Construction of QC-LDPC Codes with Low Error Floor by Efficient Systematic Search and Elimination of Trapping Sets

by

Bashirreza Karimi

A dissertation submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
January, 2021

Abstract

In this thesis, we propose a systematic design of binary protograph-based quasi-cyclic (QC) low-density parity-check (LDPC) codes with low error floor. We first characterize the trapping sets of such codes and demonstrate, using edge coloring techniques, that the QC structure of the code eliminates some of the trapping set structures that can exist in a code with the same degree distribution and girth but lacking the QC structure. Based on this characterization, our design aims at eliminating a targeted collection of trapping sets. Considering the parent/child relationship between the structures of the collection, we search for and eliminate those trapping sets that are in the collection but are not a child of any other trapping set in the collection. An efficient layered algorithm is designed for the search of these targeted trapping sets. Compared to the existing codes in the literature, the designed codes are superior in the sense that they are free of the same collection of trapping sets while having a smaller block length, or a larger collection of trapping sets while having the same block length. In addition, the efficiency of the search algorithm makes it possible to design codes with larger degrees which are free of trapping sets within larger ranges compared to the state-of-the-art.

Moreover, we design finite-length binary irregular protograph-based QC-LDPC codes with good waterfall performance and low error floor. To achieve a low error floor, we eliminate a targeted set of dominant elementary trapping sets (ETS) $\mathcal{L}$ in the Tanner graph of the code. This is a more challenging task since the variety of problematic structures is much larger than that of regular codes. For a given rate and girth, the codes are designed to be free of the largest set of problematic ETSs for a given block length, or to have the shortest block length while a given set of ETSs is avoided. The design is based on a search algorithm that identifies whether any instance of any structure within $\mathcal{L}$ exists in the Tanner graph of the constructed code or not. The search algorithm performs this task with minimal complexity, making it feasible to construct practical codes by running the search algorithm a large number

of times. Simulation results are provided to demonstrate the superior performance of designed codes compared to similar state-of-the-art irregular QC-LDPC codes.

In addition, we address the problem of characterizing the trapping sets for non-binary (NB) LDPC codes. To perform this task, not only the topological condition for a trapping set should be checked, but also specific algebraic constraints must be satisfied which makes the search process complex. The proposed search approach finds the NB trapping sets of both regular and irregular NB-LDPC code exhaustively and with minimum amount of complexity. The proposed search algorithm is efficient since the complexity of checking topological and algebraic conditions is minimized. Using the proposed search technique, we design regular and irregular NB-QC-LDPC codes with low error floor by removing a given collection of harmful structures. Compared to the existing codes in the literature, the constructed codes have better performance and trapping set distribution or smaller length.

To my wife Arefeh,
my daughter Rosa,
and my supportive parents
Amir and Fariba.

Acknowledgments

First, I would like to thank my supervisor, Professor Amir H. Banihashemi for leading and encouraging me during my research. His guidance helped me in all the time of research and writing of this thesis.

Thanks must also go to my committee members for their helpful suggestions and discussions.

I can not express my appreciation to my wonderful wife, Arefeh who devoted her never ending love to me and was my partner towards success. I also appreciate the love and support of my family, specially my parents who are always the source inspiration and encouragement for me. And finally, many thanks to my friends, especially my friends in our research group.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Protograph-based QC-LDPC codes are an important category of LDPC codes, adopted in many standards [1–3]. The Tanner graphs for such codes are obtained by cyclically lifting a small bipartite graph, called *base graph* or *protograph*. Protograph-based QC-LDPC codes not only have a competitive performance under iterative decoding algorithms over a variety of channels, but also enjoy efficient implementations which take advantage of the QC structure of the code. A potential problem in using QC-LDPC codes in applications that require low error rates is the *error floor*, characterized by a change in the slope of error rate curves as the channel quality improves. There are two main approaches to improve the error floor of LDPC codes: (1) modification of the decoding algorithm, and (2) new/modified code constructions. In this thesis, we focus on the second approach and deal with construction of QC-LDPC codes with low error floors.

Spatial coupling [45–47] is considered as another method of reducing the error floor of LDPC codes. Also, significant number of studies conducted on lowering the error floor of LDPC codes based on decoder design [43, 48, 50, 51, 86]. Averaged belief propagation decoding algorithm has been proposed in [80] where a sudden magnitude change in the values of certain variable messages, or fast convergence to an unreliable estimate considered as a possible indicator of trapping sets. As a result, reducing the magnitude of large changes or slowing down of convergence for certain variable nodes can improve the error floor. In [81–83], two or multi-stage decoding algorithms were proposed where variable nodes with a high probability of being wrongly decoded are chosen for flipping or erasing to break trapping sets.

Compared to the averaged decoding method, they have faster convergence speed at low and moderate SNR, but unstable error events may occur due to accidental flipping of correct bits. Moreover, [84–87] deal with error floor in the decoding algorithm by identifying and resolving the harmful structures. A multi-step quantization technique has been also proposed in [88].

Elementary trapping sets (ETSs) of LDPC codes are known to be the main culprits in the error floor region over the additive white Gaussian noise (AWGN) channel (see, e.g., [4,5], and the references therein). These sets have been, in general, characterized in [4, 5] using the so-called *dpl characterization*. The *dpl* characterization describes each and every ETS S as an embedded sequence of ETSs that starts from a simple cycle and expands recursively, in each step by one of the three simple expansions *dot*, *path* and *lollipop*, to reach S. Associated with the *dpl* characterization is an efficient *dpl* search algorithm [4, 5], that can find all the ETSs within a range of interest, exhaustively.

It is well-established in the literature that the waterfall performance of LDPC codes is improved by introducing irregularity in the degree distribution of the codes. This improvement in the waterfall performance however, often comes at the expense of higher error floor. This is due to the increase in the variety of the problematic structures, i.e. trapping sets [6], that trap the iterative decoder in the error floor region. The design of irregular LDPC codes with low error floor is thus, in general, a challenging task compared to regular ones. Nevertheless, there are a variety of applications for irregular QC-LDPC codes [1–3, 7, 8], many of them such as optical links, storage devices and space communications requiring low error floors.

LDPC codes were also generalized to non-binary (NB) case over the Galois field (GF) with an order $q > 2$ by Davey and Mackay [9]. They showed that NB-LDPC codes can offer better performance under iterative decoding compared to their binary counterparts for short and moderate code lengths [9, 10] over the (AWGN) channels. NB codes are being actively used for a number of modern communication systems such as space telecommunication systems [11] and data storage devices [12, 13]. It is well-known that elementary trapping sets in NB-LDPC codes play an important role as error prone structures over AWGN channels in the error floor region [14].

1.2 Related Works

The parity-check matrix of a code can be constructed by optimizing different properties of the code such as degree distribution and *girth* (the length of the shortest cycle(s) within the Tanner graph) to achieve better waterfall and error floor performance, respectively. In the design process, the non-zero elements of the parity-check matrix should be chosen from GF(q) where the order of field is usually considered as a power of two, i.e. $q = 2^p$, where for binary codes $p = 1$. In this thesis, we refer to the cycles in binary parity-check matrices as *binary cycles* and the cycles in the NB parity-check matrices as *non-binary cycles*. For a binary cycle, we are only concerned with the topology of a cycle. However, in non-binary cycles, not only the topology is important, but also the labels of edges within the cycle should be taken in to account. Thus, certain topological and algebraic conditions should be satisfied to have a valid non-binary cycle. The length of shortest non-binary cycle(s) in the Tanner graph of a NB-LDPC code is called *algebraic girth*.

Increasing the girth of QC-LDPC codes and removing the dominant trapping set structures in the Tanner graph of codes are two main methods dealing with the error floor in the construction process of regular LDPC codes. Progressive-edge-growth (PEG) is one of the most well-known algorithms to design LDPC codes with large girth [15]. The PEG algorithm also has been extended to irregular [16] and QC structures [17]. In [18], Asvadi *et al.* designed QC-LDPC codes with low error floor by removing the short cycles that were part of dominant trapping sets. The same authors [19] also proposed another technique based on the approximate cycle extrinsic message degree (ACE) spectrum [36] to design irregular QC-LDPC codes with good error floor. In [20], Nguyen *et al.* constructed structured regular LDPC codes with low error floor over the binary symmetric channel (BSC). The low error floor in [20] was achieved by ensuring that certain small trapping sets were absent in the code. Khazraei *et al.* [21] modified the PEG algorithm for the construction of LDPC codes to avoid the creation of dominant trapping sets in the construction process. In [22], Wang *et al.* used the cycle consistency matrix to design separable circulant-based LDPC codes, in which certain absorbing sets were avoided. More recently, Diouf *et al.* [23] proposed an improved PEG algorithm to construct regular LDPC codes with variable degree 3 and girth 8 without $(5, 3)$ trapping sets and with the minimum number of $(6, 4)$ trapping sets. (An (a, b) trapping set has a variable nodes and b odd-degree check nodes in its subgraph.) Most recently, in [24], Tao *et al.* proposed a

construction of QC-LDPC codes with variable node degree 3 and girth 8 from fully-connected protographs, where (a, b) elementary trapping sets (ETSs) with $a \leq 8$ and $b \leq 3$ were removed by avoiding certain cycles of length 8 (8-cycles) in the Tanner graph. Based on their approach, the authors of [24] also derived lower bounds on the lifting degree, and as a result on the block length, of the designed codes, and were able to find codes, using random search, to either achieve or approach the bound. Based on an approach similar to that of [24], by avoiding certain 8-cycles, Amirzadeh and Sadeghi [25] constructed QC-LDPC codes with girths 6 and 8 that are cyclic liftings of the $3 \times n$ fully-connected base graph. For girth 6, the constructed codes are free of (a, b) ETSs with $a \leq 5$ and $b \leq 2$, and for girth 8, they are free of ETSs with $a \leq 8$ and $b \leq 3$. The codes constructed in [25], however, do not seem to provide any improvement compared to the state-of-the-art including the codes designed in [24].[1] The same authors also constructed QC-LDPC codes with girths 6 and 8 that are cyclic liftings of the $4 \times n$ fully-connected base graph in [26]. For girth 6, the constructed codes are free of $(5, b)$ ETSs with $b \leq 4$, and $(6, b)$ ETSs with $b \leq 2$. For girth 8, they are only free of $(7, 4)$ ETSs. To derive sufficient conditions for the elimination of the targeted ETSs, the authors used discussions based on edge coloring. In [27], small TSs were eliminated from regular QC-LDPC codes by imposing certain constraints on cycles of length 8.

Many of the existing works on the design of irregular LDPC codes with low error floor use indirect measures of the error floor performance as design criteria. Many such measures are related to the girth [15, 16, 28–31], the multiplicity of short cycles in the code's Tanner graph [32–35], and how connected the short cycles are to the rest of the graph [16, 28, 29, 31, 36, 37]. Progressive-edge-growth (PEG) [15] is one of the most efficient methods which aims to maximize the local girth of the Tanner graph and it has been used to construct both regular and irregular LDPC codes. Alternatively, girth profile and sliding-window girth metrics have been defined in [30] to increase the girth value and reduce the number of short cycles of irregular structured LDPC codes. In [35], a weighted sum of cycles with different length has been introduced as a new metric of constructing good regular and irregular QC-LDPC codes. Other methods such as [32–34] have also attempted to design regular and irregular QC-LDPC codes with improved girth value and lower numbers of short

[1] We also note that the results presented in Table II of [25] for 3×5 and 3×6 base graphs are erroneous because in each case, some of the elements of the exponent matrix are larger than the lifting degree, and the lifting degree itself violates the lower bound provided in Corollary 2 of [25].

cycles by designing a masking matrix which removes some edges of the Tanner graph of a regular base code. Mentioned methods use girth as the main prediction of error floor performance of LDPC codes. There exist other works that consider different design criteria such as the approximate cycle extrinsic message degree (ACE) [36] of cycles and ACE spectrum [37] of the code's Tanner graph to improve the error floor performance of irregular LDPC codes. Due to the superior performance of PEG method, various PEG-based algorithms have been proposed in the literature in which the PEG algorithm has been used in conjunction with other metrics such as extrinsic message degree (EMD) [31], ACE [16], ACE spectrum [28], approximate minimum cycle set EMD (ACSE) [29] to achieve further improvement of error floor performance. The EMD and ACE properties explain that not only the length of the shortest cycle (girth) is important, but also the cycle connectivity in the Tanner graph plays significant role in the error-floor of LDPC codes. Thus, in some cases larger cycles may be more harmful than short cycles because of poor graph connectivity.

There are also a number of papers that tackle the design of codes with low error floor by the direct elimination of TSs, see, e.g., [18, 19, 21, 27, 38, 39]. In [38], small stopping sets [40] in irregular LDPC codes were eliminated by adding some new parity-check equations. A modified PEG algorithm [15] was proposed in [39] to avoid small stopping sets in the Tanner graph of irregular LDPC codes. In [18, 19], Asvadi *et al.* proposed a technique based on cyclic liftings to design both regular and irregular QC-LDPC codes by removing some dominant TSs of a base code. This was achieved at the expense of increasing the block length. In [21], PEG algorithm was modified to avoid some dominant TSs in irregular codes.

There is a body of work that considers absorbing sets to be responsible for the error floor of LDPC codes (as an alternative to TSs), see, e.g., [22, 41–43]. There are close relationships between absorbing sets and TSs. In particular, elementary absorbing sets, which are a category of absorbing sets widely studied in the literature [22, 41–43], are in general a subset of leafless ETSs (LETSs) [4, 44], considered in this work. (The two categories of LETSs and elementary absorbing sets are identical for LDPC codes with maximum variable node degree 3.) Therefore, the general characterization of LETSs for QC-LDPC codes and the layered search algorithm of LETSs proposed in this thesis are both readily applicable to elementary absorbing sets. We also note that an example of an LDPC code with variable node degree 4 is provided in [4] ($\mathcal{C}_{10}$), for which the error-prone structures of quantized iterative decoders over the

AWGN channel in the error floor region are shown to be LETSs and not elementary absorbing sets. The existence of such cases further justifies the choice made in this study to focus on LETSs rather than elementary absorbing sets.

A proper selection of non-zero elements for NB-LDPC code (or equivalently labeling the binary Tanner graph) can lead to better performance in both waterfall and error floor regions compared to random labeling. Here, we focus on the error floor performance of the NB-LDPC codes. In this regard, the existence of short length cycles in the Tanner graph of the code results in a degraded error floor performance. For NB Tanner graphs, a structure is called *non-binary trapping set* (NB-TS) when it is formed by non-binary cycles. Improving the error floor performance of NB-LDPC codes can be achieved directly via removing small trapping sets, and indirectly by increasing the algebraic girth and minimizing the length of shortest NB cycles within the Tanner graph of the code.

Considering indirect measures as the design criteria for improving the error floor performance of NB codes, a cycle cancellation technique for regular $(2, d_c)$ codes with large algebraic girth was proposed in [52]. In this regard, through a labeling process, all non-binary cycles smaller than a certain size l_{max} were avoided by assuring that full-rank condition (which is equivalent to the algebraic condition mentioned before) is not satisfied for any cycle of length less than or equal to l_{max}. This method of assigning NB values has been also extended to regular NB quasi-cyclic (QC) codes in [53]. Note that in both [52, 53], a binary LDPC code is first generated using the well-known PEG algorithm [15]. Then, the labeling process is done such that the final NB code is free of small NB cycles. It is worth to mention that the degree distribution is selected with respect to binary Tanner graph. There exist other studies that consider the optimization problem related to measures such as maximizing the algebraic girth and minimizing the multiplicity of short non-binary cycles [30, 54–56], and the connection of short NB cycles to the rest of the NB Tanner graph [57]. In particular, [54, 55] attempt to design NB-LDPC codes with large algebraic girth and small multiplicity of short NB cycles for regular and irregular Tanner graphs. In [30], irregular NB-QC-LDPC codes are constructed with large algebraic girth using binary QC-LDPC codes and labeling the binary parity-check matrix. A low-complexity algorithm for counting the number of short non-binary cycles based on a message passing algorithm was presented in [56] to design regular/irregular NB-LDPC codes

with large algebraic girth and fewer number of NB cycles. Also, irregular NB-QC-LDPC codes was constructed such that a predetermined *approximate cycle extrinsic message degree* (ACE) spectrum were satisfied. The ACE criteria was applied for both binary cycles and NB cycles during the design process.

While there are a large body of research dealing with error floor improvement indirectly via increasing the algebraic girth and reducing the number of shortest NB cycles, only limited works exist that consider non-binary trapping sets or absorbing sets in the design process of NB-LDPC codes. In [14], Amiri gave a clear definition of *non-binary absorbing sets* (NB-ASs) in codes with column weight equal or larger than two. The authors mentioned that in the case of AWGN channels, only elementary NB-ASs (NB-EASs) impact the error floor performance of NB-LDPC codes. Thus, elimination of NB-EASs was considered as their goal to achieve low error floor performance. In the design process, a binary LDPC code is first generated using the PEG algorithm and the distribution of binary EASs in a range of interest is obtained. Then, the non-zero elements of the parity-check matrix are chosen randomly. Finally, through an iterative search routine, a collection of NB-EASs are avoided by making sure that at least one cycle is not non-binary cycle per each structure. The elimination process is done by changing the edge values that belong to GF(q). Obviously, after removing one structure within the collection, one needs to check all previously removed NB-EASs in terms of NB cycle. The same authors extend their design technique to QC codes in [58]. In [59, 60], Hareedy *et al.* addressed the error floor problem for asymmetric channels such as partial-response (PR) channels and mentioned that error prone structures in the error floor region are not limited to only NB-EASs. Thus, the authors considered non-elementary structures and proposed *weight consistency matrix* (WCM) framework to remove dominant structures. Note that, in [14, 58–60], NB structures were found using binary Tanner graph instead of considering the original NB graph which make their method inefficient.

1.3 Summary of Contributions

In [62], we first characterize the ETSs of QC-LDPC codes, and demonstrate that some of the ETS structures that can exist in a general (randomly constructed) LDPC code are absent in QC-LDPC codes due to the QC structure of the code. To find such structures, we translate the problem into an edge coloring problem involving the

normal graph [61] of the ETS structure. Our characterization still follows the *dpl* principle but with fewer structures compared to those of a general LDPC code, as considered in [4, 5]. Based on the characterization of ETSs in QC-LDPC codes, we then propose a systematic approach to design protograph-based QC-LDPC codes that are free of a certain collection of trapping sets. The design is based on investigating the parent/child relationship between all the ETS structures within the collection and target those that are not child to any structure within the collection. We then devise an efficient layered algorithm to search for the targeted structures in the construction process. Compared to the exhaustive *dpl* search of [4,5], the proposed search algorithm is significantly less complex. This is mainly due to the fact that while the goal of the *dpl* algorithm of [4, 5] is to find all the instances of a certain collection of ETS structures, our goal here is only to verify whether any instances of at least one of the targeted structures exists in the code. A number of techniques are then employed to solve this new problem efficiently. In particular, the problem is formulated as a backward recursion in which the goal is to minimize the number of intermediate structures to reach the targeted structures and to use structures that have a higher chance of having a smaller multiplicity in the graph. It is important to note that the proposed layered characterization/search algorithm of ETSs can also be used to design LDPC codes with low error floor that lack the QC structure. The only difference is that for such codes the number of possible ETS structures is larger.

The constructed codes in [62] are superior to the state-of-the-art codes in the literature in the sense that, with the same protograph, they are either free of the same collection of ETSs while having a shorter block length, or are free of a larger range of trapping sets (and thus have a superior error floor) while having the same block length. To the best of our knowledge, the proposed design is the first that can systematically, efficiently and optimally construct QC-LDPC codes that are free of a certain collection of trapping sets. The systematic (general) nature of the design means that it is applicable to a variety of QC-LDPC codes with different node degrees, girths and different choices of targeted ETS structures. The high efficiency (low complexity) makes it possible to design codes of larger block length with wider range of node degrees and to target larger collections of trapping sets. The optimality of the design, which ensures that only the structures of interest are targeted for elimination, guarantees that the block length is minimized (for the elimination of a given set of ETSs) or that the largest collection of ETSs are removed (for a given block length).

This is in contrast with some of the existing work, such as [23, 24], in which, some parent structures which are not of direct interest are targeted for elimination just to eliminate some of their children.

In [63], we consider the construction of irregular QC-LDPC codes that are free of a certain collection of harmful ETSs. The category of ETSs includes LETSs, and is known to contain the problematic structures in the error floor of irregular LDPC codes over the AWGN channel [5]. The Tanner graphs of the codes constructed here are cyclic liftings of some irregular base graphs whose degree distributions and topology are designed for good waterfall performance. The superior error floor performance is then achieved through the design of the exponent matrix, that contains the cyclic permutation shifts assigned to different edges of the base graph. The most computationally expensive part of the design is a search algorithm that determines whether an irregular Tanner graph contains any instance of ETS structures that are targeted to be eliminated from the code. This search algorithm would be used hundreds or even thousands of times in the construction process. The search algorithm is carefully devised in multiple layers using three simple expansion techniques of *dot*, *path* and *lollipop*. These expansions were originally introduced as part of the *dpl characterization and search* of ETSs in [4, 5]. Compared to the exhaustive *dpl* search algorithm of [5], the search algorithm proposed here is significantly less complex. There are two reasons for this lower complexity: (a) while the algorithm of [5] is devised to find *all* the ETSs within a certain collection of structures $\mathcal{L}$, the algorithm designed here is intended to check whether *any* instance of *any* structure in $\mathcal{L}$ exists in the Tanner graph. The latter problem is fundamentally different from the former, and is simpler to solve. (b) The exhaustive search algorithm of [5] appears to have a lot of redundancy, in the sense that the same child structure can be searched for through multiple parents with different expansions, rather than through only one parent. This was due to the large variety of ETS structures in irregular graphs which made it difficult to establish such one-to-one correspondence between children and parents. In this work, however, by careful inspection of parent/child relationships between different ETS structures, we establish such correspondences. Through these relationships/correspondences, we then find a minimal set of structures $\mathcal{L}_t$ within $\mathcal{L}$ that need to be considered, and subsequently, a minimal set of structures outside $\mathcal{L}$ that are parents to the structures in $\mathcal{L}_t$. To further reduce the complexity, we implement the search in multiple layers, where the structures of smaller size are searched for earlier. To

the best of our knowledge, [63] is the first to construct irregular QC-LDPC codes by direct elimination of TSs. We demonstrate by a number of examples that the codes designed by the proposed technique are significantly superior to the existing irregular QC-LDPC codes with similar degree distributions, rates and block lengths.

For NB Tanner graphs, we first consider the problem of characterization and search of *non-binary elementary trapping sets* (NB-ETSs) which are the main problematic structures in the error floor region over the AWGN channel [14]. To the best of our knowledge, this work is the first study that attempts to find the graphical structures in the NB Tanner graphs directly and with minimum complexity. To perform this task, we generalize the *dpl*-based method of [4, 5] to the NB case for regular and irregular LDPC codes. Thus, the main idea is to generate an structure by applying a sequence of three different expansions referred to *dot, path* and *lollipop* and checking the algebraic condition after each expansion. However, the main issue is the complexity of search algorithm caused by finding and evaluating non-binary cycles through checking algebraic condition. The proposed method aims at optimizing the search complexity of finding NB-ETSs where the minimum number of cycles are evaluated to check whether an expansion can generate a NB child within the search process. Since the algebraic condition only impacts the cycle of a Tanner graph, the proposed search algorithm can be easily extended to irregular NB-LDPC codes. In this thesis, The distribution of NB trapping sets for a number of existing codes is reported for the first time. The lower complexity of proposed method enables us to find the trapping sets within a large range of interest. Furthermore, similar to what we did for binary QC-LDPC codes in [62, 63], we modified the proposed search algorithm to design NB-QC-LDPC codes with good performance in the error floor region. To achieve this goal, NB-QC-LDPC codes free of a certain collection of harmful structures are constructed. It is worth to mention that our method is applicable to both regular and irregular codes. Also, our proposed algorithm uses trapping sets instead of absorbing sets which are relevant structures for AWGN channels.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows. Basic definitions, notations and backgrounds are provided in Chapter **??**. In Chapter **??**, we considered the problem of constructing binary regular QC-LDPC codes lifted from fully-connected base graphs

that are free of a certain collection of trapping sets. We also study the graphical structures of leafless elementary trapping sets in variable-regular QC-LDPC codes and show that some structures are absent due to QC structure of these codes. In Chapter **??**, we study the design of binary irregular QC-LDPC codes with good waterfall performance and low error floor. We devise an efficient algorithm to find and remove a targeted set of trapping sets corresponding to a range of interest. In Chapter **??**, we study the non-binary elementary trapping sets in both regular and irregular NB-LDPC code and devise an efficient algorithm to find them exhaustively and with minimum complexity. Also, the construction of NB codes free of certain collection of harmful trapping sets is addressed in Chapter **??**. Conclusion and future works are presented in Chapter **??**.

Chapter 2

Preliminaries

2.1 QC-LDPC Codes

Consider an undirected bipartite graph $G'(V' = U' \cup W', E')$, where V' and E' are the sets of nodes and edges of G', respectively, and U' and W' are the sets of nodes on the two sides of the bipartition. Suppose that $|U'| = n$ and $|W'| = m$. In this work, we consider bipartite graphs with no parallel edges. Suppose the graph $G(V = U \cup W, E)$ is constructed from the bipartite graph $G'(V' = U' \cup W', E')$ through the following process: Make N copies of G'. Corresponding to every node $v' \in V'$ and every edge $e' \in E'$, generate a set of nodes $\mathbf{v} = \{v'^0, ..., v'^{N-1}\}$ and a set of edges $\mathbf{e} = \{e'^0, ..., e'^{N-1}\}$. Assign a circular permutation $\pi^{e'}$ over the set $\{0, 1, \ldots, N-1\}$ to each edge e' in E', and connect the nodes in V by the edges in E such that if $e' = \{u', w'\}$, for $u' \in U'$ and $w' \in W'$, is in G', then $\{u'^i, w'^j\}$ belongs to G if and only if $\pi^{e'}(i) = j$. The graph G so constructed is referred to as a *cyclic N-lifting* of G', with N called the *lifting degree*. Graph G', on the other hand, is called the *base graph* or *protograph*.

Now, consider an LDPC code whose Tanner graph is G with the set of variable and check nodes equal to U and W, respectively. Such an LDPC code is a protograph-based QC-LDPC code whose $mN \times nN$ parity-check matrix H, given by the bi-adjacency matrix of G, has the following form:

$$H = \begin{bmatrix} I^{p_{00}} & I^{p_{01}} & \cdots & I^{p_{0(n-1)}} \\ I^{p_{10}} & I^{p_{11}} & \cdots & I^{p_{1(n-1)}} \\ \vdots & \vdots & \ddots & \vdots \\ I^{p_{(m-1)0}} & I^{p_{(m-1)1}} & \cdots & I^{p_{(m-1)(n-1)}} \end{bmatrix}. \tag{2.1}$$

In (**??**), the parameters p_{ij}, called *permutation shifts*, are in the set $\{0, 1, \cdots, N-1, \infty\}$ for $0 \leq i \leq m-1$, $0 \leq j \leq n-1$. The matrix $I^{p_{ij}}$ is obtained by cyclically shifting the rows of the identity matrix $I_{N\times N}$ to the left by p_{ij} units, if $p_{ij} \neq \infty$. Otherwise, I^{∞} denotes the $N \times N$ all-zero matrix. The collection of permutation shifts p_{ij} as a matrix is denoted by $P = [p_{ij}]$ and is called *exponent matrix*.

A QC-LDPC code is called *regular*, if all variable nodes (check nodes) in the code's Tanner graph have the same degree d_v (d_c). Otherwise, the code is called *irregular*. We use notations $\delta(G)$ ($\Delta(G)$) to denote the minimum (maximum) variable node degree in an irregular Tanner graph G. Note that here we focus on binary QC-LDPC codes.

Consider protograph-based QC-LDPC codes whose base graphs are fully-connected, i.e., every node on each side of the graph is connected to all the nodes on the other side. For such base graphs, all the nodes in U' have the same degree $d_v = m$ and all the nodes in W' have the same degree $d_c = n$. Parameters d_v and d_c are the variable and check node degrees of the constructed regular QC-LDPC code. It is well-known that for any QC-LDPC code with a fully-connected base graph, there exists an isomorphic QC-LDPC code with the exponent matrix in the following form (see, e.g., [64]):

$$P = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & p_{11} & \cdots & p_{1(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & p_{(m-1)1} & \cdots & p_{(m-1)(n-1)} \end{bmatrix}, \tag{2.2}$$

where $0 \leq p_{11} \leq \cdots \leq p_{1(n-1)}$.

Given the exponent matrix of a QC-LDPC code, the necessary and sufficient condition for having a $2l$-cycle in the Tanner graph is [64, 65]:

$$\sum_{i=0}^{l-1} (p_{m_i n_i} - p_{m_i n_{i+1}}) = 0 \bmod N, \tag{2.3}$$

where, $n_0 = n_l$, $m_i \neq m_{i+1}$, $n_i \neq n_{i+1}$. The sequence of permutation shifts in (**??**) corresponds to a tailless backtrackless closed (TBC) walk [64] in the base graph whose permutation shift is equal to the left hand side of (**??**). An additional condition for having a $2l$-cycle in the Tanner graph is that the TBC walk corresponding to (**??**)

has no TBC subwalk whose permutation shift is also equal to zero. The length of the shortest cycle(s) in a Tanner graph is called *girth*, and is denoted by g. It is well-known that for good performance 4-cycles should be avoided and that codes with larger girth generally perform better both in waterfall and error floor regions. In our constructions, we impose a lower bound on g. When the goal is to find a QC-LDPC code with $g \geq g_0$, we choose p_{ij} values in (**??**) such that (3) is not satisfied for any $l < g_0/2$, and any sequence of $2l$ permutation shifts.

2.2 Trapping Sets

It is well-known that certain substructures of Tanner graphs are responsible for the error floor of LDPC codes. These substructures are generally referred to as *trapping sets*. A trapping set S is often characterized by its size (the number of variable nodes) $|S| = a$ and the number of unsatisfied (odd-degree) check nodes b in its induced subgraph. Such a trapping set is said to belong to the (a, b) *class*. Among trapping sets, *elementary trapping sets (ETS)*, whose subgraphs have only degree-1 and degree-2 check nodes, are known to be the most problematic ones [4], [5]. Within ETS category, *leafless ETSs (LETS)*, those in which each variable node is connected to at least two satisfied (even-degree) check nodes, are the most harmful [4], [44]. If there is at least one variable node in the ETS S that is connected to only one degree-2 check node, then S is called an ETS with leaf or ETSL. ETSs are known to be the culprits in the error floor region of irregular LDPC codes over the AWGN channel, with the majority of errors being LETSs [5].

Another way of categorizing trapping sets is by tracking the error positions at the output of decoder throughout iterations. As a result the trapping sets may classified into three classes; (i) fixed TSs in which the output of the decoder remain unchanged after a finite number of iterations, (ii) Oscillatory TSs where the error positions at the output of the decoder oscillate periodically within a small set of variable nodes, and (iii) random-like (chaotic) TSs that the error positions change with iterations in a random fashion [89].

In recent years, a number of characterizations, and correspondingly, exhaustive search algorithms for ETSs and LETSs have been developed [4,5,62,66,67]. In [66,67], a LETS structure was characterized as an embedded sequence of structures that starts from a simple cycle or a small *prime* structure, and is expanded one variable

node at a time until it reaches the target structure. This was dubbed as *layered superset (LSS)* characterization. The expansion by the addition of one variable node was subsequently called *dot* expansion in [4]. In [4], two new expansions of *path* and *lollipop* were also introduced, and a new characterization of LETS structures in variable-regular [1] Tanner graphs, called *dpl*, was proposed that was based on the three expansions *dot*, *path* and *lollipop* for general LDPC codes. The *dpl* characterization was then extended in [5] to LETS structures and ETS structures with leafs (ETSLs) of irregular codes. The goal in all the papers [4, 5, 66, 67] has been to develop an efficient algorithm that can exhaustively find all (a, b) LETSs or (a, b) ETSs within a certain range of $a \leq a_{\max}$ and $b \leq b_{\max}$.

Consider the induced subgraph $G(S)$ of an ETS S in a Tanner graph G. Replace any degree-2 check node and its adjacent edges in $G(S)$ with a single edge, and remove all the degree-1 check nodes from $G(S)$. The resulting hypergraph is called the *quasi-normal* hypergraph of S [5], and is denoted by $\check{G}(S)$ in this work. If one also removes the edges adjacent to degree-1 check nodes of $G(S)$ from $\check{G}(S)$, the resulted graph is called the *normal graph* of S [61], and is denoted by $\hat{G}(S)$ here. As an example, Fig. **??** shows a $(5, 4)$ LETS structure along with its normal graph and quasi-normal hypergraph in an irregular Tanner graph G with $\delta(G) = 2$ and $\Delta(G) = 4$ (symbols $\circ$, $\square$, and $\blacksquare$ represent variable nodes, satisfied check nodes, and unsatisfied check nodes, respectively).

While there is a one-to-one-correspondence between $\hat{G}(S)$ and S in variable-regular graphs, such correspondence does not, in general, exist in irregular graphs, i.e., in irregular graphs, multiple non-isomorphic structures can have the same normal graph. For irregular graphs, however, there is a one-to-one correspondence between S and $\check{G}(S)$. When a number of non-isomorphic ETSs, with different quasi-normal hypergraphs, have the same normal graph, we say that those quasi-normal hypergraphs are *projected* into the normal graph.

2.3 Non-Binary Tanner Graphs

Based on the definition of bipartite graph in Section **??**, a variable node and a check node can either be connected or disconnected. Thus, in the binary Tanner graphs,

[1] An LDPC code is called *variable-regular*, if all the variable nodes in the Tanner graph of the code have the same degree.

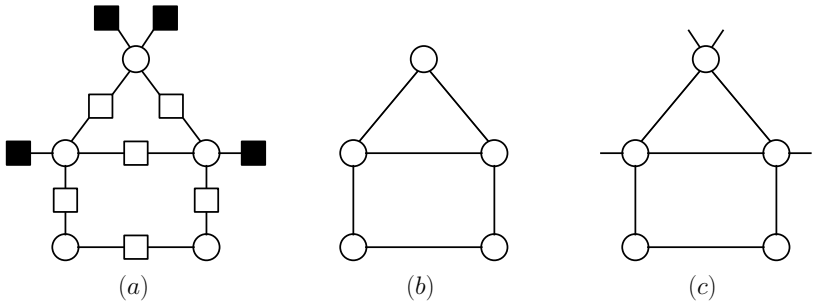


Figure 2.1: (a) A $(5, 4)$ LETS structure S in an irregular Tanner graph G with $\Delta(G) = 4$ and $\delta(G) = 2$, (b) $\hat{G}(S)$, (c) $\check{G}(S)$.

the edge weights are selected form set $\{0, 1\}$ or equivalently finite field of size 2. Binary Tanner graphs can be extended to non-binary (NB) where each edge takes a non-zero value from the finite field with q elements (GF(q)). Thus, the graph can be represented as $G'(V', E', S')$ where S' is a set of non-zero edge weights corresponding to the set of edges E'.

Now, consider an LDPC code whose Tanner graph is G with the set of variable and check nodes equal to U and W, respectively. Also, suppose that the edge weights belong to $GF(q)$ where $q = 2^p$ and $p \geq 1$. Such an LDPC code is a protograph-based non-binary QC-LDPC (NB-QC-LDPC) code whose $mN \times nN$ parity-check matrix H, given by the bi-adjacency matrix of G, has the following form:

$$H = \begin{bmatrix} s_{00}I^{p_{00}} & s_{01}I^{p_{01}} & \cdots & s_{0(n-1)}I^{p_{0(n-1)}} \\ s_{10}I^{p_{10}} & s_{11}I^{p_{11}} & \cdots & s_{1(n-1)}I^{p_{1(n-1)}} \\ \vdots & \vdots & \ddots & \vdots \\ s_{(m-1)0}I^{p_{(m-1)0}} & s_{(m-1)1}I^{p_{(m-1)1}} & \cdots & s_{(m-1)(n-1)}I^{p_{(m-1)(n-1)}} \end{bmatrix}. \tag{2.4}$$

In (**??**), the parameters s_{ij} belong to the set $\{\alpha^k : k = 0, 1, \cdots, q-2\}$ for α being a primitive element of $GF(q)$ with $q > 2$, respectively. One can form matrix $S = [s_{ij}]$ containing edge weights s_{ij} of different sub-blocks of parity-check matrix referred to as *edge weight matrix*. Obviously, a NB-QC-LDPC code can be specified by its exponent (P) and edge weight (S) matrices. A NB-QC-LDPC code is called regular if the degree of all variable nodes and check nodes are the same. Otherwise, the Tanner graph of the code is called irregular. In this thesis, wherever we do not mention

anything about the edge weights of the matrix, the Tanner graph and LDPC code are supposed to be binary.

Chapter 3

Construction of Regular QC-LDPC Codes with Low Error Floor

3.1 Introduction

In this chapter, the ETS characterization for binary QC-LDPC codes is discussed in Section **??**. Section **??** describes the approach for determining the ETSs within a given collection that are targeted for elimination based on parent/child relationships within the collection, and the proposed layered search algorithm for finding the targeted ETSs efficiently. In Section **??**, we present the method for the construction of QC-LDPC codes with low error floors based on the proposed layered search algorithm. Section **??** is devoted to numerical results and presents some of the constructed codes and comparisons with existing codes in the literature.

3.2 Characterization of ETS Structures in QC-LDPC codes

In this section, we investigate the constraints that the QC structure of LDPC codes imposes on the ETS structures. We demonstrate that as a result of such constraints, certain ETS structures that can appear in (randomly constructed) LDPC codes, cannot exist in similar QC-LDPC codes. To show this, we transform the problem into a graph coloring problem.

To investigate the constraints imposed by the QC structure of the codes, in the following, we work with the normal graph representation of ETSs. We note that for

Tanner graphs with girth at least 6, the normal graph of any ETS is simple, i.e., has no parallel edges.

For a graph $\mathcal{G}$, a *k-edge coloring* is defined as a function $f : E(\mathcal{G}) \to C$, such that $|C| = k$, and $f(e) \neq f(e')$ for any two adjacent edges e and e' of $\mathcal{G}$. A graph $\mathcal{G}$ is *k-edge colorable* if $\mathcal{G}$ has a k-edge coloring. The *chromatic index* of $\mathcal{G}$, denoted by $\chi(\mathcal{G})$, is the minimum value of k for which $\mathcal{G}$ has a k-edge coloring. It is well-known that for a simple graph $\mathcal{G}$, $\Delta(\mathcal{G}) \leq \chi(\mathcal{G}) \leq \Delta(\mathcal{G}) + 1$, where $\Delta(\mathcal{G})$ is the maximum node degree of $\mathcal{G}$ [68]. A graph $\mathcal{G}$ is said to be of Class 1 (resp., Class 2) if $\chi(\mathcal{G}) = \Delta(\mathcal{G})$ (resp., $\chi(\mathcal{G}) = \Delta(\mathcal{G}) + 1$).

Proposition 1. *Consider a QC-LDPC code with the parity-check matrix given by (??). For an ETS S to exist in the code, the normal graph $\hat{G}(S)$ must be m-edge colorable.*

Proof. It is clear that for protograph-based QC-LDPC codes with the parity-check matrix H of the form (??), each variable node in an ETS S can be connected to at most m check nodes, where each such check node must belong to a distinct row block of H. Now, suppose that we assign m different colors to different row blocks of H, and consider the following assignment of colors to the edges of $\hat{G}(S)$: for each edge e in $\hat{G}(S)$, find the row block $\mathcal{R}$ of H corresponding to the degree-2 check node in $G(S)$ that has been replaced by e. Then, assign the color of $\mathcal{R}$ to e. It is easy to see that for an ETS S to exist in a QC-LDPC code, the aforementioned color assignment must be an m-edge coloring of $\hat{G}(S)$, or in other words, $\hat{G}(S)$ must be m-edge colorable. ■

We thus have the following result.

Corollary 1. *Consider a QC-LDPC code with the parity-check matrix given by (??). An ETS S cannot exist in the Tanner graph of the code if $\chi(\hat{G}(S)) > m$.*

We note that the result of Corollary ?? is applicable to any QC-LDPC code with the parity-check matrix given by (??). This includes cases in which some of the sub-matrices of (??) are all-zero, and cases where the degree distribution is irregular. In the latter case, there is no one-to-one correspondence between the normal graph $\hat{G}(S)$ of an ETS S and S.

Corollary 2. *An ETS S whose normal graph $\hat{G}(S)$ is of Class 2 with $\Delta(\hat{G}(S)) = m$ cannot exist in a QC-LDPC code with $g \geq 6$ and the parity-check matrix given by (??).*

A graph $G = (V, E)$ is called *overfull* if $|E| > \lfloor \frac{|V|}{2} \rfloor \times \Delta(G)$. It is known that an overfull graph is of Class 2 [68]. We thus have the following result.

