Error Floor Analysis of Quasi-Cyclic LDPC and Spatially Coupled-LDPC Codes and Construction of Codes with Low Error Floor

by

Sima Naseri, **M.Sc.**

A dissertation submitted to the
Faculty of Graduate and Postdoctoral Affairs
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

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the dissertation

Error Floor Analysis of Quasi-Cyclic LDPC and Spatially Coupled-LDPC Codes and Construction of Codes with Low Error Floor

submitted by **Sima Naseri**, **M.Sc.**

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Professor Amir H. Banihashemi, Thesis Supervisor

Professor Michael Lentmaier, External Examiner

Professor James Green, Chair,
Department of Systems and Computer Engineering

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

Abstract

Forward error-correcting (FEC) codes play an important role in transmitting data with extremely high reliability through modern communication systems. Many applications such as optical communication channels require a very low error rate when transmitting data. In this thesis, we study the design and analysis of low-density parity-check (LDPC) codes in general and spatially coupled (SC) LDPC codes in particular. At first, we analyze the error floor performance of finite-length protograph-based spatially coupled LDPC codes in terms of their design parameters. We conduct a comprehensive analysis to show that the parameter *syndrome former memory (m)* plays the main role in the average number of cycles and trapping sets in the Tanner graph of finite-length SC-LDPC codes. This, in fact, gives an insight into the error floor performance of protograph-based SC-LDPC codes, and demonstrates the superiority of these codes in the error floor region, compared to their block code counterparts.

To complement the theoretical analysis conducted in the first stage of this research, we develop efficient design techniques for the construction of high-performance quasi-cyclic (QC)-LDPC and SC-LDPC codes, as the second part of the research. Our design approach is basically aimed at improving the performance of finite-length (SC) LDPC codes while maintaining the decoder complexity and latency small. The improvement in error floor is achieved by minimizing (elimination of) the most harmful *trapping set (TS)*s. In this regard, we present two design approaches: i) imposing simple conditions on the small cycles to result in the elimination of specific classes of trapping sets, ii) developing a search-based design technique such that specific trapping sets are targeted for minimization/elimination through the algorithm. Our constructed QC-LDPC and time-invariant SC-LDPC codes are superior to the state-of-the-art both in terms of their error floor performance and their low decoding latency and complexity.

Finally, we look into the design of finite-length time-invariant QC SC-LDPC codes

with a small constraint length and a specific girth. In this respect, different scenarios for the construction process are proposed such that the final QC SC-LDPC code has a specific girth 6 or 8 with a small constraint length. Bounds on memory and lifting degree are derived accordingly to fulfill the girth constraint associated with the specific scenario. Numerical results are provided to compare with the proposed theoretical bounds.

To My Parents

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to my supervisor, Prof. Amir H. Banihashemi, for his invaluable guidance, continuous support and excellent mentorship during the course of my PhD studies. His immense knowledge and supervision always steered me to the right direction during this journey.

Besides my supervisor, I would like to thank my committee members for their helpful comments and feedback on this thesis, and for their generosity in contributing their time to serve on my dissertation defense.

I would like to mention the government of Ontario in acknowledgment of the Ontario Trillium Scholarship which pledged an ample support to my PhD studies. Thanks also goes to our collaborator, Dr. Ali Dehghan, for his constructive comments and the fruitful discussions throughout my research.

I truly owe to my caring parents, Fereshteh and Hossein, and my wonderful brother, Saeed, whose love and prayers have been always with me to keep me motivated and strong through ups and downs. I would not have been where I stand today without their unconditional love, encouragement, patience and endless support. Last but not least, I am thankful to all my friends and relatives here in Ottawa and elsewhere for sharing great moments together.

Table of Contents

List of Tables

List of Figures

Nomenclature

Acronym	Description
AB	Array Based
AS	Absorbing Set
AWGN	Additive White Gaussian Noise
BEC	Binary Erasure Channel
BER	Bit Error Rate
BP	Belief Propagation
BPSK	Binary Phase Shift Keying
BSC	Binary Symmetric Channel
CB	Circulant Based
EAS	Elementary Absorbing Set
FEAS	Fully Elementary Absorbing Set
ETS	Elementary Trapping Set
FEC	Forward Error Correcting
FER	Frame Error Rate
LDPC	Low Density Parity Check
LETS	Leafless Elementary Trapping Set
LSS	Layered SuperSet
LSS-E	Layered SuperSet Expandable

Acronym	Description
LSS-nE	Layered SuperSet Non-Expandable
MAP	Maximum A Posteriori
PEG	Progressive Edge Growth
QC	Quasi-Cyclic
SC-LDPC	Spatially Coupled Low Density Parity Check
SMC	Sequentially Multiplied Columns
SNR	Signal to Noise Ratio
TS	Trapping Set

Chapter 1

Introduction

1.1 Motivation

In 1962, low-density parity-check (LDPC) codes were introduced by Gallager [29] and later on rediscovered by Mackay and Neal in 1996 [56]. These codes are attractive due to their near Shannon limit performance when decoded using message passing algorithms. Message passing decoders are sub-optimal decoders operating on the Tanner graph [79] of LDPC codes to compute the messages at each node and send them through the neighboring edges iteratively. So far, a large number of standards such as IEEE 802.11n (WiFi) [99], IEEE 802.16e (WiMAX) [100], IEEE 802.22 (WRAN) [101], IEEE 802.3an (10GBASE-T Ethernet) [102], DVB-T2 [103] and DVB-S2 [104] have practically adopted LDPC codes. More recently, LDPC codes have been selected for the 5G standard [73], [105]. LDPC codes, compared to the other state-of-the-art coding schemes such as polar codes offer several advantages including the low complexity decoding algorithm and the performance flexibility with respect to different block lengths and code rates. In fact, polar codes, despite the beauty of their construction and their capacity achieving property, do not offer a good performance for large block length applications (such as optical communication systems). Moreover, the decoding algorithm for polar codes is sequential, which makes it inappropriate for high-speed communication systems [76].

In practice, many observations declare that finite-length LDPC codes suffer from the *error floor* phenomenon, which corresponds to a change in the slope of the error rate curve where the quality of channel improves beyond a specific point. This phenomenon mostly affects those applications such as data storage or optical communications where a very low error rate is desired. It is well-known that the error floor

is due to the existence of some graphical structures (error-prone structures) in the Tanner graph of the code, known as *trapping sets* (TS) [72], [69], [52]. As a result, it is crucial to design practical finite-length codes such that they are free of most of these error-prone structures.

Spatially-coupled (SC) LDPC codes [41], as a subset of LDPC codes, are constructed by coupling a number of L disjoint LDPC Tanner graphs. As the process of spatial coupling is equivalent to adding the *memory* to the encoding part, we can refer to SC-LDPC codes as LDPC convolutional codes. SC-LDPC codes have been introduced as a good candidate of error correction codes for optical communications due to their capacity-approaching performance [71], [77], [13]. This property which is known as *threshold saturation* demonstrates that spatially-coupled LDPC codes are capable of improving their *belief propagation* (BP) performance towards the *maximum a posteriori* (MAP) performance of their underlying block LDPC codes in an asymptotic fashion [50], [53]. Nevertheless, in the error floor region, finite-length SC-LDPC codes still confront a performance degradation due to the existence of TSs in their Tanner graph.

One way to characterize trapping sets is through their graphical representation. In this method, each TS is said to belong to the (a, b) class, where a represents the number of variable nodes in this TS, and b denotes the number of its odd-degree (unsatisfied) check nodes. Elementary TSs (ETSs), as a subset of TSs, are said to those TSs whose induced subgraphs contain only degree-1 or degree-2 check nodes, and *leafless ETSs* (LETSs) are ETSs such that every variable node is connected to at least two degree-2 (satisfied) check nodes. In additive white Gaussian noise (AWGN) channels, it is known that the LETSs are the most problematic TSs in the error floor region [45], [32]. In addition, a codeword of weight a belongs to the $(a, 0)$ class of TS, and the weight of the smallest non-zero codeword in the Tanner graph determines the minimum distance d_{min} of the code. Using iterative decoding algorithms, minimum distance still plays an important role in the error floor performance of the codes and is commonly called as undetected error.

As a result of the above discussion, it is crucial to improve the error floor performance either in block LDPC codes or SC-LDPC codes. Generally speaking, reducing the error floor is possible by means of two main approaches: (1) modification on the decoding algorithm to improve the error rate performance [51], [96], [93], [85], and (2) new construction of codes with a set of desired properties. In our work, we focus on

the second approach to design either quasi-cyclic LDPC block codes or spatially coupled LDPC codes with low error floor. In general, increasing the size of shortest cycle in the graph, known as girth (g), in (SC) LDPC codes or removing the dominant TSs in the Tanner graph are identified as two main approaches to improve the error floor performance of the designed codes. Many research work either in the context of LDPC block or SC-LDPC codes are based on the first approach [57], [37], [55], [7], [78], [15]. However, focusing on the girth as the only factor contributing to the error floor performance and attempting to increase it does not necessarily result in an optimized (SC) LDPC code. On the other hand, one needs to simultaneously consider girth and a set of harmful TSs to not only maintain the girth large, but also eliminate those harmful TSs.

With regards to the design of QC-LDPC codes through an investigation over other error-prone structures (rather than only considering girth) there exist some research works including [69], [3], [89], [22], [80], [42]. These works are all focused on the design of *exponent matrix* whose elements are chosen from the set of integers in $\{0, 1, ..., M-1\}$, where M denotes the *lifting degree*. It is known that for the larger block lengths, and subsequently larger values of M, the search space for the design of the exponent matrix becomes too large to handle. In addition to the large search space, the relationship between the exponent matrix and the trapping set distribution of the code, which determines its error-floor performance, is often complicated. As a result, in many publications, the authors consider only a small number of trapping set structures [69], [3], [89], [22], or choose a surrogate graphical structure which is easier to enumerate and examine [80]. One can verify that the second approach is efficient especially for the design of codes with high rates and large block lengths as it significantly reduces the complexity of the search algorithm. In this respect, developing such approaches for the design of LDPC block codes or SC-LDPC codes with low implementation complexity is of great interest in many applications.

Knowledge of the average number of cycles and trapping sets either in LDPC block codes or SC-LDPC codes are of interest due the impact of these structures in the error floor performance. Depending on the channel, the decoding algorithm, and the quantization, different categories of TSs contribute to the transmission errors. For instance, stopping sets are the main reason of error floor for LDPC codes over the binary erasure channel (BEC) under the belief propagation decoding algorithm [21].

For the additive white Gaussian noise (AWGN) channel and the binary symmetric channel (BSC), it is known that the most problematic structures are LETSs for variable-regular LDPC codes [32, 43, 45], and the wider category of ETSs for irregular codes [34]. Absorbing sets [23, 93] are to blame for the error floor for bit-flipping algorithms [93] and quantized decoders over the AWGN channel [23], [86]. For the error floor analysis of LDPC codes based on TSs, see, e.g., [27], [35], [36], and the references therein. Regardless of the category of TSs, the most dominant TSs contain short cycles [20]. In [18] and [20], an asymptotic analysis on the average number of cycles and trapping sets for the LDPC block codes is performed. In particular, the authors in [20] demonstrated that depending on the number of cycles existing in a *local structure*, the average number of that structure tends to infinity (if no cycle is contained in that structure), a positive number (if there exists only one cycle within the structure) or to zero (if more than one cycle are contained in that local structure) when the code's length goes to infinity. Analysis regarding the error floor of SC-LDPC codes (or in general LDPC convolutional codes) have been conducted in some research work including [59], [58], [87]. These analyses mainly involve the asymptotic methods to find bounds for the minimum distance of LDPC convolutional codes. In particular, the work of Mitchell *et al.* in [59] demonstrated that the size of smallest trapping set in a protograph-based LDPC convolutional code grows linearly with the *constraint length* of these ensembles. In addition, [11] showed that the eponymous coupling step results in a reduction in the multiplicity of 6-cycles in a protograph-based SC code. In general, the authors of [24], [62], [10], [26] empirically concluded that increasing memory reduces the multiplicity of small ASs and results in an improved error floor performance. In this regard, it would be beneficial to perform an analysis on the error floor of finite-length protograph-based SC codes to mathematically investigate the impact of memory on the distribution of cycles and harmful trapping sets in protograph-based SC-LDPC codes.

Finite length construction of SC-LDPC codes has been also investigated in several research works including [1], [24], [62], [10], [26], [8], [30], where the authors moved beyond the girth and considered other error-prone structures as well, to design high-performance finite-length SC codes. However, in all of these works, the problem is addressed indirectly through a minimization over the small cycles contributing to those harmful TSs. Said differently, instead of focusing directly on the most harmful TSs in the error floor region, the authors proposed to reduce the multiplicity of small

cycles contributing to those structures. This work could indirectly eliminate some of those harmful TSs, however, it does not necessarily lead to the most optimized code. It is because such target cycles are not necessarily problematic themselves in the error floor region. Instead, the interaction between these cycles that results in generating problematic TS structures needs to be avoided in order to improve the error floor performance in a more efficient way.

In addition to the high performance of the designed codes, another critical factor in channel coding is the complexity and latency of the decoding algorithm, particularly in machine-to-machine applications and optical communications. In this direction, as it has been studied in [7], [78], [6], the parameters *memory* and *constraint length* of SC-LDPC codes play an important role in the complexity associated with the iterative decoders. Furthermore, in the context of QC SC-LDPC codes, we define the constraint length as $\nu_s = (m+1)Mc$, where m, M and c refer to the memory, lifting degree and the number of columns in block code base graph, respectively. Thus, for a fixed value of c, both memory and lifting degree contribute to the constraint length of a QC SC-LDPC code. In this respect, any knowledge of a theoretical bound on either the lifting degree or the memory for QC SC-LDPC codes to achieve a specific girth (larger than 4) can be useful in the design of QC SC-LDPC codes with small constraint length. In fact, most of the works in the literature for the design of SC-LDPC codes do not consider simultaneously both the error floor performance and the decoding complexity in the design of SC-LDPC codes. A few works that considered both, in fact, addressed the performance improvement indirectly by focusing only on the girth. We, however, show that focusing on the girth as the only parameter for performance improvement is not necessarily the best metric of the code's performance in either the waterfall or the error floor region [84, 92]. This is due to the fact that imposing a larger girth value on a Tanner graph creates constraints that would limit the available degrees of freedom for other constraints such as the removal of TSs. In this context, although increasing the girth removes some of the harmful trapping sets indiscriminately, it also removes some that are benign. Our experiments also show that increasing the girth, i.e., eliminating all the cycles of the shortest length, often comes at the expense of increasing the multiplicity of the other larger cycles in the graph. This could worsen the performance of the code if the most harmful structures responsible for decoding failures contain such larger cycles. Therefore, it is important to come up with a design method that aims at improving the error floor performance

(by looking at the harmful TS structures and the girth requirement simultaneously), while the decoding complexity and latency of the designed codes are small. This objective is significantly crucial for the next generation of communication systems.

