_v and L_c)**.** *Assume the given factor graph to be (d_v, d_c)-regular. Denote the second largest eigenvalue of L_v, L_c to be Λ_{n-1}^v and Λ_{m-1}^c, respectively. Then they have the following relations,*

$$\Lambda_{n-1}^v = 2(1 - \frac{d_c}{d_v}) + \frac{d_c}{d_v} \Lambda_{m-1}^c = 2(1-c) + c \cdot \Lambda_{m-1}^c$$

where $c = \frac{d_c}{d_v}$ for simplifying the notation.

Proof. By definition 18, $L_v(\mathcal{G}) := I_n + D_v^{-1} A_v = D_v^{-1}(2D_v - (D_v - A_v)) = D_v^{-1}(2D_v - \mathcal{L}_v)$. For simplicity, we assume the factor graph to be regular, then $L_v(\mathcal{G}) = 2 - \frac{1}{d_v}\mathcal{L}_v$. Hence, the second largest eigenvalue of the variable normalized Laplacian Λ_{n-1}^v can be represented as $\Lambda_{n-1}^v = 2 - \frac{1}{d_v}\lambda_2^v$, where λ_2^v is the second smallest eigenvalue of the un-normalized variable Laplacian $\mathcal{L}_v$. Similarly, for the normalized check Laplacian defined in 18, its second largest eigenvalue $\Lambda_{m-1}^c = 2 - \frac{1}{d_c}\lambda_2^c$.

Since it is well known in graph theory that the Laplacian can also be obtained by the boundary operator(sometimes called incidence matrix of a graph) ∂, i.e. $\mathcal{L}_v = \partial \cdot \partial^T$,

this result can be directly generalized to our factor graph(hypergraph) case, i.e. $\mathcal{L}_v = H \cdot H^T, \mathcal{L}_c = H^T \cdot H$, where H is just the parity check matrix associated with the factor graph(one can also verify this is exactly the incidence matrix of the corresponding hypergraph from hyperedges to vertices). Since the non-zero eigenvalue of $H \cdot H^T$ and $H^T \cdot H$ are equal, we obtain $\lambda_2^v = \lambda_2^c$.

Therefore, simply by replacing λ_2^v with λ_2^c in $\Lambda_{n-1}^v = 2 - \frac{1}{d_v}\lambda_2^v$, we have,

$$\Lambda_{n-1}^v = 2 - \frac{1}{d_v}\lambda_2^v = 2 - \frac{1}{d_v}\lambda_2^c = 2 - \frac{1}{d_v}d_c(2 - \Lambda_{m-1}^c) = 2(1 - \frac{d_c}{d_v}) + \frac{d_c}{d_v}\Lambda_{m-1}^c$$

□

Based on the generalized Cheeger inequality in theorem 3.1 in [21], we can correspondingly define the check and variable Cheeger inequality for the factor graph.

Theorem 6 (Variable and check node Cheeger Inequality)**.** *See Theorem 3.1 in [21]. Let (d_v, d_c)-regular factor graph $\mathcal{G}$ be connected factor graph. Then, the variable Cheeger inequality is,*

$$\frac{d_c - \Lambda_{n-1}^v}{2(d_c - 1)} \leqslant h_v \leqslant \sqrt{2(d_c - 1)(d_c - \Lambda_{n-1}^v)}$$

Similarly, the check Cheeger inequality is,

$$\frac{d_v - \Lambda_{n-1}^c}{2(d_v - 1)} \leqslant h_c \leqslant \sqrt{2(d_v - 1)(d_v - \Lambda_{n-1}^c)}$$

It is well known that the high expansion could offer many good properties for codes. However, there is some vagueness for the term "expansion" in the context of coding theory, and we should state clearly what the exact "expansion" people should look for in better code design and engineering. The following so-called *left-expansion* and the corresponding left expander are exactly the "expansion" which benefits many code properties [46].

Definition 19 (Left expander). *[46] A $(n, m, d_v, \gamma, \alpha)$ bipartite expander is a d_v-left-regular bipartite graph $G(L \cup R, E)$ where $|L| = n$ and $|R| = m$ such that $\forall S \subseteq L$ with $|S| \leqslant \gamma n$, $|\partial(S)| \geqslant \alpha|S|$.*

For simplifying the terminologies, the terminology "the left-expansion parameter" which is used in the later proof refers to the parameter α in the definition 19. The following theorem shows that the left expansion and the left expansion are the right "expansion" properties for code design and engineering.

Theorem 7 (Left expander offer good code and decoder). *[46]. Let $\mathcal{G}$ be a $(n, m, d_v, \gamma, d_v(1-\epsilon))$ expander. Then the minimum distance $d \geqslant \hat{d} = 2\gamma(1-\epsilon)n$. Also, there exists a low complexity flip algorithm that is guaranteed to successfully decode errors $\underline{e}$ with weight less than $\gamma(1-2\epsilon)n$.*

As it has been proved in [46], theorem 7 says that the left expansion is what people need for good code parameters and the low-complexity decoding algorithm. However, not surprisingly, it is presumably NP-hard to compute the exact left-expansion parameter α given γ like the NP-hardness of computing the graph Cheeger constant [20]. However, since there is already a mathematical tool developed for computing the bound of the variable and check the node Cheeger constant of a factor graph, the following lemma is proved to imply that such a computational tool can also utilized to bound the left-expansion by connecting the factor graph Cheeger constant with the left-expansion parameter α.