Proposition 2. *Any (a, b) ETS with an odd value of a and with $b < \min(a, d_v)$ cannot exist in a variable-regular QC-LDPC code with variable degree d_v and $g \geq 6$, that is a cyclic lifting of a fully-connected base graph.*

Proof. We first note that for an ETS S with $b < a$, we have $\Delta(\hat{G}(S)) = d_v$, because there is at least one variable node in $G(S)$ that is not connected to any degree-1 check nodes. We then show that under the conditions of the proposition, the graph $\hat{G}(S)$ is also overfull and thus of Class 2, This, based on Corollary 2, completes the proof. To show that $\hat{G}(S)$ is overfull, we note that the number of edges in $\hat{G}(S)$ is $|E| = (ad_v - b)/2$, by the definition of a normal graph. Since $b < d_v$, we have $|E| > (a-1)d_v/2 = \lfloor a/2 \rfloor d_v$, where the equality is a result of a being odd. ■

We note that the results of Corollaries **??**, **??** and Proposition **??** hold regardless of the lifting degree and permutation shifts of the QC-LDPC code.

Using Proposition **??**, we can conclude that some of the ETS classes that can generally exist in (randomly constructed) variable-regular LDPC codes will not appear in similar QC-LDPC codes. As an example, consider a variable-regular LDPC code with $d_v = 3$ and $g = 6$. Based on Table VI of [4], this code can have LETSs in classes $(5, 1)$, $(7, 1)$ and $(9, 1)$. The multiplicity of non-isomorphic structures in these classes are 1, 4 and 19, respectively. Based on Proposition **??**, however, none of these structures can possibly exist in a QC-LDPC code, lifted from a fully-connected base graph, with similar d_v and g values.

It is important to note that while one can use the vast literature on the edge coloring of graphs and derive analytical results similar to Proposition **??** for other classes or structures of ETSs, for practical purposes, one can also use an algorithm for finding the chromatic index of a graph, such as that of [69], to examine different ETS structures and see if their chromatic index is larger than d_v (i.e., it is $d_v + 1$). Clearly, a "yes" answer would mean that such a structure cannot exist in a variable-regular QC-LDPC code with variable degree d_v.

In Table **??**, we have listed all the classes in which at least one LETS structure does not exist in the QC category of the corresponding variable-regular LDPC codes. Similar to [4], in Table **??**, the results are separated based on the values of d_v and g. For each value of d_v and g, and for each class with at least one missing structure,

we have provided the multiplicity of the non-isomorphic structures within the class as the bottom entries, where the right and left entries are for the general and the QC cases, respectively. For the entries, we have followed the same notation as in [4], where the number in the brackets shows the multiplicity of the structures and the notation s_k indicates the simple cycle parent of those structures which has a length of $2k$. As an example, for $d_v = 3$ and $g = 6$, the bottom entries for the $(5, 1)$ class are $s_3(0)/s_3(1)$, which means while a general LDPC code with $d_v = 3$ and $g = 6$ can have one structure within this class with the parent being a 6-cycle, this structure cannot exist in the QC category. The boldfaced entries in the table highlight the classes for which all the structures are non-existent for the QC category. The non-existence of all these classes follow from Proposition **??**.

The non-existence of some LETS structures in QC-LDPC codes compared to their random counterparts not only reduces the search complexity of the proposed technique, as discussed in Subsection **??**, but also can be potentially beneficial in achieving a lower error floor.

We note that edge coloring was also used in [26] to show that some ETS structures cannot exist in certain QC-LDPC codes. In particular, Proposition 1 of [26] is similar to Proposition **??** in this work except that Proposition 1 of [26] is limited only to fully-connected base graphs.[1] Proposition **??** of this work, however, is also applicable to base graphs that are not fully-connected, and covers base graphs that are both regular and irregular. The only other results in [26] related to the application of edge coloring to the existence of ETSs in QC-LDPC codes are Example 1 and Corollary 1 of [26]. Example 1 of [26] indicates that a cyclic lifting of a $2\ell \times n$ fully-connected base graph ($d_v = 2\ell$) whose girth is 6 cannot contain any $(2\ell + 1, 0)$ LETS. Corollary 1 of [26] indicates that under the same conditions, $(2\ell + 1, 2)$ LETSs also cannot exist. Both these results are special cases of Proposition **??** in this work. In addition to these two special cases, Proposition **??** covers numerous other cases, which include odd variable degrees, classes of ETSs with the size different than just $d_v + 1$ and with b values other than just 0 or 2, and codes with girths larger than 6.

In the rest of this section, similar to the existing literature [4, 29, 48, 61], we focus on LETSs as the main problematic structures in the error floor. This is due to the fact that a vast majority of trapping sets of variable-regular LDPC codes are known to be LETSs , see, e.g., [4]. In the following, whenever we discuss QC-LDPC codes,

[1] No proof is provided in [26] for Proposition 1 of [26].

we exclusively consider protograph-based QC-LDPC codes whose base graphs are fully-connected. We then use the results presented in Table **??** to reduce the number of structures that we need to search for in the process of constructing such codes.

3.3 Efficient Search Algorithm for A Targeted Set of LETSs

The goal of this chapter is to construct QC-LDPC codes free of a certain collection $\mathcal{L}$ of LETS structures.[2] In the literature, this collection is often identified by a certain range of a and b values, i.e., $a \leq a_{max}$ and $b \leq b_{max}$. As we will discuss in Section **??**, to construct the exponent matrix of a QC-LDPC code (for a given lifting degree), a greedy column-by-column search algorithm is used to assign the permutation shifts. In the search process, after assigning the permutation values of a new column, one needs to check whether the Tanner graph corresponding to the exponent matrix constructed so far contains any instances of the LETS structures within $\mathcal{L}$. A naive approach to perform this task would be to use the exhaustive *dpl* search of [4] within the specified range, and see if the algorithm can find any LETSs within the range. This approach, however, is too complex to use in the construction process that may require hundreds or even thousands of such searches. Moreover, the information provided by the *dpl* algorithm of [4] is much more than what we need in the construction process, i.e., the algorithm provides an exhaustive list of all LETSs within the range of interest. However, what we need is just to know whether there exists at least one instance of one of the structures of $\mathcal{L}$ within the graph. In this section, we devise an efficient search algorithm for this problem. The efficiency of the algorithm is a result of the following considerations: (a) the algorithm only searches for a minimal number of structures within $\mathcal{L}$. These targeted structures, denoted by $\mathcal{L}_t$, are the ones that are not child to any of the other structures in $\mathcal{L}$, and (b) the algorithm aims to minimize the number of LETS structures that do not belong to $\mathcal{L}$ but are needed in the search for the structures in $\mathcal{L}_t$. For the choice of these out-of-range structures, the algorithm also gives priority to LETS classes that have a higher chance of having smaller multiplicities in the graph. Our search algorithm

[2] The proposed layered characterization/search algorithm of LETSs can also be used to construct LDPC codes with low error floor that lack the QC structure. The only difference is that for such codes the number of possible LETS structures is larger.

Table 3.1: LETS classes in which at least one structure is non-existent for LDPC codes with QC structure lifted from a fully-connected base graph (notation $s_k(i)$ as an entry of a class means that there are i non-isomorphic structures in the class whose parent is a simple cycle of length $2k$. Right and left entries are for the general and the QC cases, respectively).

$d_v = 3,\ g = 6$	$(\mathbf{5,1})$ $---$ $\mathbf{s_3(0)/s_3(1)}$	$(\mathbf{7,1})$ $---$ $\mathbf{s_3(0)/s_3(3)}$ $\mathbf{s_4(0)/s_4(1)}$	$(8,2)$ $---$ $s_3(13)/s_3(14)$	$(\mathbf{9,1})$ $---$ $\mathbf{s_3(0)/s_3(15)}$ $\mathbf{s_4(0)/s_4(4)}$	$(9,3)$ $---$ $s_3(42)/s_3(44)$ $s_5(1)/s_5(2)$
	$(10,0)$ $---$ $s_3(12)/s_3(13)$ $s_5(0)/s_5(1)$	$(10,2)$ $---$ $s_3(77)/s_3(85)$ $s_5(0)/s_5(1)$	$(10,4)$ $---$ $s_3(126)/s_3(129)$	$(\mathbf{11,1})$ $---$ $\mathbf{s_3(0)/s_3(91)}$ $\mathbf{s_4(0)/s_4(22)}$ $\mathbf{s_5(0)/s_5(1)}$	$(11,3)$ $--- \ s_3(337)/s_3(355)$ $s_4(117)/s_4(120)$ $s_5(6)/s_5(7)$
	$(11,5)$ $---$ $s_3(324)/s_3(328)$	$(12,0)$ $---$ $s_3(58)/s_3(63)$	$(12,2)$ $---$ $s_3(584)/s_3(641)$ $s_4(180)/s_4(184)$ $s_5(8)/s_5(10)$	$(12,4)$ $---$ $s_3(1279)/s_3(1315)$ $s_4(521)/s_4(524)$ $s_5(50)/s_5(52)$	
$d_v = 3,\ g = 8$	$(\mathbf{7,1})$ $---$ $\mathbf{s_4(0)/s_4(1)}$	$(\mathbf{9,1})$ $---$ $\mathbf{s_4(0)/s_4(4)}$	$(9,3)$ $---$ $s_5(0)/s_5(1)$	$(10,0)$ $---$ $s_5(0)/s_5(1)$	$(10,2)$ $---$ $s_5(0)/s_5(1)$
	$(\mathbf{11,1})$ $---$ $\mathbf{s_4(0)/s_4(22)}$ $\mathbf{s_5(0)/s_5(1)}$	$(11,3)$ $---$ $s_4(114)/s_4(115)$ $s_5(6)/s_5(7)$	$(12,2)$ $---$ $s_4(178)/s_4(179)$ $s_5(9)/s_5(11)$	$(12,4)$ $---$ $s_4(479)/s_4(481)$ $s_5(47)/s_5(48)$	
$d_v = 4,\ g = 6$	$(\mathbf{5,0})$ $---$ $\mathbf{s_3(0)/s_3(1)}$	$(\mathbf{5,2})$ $---$ $\mathbf{s_3(0)/s_3(1)}$	$(6,2)$ $---$ $s_3(2)/s_3(3)$	$(\mathbf{7,0})$ $---$ $\mathbf{s_3(0)/s_3(2)}$	$(\mathbf{7,2})$ $---$ $\mathbf{s_3(0)/s_3(9)}$
	$(8,2)$ $---$ $s_3(27)/s_3(34)$	$(8,4)$ $---$ $s_3(120)/s_3(122)$	$(8,6)$ $---$ $s_3(222)/s_3(224)$	$(\mathbf{9,0})$ $---$ $\mathbf{s_3(0)/s_3(16)}$	$(\mathbf{9,2})$ $---$ $\mathbf{s_3(0)/s_3(152)}$ $\mathbf{s_4(0)/s_4(2)}$
	$(9,4)$ $---$ $s_3(642)/s_3(656)$	$(9,6)$ $---$ $s_3(1352)/s_3(1360)$	$(9,8)$ $---$ $s_3(1558)/s_3(1561)$	$(10,0)$ $---$ $s_3(56)/s_3(57)$	$(10,2)$ $---$ $s_3(709)/s_3(840)$
	$(10,4)$ $---$ $s_3(4106)/s_3(4140)$	$(10,6)$ $---$ $s_3(9334)/s_3(9382)$	$(10,8)$ $---$ $s_3(11698)/s_3(11719)$	$(10,10)$ $---$ $s_3(8763)/s_3(8767)$	
$d_v = 4,\ g = 8$	$(\mathbf{9,2})$ $---$ $\mathbf{s_4(0)/s_4(2)}$	$(\mathbf{11,0})$ $---$ $\mathbf{s_4(0)/s_4(2)}$	$(\mathbf{11,2})$ $---$ $\mathbf{s_4(0)/s_4(19)}$	$(11,4)$ $---$ $s_4(163)/s_4(164)$	
$d_v = 5,\ g = 6$	$(\mathbf{7,1})$ $---$ $\mathbf{s_3(0)/s_3(1)}$	$(\mathbf{7,3})$ $---$ $\mathbf{s_3(0)/s_3(6)}$	$(8,2)$ $---$ $s_3(13)/s_3(16)$	$(8,4)$ $---$ $s_3(68)/s_3(75)$	$(\mathbf{9,1})$ $---$ $\mathbf{s_3(0)/s_3(28)}$
	$(\mathbf{9,3})$ $---$ $\mathbf{s_3(0)/s_3(289)}$	$(9,5)$ $---$ $s_3(1350)/s_3(1356)$	$(9,7)$ $---$ $s_3(3776)/s_3(3786)$	$(9,11)$ $---$ $s_3(9526)/s_3(9527)$	

still follows the general framework of *dpl* characterization/search of [4], and thus has all the advantages of the *dpl* technique. This means that in the proposed search algorithm, each LETS is characterized (and searched for) as an embedded sequence of LETS structures that starts from a simple cycle and is expanded, at each step, by one of the three expansions *dot*, *path* and *lollipop* until it reaches the LETS structure of interest. (For a review of these expansions and the corresponding notations, which we closely follow, the reader is referred to [4].)

3.3.1 Finding the Target LETSs $\mathcal{L}_t$

Suppose that we are interested in constructing a QC-LDPC code free of LETSs in the range of $a \leq a_{max}$ and $b \leq b_{max}$. To perform this task, we first identify all the non-isomorphic LETS structures within this range based on the values of d_v and g, and investigate the parent/child relationship between all such structures. (See [4].) We then start from the smallest size of LETSs in the range, i.e., $a = g/2$, and identify all the non-isomorphic $(g/2, b)$ LETS structures with $b \leq b_{max}$. These structures are stored in $\mathcal{L}_t$. We then go through an iterative process, where in each iteration, we increase a by one, until we reach a_{max}. In each iteration, we examine all the (a, b) LETS structures with the specific a value of that iteration, and b values in the range $b \leq b_{max}$. If there is any such structure that is not a child of previously stored structures in $\mathcal{L}_t$, we add that structure to $\mathcal{L}_t$. By the way that the list $\mathcal{L}_t$ is constructed, it is clear that it contains the minimum number of structures in $\mathcal{L}$ that need to be targeted for elimination such that none of the LETS structures in $\mathcal{L}$ can exist in the code. (Elimination of a parent guarantees the elimination of all its children.) The following is an example of finding $\mathcal{L}_t$.

Example 1. *In this example, we consider a QC-LDPC code construction with* $d_v = 3$, $g = 8$, *and four different ranges of interest* $r_1 : a \leq 6, b \leq 3$, $r_2 : a \leq 8, b \leq 3$, $r_3 : a \leq 10, b \leq 3$, *and* $r_4 : a \leq 12, b \leq 3$. *The parent/child relationships between non-isomorphic LETS structures within the largest range* r_4 *are shown in Fig.* **??**. *In Fig.* **??**, *the direct (resp., indirect) children are those that are created from their parents by one expansion (resp., multiple expansions). Different non-isomorphic structures within the same class are identified by different numbers in braces. From Fig.* **??**, *it is easy to see* $\mathcal{L}_t$ *sets for ranges* r_1 *to* r_4 *are* $\mathcal{L}_{t_1} = \{(5, 3)\}$, $\mathcal{L}_{t_2} = \mathcal{L}_{t_1} \cup \{(7, 3)\{1\}, (7, 3)\{2\}\}$, $\mathcal{L}_{t_3} = \mathcal{L}_{t_2} \cup \{(9, 3)\{7, 8, 9, 10, 13, 14, 15, 16, 17\}\}$, *and* $\mathcal{L}_{t_4} = \mathcal{L}_{t_3} \cup \{(11, 3)\{45, \ldots, 48, 50, \ldots, 84, 89, 90, 91, 94, 95, 98, \ldots, 108, 112, 117, \ldots, 122\}\}$,

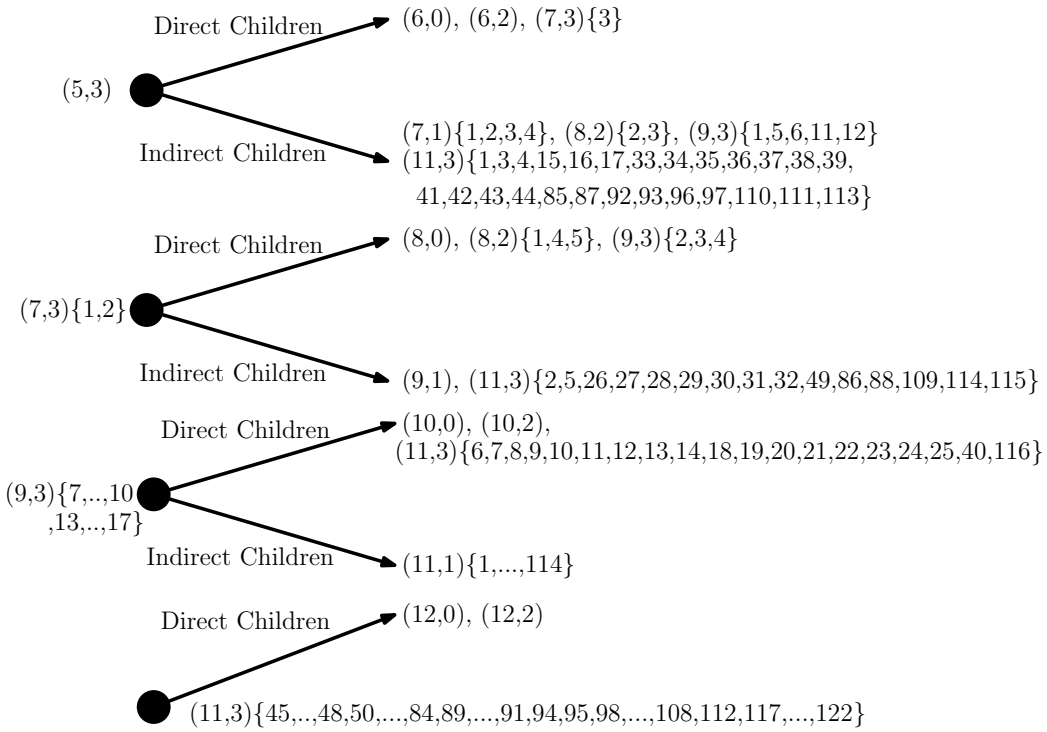


Figure 3.1: Parent/Child relationships between LETS structures of variable-regular QC-LDPC codes with $d_v = 3$, $g = 8$, within the range $a \leq 12$ and $b \leq 3$ (different non-isomorphic structures within the same class are identified by different numbers in braces).

respectively. As can be seen, $|\mathcal{L}_t|$ *is considerably smaller than* $|\mathcal{L}|$ *for each range. For example for* r_4, $|\mathcal{L}_{t_4}| = 74$ *versus* $|\mathcal{L}| = 392$.

3.3.2 Efficient Search Algorithm for LETSs in $\mathcal{L}_t$

To search for all the LETS structures in $\mathcal{L}_t$, we devise a backward recursion that starts from the LETS structures in $\mathcal{L}_t$ with the largest size (largest a). Let such structures be denoted by $\mathcal{S}_1$, and let $\mathcal{S} = \mathcal{S}_1$. Since the structures in $\mathcal{S}$ cannot be reached through any LETS structure within $\mathcal{L}_t$ (or $\mathcal{L}$) using *dpl* expansions, we consider all the possible direct parents $\mathcal{P}$ of such structures outside $\mathcal{L}$. The structures in $\mathcal{P}$ are those that can reach at least one of the structures in $\mathcal{S}$ by the application of just one of the three *dpl* expansions. We then prioritize the classes of the structures in $\mathcal{P}$ according to certain criteria discussed later. Within the class with the highest priority, we then select structures using a greedy iterative process. The process starts

by selecting the structure in $\mathcal{P}$ that has the largest number of direct children in $\mathcal{S}$. Denote this structure by ξ. We move ξ from $\mathcal{P}$ to a set Π_1 (Π_1 is initially empty), and remove all the direct children of ξ from $\mathcal{S}$. This ends the first iteration. The iterations continue until the set $\mathcal{S}$ is empty or until no direct parent of the structures in $\mathcal{S}$ is left in the class with the highest priority. In the former case, the first step of the recursion is completed. In the latter case, we move on to the class with the second highest priority and apply the iterative process. This will continue until the set $\mathcal{S}$ is empty and thus the first step of the recursion is completed. At this point, the set Π_1 contains all the direct parents of the structures in $\mathcal{S}$. To start the next step of recursion, we consider $\mathcal{S} = \Pi_1 \cup \mathcal{S}_2$ as the new set $\mathcal{S}$, where $\mathcal{S}_2$ denotes the LETS structures in $\mathcal{L}_t$ with the second largest size. The steps of recursion will continue until all the LETSs in $\mathcal{L}_t$ are covered. At the end of the recursion, the union of sets Π_i contains all the out-of-range LETS structures that need to be included in the search process. A pseudo-code for the process of obtaining out-of-range parent structures is given in Algorithm **??**.

Now, we discuss the criteria that we use to prioritize the classes of structures in $\mathcal{P}$. The main idea is to give priority to classes that have a higher chance of having a smaller multiplicity in the graph. This translates to a less complex search and smaller memory requirement. Since there is no theoretical result available to predict the multiplicity of different LETS classes within a finite Tanner graph, we rely on empirical results. In general, experimental results show that, for a given a, LETS classes with smaller value of b have smaller multiplicity [4]. The empirical results also show that for two LETS classes (a, b) and (a', b'), where $a < a'$ and $b < b'$, the multiplicity of (a, b) class is generally smaller than that of (a', b') class [4]. Moreover, consider two classes (a, b) and (a', b') of structures in $\mathcal{P}$, where $a < a'$, and let S and S' be two structures in the two classes, respectively. Also suppose that S and S' are direct parents of two structures ξ and ξ', respectively, where both ξ and ξ' are in the same class $(a", b")$ in $\mathcal{S}$. The following lemma (Lemma 1) shows that, based on the *dpl* characterization, we must have $b < b'$. (This implies that when all the structures in $\mathcal{S}$ belong to a single class, then one can easily order the direct parent classes of such structures in $\mathcal{P}$ in accordance with the a or the b value of the classes, with classes of smaller a and b values, or the same a value and smaller b, having a higher priority in the selection process.)

Algorithm 1 Finding the out-of-range parent structures needed for the proposed algorithm to search for LETSs within $\mathcal{L}_t$.

Input: $\mathcal{L}_t$ ▷ Structures in $\mathcal{L}_t$ belong to classes with sizes $a_1, a_2, \ldots, a_\eta$, such that $a_1 < a_2 < \ldots < a_\eta$.
Initialization: $\mathcal{S} = \emptyset$.
for $k = 0, \ldots, \eta - 1$ **do**
    $\Pi_k \leftarrow \emptyset$.
end for
for $i = \eta, \ldots, 1$ **do**
    $\mathcal{S}_{\eta-i+1} \leftarrow$ LETS structures in $\mathcal{L}_t$ with size a_i.
    $\mathcal{S} = \mathcal{S}_{\eta-i+1} \cup \Pi_{\eta-i}$.
    $\mathcal{P} \leftarrow$ all possible direct parents of structures in $\mathcal{S}$. ▷ For each value of i, structures in $\mathcal{P}$ belong to classes $(a_j^i, b_j^i), j = 1, \ldots, \theta_i$, where classes with a smaller index j have a higher priority.
    for $j = 1, \ldots, \theta_i$ **do**
        $\Gamma_j \leftarrow$ structures in $\mathcal{P}$ that are in class (a_j^i, b_j^i).
        while $\Gamma_j \neq \emptyset$ **do**
            Choose $\xi \in \Gamma_j$ with the largest number of direct children in $\mathcal{S}$.
            $\Pi_{\eta-i+1} = \Pi_{\eta-i+1} \cup \{\xi\}$.
            Remove ξ from Γ_j, and remove all the direct children of ξ from $\mathcal{S}$.
            if $\mathcal{S} = \emptyset$ **then**
                Break the **for** loop over variable j.
            end if
        end while
    end for
end for
$\Pi = \Pi_1 \cup \ldots \cup \Pi_\eta$.
Output: Π.

Lemma 1. *Consider variable-regular LDPC codes with variable degree $d_v \geq 3$ and the dpl characterization of LETS structures within such codes. For a given LETS class (a, b), suppose that classes $(a_1, b_1), (a_2, b_2), \ldots, (a_q, b_q)$ are all the direct parent classes of the (a, b) class. For any two such parent classes, if $a_i < a_j$, then $b_i < b_j$.*

Proof. From [4], one can see that direct parent classes of the (a, b) class can be one of the two following classes: $(a-1, b+2m-d_v))$, or $(a-m, b+2-m(d_v-2))$. The first parent class is for dot_m expansion with $2 \leq m \leq d_v$, and the second parent class is for pa_m expansion with $2 \leq m \leq min\{b/(d_v-2), a-g/2-1\}$, and for lo^c_m expansion with $g/2 \leq m \leq min\{(b+1)/(d_v-2), a-g/2-1\}$ and $g/2 \leq c \leq m$.

Since $m \geq 2$, it is clear that the a value for the parent classes corresponding to pa_m and lo^c_m expansions is always smaller than that of the dot_m expansion. So, we first show that the corresponding b values follow the same trend. For this, we prove that the smallest b value for a parent class corresponding to dot_m is larger than the largest b value of a parent class corresponding to pa_m or lo^c_m. In the case of dot_m expansion, the smallest b value of a parent class is $b + 2 \times 2 - d_v = b - d_v + 4$ since $m \geq 2$, and for pa_m or lo^c_m expansion, the largest b value of a parent class is $b + 2 - 2 \times (d_v - 2) = b - 2d_v + 6$, which is smaller than $b - d_v + 4$ for $d_v > 2$.

Considering that the parent classes corresponding to pa_m and lo^c_m expansions are identical, to complete the proof, we just need to show that if $a - m_1 < a - m_2$, for some values of m_1 and m_2, then $b + 2 - m_1(d_v - 2) < b + 2 - m_2(d_v - 2)$, which is clearly true under the condition that $d_v > 2$. ■

Based on the above discussions, to prioritize the structures in $\mathcal{P}$, we first prioritize the classes: classes with smaller a and b values have a higher priority and for the same value of a, classes with smaller b values have a higher priority. Within each class, we then prioritize the structures based on the number of direct children that they have in $\mathcal{S}$, where larger number of children means higher priority.

Example 2. *Consider the code construction discussed in Example* **??**, *where all the LETSs in r_4 are to be avoided. In this case, the set $\mathcal{S}_1$ consists of* 62 *structures in the* $(11, 3)$ *class. The careful study of these structures reveals that all the (out-of-range) direct parents of these structures belong to* $(10, 4)$ *and* $(10, 6)$ *classes, and that the structures in $\mathcal{S}_1$ are generated by the application of dot_2 and dot_3 expansions to these parent structures. Between the two classes of* $(10, 4)$ *and* $(10, 6)$, *the priority is given to the parent structures in the* $(10, 4)$ *class since for the same value of a, this class*

has a smaller b value compared to the $(10, 6)$ *class. In total,* 63 *structures exist in the* $(10, 4)$ *class. By prioritizing these structures based on the number of their direct children and choosing them iteratively in the order of their priority, we can cover all the structures in* $\mathcal{S}_1$ *by choosing only* 22 *structures in the* $(10, 4)$ *class. These structures can generate all the structures in* $\mathcal{S}_1$ *with the* dot_2 *expansion. This ends the first step of the recursion. (see the last level of the trellis diagram in Fig.* **??***. To prevent the figure from being over crowded, rather than making a connection between each parent/child pair, for the last level, we have shown parents and their children collectively.)*

For the second step, we have $\mathcal{S} = \Pi_1 \cup \mathcal{S}_2$*, where* Π_1 *is the set of the* 22 *structures in the* $(10, 4)$ *class and* $\mathcal{S}_2$ *consists of the* 9 *LETS structures in the* $(9, 3)$ *class of* $\mathcal{L}_t$*. Investigating the direct (out-of-range) parent structures for the set* $\mathcal{S}_2$*, we find that they all belong to* $(8, 4)$ *and* $(8, 6)$ *classes, while for the* 22 *structures in the* $(10, 4)$ *class, the direct parent structures are in* $(8, 4)$*,* $(9, 5)$*, and* $(9, 7)$ *classes. Among the direct parent classes of the set* $\mathcal{S}$*, the highest priority is given to the* $(8, 4)$ *class since it has the smallest a and b values. By the application of the iterative selection process to the structures in the* $(8, 4)$ *class, we can generate all the* 9 *structures in the* $(9, 3)$ *class and* 7 *out of* 22 *structures in the* $(10, 4)$ *class, by using only* 7 *structures in the* $(8, 4)$ *class. The remaining* 15 *structures in the* $(10, 4)$ *class, however, cannot be generated by the structures in the* $(8, 4)$ *class. To generate these remaining structures, the parent class with the highest priority (out of the two remaining classes that are direct parent classes of the* $(10, 4)$ *class) is the* $(9, 5)$ *class. By the application of the iterative process to this class, the remaining* 15 *structures of the* $(10, 4)$ *class are generated through only* 5 *structures in the* (9.5) *class. (See the second last level of the trellis diagram in Fig.* **??***.)*

In the next step of the recursion, the 7 *and* 5 *structures from the* $(8, 4)$ *and* $(9, 5)$ *classes, respectively, along with the two* $(7, 3)$ *structures in* $\mathcal{L}_t$ *form the set* $\mathcal{S}$*. The outcome of this step of recursion can be seen in Fig.* **??***, together with the final step of the recursion, that has all the simple cycles of length* 8 *and* 10 *as the output set* Π_4*.*

Corresponding to the trellis diagram of Fig. **??***, we have the characterization table of LETSs for the proposed search, given at the top of Table* **??***, in comparison with the characterization table corresponding to the exhaustive search of [4] (limited to structures that exist in QC-LDPC codes, as explained in Section* **??***), given at the*

bottom. The entries of the table correspond to different classes of LETSs. For each class, the top entry shows the non-isomorphic structures within the class that are involved in the search process and the bottom entry shows the expansions that will have to be applied to all the instances of those structures. By comparing the entries for similar classes in the top and the bottom parts of Table **??***, one can see that both the number of structures and the variety of required expansions have decreased substantially in the proposed search algorithm compared to the exhaustive search of [4]. As an example, for the* $(9, 5)$ *class, the proposed search algorithm only needs to apply* dot_2 *expansion to* 5 *structures. In comparison, the exhaustive search algorithm of [4] requires the application of* dot_2, dot_3 *and* pa_2 *expansions to* 20 *structures.*

3.3.3 Layering of the Search Algorithm

To further reduce the complexity and memory requirement of our proposed search algorithm, we implement the algorithm in multiple layers. We recall that the purpose of our search is to determine whether there exists at least one instance of one of the structures of $\mathcal{L}$ in the graph. Consider the structures in $\mathcal{L}_t$, and assume that they belong to classes with size $a_1, a_2, \ldots, a_\eta$, where $a_1 < a_2 < \ldots < a_\eta$. To make the determination that whether there exists at least one instance of one of the structures of $\mathcal{L}$ in the graph, rather than searching for all the structures in $\mathcal{L}_t$ in one shot, we first search for the structures in the class with size a_1 (Layer 1). If we can find at least one instance of a structure in this class, we terminate the search with a positive response. If the graph has none of the structures of $\mathcal{L}_t$ with size a_1, we continue our search to Layer 2, which contains structures in $\mathcal{L}_t$ with size a_2. We continue this process, until we either find one instance of one structure within one of the layers, or finish the search through all layers without finding any LETS in $\mathcal{L}_t$. In the former case, the search is terminated with a positive response, while in the latter case, the response is negative.

As an example, the four layers of the proposed search algorithm for the construction discussed in Examples 1 and 2, corresponding to classes $(5, 3)$, $(7, 3)$, $(9, 3)$ and $(11, 3)$ in $\mathcal{L}_t$ are identified by different line types in Fig. **??**.

Remark 1. *For further memory reduction, in each layer of the search algorithm, we only keep those LETSs that are needed for next layer(s) of our search algorithm. For*

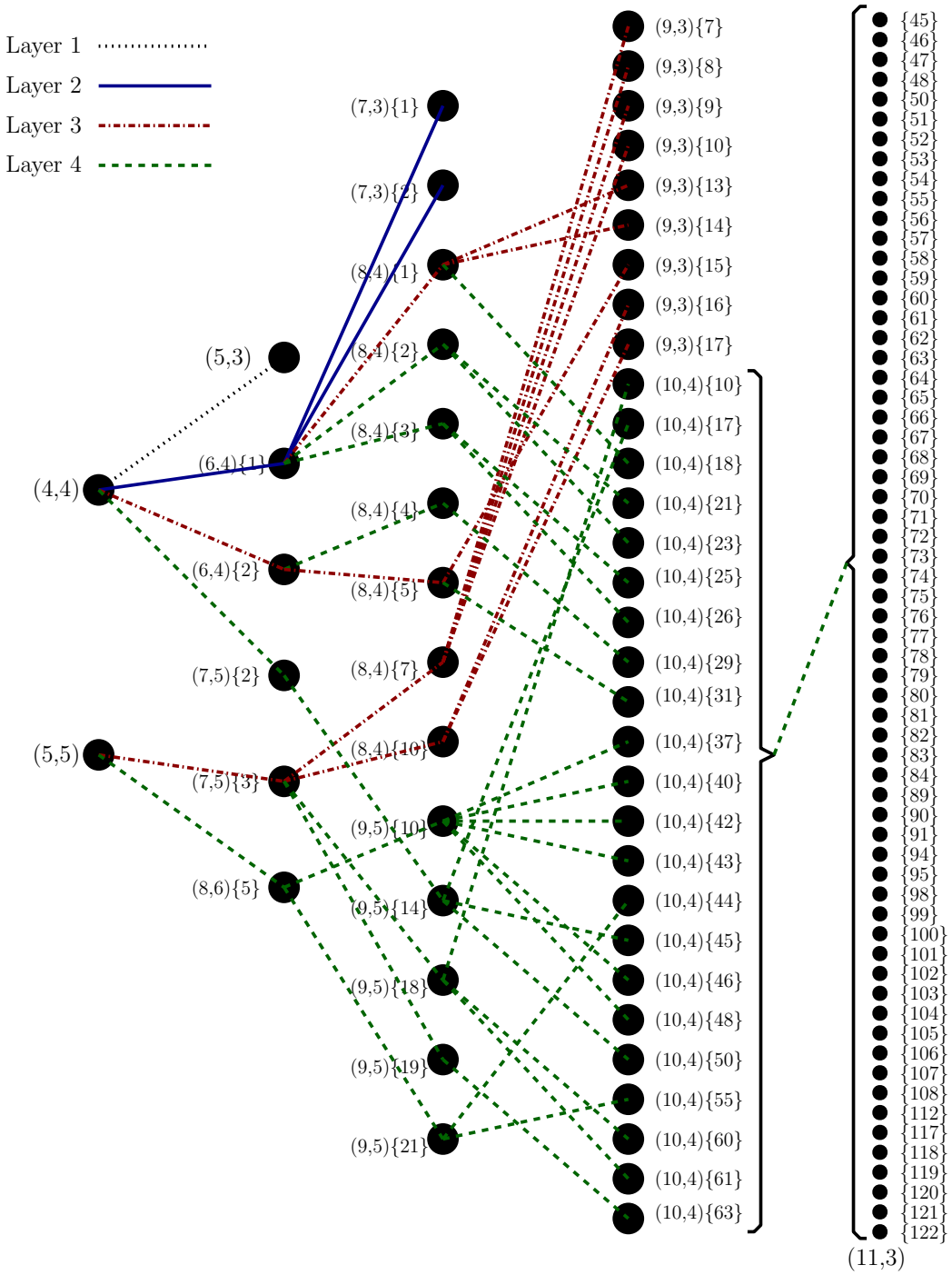


Figure 3.2: Search tree (trellis diagram) corresponding to LETS structures in the range $a \leq 12$ and $b \leq 3$ for QC-LDPC codes with $d_v = 3$ and $g = 8$.

Table 3.2: Characterization tables of LETSs in QC-LDPC codes with $d_v = 3$, $g = 8$, within the range $a \leq 12$ and $b \leq 3$: proposed search algorithm (top) and exhaustive search algorithm of [4] (bottom)

Proposed	$a=4$	$a=5$	$a=6$	$a=7$	$a=8$	$a=9$	$a=10$	$a=11$	$a=12$
$b=0$	–	–	–	–	–	–	–	–	–
$b=1$	–	–	–	–	–	–	–	–	–
$b=2$	–	–	–	–	–	–	–	–	–
$b=3$	–	$s_4(1)$ – – – –	–	$s_4(2)$ – – – –	–	$s_4(3), s_5(6)$ – – – –	–	$s_4(52)$ $s_5(10)$ – – – –	–
$b=4$	$s_4(1)$ – – – dot_2, pa_2 pa_3	–	$s_4(2)$ – – – dot_2, pa_2	–	$s_4(5)$,$s_5(2)$ – – – dot_2, pa_2	–	$s_4(9)$,$s_5(13)$ – – – dot_2	–	–
$b=5$	–	$s_5(1)$ – – – pa_2, pa_3	–	$s_4(1), s_5(1)$ – – – dot_2, pa_2	–	$s_4(1), s_5(4)$ – – – dot_2	–	–	–
$b=6$	–	–	–	–	$s_5(1)$ – – – dot_2	–	–	–	–
[4]	$a=4$	$a=5$	$a=6$	$a=7$	$a=8$	$a=9$	$a=10$	$a=11$	$a=12$
$b=0$	–	–	$s_4(1)$ – – – –	–	$s_4(2)$ – – – –	–	$s_4(5)$ – – – –	–	$s_4(20)$ $s_5(2)$ – – – –
$b=1$	–	–	–	–	–	–	–	–	–
$b=2$	–	–	$s_4(1)$ – – – pa_3 pa_4, lo_4^4	–	$s_4(5)$ – – – pa_3	–	$s_4(27)$ – – – pa_2	–	$s_4(178)$ $s_5(9)$ – – – –
$b=3$	–	$s_4(1)$ – – – dot_2, dot_3 pa_3, pa_4, lo_4^4	–	$s_4(3)$ – – – dot_2, dot_3 pa_2, pa_3	–	$s_4(16)$ – – – dot_2, dot_3 pa_2	–	$s_4(115)$ $s_5(7)$ – – – dot_2, dot_3	–
$b=4$	$s_4(1)$ – – – dot_2, pa_2 pa_3	–	$s_4(2)$ – – – dot_2, dot_3 pa_2, pa_3	–	$s_4(9)$,$s_5(1)$ – – – dot_2, dot_3 pa_2	–	$s_4(57)$,$s_5(6)$ – – – dot_2, dot_3	–	–
$b=5$	–	$s_5(1)$ – – – dot_2, dot_3 pa_2, pa_3	–	$s_4(2), s_5(1)$ – – – dot_2, dot_3 pa_2	–	$s_4(16), s_5(4)$ – – – dot_2, dot_3 pa_2	–	–	–
$b=6$	–	–	–	–	$s_5(2)$ – – – dot_2, dot_3	–	–	–	–

example, in the second layer of Fig. **??** *(solid lines), we only need to keep* $(6,4)\{1\}$ *LETSs since these trapping sets are needed in third and fourth layers. Moreover, to further reduce the complexity, we can perform the search process within each layer sequentially. For example, in the third layer of Fig.* **??** *(dash-dot lines), we can first apply the expansions corresponding to the* $(9,3)$ *structures originated from cycles of length* 8*, and in the case that these trapping sets were missing in the graph, we can continue to search for the rest of the structures in the* $(9,3)$ *class that are originated from cycles of length 10.*

3.3.4 Complexity of the Search Algorithm

In general, the complexity of the search algorithm depends on the multiplicity of different LETS structures involved in the search of the graph and the expansions that are applied to them. There is however no theoretical result for the multiplicity of different LETS structures in finite graphs. Moreover, as explained before, the graph itself changes throughout the construction process. In addition, in the layered implementation of the algorithm, the complexity of the search can highly vary depending on the layer at which an instance of a structure may be found by the algorithm. For these reasons, it is difficult to evaluate the complexity of the proposed search algorithm theoretically.