1.2 Related Works

With regards to the construction of LDPC block codes, one of the main algorithms is known as Progressive-edge-growth (PEG), which is aimed at the design of LDPC codes with a large girth [37]. The authors in [55], later extended this algorithm to the QC structures. In [3], Asvadi *et al.* proposed an algorithm which is based on the design of cyclic liftings to remove some short cycles as part of the problematic trapping sets. In [88], Vasic *et al.* presented the *trapping set ontology*, a database of trapping sets related by their topological structures. Such a method was then used in [69] for the design of structured regular LDPC codes with low error floor for the binary symmetric channel (BSC) such that a specific set of small trapping sets are eliminated from the Tanner graph. In [38], a method was proposed for the construction of LDPC codes with good error floor performance, at the expense of increasing the code length. The cycle consistency approach was introduced in [89] for the design of separable circulant-based LDPC codes free of certain absorbing sets. The work in [46] was also focused on the design of LDPC codes with respect to the PEG construction and with the aim of avoiding harmful trapping sets. Diouf *et al.* in [22] improved the PEG algorithm to design regular LDPC codes with variable node degree 3 and girth 8, free of trapping sets in the $(5, 3)$ class and the minimum number of $(6, 4)$ TSs. Recently, in [80], QC-LDPC codes with variable node degree 3 and girth 8 and free of LETSs within the range $a \leq 8$, $b \leq 3$ are constructed by avoiding certain 8-cycles in the Tanner graph. They also complemented their results by presenting the lower bounds for lifting degree based on the constraints imposed in their approach and thus, some codes were constructed to either achieve or approach this bound. More recently, in [42], QC-LDPC codes were designed, using a systematic scheme to search and eliminate a range of target LETSs. This technique is, in fact, a search-based technique based on the *dpl characterization* of [32], which becomes rather complex for high rate codes with large block lengths. Most recently, in [75], an optimization algorithm based on the simulated annealing algorithm was proposed to construct regular QC-LDPC codes with variable node degree 3 and girth

8 and free of a certain range of LETSs, however, this approach was based on the *LSS characterization* of [45] and thus imposed a huge complexity to search for the desired code.

Asymptotic analysis of spatially coupled codes has been addressed adequately in the literature [71], [50], [53] [54], [90] where the studies showed that SC-LDPC codes offer an excellent capacity-achieving property in the asymptotic regime. Mitchell *et al.* in [61] constructed protograph-based SC-LDPC codes and demonstrated that such codes outperform their block code counterparts asymptotically in both waterfall and error floor region, when the *coupling length* L is sufficiently large. This is due to the two main properties of SC-LDPC codes including capacity approaching behavior under belief propagation algorithm, and the linear growth of minimum distance with block length. Truhachev *et al.* in [87] derived lower bounds on the free distance of periodically time-varying LDPC convolutional codes and also on the minimum distance of tail-biting LDPC convolutional codes. In [59], an analysis over the minimum distance and trapping sets of protograph-based convolutional codes was performed. As a part of work in [59], the authors demonstrated that the size of the smallest non-empty trapping set in protograph-based SC codes grows linearly with the constraint length.

Regarding the finite-length construction of SC codes, there are several research works. A study on absorbing set (AS) properties of array-based (AB) SC-LDPC codes was conducted in [60], where the authors proved that the smallest AS in the AB-SC code Tanner graph has the same size as that of the underlying array-based code, and many ASs are broken by performing the spatial coupling. In [1], a special case of *unwrapping process* based on the *cutting vectors* with memory $m = 1$ was studied and the optimized cutting vector to reduce the number of minimal absorbing sets (including $(3, 3)$ ASs) in column-weight 3 separable-circulant based (SCB) SC-LDPC codes have been presented. Esfahanizadeh *et al.* in [25] followed the cutting vector approach and designed SC-LDPC codes with the smallest number of 8-cycles, optimized with respect to the magnetic recording channels. The same authors in [24] extended the construction method to larger memory ($m = 1, 2$) for SCB-SC codes and proposed the *minimum overlapping (MO)* partitioning technique to conduct the edge spreading process with the aim of reducing the multiplicity of minimal ASs (6-cycles in that work) to be used in additive-white Gaussian noise (AWGN) channels. A construction design approach to improve the girth of SC-LDPC codes is given

in [15, 63], where the authors focused on minimizing the multiplicity of 4-cycles in the base graph using a systematic method, and then a heuristic method to increase the girth in the lifted stage was utilized. However, the proposed algorithm in [15] does not incorporate the special repetitive structure of SC codes and performs an optimization over the whole SC base graph. This results in a large lifting degree required for their designed codes to ensure good performance. Moreover, both works in [15, 63] were only focused on the girth optimization and no study over the TS structures or the latency/complexity criteria has been conducted. Joint optimization of AS spectrum and minimum distance in array-based SC codes using the general edge spreading process for different values of memory has been presented in [62], at which the authors concluded that there exists a trade-off between the multiplicity of 6-cycles and the minimum distance of an AB-SC code. So, by relaxing the number of $(3, 3)$ ASs they could design AB-SC codes with larger minimum distance than their block code counterparts. The authors in [10], proposed an algorithmic optimization to obtain AB-SC codes with a fewer number of 6-cycles. More recently, the authors in [26] proposed a two-stage framework to construct circulant-based (CB) SC codes, where the first stage includes an optimal overlap partitioning to design the SC base graph, and the second step is for choosing the circular permutation shifts in the lifted graph. The main objective in their optimization problem is to achieve the smallest number of 6-cycles in the designed CB SC-LDPC codes. They also showed that this two-step approach results in a better performance compared to the codes designed by the cutting vector or MO partitioning technique. Hareedy *et al.* in [30] proposed a two-step optimization technique to construct CB SC-LDPC codes for partial response (PR) channels. In this regard, the authors developed a design approach with the aim of minimizing the multiplicity of 8-cycles, as they demonstrated that 8-cycles are considered as the common denominator of most harmful structures for CB SC codes over PR channels. Most recently, the work in [8] has been devoted to the construction of QC SC-LDPC codes for a given underlying QC-LDPC block code such that the multiplicity of cycles of specific lengths, contributing to harmful structures in the underlying block code, are minimized. Schmalen *et al.* in [76] introduced a number of design options to construct SC-LDPC codes and assessed how fast they are decodable by optimizing the contributing parameters. Lower bounds on the performance of quantized SC-LDPC decoders either in a flooding schedule or a window decoding fashion were proposed in [35], where the authors demonstrated the superiority of error

floor performance for SC-LDPC codes compared to their block code counterparts. In [9], harmfulness of trapping sets with respect to the windowed decoding was studied, and the authors proposed an edge-spreading algorithm to remove trapping sets from the resultant SC base graph under binary symmetric channel (BSC). This technique, however, does not perform any optimization regarding the constraint length and the designed codes require a large lifting degree to obtain a good performance.

In practice, SC-LDPC codes are encoded using shift registers and are decoded using a sliding window decoder based on message-passing algorithms [40]. In this case, both the encoding and the decoding complexity increase linearly with syndrome former constraint length. The *structural latency*,[1] which is considered in our work, also depends on the constraint length. In [4], a class of LDPC convolutional codes with small constraint length, called as *progressive differences convolutional* (PDC) LDPC codes has been introduced. These codes have rate $R = \frac{c-1}{c}$ (where c is a positive integer to denote the column weight of the corresponding code) and a fixed minimum distance. In this regard, Cho and Schmalen [16], proposed a method to construct the protographs of SC codes with $g = 6$ and rate $R = \frac{c-1}{c}$, where a small constraint length is an objective. An extension of direct design of symbolic matrix for codes with rate $R = \frac{c-\gamma}{c}$ was presented in [5], where c and γ are integers, and $\gamma < c$ corresponds to the row weight of their constructed codes. In [7], Battaglioni *et al.* focused on the problem of minimizing the constraint length for a given degree distribution and girth. In the same paper, lower bounds were derived on the constraint length as a function of girth, and time-invariant SC-LDPC codes have been designed that achieved or approached these bounds. In [78], [6], the same authors proposed a new construction technique for the design of time-invariant SC-LDPC codes to extend their work in [7] to larger girths ($g = 10$ and 12). Thus, they designed such SC codes with the smallest constraint length that could achieve or approach the theoretical lower bounds of memory (and constraint length) corresponding to a specific girth. Although the error floor of the designed codes was a concern in [7] and [78], the authors only addressed it indirectly through the girth. Our examination of the codes designed in [7] and [78] revealed that many of them suffer from rather high error floors due to small TSs or small minimum distance values. Most recently, in [74], Sadeghi and Amirzade introduced sufficient conditions based on edge coloring technique to

[1]The amount of time that the encoder or decoder needs to wait for the input bits before the mapping is performed is referred to as the structural latency, which depends on the structure of the code. This is different than the computational latency from the perspective of hardware design.

design SC-LDPC codes free of a small range of LETSs. However, these codes are not optimized with respect to the constraint length, and their proposed range of elimination is also not large enough to cover the majority of harmful LETSs.

A part of the work in [7] is on the derivation of lower bounds on the syndrome former memory (m) to obtain time-invariant SC-LDPC codes with a specific girth ($g = 6, 8, 10, 12$) and variable node degrees $d_v \geq 3$. In [78], a design technique based on *sequentially multiplied columns (SMCs)* was proposed to approach such lower bounds on m, exclusively for time-invariant SC-LDPC codes with $g = 10, 12$. Since there is no lifting stage in the approach of [7, 78], one can assume that their proposed bounds on m are considered as special case of QC SC-LDPC codes with lifting degree $M = 1$. To the best of our knowledge, there is no comprehensive study on the derivation of bounds for both the lifting degree and memory in QC SC-LDPC codes. For QC-LDPC codes, on the other hand, there are some existing works including [44], [12], and [28], which have been dedicated to the derivation of bounds for the lifting degree to achieve a specific desired girth. In particular, Karimi and Banihashemi in [44] investigated the relationship between short cycles in the base graph giving rise to the creation of cycles in the lifted graph. Based on that, lower bounds for the lifting degree to achieve QC-LDPC codes with $g = 6, 8$ and 10 have been presented thereafter.

All over, there is a need for a comprehensive analysis of the error floor performance of finite-length protograph-based SC-LDPC codes, with respect to the TS structures along with the girth. This mathematically demonstrates some of the observations currently presented in the literature and provides a justification for an improved finite-length performance of protograph-based SC LDPC codes compared to their block code counterparts. Furthermore, the majority of the research on finite-length construction of SC codes only focus on increasing the girth, or they indirectly target 6-cycles or 8-cycles for minimization, while they neglect the effect of harmful TSs and the interaction between the short cycles contributing to these structures. Moreover, in most of the finite-length SC code construction techniques, the importance of small constraint length and its impact on the implementation complexity and latency have not been taken into consideration. Thus, there is still room for improvement in this respect. To complement the design techniques, it is also beneficial to have knowledge of the smallest memory and lifting degree required for the design of QC SC-LDPC

codes with a specific girth. This, particularly, gives insight into the different approaches one may choose to achieve a small constraint length in QC SC-LDPC codes having a specific girth.

1.3 Summary of Contributions

In this thesis, we address the above discussions and challenges, as described in the rest of this chapter. Our main contributions, which are published or (to be) submitted for publication in [64], [65], [66], [67], [68] can be summarized as follows:

- A mathematical analysis of the general edge spreading process is presented in [67]. To this end, we characterize breakage of cycles during the edge spreading process and calculate the probability distribution functions that cycles of different lengths in protograph-based spatially coupled LDPC codes remain unbroken when a general edge spreading process based on a random spreading matrix is performed. We prove that the probability of the existence of cycles in a protograph-based SC code is a monotonically decreasing function in terms of memory (m), i.e., increasing memory reduces the probability of the existence of each cycle in the corresponding SC code. Furthermore, we extend our analysis for the TSs and prove that the probability of the existence of a trapping set not only depends on m, but also depends on the number of fundamental cycles included in that trapping set. So that larger values of m and a larger number of fundamental cycles involved in a trapping set make it more likely to break during a random edge spreading process. Moreover, an enumeration technique for the multiplicity of cycles in protograph-based SC codes is proposed, which is based on the investigation of spreading matrix. In addition to that, we find the expected value for the multiplicity of cycles and trapping sets in the resultant protograph-based SC code when the coupling length L is large enough. Some numerical results which match our theoretical derivations are provided. Our results present a theoretical foundation for the improved performance of SC codes in the error floor region.

- In [64], we devise a new technique for the construction of QC-LDPC codes with girth 8 and variable node degrees 3 and 4, free of small (a, b) LETSs within a specific range. This technique is based on imposing simple constraints

on the cycles of length 8 in the Tanner graph. We also derive lower bounds for the lifting degree to satisfy such constraints. Our constructed codes offer improvement compared to similar work in the literature in the sense that they are free of the same range of LETSs with a smaller block length. Also, this technique compared to the search-based approaches offers a less complex scheme for the design of QC-LDPC codes free of a specific range of LETSs.

- In [66], we design time-invariant SC-LDPC codes with girth 6 and 8 and variable node degrees $3, 4$ and 5, that are free of some harmful LETSs, and have small constraint lengths. Our approach is based on imposing simple constraints on the short cycles in the Tanner graph of the code. We also derive lower bounds on the memory to satisfy some of these proposed constraints in the designed SC codes. Using extensive simulation results, we show that the designed codes have smaller constraint length, superior LETS distribution, and lower error floors compared to some of their counterparts in the literature. In other cases, where the designed codes have similar or larger constraint lengths in comparison with the state-of-the-art codes, we demonstrate that, for the same decoding complexity and latency, the designed codes still have a superior LETS distribution and error floor performance. In particular, we show that our designed codes with the smaller girth 6 could compete with the state-of-the-art with the larger girth 8.