Lemma 7. *Given the generalized variable Cheeger constant $h_v(\mathcal{G})$ of a regular factor graph $\mathcal{G}$, such (d_v, d_c)-regular factor graph is then a $(n, m, d_v, 0.5, \lceil \frac{d_v h_v}{d_c - 1} \rceil)$ bipartite expander.*

Proof. By definition 16, the numerator of the variable Cheeger constant $h_v(V)$ counts exactly the number of pairwise connections between $V, \bar{V}$. Given the fact that the factor graph is regular (d_c is constant), the size of the factor nodes at the boundary of the variable node set V(denoted by $|\partial V|$) is then also lower bounded by as follows,

$$|\partial V| \geqslant \lceil \frac{\sum_{r=1}^{d_c-1} |E_r(V)| r(d_c - r)}{d_c - 1} \rceil$$

where the denominator being $d_c - 1$ is because of the simple $V \nsubseteq \bar{V}$ which means the node in set V has to be excluded.

Without loss of generality, assume the size $|V| \leqslant |\bar{V}|$, then the above inequality can be replaced by the following,

$$|\partial V| \geqslant \lceil \frac{h(V)}{d_c - 1} \rceil \cdot d_v |V|$$

Thus, by definition 16, it is equivalent to say $\forall V \subseteq L$ with $|V| \leqslant 0.5n$, $|\partial(V)| \geqslant \lceil \frac{d_v}{d_c - 1} \cdot h_v(\mathcal{G}) \rceil |V|$. By definition 19, we prove the lemma. □

So far, we have shown that the generalized Cheeger constant and its hypergraph spectral bound developed in [21] can be converted into the bound of the left-expansion. Next, as almost all of the high-rate qLDPC codes exhibit a product structure, it is then natural to extend the hypergraph spectral bound to the product factor graph. The following lemma shows how the second-largest eigenvalue of the normalized Laplacian of the product factor graph is derived from its components.

Lemma 8 (Second largest eigenvalue Λ^v_{n-1} of the product factor graphs)**.** *Given two regular factor graphs $\mathcal{G}_1, \mathcal{G}_2$ with degree parameter (d_{v1}, d_{c1}) and (d_{v2}, d_{c2}), respectively. For the Cartesian product of these two factor graphs, the second largest eigenvalue of the normalized variable Laplacian defined in 18 $\Lambda^v_{n-1}(\mathcal{G}_1 \square \mathcal{G}_2)$ is no smaller than that of its two components.*

$$\Lambda^v_{n-1}(\mathcal{G}_1 \square \mathcal{G}_2) = \max\left\{ \Lambda^v_{n-1}(\mathcal{G}_1),\ \Lambda^v_{n-1}(\mathcal{G}_2) \right\}$$

Proof. Let A_{c1}, A_{c2} denote the check adjacency matrix of the factor graph $\mathcal{G}_1, \mathcal{G}_2$, respectively. Also, let D_{c1}, D_{c2} be the check degree matrix of the graph $\mathcal{G}_1, \mathcal{G}_2$. Then their check Laplacian operators $\mathcal{L}_{c1}, \mathcal{L}_{c2}$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ are $\mathcal{L}_{c1} = D_{c1} - A_{c1}$, $\mathcal{L}_{c2} = D_{c2} - A_{c2}$. From the definition of the factor graph Cartesian product 7, the factor node $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2$. Hence, we can easily derive the following check degree matrix D_c and check adjacency matrix A_c for product factor graph $(\mathcal{G}_1 \ \square \ \mathcal{G}_2)$ as follows.

$$A_c(\mathcal{G}_1 \square \mathcal{G}_2) = A_{c1} \otimes I + I \otimes A_{c2}, \quad D_c(\mathcal{G}_1 \square \mathcal{G}_2) = D_{c1} \otimes I + I \otimes D_{c2}$$

Thus, the un-normalized check Laplacian of the product graph $\mathcal{L}_c(\mathcal{G}_1 \square \mathcal{G}_2)$ is,

$$\mathcal{L}_c(\mathcal{G}_1 \square \mathcal{G}_2) = D_c(\mathcal{G}_1 \square \mathcal{G}_2) - A_c(\mathcal{G}_1 \square \mathcal{G}_2)$$

$$= (D_1 \otimes I + I \otimes D_2) - (A_1 \otimes I + I \otimes A_2)$$

$$= (D_1 \otimes I - A_1 \otimes I) + (I \times D_2 - I \otimes A_2)$$

$$= (D_1 - A_1) \otimes I - I \otimes (D_2 - A_2)$$

$$= \mathcal{L}_{c1} \otimes I + I \otimes \mathcal{L}_{c2}$$

It is then obvious that the $\mathcal{L}_c(\mathcal{G}_1 \square \mathcal{G}_2) = \mathcal{L}_{c1} \otimes I + I \otimes \mathcal{L}_{c2}$ have the spectrum as follows,

$$\lambda^c_{ij}(\mathcal{G}_1 \square \mathcal{G}_2) = \lambda^c_i(\mathcal{G}_1) + \lambda^c_j(\mathcal{G}_2)$$