To compare the complexity of the proposed algorithm with that of the exhaustive search algorithm of [4], however, we count the number of different LETS structures involved in the search and the different expansions that are applied to them. For this, we use characterization tables such as those of Table **??**, and calculate the weighted sum of each expansion over different entries of the table with the weights being the multiplicity of the structures in the corresponding class. Since different expansions have different complexities [4], we count the weighted sum for each expansion, separately. These results for three construction scenarios are listed in Table **??**: $d_v = 3, g = 8, r : a \leq 12, b \leq 3$; $d_v = 4, g = 6, r : a \leq 8, b \leq 5$; and $d_v = 3, g = 6, r : a \leq 11, b \leq 2$. For each scenario, we have listed the results for four search algorithms in four columns. The first column corresponds to the exhaustive search algorithm of [4] where the QC structure of the graph is not taken into account. These results correspond to the characterization tables reported in [4]. In the second column, labeled as "QC," we have reported the results corresponding to the exhaustive search algorithm of [4], where the QC structure of the graph is taken into account.

The difference between this case and the first was explained in Section **??** (Table **??**). Finally, the third and fourth columns correspond to the proposed search algorithm for a general and a QC graph, respectively (where the advantages of layering are ignored). As an example of how the results in Table **??** are obtained, consider the first construction scenario and the expansion pa_3. For this case, the results reported in Table **??** for the proposed QC search and QC search are 2 and 14, respectively. By examining the top part of Table **??**, one can see that pa_3 appears in only two entries corresponding to classes $(4, 4)$ and $(5, 5)$, where each class has only one structure, hence the result $1 + 1 = 2$. In the bottom table, however, pa_3 appears in 7 entries corresponding to classes $(4, 4), (5, 3), (5, 5), (6, 2), (6, 4), (7, 3)$, and $(8, 2)$, where each class has the following number of structures: $1, 1, 1, 1, 2, 3$, and 5, respectively, which add up to 14.

The comparison of the results presented in Table **??** shows the considerable advantage of the proposed algorithm over the exhaustive search of [4], for both a general and a QC graph. One should also note that further advantage is gained through the layered implementation of the proposed search algorithm. Table **??** also demonstrates that the difference between the complexity of searching a general graph and that of searching a QC graph is smaller for the proposed algorithm compared to the exhaustive search algorithm of [4].

3.4 Construction of QC-LDPC Codes with Low Error Floor

In this part, we propose a construction method for protograph-based QC-LDPC codes with low error floor. The low error floor is achieved by avoiding the LETS structures within a predetermined list of structures $\mathcal{L}$. This is performed using the proposed search algorithm discussed in Section **??**.

In this work, we tackle two problems related to designing QC-LDPC codes: (a) For a given base graph of size $m \times n$, a given even integer g_0, and a given range $a \leq a_{max}$ and $b \leq b_{max}$, find a code with girth at least g_0 that has the minimum lifting degree (block length) and does not have any (a, b) LETS with $a \leq a_{max}$ and $b \leq b_{max}$; (b) For a given base graph, a given even integer g_0, a given positive integer b_{max}, and a fixed lifting degree N, find a code with girth at least g_0 and lifting degree N that

Table 3.3: Complexity comparison between four search algorithms of LETSs: exhaustive search algorithm of [4] for a general and a QC graph, and the proposed search algorithm for a general and a QC graph

	$d_v = 3, g = 8$ $a \leq 12, b \leq 3$				$d_v = 4, g = 6$ $a \leq 8, b \leq 5$				$d_v = 3, g = 6$ $a \leq 11, b \leq 2$			
	[4]	QC	Proposed	Proposed QC	[4]	QC	Proposed	Proposed QC	[4]	QC	Proposed	Proposed QC
dot_2	279	244	42	40	70	67	3	3	132	108	62	59
dot_3	278	243	0	0	124	103	30	29	108	73	46	44
dot_4	–	–	–	–	110	90	1	1	–	–	–	–
pa_2	85	83	14	13	5	4	1	1	40	39	11	11
pa_3	14	14	2	2	–	–	–	–	8	8	3	3
pa_4	2	2	–	–	–	–	–	–	–	–	–	–
lo_3^3	–	–	–	–	–	–	–	–	5	1	–	–
lo_4^3	–	–	–	–	–	–	–	–	2	1	1	1
lo_4^4	3	2	–	–	–	–	–	–	2	1	–	–

has no (a, b) LETS in the range $a \leq a_{max}$ and $b \leq b_{max}$, where a_{max} is maximized. We note that formulation of Problem (a) is similar to the formulation used in [24]. A similar formulation is also commonly used in the context of designing QC-LDPC codes of a certain girth with minimum length (see, e.g., [64]). We further note that in Problem (a), rather than a range for a and b values, one can use a list $\mathcal{L}$ of LETS structures to be avoided.

To design a QC-LDPC code based on a given base graph and a given lifting degree, we need to determine the nonzero elements of the exponent matrix of (**??**). The design constraints are to maintain the girth to at least g_0, and to ensure that no target LETS structure from the list $\mathcal{L}$ exists in the code. In this work, to design the code, we use a greedy search algorithm in which the nonzero elements of the exponent matrix P are selected column by column starting from the leftmost column. At each step, all the nonzero elements of one column of P are assigned in random simultaneously. The sub-matrix of the parity-check matrix H corresponding to the selected columns of P so far is then searched to see if the girth constraint is satisfied (no cycle of length less than g_0 exists in the corresponding graph) and that none of the structures in $\mathcal{L}$ exists in the sub-matrix. If the sub-matrix satisfies these constraints, the algorithm moves on to the next step and assigns the permutation shifts of the next column of P. If the

selected column fails to satisfy the constraints, a new random column is selected and tested. This process will continue until a column is found that satisfies the constraints or all the possible choices for that specific column are exhausted (although the choices are made randomly, repeated choices are avoided). In the latter case, the algorithm back tracks to the previously selected column and makes a new random choice for it. The algorithm will continue until all the columns of P are assigned and the corresponding H matrix satisfies all the constraints, or until all the possibilities are exhausted without reaching a solution. A third possibility would be to stop the algorithm if it fails to find a solution within a predetermined time period. Note that to search for the LETSs (of $\mathcal{L}$) in the graph corresponding to the sub-matrix of H at each step of the algorithm, we use the proposed layered search algorithm of Section **??**.

3.5 Numerical Results

In this section, we report some of the constructed QC-LDPC codes based on the technique described in Section **??**. All the simulation results are for binary-input AWGN channel, and a 5-bit min-sum decoder with clipping threshold equal to 2 [70], and maximum number of iterations 100. For each simulated point, at least 100 block errors are collected.

As the first experiment, we tackle Problem (a) in Section **??** for a fully-connected 3×5 base graph ($d_v = 3, d_c = 5$), with the constraint $g_0 = 8$, and with no LETSs in ranges r_1 to r_4, described in Example 1. The lifting degrees and the exponent matrices of the constructed codes are given in Table **??**. To obtain the results for $r_1 : a \leq 6, b \leq 3$, we start from the minimum lifting degree required for having a code with girth 8, which is $N = 13$ [64], and increase the lifting degree N by one at each step until we can find a code which is free of LETSs within the range of interest.

For r_1, the result presented in Table **??** is optimal in that $N = 18$ is the smallest lifting degree that can result in no LETSs within r_1, (i.e., all the possible exponent matrices for all the values of $13 \leq N \leq 17$ were checked and none was completely free of LETSs in r_1). To obtain the results for the other ranges, since $r_1 \subset r_2 \subset r_3 \subset r_4$, for each range $r_i, 2 \leq i \leq 4$, we start from the smallest N value obtained for r_{i-1}. For ranges r_2, r_3, r_4, however, the value of N presented in Table **??** is an upper bound on the smallest lifting degree satisfying the design constraints, i.e., as the value of N

Table 3.4: Upper bounds on the lifting degree of girth-8 QC-LDPC codes with fully-connected 3×5 base graph with no LETSs within different ranges, and the corresponding exponent matrices

Range	$(a \leq 6, b \leq 3)$	$(a \leq 8, b \leq 3)$	$(a \leq 10, b \leq 3)$	$(a \leq 12, b \leq 3)$
N	18	26	36	46
Exponent Matrix	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 7 & 8 \\ 0 & 2 & 11 & 5 & 14 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 6 & 16 \\ 0 & 3 & 21 & 12 & 23 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 14 & 29 & 34 \\ 0 & 2 & 24 & 32 & 12 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 12 & 28 & 33 \\ 0 & 2 & 22 & 35 & 39 \end{bmatrix}$

Table 3.5: Upper bounds on the lifting degree of girth-8 QC-LDPC codes with fully-connected 3×6 base graph with no LETSs within different ranges, and the corresponding exponent matrices

Range	$(a \leq 6, b \leq 3)$	$(a \leq 8, b \leq 3)$	$(a \leq 10, b \leq 3)$	$(a \leq 12, b \leq 3)$
N	32	41	60	80
Exponent Matrix	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 11 & 17 & 24 & 29 \\ 0 & 14 & 30 & 5 & 3 & 6 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16 & 18 & 19 & 22 & 33 \\ 0 & 32 & 21 & 26 & 39 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 23 & 33 & 38 & 40 & 59 \\ 0 & 22 & 45 & 54 & 48 & 42 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 39 & 41 & 45 & 61 \\ 0 & 35 & 43 & 51 & 66 & 36 \end{bmatrix}$

was increased at each step by one, we limited the search time for each N to 3 hours.[3] If no exponent matrix satisfying all the constraints was found during this time, we would increase N by one and restart the search.

Similar approach was used for the 3×6 base graph to design rate-1/2 QC-LDPC codes with girth 8 and free of LETSs within ranges r_1 to r_4. The results are presented in Table **??**.

In [24], Problem (a) was tackled for fully-connected base graphs of size 3×5 and 3×6, $g = 8$ and r_2, i.e., $a \leq 8, b \leq 3$. Based on the design approach of [24], it was shown that the minimum lifting degrees required to satisfy the trapping set constraint for the two base graphs are $N = 41$ and $N = 61$, respectively. Using a search algorithm, the authors of [24] were able to find QC-LDPC codes with girth 8 and lifting degrees 41 and 63 for the two base graphs, respectively, that were free of LETSs within r_2, In comparison, the codes designed here have $N = 26$ and $N = 41$ for 3×5 and 3×6 base graphs, respectively. These codes are significantly superior to those found in [24], in the sense that they have the same girth and degree distribution and satisfy the same trapping set constraint but have a much smaller block length.

[3] Our search algorithm is implemented in MATLAB and is run on a PC with 3.50 GHZ CPU and 32 GB RAM.

Table 3.6: Multiplicities of LETS structures in the range $a \leq 12$ and $b \leq 4$ for Code $\mathcal{C}_1$ and the code designed in [24]

(a, b) class	$\mathcal{C}_1$	Code of [24]	(a, b) class	$\mathcal{C}_1$	Code of [24]
$(4, 4)$	451	451	$(10, 4)$	8651	12956
$(6, 4)$	533	820	$(11, 3)$	328	1230
$(8, 4)$	1599	3485	$(12, 2)$	0	123
$(9, 3)$	0	246	$(12, 4)$	42599	57195

The main reason for the improved results in this work compared to [24] is that, contrary to the approach adopted here which in fact imposes a necessary and sufficient condition for removing all the LETS structures within the targeted range, in [24], the authors targeted the $(5, 3)$ and $(6, 4)\{1\}$ structures. Note that $(6, 4)\{1\}$ structure is a parent to some of the targeted structures in the range but is not of interest itself. This imposes a sufficient but not necessary condition for removing the targeted LETSs. This unnecessary constraint imposed on the design degrades the quality of the achievable solution.

As another example, we design a QC-LDPC code $\mathcal{C}_1$ lifted from the 3×5 fully-connected base graph with lifting degree $N = 41$, with $g = 8$, and free of LETSs within the union of two rectangular regions $a \leq 10, b \leq 3$, and $a \leq 12, b \leq 2$. The exponent matrix of $\mathcal{C}_1$ is

$$P_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 5 & 7 & 26 \\ 0 & 3 & 13 & 30 & 37 \end{bmatrix}. \tag{3.1}$$

This code has the same degree distribution and block length as the code designed in [24], but is free of LETS structures within a much larger region. (See Table **??** for the comparison of LETS distributions. In the table, we have only listed the classes for which at least one code has non-zero multiplicity).

As another experiment, we construct a QC-LDPC code which has similar parameters to the well-known $(155, 64)$ Tanner code [71], but has a lower error floor. Tanner code is a cyclic lifting of the fully-connected 3×5 base graph with $N = 31$ and has $g = 8$. A similar code was also designed in [23] with the goal of reducing the error

floor by removing all $(5,3)$ LETSs and minimizing the number of $(6,4)$ LETSs in the Tanner graph of the code. In this case, we use the formulation of Problem (b) in Section V for the fully-connected 3×5 base graph with $N = 31$ and $g_0 = 8$. We consider two cases of $b_{\max} = 3$ and $b_{\max} = 2$, and are able to construct two codes that are free of LETS structures up to size $a_{\max} = 8$ and $a_{\max} = 10$, respectively. Then, we target the union of the LETSs within the two regions $a \leq 8, b \leq 3$, and $a \leq 10, b \leq 2$, and are able to design a code $\mathcal{C}_2$ with $N = 31$ and $g = 8$ that has no LETS within the targeted region. The exponent matrix of this code is as follows

$$P_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 5 & 21 & 30 \\ 0 & 3 & 13 & 6 & 20 \end{bmatrix} . \tag{3.2}$$

We have presented the multiplicities of LETSs of the designed code and those of the Tanner code as well as the code designed in [23] in Table **??**. It is well-known that the most dominant trapping sets of the Tanner code are $(8,2)$ and $(10,2)$ LETSs, respectively. Both structures are completely removed from $\mathcal{C}_2$. On the other hand, for the code of [23], although the $(8,2)$ LETSs are removed, there are still a number of $(10,2)$ LETSs present. This code also has 31 instances of the $(7,3)$ LETS, another potentially problematic structure. We have compared the frame error rate (FER) of the constructed code $\mathcal{C}_2$ with those of Tanner code and the code of [23] in Fig. **??**. The superior performance of $\mathcal{C}_2$ over both Tanner code and the code of [23] in the error floor region can be observed. Based on the LETS multiplicities presented in Table **??**, we expect the performance gap between $\mathcal{C}_2$ and the code of [23] to increase by further increase in the signal-to-noise ratio (SNR). In fact, based on the simulation results, at SNR = 6 dB, 44 out of 100 errors of the code designed in [23] are $(10,2)$ LETSs, which are completely absent in $\mathcal{C}_2$.

As another example, we construct girth-6 codes following the formulation of Problem (a), and by using the fully-connected 3×5 base graph with the constraint that the code is free of LETSs within 4 different ranges as shown in Table **??**. The smallest lifting degree and the exponent matrix for each case are also given in Table **??**. For the first range, $a \leq 5$ and $b \leq 2$, our search result is exhaustive and $N = 10$ is in

Table 3.7: Multiplicities of LETS structures in the range $a \leq 12$ and $b \leq 3$ for $\mathcal{C}_2$, $(155, 64)$ Tanner code, and the code of [23]

(a, b) class	Tanner Code	Code of [23]	$\mathcal{C}_2$	(a, b) class	Tanner Code	Code of [23]	$\mathcal{C}_2$
$(4, 4)$	465	527	558	$(9, 3)$	1860	558	465
$(5, 3)$	155	0	0	$(10, 2)$	1395	93	0
$(7, 3)$	930	31	0	$(11, 3)$	6200	4960	4154
$(8, 2)$	465	0	0	$(12, 2)$	930	992	682

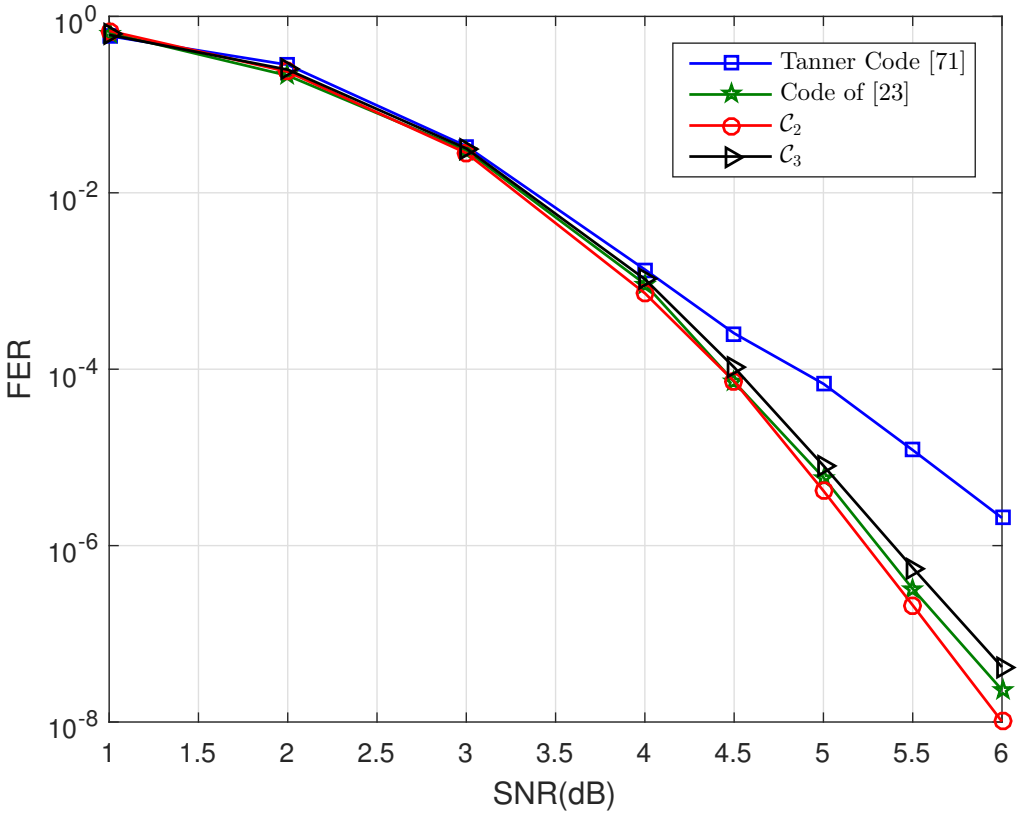


Figure 3.3: FER comparison among constructed codes $\mathcal{C}_2$, $\mathcal{C}_3$, $(155, 64)$ Tanner code, and the code of [23] (All codes are QC, and have $d_v = 3$, $d_c = 5$. The code $\mathcal{C}_2$, $(155, 64)$ Tanner code, and the code of [23] have $g = 8$ and block length 155, while $\mathcal{C}_3$ has $g = 6$ and block length 145).

Table 3.8: Upper bounds on the lifting degree of girth-6 QC-LDPC codes with fully-connected 3×5 base graph with no LETSs within different ranges, and the corresponding exponent matrices

Range	$(a \leq 5, b \leq 2)$	$(a \leq 7, b \leq 2)$	$(a \leq 9, b \leq 2)$	$(a \leq 11, b \leq 2)$
N	10	15	22	29
Exponent Matrix	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 5 & 6 & 8 \\ 0 & 2 & 8 & 4 & 9 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 7 & 10 & 14 \\ 0 & 12 & 11 & 2 & 13 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 12 & 13 & 18 \\ 0 & 21 & 19 & 14 & 20 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 9 & 15 & 16 \\ 0 & 8 & 16 & 1 & 18 \end{bmatrix}$

fact the smallest possible lifting degree that can satisfy the LETS constraint. For the other ranges, the search is not exhaustive and the given value of N provides an upper bound on the smallest lifting degree. We have included the FER of the code $\mathcal{C}_3$ designed to be free of LETSs in the range $a \leq 11, b \leq 2$, in Fig. **??**. As can be seen, this code handily outperforms the Tanner code in the error floor region. This is impressive, considering that both the girth and the block length of $\mathcal{C}_3$ are smaller than those of the Tanner code (6 vs. 8, and 145 vs. 155, respectively).

To further demonstrate the strength of the designed codes, we consider the code whose exponent matrix is given in the last column of Table **??**, and compare it with similar codes (3×6 fully-connected base graph, $N = 80$ and $g = 8$) constructed using the well-known QC-PEG [17] and Improved QC-PEG [23] methods. The multiplicities of LETSs for the three codes are presented in Table **??**. As can be seen from the table, many of LETSs that dominate the error floor performance of the other two codes are absent from the Tanner graph of the designed code. To investigate this further, in Fig. **??**, we have provided the FER of the three codes. Fig. **??** shows the superior error floor performance of the designed code. Based on the simulation results, the dominant trapping set structures of QC-PEG code are $(8, 2)$ and $(10, 2)$. In the code of [23], the $(8, 2)$ structure has been removed and thus the error floor performance has improved compared to the QC-PEG code. The dominant trapping sets of the code of [23] are in $(10, 2)$ and $(12, 2)$ classes, followed by $(7, 3)$ and $(9, 3)$ classes. All of these classes however, are absent from our designed code, thus the superior error floor performance.

All the examples given so far, similar to the majority of the results available in the literature, were for codes with $d_v = 3$. To demonstrate the generality of our method,

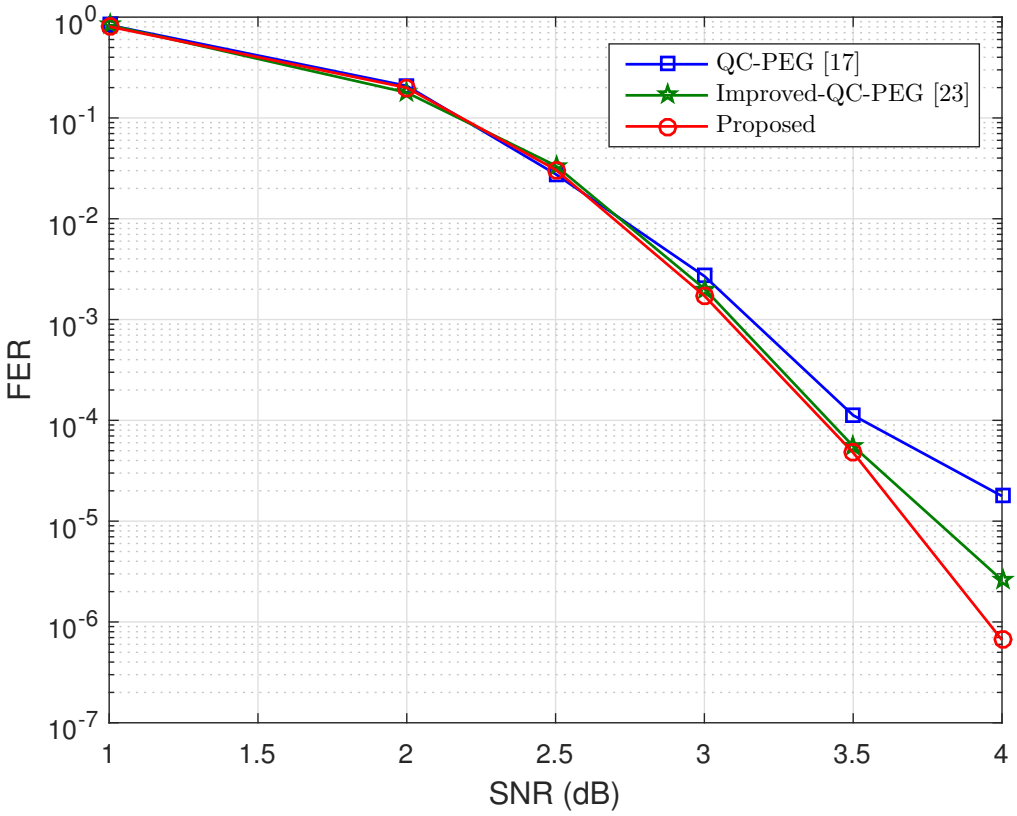


Figure 3.4: FER performance of the constructed $(480, 240)$ QC-LDPC code (last column of Table **??**) in comparison with the FER of similar codes constructed by QC-PEG [17] and Improved QC-PEG [23].

Table 3.9: Multiplicities of LETS structures in the range $a \leq 12$ and $b \leq 3$, for the designed code (last column of Table **??**) and similar codes designed by QC-PEG methods

(a, b) class	QC-PEG [17]	Improved QC-PEG [23]	Designed Code
$(7, 3)$	160	160	0
$(8, 2)$	80	0	0
$(9, 3)$	160	480	0
$(10, 2)$	160	80	0
$(11, 3)$	1280	1120	0
$(12, 2)$	240	160	0

we also construct girth-6 codes from the fully-connected 4×6, 4×8 and 4×16 base graphs following the formulation of Problem (a) in Section V. For each base graph, four codes are designed, where the LETS structures within 4 different ranges are avoided. The results are presented in Tables **??**, **??**, and **??**, respectively. All the values of N in these tables are upper bounds on the smallest lifting degree that can satisfy the corresponding LETS constraint, with the exception of the result of $N = 7$ in Table **??**, which is in fact the smallest lifting degree that can result in a code free of LETSs within the range $a \leq 5, b \leq 5$. We note that Diouf [72] has constructed a protograph-based QC-LDPC code with $g = 6$ and free of the $(4, 4)$ LETS structure using the 4×6 fully-connected base graph with $N = 7$. An examination of the code of [72] reveals that its LETS distribution within the range $a \leq 8, b \leq 5$ is the same as that of the code designed here.

As another example, we consider the regular $(576, 432)$ QC-LDPC code with $d_v = 4$ designed in [34]. This code is designed based on array dispersion method, where first a 4×36 exponent matrix with lifting degree $N = 36$ is constructed, and then, 16 columns of this matrix are selected such that the resulting code contains fewer short cycles and larger girth. For comparison, we design a cyclic lifting $\mathcal{C}_4$ of the 4×16 base graph with $N = 36$ and $g = 6$ and free of LETSs within the union of the ranges $a \leq 5, b \leq 5$ and $a \leq 8, b \leq 3$:

$$P_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 4 & 5 & 8 & 10 & 11 & 12 & 16 & 18 & 20 & 22 & 23 & 28 & 29 & 33 \\ 0 & 34 & 22 & 6 & 25 & 20 & 30 & 23 & 32 & 5 & 35 & 28 & 21 & 31 & 7 & 1 \\ 0 & 33 & 17 & 2 & 26 & 8 & 20 & 4 & 10 & 35 & 19 & 32 & 31 & 3 & 14 & 29 \end{bmatrix}. \tag{3.3}$$

In Table **??**, we have listed the LETSs of $\mathcal{C}_4$ and the code of [34] in the range $a \leq 8, b \leq 6$. The FER curves for the two codes are also given in Fig. **??**. These results clearly show the superior LETS distribution and error floor performance of $\mathcal{C}_4$.

As the final example, we consider a $(2133, 1817)$ array-based code with lifting degree $N = 79$, $d_v = 4$, and $d_c = 27$. To construct this code, similar to the previous

Table 3.10: Upper bounds on the lifting degree of girth-6 QC-LDPC codes with fully-connected 4×6 base graph with no LETSs within different ranges, and the corresponding exponent matrices

Range	$(a \leq 5, b \leq 5)$	$(a \leq 6, b \leq 5)$	$(a \leq 7, b \leq 5)$	$(a \leq 8, b \leq 5)$
N	7	13	15	17
Exponent Matrix	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 2 & 4 & 6 & 1 & 3 \\ 0 & 4 & 1 & 5 & 2 & 6 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 9 & 10 & 11 & 12 \\ 0 & 3 & 10 & 8 & 4 & 2 \\ 0 & 2 & 4 & 6 & 1 & 11 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 7 & 8 & 13 \\ 0 & 2 & 6 & 3 & 12 & 7 \\ 0 & 3 & 10 & 6 & 5 & 4 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 13 & 14 & 15 & 16 \\ 0 & 4 & 15 & 10 & 5 & 7 \\ 0 & 3 & 6 & 4 & 2 & 5 \end{bmatrix}$

Table 3.11: Upper bounds on the lifting degree of girth-6 QC-LDPC codes with fully-connected 4×8 base graph with no LETSs within different ranges, and the corresponding exponent matrices

Range	$(a \leq 5, b \leq 5)$	$(a \leq 6, b \leq 5)$
N	15	18
Exponent Matrix	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 5 & 9 & 11 & 12 & 13 & 14 \\ 0 & 7 & 6 & 14 & 5 & 10 & 12 & 1 \\ 0 & 14 & 3 & 8 & 13 & 9 & 1 & 6 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 10 & 13 & 14 & 15 & 16 & 17 \\ 0 & 5 & 2 & 4 & 15 & 1 & 11 & 16 \\ 0 & 12 & 7 & 1 & 9 & 17 & 2 & 4 \end{bmatrix}$
Range	$(a \leq 7, b \leq 5)$	$(a \leq 8, b \leq 5)$
N	21	24
Exponent Matrix	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 15 & 16 & 17 & 18 & 19 & 20 \\ 0 & 17 & 9 & 5 & 4 & 1 & 20 & 19 \\ 0 & 6 & 5 & 4 & 15 & 3 & 14 & 12 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 11 & 15 & 17 & 19 & 20 & 21 & 23 \\ 0 & 22 & 5 & 21 & 13 & 2 & 14 & 20 \\ 0 & 10 & 9 & 18 & 7 & 16 & 6 & 13 \end{bmatrix}$

Table 3.12: Upper bounds on the lifting degree of girth-6 QC-LDPC codes with fully-connected 4×16 base graph with no LETSs within different ranges, and the corresponding exponent matrices

Range	Exponent Matrices
$N = 32$, $(a \leq 5, b \leq 5)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 4 & 6 & 10 & 12 & 14 & 16 & 18 & 20 & 23 & 24 & 25 & 28 & 31 \\ 0 & 17 & 16 & 25 & 20 & 3 & 29 & 22 & 11 & 6 & 27 & 2 & 8 & 23 & 15 & 5 \\ 0 & 12 & 21 & 2 & 8 & 25 & 18 & 7 & 10 & 13 & 31 & 30 & 9 & 16 & 27 & 3 \end{bmatrix}$
$N = 44$, $(a \leq 6, b \leq 5)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 5 & 8 & 9 & 11 & 12 & 16 & 17 & 20 & 23 & 24 & 26 & 33 & 39 \\ 0 & 15 & 31 & 13 & 21 & 2 & 35 & 32 & 3 & 34 & 42 & 9 & 30 & 11 & 29 & 7 \\ 0 & 20 & 9 & 30 & 16 & 37 & 29 & 11 & 18 & 22 & 43 & 21 & 7 & 17 & 13 & 4 \end{bmatrix}$
$N = 54$, $(a \leq 7, b \leq 5)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 5 & 7 & 10 & 12 & 13 & 16 & 18 & 20 & 21 & 22 & 23 & 29 & 50 \\ 0 & 47 & 7 & 28 & 39 & 20 & 53 & 27 & 6 & 25 & 8 & 24 & 11 & 35 & 22 & 34 \\ 0 & 5 & 28 & 45 & 9 & 29 & 13 & 47 & 33 & 30 & 24 & 53 & 3 & 17 & 52 & 11 \end{bmatrix}$
$N = 60$, $(a \leq 8, b \leq 5)$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 8 & 10 & 12 & 14 & 15 & 17 & 20 & 21 & 26 & 28 & 37 & 40 \\ 0 & 5 & 49 & 25 & 40 & 15 & 27 & 35 & 29 & 24 & 9 & 30 & 42 & 4 & 18 & 39 \\ 0 & 3 & 40 & 47 & 36 & 23 & 19 & 45 & 6 & 21 & 8 & 55 & 49 & 42 & 1 & 32 \end{bmatrix}$

Table 3.13: Multiplicities of LETS structures in the range $a \leq 8$ and $b \leq 6$ for the constructed code $\mathcal{C}_4$ and the code of [34]

(a, b) class	$\mathcal{C}_4$	Code of [34]	(a, b) class	$\mathcal{C}_4$	Code of [34]
$(3, 6)$	14580	12456	$(7, 4)$	2340	20160
$(4, 4)$	0	144	$(7, 6)$	590724	1122480
$(4, 6)$	27000	27648	$(8, 0)$	0	171
$(5, 4)$	0	324	$(8, 2)$	0	2520
$(5, 6)$	59508	88416	$(8, 4)$	14634	96732
$(6, 4)$	756	7236	$(8, 6)$	2345328	3616272
$(6, 6)$	189360	359136			

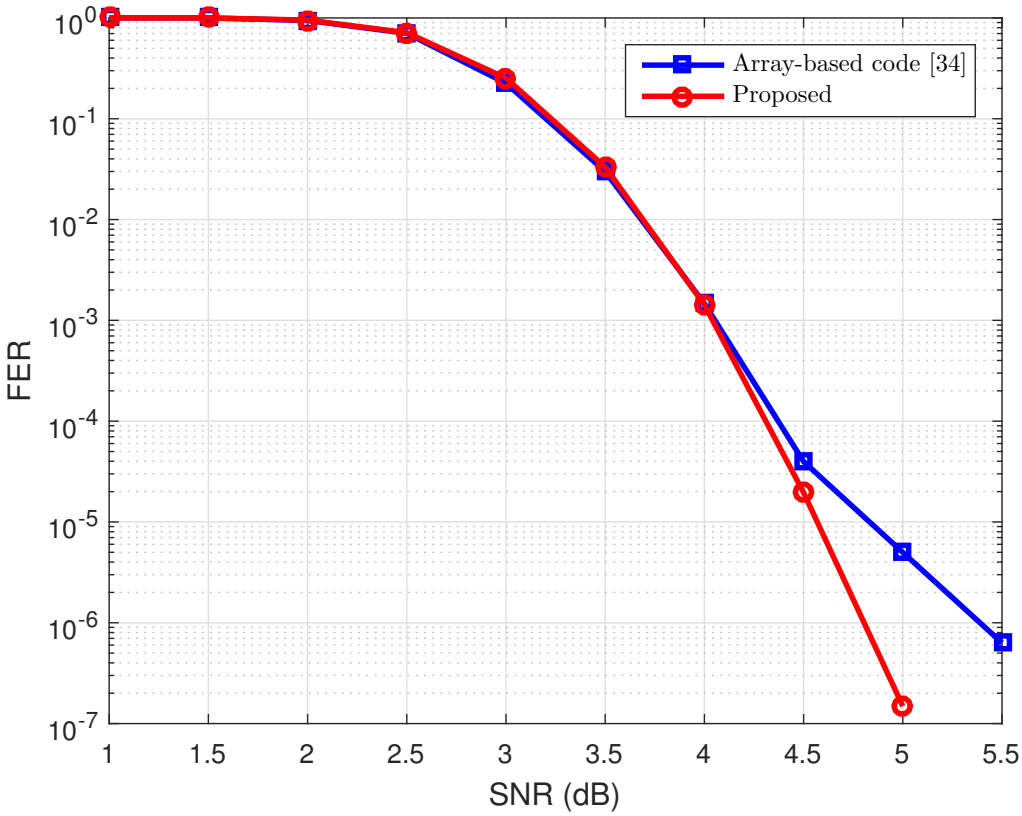


Figure 3.5: FER curves of the constructed $(576, 432)$ code $\mathcal{C}_4$ and the similar code of [34].

example, first a 4×79 exponent matrix is generated, and then 27 columns of this matrix are chosen such that the resulting code has fewer short cycles and larger girth [34]. For comparison, we then use our technique to construct a code $\mathcal{C}_5$ with similar parameters ($N = 79$, $d_v = 4$, $d_c = 27$) which is free of LETSs within the union of the ranges $a \leq 7, b \leq 5$, and $a \leq 8, b \leq 3$. The exponent matrix of $\mathcal{C}_5$ is given by:

$$P_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 5 & 6 & 9 & 11 & 12 & 13 & 17 & 19 & 21 & 24 \\ 0 & 24 & 68 & 66 & 36 & 59 & 37 & 45 & 29 & 58 & 64 & 75 & 34 \\ 0 & 4 & 12 & 76 & 43 & 53 & 8 & 54 & 34 & 66 & 22 & 77 & 72 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 28 & 30 & 34 & 36 & 37 & 38 & 46 & 49 & 51 & 52 & 55 & 60 & 64 & 69 \\ 2 & 57 & 70 & 55 & 35 & 40 & 27 & 1 & 11 & 67 & 72 & 65 & 23 & 32 \\ 55 & 36 & 35 & 15 & 25 & 13 & 41 & 62 & 68 & 56 & 78 & 10 & 38 & 9 \end{bmatrix}. \tag{3.4}$$

The exhaustive search of LETSs within the range of $a \leq 8$ and $b \leq 5$ for $\mathcal{C}_5$ reveals that this code has only one class, i.e., $(8, 4)$, within this range with non-zero multiplicity. The multiplicity of $(8, 4)$ LETSs in $\mathcal{C}_5$ is 5925. On the other hand, the code of [34] has three classes with non-zero multiplicity in the range $a \leq 8$, $b \leq 5$. These classes are $(7, 4), (8, 2)$ and $(8, 4)$, with multiplicities $9006, 3634$, and 122450, respectively. As can be seen, the designed code has a superior LETS distribution, and thus a lower error floor, compared to the code of [34]. In fact, our simulations show that the most dominant class of trapping sets in the code of [34] is the $(8, 2)$ class, which is completely absent in the designed code.

Finally, in order to demonstrate the complexity reduction of the proposed search technique in comparison with the *dpl* search of [4], in Table **??**, we have listed the run-time of both algorithms for finding the solutions in the largest ranges of Tables **??**, **??**, **??**, **??**, and **??**. As can be seen, in all cases, the proposed algorithm is much faster than that of [4] by up to more than one order of magnitude.

Table 3.14: Comparison of run-times for the proposed method and the method of [4]

Range of a and b values	$(a \leq 12, b \leq 3)$	$(a \leq 12, b \leq 3)$	$(a \leq 8, b \leq 5)$	$(a \leq 8, b \leq 5)$	$(a \leq 8, b \leq 5)$
Girth g	8	8	6	6	6
Variable node degree d_v	3	3	4	4	4
Check node degree d_c	5	6	6	8	16
Lifting degree N	46	80	17	24	60
Run-time of the proposed method (sec.)	2889	3425	755	2642	487
Run-time of the method of [4] (sec.)	36293	42955	5518	14718	3591

Chapter 4

Construction of Irregular QC-LDPC Codes with Low Error Floor

4.1 Introduction

In chapter **??**, we studied a related but fundamentally different problem of finding out whether any instance from a collection of LETS structures exists in a given variable-regular LDPC code or not. We then devised an efficient *dpl*-based algorithm that can solve this problem, and used it to design QC-LDPC codes from fully-connected base graphs. In this chapter, we extend the algorithm of [62] to irregular binary LDPC codes and use it to design irregular protograph-based QC-LDPC codes. The extension from regular to irregular codes is a challenging task since the variety of LETS structures of irregular codes within a given class is often significantly larger than that of regular codes. This makes the study of parent/child relationships between the LETS structures of interest through the three *dpl* expansions complex. Establishing such relationships is an essential step in achieving the ultimate goal of devising a search algorithm with minimal complexity that can solve the problem of interest. To minimize the complexity of the search, one needs to find the minimum number of structures within the range of interest that need to be targeted. After that, one needs to devise a low complexity *dpl*-based search algorithm, starting from simple cycles and by using out-of-range structures, to systematically search for the targeted in-range structures.

As we mentioned in chapter **??**, ETSs are divided into LETSs and ETSLs. ETSLs can be partitioned into two categories: those with cycles, denoted by ETSL_1, and those without any cycle, denoted by ETSL_2 [5]. Based on our experiments with

irregular QC-LDPC codes, for such codes, there are no ETSL_2's in the error floor region. In this work, we are thus only concerned about LETSs and ETSL_1's. The following proposition shows that removing LETSs within a certain range guarantees that all the ETSL_1's are also removed in that range.

Proposition 3. *Consider an irregular Tanner graph G with $\delta(G) \geq 2$, and free of LETSs within the range of $a \leq a_{max}, b \leq b_{max}$. Then, graph G is also free of $ETSL_1$'s within the range $a \leq a_{max} + 1, b \leq b_{max}$.*

Proof. Let S be an (a, b) ETSL_1 structure. Call the largest LETS subset of S, S', and suppose that S' is in the (a', b') class. We then have $a' < a$ and $b' \leq b$. Now, in contrast to the claim of the proposition, assume that G has an (a, b) ETSL_1, S, within the range $a \leq a_{max} + 1$ and $b \leq b_{max}$. This implies that the largest LETS subset of S is within the range $a \leq a_{max}, b \leq b_{max}$, which is a contradiction. ■

Based on Proposition **??**, throughout this chapter, we only focus on the elimination of LETSs from the constructed codes.

The remainder of this chapter is organized as follows. In Section **??**, we propose the low-complexity layered search algorithm for finding LETSs of an irregular QC-LDPC code. Section **??** is devoted to the comparison between the proposed search algorithm and that of [5]. In Section **??**, a method for construction of irregular QC-LDPC codes with good waterfall performance and low error floor is described. We present some constructed codes and simulation results comparing them with similar existing codes in Section **??**.