- Our work in [65] is devoted to the design of time-invariant SC-LDPC codes with small constraint length and low error floor, based on our proposed efficient search approach. To this end, we modify the counterpart codes in the literature which are optimized to have a (close to) minimal constraint length for a given degree distribution and girth. The performance improvement is achieved by eliminating (or minimizing the multiplicity of) some of the most harmful TSs in the Tanner graph and/or increasing the minimum distance. Our proposed technique is based on the parent/child relationship between TSs such that we successively minimize their multiplicities in an efficient way. Our constructed codes compared to their counterparts in the literature, with the same degree distribution, the same memory, and the same girth, offer superior performance. This comes along with a low decoding complexity/latency due to the small constraint length in our designed codes.

- In [68], we study the relationships between the girth of the Tanner graph of

a QC SC-LDPC code, the lifting degree, the memory, and the size and the structure of the base graph. We investigate different scenarios including the optimization of lifting degree and memory together or sequentially. For each scenario, we present some upper and lower bounds for the lifting degree and memory to obtain a QC SC-LDPC code with girth 6 or 8. These theoretical bounds give an insight into the ranges of lifting degree and memory one needs to search within if a small constraint length (and accordingly small lifting degree and memory) is an objective.

1.4 Organization of the Thesis

Our thesis is divided into two main parts. At the first part, Chapter 3, we provide analysis on the error floor of protograph-based SC-LDPC codes. The second part, Chapters 4-7, are devoted to the design of high-performance QC-LDPC block codes and SC-LDPC codes. In this regard, the rest of this thesis is organized as follows. Preliminaries including the basic definitions, notations and backgrounds related to our work is provided in Chapter 2. In Chapter 3, we analyze the error floor of protograph-based SC-LDPC codes with respect to a random edge spreading process. Chapter 4, and Chapter 5 are devoted to the introduction of simple conditions (including sufficient conditions, or necessary and sufficient conditions) for the design of QC-LDPC and time-invariant SC-LDPC codes, respectively, where both types of codes are free of small harmful trapping sets and benefit from a small block length or small constraint length. We design time-invariant SC-LDPC codes with small constraint length and low error floor using an efficient search technique in Chapter 6. Derivation of bounds on memory and lifting degree for QC SC-LDPC codes with a specific girth is presented in Chapter 7. Conclusion and future work are discussed in Chapter 8.

Chapter 2

Preliminaries

2.1 Graph Theory

Assume an undirected graph $G = (F, E)$, with the set of nodes $F = \{f_1, ..., f_k\}$ and edges $E = \{e_1, ..., e_m\}$. If an edge e is connected to node f, we say e is *incident* to f. Likewise, if there exists an edge e_k incident to nodes f_i and f_j, these nodes are *adjacent*. The *neighborhood* of a node f, which is denoted by $\mathcal{N}(f)$ is said to the set of nodes adjacent to f. With this regard, the number of edges incident to a node f is known as the *degree* of f, denoted by $deg(f)$. In an undirected graph $G = (F, E)$, every set of nodes and edges such as $f_1, e_1, f_2, e_2, ..., f_k, e_k, f_{k+1}$, where e_i=$f_i f_{i+1}$, makes a *walk* between nodes f_1 and f_{k+1}. Here, the nodes $f_1, f_2, ..., f_{k+1}$ and the edges $e_1, e_2, ..., e_k$ are not necessarily distinct. A special type of walk is *path* at which only the first and last nodes can be the same and others are all distinct. If the first and last nodes are different, this path is an *open path*. Otherwise, it is a closed path known as *cycle*. If there exists an edge incident to two distinct nodes of a cycle, while it is not a part of that cycle, we call such an edge as a *chord*. Therefore, a *simple cycle* is a type of cycle without any chord. Throughout the thesis, we use notation s_k to refer to the simple cycle of length $2k$. The length of the smallest simple cycle in a graph is called *girth* which is denoted by g in this thesis. A path containing m variable nodes is denoted by pa_m. Also, a lollipop walk of length $m + 1$, denoted by lo_m^c, consists of a path of length $m + 1 - c$ followed by a cycle of length c, which is attached to the parent structure through a degree-1 check node. A graph with a *path* between every pair of its nodes is said to be *connected*. A *tree* is a connected graph without any cycle inside it. We can assign one node of a tree as a *root* and call such a tree as *rooted tree*. For each node, we can define a *depth of node*, which is

the length of a path connecting that node to the root. Therefore, *depth of a tree* is the maximum depth of any node inside the tree. *Depth-one tree (dot)* is a tree whose depth is equal to one. A node f with $deg(f) = 1$ is called *leaf*. In this regard, a *leafless graph* is said to a connected graph whose nodes have the minimum degree of 2.

Two graphs G_1 and G_2 represented by their node and edge sets as $G_1 = (F_1, E_1)$ and $G_2 = (F_2, E_2)$ are *isomorphic*, if there exists a bijection $p : F_1 \rightarrow F_2$ such that a pair of nodes $f_1, f_2 \in F_1$ are connected by an edge if and only if $p(f_1)$ and $p(f_2)$ are also connected through an edge. Otherwise, two graphs are *non-isomorphic*.

2.2 LDPC Codes

We call graph $G = (F, E)$ a *bipartite graph* if the set F could be partitioned into two disjoint subsets V and C such that $F = V \cup C$ and $V \cap C = \varnothing$. Any $m \times n$ parity check matrix H of a (m, k) binary LDPC code C is represented by a *bipartite Tanner graph* $G = (V \cup C, E)$, where $V = \{v_1, v_2, ...v_n\}$ and $C = \{c_1, c_2, ..., c_m\}$ denote the set of variable nodes and check nodes of this code, respectively. The degrees of nodes v_i and c_i are denoted by d_{v_i} and d_{c_i}, respectively. In this regard, we call a Tanner graph *variable regular* if all variable nodes in this graph have degree d_v (i.e., $d_{v_i} = d_v, \forall v_i \in V$). A (d_v, d_c)-regular Tanner graph, thus, is a variable-regular graph for which $d_{c_i} = d_c, \forall c_i \in C$. Consider $\mathcal{S} \subset V$, the subset $\Gamma(\mathcal{S})$ of C denotes the set of neighbors of $\mathcal{S}$ in the graph G. The *induced subgraph* of $\mathcal{S}$ in the Tanner graph G which is represented by $G(\mathcal{S})$ is composed of the set of nodes $\mathcal{S} \cup \Gamma(\mathcal{S})$ and the set of edges defined by $\{f_i f_j \in E \text{ s.t. } f_i \in \mathcal{S},\ f_j \in \Gamma\big(\mathcal{S}\big)\}$. We can partition the set of check nodes in $\Gamma(\mathcal{S})$ into two groups of *even-degree* check nodes and *odd-degree* check nodes denoted by $\Gamma_{\mathrm{e}}(\mathcal{S})$ and $\Gamma_{\mathrm{o}}(\mathcal{S})$, respectively. Here, two equivalent definitions *satisfied check nodes* and *unsatisfied check nodes* are used for $\Gamma_{\mathrm{e}}\big(\mathcal{S}\big)$ and $\Gamma_{\mathrm{o}}\big(\mathcal{S}\big)$, respectively. We define the *size* of an induced subgraph $G(\mathcal{S})$ as the number of variable nodes included in this subgraph. In our work, we study the variable-regular graphs with $d_v \geq 3$ and we assume that the Tanner graphs are free of 4-cycles ($g \geq 6$). Also, all the induced subgraphs having the same size a and the same number of unsatisfied check nodes b are assumed to belong to the same (a, b) *class*.

Given a Tanner graph G, a set of variable nodes $\mathcal{S} \subset V$ represents an (a, b) *trapping set (TS)* if $|\mathcal{S}| = a$ and $\Gamma_{\mathrm{o}}(\mathcal{S}) = b$. In this thesis, we use the term "trapping

set" to either refer to the set of variable nodes $\mathcal{S}$, or to the induced subgraph $G(\mathcal{S})$ of $\mathcal{S}$ in the Tanner graph G. Minimum distance of a linear code is known as the weight of its smallest non-zero codeword. We can denote a non-zero codeword of weight a in an LDPC code by an $(a, 0)$ trapping set. An *elementary trapping set (ETS)* is a type of TS with check nodes of degree only one or two. As it was proposed in [32], [45], the ETSs of variable-regular graphs could be represented by a *normal graph*, which is obtained by removing all degree-one check nodes and replacing degree-two check nodes and their incident edges with one edge. Given a graph *G=(F,E)*, an induced subgraph $\mathcal{S}$ is called a *leafless elementary trapping set* if it is an *(a,b) ETS* and its normal graph has no leaf. We call set $\mathcal{S} \subset V$ an (a, b) *absorbing set (AS)* if $\mathcal{S}$ is an (a, b) TS and all the variable nodes in $\mathcal{S}$ are connected to more even-degree check nodes $\Gamma_{\mathrm{e}}(\mathcal{S})$ than odd-degree $\Gamma_{\mathrm{o}}(\mathcal{S})$ ones. In addition, we refer to *elementary absorbing set (EAS)* as a class of absorbing sets, at which every check node in the graph has only degree one or two. In addition, a subgraph $\mathcal{S}$ in a Tanner graph G is said to be an (a, b) *Fully elementary absorbing set (FEAS)*, if (a, b) itself is an EAS and if the variable nodes in $V \backslash \mathcal{S}$ are connected to more check nodes in $C \backslash \Gamma_{\mathrm{o}}(\mathcal{S})$ than in $\Gamma_{\mathrm{o}}(\mathcal{S})$. A *non-elementary trapping set (NETS)* is said to a trapping set which is not elementary (i.e., the check nodes can have degree larger than 2).

2.2.1 QC-LDPC code construction

Consider a graph $G = (F, E)$, one is able to construct $\tilde{G} = (\tilde{F}, \tilde{E})$ by using *graph lifting* operation. For this, we first make M copies of G such that for each node $f \in F$, there exist M copies in $\tilde{F}$ such as $\tilde{f} = \{f^0, \ldots, f^{M-1}\}$. Corresponding to every edge $e = \{v, c\}$ in E, we can apply a circular permutation π^e over the set $\{0, 1, \ldots, M-1\}$ to the M copies of this edge in $\tilde{E}$ such that an edge $\{v^i, c^j\}$ is in the set $\tilde{E}$ if and only if $\pi^e(i) = j$. Then, we represent the set of these edges in $\tilde{E}$ by $\tilde{e} = \{e^0, \ldots, e^{M-1}\}$. The constructed graph $\tilde{G}$ is referred to as a *cyclic M-lifting* of G where M is the *lifting degree*. Also, we call graph G as the *base graph* or equivalently *protograph*.

A (γ, c) base graph (protograph) could be represented by its $\gamma \times c$ bi-adjacency matrix known as *base matrix*. In our work, we denote the base matrix of block codes by B_{block}. Each entry of the base matrix, such as $B_{block}(i, j)$, represents the number of edges connecting check node c_i to variable node v_j in the corresponding block code base graph. Similarly, we denote the parity check matrix of a protograph-based block code by $H_{block}(\gamma, c)$. When the base graph of a block code is fully-connected, it is clear

that γ and c correspond to the column weight and row weight of the block code base matrix, respectively. The parity check matrix of such a block code, by performing graph lifting process, is represented by a $\gamma \times c$ array of $M \times M$ permutation matrices with *row group* index $i \in [1, \gamma]$ and *column group* index $j \in [1, c]$. The following matrix represents such a parity check matrix:

$$H_{block}(\gamma, c) = \begin{bmatrix} I^{p_{1,1}} & I^{p_{1,2}} & \dots & I^{p_{1,c}} \\ I^{p_{2,1}} & I^{p_{2,2}} & \dots & I^{p_{2,c}} \\ \vdots & \vdots & \ddots & \dots \\ I^{p_{\gamma,1}} & I^{p_{\gamma,2}} & \dots & I^{p_{\gamma,c}} \end{bmatrix}. \tag{2.1}$$

Where $p_{i,j}$ values are chosen from the set $\{0, 1, \dots, M-1, \infty\}$. Also, $I^{p_{i,j}}$ denotes an $M \times M$ identity matrix whose rows are shifted by $p_{i,j}$ to the left if $p_{i,j} \neq \infty$. Otherwise, $p_{i,j} = \infty$ implies a missing edge in the base graph G and so, they are replaced by an $M \times M$ all-zero matrix in the parity check matrix of the lifted graph. The combination of all permutation shifts $p_{i,j}$ create the *exponent matrix* $P = [p_{i,j}]$.

It has been demonstrated that for any QC-LDPC code constructed from a fully-connected base graph with no parallel edges, there exists an isomorphic QC-LDPC code whose exponent matrix is defined as [28]:

$$P = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & p_{2,2} & \cdots & p_{2,c} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & p_{\gamma,2} & \cdots & p_{\gamma,c} \end{bmatrix}, \tag{2.2}$$

where, $0 \leq p_{2,2} \leq ... \leq p_{2,c}$ [81].

Array-based (AB)-LDPC codes [23], as a subset of quasi-cyclic LDPC codes, are denoted as (γ, p)-AB code, such that $\gamma \leq p$, the lifting degree $M = p$, and p is an odd prime. The parity check matrix for a (γ, p)-AB block code is composed of a $\gamma \times p$

array of $p \times p$ matrices as follows:

$$H_{block}^{AB}(\gamma, p) = \begin{bmatrix} I & I & I & \dots & I \\ I & I^1 & I^2 & \dots & I^{p-1} \\ I & I^2 & I^4 & \dots & I^{2(p-1)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ I & I^{\gamma-1} & I^{(\gamma-1)2} & \dots & I^{(\gamma-1)(p-1)} \end{bmatrix}. \tag{2.3}$$

It is known that the cycles in the lifted graph come from some structures in the base graph known as *tailless backtrackless closed (TBC) walk*s. In the following lemma, we clarify the relationship between TBC walks of a specific length in the base graph and the cycles with the same length in lifted graph.