Then the second smallest eigenvalue $\lambda^c_2(\mathcal{G}_1 \square \mathcal{G}_2) = \min\left\{\lambda^c_2(\mathcal{G}_1), \lambda^c_2(\mathcal{G}_2)\right\}$.

To map this result to the normalized variable Laplacian of the product factor graphs, we perform the following substitution,

By definition 18, the normalized Laplacian for the product factor graph is,

$$L_v(\mathcal{G}_1 \square \mathcal{G}_2) = I_N + D_v^{-1}(\mathcal{G}_1 \square \mathcal{G}_2) \cdot A_v(\mathcal{G}_1 \square \mathcal{G}_2)$$

where $N = n_1 m_2 + m_1 n_2$. To represent $L_v(\mathcal{G}_1 \square \mathcal{G}_2)$ using un-normalized Laplacian $\mathcal{L}_v(\mathcal{G}_1 \square \mathcal{G}_2)$, we have the following equality(see the proof of the lemma 3.4),

$$L_v(\mathcal{G}_1 \ \square \ \mathcal{G}_2) = 2I_N - D_v^{-1}(\mathcal{G}_1 \square \mathcal{G}_2) \cdot \mathcal{L}_v(\mathcal{G}_1 \square \mathcal{G}_2)$$

$$= 2I_N - D_v^{-1}(\mathcal{G}_1 \square \mathcal{G}_2) \cdot \mathcal{L}_c(\mathcal{G}_1 \square \mathcal{G}_2)$$

Then since we assume that factor graphs are regular, its second-largest eigenvalue $\Lambda^v_{n-1}(\mathcal{G}_1 \square \mathcal{G}_2)$ is

$$\Lambda^v_{n-1}(\mathcal{G}_1 \square \mathcal{G}_2) = 2 - \frac{1}{d_v} \min\left\{\lambda^c_2(\mathcal{G}_1), \lambda^c_2(\mathcal{G}_2)\right\}$$

Since the variable node degree stays the same for the factor graph Cartesian product(see definition 7) and assuming $d_{v1} = d_{v2}$ for simplicity, then we finally prove the lemma,

$$\Lambda^v_{n-1}(\mathcal{G}_1 \square \mathcal{G}_2) = \max\left\{\Lambda^v_{n-1}(\mathcal{G}_1),\ \Lambda^v_{n-1}(\mathcal{G}_2)\right\}$$

□

Example: (Product of Ramanujan graphs is not Ramanujan)

While it is not related to the factor graph expansion for the code, we show the product of Ramanujan graphs cannot be Ramanujan as a simple example of the lemma 8. From the proof of the lemma 8, we see that the adjacency matrix of the product graphs $A = A_1 \otimes I + I \otimes A_2$, and hence its eigenvalue is $\lambda_{ij}(\mathcal{G}_1 \square \mathcal{G}_2) = \lambda_i(\mathcal{G}_1) + \lambda_j(\mathcal{G}_2)$ where λ_i is the i^{th} eigenvalue of the adjacency matrix A.

For classical graphs, the second largest eigenvalue of the adjacency matrix λ_{n-1} of d_v-regular of graphs must be lower bounded by Alon-Boppana bound, i.e. $\lambda_{n-1} \geqslant \sqrt{2d_v - 1} - o(1)$. Hence, a graph is called a Ramanujan graph if it saturates the Alon-Boppana bound [45]. There exist a few explicit constructions to obtain the Ramanujan graph given in [45].

Assume two Ramanujan graphs are regular $d_{v_1} = d_{v_2} = d$, the second eigenvalue of the adjacency matrix $A(\mathcal{G}_1 \square \mathcal{G}_2) = d + \lambda(\mathcal{G}_1) = d + 2\sqrt{d-1} - o(1)$ as the biggest eigenvalue of the adjacency matrix is always the degree d. In this case, it is observed that $A(\mathcal{G}_1 \ \square\ \mathcal{G}_2) = d + 2\sqrt{d-1} - o(1) > 2\sqrt{2d-1} - o(1)$ given $d >> 1$(asymptotically, this is obvious assumption). Therefore, it is concluded that the Cartesian product of two Ramanujan graphs can never yield a Ramanujan graph. In this sense, the product structure ruins the high expansion property.