4.2 Efficient Search Algorithm for Finding LETSs of an Irregular LDPC Code

In this chapter, our goal is to design irregular protograph-based QC-LDPC codes free of certain collection of LETS structures denoted by $\mathcal{L}$. We achieve this goal in two steps. In the first step, we design a base graph with m check nodes and n variable nodes to obtain a code rate of $R \geq 1 - m/n$. The degree distribution of the base graph is selected based on a constraint on the maximum variable node degree $d_{v_{\max}}$, and to obtain the best waterfall performance. In the second step, given the base graph, we design the exponent matrix for the best error floor performance. To obtain

the non-infinity elements of the exponent matrix, we assign them column by column. Each time that a new column is added, we use the search algorithm devised in this section to find the answer to the following decision problem: Does there exist any instance of LETS structures in $\mathcal{L}$ in the Tanner graph constructed so far? If the answer is negative, we move on to assign the next column of the exponent matrix. Otherwise, a new candidate for the present column is selected. In the construction process, we may need to run the search algorithm hundreds or thousands of times. For the construction process to be feasible in practice, it is thus essential to have a low complexity search algorithm. A naive approach to perform the search would be to use the exhaustive search algorithm of [5], and see if the algorithm can find any instance of LETSs in $\mathcal{L}$. This however appears to be prohibitively complex for our application. In this section, we thus develop an efficient algorithm to solve the decision problem, described above, for irregular Tanner graphs. Compared to the case of regular Tanner graphs considered in [62], this is considerably more challenging due to the much larger number of non-isomorphic structures within each class that need to be identified and enumerated, and the larger variety of parent/child relationships among different structures that need to be identified and then selectively chosen to minimize the search complexity.

4.2.1 Finding all non-isomorphic LETS structures within a given class of an irregular Tanner graph

To achieve a low error floor, the design problem is often formulated such that TSs within a collection of dominant classes are avoided. Such classes are often identified by the range of their a and b values, i.e., $a \leq a_{\max}$ and $b \leq b_{\max}$. As the first step of the design, we thus need to obtain the collection $\mathcal{L}$ of all the LETS structures within different classes in the range of interest. To demonstrate the complexity of this task for irregular codes in comparison with that of regular codes, it is helpful to consider a variable-regular code with $d_v = d_{v_{\max}}$, where $d_{v_{\max}}$ is the maximum variable node degree of the irregular code. The next proposition and the example that follows demonstrate that corresponding to each (a, b) LETS structure of the variable-regular code, there exist a large number of (a, b') LETS structures with $b' \leq b$ in the irregular code.

Proposition 4. *Consider the normal graph $\hat{G}(S)$ of an (a, b) LETS structure S*

in a variable-regular Tanner graph G with variable node degree d_v. Also, consider all the non-isomorphic (a, b') LETS structures S' in an irregular Tanner graph G' with $d_{v_{\max}} = \Delta(G') = d_v$, whose normal graph is also $\hat{G}(S)$. If G' can have all the variable node degrees in the range $[2, d_v]$, then the multiplicity of such TSs (S') is $\prod_{i=1}^{a}(d_v - d_{v_i} + 1)$, where d_{v_i} is the degree of the i-th node in $\hat{G}(S)$, and for all such TSs, we have $b' \leq b$.

Proof. For the LETS of the variable-regular graph, we have $b = \sum_{i=1}^{a}(d_v - d_{v_i})$. On the other hand, for every quasi-normal hypergraph $\check{G}'(S')$ in G', whose projection is $\hat{G}(S)$, $b' = \sum_{i=1}^{a}(d'_{v_i} - d_{v_i})$, where d'_{v_i} is the degree of the i-th variable node in S'. Since $d'_{v_i} \leq d_v$, it can be concluded that $b' \leq b$. Moreover, to obtain a quasi-normal hypergraph, as an inverse image of $\hat{G}(S)$, one can add $0, \ldots, d_v - d_{v_i}$ edges to the i-th node of $\hat{G}(S)$. Since these choices for different nodes of $\hat{G}(S)$ can be freely combined to obtain different (non-isomorphic) quasi-normal hypergraphs, the total number of quasi-normal hypergraphs is $\prod_{i=1}^{a}(d_v - d_{v_i} + 1)$. ■

Example 3. *Consider the $(5, 4)$ LETS structure in a variable-regular LDPC code with $d_v = 4$ whose normal graph is shown in Fig. **??**(a). Corresponding to this normal graph, there exist 12 non-isomorphic $(5, b')$ LETS structures with $b' \leq 4$, in an irregular code with $d_{v_{\max}} = 4$, whose variable node degrees can be 2, 3 or 4. The quasi-normal hypergraphs of these structures are shown in Fig. **??**(b).*

Suppose that our goal is to find all non-isomorphic (a, b') LETS structures of an irregular graph with maximum variable node degree $d_{v_{\max}}$ in the range of $a \leq a_{\max}$ and $b' \leq b_{\max}$. To perform this task, for each (a, b) class with $a \leq a_{\max}$ and $b \leq b_{\max}$, we first find the normal graphs of all the (non-isomorphic) LETS structures within that class for variable-regular Tanner graphs with $d_v = d_{v_{\max}}$. For each value of $a \leq a_{\max}$, we then go through all the classes with different values of b, and based on Proposition **??**, generate all the quasi-normal hypergraphs that are the inverse images of the normal graphs in those classes. As these quasi-normal hypergraphs are generated, we correspondingly assign them to proper (a, b') classes, depending on their b' values. At the end of this process, we have all the non-isomorphic LETS structures of the irregular graphs within all the (a, b') classes with $a \leq a_{\max}$ and $b' \leq b_{\max}$.

We note that avoiding odd-degree variable nodes can significantly reduce the variety of LETS structures, and thus the complexity of the search algorithm. In addition, it can improve the error floor as a result of smaller number of LETS structures within

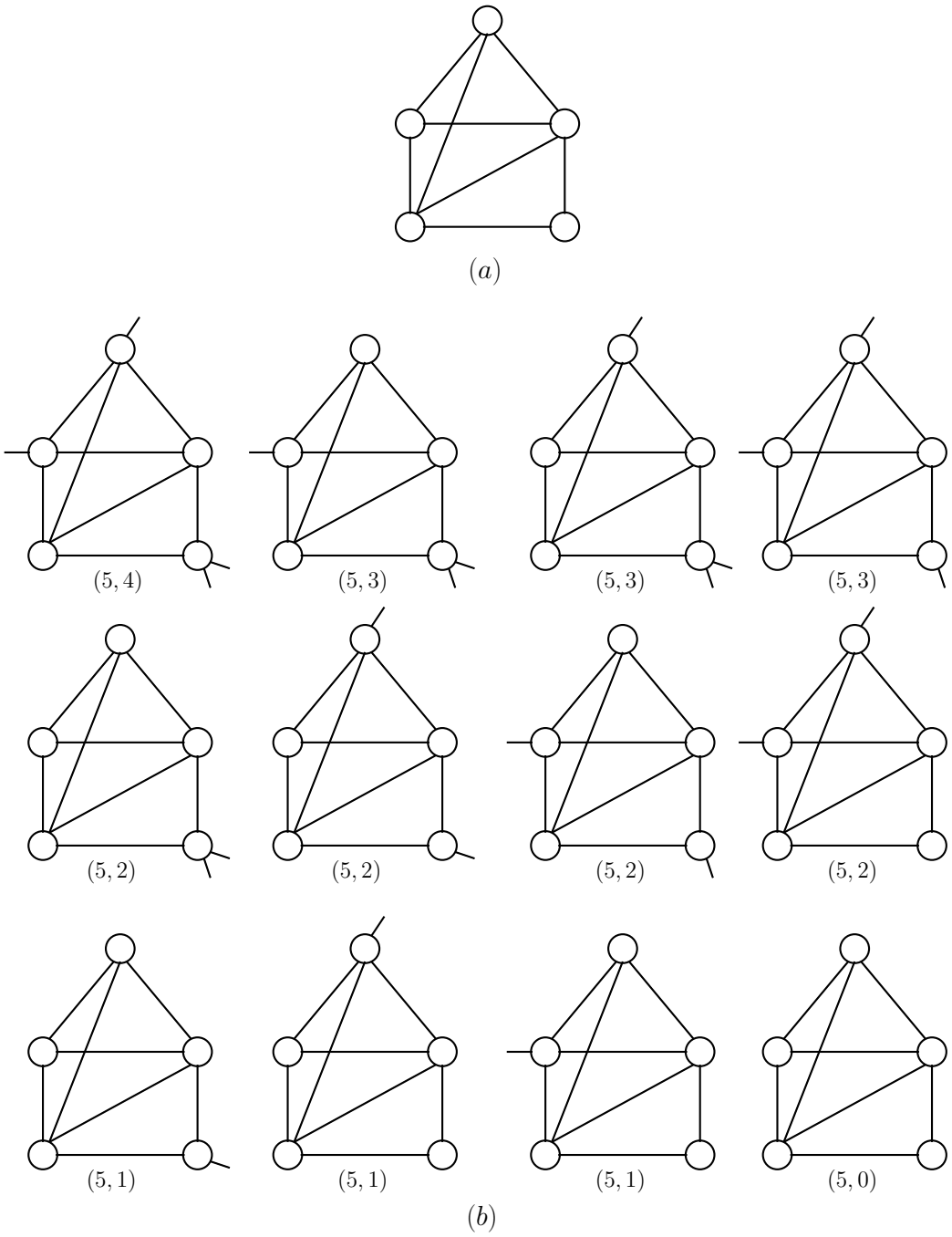


Figure 4.1: (a) Normal graph of a $(5, 4)$ LETS structure in variable-regular Tanner graphs with $d_v = 4$, (b) quasi-normal hypergraphs of all possible LETS structures in an irregular Tanner graph with $d_{v_{\max}} = 4$ with the same normal graph as in Part (a).

a certain range of interest. These advantages come at the expense of a rather small sacrifice in the performance of the designed codes in the waterfall region.

Example 4. *Fig.* **??** *provides all possible LETS structures with* $a = 3$ *and* $a = 4$ *in irregular Tanner graphs with* $g = 6$ *and* $d_{v_{\max}} = 4$*. As can be seen, there are* 10 *and* 45 *structures with* $a = 3$ *and* $a = 4$*, respectively. If variable nodes with degree* 3 *are avoided, the number of structures is reduced to* 4 *and* 10*, respectively. These structures are boldfaced in Fig.* **??***.*

Proposition 5. *In an irregular Tanner graph* G' *with no odd-degree variable nodes, there does not exist any* (a, b') *ETS with odd value of* b'*.*

Proof. For an (a, b') ETS S' in the irregular Tanner graph G', we have $b' = \sum_{i=1}^{a}(d'_{v_i} - d_{v_i})$, where d'_{v_i} is the degree of the i-th variable node in S', and d_{v_i} is the degree of the i-th node in the corresponding normal graph $\hat{G}'(S')$. By the definition of normal graph, we have $\sum_{i=1}^{a} d_{v_i} = 2 \times |E(\hat{G}'(S'))|$, where $E(\hat{G}'(S'))$ is the set of edges of $\hat{G}'(S')$. We thus conclude that $\sum_{i=1}^{a} d_{v_i}$ is an even number. By the assumption of the proposition, $\sum_{i=1}^{a} d'_{v_i}$ is also an even number. Thus, since b' is the difference of the two even numbers, it must also be even. ■

Example **??** and Proposition **??** demonstrate the benefits of not having odd-degree variable nodes in the code's Tanner graph. In our constructions, therefore, we consider irregular LDPC codes without odd-degree variable nodes.

Example 5. *Table* **??** *shows the multiplicity of LETS structures of an irregular Tanner graph with* $g = 6$ *and* $d_{v_{\max}} = 4$*, and without any degree-*3 *variable nodes, within different* (a, b') *classes in the range of* $a \leq 8$ *and* $b' \leq 16$*. As an example, there are* 72 *non-isomorphic LETS structures in the* $(8, 0)$ *class. The quasi-normal hypergraphs of such structures are obtained, based on Proposition* **??***, by finding the inverse images of the normal graphs of all possible* $(8, b)$ *LETS structures with* $b \leq 16$ *of a variable-regular Tanner graph with* $g = 6$ *and* $d_v = 4$*, and then removing those that include any degree-*3 *nodes.*

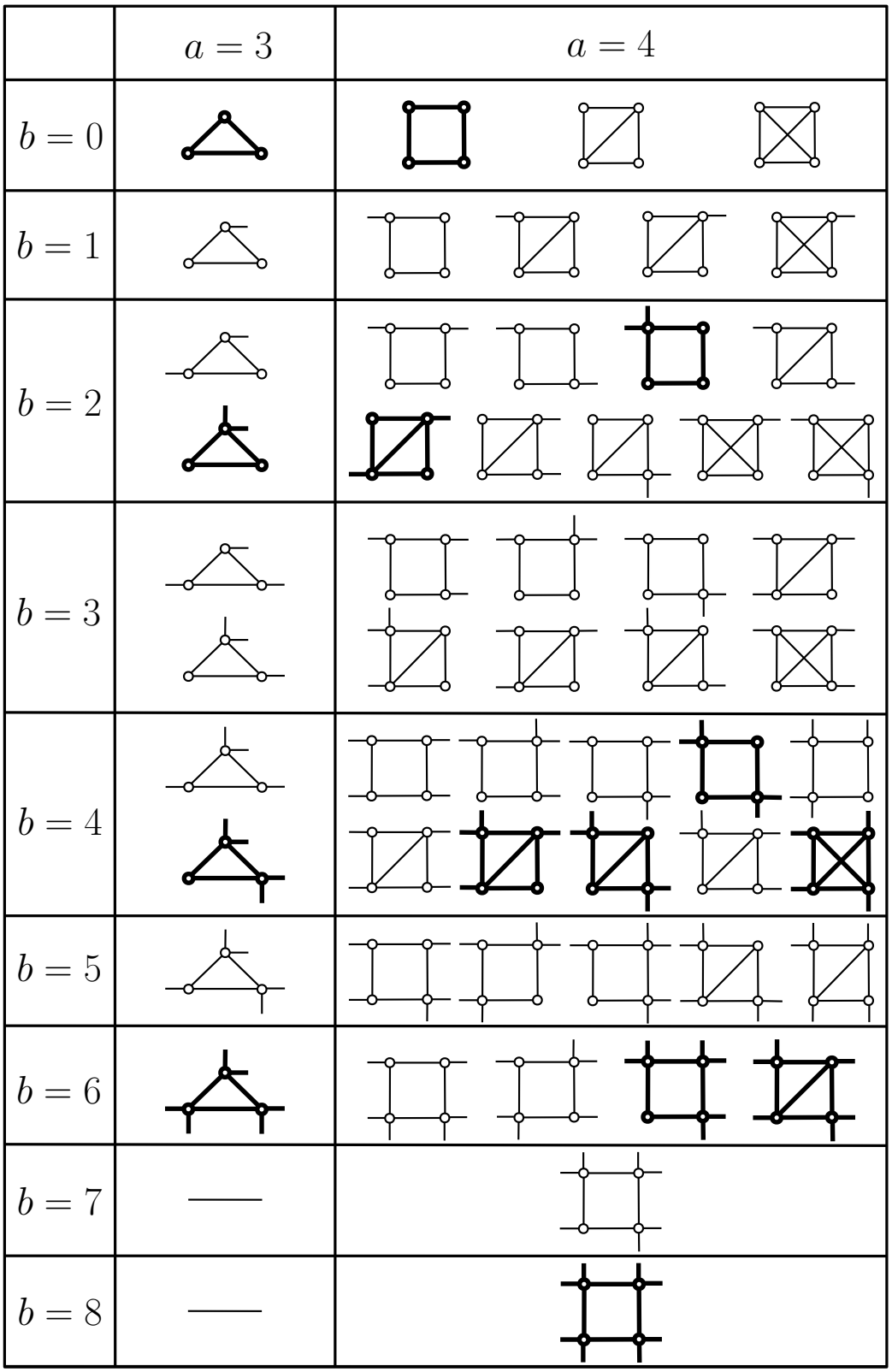


Figure 4.2: The quasi-normal hypergraphs of all possible non-isomorphic LETS structures for irregular Tanner graphs with $g = 6$ and variable node degrees 2, 3 and 4. If variable node degrees are limited to 2 and 4, only the boldfaced structures remain.

Table 4.1: Multiplicities of all possible non-isomorphic (a, b') LETS structures in the range $a \leq 8$ and $b' \leq 16$ of an irregular Tanner graph with $g = 6$ and variable node degrees 2 and 4.

	$a = 3$	$a = 4$	$a = 5$	$a = 6$	$a = 7$	$a = 8$
$b' = 0$	1	1	4	8	21	72
$b' = 1$	0	0	0	0	0	0
$b' = 2$	1	2	8	28	110	510
$b' = 3$	0	0	0	0	0	0
$b' = 4$	1	4	12	50	242	1315
$b' = 5$	0	0	0	0	0	0
$b' = 6$	1	2	9	46	256	1631
$b' = 7$	–	0	0	0	0	0
$b' = 8$	–	1	4	22	147	1105
$b' = 9$	–	–	0	0	0	0
$b' = 10$	–	–	1	6	50	429
$b' = 11$	–	–	–	0	0	0
$b' = 12$	–	–	–	1	9	101
$b' = 13$	–	–	–	–	0	0
$b' = 14$	–	–	–	–	1	13
$b' = 15$	–	–	–	–	–	0
$b' = 16$	–	–	–	–	–	1

4.2.2 Parent/child relationships among LETSs of irregular codes and the determination of minimal target set $\mathcal{L}_t$ of LETSs

After the determination of all the LETS structures $\mathcal{L}$ that are to be avoided in the constructed LDPC code, we need to efficiently solve the decision problem of whether any instances of such structures exists in an irregular Tanner graph. For this, we investigate the parent/child relationships among all the structures in $\mathcal{L}$, with the aim to find a minimal subset $\mathcal{L}_t$ of $\mathcal{L}$ such that solving the decision problem for $\mathcal{L}_t$ would be equivalent to solving it for $\mathcal{L}$. This minimal target set contains all the structures in $\mathcal{L}$ that are not child to any other structure in $\mathcal{L}$. The reason is that, by the definition of $\mathcal{L}_t$, any structure in $\mathcal{L} \setminus \mathcal{L}_t$ is a child to at least one of the structures in $\mathcal{L}_t$ and thus avoiding all the structures in $\mathcal{L}_t$ implies that all the structures in $\mathcal{L} \setminus \mathcal{L}_t$ are also avoided.

Here, we recall the definition of the three *dpl* expansions that govern parent/child relationships among LETSs [4, 5]. The notation dot_m^k is used when a variable node of degree k is appended to a parent structure S through m degree-1 check nodes of $G(S)$ to produce a child structure. Notation pa_m is used for the expansion by a path of length $m+1$ that is appended to the parent structure S through two degree-1 check nodes of $G(S)$. Finally, the notation lo_m^c is used for an expansion of a parent structure S by appending a lollipop walk of length $m+1$ that includes a cycle of length c through a degree-1 check node of $G(S)$. The following example demonstrates the variety of parent/child relationships that can exist among different LETS structures of an irregular code.

Example 6. *Consider the* $(6, 6)$ *LETS structure of irregular Tanner graphs with* $g = 6$ *and* $d_{v_{\max}} = 4$ *depicted in Fig.* **??**. *This structure can be generated through the application of different (series of) expansions to different parent structures. Four such possibilities are shown in Fig.* **??***(a)-(d).*

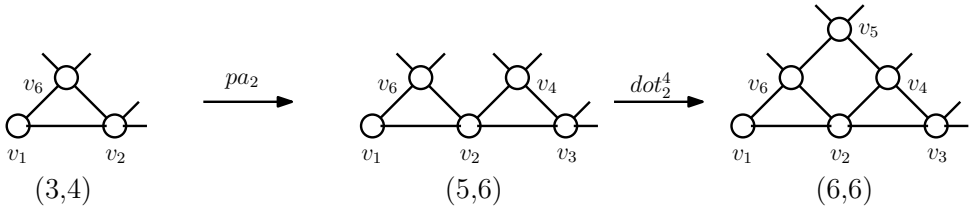
Example 7. *Consider irregular LDPC codes with* $g = 6$ *and variable node degrees* 2 *and* 4*, and the following embedded ranges for* a *and* b *values of LETS classes:* $r_1 : a \leq 3, b \leq 3$, $r_2 : a \leq 4, b \leq 3$, $r_3 : a \leq 5, b \leq 3$, $r_4 : a \leq 6, b \leq 3$, $r_5 : a \leq 7, b \leq 3$, *and* $r_6 : a \leq 8, b \leq 3$. *The sets* $\mathcal{L}_t$ *corresponding to these six ranges are provided in Table* **??**. *The entry* $i \times (j, k)$ *in the table indicates that* i *structures within the* (j, k)

(a)

$V_1 = \{v_1, v_2, v_6\}$ in (3,4)

$\xrightarrow{pa_2}$ $V_2 = \{v_1, v_2, v_6, v_3, v_4\}$ in (5,6)

$\xrightarrow{dot_2^4}$ $V_3 = \{v_1, v_2, v_6, v_3, v_4, v_5\}$ in (6,6)



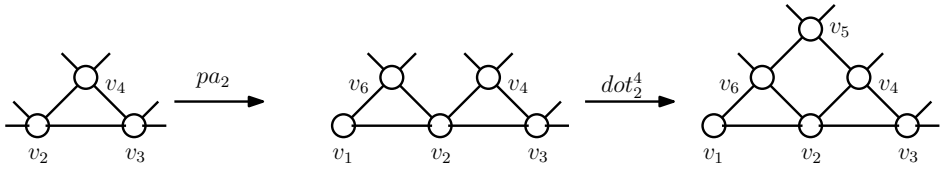
(3,4) (5,6) (6,6)

(b)

$V_1 = \{v_2, v_3, v_4\}$ in (3,6)

$\xrightarrow{pa_2}$ $V_2 = \{v_2, v_3, v_4, v_1, v_6\}$ in (5,6)

$\xrightarrow{dot_2^4}$ $V_3 = \{v_2, v_3, v_4, v_1, v_6, v_5\}$ in (6,6)



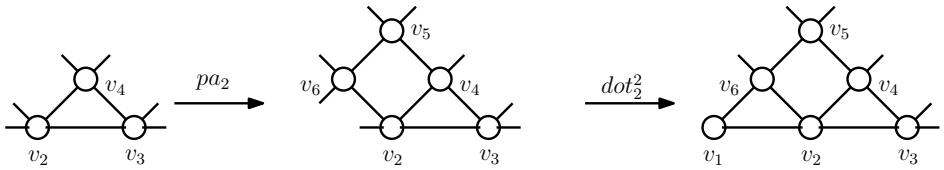
(3,6) (5,6) (6,6)

(c)

$V_1 = \{v_2, v_3, v_4\}$ in (3,6)

$\xrightarrow{pa_2}$ $V_2 = \{v_2, v_3, v_4, v_5, v_6\}$ in (5,8)

$\xrightarrow{dot_2^2}$ $V_3 = \{v_2, v_3, v_4, v_5, v_6, v_1\}$ in (6,6)



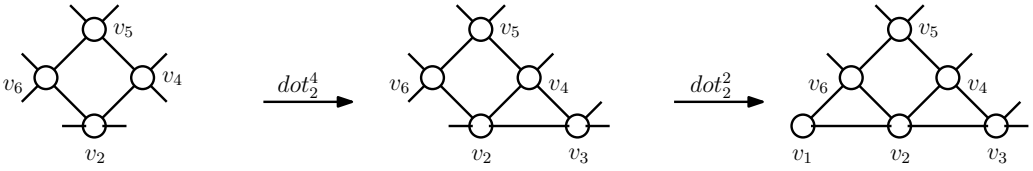
(3,6) (5,8) (6,6)

(d)

$V_1 = \{v_2, v_4, v_5, v_6\}$ in (4,8)

$\xrightarrow{dot_2^4}$ $V_2 = \{v_2, v_4, v_5, v_6, v_3\}$ in (5,8)

$\xrightarrow{dot_2^2}$ $V_3 = \{v_2, v_4, v_5, v_6, v_3, v_1\}$ in (6,6)



(4,8) (5,8) (6,6)

Figure 4.3: Different possibilities that a $(6, 6)$ LETS structure of an irregular Tanner graph can be generated from its parent structures.

Table 4.2: Minimal set of target LETS structures for different ranges of LETS classes in an irregular code with $g = 6$ and variable node degrees 2 and 4

Range of interest	**Minimal target set $\mathcal{L}_t$**
$r_1 = a \leq 3, b \leq 3$	$1 \times (3,0), 1 \times (3,2)$
$r_2 = a \leq 4, b \leq 3$	$1 \times (3,0), 1 \times (3,2), 1 \times (4,0), 2 \times (4,2)$
$r_3 = a \leq 5, b \leq 3$	$1 \times (3,0), 1 \times (3,2), 1 \times (4,0), 2 \times (4,2), 2 \times (5,0), 6 \times (5,2)$
$r_4 = a \leq 6, b \leq 3$	$1 \times (3,0), 1 \times (3,2), 1 \times (4,0), 2 \times (4,2), 2 \times (5,0), 6 \times (5,2), 2 \times (6,0), 17 \times (6,2)$
$r_5 = a \leq 7, b \leq 3$	$1 \times (3,0), 1 \times (3,2), 1 \times (4,0), 2 \times (4,2), 2 \times (5,0), 6 \times (5,2), 2 \times (6,0), 17 \times (6,2)$ $3 \times (7,0),\ 68 \times (7,2)$
$r_6 = a \leq 8, b \leq 3$	$1 \times (3,0), 1 \times (3,2), 1 \times (4,0), 2 \times (4,2), 2 \times (5,0), 6 \times (5,2), 2 \times (6,0), 17 \times (6,2)$ $3 \times (7,0),\ 68 \times (7,2),\ 7 \times (8,0), 302 \times (8,2)$

class are part of the corresponding $\mathcal{L}_t$. We note that for the largest range r_6, we have $|\mathcal{L}_t| = 412$ versus $|\mathcal{L}| = 766$.

4.2.3 Finding the minimal set of out-of-range LETS structures and the layered search algorithm

The next step, after finding the set $\mathcal{L}_t$, is to devise a low-complexity search algorithm that can solve the decision problem of whether any instance of any of the LETS structures in $\mathcal{L}_t$ exists in a Tanner graph. To solve this problem, we examine the out-of-range parent structures of the structures in $\mathcal{L}_t$, and devise a recursive algorithm that can find a minimal set of such structures, going all the way back to simple cycles. The main tool to find such parent structures is the following lemma.

Lemma 2. *[5] In an irregular Tanner graph with girth g, structures in (a, b) LETS class can be generated through expansions dot_m^k with $2 \leq m \leq k$, pa_m with $m \geq 2$, and lo_m^c with $m \geq g/2$ and $g/2 \leq c \leq m$, where the parent structures are respectively in classes $(a-1, b+2m-k)$, $(a-m, b+2-\sum_{i=1}^{m}(d_{v_i}-2))$, and $(a-m, b+2-\sum_{i=1}^{m}(d_{v_i}-2))$, with d_{v_i} being the degree of the i-th variable node in the expansion. If the number of odd-degree check nodes of the parent structure is represented by b', for dot_m^k, pa_m, and lo_m^c expansions, we must have $b' \geq 2$, $b' \geq 2$, and $b' \geq 1$, respectively.*

Example 8. *Consider irregular LDPC codes with $g = 6$ and variable node degrees 2 and 4, and suppose that one is interested in finding all the possible parent classes*

that can generate LETS structures in the $(8,2)$ *class. Using Lemma* **??**, *one can see that LETSs in* $(7,2)$, $(7,4)$, $(7,4)$, *and* $(7,6)$ *classes can potentially generate* $(8,2)$ *structures through* dot_2^4, dot_3^4, dot_2^2, *and* dot_4^4, *respectively. For path expansions,* pa_2 *on* $(6,2)$ *and* $(6,4)$ *structures,* pa_3 *on* $(5,2)$ *and* $(5,4)$ *structures,* pa_4 *on* $(4,2)$ *and* $(4,4)$ *structures, and* pa_5 *on* $(3,2)$ *and* $(3,4)$ *structures may generate LETS structures in the* $(8,2)$ *class. Finally,* lo_3^3, $\{lo_4^3, lo_4^4\}$, *and* $\{lo_5^3, lo_5^4, lo_5^5\}$ *can possibly result in* $(8,2)$ *LETS structures from* $(5,2)$, $(4,2)$, *and* $(3,2)$ *structures, respectively.*

To find the out-of-range structures required for searching the structures in $\mathcal{L}_t$, we start from the structures in $\mathcal{L}_t$ with the largest size, and work our way down to the structures with the smallest size. At the first step, we store the largest size structures of $\mathcal{L}_t$ in $\mathcal{S} = \mathcal{S}_1$, and among the set of all possible out-of-range direct parent structures $\mathcal{P}$ of these structures, we look for a subset Π_1 that can generate all the structures in $\mathcal{S}$ and has the minimum cardinality $|\Pi_1|$, and a higher chance of having a smaller multiplicity of instances in a code. In the selection process of structures in Π_1, if there are two parent classes (a_1, b_1) and (a_2, b_2) for the same child class in $\mathcal{S}$, we prioritize them as follows: (i) if $a_1 = a_2$, the class with smaller b has higher priority; (ii) if $b_1 = b_2$, then the class with smaller a has higher priority; (iii) if $a_1 < a_2$ and $b_1 < b_2$, the priority is given to (a_1, b_1); (iv) finally, if $a_1 > a_2$ and $b_1 < b_2$, we give the priority to trapping set class (a_1, b_1). Moreover, within a given parent class, the priority is given to the structures with the largest number of children in $\mathcal{S}$. The reason for the choices (i)-(iv) is that the multiplicity of LETSs often increases with the increase in the size and the number of odd-degree check nodes, with the latter often having a more dominant effect. At the second step, we store the structures in $\mathcal{L}_t$ with the second largest size in $\mathcal{S}_2$, and choose $\mathcal{S} = \mathcal{S}_2 \cup \Pi_{temp}$, where Π_{temp} are those structures in Π_1 that have the same size as those in $\mathcal{S}_2$. We then continue by finding the set of parent structures Π_2 of $\mathcal{S}$ following the same criteria as explained for Π_1. This ends the second step. This recursive process continues until we cover all the LETS structures in $\mathcal{L}_t$ down to those with size $a = g/2$. At step i, we have $\mathcal{S} = \mathcal{S}_i \cup \Pi_{temp}$, where Π_{temp} consists of all the structures in $\Pi_1 \cup \cdots \cup \Pi_{i-1}$ that have the same size as those in $\mathcal{S}_i$. A pseudo-code of this process in given in Algorithm **??**. The output of the algorithm is the set $\Pi = \Pi_1 \cup \cdots \cup \Pi_\eta$, where η is the number of different TS sizes in $\mathcal{L}_t$.

Finally, to make the search algorithm more efficient, similar to [62], we implement the algorithm in multiple layers, where all the structures in Π and $\mathcal{L}_t$ are partitioned

into groups according to the size of the LETS structures. Groups containing smaller size structures are then searched for earlier in the process. If an instance of a LETS structure in one group is found in the graph, the search will stop without any need to continue the search for structures within the groups that contain larger size structures. For further memory reduction, in each layer of the search algorithm, we only keep those LETSs that are needed for the next layer(s) of the search algorithm.

Example 9. *Consider the case of Example* **??**, *where all LETSs in the range* r_6 *are to be avoided. The set* $\mathcal{L}_t$ *for* r_6 *is given in the last row of Table* **??**. *Accordingly, the set* $\mathcal{S}_1$ *consists of* 7 *structures in the* $(8,0)$ *class and* 302 *structures in the* $(8,2)$ *class. Set* $\mathcal{P}$ *in this case consists of structures in* $(3,4),(4,4),(5,4),(6,4),(7,4)$, *and* $(7,6)$ *classes which generate the structures of* $\mathcal{S}_1$ *through* pa_5, pa_4, pa_3, pa_2, $\{dot_2^2, dot_3^4, dot_4^4\}$, *and* dot_4^4, *respectively. These parent classes are also sorted in the same priority order that they will be selected as part of* Π_1. *Following Algorithm* **??**, *the set* Π_1 *contains* 1, 4, 7, 38 *and* 28 *structures in* $(3,4)$, $(4,4)$, $(5,4)$, $(6,4)$, *and* $(7,4)$ *classes, respectively. For the second step of the recursion, we have* $\mathcal{S}_2 = \{3 \times (7,0), 68 \times (7,2)\}$, *following the same notation as in Table* **??**. *In this step,* $\mathcal{S} = \mathcal{S}_2 \cup \Pi_{temp}$, *where* Π_{temp} *contains the* 28 $(7,4)$ *structures of* Π_1. *For this set* $\mathcal{S}$, *the parent structures* $\mathcal{P}$ *are in classes* $(3,4),(4,4),(5,4),(6,4),(3,6),(4,6),(5,6)$, $(6,6)$, *and* $(6,8)$, *with corresponding expansions* pa_4, pa_3, pa_2, $\{dot_2^2, dot_2^4, dot_3^4, dot_4^4\}$, pa_4, pa_3, pa_2, $\{dot_2^2, dot_3^4, dot_4^4\}$, *and* $\{dot_4^4\}$, *respectively. In this case the set* Π_2 *has* 1, 4, 10, 7, 4 *and* 4 *structures in* $(3,4)$, $(4,4)$, $(5,4)$, $(6,4)$, $(5,6)$, *and* $(6,6)$ *classes, respectively. This completes the second step of recursion. We continue this process until we reach down to TSs of size* $a = g/2 = 3$. *The resulting characterization/search process is summarized in Table* **??**. *In Table* **??**, *we have listed all the required LETS structures in different classes and their corresponding expansions to search for all the target LETS structures of* $\mathcal{L}_t$. *For each class, the top entry shows the multiplicity of non-isomorphic structures within the class that are involved in the search process. The bottom entry of each class shows the expansions that will have to be applied to all the instances of those structures. In some classes, there are two top and bottom entries. In such cases, the first (second) top entry corresponds to the first (second) expansion(s) in the bottom entry. For example, for the* $(6,4)$ *class, we need to apply* $\{dot_3^4, dot_4^4\}$ *and* pa_2 *to* 7 *and* 38 *structures, respectively.*

In Table **??**, *the expansions corresponding to different layers of the algorithm are*

Algorithm 2 Finding the out-of-range parent structures needed for the proposed algorithm to search for LETS structures within $\mathcal{L}_t$

Input: $\mathcal{L}_t$ ▷ Structures in $\mathcal{L}_t$ belong to classes with sizes $a_1, a_2, \ldots, a_\eta$, such that $a_1 < a_2 < \ldots < a_\eta$.
Initialization: $\mathcal{S} = \emptyset$, $\Pi_{temp} = \emptyset$.
for $k = 0, \ldots, \eta - 1$ **do**
    $\Pi_k \leftarrow \emptyset$.
end for
for $i = \eta, \ldots, 1$ **do**
    $\mathcal{S}_{\eta-i+1} \leftarrow$ LETS structures in $\mathcal{L}_t$ with size a_i.
    $\mathcal{S} = \mathcal{S}_{\eta-i+1} \cup \Pi_{temp}$.
    $\mathcal{P} \leftarrow$ all possible direct parents of structures in $\mathcal{S}$. ▷ For each value of i, structures in $\mathcal{P}$ belong to classes $(a_j^i, b_j^i), j = 1, \ldots, \theta_i$, where classes with a smaller index j have a higher priority.
    for $j = 1, \ldots, \theta_i$ **do**
        $\Gamma_j \leftarrow$ structures in $\mathcal{P}$ that are in class (a_j^i, b_j^i).
        while $\Gamma_j \neq \emptyset$ **do**
            Choose $\xi \in \Gamma_j$ with the largest number of direct children in $\mathcal{S}$.
            $\Pi_{\eta-i+1} = \Pi_{\eta-i+1} \cup \{\xi\}$.
            Remove ξ from Γ_j, and remove all the direct children of ξ from $\mathcal{S}$.
            if $\mathcal{S} = \emptyset$ **then**
                Break the **for** loop over variable j.
            end if
        end while
    end for
    $\Pi_{temp} \leftarrow$ all structures in $\Pi_{\eta-i+1} \cup \cdots \cup \Pi_1$ with sizes of a_{i-1}.
end for
$\Pi = \Pi_1 \cup \ldots \cup \Pi_\eta$.
Output: Π.

Table 4.3: Proposed characterization/search table of (a, b) LETSs within the range $a \leq 8$ and $b \leq 3$, for irregular LDPC codes with $g = 6$ and variable node degrees 2 and 4 (superscripts on expansions indicate the layer in which the expansions are applied)

	$a=3$	$a=4$	$a=5$	$a=6$	$a=7$	$a=8$
$b=0$	1	1	2	2	3	7
$b=1$	–	–	–	–	–	–
$b=2$	1	2	6	17	68	302
$b=3$	–	–	–	–	–	–
$b=4$	1 – – – $(dot_2^2, dot_3^4)^{(1)}, pa_2{}^{(2)}$ $pa_3{}^{(3)}, pa_4{}^{(4)}, pa_5{}^{(5)}$	4 – – – $(dot_2^2, dot_3^4, dot_4^4)^{(2)}$ $pa_2{}^{(3)}, pa_3{}^{(4)}, pa_4{}^{(5)}$	12 – – – $(dot_2^2, dot_3^4, dot_4^4)^{(3)}$ $pa_2{}^{(4)}, pa_3{}^{(5)}$	7, 38 – – – $(dot_3^4, dot_4^4)^{(4)}$ $pa_2{}^{(5)}$	6, 28 – – – $(dot_4^4)^{(5)}$,$(dot_3^4)^{(5)}$	–
$b=5$	–	–	–	–	–	–
$b=6$	1 – – – $(dot_2^2, dot_3^4)^{(1)}, pa_2{}^{(2)}$	2 – – – $pa_2{}^{(3)}$	1, 4 – – – $(dot_2^2, dot_3^4)^{(3)}, pa_2{}^{(4)}$	4 – – – $(dot_3^4)^{(4)}$	–	–

identified by superscripts equal to the layer number in which they are applied. In this example, the algorithm is implemented in 5 *layers, corresponding to the rows* 2 *to* 6 *of Table* **??**. *The algorithm starts by searching for the structures indicated in the first row of the table. These are all cycles of length* 6. *After this initial step, the expansions are applied in layers. For example in Layer 1, expansions* dot_2^2 *and* dot_3^4 *are applied to the one structure in the* $(3, 4)$ *class, and the one structure in the* $(3, 6)$ *class.*

4.3 Comparisons with the exhaustive search algorithm of [5]

In [5], two techniques for exhaustively finding all (a, b) LETSs of irregular LDPC codes within a certain range $a \leq a_{max}, b \leq b_{max}$, are proposed. Here, we focus on the more efficient technique of the two, which is presented in Section V of [5]. This technique is based on recursively finding upper bounds b_{max}^a on the b values of classes

with different values of $a \leq a_{max}$, starting from $a = a_{max}$ and going all the way down to $a = g/2$. These upper bounds, for a fixed value of $a \in [g/2, a_{max}]$, identify any (a, b) class whose structures can eventually, through *dpl* expansions, result in at least one structure within the range $a \leq a_{max}, b \leq b_{max}$. After the derivation of these bounds, for each class within the range of these upper bounds, one needs to identify all the expansions that if applied to at least one structure within the class can eventually result in at least one structure within the range of interest. The search algorithm is then devised based on the application of all the identified expansions for each class to all the structures within the class by starting from the simple cycles.

Compared to the carefully devised search algorithm in this work, the technique of [5] is significantly less efficient. The main reason is the abundance of redundancy that exists in the search process of [5]. For example, the same child may be generated/searched for through many different sequences of parent/child relationships, or some structures may be searched for that are neither of direct interest themselves nor are parents to any structure of interest.

Example 10. *We consider the same scenario discussed in Example* **??**, *where one is interested in the exhaustive search of all LETSs within the range of* $a \leq 8, b \leq 3$, *for an irregular LDPC code with* $g = 6$ *and variable node degrees* 2 *and* 4. *The characterization table for the search algorithm of [5] is shown in Table* **??**. *In this table, for each class of LETSs, all the expansions at the bottom will need to be applied to all the non-isomorphic structures within the class. The multiplicity of such structures are given as the top entry for each class. In the table, for simplicity of notations, dot is used to represent* $dot_2^2, dot_2^4, dot_3^4$, *or* dot_4^4 *expansions. The multiplicity of dot expansions is then given in brackets. For example, notation* $dot(2)$ *for a given class means that two of the four dot expansions are applied to the structures of that class. Similarly, notation* $lo_m(i)$ *is used to show the application of* i *different* lo_m^c *expansions (with different* c *values). Compared to the proposed characterization of Table* **??**, *one can see that Table* **??** *involves a larger number of structures within a larger range of* a *and* b *values, with many more expansions. For complexity comparison of our proposed search algorithm and the exhaustive search algorithm of [5], we count the number of different LETS structures involved in the search and the different expansions that are applied to them. For this, similar to [62], we calculate the weighted sum of each expansion over different entries of the table, where the weights are the multiplicity of structures within the corresponding class. The results are listed in Table* **??**.