Lemma 1. *[44] Suppose that we have a tailless backtrackless closed walk W with length l in the base graph of an LDPC code. Corresponding to this walk there exists a sequence of edges $e_1, e_2, ..., e_l$. Associated with these edges, we have a sequence of permutation shifts $p_1, p_2, ..., p_l$, corresponding to a cyclic lifting of degree M. We define the permutation shift of W as*

$$p_W \stackrel{M}{=} \sum_{i=0}^{l-1} (-1)^i p_{i+1}. \tag{2.4}$$

We use the notation " $\stackrel{M}{=}$ " to denote modulo M operation. According to (2.4), the inverse image of W is a set of l-cycles in the lifted graph if and only if

$$p_W = 0, \tag{2.5}$$

and W does not contain any TBC walk of length smaller than l whose permutation shift is zero.

So, the TBC walks and their permutation shifts play an important role in the existence of cycles in the lifted graph.

2.3 SC-LDPC Codes

2.3.1 SC-LDPC codes constructed from protographs

Graph representation

Suppose that we are given the base graph of a protograph-based regular (γ, c) LDPC block code, where $d_v = \gamma = 3$ and $d_c = c = 6$, as illustrated in Fig. 2.1 (a). Consider the parameter L to be the *coupling length* of the corresponding SC protograph, which represents the number of disjoint copies of the unwrapped block code protograph. First we need to copy the block code protograph L times at time instants $0, 1, ..., L-1$ so that each variable node v_i is connected to a check node c_j through an edge at the same time instant. Now, using an *edge spreading* technique [61], each variable node at time t could be connected to check nodes at time instants $t, t+1, ..., t+m$, where m denotes the *coupling width* or *memory* of the corresponding SC code. By this work, the base graph for the protograph-based SC-LDPC code is generated, which is depicted in Fig. 2.1 (c). Choosing a *lifting degree* M and applying the *graph lifting* operation [17] (through circular shifts) to this protograph, one can create the Tanner graph of the corresponding QC SC-LDPC code.

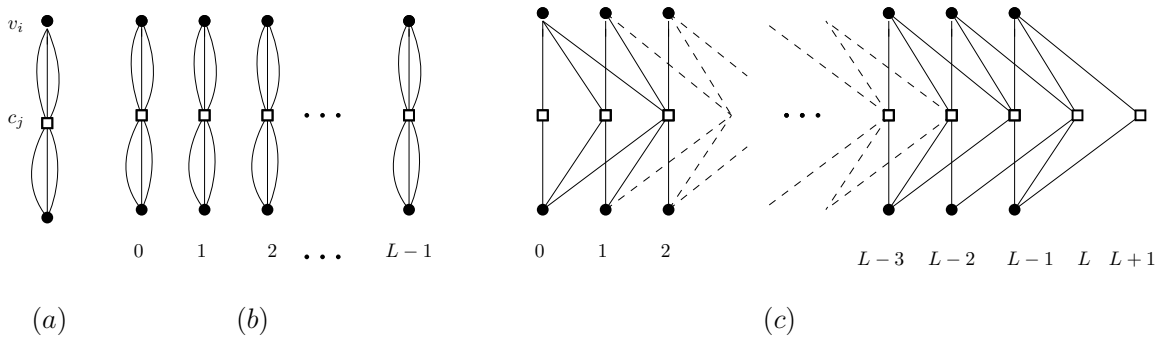


Figure 2.1: (a) Protograph representation of a $(3, 6)$-regular LDPC block code, (b) sequence of L protographs, and (c) $(3, 6)$ spatially-coupled protograph illustration, where $m = 2$.

Matrix representation

As it has been mentioned in [41] and [17], by using the *unwrapping* technique on the parity check matrix H_{block} of any block code we can create the spatially-coupled

$$H_{block}(\gamma, c) = \begin{bmatrix} I^{p_{1,1}} & I^{p_{1,2}} & I^{p_{1,3}} & I^{p_{1,4}} & I^{p_{1,5}} \\ I^{p_{2,1}} & I^{p_{2,2}} & I^{p_{2,3}} & I^{p_{2,4}} & I^{p_{2,5}} \\ I^{p_{3,1}} & I^{p_{3,2}} & I^{p_{3,3}} & I^{p_{3,4}} & I^{p_{3,5}} \end{bmatrix},$$

$$H_0 = \begin{bmatrix} I^{p_{1,1}} & I^{p_{1,2}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ I^{p_{2,1}} & I^{p_{2,2}} & I^{p_{2,3}} & \mathbf{0} & \mathbf{0} \\ I^{p_{3,1}} & I^{p_{3,2}} & I^{p_{3,3}} & I^{p_{3,4}} & I^{p_{3,5}} \end{bmatrix},$$

$$H_1 = \begin{bmatrix} \mathbf{0} & \mathbf{0} & I^{p_{1,3}} & I^{p_{1,4}} & I^{p_{1,5}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I^{p_{2,4}} & I^{p_{2,5}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$H_{SC}(\gamma, c, L, \boldsymbol{\xi}) = \begin{bmatrix} I^{p_{1,1}} & I^{p_{1,2}} & & & & & & & & \\ I^{p_{2,1}} & I^{p_{2,2}} & I^{p_{2,3}} & & & & & & & \\ I^{p_{3,1}} & I^{p_{3,2}} & I^{p_{3,3}} & I^{p_{3,4}} & I^{p_{3,5}} & & & & & \\ & & I^{p_{1,3}} & I^{p_{1,4}} & I^{p_{1,5}} & I^{p_{1,1}} & I^{p_{1,2}} & & & \\ & & & I^{p_{2,4}} & I^{p_{2,5}} & I^{p_{2,1}} & I^{p_{2,2}} & I^{p_{2,3}} & & \\ & & & & & I^{p_{3,1}} & I^{p_{3,2}} & I^{p_{3,3}} & I^{p_{3,4}} & I^{p_{3,5}} \\ & & & & & & & I^{p_{1,3}} & I^{p_{1,4}} & I^{p_{1,5}} \\ & & & & & & & & I^{p_{2,4}} & I^{p_{2,5}} \\ \underbrace{\qquad\qquad}_{L=1} & & & & & \underbrace{\qquad\qquad\qquad\qquad}_{L=2} & & & & \end{bmatrix}.$$

Figure 2.2: The parity check matrix of a QC block code, $H_{block}(\gamma, c)$, and its corresponding SC code $H_{SC}(\gamma, c, L,\ \xi)$, using cutting vector $\boldsymbol{\xi} = [2, 3, 5]$ and $L = 2$. Here, $I^{p_{i,j}}$ denotes a permutation matrix at row group i and column group j, and $\mathbf{0}$ entries correspond to all-zero matrices. For simplicity, the all-zero matrix entries of H_{SC} is not shown in the figure.

version of that code. In particular, suppose that we have a (γ, c) protograph-based LDPC code, lifted by lifting degree M. In this case, unwrapping technique could be characterized by a *cutting vector* $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_\gamma]$, such that $0 \leq \xi_1 < \xi_2 < \ldots < \xi_\gamma \leq c$, and it is assumed that ξ_i parameters, $i \in \{1, 2, \ldots, \gamma\}$ are distinct. By this process, we are able to partition H into two different matrices H_0 and H_1 of size $\gamma M \times cM$. Where, H_0 is formed by assigning circulant matrices in row group $i \in [1, \gamma]$ and column group j, $j \leq \xi_i$, from $H_{block}(\gamma, c)$ to the equivalent position in H_0. Rest of the positions in H_0 are then replaced by zero. Consequently, H_1 is defined to be $H_1 = H_{block}(\gamma, c) - H_0$. In this regard, Fig. 2.2 illustrates the unwrapping procedure, starting from a $(3, 5)$-regular protograph-based LDPC code, using the cutting vector $\boldsymbol{\xi} = [2, 3, 5]$.

In fact, unwrapping procedure based on cutting vectors could be generalized to the *edge spreading process* based on a *spreading matrix* B with $m \geq 1$. In this regard, we can denote the parity check matrix of a protograph-based SC-LDPC code by $H_{SC}(\gamma, c, m, L, B)$ as follows

$$H_{SC}(\gamma, c, m, L, B) = \underbrace{\begin{bmatrix} H_0 & \mathbf{0} & \ddots & \mathbf{0} \\ H_1 & H_0 & \ddots & \mathbf{0} \\ H_2 & H_1 & \ddots & \mathbf{0} \\ \vdots & H_2 & \ddots & \mathbf{0} \\ H_m & \vdots & \vdots & \vdots \\ \mathbf{0} & H_m & \ddots & H_0 \\ \mathbf{0} & \mathbf{0} & \ddots & H_1 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ddots & H_m \end{bmatrix}}_{L}. \tag{2.6}$$

In this notation, m refers to the memory of SC code, L is the coupling length, H_i's are the components, $\mathbf{0}$ denote the all-zero matrices, and B corresponds to the spreading matrix. In the followings, we define these parameters.

Definition 1. *[24] Consider the parity check matrix of a protograph-based SC code*

(H_{SC}) with coupling length L and memory m whose corresponding block code is lifted from a $\gamma \times c$ base graph with lifting degree M. The replica, Rep_i, $i \in \{1, \ldots, L\}$ is defined as

$$Rep_i = H_{SC}[1 : (L+m)\gamma M][(i-1)Mc + 1 : iMc]. \tag{2.7}$$

In fact, Rep_i in our definition, corresponds to a set of circulant matrices (including all zero matrices) in H_{SC} whose rows and columns indices belong to the set $\{1, \ldots, (L+m)\gamma M\}$ and $\{(i-1)Mc+1, \ldots, iMc\}$, respectively, thus all the rows are involved in all replicas.

Furthermore, component H_i, $i \in \{0, 1, \ldots, m\}$ is defined as $H_i = H_{SC}[i\gamma M + 1 : (i+1)\gamma M][1 : Mc]$.

Next, we define the notion of *spreading matrix (B)* corresponding to the general edge spreading process.

Definition 2. Spreading matrix (B). *[62] In the process of converting $H_{block}(\gamma, c)$ to $H_{SC}(\gamma, c, m, L, B)$, a spreading matrix B of size $\gamma \times c$ determines how to spread the edges of the block code protograph. As a result of this process, $m+1$ components H_0, H_1, ..., H_m is generated. In other words, $B_{i,j} = t$ $(0 \leq t \leq m)$[1] implies that the $M \times M$ permutation matrix in row group i and column group j of $H_{block}(\gamma, c)$ is copied into the equivalent position in H_t.*

Remark 1. *General edge spreading process is also applicable to a base graph which is not fully-connected. In this case, the entries of B are chosen from $[-1, m]$ so that every edge with a "zero" entry in the block code base matrix is assigned by "-1" in the corresponding spreading matrix B.*

Example 1. *Consider a protograph-based LDPC block code with the following 3×5 base matrix, B_{block}, and its corresponding parity check matrix, $H_{block}(3, 5)$, using lifting degree M*

$$B_{block} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}, \quad H_{block}(3,5) = \begin{bmatrix} \mathbf{0} & I^{p_{1,2}} & I^{p_{1,3}} & I^{p_{1,4}} & \mathbf{0} \\ I^{p_{2,1}} & I^{p_{2,2}} & \mathbf{0} & I^{p_{2,4}} & I^{p_{2,5}} \\ I^{p_{3,1}} & \mathbf{0} & I^{p_{3,3}} & I^{p_{3,4}} & I^{p_{3,5}} \end{bmatrix}. \tag{2.8}$$

[1] Every entry of B matrix at the i-th row and j-th column is referred to as $B_{i,j}$.

In this notation, $I^{p_{i,j}}$ and $\mathbf{0}$ are both $M \times M$ matrices, where the former denotes an identity matrix shifted by $p_{i,j}$ to the left, and the latter refers to an all-zero matrix. As is evident from (2.8), the block code base graph is not fully-connected. Now, suppose that the general edge spreading process with $m = 2$ using the following spreading matrix is applied on this block code.

$$B = \begin{bmatrix} -1 & 1 & 2 & 2 & -1 \\ 1 & 0 & -1 & 1 & 2 \\ 0 & -1 & 1 & 0 & 2 \end{bmatrix}, \tag{2.9}$$

where, the non-zero entries of B_{block} are assigned to a value $\in [0, m]$ in the spreading matrix and the zero entries of B_{block} have -1 value in the spreading matrix. As a result of this edge spreading process we will have $m + 1 = 3$ components in the protograph (and also Tanner graph) of SC code. Considering $L = 3$, the base matrix of the resultant SC code (B_{SC}) is

$$B_{SC} = \begin{bmatrix} \boxed{\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{matrix}} & & & \\ \boxed{\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{matrix}} & \boxed{\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{matrix}} & & \\ \boxed{\begin{matrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}} & \boxed{\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{matrix}} & \boxed{\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{matrix}} & \\ & \boxed{\begin{matrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}} & \boxed{\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{matrix}} & \\ & & \boxed{\begin{matrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}} & \end{bmatrix}.$$

And, three components H_0, H_1, H_2 *are*

$$H_0 = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I^{p_{2,2}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ I^{p_{3,1}} & \mathbf{0} & \mathbf{0} & I^{p_{3,4}} & \mathbf{0} \end{bmatrix}, \quad H_1 = \begin{bmatrix} \mathbf{0} & I^{p_{1,2}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ I^{p_{2,1}} & \mathbf{0} & \mathbf{0} & I^{p_{2,4}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I^{p_{3,3}} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

$$H_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} & I^{p_{1,3}} & I^{p_{1,4}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & I^{p_{2,5}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & I^{p_{3,5}} \end{bmatrix}.$$

Finally, the parity check matrix $H_{SC}(3, 5, 2, L, B)$ of the resultant SC code is created by putting together H_0, H_1, H_2 in the form of (2.6).

2.3.2 Time-invariant SC-LDPC convolutional codes

A time-invariant SC-LDPC covolutional code is characterized by the following semi-infinite parity check matrix:

$$H_{SC} = \begin{bmatrix} H_0 & \mathbf{0} & \mathbf{0} & \ddots \\ H_1 & H_0 & \mathbf{0} & \ddots \\ H_2 & H_1 & H_0 & \ddots \\ \vdots & H_2 & H_1 & \ddots \\ H_m & \vdots & H_2 & \ddots \\ \mathbf{0} & H_m & \vdots & \ddots \\ \mathbf{0} & \mathbf{0} & H_m & \ddots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{2.10}$$

In this notation, each block $H_i, i = 0, 1, ..., m$, is assumed to be a binary $\gamma \times c$ matrix, and m defines the (syndrome former) memory of the code. Considering a coupling length L, one can terminate the matrix in (2.10) and create the parity check

matrix of the terminated time-invariant SC-LDPC code with a structure similar to (2.6). Correspondingly, the (syndrome former) constraint length for this code is defined as $\nu_s = (m+1)c$.