Remark: (Warning of the factor graph Cartesian product)

From the above example, people might think that the factor graph expansion of the hypergraph product construction can be discussed in a similar way since $\mathcal{G}_x, \mathcal{G}_z$ are also defined as a Cartesian product of two factor graphs $\mathcal{G}_1, \mathcal{G}_2$. However, the factor graph Cartesian

product has some subtle points. Firstly, based on the definition 28, if we have to make an analogy by mapping the factor graph into the corresponding hypergraph, the vertex of the hypergraph will correspond to the factor node, but the matter of interest is the variable node instead, which will be mapped into the hyperedges. Therefore, the expansion from the above example is technically the factor node expansion rather than the variable node expansion. Hence, some twists and connections between factor node expansion and variable node expansion are needed for our purpose.

The second difference is that the term "expansion" has different definitions for code and graph. In coding theory, it is the left-expansion defined in 19 that benefits the code properties whereas, in graph theory, the expansion is mostly defined through the Cheeger constant. Fortunately, by lemma 7, the different expansions can be converted back and forth. Therefore, it is crucial to analyze the left-expansion of the factor graph rather than the generalized Cheeger constant.

Finally, with the above lemma, we can give a computable bound of the left expansion for the product factor graph $\mathcal{G}_x$, $\mathcal{G}_z$ once we obtain the spectrum of the normalized variable and check Laplacian of $\mathcal{G}_1$ and $\mathcal{G}_2$.

Corollary 7 (Computable left-expansion bound for HGP construction). *Given two classical factor graphs $\mathcal{G}_1, \mathcal{G}_2$, the Z-factor graphs $\mathcal{G}_z$ and X-factor graphs $\mathcal{G}_x$ from the hypergraph product construction $HGP(\mathcal{G}_1, \mathcal{G}_2)$ have their generalized variable Cheeger constant h_{v_z}, h_{v_x} and the key left-expansion α bounded as follows,*

For $\mathcal{G}_z$, the variable Cheeger constant h_{v_z} is bounded by,

$$h_{v_z} \geqslant \frac{(d_{v_1} + d_{c_2}) - \max\left\{\Lambda^c_{m-1}(\mathcal{G}_1), \Lambda^v_{n-1}(\mathcal{G}_2)\right\}}{2((d_{v_1} + d_{c_2}) - 1)} = h^{\min}_{v_z}$$

$$h_{v_z} \leqslant \sqrt{2\left((d_{v_1} + d_{c_2}) - 1\right)\left((d_{v_1} + d_{c_2}) - \max\left\{\Lambda^c_{m-1}(\mathcal{G}_1), \Lambda^v_{n-1}(\mathcal{G}_2)\right\}\right)}$$

$$\alpha_z \geqslant \lceil \frac{d_v h_{v_z}}{d_c - 1} \rceil \geqslant \lceil \frac{d_v \cdot h^{\min}_{v_z}}{d_{v_1} + d_{c_2} - 1} \rceil$$

For $\mathcal{G}_x$, the variable Cheeger constant h_{v_x} is bounded by,

$$h_{v_x} \geqslant \frac{(d_{c_1} + d_{v_2}) - \max\left\{\Lambda^c_{m-1}(\mathcal{G}_2), \Lambda^v_{n-1}(\mathcal{G}_1)\right\}}{2((d_{c_1} + d_{v_2}) - 1)} = h^{\min}_{v_x}$$

$$h_{v_x} \leqslant \sqrt{2\left((d_{c_1} + d_{v_2}) - 1\right)\left((d_{c_1} + d_{v_2}) - \max\left\{\Lambda^c_{m-1}(\mathcal{G}_2), \Lambda^v_{n-1}(\mathcal{G}_1)\right\}\right)}$$

$$\alpha_x \geqslant \lceil \frac{d_v h_{v_x}}{d_c - 1} \rceil \geqslant \lceil \frac{d_v \cdot h^{\min}_{v_x}}{d_{v_2} + d_{c_1} - 1} \rceil$$

Proof. From the topological definition of lr-LP construction 11, the $\mathcal{G}_z = \mathcal{G}_1^* \square \mathcal{G}_2$ and $\mathcal{G}_x = \mathcal{G}_1 \square \mathcal{G}_2^*$ as the lifting size $l = 1$. It is then easy to see that their factor node degrees $d_{c_z} = d_{v_1} + d_{c_2}$ and $d_{c_x} = d_{c_1} + d_{v_2}$ from the definition. Then from lemma 8, substituting the check degree and eigenvalues to the variable Cheeger inequality 6, we obtain the equation in the corollary. □

So far, we have given a computational bound for the left expansion parameter of the hypergraph product construction. Then, built on top of the previous results, it is worth giving a rigorous analysis to show how the product structure ruins the left expansion properties of the product factor graph $\mathcal{G}_x, \mathcal{G}_z$ compared with its two classical components $\mathcal{G}_1, \mathcal{G}_2$. In the meantime, the analysis result could also be another evidence of the inferior position of Hypergraph product construction compared with the Lifted product code in the sense that the LP construction expectedly gives a better left-expansion parameter α.