As an example of how the results in Table **??** *are obtained, consider the expansion* pa_4*. By examining different entries of Table* **??**, *one can see that* pa_4 *appears in only two entries corresponding to classes* $(3,4)$ *and* $(4,4)$, *with one and four structures, respectively. For the proposed search, therefore, the result in Table* **??** *is* $1+4=5$. *In Table* **??**, *however,* pa_4 *appears in classes* $(3,2),(3,4),(3,6),(4,2)$, *and*$(4,4)$ *with 1,1,1,2, and 4 structures, respectively. This corresponds to the entry* 9 *in Table* **??** *for the search algorithm of [5]. By comparing the results presented in Table* **??**, *it can be seen that both the variety of the expansions and the number of structures that they are applied to have decreased significantly in the proposed search algorithm compared to the exhaustive search algorithm of [5].*

One should note that further advantage compared to the algorithm of [5] is gained through the layered implementation of the proposed search algorithm.

Finally, we note that, assuming that the degree distribution of the graph and the range of search are fixed, the complexity of the search algorithm of [5] increases linearly with the block length L in the worst case, and is a constant in L on average (not considering the complexity of finding the input simple cycles). Since the search algorithm proposed here is more efficient than the algorithm of [5], its worst-case complexity is at most linear in L and its average complexity is constant in L.

4.4 Construction of Irregular QC-LDPC Codes with Good Waterfall Performance and Low Error Floor

Our goal in this chapter is to design irregular QC-LDPC codes with good waterfall and error floor performance. The problem formulation is to design an LDPC code that is a cyclic N-lifting of a base graph with an $m \times n$ base matrix. The designed code is aimed to havea rate $R \geq R_0 = 1 - m/n = \bar{d_v}/\bar{d_c}$, and girth $g \geq g_0$, to perform well in the waterfall region, and to be free of (a,b) LETSs in the range $a \leq a_{max}$, and $b \leq b_{max}$, where a_{max} is maximized. (The notations $\bar{d_v}$ and $\bar{d_c}$ denote the average variable and check node degrees of the code.) We further note that instead of a range for a and b values, one can use a list $\mathcal{L}$ of LETS structures to be avoided. Another possible design scenario, as related to the error floor performance, is to consider a

Table 4.4: Characterization table of [5] for LETSs in the range $a \leq 8, b \leq 3$ of irregular LDPC codes with $g = 6$, and variable node degrees 2 and 4

	$a = 3$	$a = 4$	$a = 5$	$a = 6$	$a = 7$	$a = 8$
$b = 0$	1	1	4	8	21	72
$b = 1$	–	–	–	–	–	–
$b = 2$	1 – – – pa_2, pa_3, pa_4, pa_5 $lo_3(1), lo_4(2), lo_5(3)$	2 – – – $dot(3), pa_2, pa_3, pa_4$ $lo_3(1), lo_4(2)$	8 – – – $dot(3), pa_2$ $pa_3, lo_3(1)$	28 – – – $dot(3), pa_2$	110 – – – $dot(3)$	510
$b = 3$	–	–	–	–	–	–
$b = 4$	1 – – – $dot(4), pa_2, pa_3, pa_4$ $pa_5, lo_3(1), lo_4(2)$	4 – – – $dot(4), pa_2, pa_3$ $pa_4, lo_3(1)$	12 – – – $dot(4), pa_2, pa_3$	50 – – – $dot(4), pa_2$	242 – – – $dot(3)$	–
$b = 5$	–	–	–	–	–	–
$b = 6$	1 – – – $dot(4), pa_2, pa_3$ $pa_4, lo_3(1), lo_4(2)$	2 – – – $dot(4), pa_2, pa_3, lo_3(1)$	9 – – – $dot(4), pa_2$	46 – – – $dot(4)$	256 – – – $dot(1)$	–
$b = 7$	–	–	–	–	–	–
$b = 8$	–	1 – – – $dot(4), pa_2, pa_3$	4 – – – $dot(4), pa_2$	22 – – – $dot(3)$	–	–
$b = 9$	–	–	–	–	–	–
$b = 10$	–	–	1 – – – $dot(4)$	6 – – – $dot(1)$	–	–

Table 4.5: Complexity comparison between the proposed search algorithm and that of [5]

Expansion	Proposed	[5]
dot	106	2022
pa_2	62	123
pa_3	17	32
pa_4	5	9
pa_5	1	2
lo_3	0	19
lo_4	0	10
lo_5	0	3

specific range $a \leq a_{max}, b \leq b_{max}$, and minimize the lifting degree N such that the designed code has no (a, b) LETS in the range.

Our experiments show that the waterfall performance is mainly determined by the design of the base graph (base matrix) and the error floor performance is highly dependent on the design of the exponent matrix through the choice of permutation shifts. In the following, we discuss these two steps of the design.

4.4.1 Design of base matrix

The design of the base matrix H' involves the determination of column degree distribution, and the construction of the base graph after the degree distribution is determined. The construction of the base graph is equivalent to specifying the exact location of zeros and ones in H'. To determine the column degree distribution, we either use ensemble *threshold* [73], or the frame error rate (FER) of the lifted code in the waterfall region as the metric. Our simulations show that for the vast majority of cases where the block length is larger than about 500 bits, the two approaches result in the same degree distributions. For shorter block lengths, the latter approach provides a more accurate metric for the waterfall performance, see, e.g., [30]. To obtain the threshold of the protograph-based LDPC codes, we use the extrinsic information transfer (EXIT) analysis of [74], referred to as PEXIT analysis. For the FER analysis, our experiments show that, for a given N, different random choices of permutation

shifts have a rather small effect on the waterfall performance. We thus select the permutation shifts randomly to evaluate different degree distributions. We also note that the relative performance of different degree distributions is rather insensitive to the choice of signal-to-noise ratio (SNR) as long as the performance is measured in the waterfall region.

After the column degree distribution of H' is selected, we design the base graph to maximize the girth of the subgraph induced by the variable nodes corresponding to the columns of H' with weight lower than m. Among candidates with the same maximum girth g', we then choose one with the smallest number of g'-cycles. Alternatively, PEG algorithm [15] can be used to construct this subgraph. We call this approach in designing the base graph "cycle conditioning."

To summarize, we evaluate a number of column degree distribution candidates by measuring the threshold of their corresponding base graphs, or the FER performance of their corresponding lifted base graphs in the waterfall region. The base graphs are constructed based on cycle conditioning. For the FER evaluation, the final graphs (codes) are obtained by a random cyclic lifting of the base graphs. The constructed codes are then evaluated for the FER performance in the waterfall region, and we select a column degree distribution/base graph that results in the best FER performance in the waterfall region.

4.4.2 Design of exponent matrix

After the base matrix is designed, we find the non-infinity elements of the exponent matrix P to eliminate all the instances of LETS structures in the range $a \leq a_{max}, b \leq b_{max}$, or in the set $\mathcal{L}$. To design P, similar to chapter **??**, we use a greedy search algorithm with backtracking in which the elements of P are selected column by column randomly. The sub-matrix of the parity-check matrix H corresponding to the selected columns of P is then searched to see if the girth constraint is satisfied, i.e., no cycle of length less than g_0 exists in the corresponding subgraph, and that no instance of the targeted LETS structures exists in the subgraph. To search for the LETSs in the graph corresponding to the sub-matrix of H at each step of the algorithm, we use the proposed layered search algorithm of Section **??**.

4.5 Simulation Results

In this section, we construct some irregular QC-LDPC codes by the method explained in Section **??**, and compare their performance with similar codes in the literature. For simulations, we consider a binary-input AWGN channel, with noise power spectral density $N_0/2$, and a 5-bit min-sum decoder with clipping threshold equal to 2 [70], and maximum number of iterations 100. For each simulated point, at least 100 block errors are collected, and we plot the frame error rate (FER) of the code versus SNR defined as E_b/N_0, where E_b is the energy per information bit. While our design technique has no limitation, in principle, in terms of variable node degrees, here, we often limit the constructions to codes with $d_{v_{\max}} = 4$, and with no degree-3 variable nodes. The choice of $d_{v_{\max}} = 4$ reduces the complexity of the search algorithm significantly at the cost of some rather small degradation in the waterfall performance. The choice of avoiding degree-3 variable nodes is beneficial from both the complexity viewpoint and error floor reduction, and has little effect on the waterfall performance.

As the first experiment, we construct irregular QC-LDPC codes with rate $R \geq R_0 = 0.5$ and $g = g_0 = 6$. For this, we select a 4×8 base matrix. The column weights for the base matrix can thus be 2 or 4. With only two possible weights for the columns of H', there are only 7 possible irregular column degree distributions. Among those, the one with four columns of weight 2 and four columns of weight 4 with the following base matrix happens to have the best FER in the waterfall region:

$$H'_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.1}$$

To design the exponent matrix, we consider a similar irregular QC-LDPC code designed in [34] with the same size of base matrix. In [34], the irregular code was designed by masking a regular code constructed by array dispersion, and with the goal of minimizing the multiplicity of short cycles. The code of [34] has $N = 72$, $R = 0.5$, $g = 6$ and is free of LETSs in the range $a \leq 8, b \leq 2$. The base matrix of the code of [34] has seven columns of weight 3 and one column of weight 2.

We note that the application of PEXIT analysis also results in the same choice

of base matrix, which has a threshold of 0.92 dB. To design the exponent matrix, we consider a similar irregular QC-LDPC code designed in [34] with the same size of base matrix. In [34], the irregular code was designed by masking a regular code constructed by array dispersion, and with the goal of minimizing the multiplicity of short cycles. The code of [34] has $N = 72$, $R = 0.5$, $g = 6$ and is free of LETSs in the range $a \leq 8, b \leq 2$. The base matrix of the ode of [34] has seven columns of weight 3 and one column of weight 2, and a threshold of 1.07 dB. The FER and LETS distribution of this code are presented in Fig. **??** and Table **??**, respectively. At SNR of 4.25 dB, all the LETSs that trap the decoder for this code have $b = 3$ with a values of $5, 6, 7, 8, 9, 10$ and 11. In our designs that follow, all such LETSs will be eliminated as a result of Proposition **??**.

Here, we consider two scenarios. In the first scenario, we aim at constructing a code with base matrix (**??**) that has no LETS in the same range as the code of [34] (which is $a \leq 8, b \leq 2$), and has the minimum lifting degree N. The result is Code $\mathcal{C}_6$ with rate $R = 0.5025$ and $N = 50$, which is much shorter than the code of [34] (400 vs. 576), and has the following exponent matrix:

$$P_6 = \begin{bmatrix} 0 & \infty & \infty & 0 & 0 & 0 & 0 & 0 \\ 0 & \infty & 7 & \infty & 19 & 21 & 35 & 39 \\ \infty & 9 & 1 & \infty & 23 & 35 & 32 & 42 \\ \infty & 35 & \infty & 4 & 22 & 32 & 27 & 46 \end{bmatrix}. \tag{4.2}$$

The FER performance of $\mathcal{C}_6$ is compared with that of the code of [34] in Fig. **??**. It can be seen that although $\mathcal{C}_6$ has slightly worse waterfall performance, due to the shorter block length, the error floor performance is superior to that of the code of [34]. The reason is that $\mathcal{C}_6$ is free of LETSs with $b = 3$ due to the lack of odd-degree variable nodes. The LETS distributions of the two codes are also shown in Table **??**, which demonstrate the superiority of $\mathcal{C}_6$ in the error floor region. It should be noted that based on our simulations at SNR of 4.25 dB, the LETSs that trap the decoder for $\mathcal{C}_6$ all have $b = 2$, with sizes $a \geq 11$.

In the second scenario, we aim at constructing a code that in addition to the range $a \leq 8, b \leq 3$, is also free of LETSs within the range $a \leq 5, b \leq 5$, and has the same

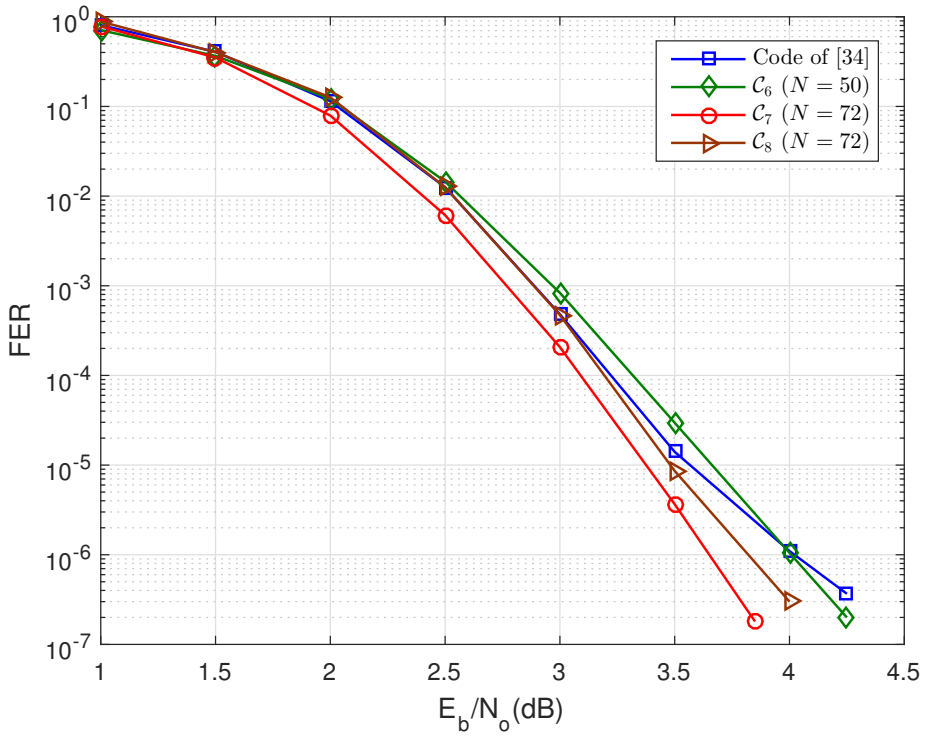


Figure 4.4: FER comparison of the designed codes $\mathcal{C}_6$, $\mathcal{C}_7$, and $\mathcal{C}_8$ with the code of [34].

Table 4.6: Multiplicities of (a, b) LETSs in the range $a \leq 8$ and $b \leq 4$ for designed codes $\mathcal{C}_6, \mathcal{C}_7, \mathcal{C}_8$, and the code of [34]

(a, b) class	Code of [34]	$\mathcal{C}_6$	$\mathcal{C}_7$	$\mathcal{C}_8$
$(3, 3)$	72	0	0	0
$(4, 3)$	72	0	0	0
$(4, 4)$	504	150	0	504
$(5, 3)$	144	0	0	0
$(5, 4)$	1440	400	0	1728
$(6, 3)$	216	0	0	0
$(6, 4)$	1944	850	1656	1512
$(7, 3)$	360	0	0	0
$(7, 4)$	2736	1550	648	2376
$(8, 3)$	360	0	0	0
$(8, 4)$	7164	3025	1404	3888

lifting degree $N = 72$ as the code of [34]. As a result, we design $\mathcal{C}_7$ with $R = 0.5017$. The exponent matrices of $\mathcal{C}_7$ is given by:

$$P_7 = \begin{bmatrix} 0 & \infty & \infty & 0 & 0 & 0 & 0 & 0 \\ 0 & \infty & 4 & \infty & 13 & 30 & 33 & 52 \\ \infty & 17 & 5 & \infty & 39 & 12 & 52 & 47 \\ \infty & 58 & \infty & 64 & 6 & 1 & 24 & 29 \end{bmatrix}. \tag{4.3}$$

The LETS distribution and FER performance of $\mathcal{C}_7$ are shown in Table **??** and Fig. Fig. **??**, respectively. As can be seen, with the same block length and slightly larger rate, $\mathcal{C}_7$ outperforms the code of [34], particularly in the error floor region.

Finally, as the last part of this experiment, we construct a code $\mathcal{C}_8$ that has the exact same base matrix as the code of [34], but is free of all the LETSs within the

range $a \leq 8, b \leq 3$. The exponent matrix of $\mathcal{C}_8$ is given by

$$P_8 = \begin{bmatrix} \infty & \infty & 0 & 0 & 0 & 0 & 0 & 0 \\ \infty & 6 & \infty & 16 & 24 & \infty & 46 & 57 \\ 0 & 1 & 17 & 49 & \infty & 35 & \infty & 26 \\ 0 & 26 & 56 & \infty & 30 & 37 & 34 & \infty \end{bmatrix}. \tag{4.4}$$

The FER and LETS distribution of $\mathcal{C}_8$ are provided in Fig. Fig. **??** and Table **??**, respectively. As can be seen, while $\mathcal{C}_8$ has a waterfall performance similar to the code of [34], its error floor is lower due to the removal of a larger range of LETSs. The performance of $\mathcal{C}_8$ however is still inferior to C_7 both in the waterfall and error floor regions.

As another example, we construct irregular QC-LDPC codes that have similar parameters to the well-known WiMAX standard code with rate $R = 0.6667$ and block length $L = 576$ [2], but have better waterfall and error floor performances. The WiMAX code has a base matrix of size 8×24 and lifting degree $N = 24$. The base graph has seven degree-2, one degree-3, and sixteen degree-4 variable nodes. LETS distribution of this code is shown in Table **??**. As can be seen, the code has many LETS structures with small values of a and b. To construct a code with similar R and L, we consider two different base matrix sizes of 4×12 and 8×24, with $N = 48$ and $N = 24$, respectively. For the two cases, we are able to remove all LETSs within the ranges $a \leq 7, b \leq 3$, and $a \leq 8, b \leq 3$, respectively. The exponent matrices of the designed codes $\mathcal{C}_9$ and $\mathcal{C}_{10}$ are given by:

$$P_9 = \begin{bmatrix} \infty & \infty & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \infty & 1 & \infty & 11 & 16 & 20 & 24 & 28 & 30 & 33 & 38 & 45 \\ 0 & 7 & \infty & \infty & 41 & 31 & 18 & 14 & 33 & 30 & 37 & 46 \\ 0 & \infty & 28 & \infty & 20 & 6 & 46 & 38 & 27 & 34 & 40 & 11 \end{bmatrix},$$

Table 4.7: Multiplicities of LETS structures in the range $a \leq 8, b \leq 3$, for $\mathcal{C}_9$, $\mathcal{C}_{10}$, and the WiMAX code of rate $R = 0.6667$

(a, b) class	WiMAX Code	$\mathcal{C}_8$	$\mathcal{C}_9$
$(3, 2)$	7	0	0
$(4, 2)$	0	0	0
$(4, 3)$	24	0	0
$(5, 2)$	24	0	0
$(5, 3)$	120	0	0
$(6, 2)$	144	0	0
$(6, 3)$	288	0	0
$(7, 2)$	216	0	0
$(7, 3)$	1152	0	0
$(8, 1)$	48	0	0
$(8, 2)$	912	624	0
$(8, 3)$	3768	0	0

$$P_{10} = \begin{bmatrix}
0 & \infty & \infty & \infty & \infty & \infty & \infty & 22 & 0 & \infty & \infty & \infty & \infty & 0 & \infty & 0 & \infty & 0 & 0 & \infty & 0 & \infty & 0 & 0 \\
0 & 23 & \infty & \infty & \infty & \infty & \infty & \infty & 23 & 0 & \infty & \infty & \infty & \infty & 0 & 22 & \infty & \infty & 5 & \infty & \infty & 0 & \infty & \infty \\
\infty & 5 & 16 & \infty & \infty & \infty & \infty & \infty & 7 & 13 & 0 & \infty & \infty & 13 & \infty & \infty & 0 & \infty & \infty & 0 & \infty & 16 & 21 & \infty \\
\infty & \infty & 22 & 1 & \infty & \infty & \infty & \infty & 21 & 18 & 15 & 0 & \infty & \infty & 9 & \infty & 12 & 8 & \infty & 2 & \infty & \infty & \infty & 4 \\
\infty & \infty & \infty & 2 & 2 & \infty & \infty & \infty & \infty & 6 & 23 & 3 & 0 & 7 & \infty & 15 & \infty & \infty & 23 & 19 & 18 & 19 & 3 & \infty \\
\infty & \infty & \infty & \infty & 15 & 11 & \infty & \infty & \infty & \infty & 17 & 10 & 11 & \infty & 14 & 3 & \infty & \infty & \infty & \infty & 17 & 10 & \infty & 19 \\
\infty & \infty & \infty & \infty & \infty & 7 & 18 & \infty & \infty & \infty & \infty & 2 & 10 & 20 & \infty & \infty & 17 & 5 & \infty & \infty & 10 & \infty & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & 2 & 17 & \infty & \infty & \infty & \infty & 1 & \infty & 12 & \infty & 5 & 21 & 8 & 14 & \infty & \infty & 23 & 18
\end{bmatrix}.$$

The two codes have thresholds of 1.66 dB and 1.67 dB, respectively. We have presented the LETS distributions of $\mathcal{C}_9$ and $\mathcal{C}_{10}$ in Table **??**, and their FER performances in Fig. **??**. As can be seen, in comparison with the WiMAX code, both designed codes have a superior LETS distribution and a better performance particularly in the error floor region.

As another example, we consider the irregular $(576, 432)$ QC-LDPC code with a

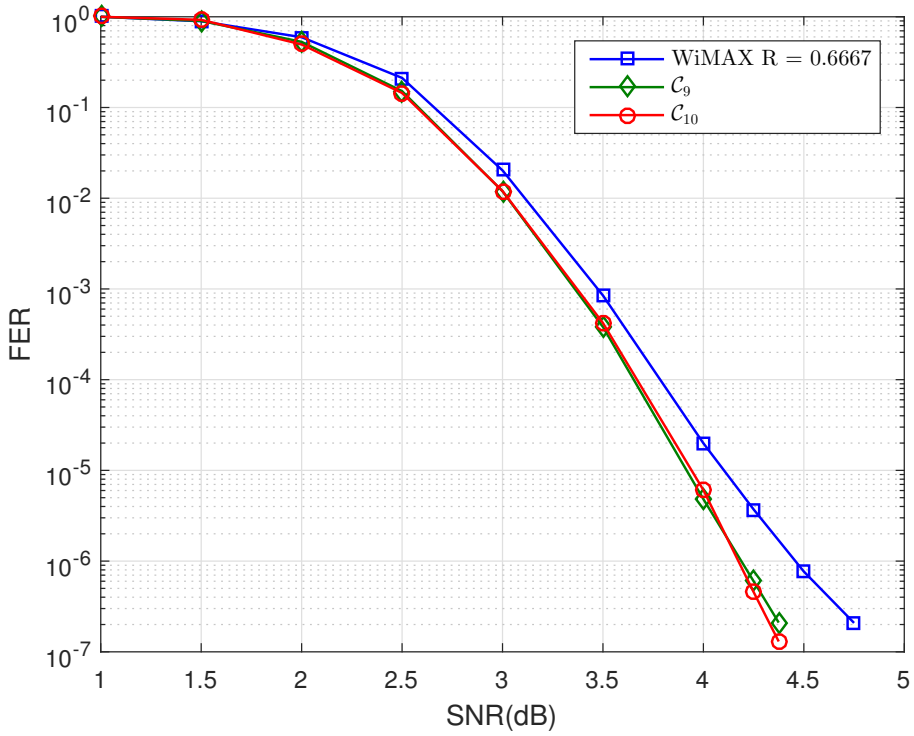


Figure 4.5: FER performance of the designed codes $\mathcal{C}_9$ and $\mathcal{C}_{10}$ in comparison with the similar code from the WiMAX standard.

base matrix of size 4×16, girth $g = 6$, and lifting degree $N = 36$ designed in [34]. The rate of this code is $R = 0.75$, and its base graph has two degree-2, twelve degree-3 and two degree-4 variable nodes. The threshold for this base graph is 2.23 dB. To show the superiority of our design technique, we design a code with exactly the same length, rate, girth, and base matrix size, but with different degree distribution and exponent matrix. Using the method described in Section **??**, we are able to remove all the LETSs within the union of the ranges $a \leq 5, b \leq 3$, and $a \leq 8, b \leq 1$. The exponent matrix of the designed code $\mathcal{C}_{11}$ is as follows:

$$P_{11} = \begin{bmatrix} \infty & \infty & 0 & 0 & \infty & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \infty & 2 & \infty & 5 & 7 & 10 & 12 & 14 & 15 & 17 & 21 & 24 & 25 & 26 & 29 & 31 \\ 0 & \infty & 22 & \infty & 1 & 3 & 23 & 26 & 19 & 4 & 17 & 12 & 20 & 28 & 34 & 13 \\ 0 & 18 & \infty & \infty & \infty & 24 & 3 & 33 & 28 & 5 & 27 & 16 & 23 & 25 & 30 & 2 \end{bmatrix}.$$

The base graph for this code has the threshold of 2.14 dB. We have listed the multiplicities of (a, b) LETSs of both $\mathcal{C}_{11}$ and the code of [34], in range $a \leq 8, b \leq 3$, in Table **??**. As can be seen, none of the dominant LETSs of the code of [34] with $b = 0$, $b = 1$ or $b = 3$ exists in $\mathcal{C}_{11}$. Moreover, the multiplicities of all the LETSs with $b = 2$ are smaller in $\mathcal{C}_{11}$ compared to those in the code of [34]. Consistent with these results are the FER comparison shown in Fig. **??**, where $\mathcal{C}_{11}$ significantly outperforms the code of [34] in the error floor region.

The examples provided so far all had relatively short block lengths. In the following, we provide some examples of codes with larger block lengths.

We consider two codes from ITU-T G.9660 standard [3]: a rate-1/2 code with $L = 8640$ and a rate-2/3 code with $L = 6480$. The base matrices of these codes have sizes 12×24 and 8×24, respectively, and the codes are cyclic liftings of the corresponding base graphs with lifting degrees $N = 360$ and $N = 270$, respectively. The base matrix of the first code has 10 columns of weight 2, 9 columns of weight 3, 5 columns of weight 6, and the threshold of 0.86 dB. For the second code, the base matrix has 6 columns of weight 2, 2 of weight 3, 16 of weight 4, and the threshold of 1.70 dB.

Table 4.8: Multiplicities of (a, b) LETSs in range $a \leq 8, b \leq 3$, for $\mathcal{C}_{11}$ and the code of [34]

(a, b) class	Code of [34]	$\mathcal{C}_{11}$	(a, b) class	Code of [34]	$\mathcal{C}_{11}$
$(3, 2)$	108	0	$(7, 0)$	36	0
$(3, 3)$	504	0	$(7, 1)$	612	0
$(4, 2)$	108	0	$(7, 2)$	12996	4644
$(4, 3)$	2484	0	$(7, 3)$	177768	0
$(5, 2)$	900	0	$(8, 0)$	45	0
$(5, 3)$	8424	0	$(8, 1)$	2340	0
$(6, 1)$	180	0	$(8, 2)$	58104	17172
$(6, 2)$	3204	1188	$(8, 3)$	830448	0
$(6, 3)$	40032	0			

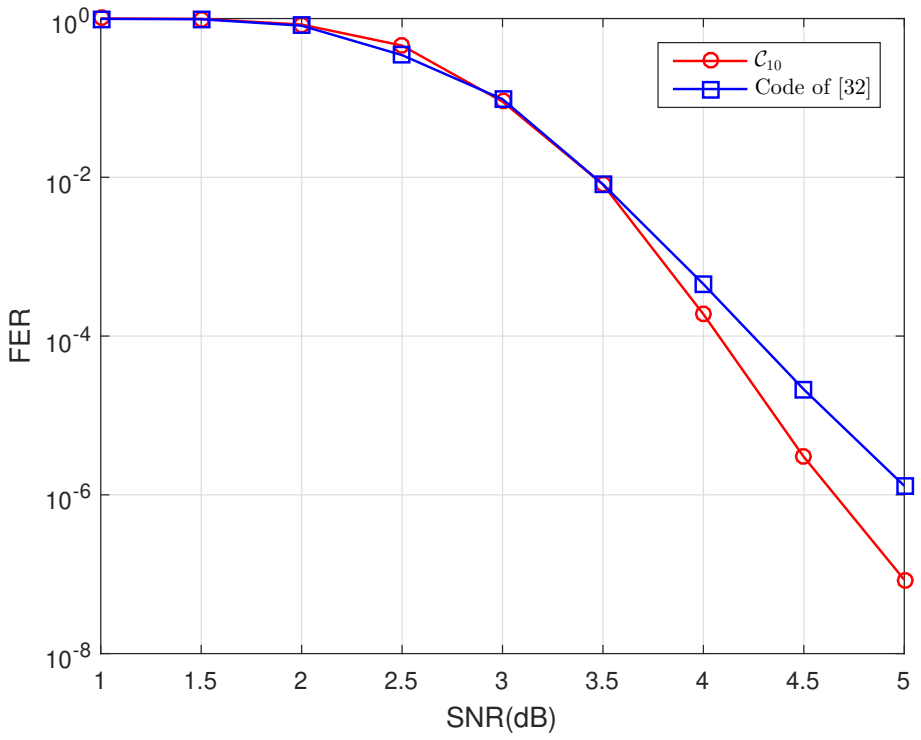


Figure 4.6: FER comparison of the designed code $\mathcal{C}_{10}$ and the similar code of [34].

Our simulations for the first code show that, in the error floor region, the majority of sets that trap the decoder are LETSs within classes $(6,3),(6,4),(7,4),(8,1)$, and $(8,4)$. Using the proposed technique, we have been able to design a code $\mathcal{C}_{12}$ with the exact same base matrix and lifting degree as the standard code but free of LETSs within the range $a \leq 8, b \leq 4$. The exponent matrix of $\mathcal{C}_{12}$ is given by

$$P_{12} = \begin{bmatrix}
\infty & 0 & \infty & 0 & \infty & 0 & \infty & \infty & \infty & \infty & 0 & \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & \infty & 0 & \infty & 0 & \infty & \infty & \infty & \infty & 0 & 298 & 0 & \infty & 39 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
0 & \infty & 40 & \infty & \infty & \infty & \infty & \infty & 0 & 211 & \infty & 64 & \infty & \infty & 230 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & 230 & \infty & 0 & \infty & 183 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 263 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\infty & 36 & \infty & 113 & 197 & \infty & \infty & 0 & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & 230 & 0 & \infty & \infty & \infty & \infty & \infty & \infty \\
303 & \infty & 126 & \infty & \infty & \infty & \infty & \infty & \infty & 11 & \infty & 323 & \infty & \infty & \infty & \infty & \infty & 176 & 0 & \infty & \infty & \infty & \infty & \infty \\
\infty & 55 & \infty & 3 & \infty & \infty & \infty & 289 & 281 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 216 & 0 & \infty & \infty & \infty & \infty \\
\infty & 214 & \infty & 127 & 253 & \infty & \infty & \infty & \infty & \infty & 280 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 188 & 0 & \infty & \infty & \infty \\
188 & \infty & 27 & \infty & \infty & \infty & 233 & \infty & \infty & 5 & \infty & 112 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 75 & 0 & \infty & \infty \\
\infty & \infty & 151 & \infty & \infty & \infty & \infty & 240 & \infty & 56 & \infty & 347 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 202 & 0 & \infty \\
\infty & 240 & \infty & 319 & \infty & 309 & \infty & \infty & 140 & \infty & \infty & \infty & 138 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 38 & 0 \\
\infty & \infty & 297 & \infty & \infty & \infty & 295 & \infty & \infty & 98 & \infty & 70 & 36 & 269 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 103
\end{bmatrix}.$$

The FER curves of $\mathcal{C}_{12}$ and the standard code are given in Fig. **??**. The comparison of the two curves show a large improvement in the error floor region of $\mathcal{C}_{12}$ over that of the standard code.

Our simulations of the rate-2/3 standard code in the error floor region reveal that the dominant LETSs of this code are in the $(4,1)$ class. Using PEXIT analysis, and by employing only degree-2 and degree-4 variable nodes, we find a base matrix with 10 columns of weight 2 and 14 columns of weight 4 that has the best threshold (1.62 dB). We then design the following exponent matrix to eliminate all the LETSs within the range $a \leq 8, b \leq 3$:

$$P_{13} = \begin{bmatrix}
0 & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & 0 & \infty & 0 & 0 & \infty & 0 & \infty & \infty & 0 & 0 & \infty & \infty & 0 & 0 & \infty \\
134 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & 0 & \infty & 0 & 0 & \infty & 30 & \infty & \infty & 16 & 44 & \infty \\
\infty & 70 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 44 & \infty & 37 & \infty & 40 & 20 & \infty & \infty & \infty & 0 & 0 & \infty & 170 & \infty \\
\infty & \infty & 115 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 25 & 69 & \infty & \infty & 76 & \infty & 32 & \infty & 43 & 8 & \infty & 40 & \infty \\
\infty & \infty & \infty & 80 & 0 & \infty & \infty & \infty & \infty & 62 & 16 & \infty & \infty & 29 & \infty & \infty & 21 & 168 & 27 & \infty & 26 & \infty & \infty & 0 \\
\infty & \infty & \infty & \infty & 166 & 0 & \infty & \infty & 0 & \infty & \infty & 150 & 6 & \infty & 105 & \infty & 102 & \infty & 235 & \infty & 11 & \infty & \infty & 42 \\
\infty & \infty & \infty & \infty & \infty & 65 & 0 & \infty & \infty & \infty & \infty & 6 & \infty & 49 & \infty & 28 & 63 & \infty & \infty & 216 & \infty & 27 & \infty & 39 \\
\infty & \infty & \infty & \infty & \infty & \infty & 171 & 206 & 10 & \infty & 20 & \infty & \infty & 233 & 28 & \infty & \infty & 51 & \infty & 136 & \infty & 62 & \infty & 7
\end{bmatrix}.$$

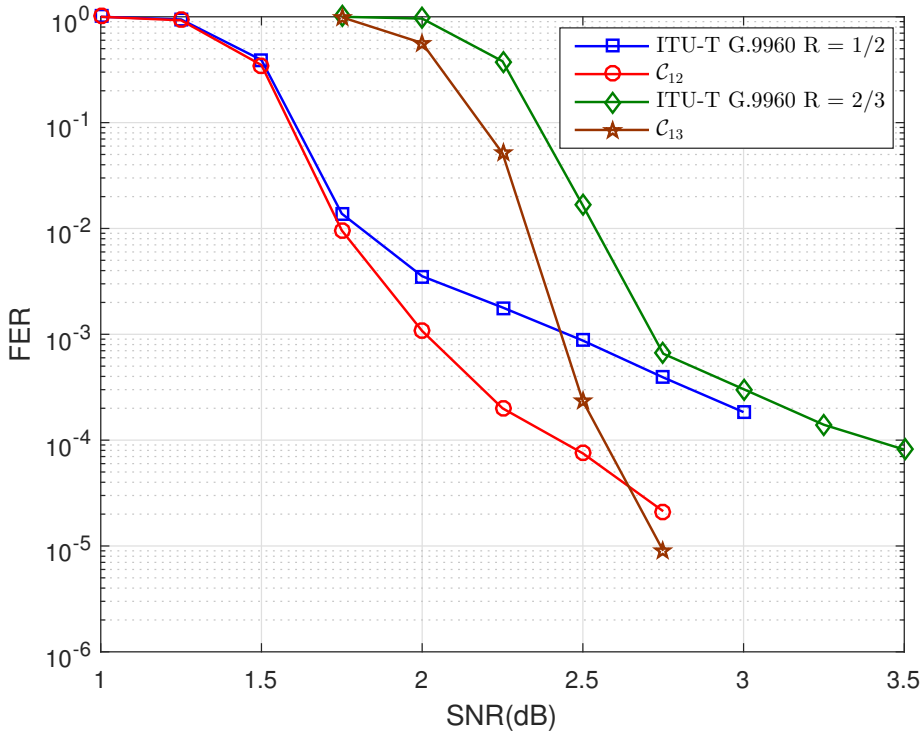


Figure 4.7: FER performance of the designed codes $\mathcal{C}_{12}$ and $\mathcal{C}_{13}$ compared with the similar codes of ITU-T G.9960 standard [3].

The FER of the designed code $\mathcal{C}_{13}$ together with that of the standard code are given in Fig. **??**, and show the significant improvement that the former has over the latter in both error floor and waterfall regions.

As the last example, we consider the rate-1/2 accumulate-repeat-jagged-accumulate (ARJA) LDPC code of CCSDS standard [8]. This code has a 12×20 base matrix, $N = 128$, and $L = 2048$. The base matrix has 4 columns of weight 1, 4 of weight 2, 8 of weight 3, and 4 of weight 6. The bits corresponding to the columns of weight 6 are punctured. This base matrix has a threshold of 0.62 dB. Using the proposed technique, we design a code $\mathcal{C}_{14}$ with the exact same base matrix and lifting degree that is free of LETSs within the range $a \leq 9, b \leq 4$, and has the following exponent matrix:

$$P_{14} = \begin{bmatrix}
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 63 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & 0 \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 34 & \infty & \infty & \infty & \infty & \infty & \infty & 65 & 0 & \infty & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 79 & \infty & \infty & \infty & \infty & \infty & \infty & 91 & 0 & \infty \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 110 & \infty & \infty & \infty & \infty & \infty & \infty & 55 & 104 \\
0 & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & 82 & 43 & 118 & \infty \\
\infty & 0 & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & 76 & 66 & 32 \\
\infty & \infty & 0 & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & 103 & \infty & 7 & 87 \\
\infty & \infty & \infty & 0 & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 27 & 17 & \infty & 111 \\
28 & \infty & \infty & \infty & \infty & \infty & 11 & 2 & \infty & \infty & \infty & \infty & 10 & 3 & \infty & \infty & 119 & \infty & \infty & \infty \\
\infty & 43 & \infty & \infty & 21 & \infty & \infty & 126 & \infty & \infty & \infty & \infty & \infty & 115 & 15 & \infty & \infty & 29 & \infty & \infty \\
\infty & \infty & 57 & \infty & 96 & 20 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 38 & 5 & \infty & \infty & 96 & \infty \\
\infty & \infty & \infty & 17 & \infty & 40 & 119 & \infty & \infty & \infty & \infty & \infty & 53 & \infty & \infty & 19 & \infty & \infty & \infty & 12
\end{bmatrix}.$$

In Table **??**, we have compared the LETS distribution of $\mathcal{C}_{14}$ with that of the standard code in the range $a \leq 10, b \leq 5$. As can be seen, except for the $(8, 5)$ class, in which $\mathcal{C}_{14}$ has a larger multiplicity of TSs, in all the other classes, the multiplicity of TSs are larger for the standard code. Finally, we construct another code $\mathcal{C}_{15}$ with a 12×20 base matrix and $N = 128$, but with a different degree distribution than the standard code. We consider column weights of 1, 2, 4 and 6 for the base matrix and optimize the threshold of the base matrix using PEXIT analysis. As a result, we find a base matrix with the threshold of 0.52 dB that has 4, 4, 8, and 4 columns of weights 1, 2, 4, and 6, respectively. We then design the exponent matrix to eliminate all the LETSs within the range $a \leq 10, b \leq 5$. This code has the following exponent matrix:

$$P_{15} = \begin{bmatrix}
\infty & \infty & \infty & \infty & 0 & 0 & \infty & \infty & 100 & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & 0 & 0 & \infty \\
\infty & \infty & \infty & \infty & \infty & 25 & 0 & \infty & \infty & 81 & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & 79 & 0 \\
\infty & \infty & \infty & \infty & \infty & \infty & 13 & 0 & \infty & \infty & 27 & \infty & \infty & \infty & 0 & \infty & 0 & \infty & \infty & 92 \\
\infty & \infty & \infty & \infty & 10 & \infty & \infty & 17 & \infty & \infty & \infty & 62 & \infty & \infty & \infty & 0 & 7 & 37 & \infty & \infty \\
0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 19 & 124 & 6 & \infty \\
\infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 13 & 104 & 44 \\
\infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 56 & \infty & 64 & 72 \\
\infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 119 & 65 & \infty & 102 \\
28 & \infty & \infty & \infty & \infty & \infty & 82 & 47 & \infty & \infty & \infty & \infty & 1 & 12 & \infty & 7 & 120 & \infty & \infty & \infty \\
\infty & 104 & \infty & \infty & 80 & \infty & \infty & 54 & \infty & \infty & \infty & \infty & 85 & 56 & 23 & \infty & \infty & 25 & \infty & \infty \\
\infty & \infty & 123 & \infty & 123 & 17 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 118 & 123 & 74 & \infty & \infty & 57 & \infty \\
\infty & \infty & \infty & 8 & \infty & 24 & 40 & \infty & \infty & \infty & \infty & \infty & 35 & \infty & 121 & 100 & \infty & \infty & \infty & 9
\end{bmatrix}.$$

In Fig. **??**, we have provided the FER of $\mathcal{C}_{15}$ in comparison with that of the standard code. (Note that all the bits corresponding to degree-6 variable nodes are punctured.) As can be seen, $\mathcal{C}_{15}$ outperforms the standard code in the waterfall region. Moreover, since $\mathcal{C}_{15}$ is free of LETSs within the range $a \leq 10, b \leq 5$, while the standard code has many LETSs within different classes of this range, we expect the error floor of $\mathcal{C}_{15}$ to be much lower than that of the standard code. The FER of $\mathcal{C}_{14}$ is also given in Fig. 9 for comparison. While the waterfall performance of $\mathcal{C}_{14}$ is practically identical to that of the standard code due to having the same base matrix, we expect the error floor of $\mathcal{C}_{14}$ to be lower than that of the standard code based on the superior LETS distribution provided in Table **??**.