An alternate representation for matrices $H_i, i = 0, 1, ..., m$, is through a $\gamma \times c$ *symbolic matrix*

$$H(x) = \begin{bmatrix} h_{1,1}(x) & h_{1,2}(x) & \cdots & h_{1,c}(x) \\ h_{2,1}(x) & h_{2,2}(x) & \cdots & h_{2,c}(x) \\ \vdots & \vdots & \ddots & \vdots \\ h_{\gamma,1}(x) & h_{\gamma,2}(x) & \cdots & h_{\gamma,c}(x) \end{bmatrix}, \tag{2.11}$$

where $h_{i,j}(x), i \in \{1, ..., \gamma\}, j \in \{1, ..., c\}$, are from the ring of polynomials whose coefficients are in the Galois field $GF[2]$. The connection between the two representations is through

$$h_{i,j}(x) = \sum_{k=0}^{m} h_k^{(i,j)} x^k, \tag{2.12}$$

where $h_k^{(i,j)}$ denotes the (i,j)-th entry of matrix H_k in (2.10). The highest weight of polynomial entries in $H(x)$ defines the type of the code.

In our work, we only consider Type-1 codes, for which the entries of $H(x)$ are either monomials or zero. For Type-1 codes, the $\gamma \times c$ *exponent matrix* $P = [p_{ij}]$ is defined by $p_{ij} = log_x(h_{i,j}(x))$, if $h_{i,j}(x) \neq 0$, and $p_{ij} = -1$ if $h_{i,j}(x) = 0$. So, if $p_{ij} \neq -1$, then $0 \leq p_{ij} \leq m$.

In fact, time-invariant LDPC-convolutional codes are not quasi-cyclic and their construction is performed by directly designing the exponent matrix P. This is in contrast with the other construction approach based on the edge spreading process, that starts from a block code. Without loss of generality, throughout our thesis, we only consider the terminated (finite-length) SC codes, and simply use the terminology *time-invariant SC-LDPC* code. So, the main difference between time-invariant SC-LDPC codes and the other type of SC codes introduced in Section 2.3.1 is that the former are not QC, but the latter are QC SC-LDPC codes.

2.4 Decoding of SC-LDPC Codes

In order to decode SC-LDPC codes, one may still use the conventional flooding scheme based on the belief propagation (BP) algorithm [49]. However, due to the large block lengths of such codes, this technique is not efficient in terms of the complexity and latency, and thus a new decoding scheme has been proposed for the SC-LDPC codes.

Consider a protograph-based SC-LDPC code with parity check matrix $H_{SC}(\gamma, c, m, L, B)$ as described in (2.6). According to the special convolutional structure of this parity check matrix, one can see that two variable nodes of the protograph that are at least $(m+1)c$ columns apart from each other do not share any common check node, and so they are not involved in the same parity check equation. Using this property, an efficient decoding scheme known as *sliding window decoding* [39], [40] has been developed for the decoding of SC-LDPC codes. Consider Fig. 2.3, where we illustrate the windowed decoding on a protograph-based SC-LDPC code with $m = 2$, $L = 8$ decoded by a sliding window decoder using a window of size $W = 3$. We refer to the set of edges included in the window at a specific decoding time instant as the *window configuration*. For the sliding window decoder shown in Fig. 2.3, the window configuration at the first time instant consists of $W\gamma M$ rows (in bits) and WcM columns (in bits) of the parity check matrix H_{SC}, where M is the lifting degree of the corresponding SC code. Thus, at first time instant, the decoder performs belief prorogation over the edges within the window to decode the first cM symbols known as *targeted symbols*. After performing the maximum number of iterations, the window slides down γM rows and right cM columns to continue the decoding process at the new time instant. In the terminated portion of the SC code, the window configuration has fewer edges than other configurations within the code. The decoder depicted in Fig. 2.3 is currently at the fourth decoding time instant, which means that the window of size $W = 3$ starting from the top left corner of H_{SC} has already moved down by $3\gamma M$ rows and moved right by $3cM$ columns. In this figure, one can see that the edges highlighted in blue from the previous window, are connected to the current targeted symbols. Also, those symbols that have been processed so far and have no connection with the current window are identified by a red arrow. After L time instants, the sliding window reaches the end of the matrix and the entire codeword is recovered.

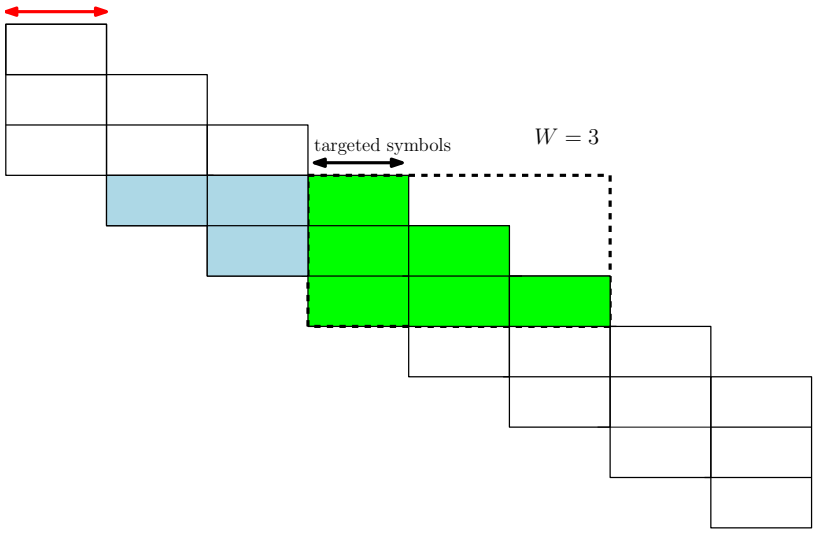


Figure 2.3: Windowed decoding illustration on a protograph-based SC-LDPC code with parity check matrix $H_{SC}(\gamma, c, 2, 8, B)$. This window configuration consists of $W\gamma M = 3\gamma M$ rows of the parity check matrix and all the $(W + m)cM = 5cM$ columns are currently involved. This includes the edges highlighted in blue and green.

Chapter 3

Analysis of Error Floor in Finite-length Protograph-based SC-LDPC Codes

3.1 Introduction

It is known that cycles and small trapping sets play a vital role in the error floor region of LDPC codes including SC-LDPC codes. In this chapter, we perform a mathematical analysis of the general edge spreading process to find the distribution of cycles and small TSs in a protograph-based SC-LDPC code. In general, having an arbitrary protograph-based LDPC block code (based on a cyclic lifting or a random lifting) with a given distribution of cycles and TSs, one can analyze the distribution of cycles and TSs in the corresponding protograph-based SC-LDPC code, after applying a random edge spreading process. In this chapter, to be more specific, we limit our analysis to the protograph-based SC-LDPC codes lifted based on a cyclic lifting. This helps us analyze the distribution of cycles and TSs in protograph-based SC-LDPC codes by starting from the block code base graph (according to the discussions in Section 2.3.1). This, in fact, simplifies the analysis and makes it applicable to a wide range of quasi-cyclic protograph-based SC-LDPC codes. In this regard, and motivated by [18], [20], [11] and the empirical results in the literature regarding the cycle/TS distribution in SC codes, we conduct an analysis to look into the effect of *memory (m)* on the multiplicity of cycles of different lengths and harmful TSs in finite-length SC codes. Thus, in Section 3.2, we consider a general edge spreading process and derive probability distribution function for the existence of cycles of different lengths in the resultant SC code when the spreading matrix is random. We prove that the probability distribution function is monotonically decreasing as m increases.

Therefore, increasing memory reduces the average multiplicity of cycles in an SC code. Furthermore, in Section 3.4, we present an analysis over the TS distribution in protograph-based SC codes and we prove that the number of (fundamental) cycles involved in a TS also impacts the multiplicity of TSs in SC codes. In this regard, we demonstrate that the probability of the existence a TS in an SC code is monotonically decreasing with m^k, where k refers to the number of fundamental cycles included in a TS. Thus, the larger number of cycles included in a TS, and/or a larger value of m is translated to a higher probability of breakage for such a TS during the edge spreading process. As an example, we calculate the probability of the existence of small leafless elementary trapping set (LETS)s in the $(4, 2)$ class (in a variable-regular graph with $d_v = 3$ and $g = 6$) to clearly demonstrate the role of m on the probability distribution function for the existence of this class of LETS during the edge spreading process. This trapping set is, in fact, known as one of the dominant trapping sets in the error floor region for variable-regular (SC) LDPC codes over the additive white Gaussian noise (AWGN) channel [26]. As a complementary study for our analysis, we look at the problem from the deterministic point of view and thus, for a given spreading matrix, we provide a theoretical approach to find the exact multiplicity of cycles of different lengths in terms of the coupling length L in a finite-length protograph-based SC code. This technique, which is presented in Section 3.3, is an alternative approach for the search-based technique and only inspects the spreading matrix to conclude a closed form for the multiplicity of cycles in the protograph-based SC codes. Finally, in Section 3.5, we provide some numerical results to first evaluate our theoretical analysis and then validate the general conclusion by making some comparisons with the actual SC codes. Our results present a theoretical foundation for the improved performance of protograph-based SC codes (compared to their underlying block codes) in the error floor region and some empirical results reported in the literature are demonstrated as a result of our analysis.

3.2 Probability of the Existence of Cycles in Protograph-based SC Codes

3.2.1 Primary results

Let B be a $\gamma \times c$ spreading matrix. For every two distinct columns $j_1, j_2 \in [1, c]$, and row $i \in [1, \gamma]$ such that $B_{i,j_1}, B_{i,j_2} \geq 0$, the *differential parameter* is defined as $\Delta B_i(j_1, j_2) = B_{i,j_1} - B_{i,j_2}$. Similarly, corresponding to two rows i_1, i_2 and column j where $B_{i_1,j}, B_{i_2,j} \geq 0$ we can also define a *vertical differential parameter* as $\Delta B^v_j(i_1, i_2) = B_{i_1,j} - B_{i_2,j}$. By *vertical* we mean that the two consecutive edges share a variable node in column j. These two definitions are equivalent and thus, throughout this chapter we mainly focus on the first definition. The other definition *vertical differential parameter* will be used occasionally.

Next, we introduce the necessary and sufficient conditions for a TBC walk in the base graph of the block code to remain a TBC walk in the base graph of the corresponding SC code, after performing the edge spreading process.[1] It should be noticed that our work in this chapter is applicable to block code base graphs regardless of being fully-connected or not. According to Remark 1, we can define the spreading matrix for base graphs that are not fully-connected.

Lemma 2. *A sequence of variable nodes and check nodes such as* $v_{j_1}, c_{i_1}, v_{j_2}, c_{i_2}, \cdots, v_{j_{\ell/2}}, c_{i_{\ell/2}}, v_{j_1}$, *corresponding to a TBC walk of length* ℓ *in the block code base graph will remain as a TBC walk of the same length in the base graph of resultant SC code, if and only if the coupling length* L *is large enough and*

$$\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_2, j_3) + \ldots + \Delta B_{i_{\ell/2}}(j_{\ell/2}, j_1) = 0, \tag{3.1}$$

where B *is the spreading matrix used for edge spreading process,* $i_1 \neq i_2$, $i_2 \neq i_3, \ldots, i_{\ell/2} \neq i_1$ *and* $j_1 \neq j_2$, $j_2 \neq j_3$, $\ldots$, $j_{\ell/2} \neq j_1$.

We note that the similar result as Lemma 2 is presented in [28] and [7] in the context of QC lifting of block codes, and edge spreading of time-invariant SC-LDPC convolutional codes, respectively.

[1] It should be noted that we only focus on single-edge protograph-based codes so that no parallel edge is assumed to be in the base graph of our SC codes and the base matrix is, in fact, a binary matrix.

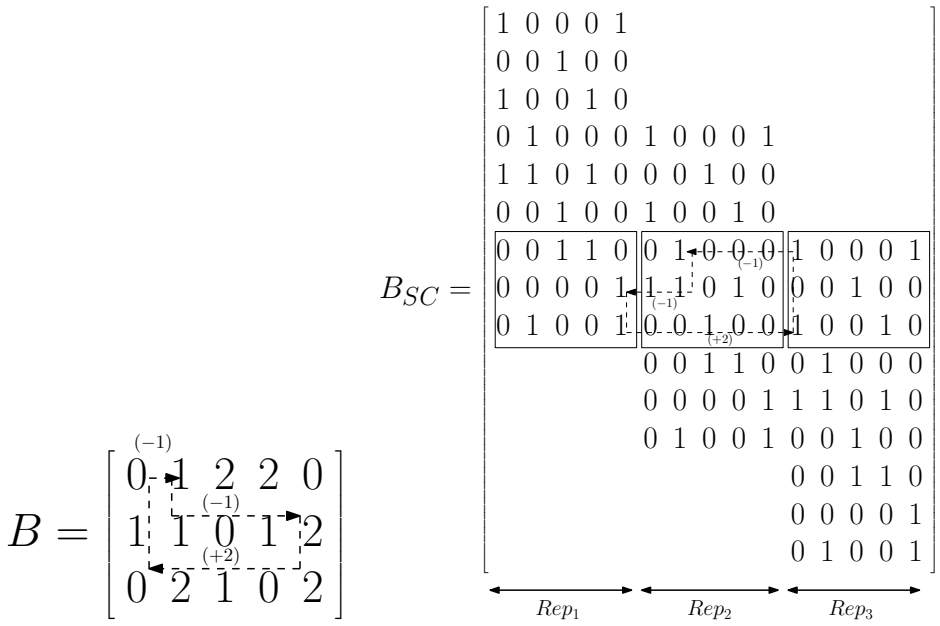


Figure 3.1: Representation of a TBC walk of length 6 in the spreading matrix B and the base matrix of its corresponding SC code with $\gamma = 3, c = 5, m = 2$ and $L = 3$. Numerical values of $\Delta B_i(j_1, j_2)$ for every pair of variable nodes v_{j_1}, v_{j_2} sharing a check node c_i are shown below the arrows.