Theorem 8 (Generalized Cheeger Inequality for product factor graph)**.** *Let two variable Cheeger constants of the factor graphs $\mathcal{G}_1, \mathcal{G}_2$ be h_{v1} and h_{v_2} such that $h^{\min}_{v_1} \leqslant h_{v_1} \leqslant h^{\max}_{v_1}$ and $h^{\min}_{v_2} \leqslant h_{v_2} \leqslant h^{\max}_{v_2}$ obtained from the variable Cheeger inequality, see theorem 6. Hence, the Cartesian product of the two factor graphs $\mathcal{G} = \mathcal{G}_1 \square \mathcal{G}_2$ has the generalized variable Cheeger inequality as follows,*

$$\min\{h^{\min}_{v_1},\ h^{\min}_{v_2}\} \leqslant h_v \leqslant 2 \cdot \max\{h^{\max}_{v_1},\ h^{\max}_{v_2}\}$$

where for simplicity of the following analysis, we assume $d_{c_1}, d_{c_2} >> 2$*, which is a reasonable assumption in the asymptomatic context.*

Proof. From theorem 6, the variable Cheeger inequality for the product factor graph $\mathcal{G} = \mathcal{G}_1 \square \mathcal{G}_2$ is,

$$\frac{d_c - \Lambda^v_{n-1}}{2(d_c - 1)} \leqslant h_v \leqslant \sqrt{2(d_c - 1)(d_c - \Lambda^v_{n-1})}$$

From definition 7, it is easy to see that the factor node degree of the product factor graph is the sum of the factor node degree of its two components, i.e. $d_c = d_{c_1} + d_{c_2}$. Also, from lemma 8, the second largest eigenvalue of the normalized variable Laplacian of the product graph $\Lambda^v_{n-1}(\mathcal{G}_1 \ \square \ \mathcal{G}_2) = \max\left\{\Lambda^v_{n-1}(\mathcal{G}_1),\ \Lambda^v_{n-1}(\mathcal{G}_2)\right\}$. The, by substitution, we have,

$$h_v \geqslant \frac{(d_{c_1} + d_{c_2}) - \max\left\{\Lambda^v_{n-1}(\mathcal{G}_1),\ \Lambda^v_{n-1}(\mathcal{G}_2)\right\}}{2(d_{c_1} + d_{c_2} - 1)}$$

$$h_v \leqslant \sqrt{2\Big((d_{c_1} + d_{c_2}) - 1\Big)\left((d_{c_1} + d_{c2}) - \max\left\{\Lambda^v_{n-1}(\mathcal{G}_1),\ \Lambda^v_{n-1}(\mathcal{G}_2)\right\}\right)}$$

For simplicity of the analysis, assume $d_{c_1}, d_{c_2} >> 2$, then the above equations can be simplified as,

$$h^{\min}_v = \frac{(d_{c_1} + d_{c_2}) - \max\left\{\Lambda^v_{n-1}(\mathcal{G}_1),\ \Lambda^v_{n-1}(\mathcal{G}_2)\right\}}{2(d_{c_1} + d_{c_2})} \approx \min\{h^{\min}_{v_1},\ h^{\min}_{v_2}\}$$

$$h^{\max}_v = \sqrt{2 \cdot (d_{c_1} + d_{c_2})\left((d_{c_1} + d_{c2}) - \max\left\{\Lambda^v_{n-1}(\mathcal{G}_1),\ \Lambda^v_{n-1}(\mathcal{G}_2)\right\}\right)}$$

$$\leqslant 2 \cdot \max\{h^{\max}_{v_1},\ h^{\max}_{v_2}\}$$

□

Finally, we have all of the mathematical tools needed to analyze how topological products affect the left-expansion properties rigorously. Even though people might have already

realized that the product construction can hurt the high expansion properties, it is still the first rigorous analysis to prove and demonstrate the exact degradation of the left-expansion resulting from the factor graph Cartesian product.

Given two class of factor graphs $\{\mathcal{G}_1\}, \{\mathcal{G}_2\}$ which are both $(n, m, d_v, 0.5, \lceil \frac{d_v h_v^{\min}}{d_c - 1} \rceil)$ spectral left expander, with left expansion parameters $\alpha = \lceil \frac{d_v h_v^{\min}}{d_c - 1} \rceil$. Then, the factor graph Cartesian product will decrease the left-expansion parameter of the product factor graph α by half.

Corollary 8 (Product degrades the left expansion)**.** *Given $\{\mathcal{G}_1\}$ and $\{\mathcal{G}_2\}$ with the same variable node and check node degree (d_v, d_c) for simplicity, the factor graph Cartesian product result in the class of the product graph $\left\{ \mathcal{G}_1 \square \mathcal{G}_2 \right\}$ with only half of the original left expansion, i.e.* $\alpha(\mathcal{G}_1 \square \mathcal{G}_2) = 0.5 \cdot \alpha(\mathcal{G}_1) = 0.5 \cdot \alpha(\mathcal{G}_2)$.