Table 4.9: Multiplicities of (a, b) LETSs in range $a \leq 10, b \leq 5$, for $\mathcal{C}_{14}$, $\mathcal{C}_{15}$ and the code of [8]

(a, b) class	Code of [8]	$\mathcal{C}_{14}$	$\mathcal{C}_{15}$
$(4, 4)$	64	0	0
$(5, 4)$	128	0	0
$(5, 5)$	128	0	0
$(6, 4)$	256	0	0
$(6, 5)$	1152	512	0
$(7, 4)$	256	0	0
$(7, 5)$	4480	3200	0
$(8, 4)$	128	0	0
$(8, 5)$	3072	4992	0
$(9, 4)$	128	0	0
$(9, 5)$	3200	2304	0
$(10, 4)$	128	0	0
$(10, 5)$	4224	896	0

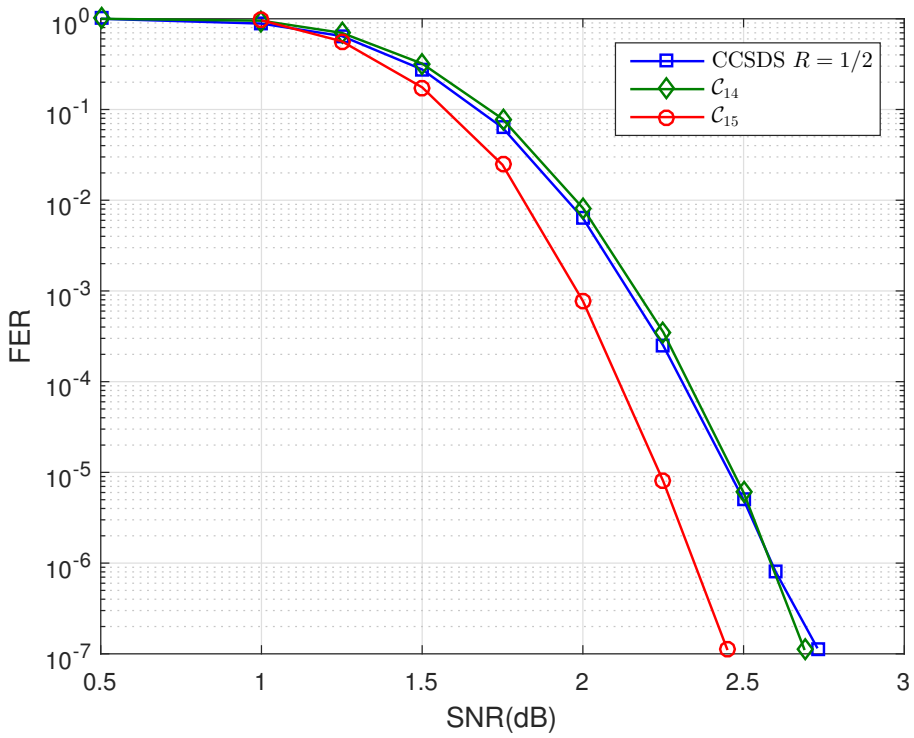


Figure 4.8: FER comparison of the designed codes $\mathcal{C}_{14}$ and $\mathcal{C}_{15}$ and the code of CCSDS standard [8].

Chapter 5

Characterization and Search of Trapping Sets of NB-LDPC Codes

5.1 Introduction

In this chapter, we study the ETS structures of NB-LDPC codes which are responsible for the error floor of NB-LDPC codes when AWGN channel is used [14,59]. As it was mentioned earlier in chapter **??**, in the Tanner graph of non-binary codes, a certain subgraph not only needs to satisfy some topological conditions (check nodes and variable nodes connections) but also the edge weights must satisfy certain algebraic conditions as well. This means that if an induced subgraph satisfies conditions of trapping set topologically, the topology may or may not result in decoding failure depending on how the edge weights are selected from the Galois Field GF(q). The following example illustrates the difference between trapping sets in binary and non-binary LDPC codes.

Example 11. *Fig.* **??***(a) depicts the induced subgraph of a LETS structure in an irregular Tanner graph with minimum variable degree* $\delta(G) = 2$ *and the maximum variable degree* $\Delta(G) = 4$ *(symbols* $\circ$ *and* $\square$ *represents variable nodes and satisfied /unsatisfied check nodes). Considering this graphical structure over* $GF(q)$ *and* $q = 2^p$*, there are five variable nodes and six degree-2 check nodes. Depending on the non-zero values of variable nodes* $(u_1, u_2, u_3, u_4, u_5)$ *and weights of the edges* $\{s_1, \cdots, s_{12}\}$*, this configuration can have* $4 \leq z \leq 10$ *unsatisfied check nodes. The minimum number of unsatisfied check nodes happens when all degree-2 check nodes are satisfied or mathematically*

$$\begin{aligned} u_1 s_1 &= u_2 s_2, \qquad u_2 s_3 = u_3 s_4, \qquad u_3 s_5 = u_1 s_6 \\ u_1 s_7 &= u_5 s_8, \qquad u_5 s_9 = u_4 s_{10}, \quad u_4 s_{11} = u_3 s_{12} \end{aligned} \tag{5.1}$$

where all equations are over $GF(q)$. These conditions leads to the following equations

$$\begin{aligned} s_1 s_3 s_5 &= s_2 s_4 s_6 \\ s_5 s_7 s_9 s_{11} &= s_6 s_8 s_{10} s_{12} \\ s_1 s_3 s_8 s_{10} s_{12} &= s_2 s_4 s_7 s_9 s_{11} \end{aligned} \tag{5.2}$$

where all s values belong to Galois Field of size q. For example, for q=4, Fig. **??***(b) demonstrates an assignment of edge weights satisfying conditions of (***??***). With these weights, there are $q-1$ choices out of $(q-1)^5$ for the set of variable nodes $(u_1, u_2, u_3, u_4, u_5)$ such that all degree-2 check nodes are satisfied and $z = 4$. One example for the values of variable nodes that satisfies these conditions is $(\alpha^2, 1, \alpha^2, 1, \alpha)$ where α is a primitive element of $GF(4)$ based on the primitive polynomial $p(x) = x^2 + x + 1$. If the conditions of (***??***) are not satisfied, the number of unsatisfied check nodes can be in the range of $5 \leq z \leq 10$ and the structure belongs to class $(5, z)$ depending on the values of variable nodes. For example, consider the weights of edges as of Fig.* **??***(c), it can be seen that all degree-2 check nodes are unsatisfied for the same values of variable nodes $(\alpha^2, 1, \alpha^2, 1, \alpha)$ and the configuration is in class $(5, 10)$. Clearly, this structure with that many unsatisfied check nodes is expected to be less problematic in the error floor region.*

Example **??** motivates us to give a clear definition of non-binary trapping sets for NB Tanner graphs. Definition 1 determines the conditions for a subset of a variable nodes to form an (a, b) non-binary trapping set. We also assume that all-zero codeword is transmitted and only these a variable nodes are in error and all other variable nodes outside this structure are 0 ($\in GF(q)$).

Consider an induced subgraph S with a subset of $\mathcal{U}$ variable nodes of size $|\mathcal{U}| = a$ and l check nodes connected to S. We can form matrix A of size $l \times a$ as a sub-matrix of parity-check matrix H corresponding to a columns and l rows of H.

Definition 1. *[14] Structure S is an (a, b) trapping set over GF(q) ($q = 2^p$ and*

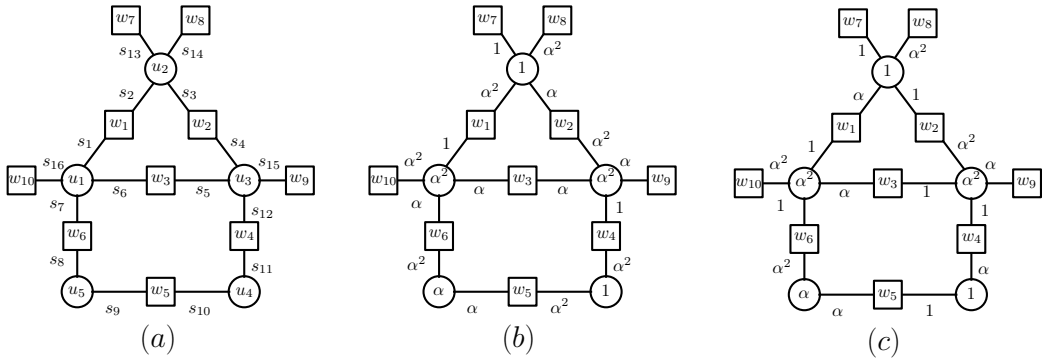


Figure 5.1: Tanner graph of a non-binary structure in an irregular graph with $\Delta(G) = 4$ and $\delta(G) = 2$ (a), structure in class $(5, 4)$ (b), structure in class $(5, 10)$ (c) where α is a primitive element of GF(4) based on the primitive polynomial $p(x) = x^2 + x + 1$.

$p \geq 1$) if there exists an $(l - b) \times a$ sub-matrix B of matrix A where the following conditions are satisfied: (1) Let $N(B)$ be the null-space of matrix B, then there exist a vector $\mathbf{x} = [x_1, \cdots, x_a]^T \in N(B)$ such that $x_i \neq 0$ for all $i \in \{1, \cdots, a\}$; (2) Let D be a sub-matrix of A by excluding matrix B from A and $\mathbf{d_i}$ is the i-th row of matrix D, then $\mathbf{d_i x} \neq 0$ for all $i \in \{1, \cdots, a\}$.

Condition 1 in Definition 1 requires that there exists a vector $\mathbf{x} \in N(B)$ whose elements are all non-zero. This means that there exist a solution for $B\mathbf{x} = 0$ over GF(q) such that all components of the solution have non-zero values. The conditions of the above definition also guarantee that for vector $\mathbf{x}$ in the null-space of B, all of the check nodes associated with the rows of matrix B are satisfied and all of the check nodes corresponding to the rows of matrix D remain unsatisfied (otherwise the structure belongs to class (a, b') where $b' \neq b$).

Remark 2. *Based on the above discussion it can be concluded that if a non-binary trapping set of class (a, b) results in a decoding error, the values of a variable nodes belong to the set of all $\mathbf{x}$'s which satisfy the conditions of Definition 1. Thus, other choices of variable nodes can lead to structures with unsatisfied check nodes value $b' > b$.*

Remark 3. *Considering an (a, b) non-binary ETS (NB-ETS) structures in which all check nodes are of degree-1 or degree-2, Definition 1 reveals that all b unsatisfied check nodes have degree-1. Therefore, all degree-2 check nodes must be satisfied as the structure is in class (a, b). Obviously, the satisfaction of degree-2 check nodes depends on the values of variable nodes.*

The length of shortest NB cycle(s) in the parity-check matrix of a NB code is called *algebraic girth*, and is denoted by g. For a cycle of length $2l$ (C_l) in a NB Tanner graph, we can form a $l \times l$ submatrix B_{C_l} of the parity-check matrix H which corresponds to l variable nodes and l degree-2 check nodes in C_l as follows

$$B_{c_l} = \begin{bmatrix} s_1 & s_2 & 0 & \cdots & 0 \\ 0 & s_3 & s_4 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & s_{2l-3} & s_{2l-2} \\ s_{2l} & 0 & \cdots & 0 & s_{2l-1} \end{bmatrix}. \tag{5.3}$$

where s_i's belong to GF$(q)\backslash\{0\}$. As the l check nodes in B_{C_l} should be satisfied to have a 2l-cycle, we need to check under which conditions there exists a vector $\mathbf{x}$ such that $B_{C_l}\mathbf{x} = 0$ over GF(q). This system of equations has a non-zero solution over GF(q) when square matrix B_{C_l} is singular or $det(B_{C_l}) = 0$ over GF(q). Thus, to satisfy zero determinant the full rank condition (FRC) [52] must hold and can be written as

$$\prod_{i=1}^{l} s_{2i-1} = \prod_{i=1}^{l} s_{2i} \quad \text{over GF}(q). \tag{5.4}$$

As a result of the above discussion, if the FRC condition is satisfied for a cycle of length $2l$, then there exists a solution for the variable nodes of the cycle that results in satisfying all degree-2 check nodes. This FRC condition can be used to simplify the NB-ETS definition. The following lemma shows a necessary and sufficient condition for an induced subgraph of the Tanner graph of a NB-LDPC code to be a NB-ETS.

Lemma 3. *Suppose that structure S is an (a, b) NB-ETS (i.e. the conditions in Definition 1 are satisfied). Then, FRC condition is satisfied for every cycle C_l of structure S with length $2l$ and edge weights s_i where $i \in \{1, \cdots, 2l\}$.*

Proof. Based on the conditions in Definition 1, as S is a non-binary ETS structure there exist a solution for its variable nodes with all non-zero elements that makes all degree-2 check nodes satisfied. Now, consider cycle C_l and corresponding $l \times l$ submatrix B_{C_l} of H. Since all check nodes in B_{C_l} should be satisfied, we can find a non-zero solution $\mathbf{x}$ for variable nodes of matrix B_{C_l} over GF(q) such that $B_{C_l}\mathbf{x} = 0$

which means that FRC condition of (**??**) is satisfied for cycle C_l. The same proof can be done for every cycle of the NB-ETS structure. ■

Now, suppose that G is a finite and connected undirected graph and a vector space is defined on the set of all cycles of G. Also, let E and V be the edge and vertex set of graph G, respectively. Cycles of undirected graph G can be represented by edge-incidence vectors $\{0,1\}^{|E|}$. Addition operation of two cycles C_1 and C_2 ($C_1 \oplus C_2$) is defined by the symmetric difference of edge sets, i.e. $(C_1 \cup C_2) - (C_1 \cap C_2)$. This operation is equivalent to modulo 2 addition of incidence vectors of two cycles. The collection of all cycles forms a vector space over GF(2) with identity function as negation and empty vector as zero which is called *c*ycle space. A set of cycles in G denoted by $\mathcal{B}$ that forms a basis for the cycle space of the undirected graph is called *c*ycle basis. $\mathcal{B}$ is called cycle basis if it has the minimum number of cycles and every cycle in G can be constructed by the combination of cycles in $\mathcal{B}$. The dimension of a cycle basis (or the number of cycles in the cycle basis) for a connected graph G is obtained as $|E| - |V| + 1$. A cycle basis is called *minimum cycle basis* (MCB) if the sum of cycle lengths and the length of the largest cycle(s) are minimized in the cycle basis.

Graph G is called *acyclic* if it has no cycle. A *tree* is a connected acyclic graph. A tree of a graph G is a connected acyclic subgraph of G. A *spanning tree* of a graph G is a tree of G having all the vertices of G. The edges in graph G but not in spanning tree of G are called *chords* of that spanning tree. Note that the spanning tree of a graph is not unique. If a chord is added to the spanning tree of a connected graph G, a unique cycle is generated which is called a *fundamental cycle*. A *fundamental cycle basis* (FCB) is formed from any spanning tree of the given graph G by selecting different chords. Note that not all cycle bases are fundamental and finding minimum FCB is NP-hard.

Remark 4. *Based on Lemma* **??**, *a non-binary ETS not only satisfies the topological condition (i.e. the connection of edges forms an ETS topology), FRC condition of* (**??**) *is satisfied for all cycles of the structure. However, it can be shown that if the FRC condition is satisfied for all the cycles within the cycle basis,* (**??**) *also holds for every cycles in the Tanner graph of the structure. Therefore, to check whether a given set of variable nodes which satisfies certain topological conditions is a non-binary trapping set, we only need to check the FRC condition for the cycles within the cycle basis of the structure as all other cycles can be generated by this set of cycles.*

It is well-established in the literature that the main problematic structures in regular and irregular NB-LDPC codes are ETSs [14, 60]. Thus, in this work, we first extend the binary characterization and search algorithm of trapping sets to the non-binary case for both regular and irregular LDPC codes in **??**. It is worth to mention that the main goals of this paper are: (1) to find the exhaustive list of trapping sets in a given range of interest for certain variable node and check node degree distribution under given Galois field GF(q); (2) to construct regular and irregular NB-QC-LDPC codes free of a certain collection of NB-LETSs within a range of interest. As we will discuss in Subsection **??**, the design process aims at optimizing error floor performance of NB-QC-LDPC codes using a column-by-column construction parity-check matrix of the code free of a certain collection of structures.

5.2 Characterization and Search of LETSs in Regular and Irregular Non-Binary Graphs

Recently, trapping sets have been characterized in [4, 5] using the so-called *dpl characterization* for both regular and irregular binary Tanner graphs. A careful examination of the trapping set structures and their graphical properties in the space of normal graphs reveals that simple graph-based expansions techniques referred to *dot*, *path*, and *lollipop* can generate each and every ETS S by applying a combination of three types of expansions to a sequence of ETSs starting from a simple cycle or a single node. These three expansion techniques have been shown in Fig. **??**. The *dpl* characterization can be used to search for all instances of ETS structures within a range of interest (usually a rectangular region with $a \leq a_{max}$ and $b \leq b_{max}$), exhaustively and with an efficient search algorithm [4, 5]. However, the algorithms for both regular and irregular LDPC codes were designed with respect to binary Tanner graph of the codes. In fact, [4, 5] assume that the edge weights in the Tanner graph of the code are all ones over GF(2). In this section, we consider non-binary graphs and attempt to extend the *dpl* characterization of ETSs for NB-LDPC codes. It is important to note that based on [59], the most harmful structures impacting on the error floor of NB-LDPC codes over AWGN channel are NB-ETSs. Thus, characterizing and removing these kinds of trapping sets can improve the error floor performance significantly. As a result throughout this section, we focus on the NB-ETS structures of the NB-LDPC codes. Also, for regular and irregular Tanner graphs, we use the normal graph and

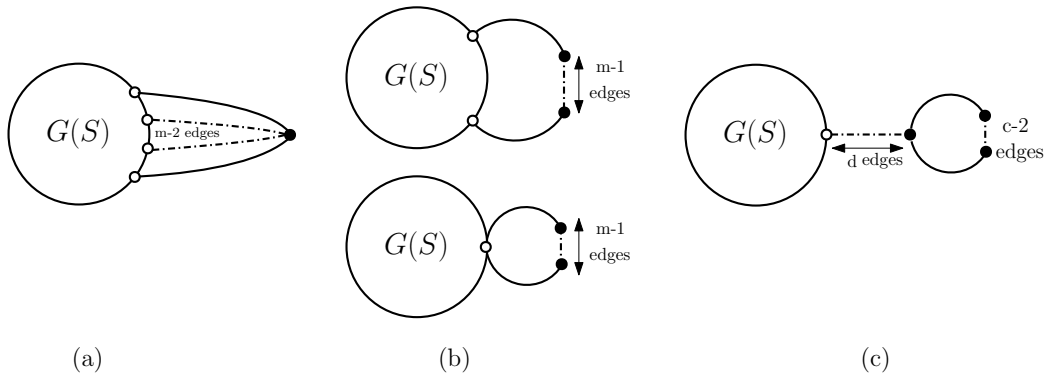


Figure 5.2: Different type of expansions applying on the LETS structure S: dot_m (*dot*) expansion with m edges (a), open (up) and closed (down) pa_m (*path*) expansion with $m+1$ edges (b), lo_m^c (*lollipop*) expansion with $m+1=d+c$ edges (c).

quasi-normal hypergraphs representations of NB-ETSs, respectively.

5.2.1 Regular NB-LDPC Codes

As we discussed in section II, to have a NB-ETS structure in the Tanner graph of the code, Definition 1 states that the structure must satisfy both topological and algebraic conditions at the same time. Based on **??** and Remark **??**, for checking the algebraic conditions one can only evaluate the FRC conditon in (**??**) for the set of cycles within the cycle basis. Also, to minimize the complexity of FRC check process it is necessary to consider the minimum cycle basis for a target NB-ETS structure. It is important to mention that there is a one-to-one correspondence between the cycle basis of an ETS structure in a regular NB Tanner graph and its normal graph representation. Thus, we consider the normal graph of the trapping set structures in this section. The following lemma (Lemma **??**) and theorem (**??**) are useful to find the minimum cycle basis ($\mathcal{B}_{min}$) of a NB-ETS structure and investigate the parent/child relationships in NB Tanner graphs.

Lemma 4. *[78] Suppose that for a connected graph G, $\mathcal{B}$ is a cycle basis. If cycle $C \in \mathcal{B}$ and $C = C_1 \oplus C_2$, then both sets $\mathcal{B}\backslash C \cup \{C_1\}$ and $\mathcal{B}\backslash C \cup \{C_2\}$ also form a cycle basis.*

The following Theorem (**??**) is a direct consequence of Lemma **??**.

Theorem 1. *[78] For a given connected graph G, the minimum cycle basis (MCB) $\mathcal{B}_{min}$ always consists of simple cycles.*

Based on **??**, all cycles in the minimum cycle basis of a potential NB-ETS structure are simple. In [78], Horton proposed a polynomial-time algorithm to find the minimum cycle basis of a finite undirected graph using a greedy search method. After finding the MCB of an structure, the FRC condition should be checked for all the cycles within the cycle basis to figure out whether the structure is a NB-ETS or not. A naive method of finding the distribution of NB-ETS structure in a NB Tanner graph that has been used in [14, 58–60] is to first find all binary structures by only considering topological condition of the structures. Then, for each structure within the range of interest the MCB is obtained and the FRC condition is evaluated. However, this method of finding NB-ETS structures imposes high complexity as the parent/child relationships have not been used in the process of checking algebraic conditions (i.e. FRC conditions).

Considering the topology of a trapping set, one can find the structure by the application of *dot*, *path*, and *lollipop* expansions staring from the short simple cycle of the structure. Obviously, the number of cycles in the MCB is increased after applying an expansion and obtaining the child structure. To minimize the complexity of search and have an optimal cycle basis at each step, we need to investigate the cycle bases of both parent and child structures and choose cycle basis such that the minimal cycle basis property is preserved. In other words, we need to show that there exists a sequence of embedded cycle bases for a sequence of embedded NB-ETS structures that starts from a simple cycle (in the normal graph representation) and is then expanded one step at a time, where at each step a number of cycles are added to the basis depending on the type of expansion used in that step such that the basis remains a MCB. In the following, we investigate the cycle bases for both parent and child structures for all three different expansions proposed in [4] and [5] and describe how the cycle basis of the child is generated as a superset of the cycle basis of the parent where both cycle bases are minimal.

Proposition 6. *Suppose that S is an (a, b) non-binary LETS (NB-LETS) structure of a NB Tanner graph with the MCB of $\mathcal{B}_{parent}$ where the FRC condition is satisfied for all cycles within the cycle basis. The application of dot_m expansion on S increases the dimension of cycle basis of child(ren) $\mathcal{B}_{child}$ by $m-1$ where $\mathcal{B}_{child}$ can be selected such that it is a subset of the cycle basis of child $\mathcal{B}_{child}$.*

Proof. The dot_m expansion can be seen in Fig. **??**(a) in which only one node and m edges are added to the normal graph $\hat{G}(S)$ of the parent structure. Based on the discussions in section II, the dimension of cycle basis of the parent structure S can be achieved as $|\mathcal{B}_{parent}| = |E(\hat{G}(S))| - a + 1$ where $|E(\hat{G}(S))|$ and a are the number of edges and variable nodes in the normal graph of S. As a result, the dimension of child can be calculated as $|\mathcal{B}_{child}| = (|E(\hat{G}(S))|+m)-(a+1)+1 = |\mathcal{B}_{parent}|+(m-1)$. Now we need to show that there exists a cycle basis $\mathcal{B}_{child}$ for child structure which is a superset of $\mathcal{B}_{parent}$. Suppose that the new variable node in the dot_m expansion is represented by u_r and it is adjacent to m variable nodes $\{u_1, \cdots, u_m\}$ in the normal graph of the parent structure. Also, assume that the shortest path between u_i and u_j in $\hat{G}(S)$ is $p(u_i, u_j)$. Let $C_{r,ij}$ be the shortest simple cycle passing through three variable nodes $\{u_r, u_i, u_j\}$ where both u_i and u_j belong to $\{u_1, \cdots, u_m\}$, then there exists $m(m-1)/2$ simple cycles. By selecting $m-1$ shortest cycles out of $m(m-1)/2$ simple cycles that are independent to each other and the cycles in $\mathcal{B}_{parent}$, we can obtain the cycle basis $\mathcal{B}_{child}$ of the child structure which is a superset of the cycle basis of the parent. ■

Based on Proposition **??**, by the application of dot_m expansion on a NB-ETS structure, we only need to check FRC condition for $m-1$ new simple cycles instead of all cycles of $\mathcal{B}_{child}$ which can reduce the complexity of characterization/search algorithm significantly. It is worth to mention that the cycle basis of the child structure is not necessarily minimal even if $\mathcal{B}_{parent}$ is a MCB. Our observations show that if the parent structure is a simple cycle of length $2l$ (denoted by s_l), then based on Proposition **??**, the application of dot_m expansion with $m \geq 3$ on s_l may not preserve the cycle basis minimal. The following example demonstrates how *dot* expansion can affect the cycle basis of the child structures.

Example 12. *Consider the NB-LETS structure of Fig.* **??***(a) in a regular Tanner graph with variable degree $d_v = 4$ and algebraic girth $g = 6$ in which FRC condition is satisfied for the only simple cycle of structure corresponding to the set of variable nodes $\{u_1, u_2, u_3, u_4\}$ and $\mathcal{B}_{parent} = \{s_4\}$. Based on Definition 1, this NB structure belongs to class $(4, 8)$. The application of dot_2 expansion generates two configuration demonstrated in Fig.* **??***(b) and Fig.* **??***(c). However, to check whether these structures are valid NB-LETS we need to evaluate their simple cycles with FRC condition. Based on Proposition* **??***, only one new simple cycle compared to the parent structure is generated which is $\{u_1, u_2, u_5\}$ for the topology of Fig.* **??***(b) and $\{u_1, u_2, u_3, u_5\}$*

in Fig. **??***(c). Note that for the latter one, there are two shortest paths from* u_1 *to* u_3 *where evaluating one of them is enough because the dimension of cycle basis is increased by one using* dot_2 *expansion. In Fig.* **??***(d),* dot_3 *expansion has been applied to the parent structure and it is obvious that three simple cycles (*$\{u_1, u_2, u_5\}$*,* $\{u_2, u_3, u_5\}$*, and* $\{u_3, u_4, u_1, u_5\}$*) are generated as a result of this expansion. However, Proposition* **??** *states that the size of cycle basis is increased by two for* dot_3 *expansion and two simple cycles of length 6 are added to the MCB of the parent. It is easy to see that here* $\mathcal{B}_{child}$ *is also a MCB and we only need to check the FRC condition for two new added cycles. In the case of* dot_4 *expansion (Fig.* **??***(d)), new simple cycles are* $\{u_1, u_2, u_5\}$*,* $\{u_2, u_3, u_5\}$*,* $\{u_3, u_4, u_5\}$*, and* $\{u_4, u_1, u_5\}$*. Using Proposition* **??***, it can be seen that we only need to add three of these cycles to the cycle basis of the child structure and evaluate FRC condition for these new simple cycles. Thus,* $\mathcal{B}_{child}$ *in this case contains 3 cycle of length 6 and one cycle of length 8. As we mentioned before, since the parent structure is a simple cycle, obtained cycle basis* $\mathcal{B}_{child}$ *is not minimal as the 8-cycle can be replaced by a simple cycle of length 6. Thus in this example,* dot_4 *does not preserve the cycle basis minimal. However, by the investigation of structure of Fig.* **??***(d) it can be observed that instead of using* dot_4 *we can generate this structure by application of* dot_2 *and* dot_3 *on* s_3 *that makes the* $\mathcal{B}_{child}$ *a minimal cycle basis. Note that* dot_2*,* dot_3*, and* dot_4 *expansions generate NB-LETSs in* $(5, 8)$*,* $(5, 6)$*, and* $(5, 4)$ *classes.*

Proposition 7. *Suppose that* S *is an* (a, b) $(b \geq 2)$ *NB-LETS structure in a regular NB Tanner graph with minimum cycle basis denoted as* $\mathcal{B}_{parent}$*. Applying the path expansion* pa_m *with* $m \geq 2$ *to* S *generate child structure(s) with minimal cycle basis* $\mathcal{B}_{child}$ *of dimension* $|\mathcal{B}_{parent}| + 1$ *and* $\mathcal{B}_{parent} \subset \mathcal{B}_{child}$*.*

Proof. Applying pa_m expansion adds m new variable nodes and $m + 1$ edges to the normal graph of parent structure. If the number of edges of $\hat{G}(S)$ is $|E(\hat{G}(S))|$, then the dimension of cycle basis of parent structure is obtained as $|\mathcal{B}_{parent}| = |E(\hat{G}(S))| - a + 1$. Thus, the dimension of cycle basis of the child structure is $|\mathcal{B}_{child}| = (|E(\hat{G}(S))| + (m + 1)) - (a + m) + 1 = |\mathcal{B}_{parent}| + 1$. Therefore, the dimension of cycle basis of child structure(s) is increased by one and it is required to check the FRC for only one simple cycle compared to the parent structure. Finding the minimum length simple cycle passing through the path in the normal graph of child results in a minimal cycle basis $\mathcal{B}_{child}$ where it is a superset of $\mathcal{B}_{parent}$. ■

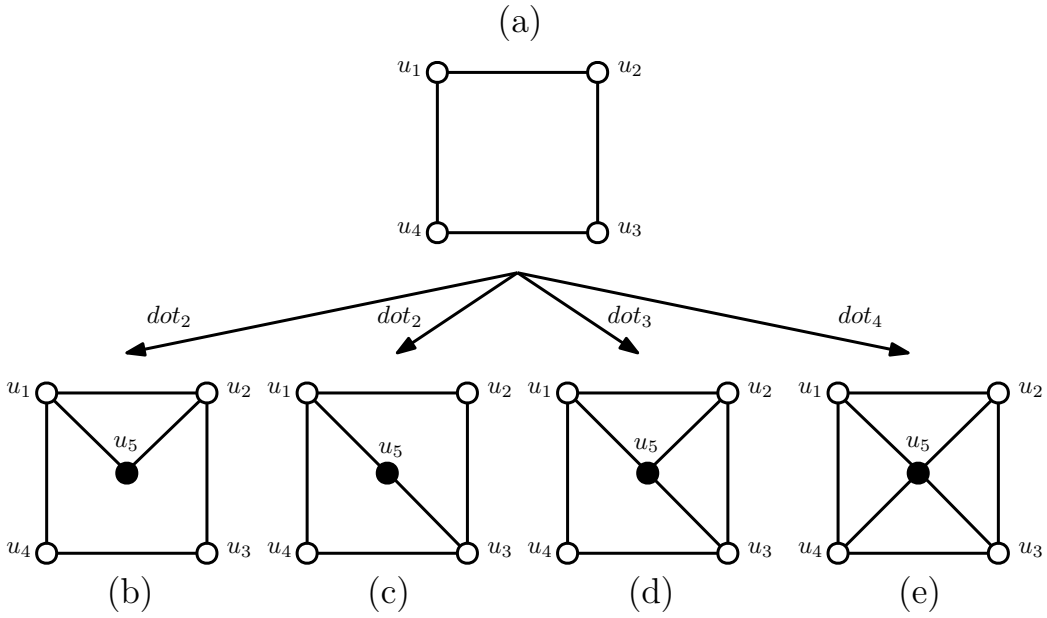


Figure 5.3: LETS structures in a Tanner graph with variable degree $d_v = 4$ and girth $g = 6$: Parent structure in class $(4, 8)$ (a), generated LETSs from dot_2 expansion in class $(5, 8)$ (b)-(c), generated LETS from dot_3 expansion in class $(5, 6)$ (d), generated LETSs from dot_4 expansion in class $(5, 4)$ (e).

Remark 5. *Based on the topology of pa_m expansion depicted in Fig.* **??***(b), the path can be connected to the parent structure via one or two variable nodes. In the former case, the minimum length simple cycle is unique and can be obtained easily. In the case of open path, the shortest path between two connected nodes of parent structure and new added nodes is not necessarily unique and this can lead to having more that one new simple cycle of minimum lengths. However, based on the Proposition* **??***, the dimension of cycle basis is increased by one and therefore it is sufficient to just evaluate the FRC condition for one of the new generated simple cycles. This can be seen more clearly in the following example.*

Example 13. *Consider a regular NB Tanner graph with variable degree $d_v = 4$ and girth $g = 6$. Also, suppose that the simple cycle of length 8 (s_4) depicted in Fig.* **??***(a) satisfies the FRC condition for this simple cycle with set of variable nodes $\{u_1, u_2, u_3, u_4\}$ and $\mathcal{B}_{parent} = \{s_4\}$. The satisfaction of topological and algebraic conditions confirm that this NB-LETS structure is in class $(4, 8)$. The application of pa_2 expansion can generate three different NB-LETS candidates in class $(6, 10)$ which can be seen in Fig.* **??***(b)-(d). Based on Proposition* **??***, the dimension of children cycle basis $\mathcal{B}_{child}$ is increased by one and to find the cycle basis for each child structure we*

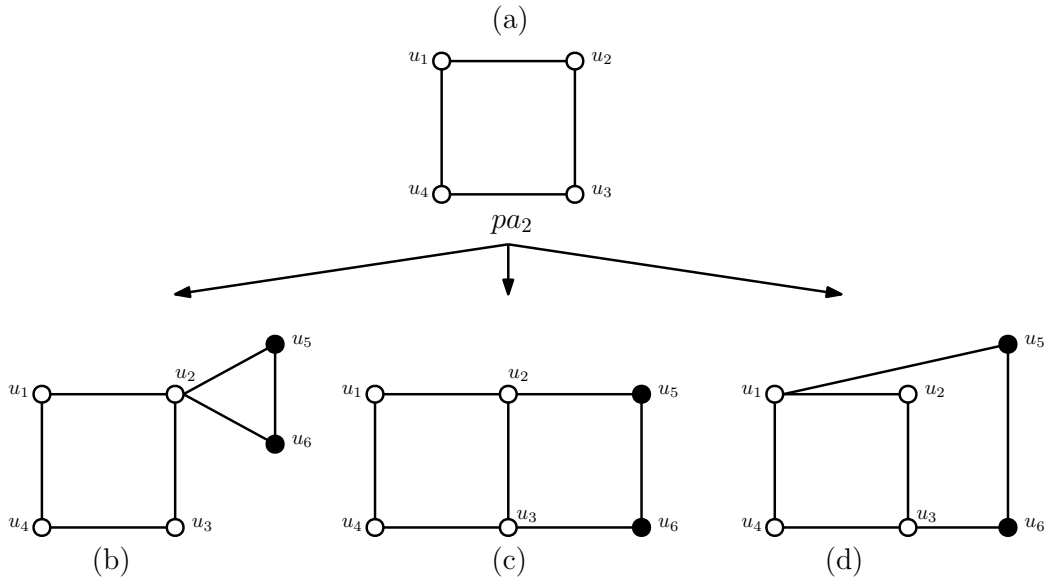


Figure 5.4: LETS structures in a Tanner graph with variable degree $d_v = 4$ and girth $g = 6$: Parent structure in class $(4, 8)$ (a), generated LETSs from pa_2 expansion in class $(6, 10)$ (b)-(d).

need to search for the minimum length simple cycle passing through added nodes in the path expansion. It is easy to see that for child structures of Fig. **??***(b) and Fig.* **??***(c), the new added simple cycle is unique. However, for structure of Fig.* **??***(d) there exist two minimum path which leads to two simple cycles of the same sizes. Thus, we only need to add one of these cycle to achieve the minimum cycle basis $\mathcal{B}_{child}$ as the superset of the cycle basis of the parent structure. Note that in all three generated NB-LETSs, we need to check FRC condition for only one cycle.*

Proposition 8. *Suppose that S is a NB-LETS structure in class (a, b) $(b \geq 1)$ with minimum cycle basis denoted by $\mathcal{B}_{parent}$. Applying the lollipop expansion lo_m^c to S with $m \geq 3$ and $3 \leq c \leq m$ results in child structure(s) with minimal cycle basis $\mathcal{B}_{child}$ of dimension $|\mathcal{B}_{parent}| + 1$ where $\mathcal{B}_{parent} \subset \mathcal{B}_{child}$.*

Proof. The lollipop expansion lo_m^c increases the edges and variable nodes of the normal graph of the parent structure by $m + 1$ and m, respectively. Thus, the dimension of $\mathcal{B}_{child}$ can be calculated as $(|E(\hat{G}(S))| + (m + 1)) - (a + m) + 1 = |\mathcal{B}_{parent}| + 1$. As a result, the dimension is increased by only one. This means that we need to check FRC condition for only one simple cycle compared to the parent structure. It can be seen from Fig. **??**(c) that lo_m^c expansion adds only one unique new simple cycle

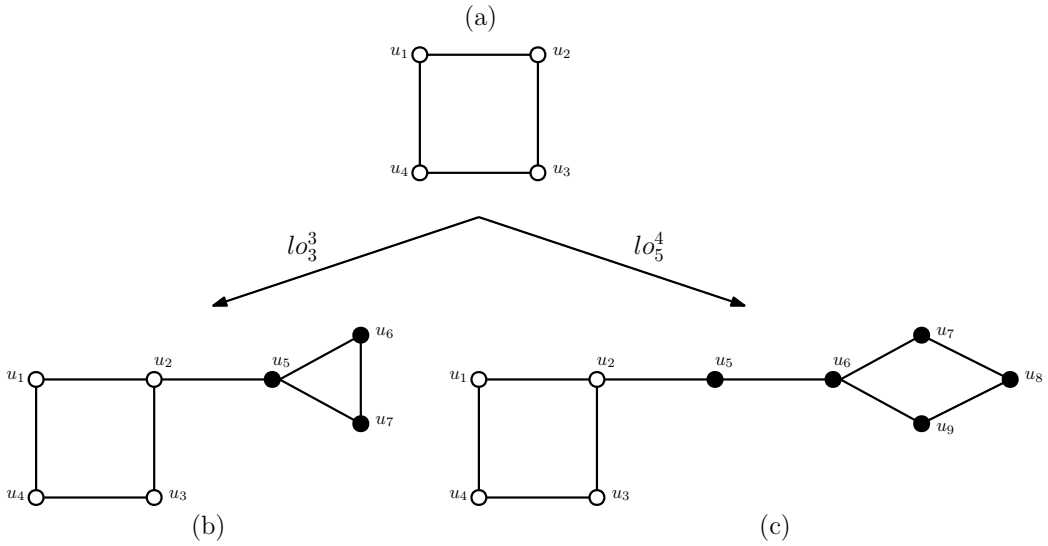


Figure 5.5: LETS structures in a Tanner graph with variable degree $d_v = 4$ and girth $g = 6$: Parent structure in class $(4, 8)$ (a), generated LETSs from lo_3^3 expansion in class $(7, 12)$ (b), generated LETSs from lo_5^4 expansion in class $(9, 16)$ (d).

of length $2c$ to the parent structure. Therefore, the child cycle basis $\mathcal{B}_{child}$ remains minimal. ■

Example 14. *Consider a NB Tanner graph with the same parameter and parent structure of Example* **??**. *The application of lo_3^3 and lo_5^4 on the NB-LETS of Fig.* **??***(a) can possibly generate NB-LETSs in $(7, 12)$ and $(9, 16)$ classes, respectively. The only condition is to check whether new simple cycle in each of the children satisfies FRC condition or not. Based on Proposition* **??** *and Fig.* **??**, *it is clear that applying a lollipop expansion results in only one new simple cycle and increase the number of simple cycles in cycle basis of the child structure by one. Thus, the necessary and sufficient condition for existence of NB-LETSs of Fig.* **??***(b) and Fig.* **??***(c) is the satisfaction of FRC condition for $\{u_5, u_6, u_7\}$ and $\{u_6, u_7, u_8, u_9\}$, respectively.*

By applying a combination of *dot*, *path*, and *lollipop* expansion techniques to short simple cycles, the topology of all NB-LETS structures with variable degree d_v in any (a, b) class (which are not simple cycles) can be obtained [4]. However, different from binary case, FRC condition should be checked after the application of each expansion. Based on Proposition **??**, **??**, and **??**, applying dot_2, *path* and *lollipop* only add one simple cycle to the cycle basis of parent structure and the cycle basis of child remains minimal and a superset of its parent. On the contrary, the

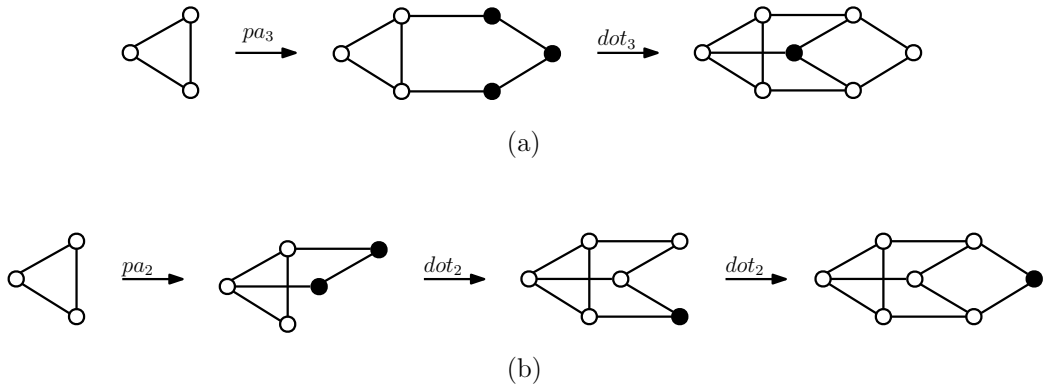


Figure 5.6: $(7, 1)$ NB-LETS structure in a Tanner graph with variable degree $d_v = 3$ and girth $g = 6$ generated from two different sequences of expansions.

application of dot_m expansion with $m \geq 3$ may result in a child structure with a MCB which is not a superset to the MCB of the parent structure. To show this consider a regular Tanner graph with $d_v = 3$, $g = 6$, and the structure in class $(7, 1)$ depicted in Fig. **??**. This structure can be generated by the application of pa_3 and dot_3 on simple cycle of length 6 (s_3) in class $(3, 3)$ and the first structure of class $(6, 4)$, respectively. The MCB of the parent structure is is $\{1 \times s_3, 1 \times s_5\}$ and the MCB for the child structure is $\{1 \times s_3, 3 \times s_4\}$. It can be seen that $\mathcal{B}_{parent} \not\subset \mathcal{B}_{child}$ and the the cycle basis are not embedded anymore. Thus, in general, the application of a series of expansions for generating a target structure does not keep the cycle basis minimal in each step and the complexity of checking FRC condition is not minimized. By a careful look at $(7, 1)$ structure it is observed that this structure has another parent in class $(6, 2)$ which can generate the target structure using dot_2 expansion. Interestingly, the MCB of parent structure is a subset of the MCB of child structure when the $(7, 1)$ NB-LETS is generated from $(6, 2)$ structure. Therefore, to search for any NB-LETSs in a range of interest with minimum amount of FRC check complexity, the embeddedness principle of cycle bases should be satisfied. Moreover, to have an optimal characterization, each structure within the range $a \leq a_{max}$ and $b \leq b_{max}$ must be generated from a parent in class (a, b) where b value is minimized. In the following, we describe our proposed method to optimize the characterization of NB-LETSs and minimizing the complexity of search algorithm in details.