Example 2. *Consider the* 3×5 *spreading matrix*

$$B = \begin{bmatrix} 0 & 1 & 2 & 2 & 0 \\ 1 & 1 & 0 & 1 & 2 \\ 0 & 2 & 1 & 0 & 2 \end{bmatrix}. \tag{3.2}$$

Using Lemma 2, a TBC walk of length 6 *in the block code base graph consisting of a sequence of variable nodes and check nodes* $v_1, c_1, v_2, c_2, v_5, c_3, v_1$ *remains a TBC walk of length* 6 *in the base graph of the corresponding SC code because*

$$B_{1,1} - B_{1,2} + B_{2,2} - B_{2,5} + B_{3,5} - B_{3,1} = 0 - 1 + 1 - 2 + 2 - 0 = 0. \tag{3.3}$$

Fig. 3.1 illustrates how this TBC walk is mapped to the base matrix of the corresponding SC code, when the spreading matrix in (3.2) is used for the edge spreading process.

In light of Lemma 1 and Lemma 2, we have the following terminology.

Definition 3. *A TBC walk of length ℓ in the block code base graph satisfying Lemma 1 (i.e., for which we have Equation (2.5) of Lemma 1 satisfied), and according to Lemma 2 does not break during the edge spreading process, is called as an ℓ-block cycle, or equivalently a block cycle of length ℓ.*

It is well known that in the construction of SC codes, the coupling length L is set to be large enough to reduce the rate-loss [61]. Thus, having L sufficiently large, for each block cycle of length ℓ, there may be multiple cycles in the SC code Tanner graph. In the following lemma, we discuss the minimum value of L to guarantee that every ℓ-block cycle has at least one copy in the SC code base graph.

Lemma 3. *[24] All variable nodes of a cycle of length ℓ in an SC protograph span at most $\lfloor \frac{\ell}{4} \rfloor m + 1$ consecutive replicas. It implies that to cover all variable nodes of a cycle of length ℓ, the coupling length of SC code must be at least $L = \lfloor \frac{\ell}{4} \rfloor m + 1$.*

In the following, we present some necessary lemmas that are required for the calculation of probability distribution functions throughout the next sections.

Lemma 4. *Let B be a spreading matrix and its non-negative entries (i.e., $B_{i,j} \neq -1$) be discrete random variables with independent and identical uniform distribution from the interval $[0, m]$. For every row i and every pair of columns (j_1, j_2) in B, we have*

$$\Pr\left(\Delta B_i(j_1, j_2) = z\right) = \frac{(m+1) - |z|}{(m+1)^2} \qquad \forall z \in [-m, m], z \in \mathbb{Z}, \tag{3.4}$$

where, $j_1 \neq j_2$, $\mathbb{Z}$ denotes the set of integer numbers, and m is the memory of corresponding protograph-based SC code.

Proof. Every discrete random variable $B_{i,j} \in \{0, 1, \ldots, m\}$ is an i.i.d. uniform random variable, so

$$\Pr\left(B_{i,j} = x\right) = \frac{1}{m+1}, \quad \forall x \in [0, m]. \tag{3.5}$$

Also, we have $\Delta B_i(j_1, j_2) = B_{i,j_1} - B_{i,j_2} = z \in [-m, m]$. We can consider three cases for the value of z as follows:
Case 1. If $z = 0$. Then $B_{i,j_1} = B_{i,j_2}$. On the other hand, $B_{i,j_1} \in [0, m]$, Thus, by

(3.5), we have

$$\begin{aligned}\Pr\left(\Delta B_i(j_1,j_2)=z\right) &= \Pr\left(B_{i,j_1}-B_{i,j_2}=0\right)\\ &= \sum_{k=0}^{m}\Pr\left(B_{i,j_2}=k\right)\Pr\left(B_{i,j_1}=k\right)=\sum_{k=0}^{m}\frac{1}{(m+1)^2}=\frac{m+1}{(m+1)^2}.\end{aligned} \tag{3.6}$$

Case 2. If $z>0$. Then $B_{i,j_1}=t$ and $B_{i,j_2}=t-z$, where $t\in[z,m]$. Thus,

$$\begin{aligned}\Pr\left(\Delta B_i(j_1,j_2)=z\right) &= \Pr\left(B_{i,j_1}-B_{i,j_2}=z\right)\\ &= \sum_{t=z}^{m}\Pr\left(B_{i,j_2}=t\right)\Pr\left(B_{i,j_1}=t-z\right)=\frac{m+1-|z|}{(m+1)^2}.\end{aligned} \tag{3.7}$$

Case 3. If $z<0$. Then $B_{i,j_1}=t-z$ and $B_{i,j_2}=t$, where $t\in[z,m]$. Thus, similar to Case 2, we have $\Pr\left(\Delta B_i(j_1,j_2)=z\right)=\frac{m+1-|z|}{(m+1)^2}$. This completes the proof. ■

Lemma 5. *Let B be a spreading matrix and its non-negative entries (i.e. $B_{i,j}\neq -1$) be discrete random variables with independent and identical uniform distribution from the interval $[0,m]$. Assuming $S=(B_{i_1,j_1}+B_{i_2,j_2})$, where $i_1\neq i_2$ or $j_1\neq j_2$, this random variable has the following probability mass function (pmf)*

$$\Pr\left(S=s\right)=\begin{cases}\frac{s+1}{(m+1)^2} & ,s\in[0,m]\\ \frac{2m-s+1}{(m+1)^2} & ,s\in[m+1,2m]\\ 0 & ,\textit{otherwise}\end{cases} \tag{3.8}$$

Proof. Random variable S is the summation of two independent random variables B_{i_1,j_1} and B_{i_2,j_2} where each of them has the following distribution

$$\Pr\left(B_{i_1,j_1}=x\right)=\frac{1}{m+1}\quad \forall\, x\in[0,m], x\in\mathbb{Z}. \tag{3.9}$$

Since two random variables B_{i_1,j_1} and B_{i_2,j_2} are independent, the probability mass function for S is obtained by performing convolution of the probability mass functions for B_{i_1,j_1} and B_{i_2,j_2} given in (3.9), and the proof is complete. ■

Lemma 6. *Let B be a spreading matrix and its entries that are not "-1", i.e., $B_{i,j}\neq -1$, be random variables with independent and identical uniform distribution*

from the interval $[0, m]$. *Suppose that* $\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2') = k \in \mathbb{Z}$, *where* $i_1 \neq i_2$, $j_1 \neq j_2$, $j_1' \neq j_2'$ *and* $k \in [-2m, 2m]$. *Depending on the value of* k *we have*

$$\Pr\left(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2') = k\right) =$$

$$\begin{cases} \frac{1}{(m+1)^4}\left[k^3/6 + k^2 m + k^2 + 2km^2 + 4km + 11\frac{k}{6} + 4\frac{m^3}{3} + 4m^2 + 11\frac{m}{3} + 1\right] & , -2m \leq k < -m \\ \frac{1}{6(m+1)^4}\left[-3k^3 - 6k^2 m - 6k^2 + 3k + 4m^3 + 12m^2 + 14m + 6\right] & -m \leq k < 0 \\ \frac{1}{6(m+1)^4}\left[3k^3 - 6k^2 m - 6k^2 - 3k + 4m^3 + 12m^2 + 14m + 6\right] & 0 \leq k < m \\ \frac{1}{(m+1)^4}\left[-k^3/6 + k^2 m + k^2 - 2km^2 - 4km - 11\frac{k}{6} + 4\frac{m^3}{3} + 4m^2 + 11\frac{m}{3} + 1\right] & m \leq k \leq 2m \end{cases} \tag{3.10}$$

Proof. The proof is given in Appendix A.2. ■

Lemma 7. *Let* B *be a* $\gamma \times c$ *spreading matrix and its non-negative entries* ($B_{i,j} \neq -1$) *be discrete random variables with independent and identically uniform distribution from the interval* $[0, m]$. *For any arbitrary integer value of* $l \geq 1$ *and* $z \in \mathbb{Z}$ *such that* $-lm \leq z \leq lm$, *we have* $\Pr\left(\sum_{t=1}^{l} \Delta B_{i_t}(j_t, j_t') = z\right) = \mathcal{O}(\frac{1}{m})$,[2] *where* $i_t \in [1, \gamma]$, $j_t \neq j_t'$ *and* $j_t, j_t' \in [1, c]$.

Proof. The non-negative entries of the B matrix are discrete random variables with independent and identically uniform distribution from the interval $[0, m]$. By having this property, we have the following:

$$\begin{aligned} &\Pr\left(\sum_{t=1}^{l} \Delta B_{i_t}(j_t, j_t') = z\right) = \Pr\left(\sum_{t=1}^{l-1} \Delta B_{i_t}(j_t, j_t') + \Delta B_{i_l}(j_l, j_l') = z\right) \\ &= \sum_{u=-(l-1)m}^{(l-1)m} \left(\Pr\left(\Delta B_{i_l}(j_l, j_l') = z - u\right) \Pr\left(\sum_{t=1}^{l-1} \Delta B_{i_t}(j_t, j_t') = u | \Delta B_{i_l}(j_l, j_l') = z - u\right)\right) \end{aligned} \tag{3.11}$$

By Lemma 4, we have $\Pr\left(\Delta B_{i_l}(j_l, j_l') = x\right) = \mathcal{O}(\frac{1}{m})$. Also, we have

[2] Suppose two functions $f(x)$ and $g(x)$ which are defined on some subsets of the real numbers. We can say $f(x) = \mathcal{O}(g(x))$ if and only if there exist a constant x_0 and a positive constant number a, such that $|f(x)| \leq a\,|g(x)|$ for all $x > x_0$. Intuitively, this implies that f does not grow faster than g.

$\sum_{u=-(l-1)m}^{(l-1)m} \Big(\Pr\big(\sum_{t=1}^{l-1} \Delta B_{i_t}(j_t, j'_t) = u | \Delta B_{i_l}(j_l, j'_l) = z - u \big) \Big) = 1$. Thus, by (3.11), we have

$$\begin{aligned}
&\Pr\Big(\sum_{t=1}^{l} \Delta B_{i_t}(j_t, j'_t) = z \Big) \\
&= \sum_{u=-(l-1)m}^{(l-1)m} \Big(\Pr\big(\sum_{t=1}^{l-1} \Delta B_{i_t}(j_t, j'_t) = u | \Delta B_{i_l}(j_l, j'_l) = z - u \big) \mathcal{O}(\frac{1}{m}) \Big) \\
&= \mathcal{O}(\frac{1}{m}) \sum_{u=-(l-1)m}^{(l-1)m} \Big(\Pr\big(\sum_{t=1}^{l-1} \Delta B_{i_t}(j_t, j'_t) = u | \Delta B_{i_l}(j_l, j'_l) = z - u \big) \Big) \\
&= \mathcal{O}(\frac{1}{m}). \qquad (3.12)
\end{aligned}$$

■

3.2.2 Probability of the existence of cycles in SC codes

Next, we calculate the probability of the existence of cycles in SC codes during the edge spreading process. For the case of 4-cycles (and 6-cycles) we note that a TBC walk of length 4 (and 6) is the same as a cycle of length 4 (and 6).

Theorem 1. *Let G be a base graph of a block code and suppose that we have a cycle $\mathcal{C}_4$ of length 4 in that base graph. If we consider memory m, coupling length $L \geq m+1$ and apply the edge spreading process using a spreading matrix B whose entries, which are not "-1",[3] are assigned independently and identically with a uniform distribution from the interval $[0, m]$, the probability that the cycle does not break during the edge spreading process and remains in the SC base graph is*

$$\Pr\big(\textit{The 4-cycle} \mathcal{C}_4 \textit{ remains in SC protograph} \big) = \frac{2m^2 + 4m + 3}{3(m+1)^3}. \qquad (3.13)$$

Proof. Consider the 4-cycle $\mathcal{C}_4$. Corresponding to this cycle we have two random variables $\Delta B_{i_1}(j_1, j_2)$ and $\Delta B_{i_2}(j_2, j_1)$. We have $L \geq m + 1$, thus, by Lemma 2 and

[3]Referring to Remark 1, the general edge spreading process is also applicable to the base graphs that are not fully-connected. For every missing edge in the base graph (where the base graph is not fully-connected), we can assign an entry " -1" to the corresponding spreading matrix B. Thus, we only consider non-negative entries of B matrix as the independent and identically distributed random variables from $[0, m]$.

Lemma 3, the cycle $\mathcal{C}_4$ exists in the base graph of the resultant SC code if and only if $\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_2, j_1) = 0$. Thus, the probability of existence of $\mathcal{C}_4$ in the SC protograph is equal to

$$\Pr\big(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_2, j_1) = 0\big). \tag{3.14}$$

By using Lemma 6, we can calculate (3.14) (we need to consider the case $k = 0$).

$$\Pr\big(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_2, j_1) = 0\big) = \frac{4m^3 + 12m^2 + 14m + 6}{6(m+1)^4} = \frac{2m^2 + 4m + 3}{3(m+1)^3}. \tag{3.15}$$

This completes the proof. ■

Next, we calculate the probability of existence of 6-cycles in the protograph of SC codes, in terms of m. We note that in [11, Theorem 3.4.1], the authors used complicated calculus for the calculation of the probability of breakage of 6-cycles in the protograph of an SC code. We, however, present an straightforward approach, which is in fact consistent with the results in [11].