Proof. Given a class of factor graph, the only computable way to bound its key left-expansion parameter α (see definition 19 and theorem 7) is through the variable Cheeger inequality stated in 6.

For a factor graph $\mathcal{G}_1, \mathcal{G}_2$ with the variable Cheeger constant bounded below by $h_{\min}$, both of these factor graph will have the left-expansion parameter $\alpha = \lceil \frac{d_v h_v^{\min}}{d_c - 1} \rceil$ from lemma 7.

Hence, from theorem 8, we prove that the $h_v^{\min}(\mathcal{G}_1 \square \mathcal{G}_2) = h_v^{\min}(\mathcal{G}_1) = h_v^{\min}(\mathcal{G}_2)$. Additionally, the variable node degree and factor node degree are $d_v(\mathcal{G}_1 \square \mathcal{G}_2) = d_v(\mathcal{G}_1) = d_v$, and $d_c(\mathcal{G}_1 \square \mathcal{G}_2) = 2d_c(\mathcal{G}_1) = 2d_c$. Therefore, we have the $\alpha(\mathcal{G}_1 \square \mathcal{G}_2)$,

$$\alpha(\mathcal{G}_1 \square \mathcal{G}_2) = \lceil \frac{d_v(\mathcal{G}_1 \square \mathcal{G}_2) \cdot h_v^{\min}(\mathcal{G}_1 \square \mathcal{G}_2)}{d_c(\mathcal{G}_1 \square \mathcal{G}_2) - 1} \rceil = \lceil \frac{d_v h_v^{\min}(\mathcal{G}_1)}{2d_c - 1} \rceil = 0.5 \cdot \alpha(G_1)$$

□

4. Conclusions

This thesis embarked on a journey to explore the landscape of high-rate quantum LDPC (qLDPC) code constructions, driven by the challenge of simplifying construction complexity and filling the gap in finite-length distance bounds. We introduced a unifying framework that simplifies the understanding and application of high-rate qLDPC codes, enhancing both theoretical insight and practical utility in code design and engineering, as detailed in a specific section.

Practically, the cycle analysis conducted in this thesis highlights the superiority of the number of dominant cycles in the left-right Lifted product code compared to the hypergraph construction. This advantage establishes a crucial benchmark for those interested in decoding algorithms, suggesting that, along with the finite-length distance, the Lifted product construction is likely to be of greater practical interest than hypergraph product codes.

Expansion is a highly sought-after property in code design, yet there has been a lack of computational tools for engineers to assess and compare expansion properties effectively. Consequently, the lower bound provided for left expansion could serve as a valuable tool for code design. Furthermore, from a theoretical standpoint, the analysis of product expansion rigorously demonstrates that the factor graph Cartesian product tends to reduce left-expansion by half in the worst-case scenario, offering critical insights for future code development.

4.1 Limitation and Future work

Despite the achievements detailed above, there remain several limitations and areas for further exploration and improvement. One such limitation pertains to the finite-length distance, where the technical condition requires the proto-matrix to be full-rank, which may not always be ideal. Relaxing this technical condition could significantly enhance the flexibility for practical code construction, opening avenues for future work.

In terms of cycle analysis, although the number of dominant 8-cycles is lower in the

left-right Lifted product code compared to the hypergraph product code, there still exists a large number of $\Theta(N)$ 8-cycles. This scenario is far from the desired condition for effective message-passing decoding. Moreover, it has been proven that the girth cannot exceed 8, presenting an additional challenge for practical decoding applications. Given these constraints, a more feasible approach may involve modifying and developing the decoder to better accommodate these cycle conditions rather than attempting to alter the code construction.

Regarding expansion, a current limitation lies in the fact that while the hypergraph Cheeger constant offers a lower bound, it is presumably a loose bound since it is designed considering only half of the variable nodes. Practically, it would be more beneficial to control both the size of the variable nodes γ and the left-expansion parameter α. This area warrants further investigation to develop tighter computational bounds, representing a significant direction for future research.

References

[1] D. Gottesman, “Stabilizer Codes and Quantum Error Correction.” arXiv, May 28, 1997. Accessed: Mar. 12, 2024. [Online]. Available: http://arxiv.org/abs/quant-ph/9705052.

[2] A. R. Calderbank and P. W. Shor, “Good quantum error-correcting codes exist,” Phys. Rev. A, vol. 54, no. 2, pp. 1098–1105, Aug. 1996, doi: 10.1103/PhysRevA.54.1098.

[3] R. G. Gallager, "Low Density Parity Check Codes," Monograph, M.I.T. Press, 1963. [Online]. Available: http://web.mit.edu. Accessed: August 7, 2013.

[4] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, “Topological quantum memory,” Journal of Mathematical Physics, vol. 43, no. 9, pp. 4452–4505, Sep. 2002, doi: 10.1063/1.1499754.