As we mentioned earlier in this section, our main goal is to characterize all non-isomorphic LETS structures of a regular NB Tanner graph with variable degree d_v and algebraic girth g, within a rectangular region $a \leq a_{max}$ and $b \leq b_{max}$. Based on the

discussion in [4], this task can be done by describing each LETS structure S as a series on three types expansion techniques starts from simple cycles. A characterization table is used to demonstrate the necessary expansions for each class of trapping sets in the range of interest. Based on the discussion in [4], from topological point of view, to have an exhaustive list of LETSs in the range of interest, we need some parent structures with unsatisfied check node value $b \geq b_{max}$. The maximum b value of necessary out of range parents is represented by b'_{max}. The *dpl*-characterization proposed by Hashemi and Banihashemi [4] is optimal in the sense that b'_{max} and the maximum length of the required simple cycles are minimized. However, in the case of NB Tanner graphs, it is necessary to check both topological and algebraic conditions at the same time. Thus, each NB-LETS structure must be generated from a direct parent with minimum b value such that the MCB of the parent is a subset of the MCB of the child structure. To fulfill both topological and algebraic conditions, first suppose that all structures within the range of interest are stored in set $\mathcal{L}$. Then, we start from the structures in $\mathcal{L}$ with the largest size and work our way down to the structures with the smallest size. At the first step, the largest size ($a = a_{max}$) structures of $\mathcal{L}$ are identified (denoted by S_1) and stored in the set $S = S_1$. Then, for every structure within S a set of all possible direct parents $\mathcal{P}$ is formed. Among all possible direct parents of structures within S, we look for a subset Π_1 that can generate all the structures in S and has the minimum cardinality $|\Pi_1|$. This subset of direct parents includes structures within and out of the range of interest denoted by $\Pi_{1,in}$ and $\Pi_{1,out}$, respectively. In the second step of our proposed optimization algorithm, the second largest size structures of set $\mathcal{L}$ are stored in set S_2. Here, we need to find the minimum number of parent structures that can generate NB-LETSs within set $S = S_2 \cup \Pi_{1,out}$. To perform this task, similar to the first step, we recognize all possible direct parents and try to find set Π_2 and the corresponding sets $\Pi_{2,in}$ and $\Pi_{2,out}$. This process is continued until we reach to the smallest structure in the range of interest. Note that in the process of evaluating parent structures for any target structure in S, we only consider those parents with minimal cycle bases being a subset of MCB of the target NB-LETS. Therefore, it is guaranteed that the complexity of checking FRC conditions is minimized based on Propositions 1, 2, and 3. It is worth to mention that the optimization method used here is following the same steps as described in Algorithm 1 of [62]. The main difference is that instead of considering a targeted set of structures within the interest range, we are searching for all structures

of the range, i.e. set $\mathcal{L}$.

Remark 6. *The proposed characterization of non-binary LETS structures follows the general framework of dpl characterization of [4], and thus has all advantages of dpl technique. The suggested expansion techniques for NB Tanner graphs minimizes the complexity of checking FRC condition after obtaining each NB-LETS candidate by minimizing the number and length of tested simple cycles in each of dot, path, and lollipop expansions. As a result, we can say that the complexity of proposed characterization/search method is optimized.*

Following the same method as in [4], we can present the characterization of NB-LETSs within a range of interest by only providing the required simple cycles and necessary expansion techniques in each (a, b) class of trapping sets to obtain all instances of NB-LETSs in $a \leq a_{max}$ and $b \leq b_{max}$. However, in addition to the expansion techniques applied on each (a, b) class, we provide the number and lengths of simple cycles that has been checked for FRC condition at each step of generating NB-LETS structures.

Now we are ready to form the characterization table for a given values of variable node degree d_v and algebraic girth g, and a targeted range $a \leq a_{max}$ and $b \leq b_{max}$ of trapping sets. Note that the characterization table corresponds to an exhaustive list of NB-LETSs in the range of interest. Also, as it is well-established in the literature, small trapping sets are among the most harmful ones. Thus, the characterization tables are designed such that all the dominant structures are covered by selecting proper a_{max} and b_{max}. In each characterization table, columns and rows indicate different values of a and b, respectively. For each (a, b) class of LETSs, the first line of top entries in the table specifies the lengths of simple cycles that are initial parents of the LETS structures within that class and the number of structures within that class that have those particular simple cycles as their initial parents which are given in the brackets. The second line of the top entries shows the number of simple cycles of different lengths evaluated for checking FRC condition of LETSs within class (a, b) when we utilize the properties mentioned in Propositions 1, 2, and 3. The third line of the top entries indicates a vector containing the number of simple cyles of different lengths required to check through FRC evaluation in that particular class when all cycles in the cycle bases of structures are evaluated and we do not take advantage of the embeddedness of cycle bases (or the parent/child relationships). Finally, the bottom entries demonstrate the required expansion techniques applied to structures

within the class. For example, consider an (a, b) class with first line top entries, second line top entries, third line top entries, and bottom entries as $\{s_i(x_1), s_j(y_1)\}$, $\{s_{i_1}(q_1), \cdots, s_{i_\gamma}(q_\gamma)\}$, $\{s_{i_1}(k_1), \cdots, s_{i_\lambda}(k_\lambda)\}$, and $\{dot_m, pa_m, lo_m^c\}$, respectively. From the first line of top entries it can be observed that the total of $x_1 + y_1$ non-isomorphic structures in class (a, b) are generated through simple cycles of length-i (x_1 structures) and length-j (y_1 structures). Also, the second line of upper entries indicates that FRC condition should be checked q_1, $\cdots$, q_γ times for simple cycles of lengths s_{i_1}, $\cdots$, s_{i_γ}, respectively. The upper third line shows that when all cycles within the MCBs of structures in the (a, b) class are considered, we need to check FRC condition k_1, $\cdots$, k_λ times for simple cycles of lengths s_{i_1}, $\cdots$, s_{i_λ}, respectively. The expansions in the lower entries mean that it is necessary to apply these expansion techniques to all $x_1 + y_1$ instances of NB-LETSs within class (a, b) for having an exhaustive search within the range of interest. The existence of symbol "$-$" as the only entries of a class or in the lower part shows that there is not any structures in that class or applying any types of expansions either is not possible or would not create new LETS structures in the range of interest. In the following example, the characterization table for a regular NB-LDPC code with variable node degree $d_v = 3$ and algebraic girth 8 is represented.

Example 15. *Table* **??** *demonstrates the characterization of NB-LETS structures of a regular non-binary Tanner graph with variable degree $d_v = 3$ and algebraic girth 8 for $a \leq a_{max} = 12$ and $b \leq b_{max} = 3$. It can be seen that for generating the topology of all instances of LETSs in the range of interest, simple cycles of length eight (s_4) and ten (s_5) are required. In terms of expansion techniques, $dot_2, dot_3, pa_2, pa_3, pa_4$, and lo_4^4 are needed to search for all LETSs within the targeted range. Let us consider the upper and lower entries of class $(9, 3)$ where there are 17 non-isomorphic structures. From the first upper entries ($s_4(16), s_5(1)$) it can be observed that the initial parent of 16 structures is simple cycle of length-8 and only one NB-LETS is generated from s_5. Considering proposed FRC check method, it can be seen from the second line of top entries (5,7,4,1,0) that FRC condition needs to be checked 5, 7, 4, and 1 times for simple cycles of lengths 8, 10, 12, and 14, respectively. In the case that we ignore the parent/child relationship in the process of FRC check and FRC condition is evaluated for all cycles within the MCB of each structure, the third line of upper entries (35,26,6,1,0) shows that FRC condition should be checked 35, 26, 6, and 1 times for simple cycles of lengths 8, 10, 12, and 14, respectively. It is obvious that our proposed*

method reduces the complexity of FRC checking process significantly. Expansion types dot_2, dot_3 *and* pa_2 *can be seen as the bottom entries that should be applied to all 17 structures of class* $(9, 3)$*. It is worth to mention that the characterization table can be divided into two parts; the first part is the classes with* $b \leq b_{max} = 3$ *and the second part with* $b > b_{max} = 3$*. The former classes are called in-range classes and the latter are referred to as out-of-range classes.*

From the characterization table of NB-LETSs, it is clear that checking FRC condition affects the complexity of search algorithm. As we mentioned before, there are two different approaches for dealing with FRC check in the search algorithm. One method is to consider the minimum cycle basis of generated structure and check the FRC condition for all cycles of the MCB. As we mentioned before, this method is useful when you do not have any prior information about the structure. However, in the search algorithm of NB-LETSs, we know that the parent structure fulfills the FRC condition for all cycles and we can take advantage of this information. In the proposed low-complexity method we just consider the new simple cycles generated as a result of applying expansions on the parent structure to minimize the FRC check complexity. To compare the complexity of the proposed algorithm with that of using minimum cycle basis, we count the numbers and the lengths of simple cycles involving in FRC check process in different classes. The comparison can be done by calculating the weighted sum of each simple cycle length over different entries of characterization table with the weights being the multiplicity of simple cycle of certain size in the corresponding class. These results for different scenarios are listed in Table **??**: $d_v = 3, g = 6, a \leq 12, b \leq 5$, $d_v = 3, g = 8, a \leq 12, b \leq 5$, $d_v = 4, g = 6, a \leq 9, b \leq 8$, $d_v = 4, g = 8, a \leq 11, b \leq 10$, and $d_v = 5, g = 6, a \leq 10, b \leq 13$ where g depicts the algebraic girth. For each scenario, we have listed the results for the proposed and MCB FRC check algorithms in the first and second columns, respectively. The results of Table **??** show that the proposed method of checking FRC condition can reduce the complexity of search significantly. For example, considering the first scenario, it is needed to check 22849 simple cycles where this number for proposed method is 3981 which is a huge save in terms of complexity.

Table 5.1: Characterization of NB-LETS Structures of (a, b) Classes for Regular Non-Binary Graphs with $d_v = 3$ and $g = 8$ for $a \leq a_{max} = 12$ and $b \leq b_{max} = 3$

	$a=4$	$a=5$	$a=6$	$a=7$	$a=8$	$a=9$	$a=10$	$a=11$	$a=12$
$b=0$	–	–	$s_4(1)$ $(2,0,0,0,0)$ $(4,0,0,0,0)$ – – – –	–	$s_4(2)$ $(4,0,0,0,0)$ $(9,1,0,0,0)$ – – – –	–	$s_4(5), s_5(1)$ $(7,4,1,0,0)$ $(19,14,3,0,0)$ – – – –	–	$s_4(20), s_5(2)$ $(20,17,5,1,1)$ $(69,62,20,2,1)$ – – – –
$b=1$	–	–	–	$s_4(1)$ $(0,1,0,0,0)$ $(3,1,0,0,0)$ – – – lo_4^4	–	$s_4(4)$ $(1,3,0,0,0)$ $(12,8,0,0,0)$ – – – –	–	$s_4(22), s_5(1)$ $(6,10,5,1,0)$ $(62,61,13,2,0)$ – – – –	–
$b=2$	–	–	$s_4(1)$ $(1,0,0,0,0)$ $(3,0,0,0,0)$ – – – dot_2 pa_2, pa_3	–	$s_4(5)$ $(2,2,1,0,0)$ $(14,5,1,0,0)$ – – – dot_2, pa_3	–	$s_4(27), s_5(1)$ $(9,14,5,0,0)$ $(71,55,13,1,0)$ – – – dot_2	–	$s_4(175), s_5(15)$ $(50,71,58,8,2)$ $(468,442,201,26,3)$ – – – –
$b=3$	–	$s_4(1)$ $(1,0,0,0,0)$ $(2,0,0,0,0)$ – – – dot_2, dot_3 pa_3, pa_4	–	$s_4(3)$ $(2,1,0,0,0)$ $(7,2,0,0,0)$ – – – dot_2, dot_3 pa_2, pa_3	–	$s_4(16), s_5(1)$ $(5,7,4,1,0)$ $(35,26,6,1,0)$ – – – dot_2, dot_3 pa_2	–	$s_4(113)$ $s_5(9)$ $(35,41,38,13,0)$ $(261,232,100,16,1)$ – – – dot_2, dot_3	–
$b=4$	$s_4(1)$ $(1,0,0,0,0)$ $(1,0,0,0,0)$ – – – dot_2, pa_2, pa_3	–	$s_4(2)$ $(1,1,0,0,0)$ $(3,1,0,0,0)$ – – – dot_2 pa_2, pa_3	–	$s_4(9), s_5(1)$ $(5,3,2,0,0)$ $(16,11,3,0,0)$ – – – dot_2, pa_2	–	$s_4(40), s_5(5)$ $(9,16,16,4,0)$ $(113,89,43,6,1)$ – – – dot_2	–	–
$b=5$	–	$s_5(1)$ $(0,1,0,0,0)$ $(0,1,0,0,0)$ – – – pa_2	–	$s_4(2), s_5(1)$ $(1,1,1,0,0)$ $(2,3,1,0,0)$ – – – dot_2, pa_2	–	$s_4(4), s_5(2)$ $(1,2,3,0,0)$ $(24,23,11,2,0)$ – – – dot_2	–	–	–
$b=6$	–	–	–	–	$s_5(2)$ $(0,1,1,0,0)$ $(0,3,1,0,0)$ – – – dot_2	–	–	–	–

Table 5.2: Complexity comparison between two characterization algorithms of NB-LETSs: Proposed method versus Original where the minimum cycle basis (MCB) is considered

	$d_v = 3, g = 6$ $a \leq 12$ and $b \leq 5$		$d_v = 3, g = 8$ $a \leq 12$ and $b \leq 3$		$d_v = 4, g = 6$ $a \leq 9$ and $b \leq 8$		$d_v = 4, g = 8$ $a \leq 11$ and $b \leq 10$		$d_v = 5, g = 6$ $a \leq 10$ and $b \leq 13$	
	Proposed	MCB	Proposed	MCB	Proposed	MCB	Proposed	MCB	Proposed	MCB
s_3	502	5912	0	0	2895	18434	0	0	29620	218663
s_4	1167	6562	163	1198	2679	14893	2031	16002	17983	82644
s_5	1048	5595	195	1037	957	3263	639	3371	1547	2775
s_6	790	3358	139	415	102	241	57	110	13	13
s_7	326	1102	28	56	3	3	1	1	0	0
s_8	127	274	3	6	0	0	0	0	0	0
s_9	21	46	0	0	0	0	0	0	0	0
Total	3981	22849	528	2712	6636	36834	2728	19484	49163	304095

5.2.2 Irregular NB-LDPC Codes

So far, the *dpl*-based characterization/search algorithm for regular NB Tanner graphs has been investigated. As a result, we can find all instances of NB-LETS structures in a range of interest given d_v, algebraic girth g, a_{max}, and b_{max}. As we mentioned earlier, one of the main goals of the current work is to find NB-LETSs in non-binary irregular Tanner graphs. In the rest of this section, we extend the proposed method to the irregular case. Based on our discussion in section II, the variable nodes in irregular Tanner graphs can have various degrees selected from set $\{d_{v,min}, \cdots, d_{v,max}\}$. Thus, the application of different expansion techniques on a parent class (a, b) can generate NB-LETSs in a number of classes with the same size and different unsatisfied check nodes b'. In [5], quasi-normal hypergraph was used to represent the exact number of unsatisfied check nodes of LETSs in irregular Tanner graphs. In this paper, we consider similar graph representation to investigate the topological conditions of NB structures. This subsection aims at the characterization and search of NB-LETS structures in irregular NB Tanner graphs with minimum complexity. Checking topological conditions and algebraic constraints are two dominant factors that impact the complexity of search algorithm. As a result, the optimization of each element would lead to a less complex search algorithm.

The variety of the degrees of variable nodes results in a larger number of non-isomorphic LETS structures of irregular codes for a given (a, b) class compared to regular Tanner graphs which makes the characterization of LETSs a more challenging problem in both binary and non-binary scenarios from topological point of view.

In [5], Considering the interest range as $a \leq a_{max}$ and $b \leq b_{max}$, two solutions for finding an exhaustive list of binary LETSs for irregular LDPC codes are proposed. Focusing on the more efficient technique of the two, a recursive algorithm is introduced to find upper bounds b^a_{max} on the b values of classes with sizes $a \leq a_{max}$, starting from $a = a_{max}$ and going all the way down to $a = g/2$. These upper bounds for each value of $a \in [g/2, a_{max}]$ represents the maximum b value for any (a, b) class whose structures can directly or indirectly generate at least one structure within the range $a \leq a_{max}, b \leq b_{max}$. Finding these upper bounds and identifying all expansions for each class within the range of these upper bounds that leads to at least one structure within the rage of interest, results in an exhaustive search algorithm by applying all expansions within each class to all the structures of that class starting from the simple cycles. However, the method of [5] suffers form high complexity due to high redundancy existing in the search process. In fact, in the proposed characterization technique of [5], the same child structure can be searched for via a number of parents with different expansion types rather than a single parent. In our recent work [63], we established the one-to-one correspondences between children and parents, and we proposed a less complex search algorithm compared to [5] by making sure that each structure within the range is generated through only one parent. To perform this task, after finding the parent/child relationships among all the structures with $a \leq a_{max}$ and $b \leq b^a_{max}$, the minimum number of parents are obtained to find each target LETS structure through only one parent with minimum b vale. However, our approach was devised to find the optimal characterization for a targeted set of LETSs within the range of interest. Different from our previous work [63], in this paper, our goal is to find all instances of structures within the specific range with minimum complexity. Thus, we follow the exact same steps as we proposed in [63] and a larger number of target structures.

As we discussed in the previous subsection and for regular Tanner graphs, in the process of searching for NB-LETS structures, in addition to considering the topological conditions we need to check algebraic constraints as well. Suppose that the topological condition is satisfied for a structure in a NB irregular Tanner graph. Based on Definition 1 and Lemma **??**, if FRC condition is satisfied for every cycle in the minimum cycle basis of the structure, then that structure would be a NB-LETS. We also pointed out that for irregular graphs, a quasi-normal hypergraph is considered. However, for the purpose of checking FRC conditions, it is enough to only evaluate

the normal graph of irregular structure since the degree-1 check nodes are always unsatisfied. Thus, a similar technique to the regular case discussed in the previous subsection can be used to check FRC condition with minimum complexity. In the following example, we form the characterization for an irregular NB Tanner graph with variable node degrees $d_v \in \{2, 4\}$ and algebraic girth $g = 6$ within the range $a \leq 8$ and $b \leq 3$.

Example 16. *The characterization table of an irregular NB Tanner graph with variable degree distribution $d_v \in \{2, 4\}$ and girth $g = 6$ has been demonstrated in Table* **??** *for range $a \leq 8$ and $b \leq 3$. The first step to achieve this table is to extract the number of non-isomorphic structures in each (a, b) class of NB-LETSs in the range of interest. This can be done by using the technique of [63] in which the regular structures are considered as the start point to generated all non-isomorphic structures in different classes. Then, starting from the largest size trapping sets ($a = a_{max}$) within the range of interest, we try to find a set of parent structures with minimal cardinality that can generate all in-range LETSs of size $a = a_{max}$. In the next step, we target the structures with size $a = a_{max} - 1$ within the range of interest along with those parents that we achieved from the previous step which have size equal to $a = a_{max} - 1$. This optimization process can lead to a characterization with minimum number of parent structures out of the range of interest that are able to search for all the structures in the range $a \leq a_{max}, b \leq b_{max}$. Note that, throughout the identification of parent/child relationships, we always consider the FRC condition and the embeddedness principle of cycle bases of the parents and their corresponding children. In fact, a structure is recognized as the potential parent for a certain NB-LETSs within any (a, b) class only if the minimum cycle basis of the parent is a subset of MCB of the target LETS structure. This results in optimizing the complexity of search process in terms of checking algebraic constraints. Similar to the characterization table of Example* **??**, *we consider two different scenarios for checking the FRC conditions. In the first scenario, only new simple cycles in the minimal cycle basis of the child structure compared to the cycle basis of the parent is evaluated. In the second scenario, we do not use the parent/child correspondences and all cycles in the MCB of a child structure are checked in terms of FRC condition. In Table* **??**, *similar to regular case, the second and third line of upper entries in each class show the number of cycle and their corresponding lengths that are required to be checked for FRC condition. As an example, to search for non-isomorphic structures in class $(7, 2)$, cycles of length 6, 8,*

10, 12, 14, and 16 (or equivalently s_3, s_4, s_5, s_6, s_7 and s_8) are needed to be evaluated 54, 53, 22, 2, 1, and 0 times through the proposed FRC check process, respectively. However, when we consider all cycles in the MCB of the structures in this class, we are required to check FRC conditions 311, 141, 24, 2, 1, and 0 time for the same mentioned vector of simple cycles. From the table, it can be seen that the complexity of checking algebraic constraints is much less for the proposed method compared to the MCB approach. In total, we need to check FRC condition for 1643 and 6072 simple cycles for proposed and MCB techniques, respectively, which demonstrates a significant complexity reduction. It is worth to mention that dot_m expansion in Table **??** *represents all dot_m^k expansions with $k \in \{2, 4\}$. For instance, dot_2 represents dot_2^2 and dot_2^4.*

5.2.3 Design of NB-QC-LDPC codes with low error floor

Throughout this section, the characterization/search of NB-LETS structures for both regular and irregular Tanner graphs was discussed thoroughly and an efficient low-complexity algorithm was proposed to find the exhaustive list of structures within a given range of interest. Based on our recent works [62,63], a direct application of being able to characterize trapping sets is to design codes with low error floor by removing harmful structures. Therefore, the same approach proposed in [62,63] can also be used to construct regular/irregular NB-QC-LDPC codes that are free of a certain collection of NB-LETSs denoted by $\mathcal{L}$. As we mentioned earlier, this collection of trapping sets can be also represented by a rectangular range of LETSs with $a \leq a_{max}$ and $b \leq b_{max}$. To specify a NB-QC-LDPC code, we need to determine the exponent and edge weight matrices for a given lifting degree. Similar to what we have done in [62, 63] and to construct NB-QC codes, we can devise a greedy column-by-column search algorithm to jointly design the exponent and edge wight matrices such that at each step the constructed code is free of certain structures within the range of interest. Thus, the search algorithm is required to check whether the constructed Tanner graph so far contains any instances of the NB-LETSs within $\mathcal{L}$. It was mentioned in [62, 63] that using the exhaustive *dpl*-based search algorithm for the specified range and check if there is any instances of LETSs in the output of algorithm would be a solution to this problem which is too complex and not usable. An alternative with lower complexity approach is to only consider those structures that are not parent of other LETSs

Table 5.3: Characterization of NB-LETS Structures of (a, b) Classes for Irregular Non-Binary Graphs with $d_v \in \{2, 4\}$ and $g = 6$ for $a \leq a_{max} = 8$ and $b \leq b_{max} = 3$

	$a = 3$	$a = 4$	$a = 5$	$a = 6$	$a = 7$	$a = 8$
b = 0	$\boldsymbol{s_3(1)}$ (1,0,0,0,0,0) (1,0,0,0,0,0) – – – –	$\boldsymbol{s_4(1)}$ (0,1,0,0,0,0) (0,1,0,0,0,0) – – – –	$s_3(3), \boldsymbol{s_5(1)}$ (5,0,1,0,0,0) (11,0,1,0,0,0) – – – –	$s_3(6), s_4(1)$ (5,4,0,1,0,0) (24,6,0,1,0,0) – – – –	$s_3(18), s_4(2), \boldsymbol{s_7(1)}$ (12,8,4,0,1,0) (71,22,4,0,1,0) – – – –	$s_3(65), s_4(6), \boldsymbol{s_8(1)}$ (36,32,11,4,0,1) (254,136,14,4,0,1) – – – –
b = 1	–	–	–	–	–	–
b = 2	$\boldsymbol{s_3(1)}$ (1,0,0,0,0,0) (1,0,0,0,0,0) – – – pa_2, pa_3, pa_4, pa_5 lo_3^3, lo_4^3, lo_4^4 lo_5^3, lo_5^4, lo_5^5	$s_3(1), \boldsymbol{s_4(1)}$ (1,1,0,0,0,0) (2,1,0,0,0,0) – – – dot_2, pa_3 pa_4, lo_4^4	$s_3(6), s_4(1), \boldsymbol{s_5(1)}$ (7,2,1,0,0,0) (18,3,1,0,0,0) – – – dot_2, pa_2 pa_3	$s_3(24), s_4(3), \boldsymbol{s_6(1)}$ (16,15,2,1,0,0) (71,24,2,1,0,0) – – – dot_2, pa_2	$s_3(99), s_4(9), s_5(1), \boldsymbol{s_7(1)}$ (54,53,22,2,1,0) (311,141,24,2,1,0) – – – dot_2	$s_3(473), s_4(33), s_5(2), s_6(1), \boldsymbol{s_8(1)}$ (244,248,104,27,2,1) (1548,944,152,28,2,1) – – – –
b = 3	–	–	–	–	–	–
b = 4	$\boldsymbol{s_3(1)}$ (1,0,0,0,0,0) (1,0,0,0,0,0) – – – dot_2, pa_2, pa_3, pa_4 pa_5, lo_3^3, lo_4^4	$s_3(2), \boldsymbol{s_4(2)}$ (3,2,0,0,0,0) (5,2,0,0,0,0) – – – dot_2, dot_3, dot_4 pa_2, pa_3, pa_4, lo_3^3	$s_3(9), s_4(1), \boldsymbol{s_5(2)}$ (9,3,2,0,0,0) (22,5,2,0,0,0) – – – dot_2, dot_3, dot_4 pa_2, pa_3	$s_3(42), s_4(5), \boldsymbol{s_6(3)}$ (20,27,6,3,0,0) (101,47,6,3,0,0) – – – dot_2, dot_3 dot_4, pa_2	$s_3(203), s_4(24), s_5(4), s_6(2), \boldsymbol{s_7(3)}$ (103,120,48,7,3,0) (551,326,64,7,3,0) – – – dot_2, dot_3, dot_4	–
b = 5	–	–	–	–	–	–
b = 6	$\boldsymbol{s_3(1)}$ (1,0,0,0,0,0) (1,0,0,0,0,0) – – – dot_2, dot_3, pa_2, pa_3	$s_3(1), \boldsymbol{s_4(1)}$ (1,1,0,0,0,0) (2,1,0,0,0,0) – – – dot_2, dot_3, pa_2	$s_3(6), s_4(1), \boldsymbol{s_5(2)}$ (3,4,2,0,0,0) (12,5,2,0,0,0) – – – dot_2, dot_3, pa_2	$s_3(24), s_4(7), s_5(1), \boldsymbol{s_6(3)}$ (11,27,8,3,0,0) (69,49,8,3,0,0) – – – dot_2	$s_3(183), s_4(31), s_5(11), s_6(5), \boldsymbol{s_7(4)}$ (71,113,64,13,4,0) (460,319,88,13,4,0) – – – dot_4	–
b = 7	–	–	–	–	–	–
b = 8	–	$\boldsymbol{s_4(1)}$ (0,1,0,0,0,0) (0,1,0,0,0,0) – – – dot_2	$s_3(2), s_4(1), \boldsymbol{s_5(1)}$ (1,2,1,0,0,0) (3,3,1,0,0,0) – – – dot_2	$s_3(15), s_4(4), s_5(1), \boldsymbol{s_6(3)}$ (3,11,6,3,0,0) (23,21,6,3,0,0) – – – dot_4	–	–

in $\mathcal{L}$ and form a new set $\mathcal{L}_t$ with minimum number of structures which are called targeted LETSs. Thus, it is possible to devise an efficient version of *dpl*-based search algorithm that could be used in the construction process. As we generalized the *dpl* search algorithm to NB Tanner graphs, we can apply the same method of [62, 63] to construct regular/irregular NB-QC-LDPC codes free of a given list of NB-LETSs.

Here, we formulate the construction problem in two different scenarios considering AWGN channel as the transmission medium. In the first scenario, the goal is to construct a NB code with the given parameters, Galois Field $GF(q)$ and $q = 2^p (p \geq 1)$, base graph size $m \times n$, code rate $R \geq 1 - m/n$, even integer g_0, positive integer b_{max}, and lifting degree N, such that the code has algebraic girth at least g_0 and rate at least R, and is free of any (a, b) NB-LETS structures within the range $a \leq a_{max}, b \leq b_{max}$, where a_{max} is maximized. Another possible design scenario is to consider a fixed range $(a \leq a_{max}, b \leq b_{max})$ and minimize the lifting degree N such that the output non-binary code has no (a, b) NB-LETS within that range of interest. In this case, all other code parameters are assumed the same as first problem. It is worth to mention that in this work, we focus on improving the error floor performance of NB-QC-LDPC codes. Thus, we supposed the degree distributions that result in good waterfall performances are known in advance.

There exist a small number of studies that are aimed at lowering the error floor of NB-LDPC codes. In [52, 56, 57], the authors use indirect measure of error floor such as algebraic girth, multiplicity of short NB cycles, and the connection of short NB cycles to the rest of the graph, to design codes with low error floor. There are also a number of research that deal with error floor by eliminating the harmful structures directly, see, e.g. [14, 58, 59]. In [14], the author proposes a design method by only optimizing the edge weight matrix for a given binary code to eliminate a collection of elementary absorbing sets. The same author mentions the construction problem for specifically NB-QC-LDPC codes in [58]. In [59], the error floor problem is considered for communication channels other than AWGN where non-elementary absorbing sets are also problematic. However, in all of direct methods, the NB code is constructed through a two-step process. At the first step, a binary code is constructed with good girth property (usually using PEG [15] algorithm). Next, the edge wight matrix is first randomly generated and then optimized such that the designed code is free of a given collection of harmful structures. Thus, the edge weight and exponent matrices are constructed separately. In this work, we construct regular/irregular NB-QC-LDPC

codes by jointly optimizing edge weight and exponent matrices using the proposed low-complexity search algorithm..

Remark 7. *While the proposed characterization/search tool works for NB codes over any field size, q, the performance gain achieved by the elimination of trapping sets for larger q are expected to be relatively smaller. The main reason is that by increasing the field size satisfying FRC condition for small cycles are less likely and NB-LDPC code benefits from improved error-floor performance. However, the complexity of decoding algorithm increases quickly by increasing the GF size [14]. Thus, in this paper, we keep the field size relatively small in our design codes.*

5.3 Simulation Results

As we mentioned earlier, finding the distribution of trapping sets for a given NB-LDPC code is important since it provides us with the information of harmful structures that are necessary to improve the error floor performance of NB codes. Better performance can be achieved by constructing codes free of a collection of harmful trapping sets. Thus, the main goal of this section is to report the exhaustive lists of NB-LETS structures for a number of regular/irregular NB-QC-LDPC codes existing in the literature. To perform this task, we bring the multiplicity of instances of NB-LETSs in different classes for these codes in tables. Each row of a table corresponds to a class of NB-LETS structure and for each class the total number of instances of NB-LETSs obtaining from different short non-binary simple cycles is listed. It is important to emphasize that the current work is the first study providing the exhaustive list of LETSs for NB (regular/irregular) LDPC codes. As the application of proposed efficient search algorithm of NB-LETSs, here, we also deal with the problem of constructing NB-QC-LDPC codes with better error floor performance compared to the state-of-the-art codes with similar parameters. In the design process, a column-by-column algorithm is used and to evaluate the distribution of NB-LETSs a modified version of proposed search algorithm is utilized as we discussed in Subsection **??**. It is worth to mention that whenever we talk about regular code we consider a fully-connected base graph. Moreover, since the ultimate goal is to improve the error floor performance, we consider the same degree distribution (or zero elements within the base graph) for the designed codes compared to their counterparts from the literature.

Finally, to compare the designed and the existing NB codes, we provide the distribution of NB-LETSs for all constructed NB-QC-LDPC codes. For FER performance comparison, we consider a binary-input AWGN channel and a $q-ary$ sum-product decoder with maximum number of iteration 30 where at least 100 block errors are collected for each simulated point.

As the first example, we consider a regular NB-QC-LDPC code of length $L = 90$ with 3×5 base matrix, lifting degree $N = 18$, over GF(4) and primitive element of $\alpha^2 + \alpha + 1$. Based on the method of [14], a binary QC-LDPC code is generated using PEG algorithm which results in a code of $g = 6$. Then, the edge weights are assigned randomly to the Tanner graph of the designed code. Finally, the edge weights are being updated repeatedly to remove the trapping sets within the range of $a \leq 8$ and $b \leq 3$. Note that to eliminate NB structures, [14] first finds the list of NB-LETSs using the technique of [61] (which is not necessarily exhaustive). Then, the FRC condition is checked for the cycles within the fundamental cycle of each structure in the design algorithm. The code of [14] canceled all cycles of length 6 by assuring that FRC condition is not satisfied for these cycles which means that the algebraic girth is 8. We consider similar code parameters and attempt to remove a larger range of NB-LETSs by designing code $\mathcal{C}_{16}$. Our constructed code has no NB-LETSs within the union of the ranges $a \leq 10, b \leq 3$ and $a \leq 11, b \leq 2$:

$$H_{16} = \begin{bmatrix} \alpha^0 I^0 & \alpha^2 I^0 & \alpha^2 I^0 & \alpha^1 I^0 & \alpha^0 I^0 \\ \alpha^2 I^0 & \alpha^1 I^{17} & \alpha^8 I^0 & \alpha^0 I^{11} & \alpha^1 I^{15} \\ \alpha^1 I^0 & \alpha^2 I^7 & \alpha^{14} I^0 & \alpha^0 I^4 & \alpha^2 I^9 \end{bmatrix} . \tag{5.5}$$

In Table **??**, we have listed the NB-LETSs of code $\mathcal{C}_{16}$ and the code designed by the algorithm of [14] within the range of $a \leq 12$ and $b \leq 4$. In this table, only classes where at least one of the two codes has non-zero multiplicity are reported. The FER curves for the two codes are also given in Fig. **??**. It can be observed that $\mathcal{C}_{16}$ has superior trapping set distribution and error floor performance.

In the second experiment, we consider the regular NB-QC-LDPC code of [59] with rate $R = 0.89$, variable degree $d_v = 3$, check node degree $d_c = 27$, and block length $L = 3996$ where the base graph is fully-connected. The field size of the designed code is equal to $q = 4$ with primitive element of $\alpha^2 + \alpha + 1$ and lifting

Table 5.4: Multiplicities of LETS structures in the range of $a \leq 12$ and $b \leq 4$ for designed code $\mathcal{C}_{16}$ and the code constructed by technique in [14]

(a, b) class	Code of [14]	$\mathcal{C}_{16}$
$(4, 4)$	234	162
$(6, 4)$	450	270
$(8, 4)$	1143	594
$(9, 3)$	144	0
$(10, 2)$	18	0
$(10, 4)$	4554	3438
$(11, 3)$	288	36
$(12, 0)$	15	0
$(12, 2)$	18	18
$(12, 4)$	9054	5544

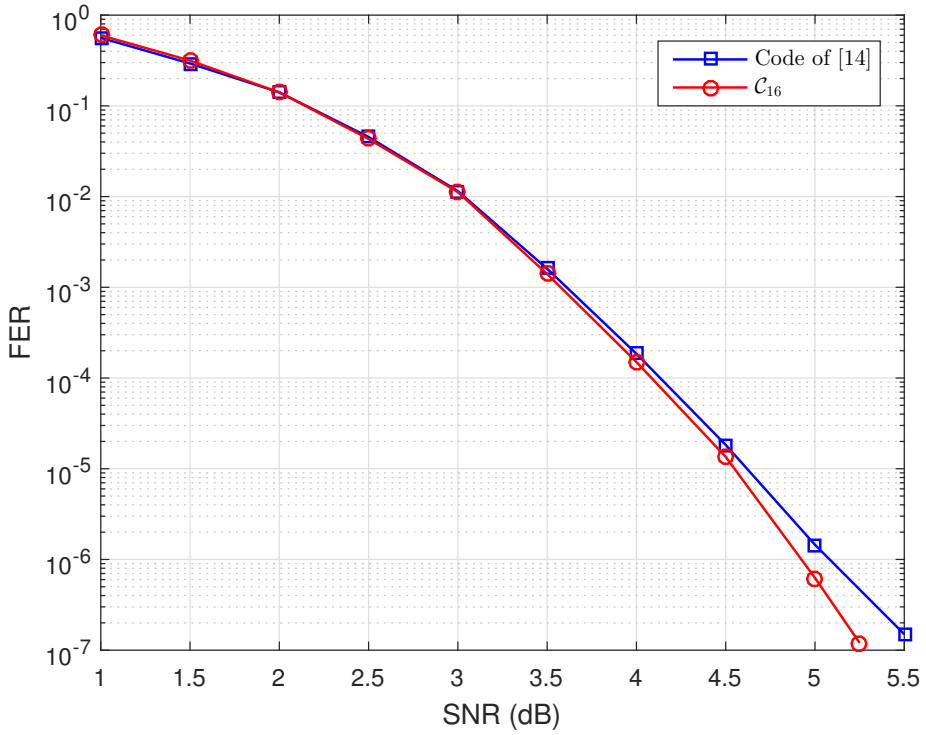


Figure 5.7: FER comparison of proposed code $\mathcal{C}_{16}$ and the NB code designed by technique of [14].