Theorem 2. *Let G be a base graph of a block code and suppose that we have a TBC walk $\mathcal{C}_6$ of length 6 in that base graph, which is in fact a cycle of length 6. If we consider memory m, coupling length $L \geq m+1$ and apply the edge spreading process using a spreading matrix B whose entries, which are not "-1" are assigned independently and identically with a uniform distribution in $[0, m]$, the cycle $\mathcal{C}_6$ remains in the base graph of the resultant SC code with the following probability:*

$$\Pr\big(\textit{The 6-cycle}\,\mathcal{C}_6\,\textit{remains in SC protograph}\big) = \frac{11m^4 + 44m^3 + 71m^2 + 54m + 20}{20(m+1)^5}. \tag{3.16}$$

Proof. Consider the 6-cycle $\mathcal{C}_6$. Corresponding to this cycle we have three random variables $\Delta B_{i_1}(j_1, j_2)$, $\Delta B_{i_2}(j_2, j_3)$ and $\Delta B_{i_3}(j_3, j_1)$. We have $L \geq m+1$, thus, by Lemma 2 and Lemma 3, the cycle $\mathcal{C}_6$ exists in the base graph of the resultant SC code if and only if $\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1) = 0$. Thus, the probability of existence of $\mathcal{C}_6$ in the SC protograph is equal to

$$\begin{aligned}&\Pr\big(\text{6-cycle}\,C_6\,\text{remains in SC protograph}\big)\\ &= \Pr\big(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1) = 0\big).\end{aligned} \tag{3.17}$$

We note that $\Delta B_{i_1}(j_1, j_2) \in [-m, m]$, so the right-side of (3.17) is written as

$$\begin{aligned}&\Pr\left(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1) = 0\right)\\ &= \sum_{y=-m}^{m} \Pr\left(\Delta B_{i_1}(j_1, j_2) = -y\right) \Pr\left(\Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1)) = y\right). \end{aligned} \tag{3.18}$$

By Lemma 4, and (3.18) we have

$$\begin{aligned}&\Pr\left(\text{6-cycle } C_6 \text{ remains in SC protograph}\right)\\ &= \sum_{y=-m}^{m} \frac{(m+1) - |(-y)|}{(m+1)^2} \Pr\left(\Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1)) = y\right)\\ &= \sum_{y=-m}^{-1} \frac{(m+1) + y}{(m+1)^2} \Pr\left(\Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1)) = y\right)\\ &+ \sum_{y=0}^{m} \frac{(m+1) - y}{(m+1)^2} \Pr\left(\Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1)) = y\right) \end{aligned} \tag{3.19}$$

Next, we use Lemma 6, to calculate $\Pr\left(\Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1)) = y\right)$. We have

$$\sum_{y=-m}^{-1} \Pr\left(\Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1)) = y\right) = \frac{1}{6(m+1)^4} \times \frac{m}{4}(11m^3 + 34m^2 + 37m + 14). \tag{3.20}$$

Also,

$$\begin{aligned}&\sum_{y=-m}^{-1} y \Pr\left(\Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1)) = y\right)\\ &= \frac{1}{6(m+1)^4} \times \frac{-m}{10}(11m^4 + 50m^3 + 85m^2 + 70m + 24). \end{aligned} \tag{3.21}$$

Similarly, we have

$$\sum_{y=0}^{m} \Pr\left(\Delta B_{i_2}(j_2, j_3) + \Delta B_{i_3}(j_3, j_1)) = y\right) = \frac{1}{6(m+1)^4} \times \left(\frac{11m^4}{4} + \frac{25m^3}{2} + \frac{85m^2}{4} + \frac{35m}{2} + 6\right). \tag{3.22}$$

Furthermore,

$$\sum_{y=0}^{m} y \Pr\left(\Delta B_{i_2}(j_2,j_3)+\Delta B_{i_3}(j_3,j_1))=y\right) = \frac{1}{6(m+1)^4}\times\frac{m}{10}\left(11m^4+50m^3+85m^2+70m+24\right). \tag{3.23}$$

By substituting (3.20), (3.21), (3.22) and (3.23) in (3.19), we have

$$\Pr\left(\text{The 6-cycle } C_6 \text{ exists.}\right) = \frac{11m^4+44m^3+71m^2+54m+20}{20(m+1)^5}. \tag{3.24}$$

■

Next, we focus on the TBC walks of length 8 and discuss their existence during a random edge spreading process. In Fig. 3.2, we illustrate all different structures for such TBC walks in the base graph of the block code. In this figure, variable and check nodes are shown by circles and squares, and denoted by u and w, respectively. The variable and check node indices correspond to different column and row blocks in the parity check matrix of an LDPC code, respectively. It should be noticed that for the shorter cycles (i.e., cycles of length 4 and 6) the TBC walks are in fact the cycles in the base graph, so we just referred to them as the cycles.

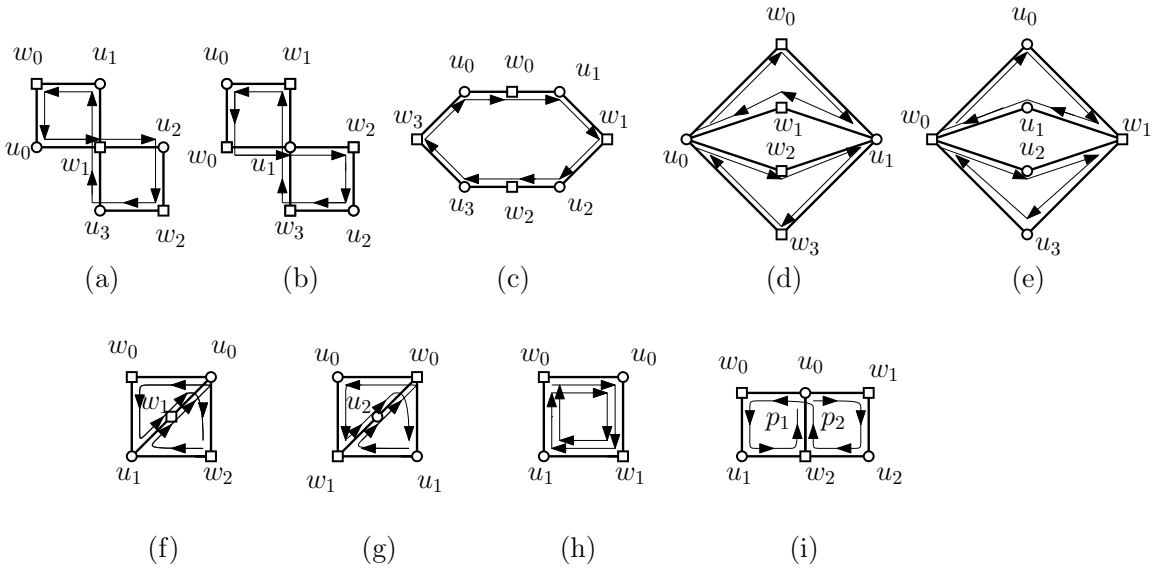


Figure 3.2: Structures in the base graph that can be mapped to 8-cycles in the lifted graph with their corresponding TBC walks.

Theorem 3. *Let G be a base graph of a block code and suppose that we have a TBC walk $\mathcal{C}_8$ of length 8 in that base graph whose structure is illustrated in Fig. 3.2. If we consider memory m, coupling length $L \geq 2m+1$ and apply the edge spreading process using a random spreading matrix B whose entries that are not "-1" are assigned independently and identically with a uniform distribution in $[0, m]$, the TBC walk $\mathcal{C}_8$ remains in the base graph of the resultant SC code with the following probabilities depending on the structure of this TBC walk:*

$$\Pr(\textit{The TBC walk } C_8 \textit{ remains in SC protograph}) = \begin{cases} \frac{151m^6+906m^5+2335m^4+3300m^3+2734m^2+1284m+315}{315(m+1)^7} & \mathcal{C}_8 \in \{\textit{Fig. 3.2(a)-(e)}\} \\ \frac{23m^4+92m^3+148m^2+112m+45}{60(m+1)^5} & \textit{odd } m,\ \mathcal{C}_8 \in \{\textit{Fig. 3.2(f)-(g)}\} \\ \frac{23m^4+92m^3+148m^2+112m+60}{60(m+1)^5} & \textit{even } m,\ \mathcal{C}_8 \in \{\textit{Fig. 3.2(f)-(g)}\} \\ \frac{2m^2+4m+3}{3(m+1)^3} & \mathcal{C}_8 \in \{\textit{Fig. 3.2(h)}\} \\ \frac{302m^4+1208m^3+1937m^2+1458m+495}{720(m+1)^5} & \textit{odd } m,\ \mathcal{C}_8 \in \{\textit{Fig. 3.2(i)}\} \\ \frac{302m^6+1812m^5+4655m^4+6540m^3+5348m^2+2448m+720}{720(m+1)^7} & \textit{even } m,\ \mathcal{C}_8 \in \{\textit{Fig. 3.2(i)}\} \end{cases} \tag{3.25}$$

Proof. The proof is given in Appendix B. ■

In Fig. 3.3 we illustrate how the probability of the existence of different TBC walks is changing by varying the values of m. From this figure, one can verify that the TBC walk in Fig. 3.2(h) has the largest probability of existence, while the structures in Fig. 3.2(f), (g) have the lowest probability. As a result, the probability that a cycle of length 8 in the block code remains in the corresponding SC code is bounded by the probability of these two types of TBC walks of length 8. It should be noted that the overall probability of the existence of 8-cycles in the corresponding protograph-based SC code would in fact depend on how many TBC walks of length 8 in a given block code base graph belong to each category of structures in Fig. 3.2.

Next, we consider the probability of existence of general TBC walks of length ℓ.

Theorem 4. *Let G be a base graph of a block code and suppose that we have a TBC*

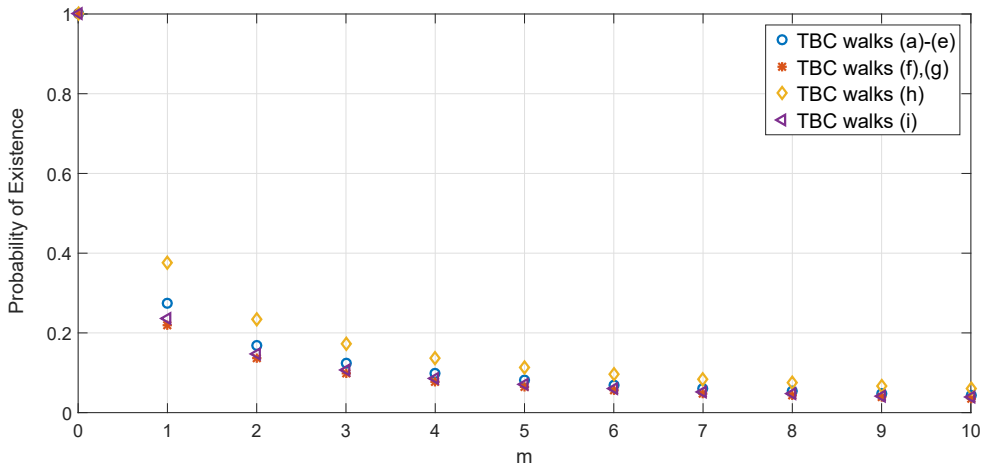


Figure 3.3: Impact of memory m on the probability of existence of TBC walks of length 8 in the protograph of an SC code.

walk $\mathcal{C}_\ell$ of length ℓ in that base graph. If we consider memory m, coupling length $L \geq \lfloor \frac{\ell}{4} \rfloor m + 1$ and apply the edge spreading process using a spreading matrix B whose entries (that are not "-1") are assigned independently and identically with a uniform distribution in $[0, m]$, the TBC walk $\mathcal{C}_\ell$ remains as a TBC walk of the same length in the base graph of the resultant SC code with the probability $\mathcal{O}(\frac{1}{m})$.

Proof. By using Lemma 7 for the case where $z = 0$ and $l = \frac{\ell}{2}$, we have the proof. ■

It should be noted that even though Theorem 4 gives an upper bound for the probability of existence of a cycle of length ℓ, in Theorems 1-3 we also derived exact probability distribution functions for the existence of cycles of length $4, 6$ and 8 such that those results are all consistent with the above upper bound.

Remark 2. *Theorem 4 implies that the probability of the existence of any TBC walk in the block code base graph is reduced monotonically by increasing memory (m). In other words, we know that the probability of existence of TBC walks in the SC code base graphs is smaller than (or equal to) one, and referring to Theorem 4 it is evident that this function is varying in the form of $\frac{1}{m}$ as m goes to infinity. Therefore, as m goes to infinity, the probability that a specific TBC walk is broken during the edge spreading process approaches* 1.

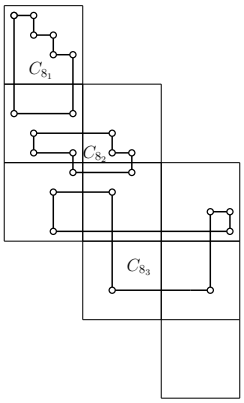


Figure 3.4: Examples of 8-cycles spanning the parity check matrix of an SC code with $m = 2$ and $L = 3$.

3.3 Enumeration of Cycles in Protograph-based SC Codes

3.3.1 Exact number of ℓ-cycles in protograph-based SC codes

It is known that the final multiplicity of cycles of a specific length in the SC code depends on the coupling length L. In the following Theorem, we present a closed-form for the multiplicity of cycles in a protograph-based SC code as a function of the spreading matrix B and the coupling length L. Referring to Definition 3, given a spreading matrix along with a set of ℓ-block cycles and the coupling length L, one can find the multiplicity of ℓ-cycles in the corresponding SC code Tanner graph by only investigating the spreading matrix.

Example 3. *See Fig. 3.4 at which three different 8-cycles spanning through the parity check matrix of an SC code with $m = 2$ is illustrated. One can see that the variable nodes of C_{8_1} span 1 replica, while those of C_{8_2} and C_{8_3} span 2 and 3 consecutive replicas, respectively. Therefore, the multiplicity of block cycle C_{8_1} is multiplied by L in the corresponding SC code with the coupling length L. Similarly, 8-cycles C_{8_2} and C_{8_3} are repeated for $L-1$ and $L-2$ times in the graph of the corresponding SC code.*

For a given block cycle C that moves to the corresponding SC Tanner graph, let $x_C + 1$ denote the number of replicas that variable nodes of C span. For instance, in Example 3, we have $x_{C_{8_1}} = 0$, $x_{C_{8_2}} = 1$ and $x_{C_{8_3}} = 2$. Thus, every block cycle C which is supposed to be present in the corresponding SC code (according to Lemma

2), has the multiplicity of $L - x_C$ in the SC code Tanner graph. For a given block cycle C, we call x_C the *width of* C. Next, we focus on calculating the width of a given block cycle.

Proposition 1. *Consider C as a block cycle of length ℓ which does not break during the edge spreading process using spreading matrix B. Corresponding to this ℓ-block cycle we have a set of differential parameters $\Delta B_k, k \in [1, \frac{\ell}{2}]$. The width x_C for this cycle is*

$$x_C = \max\{\mu_1, \mu_2\}, \tag{3.26}$$

where

$$\mu_1 = \max_{k\in[1,\frac{\ell}{2}]} |\Delta B_k|, \quad \mu_2 = \max_{k\in[1,\frac{\ell}{2}]} \left| \sum_{t=0}^{\lfloor \frac{\ell}{4} \rfloor - 1} \Delta B_{\max\{(k+t)_{mod\,\frac{\ell}{2}}, (k+t)_{mod\,(\frac{\ell}{2}+1)}\}} \right|, \tag{3.27}$$

and, $(k+t)_{mod\,\frac{\ell}{2}}$ denotes the remainder of dividing $k+t$ by $\frac{\ell}{2}$.