[5] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg, “Threshold error rates for the toric and surface codes.” arXiv, May 05, 2009. doi: 10.48550/arXiv.0905.0531.

[6] M. H. Freedman and D. A. Meyer, “Projective Plane and Planar Quantum Codes,” Foundations of Computational Mathematics, vol. 1, no. 3, pp. 325–332, Jul. 2001, doi: 10.1007/s102080010013.

[7] J.-P. Tillich and G. Zemor, “Quantum LDPC codes with positive rate and minimum distance proportional to $n^{\frac{1}{2}}$;,” in 2009 IEEE International Symposium on Information Theory, Seoul, South Korea: IEEE, Jun. 2009, pp. 799–803. doi: 10.1109/ISIT.2009.5205648.

[8] S. Evra, T. Kaufman, and G. Zémor, “Decodable Quantum LDPC Codes beyond the $\sqrt{n}$ Distance Barrier Using High-Dimensional Expanders,” SIAM J. Comput., pp. FOCS20-276, Jun. 2022, doi: 10.1137/20M1383689.

[9] T. Kaufman and R. J. Tessler, “Quantum LDPC codes with $\Omega(\sqrt{N}\log^k(N))$ distance, for any k,” ArXiv, vol. abs/2008.09495, 2020, [Online]. Available: https://api.semanticscholar.org/CorpusID:221356006

[10] M. B. Hastings, J. Haah, and R. O’Donnell, “Fiber bundle codes: breaking the n 1/2 polylog(n) barrier for Quantum LDPC codes,” in Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Italy: ACM, Jun. 2021, pp. 1276–1288. doi: 10.1145/3406325.3451005.

[11] N. P. Breuckmann and J. N. Eberhardt, “Balanced Product Quantum Codes,” IEEE Trans. Inform. Theory, vol. 67, no. 10, pp. 6653–6674, Oct. 2021, doi: 10.1109/TIT.2021.3097347.

[12] P. Panteleev and G. Kalachev, "Quantum LDPC Codes With Almost Linear Minimum Distance," IEEE Trans. Inform. Theory, vol. 68, no. 1, pp. 213–229, Jan. 2022, doi: 10.1109/TIT.2021.3119384.

[13] P. Panteleev and G. Kalachev, "Asymptotically Good Quantum and Locally Testable Classical LDPC Codes." arXiv, Jan. 21, 2022. doi: 10.48550/arXiv.2111.03654.

[14] I. Dinur, S. Evra, R. Livne, A. Lubotzky, and S. Mozes, "Locally testable codes with constant rate, distance, and locality," in Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing, in STOC 2022. New York, NY, USA: Association for Computing Machinery, 2022, pp. 357–374. doi: 10.1145/3519935.3520024.

[15] A. Leverrier and G. Zemor, "Quantum Tanner codes," in 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS), Denver, CO, USA: IEEE, Oct. 2022, pp. 872–883. doi: 10.1109/FOCS54457.2022.0 0117.

[16] P. Panteleev and G. Kalachev, "Degenerate Quantum LDPC Codes With Good Finite Length Performance," Quantum, vol. 5, p. 585, Nov. 2021, doi: 10.22331/q-2021-11-22-585.

[17] R. Wang, H.-K. Lin, and L. P. Pryadko, "Abelian and Non-Abelian Quantum Two-Block Codes," 2023 12th International Symposium on Topics in Coding (ISTC), pp. 1–5, 2023.

[18] A. A. Kovalev and L. P. Pryadko, "Quantum Kronecker sum-product low-density parity-check codes with finite rate," Physical Review A, vol. 88, 2013, arXiv:1212.6703. DOI: https://doi.org/10.1103/PhysRevA.88.022311.

[19] K.-Y. Kuo and C.-Y. Lai, "Refined Belief Propagation Decoding of Sparse-Graph Quantum Codes," IEEE J. Sel. Areas Inf. Theory, vol. 1, no. 2, pp. 487–498, Aug. 2020, doi: 10.1109/JSAIT.2020.3011758.

[20] V. Kaibel, "13. On the Expansion of Graphs of 0/1-Polytopes," in The Sharpest Cut, pp. 199–216. doi: 10.1137/1.9780898718805.ch13.

[21] R. Mulas, "A Cheeger Cut for Uniform Hypergraphs," Graphs and Combinatorics, vol. 37, no. 6, pp. 2265–2286, Nov. 2021, doi: 10.1007/s00373-021-02348-z.

[22] O. Fawzi, A. Grospellier, and A. Leverrier, "Efficient decoding of random errors for quantum expander codes," in Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, Los Angeles CA USA: ACM, Jun. 2018, pp. 521–534. doi: 10.1145/
3188745.3188886.

[23] A. Hatcher, Algebraic Topology. Cambridge: Cambridge Univ. Press, 2001. ISBN: 0-521-79160-X.

[24] W. Fulton and J. Harris, Representation Theory: A First Course. New York: Springer-Verlag, 1991, vol. 129, Graduate Texts in Mathematics, Readings in Mathematics. doi: 10.1007/978-1-4612-0979-9.