Table 5.5: Multiplicities of LETS structures in the range of $a \leq 8$ and $b \leq 3$ for designed code $\mathcal{C}_{17}$ and codes in [59]

(a, b) class	Random	Code of [59]	$\mathcal{C}_{17}$
$(3, 3)$	5920	3700	1776
$(4, 2)$	888	0	0
$(5, 3)$	34780	25604	15540
$(6, 2)$	6068	2072	0
$(7, 3)$	613312	377548	275576
$(8, 2)$	70152	38924	20868

degree is $N = 148$. The non-binary code was designed by first generating a binary QC-LDPC code using PEG algorithm with girth $g = 6$. Then, the distribution of absorbing sets is obtained using the approach proposed by [61]. In the next step, the edge weights are assigned randomly and an iterative algorithm is devised such that harmful elementary absorbing sets are removed by changing and updating the edge weights such that FRC condition is not satisfied for at least one cycle in each structure. Based on their proposed algorithm, an edge weight is updated such that all previously removed structures remain canceled. In Table **??**, the distribution of NB-LETSs for the designed code of [59] is reported where the author mentioned that the harmful structures are those in $(4, 2)$, $(6, 2)$, and $(8, 2)$ classes. It is important to mention that [59] does not provide any information about the distribution of the elementary absorbing sets. It is possible to achieve a better error floor by removing $(4, 2)$ structures and reducing the number of $(6, 2)$ trapping sets. To compare the distribution of harmful structures, we also provide the list of LETSs for the random edge weight assignment in Table **??** which is the initial code of design algorithm in [59]. The table lists NB-LETSs in any class within the range of $a \leq 8$ and $b \leq 3$ where the multiplicity of structures is non-zero for at least one of the codes.

Here to improve the performance in the error floor region, we aim at constructing a NB-QC-LDPC code $\mathcal{C}_{17}$ that has similar parameter to the code of [59], but with better LETS distribution using the proposed method explained in section III.C. We set our goal to design a code with no NB-LETSs in the range of $a \leq 7$ and $b \leq 2$ with algebraic girth greater than or equal to 6. Our observations and simulations show that this is the maximum $a_{\max}$ for $b_{\max} = 2$. Using a column-by-column construction algorithm,

we search for appropriate permutation shifts and corresponding edge weights in each column to obtain a code without any instances of NB-LETSs within the targeted range. The parity-check matrix of code $\mathcal{C}_{17}$ is as follows.

$$H_{17} = \begin{bmatrix} \alpha^0 I^0 & \alpha^0 I^0 & \alpha^1 I^0 & \alpha^2 I^0 & \alpha^1 I^0 & \alpha^2 I^0 & \alpha^1 I^0 & \alpha^2 I^0 & \alpha^1 I^0 & \alpha^1 I^0 & \alpha^1 I^0 & \alpha^0 I^0 & \alpha^1 I^0 \\ \alpha^0 I^0 & \alpha^0 I^5 & \alpha^2 I^{145} & \alpha^0 I^{128} & \alpha^1 I^{104} & \alpha^1 I^{109} & \alpha^0 I^6 & \alpha^1 I^{13} & \alpha^2 I^{142} & \alpha^1 I^{54} & \alpha^0 I^{135} & \alpha^2 I^{63} & \alpha^2 I^{96} \\ \alpha^0 I^0 & \alpha^0 I^{134} & \alpha^2 I^{138} & \alpha^2 I^{48} & \alpha^2 I^{25} & \alpha^0 I^{143} & \alpha^2 I^{67} & \alpha^2 I^{145} & \alpha^1 I^{12} & \alpha^2 I^{66} & \alpha^1 I^{24} & \alpha^0 I^{22} & \alpha^1 I^{44} \end{bmatrix}$$

$$\begin{bmatrix} \alpha^1 I^0 & \alpha^1 I^0 & \alpha^2 I^0 & \alpha^2 I^0 & \alpha^1 I^0 & \alpha^1 I^0 & \alpha^2 I^0 & \alpha^2 I^0 & \alpha^0 I^0 & \alpha^1 I^0 & \alpha^0 I^0 & \alpha^1 I^0 & \alpha^0 I^0 \\ \alpha^2 I^{23} & \alpha^1 I^8 & \alpha^2 I^{58} & \alpha^1 I^{86} & \alpha^1 I^{100} & \alpha^1 I^{49} & \alpha^0 I^{83} & \alpha^2 I^{119} & \alpha^1 I^{84} & \alpha^0 I^{117} & \alpha^1 I^{123} & \alpha^2 I^{36} & \alpha^1 I^{16} \\ \alpha^0 I^{15} & \alpha^0 I^{53} & \alpha^1 I^{74} & \alpha^1 I^{144} & \alpha^2 I^{28} & \alpha^1 I^{40} & \alpha^2 I^{17} & \alpha^1 I^{130} & \alpha^0 I^{60} & \alpha^1 I^{78} & \alpha^2 I^{81} & \alpha^0 I^{99} & \alpha^1 I^1 \end{bmatrix}. \tag{5.6}$$

Table **??** lists the NB-LETS distribution of code $\mathcal{C}_{17}$ in the range of $a \leq 8$ and $b \leq 3$. It can be seen that this code is free of any NB-LETSs in $(4, 2)$ and $(6, 2)$ classes. Also, the multiplicity of structures in other non-zero classes is reduced compared to the code of [59].

To investigate the second construction scenario mentioned in Subsection **??**, we consider the problem of finding regular NB-QC-LDPC codes with the minimum lifting degree and free of trapping sets in different ranges of interest for a given base graph sizes 3×5 and 3×6 with algebraic girth at least 8 over GF(4) and primitive element $\alpha^2 + \alpha + 1$. Here, $r_1 : a \leq 6, b \leq 3$, $r_2 : a \leq 8, b \leq 3$, $r_3 : a \leq 10, b \leq 3$, and $r_4 : a \leq 12, b \leq 3$ are considered to be the ranges of interest. The upper bounds on the minimum lifting degrees and the parity check matrices of the constructed codes are reported in Table **??** and **??** for 3×5 and 3×6 base matrices, respectively. We note that the same problem has been addressed for binary QC-LDPC codes in our previous work [62]. By comparing the achievable upper bounds on the minimum lifting degrees in the binary and non-binary codes, one can conclude that introducing the NB structure can improve the error floor of LDPC codes in terms of the distribution of NB-LETSs. In other words, NB-QC-LDPC codes can avoid a certain collection of trapping sets $\mathcal{L}$ for shorter block length compared to binary QC-LDPC codes. For example, it can be observed from Table **??** that the minimum lifting degrees are 14, 18, 23, and 29 for the base matrix of size 3×5 and ranges r_1 to r_4. However, based on Table IV of [62], the minimum lifting degrees for the similar code parameters are 18, 26, 36, and 46 in the binary case. Thus, NB structure of the code can improve

Table 5.6: Upper bounds on the lifting degree of girth-8 NB-QC-LDPC codes with fully-connected 3×5 base graph with no LETSs within different ranges, and the corresponding parity-check matrices

Range	$(a \leq 6, b \leq 3)$	$(a \leq 8, b \leq 3)$
N	14	18
Parity Check Matrix	$\begin{bmatrix} \alpha^0 I^0 & \alpha^2 I^0 & \alpha^2 I^0 & \alpha^1 I^0 & \alpha^0 I^0 \\ \alpha^2 I^0 & \alpha^1 I^5 & \alpha^2 I^4 & \alpha^0 I^8 & \alpha^1 I^9 \\ \alpha^1 I^0 & \alpha^2 I^2 & \alpha^0 I^3 & \alpha^0 I^{12} & \alpha^2 I^{10} \end{bmatrix}$	$\begin{bmatrix} \alpha^1 I^0 & \alpha^1 I^0 & \alpha^0 I^0 & \alpha^2 I^0 & \alpha^0 I^0 \\ \alpha^0 I^0 & \alpha^1 I^{12} & \alpha^1 I^{13} & \alpha^0 I^{10} & \alpha^2 I^{17} \\ \alpha^0 I^0 & \alpha^0 I^4 & \alpha^1 I^8 & \alpha^1 I^3 & \alpha^2 I^{14} \end{bmatrix}$
Range	$(a \leq 10, b \leq 3)$	$(a \leq 12, b \leq 3)$
N	23	29
Parity Check Matrix	$\begin{bmatrix} \alpha^2 I^0 & \alpha^1 I^0 & \alpha^2 I^0 & \alpha^0 I^0 & \alpha^1 I^0 \\ \alpha^0 I^0 & \alpha^2 I^6 & \alpha^0 I^{14} & \alpha^2 I^{15} & \alpha^1 I^{10} \\ \alpha^1 I^0 & \alpha^2 I^{17} & \alpha^0 I^{12} & \alpha^2 I^{19} & \alpha^0 I^{20} \end{bmatrix}$	$\begin{bmatrix} \alpha^2 I^0 & \alpha^1 I^0 & \alpha^2 I^0 & \alpha^0 I^0 & \alpha^1 I^0 \\ \alpha^0 I^0 & \alpha^2 I^{12} & \alpha^0 I^{26} & \alpha^2 I^4 & \alpha^1 I^9 \\ \alpha^1 I^0 & \alpha^2 I^{22} & \alpha^0 I^{11} & \alpha^2 I^2 & \alpha^0 I^{21} \end{bmatrix}$

the error floor of QC-LDPC codes significantly. The same results can be achieved by comparing the minimum lifting degrees of Table **??** and Table V of [62].

As the final example, we consider irregular NB-QC-LDPC codes designed in [55] and [56]. Both codes have block length of $L = 504$ and rate $R = 0.5$ over GF(64). The base matrix size is 4×8, lifting degree is $N = 63$, and the primitive polynomial of the finite field is $\alpha^6 + \alpha + 1$. In [55], the authors first construct a regular NB-QC-LDPC code with variable degree $d_v = 4$, check node degree $d_c = 8$, and algebraic girth of 6 from two arbitrary subsets of GF(q). Based on their method, the values of permutation shift and edge weight for each element of the base matrix is considered to be equal (for example $\alpha^i I^i$). Then, zero elements of the base graph are determined through a masking process such that results in an irregular NB-QC-LDPC code with maximum achievable algebraic girth which is 10 and the multiplicity of shortest non-binary cycles is 3123 based on [55]. The base matrix has 3 columns of weight-2 and 5 columns of weight-3. Obviously, both regular and irregular codes have been designed to improve the girth property of the NB code. The distribution of NB-LETSs for this code is demonstrated in Table **??**. It can be seen that although the designed code has shortest NB cycle of length 10, there are some harmful trapping sets in $(6, 0), (6, 2)$, $(9, 2)$, and $(10, 2)$ classes. The authors of [56], consider a code with similar base and exponent matrices (non-zero elements of the base matrix and the permutation shifts are kept unchanged), and attempt to design the edge weight matrix such that all 3123

Table 5.7: Upper bounds on the lifting degree of girth-8 NB-QC-LDPC codes with fully-connected 3×6 base graph with no LETSs within different ranges, and the corresponding parity-check matrices

Range	$(a \leq 6, b \leq 3)$
N	16
Parity Check Matrix	$\begin{bmatrix} \alpha^2 I^0 & \alpha^2 I^0 & \alpha^1 I^0 & \alpha^0 I^0 & \alpha^0 I^0 & \alpha^1 I^0 \\ \alpha^2 I^0 & \alpha^2 I^{10} & \alpha^0 I^9 & \alpha^0 I^8 & \alpha^1 I^3 & \alpha^1 I^{14} \\ \alpha^1 I^0 & \alpha^2 I^{15} & \alpha^0 I^4 & \alpha^0 I^6 & \alpha^2 I^{12} & \alpha^1 I^7 \end{bmatrix}$
Range	$(a \leq 8, b \leq 3)$
N	22
Parity Check Matrix	$\begin{bmatrix} \alpha^1 I^0 & \alpha^2 I^0 & \alpha^0 I^0 & \alpha^1 I^0 & \alpha^0 I^0 & \alpha^2 I^0 \\ \alpha^0 I^0 & \alpha^2 I^{17} & \alpha^1 I^{15} & \alpha^2 I^6 & \alpha^0 I^{14} & \alpha^1 I^8 \\ \alpha^2 I^0 & \alpha^0 I^{16} & \alpha^0 I^{19} & \alpha^1 I^1 & \alpha^1 I^3 & \alpha^2 I^{11} \end{bmatrix}$
Range	$(a \leq 10, b \leq 3)$
N	28
Parity Check Matrix	$\begin{bmatrix} \alpha^1 I^0 & \alpha^2 I^0 & \alpha^0 I^0 & \alpha^1 I^0 & \alpha^0 I^0 & \alpha^2 I^0 \\ \alpha^2 I^0 & \alpha^1 I^4 & \alpha^1 I^{17} & \alpha^0 I^{18} & \alpha^0 I^{27} & \alpha^2 I^{24} \\ \alpha^2 I^0 & \alpha^1 I^{25} & \alpha^0 I^{22} & \alpha^1 I^7 & \alpha^2 I^{15} & \alpha^0 I^{12} \end{bmatrix}$
Range	$(a \leq 12, b \leq 3)$
N	36
Parity Check Matrix	$\begin{bmatrix} \alpha^1 I^0 & \alpha^0 I^0 & \alpha^2 I^0 & \alpha^2 I^0 & \alpha^1 I^0 & \alpha^0 I^0 \\ \alpha^0 I^0 & \alpha^2 I^{35} & \alpha^1 I^{32} & \alpha^1 I^{17} & \alpha^0 I^{27} & \alpha^2 I^{25} \\ \alpha^2 I^0 & \alpha^1 I^{33} & \alpha^0 I^{22} & \alpha^1 I^{28} & \alpha^2 I^6 & \alpha^0 I^8 \end{bmatrix}$

NB cycles would be eliminated. This goal is achieved by proposing a message passing algorithm to count the multiplicity of NB cycles. Based on their search algorithm, they suggested a construction method aims at removing short NB cycles by changing the edge weights in the Tanner graph of the code. As a result, they could design a code in which all cycles of length up to 10 are canceled which means the algebraic girth is increased to 12. In this regard, they mentioned that the constructed code has 693, 3591, 17892, and 62244 NB cycles of length 12, 14, 16, and 18, respectively. The instances of all NB-LETS structures for this code within the range $a \leq 10$ and $b \leq 4$ have been shown in **??**. It can be seen that the code of [56] was successful to remove NB-LETSs of a large number of classes. However, the constructed code has some instances of NB structures in $(6, 2)$, $(9, 2)$, and $(10, 2)$ classes which can possibly be harmful in the error floor region. Fig. **??** demonstrates the FER performance of codes constructed in [55] and [56]. It can be observed that the code of [56] has better error floor performance compared to the code of [55].

To show the effectiveness of our construction algorithm proposed in section III.C, we design an irregular NB-QC-LDPC code of $R = 1/2$ and the same degree distribution as codes of [55, 56]. Thus, the size of base matrix is supposed to be 4×8 containing 3 and 5 columns of weight 2 and 3, respectively. Our goal is to design the edge weights and circular shifts of each sub-block of the parity-check matrix where the constructed code is free of NB-LETS within the range $a \leq a_{max}$ and $b \leq b_{max}$ such that for $b_{max} = 3$ value, a_{max} is maximized. Note that the search algorithm is devised to design the exponent and edge weight matrices column-by-column. Using the proposed technique, we designed code $\mathcal{C}_{18}$ that is free of NB-LETSs within the range $a \leq 10, b \leq 3$ and has the following parity-check matrix:

$$H_{18} = \begin{bmatrix} \alpha^{32}I^{0} & 0 & \alpha^{35}I^{43} & 0 & \alpha^{42}I^{26} & \alpha^{39}I^{23} & \alpha^{36}I^{30} & \alpha^{50}I^{42} \\ 0 & \alpha^{11}I^{39} & 0 & \alpha^{5}I^{54} & 0 & \alpha^{56}I^{21} & \alpha^{33}I^{42} & \alpha^{8}I^{29} \\ \alpha^{31}I^{0} & 0 & \alpha^{46}I^{1} & \alpha^{19}I^{18} & \alpha^{14}I^{47} & 0 & \alpha^{0}I^{51} & 0 \\ 0 & \alpha^{57}I^{43} & \alpha^{43}I^{52} & \alpha^{40}I^{21} & 0 & \alpha^{28}I^{31} & 0 & \alpha^{61}I^{15} \end{bmatrix}. \tag{5.7}$$

It can be seen that possible harmful structures in $(6, 2)$, $(9, 2)$, and $(10, 2)$ classes have been eliminated from the Tanner graph of the constructed irregular NB code. It is also observed from Fig. **??** that the error floor performance of the designed code is better compared to the codes of [55] and [56] as a result of better NB-LETS

Table 5.8: Multiplicities of LETS structures in the range of $a \leq 10$ and $b \leq 4$ for designed code $\mathcal{C}_{18}$ and codes in [55] and [56]

(a,b) class	Code of [55]	Code of [56]	$\mathcal{C}_{18}$	(a,b) class	Code of [55]	Code of [56]	$\mathcal{C}_{18}$
$(5,2)$	63	0	0	$(8,2)$	630	0	0
$(5,3)$	882	0	0	$(8,3)$	2205	0	0
$(5,4)$	1260	0	0	$(8,4)$	13734	1512	567
$(6,0)$	21	0	0	$(9,2)$	504	126	0
$(6,2)$	126	126	0	$(9,3)$	5985	0	0
$(6,3)$	1092	0	0	$(9,4)$	41454	189	63
$(6,4)$	4032	315	252	$(10,1)$	63	0	0
$(7,2)$	63	0	0	$(10,2)$	1827	504	0
$(7,3)$	882	0	0	$(10,3)$	12789	0	0
$(7,4)$	8064	0	0	$(10,4)$	114282	3465	945

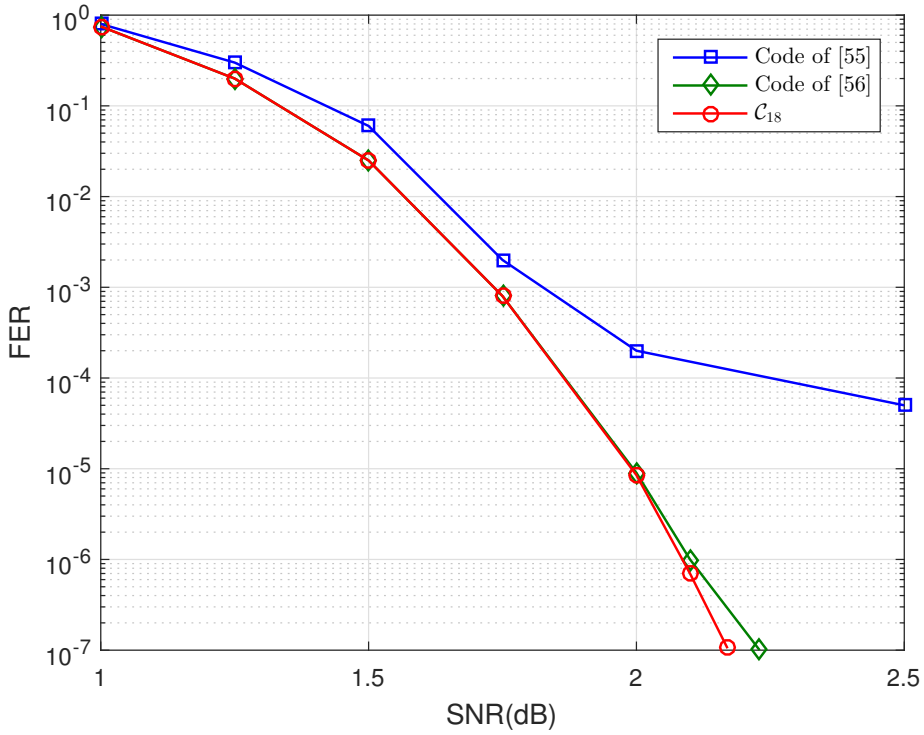


Figure 5.8: FER comparison of proposed code $\mathcal{C}_{18}$ and the NB code designed by technique of [55, 56].

distribution.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we proposed a systematic and efficient method to construct protograph-based binary QC-LDPC codes. We first examined the trapping set structures of such codes, and demonstrated that some of the structures that can exist in a general (randomly constructed) code cannot exist in codes that have QC structure. This was done through the transformation of the problem into a graph coloring problem and was the first step in a series of steps devised to simplify the design of QC-LDPC codes. The next step was to develop an efficient layered *dpl*-based search algorithm for finding a targeted set of trapping sets that are to be avoided in the code (for a good error floor performance). The algorithm was devised as a backward recursion to minimize the number of intermediate structures and the expansions that were needed to search for the targeted trapping sets, as well as to use structures that have a higher chance of having a lower multiplicity in the graph. Numerous codes were constructed using the proposed design technique with superior performance compared to the existing codes in the literature. The systematic approach of the design makes it applicable to codes with different node degrees (rate), girths and block lengths with the flexibility of selecting any set of target trapping sets. The efficiency of the search algorithm makes it possible to design codes with larger degrees and block lengths which are free of trapping sets in larger regions compared to what was achievable before.

Moreover, we designed irregular binary QC-LDPC codes with good waterfall and error floor performance. The good waterfall performance was achieved through the proper design of the base matrix. The low error floor was obtained through the proper selection of cyclic permutation shifts within the exponent matrix to avoid any instance

of LETSs within a targeted set of structures $\mathcal{L}$. At the core of the technique used for the design of the exponent matrix was an efficient search algorithm that was carefully devised to verify whether an irregular Tanner graph has any instance of a LETS within $\mathcal{L}$. While in this work the search algorithm was used within the framework of a greedy column-by-column search for the exponent matrix, it can also be used in various other search strategies that require a fast technique to search for LETSs of an irregular Tanner graph. It was demonstrated through LETS distributions and FER simulations that the designed codes are superior to the existing irregular QC-LDPC codes, particularly in the error floor region.

Furthermore, we studied the characterization and search of trapping sets in NB-LDPC codes and proposed an efficient low-complexity method to find all instances of NB-LETSs within a predetermine range of interest. Finding harmful structures of NB Tanner graphs is beneficial in two ways, (a) evaluating existing codes in the literature in terms of NB-ETS distribution, and (b) design of NB codes with good error floor by removing a collection of harmful structures. In Chapter **??**, to the best of our knowledge, the distribution of trapping sets for a number of existing codes in the literature was reported for the first time. Also, NB-QC-LDPC codes free of a collection of harmful NB trapping sets were constructed and better performance of designed codes in the error floor region were demonstrated through LETS distributions and FER simulations. We also showed that the minimum lifting degrees for construction of QC codes free of a certain collection of trapping sets are lower for NB Tanner graphs compared to the binary ones.

The results of this thesis have been published or submitted for publication in [62], [63], and [79].

6.2 Future Work

Insights gained from this work can enable further advancement of analysis and design of different types of LDPC codes. As an example, the proposed search algorithms of chapter **??**, **??** and **??** can be used in combination with any design technique that involves searching a Tanner graph for certain targeted trapping set structures throughout the construction process. Moreover, the layered characterization/search algorithm of LETSs, proposed for both regular and irregular QC-LDPC codes in this work, can also be used to construct LDPC codes with low error floor that lack

the QC structure. The only difference, compared to what is presented here for QC-LDPC codes, is that for codes lacking the QC structure the number of possible LETS structures is larger. The approached provided here can be used to construct LDPC codes with higher minimum distance and free of small trapping sets at the same time. An another example, the proposed search algorithms can be extended to design codes free of a certain collection of non-elementary trapping sets as well. Moreover, our proposed method can be used along with decoder design techniques to further improve the error floor by joint design of code/decoder.

List of References

[1] Local and Metropolitan Area Networks–Specific Requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11-2012, March 2012.

[2] Air Interface for Broadband Wireless Access Systems, IEEE Std. 802.16-2009, 2009.

[3] Unified High-Speed Wireline-Based Home Networking Transceivers—System Architecture and Physical Layer Specification, ITU-T Std. G.9960 (12/2011), 2011.

[4] Y. Hashemi and A. H. Banihashemi, "New characterization and efficient exhaustive search algorithm for leafless elementary trapping sets of variable-regular LDPC codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 6713-6736, Dec. 2016.

[5] Y. Hashemi and A. H. Banihashemi, "Characterization of elementary trapping sets in irregular LDPC codes and the corresponding efficient exhaustive search algorithms," *IEEE Trans. Inf. Theory*, vol. 64, no. 5, pp. 3411-3430, May. 2018.

[6] T. Richardson, "Error floors of LDPC codes," *in Proc. 41st Annu. Allerton Conf. Commun., Control, Comput.*, pp. 1426-1435, Oct. 2003.

[7] E. M. Kurtas, A. Kuznetsov, and I. Djurdjevic, "System perspectives for the application of structured LDPC codes to data storage devices," *IEEE Trans. Magn.*, vol. 42, no. 2, pp. 200-207, Feb. 2006.

[8] Low-density parity-check codes for use in near-earth and deep space applications, Orange Book, Issue 2, Consultative Committee for Space Data Systems (CCSDS) Experimental Specification 131.1-O-2, Sep. 2007.

[9] D. J. C. Mackay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *IEEE Electron. Lett.*, vol. 32, no. 18, pp. 1645-1648, Aug. 1996.

[10] M. Davey and D. MacKay, ""Low-density parity check codes over GF(q)," *IEEE Commun. Lett.*, vol. 2, no. 6, pp. 165-167, Jun. 1998.

[11] *Consultative Committee for Space Data Systems, Orage Book, Short block length LDPC codes for TC synchronization and channel coding, Experimental Specification*, document 231.1-O-1, Aug. 2015.

[12] A. Hareedy, B. Amiri, R. Galbraith, and L. Dolecek, "Non-binary LDPC codes for magnetic recording channels: Error floor analysis and optimized code design," *IEEE Tran. Commun.*, vol. 64, no. 8, pp. 3194-3207, Aug. 2016.

[13] P. Chen, K. Cai, L. Kong, Z. Chen, and M. Zhang, "Non-binary protograph-based LDPC codes for 2-D ISI magnetic recording channels," *IEEE Tran. Magn.*, vol. 53, no. 11, pp. 1-5, Nov. 2017.

[14] B. Amiri, J. Kliewer, and L. Dolecek, "Analysis and enumeration of absorbing sets for non-binary graph-based codes," *IEEE Tran. Commun.*, vol. 62, no. 2, pp. 398-409, Feb. 2014.

[15] X. Hu, E. Eleftheriou, and D. Arnold, "Regular and irregular progressive edge-growth Tanner Graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386-398, Jan. 2005.

[16] H. Xiao, and A. H. Banihashemi, "Improved progressive-edge-growth (PEG) construction of irregular LDPC codes," *IEEE Commun. Lett.*, vol. 8, no. 12, pp. 715-171, Dec. 2004.

[17] Z. Li, and B. Kumar, "A Class of good quasi-cyclic low-density parity check codes based on progressive edge-growth graph," *in Proc. of Asilomar Conf. Signal, Systems, and Computers*, pp. 1990-1994, Pacific Grove, California, Nov. 2004.

[18] R. Asvadi, A. H. Banihashemi, and M. Ahmadian-Attari, "Lowering the error floor of LDPC codes using cyclic liftings," *IEEE Trans. Inf. Theory*, vol. 57, no. 4, pp. 2213-2224, Apr. 2011.

[19] R. Asvadi, A. H. Banihashemi, and M. Ahmadian-Attari, "Design of finite-length irregular protograph codes with low error floors over the binary-input AWGN channel using cyclic liftings," *IEEE Trans. Commun.*, vol. 60, no. 4, pp. 902-907, April 2012.

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

[21] S. Khazraie, R. Asvadi, and A. H. Banihashemi, "A PEG construction of finite-length LDPC codes with low error floor," *IEEE Commun. Lett.*, vol. 16, no. 8, pp. 1288-1291, Aug. 2012.

[22] J. Wang, L. Dolecek, Z. Zahng, and R. D. Wesel, "The cycle consistency matrix approach to LDPC absorbing sets in separable circulant-based codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2293-2314, Apr. 2013.

[23] M. Diouf, D. Declercq, S. Ouya, and B. Vasic, "A PEG-like LDPC code design avoiding short trapping sets," *in Proc. IEEE Int. Symp. Inf. Theory*, Hong Kong, Jun. 14-19, 2015, pp. 1079-1083.

[24] X. Tao, Y. Li, Y. Liu, and Z. Hu, "On the construction of LDPC codes free of small trapping sets by controlling cycles," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 9-12, Jan. 2018.

[25] F. Amirzade and M. Sadeghi, "Efficient search of QC-LDPC codes with girths 6 and 8 and free of elementary trapping sets with small size," March 2018, available online: https://arxiv.org/abs/1803.08141.

[26] M. Sadeghi and F. Amirzade, "Edge coloring technique to remove small elementary trapping sets from Tanner graph of QC-LDPC codes with column weight 4," August 2018, available online: https://arxiv.org/abs/1808.05258.

[27] S. Naseri and A. H. Banihashemi, "Construction of girth-8 QC-LDPC codes free of small trapping sets," *IEEE Commun. Lett.*, vol. 23, no. 11, pp. 1904-1908, Nov. 2019.

[28] D. Vukobratovic, and V. Senk, "Generalized ACE constrained progressive edge-growth LDPC code design," *IEEE Commun. Lett.*, vol. 12, no. 1, pp. 32-34, Jan. 2008.

[29] X. Zheng, F. C.-M. Lau, and C. K. Tse, "Constructing short-length irregular LDPC codes with low error floor," *IEEE Trans. Commun.*, vol. 58, no. 10, pp. 2823-2834, Oct. 2010.

[30] I. E. Bocharova, B. D. Kudryashov, and R. Johannesson, "Searching for binary and nonbinary block and convolutional LDPC codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 1, pp. 163-183, Jan. 2016.

[31] C. T. Healy, R. C. de Lamare, "Design of LDPC codes based on multipath EMD strategies for progressive edge growth," *IEEE Trans. Commun.*, vol. 64, no. 8, pp. 3208-3219, Aug. 2016.

[32] J. Xu, L. Chen, I. Djurdjevic, and K. Abdel-Ghaffar, "Construction of regular and irregular LDPC codes: Geometry decomposition and masking," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 121-134, Dec. 2006.

[33] G. Han, Y. L. Guan, and L. Kong, "Construction of irregular QC-LDPC codes via masking with ACE optimization," *IEEE Commun. Lett.*, vol. 18, no. 2, pp. 348-351, Feb. 2014.

[34] H. Xu, D. Feng, R. Luo, and B. Bai, "Construction of quasi-cyclic LDPC codes via masking with successive cycle elimination," *IEEE Commun. Lett.*, vol. 20, no. 12, pp .2370-2373, 2016.

[35] X. Liu, F. Xiong, Z. Wang, and S. Liang, "Design of binary LDPC codes with parallel vector message passing," *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1363-1375, Apr. 2018.

[36] T. Tian, C.R. Jones, J.D. Villasenor, and R.D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1242-1247, Aug. 2004.

[37] D. Vukobratovic, and V. Senk, "Evaluation and design of irregular LDPC codes using ACE spectrum," *IEEE Trans. Commun.*, vol. 57, no. 8, pp. 2272-2279, Aug. 2009.

[38] G. Richter, and A. Hof, "On a construction method of irregular LDPC codes without small stopping sets," *IEEE Intrnational Conf. Commun.*, vol. 3, pp. 1119-1124, June 2006.

[39] X. Jiao, J. Mu, J. Song, and L. Zhou, "Eliminating small stopping sets in irregular low-density parity-check codes," *IEEE Commun. Lett.*, vol. 13, no. 6, pp. 435-437, June 2009.

[40] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570-1579, June 2002.

[41] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolic, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.

[42] C. Schlegel and S. Zhang, "On the Dynamics of the Error Floor Behavior in (Regular) LDPC Codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3248–3264, July 2010.

[43] A. Tomasoni, S. Bellini, and M. Ferrari, "Thresholds of Absorbing Sets in Low-Density Parity-Check Codes," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3238–3249, Aug. 2017.

[44] A. Dehghan and A. H. Banihashemi, "Finding leafless elementary trapping sets and elementary absorbing sets of LDPC codes is hard," *in Proc. IEEE Int. Symp. Inf. Theory*, Colorado, Jun. 17-22, 2018, pp. 1645-1649.

[45] D. G. Mitchell, L. Dolecek, and D. J. Costello, "Absorbing set characterization of array-based spatially coupled LDPC codes," *in Proc. IEEE Int. Symp. Inf. Theory*, Hawaii, Jun. 2014, pp. 886-890.

[46] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "Construction of regular QC-LDPC codes with low error floor by efficient systematic elimination of trapping sets," *IEEE Trans. Magn.*, vol. 53, no. 2, pp. 1-11, Sep. 2016.

[47] H. Hatami, D. G. Mitchell, D. J. Costello, and T. Fuja, "Performance bounds for quantized spatially coupled LDPC decoders based on absorbing sets," *in Proc. IEEE Int. Symp. Inf. Theory*, Colorado, Jun. 2018, pp. 826-830.

[48] Y. Zhang and W. E. Ryan, "Toward low LDPC-code floors: a case study," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1566-1573, Jun. 2009.

[49] G. B. Kyung and C. C. Wang, "Finding the exhaustive list of small fully absorbing sets and designing the corresponding low error-floor decoder," *IEEE Trans. Commun.*, vol. 60, no. 6, pp. 1487-1498, Jun. 2012.

[50] M. Ferrari, A. Tomasoni, R. Marenzi, and S. Bellini, "Thresholds of absorbing sets under scaled min-sum LDPC decoding," *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 6643-6651, Oct. 2019.

[51] H. Hatami, D. G. Mitchell, D. J. Costello, and T. Fuja, "A threshold-based min-sum algorithm to lower the error floors of quantized LDPC decoders," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2005-2015, Apr. 2020.

[52] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2, dc) LDPC codes over GF(q) using their binary images," *IEEE Tran. Commun.* , vol. 56, no. 10, pp. 1626-1635, Oct. 2008.

[53] R. H. Peng and R. R. Chen, "Design of nonbinary quasi-cyclic LDPC cycle codes," *IEEE Inf. Theory Workshop*, vol. 7, pp. 13-18, Sep. 2007.

[54] H. Xu, B. Bai, "Superposition Construction of Q-Ary LDPC Codes by Jointly Optimizing Girth and Number of Shortest Cycles," *IEEE Commun. Lett.*, vol. 20, no. 7, pp. 1285-1288, Jul. 2016.

[55] J. Li, K. Liu, S. Lin, K. Abdel-Ghaffar, "A Matrix-Theoretic Approach to the Construction of Non-Binary Quasi-Cyclic LDPC Codes," *IEEE Tran. Commun.*, vol. 63, no. 4, pp. 1057-1068, Apr. 2015.

[56] S. Cho, K. Cheun, K. Yang, "Design of Nonbinary LDPC Codes Based on Message-Passing Algorithms," *IEEE Tran. Commun.*, vol. 66, no. 11, pp. 5028-5040, Nov. 2018.

[57] A. Bazarsky, N. Presman, S. Litsyn, "Design of non-binary quasi-cyclic LDPC codes by ACE optimization," *IEEE Inf. Theory Workshop*, pp. 1-5, Spain, Dec. 2013.

[58] B. Amiri, J, A. F. Castro, L. Dolecek, "Design of non-binary quasi-cyclic LDPC codes by absorbing set removal," *IEEE Inf. Theory Workshop*, pp. 461-465, Nov. 2014.

[59] A. Hareedy, C. Lanka, L. Dolecek, "A General Non-Binary LDPC Code Optimization Framework Suitable for Dense Flash Memory and Magnetic Storage," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 9, pp. 2402-2415, Sep. 2016.

[60] A. Hareedy, C. Lanka, N. Guo, L. Dolecek, "A Combinatorial Methodology for Optimizing Non-Binary Graph-Based Codes: Theoretical Analysis and Applications in Data Storage," *IEEE Trans. Inf. Theory*, vol. 65, no. 4, pp. 2128-2154, Apr. 2019.

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

[62] B. Karimi, and A. H. Banihashemi, "Construction of QC-LDPC codes with low error floor by efficient systematic elimination of trapping sets," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 697-712, Feb. 2020.

[63] B. Karimi, and A. H. Banihashemi, "Construction of irregular protograph-based QC-LDPC Codes with Low Error," *IEEE Tran. Commun.*, DOI:10.1109/TCOMM.2020.3028302, Oct. 2020.

[64] A. Tasdighi, A. H. Banihashemi, and M. R. Sadeghi, "Efficient search of girth-optimal QC-LDPC codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1552-1564, Apr. 2016.

[65] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.

[66] M. Karimi and A. H. Banihashemi, "On characterization of elementary trapping sets of variable-regular LDPC codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 9, pp. 5188-5203, Sep. 2014.

[67] Y. Hashemi and A. H. Banihashemi, "On characterization and efficient exhaustive search of elementary trapping sets of variable-regular LDPC codes," *IEEE Commun. Lett.*, vol. 19, no. 3, pp. 323-326, March 2015.

[68] V. G. Vizing, "On an estimate of the chromatic class of a p-graph (in Russian)," Diskret. Analiz. vol. 3, pp .25-30, 1964.

[69] N. Alon, "A simple algorithm for edge-coloring bipartite multigraphs," *Elsevier Info. Proc. Lett.*, Mar. 2003.

[70] J. Zhao, F. Zarkeshvari, A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Trans. Commun.*, vol. 53, no. 4, pp. 549-554, May 2005.

[71] M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," *in Proc. Int. Symp. Commun. Theory App*, 2001.

[72] M. Diouf, "Advanced design of binary LDPC codes for practical applications," *PhD Thesis*, 2015.

[73] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599-618, Sep. 2001.

[74] G. Liva, and M. Chiani, "Protograph LDPC codes design based on EXIT analysis," *in Proc. IEEE GLOBCOM*, Washington, D.C, Nov. 2007, pp. 3250-3254.

[75] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21-28, 1962.

[76] C. Y. Chen, Q. Huang, C. C. Chao, and S. Lin, "Two low-complexity reliability-based message-passing algorithms for decoding non-binary LDPC Codes," *IEEE Tran. Commun.*, vol. 58, no. 11, pp. 3140-3147, Nov. 2010.

[77] D. Goldin and D. Burshtein, "Iterative linear programming decoding of non-binary LDPC codes with linear complexity," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 282-300, Jan. 2013.

[78] J. D. Horton, "A polynomial-time algorithm to

nd the shortest cycle basis of a graph," *SIAM J. Comput.*, vol. 16, pp. 358-366, 1987.

[79] B. Karimi, and A. H. Banihashemi, "Characterization and search of non-binary LDPC codes and its application on the construction of NB-QC-LDPC codes with low error floor," *Submitted to IEEE Tran. Commun.*, Dec. 2020.

[80] S. Landner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," *in Proc. Int. Conf. Wireless Netw., Commun. Mobile Comput.*, pp. 630-635, Dec. 2005.

[81] J. Kang, L. Zhang, Z. Ding, and S. Lin, "A two-stage iterative decoding of LDPC codes for lowering error floors," *in Proc. IEEE Global Telecommun. Conf.*, pp. 1-4, New Orleans, USA, 2008.

[82] J. Kang, Q. Huang, S. Lin, and K. A. Ghaffar, "An iterative decoding algorithm with backtracking to lower the error-floors of LDPC codes," *IEEE Trans. Commun.*, vol. 59, no. 1, pp. 64-73, Jan. 2011.

[83] S. Beomkyu and P. Hosung, "Multi-stage decoding scheme with postprocessing for LDPC codes to lower the error floor," *IEICE Trans. Commun.*, vol. E94-B, no. 8, pp. 2375–2377, Aug. 2011.

[84] E. Cavus and B. Daneshrad, "A performance improvement and error floor avoidance technique for belief propagation decoding of LDPC codes," *in Proc. 16th IEEE Int. Symp. Pers., Indoor Mobile Radio Commun.*, vol. 59, no. 1, pp. 2386-2390, Los Angeles, CA, USA, Sep. 2005.

[85] Y. Han and W. E. Ryan, "LDPC decoder strategies for achieving low error floors," *in Proc. Inf. Theory Appl. Workshop*, pp. 277-286, San Diego, CA, USA, Jan. 2008.

[86] G. B. Kyung and C.-C. Wang, "Finding the exhaustive list of small fully absorbing sets and designing the corresponding low error-floor decoder," *IEICE Trans. Commun.*, vol. 60, no. 6, pp. 1487-1498, Jun. 2012.

[87] S. Zhang and C. Schlegel, "Controlling the error floor in LDPC decoding," *IEEE Trans. Commun.*, vol. 61, no. 9, pp. 3566–3575, Sep. 2013.

[88] S. Tolouei and A. H. Banihashemi, ""Lowering the error floor of LDPC codes using multi-step quantization," *IEEE Commun. Lett.*, vol. 18, no. 1, pp. 86-89, Jan. 2014.

[89] H. Xiao and A. H. Banihashemi, "Estimation of bit and frame error rates of finite-length low-density parity-check codes on binary symmetric channels," *IEEE Trans. Commun.*, vol. 55, no. 12, pp. 2234 - 2239, Dec. 2007.