[25] R. Smarandache and P. O. Vontobel, "Quasi-Cyclic LDPC Codes: Influence of Proto- and Tanner-Graph Structure on Minimum Hamming Distance Upper Bounds," IEEE Trans. Inform. Theory, vol. 58, no. 2, pp. 585–607, Feb. 2012, doi: 10.1109/TIT.2011.2173244.

[26] P. S. Alexandrov, Combinatorial Topology Vols. I, II, III, translated by H. Komm, Graylock Press, 1956, MR 1643155.

[27] W. E. Ryan, "An introduction to LDPC codes," in CRC Handbook for Coding and Signal Processing for Recording Systems, vol. 5, no. 2, pp. 1-23, CRC Press, Sept. 2004. [Online]. Available: http://www.free.fr/[PDF]

[28] V. G. Vizing, "The Cartesian product of graphs," Vycisl. Sistemy, no. 9, pp. 30-43, 1963.

[29] Bretto, Y. Silvestre, and T. Vallee, "Cartesian product of hypergraphs: Properties and algorithms," presented at the Computing Research Repository - CORR, Sep. 2009, pp. 22–28. doi: 10.4204/EPTCS.4.3.

[30] J. L. Gross, "Voltage graphs," in Discrete Mathematics, vol. 9, no. 3, pp. 239-246, 1974. doi:10.1016/0012-365X(74)90006-5.

[31] D. G. M. Mitchell, R. Smarandache, and D. J. Costello, "Quasi-Cyclic LDPC Codes Based on Pre-Lifted Protographs," IEEE Trans. Inform. Theory, vol. 60, no. 10, pp. 5856–5874, Oct. 2014, doi: 10.1109/TIT.2014.2342735.

[32] A. Brooke-Taylor, "Products of CW complexes – the full story," 2017. Available online.

[33] R. Fritsch and R. Piccinini, Cellular Structures in Topology. Cambridge: Cambridge University Press, 1990. doi: 10.1017CBO9780511983948.

[34] R. Diestel, Graph Theory, 3rd ed., vol. 173, Graduate Texts in Mathematics. Berlin: Springer-Verlag, 2005, ISBN 3-540-26183-4.

[35] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen Years of Quantum LDPC Coding and Improved Decoding Strategies," IEEE Access, vol. 3, pp. 2492–2519, 2015, doi: 10.1109/ACCESS.2015.2503267.

[36] M. Hagiwara and H. Imai, "Quantum Quasi-Cyclic LDPC Codes," in 2007 IEEE International Symposium on Information Theory, Nice: IEEE, Jun. 2007, pp. 806–810. doi: 10.1109/ISIT.2007.4557323.

[37] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," arXiv.org. Accessed: Feb. 26, 2024. [Online]. Available: https://arxiv.org/abs/0801.1241v2.

[38] Y. Wang, S. C. Draper, and J. S. Yedidia, "Hierarchical and High-Girth QC LDPC Codes," IEEE Trans. Inform. Theory, vol. 59, no. 7, pp. 4553–4583, Jul. 2013, doi: 10.1109/TIT.2013.2253512.

[39] M. Karimi and A. H. Banihashemi, "On the Girth of Quasi-Cyclic Protograph LDPC Codes," IEEE Trans. Inform. Theory, vol. 59, no. 7, pp. 4542–4552, Jul. 2013, doi: 10.1109/TIT.2013.2251395.

[40] J.-L. Kim, U. N. Peled, I. Perepelitsa, V. Pless, and S. Friedland, "Explicit Construction of Families of LDPC Codes With No 4-Cycles," IEEE Trans. Inform. Theory, vol. 50, no. 10, pp. 2378–2388, Oct. 2004, doi: 10.1109/TIT.2004.834760.

[41] N. Baspin and A. Krishna, "Connectivity constrains quantum codes," Quantum, vol. 6, p. 711, May 2022, doi: 10.22331/q-2022-05-13-711.

[42] A. O. Quintavalle, M. Vasmer, J. Roffe, and E. T. Campbell, "Single-Shot Error Correction of Three-Dimensional Homological Product Codes," PRX Quantum, vol. 2, no. 2, p. 020340, Jun. 2021, doi: 10.1103PRXQuantum.2.020340.

[43] O. Fawzi, A. Grospellier, and A. Leverrier, "Constant overhead quantum fault tolerance with quantum expander codes," Commun. ACM, vol. 64, no. 1, pp. 106–114, Jan. 2021, doi: 10.1145/3434163.

[44] R. Meshulam, "Graph codes and local systems," arXiv: Combinatorics, 2018, [Online]. Available: https://api.semanticscholar.org/CorpusID:119156786.

[45] M. Sipser and D. Spielman, "Expander codes," IEEE Transactions on Information Theory, vol. 42, no. 6, pp. 1710-1722, 1996.

[46] V. Guruswami, and A. Sharma, "Notes 8: Expander Codes and their decoding".