Proof. For a block cycle of length ℓ, denoted by C_ℓ, we note that the value $|\Delta B_k|$ determines the distance (i.e., the number of replicas traveled) from one variable node to another variable node sharing a (satisfied) check node at row k. We also know that there are $\ell/2$ (satisfied) check nodes involved in every block cycle of length ℓ. Thus, among all of these check nodes included in C_ℓ, $\max_{k\in[1,\frac{\ell}{2}]} |\Delta B_k|$, denoted by μ_1, should be taken into account for the calculation of x, as it determines the maximum number of replicas traveled by commuting between two neighboring variable nodes in the SC code Tanner graph.[4] In addition to it, we also need to consider the distance between every two variable nodes (not necessarily those sharing an incident check node) in C_ℓ. One can verify that in a block cycle of length ℓ, the maximum number of (satisfied) check nodes included in the shortest path between every two variable nodes is $\lfloor \frac{\ell}{4} \rfloor$ [24],[5] which implies that the largest distance between two variable nodes is calculated based on the summation of $\lfloor \frac{\ell}{4} \rfloor$ terms of ΔB_k, each term corresponds to one (satisfied) check node of that block cycle. Having $\ell/2$ variable nodes in a block cycle of length ℓ, we need to calculate the absolute value of the summation of $\lfloor \frac{\ell}{4} \rfloor$

[4] By neighboring variable nodes we mean a pair of variable nodes sharing an incident check node.

[5] Note that $\lfloor \frac{\ell}{4} \rfloor$ defines the maximum number of check nodes included in the shortest path between two variable nodes that (in general) do not share any incident check node. If two variable nodes share a check node, there exists exactly one check node in the shortest path between them.

terms of ΔB_k's for each pair of variable nodes with no common check node (if there exist any)[6]. The maximum absolute value among all these $\ell/2$ summations, denoted by μ_2, is then compared with $\max_{k\in[1,\frac{\ell}{2}]}|\Delta B_k|$ to identify the number of replicas in the SC Tanner graph spanned by this specific cycle C_ℓ. The maximum of these two values (μ_1 and μ_2) is then considered as x_C, which implies that cycle C_ℓ is replicated by $L - x_C$ in the graph of the resultant SC code. ■

From Proposition 1 we immediately conclude the following result, which is well-known, see e.g., [35].

Proposition 2. *The multiplicity of cycles of length ℓ in a protograph-based SC code is*

$$N_\ell = \sum_{x=0}^{\lfloor \frac{\ell}{4} \rfloor m} (L-x) \times N^x_{\ell,block} \tag{3.28}$$

where, $N^x_{\ell,block}$ is the multiplicity of block cycles of length ℓ with width x .

Proof. Referring to Proposition 1 and Lemma 3 we showed that every block cycle of length ℓ could be replicated by $L - x$ in the corresponding SC Tanner graph, where $0 \le x \le \lfloor \frac{\ell}{4} \rfloor m$. Thus, having $N^x_{\ell,block}$ block cycles with the width x, Equation (3.28) is concluded for the enumeration of cycles of length ℓ in the corresponding SC code Tanner graph. ■

Example 4. *Consider a* 10*-block cycle in a block code base graph, so that converting this block code to its SC version using a spreading matrix B with $m = 2$ (shown in Fig. 3.5-(b)) does not break this cycle. The variable nodes of C_{10} span the parity check matrix of the corresponding SC code as depicted in Fig. 3.5(a). Using Proposition 2, we need to find the width of this bock cycle according to the values of ΔB_i's. Thus, we have*

$$\begin{aligned}
\mu_1 &= \max_{k\in[1,5]} |\Delta B_k| = \max\{|2|, |-1|, |-1|, |-1|, |1|\} = 2 \\
\mu_2 &= \max\{|\Delta B_1 + \Delta B_2|, |\Delta B_2 + \Delta B_3|, |\Delta B_3 + \Delta B_4|\} \\
&= \max\{|2-1|, |-1-1|, |-1-1|, |-1+1|, |1+2|\} = 3.
\end{aligned} \tag{3.29}$$

[6]It is known that for cycles of length 4 and 6, there is no pair of variable nodes without an adjacent check node.

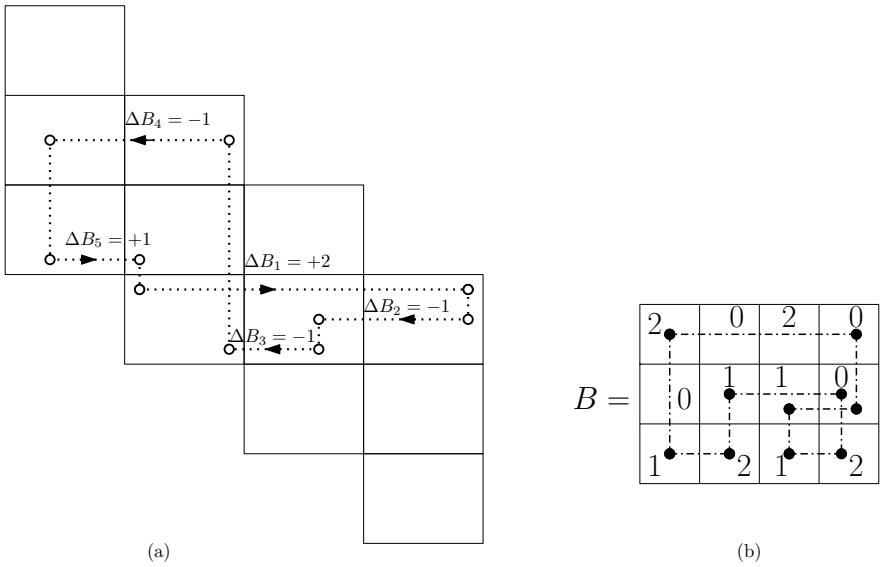


Figure 3.5: (a) Block cycle of length 10 spanning the parity check matrix of an SC code with $m = 2$ and $L = 4$, (b) the corresponding spreading matrix B.

Thus, $x = \max\{2, 3\} = 3$, and C_{10} is replicated by $L - 3$ in the corresponding SC code. This is also clear from Fig. 3.5.

Remark 3. *From Proposition 2 it is evident that the multiplicity of each cycle in an SC code is a linear function in terms of L.*

3.3.2 Asymptotic expected value for the multiplicity of ℓ-cycles in protograph-based SC codes

So far in Proposition 2 we performed a deterministic study over the cycle distribution of protograph-based SC codes. Thus, based on a fixed deterministic spreading matrix B, and the ℓ-cycle distributions in a given block code (i.e., the multiplicity of ℓ-block cycles) we could find the multiplicity of ℓ-cycles within the corresponding SC code. Now, as the second scenario, an SC code is constructed from a given block code using a random spreading matrix, and in this case, we are interested in the average multiplicity of cycles of different lengths within the SC code. In this regard, having the probability of the existence of ℓ-cycles in the SC code, one can find the expected value for the number of these cycles accordingly. For this, the values of x must be also calculated accordingly and enter the calculations. Nevertheless, if we assume that L is large enough such that $L \gg \lfloor \frac{\ell}{4} \rfloor m + 1$, since $0 \leq x \leq \lfloor \frac{\ell}{4} \rfloor m$, we have $L - x \approx L$,

which simplifies the calculation. It is worth mentioning that the assumption of large L, for which $L \gg \lfloor \frac{\ell}{4} \rfloor m + 1$, is of practical interest to reduce the rate-loss appeared in SC codes compared to their underlying block codes [61]. In this regard, we present the following general theorem.

Theorem 5. *In a protograph-based SC-LDPC code with coupling length $L \gg \lfloor \frac{\ell}{4} \rfloor m+1$ and memory m made from a protograph-based LDPC block code through an edge spreading process using a random B matrix, the expected value for the multiplicity of ℓ-cycles is approximated by*

$$\mathbb{E}(N_\ell) \approx \sum_{i=1}^{T} N_\ell^{TBC_i} \times \Pr(C_{\ell_i}) \times L, \tag{3.30}$$

where T defines the maximum number of different TBC walk structures of length ℓ in the block code, $N_\ell^{TBC_i}$ denotes the multiplicity of a specific TBC walk of length ℓ (with structure i) in the block code base graph having the zero permutation shift (i.e., satisfying Lemma 1), and $\Pr\left(C_{\ell_i}\right)$ refers to the probability that such a TBC walk with structure i remains unbroken during a random edge spreading process.[7]

Proof. Suppose that cycles of length ℓ in a given protograph-based LDPC code has T different TBC walk structures. Referring to Theorems 1-3 we have the probability distribution function that each TBC walk structure remains unbroken during a random edge spreading process (for $\ell \in \{4, 6, 8\}$). Let us denote this probability for a specific TBC walk i of length ℓ, whose permutation shift is zero, by $\Pr\left(C_{\ell_i}\right)$. In this respect, if we have $N_\ell^{TBC_i}$ number of TBC walks with structure i in the block code base graph (with permutation shift zero), then the expected value for the multiplicity of ℓ-cycles, made from such TBC walk structures, in the corresponding SC Tanner graph is

$$N_\ell^{TBC_i} \times \Pr(C_{\ell_i}) \times (L - x), \tag{3.31}$$

where x is the width of the TBC walk structure i. Assuming that $L \gg \lfloor \frac{\ell}{4} \rfloor m + 1$, we have $L - x \approx L$ and thus, Equation (3.31) turns into $N_\ell^{TBC_i} \times \Pr(C_{\ell_i}) \times L$. Summing over all T different TBC walk structures and using the approximation of $L - x \approx L$, the proof is complete. ■

[7] Note that the approximation in (3.30) is valid in the sense that we assume a large L such that the effect of x is negligible. In other words, assuming $L \gg \lfloor \frac{\ell}{4} \rfloor m+1$ and the fact that $0 \leq x \leq \lfloor \frac{\ell}{4} \rfloor m$, we conclude that $L - x \approx L$.

In light of Theorem 5 and Theorems 1-3, one is able to verify that if $\ell = 4$ or $\ell = 6$, we have $T = 1$, as the TBC walks are in fact the same as the cycles for these two cases and there exists only one TBC walk structure for either of $\ell = 4$ or $\ell = 6$. However, for $\ell = 8$, as shown in Fig. 3.2, we have $T \leq 9$, as there exist maximum 9 different TBC walk structures of length 8 in a given base graph.

Depending on the structure for the TBC walk of length ℓ in the block code base graph, we may have different $\Pr\left(C_{\ell_i}\right)$ values, each associated to one structure (the structure i). This is particularly the case for cycles of length larger than 6. For 8-cycles, we present the following corollary:

Corollary 1. *Referring to (3.30), and the fact that 8-cycles in a lifted graph come from 9 different TBC walks of the same length in the base graph (shown in Fig. 3.2), the multiplicity of 8-cycles in an SC code Tanner graph with coupling length $L \gg 2m+1$ is*

$$N_8^{TBC} \times \Pr(C_8^{(f,g)}) \times L \leq \mathbb{E}(N_8) \leq N_8^{TBC} \times \Pr(C_8^{(h)}) \times L, \tag{3.32}$$

where, N_8^{TBC} denotes the number of all cycles of length 8 in the block code (with any TBC walk structure from those in Fig. 3.2). $\Pr(C_8^{(f,g)})$ and $\Pr(C_8^{(h)})$ also refer to the probability of existence of the TBC walk structures of length 8 in Fig. 3.2(f)-(g) and Fig. 3.2(h), respectively.

Corollary 1 is based on the fact that TBC walk structures in Fig. 3.2(f)-(g) have the lowest probability of existence, while those with the structure in Fig. 3.2(h) offer the highest probability of existence in the corresponding SC code, according to Fig. 3.3.

3.4 Distribution of Trapping Sets in SC Codes

In the next subsection we calculate the distributions of specific TSs. Next, we present a general bound for the distribution of any TS.

3.4.1 Number of $(4, 2)$ LETS in protograph-based SC codes

It has been mentioned in [26] that for circulant-based codes with $d_v = 3$, simulated over AWGN channels, the ASs (equivalently TSs) in the class of $(4, 2)$, which consist

of two small cycles of length 6 (denoted by C_{6_1} and C_{6_2} in Fig. 3.6) are one of the most harmful TSs within the error floor region. In this regard, we present the following results to analyze the distribution of such harmful trapping sets for a protograph-based SC code with $d_v = 3$ and $g = 6$.

Theorem 6. *Suppose that we have a $(4, 2)$ LETS in a block code Tanner graph with $d_v = 3$ and $g = 6$. By considering memory m and a random B matrix whose non-negative entries are assigned independently and identically with a uniform distribution from the range $[0, m]$, the probability that such a LETS does not break up during the edge spreading process when $L \geq 2m + 1$ is upper and lower bounded as*

$$\Pr\big((4,2)\ \textit{LETS in block code remains unbroken in corresponding SC code}\big) \leq \frac{1597m^6 + 9582m^5 + 25369m^4 + 37596m^3 + 33832m^2 + 18024m + 5040}{5040(m+1)^8}$$

$$\Pr\big((4,2)\ \textit{LETS in block code remains unbroken in corresponding SC code}\big) \geq \begin{cases} \frac{7m^4+28m^3+48m^2+40m+21}{24(m+1)^6} & \textit{if } m \textit{ is odd,} \\ \frac{7m^4+28m^3+48m^2+40m+24}{24(m+1)^6} & \textit{if } m \textit{ is even} \end{cases} \tag{3.33}$$

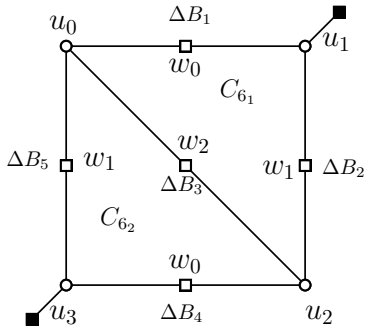


Figure 3.6: The induced subgraph of a $(4, 2)$ LETS with two common edges between the contributing 6-cycles. Satisfied and unsatisfied check nodes are denoted by white and black squares, respectively.

Proof. The proof is presented in Appendix C. ∎

According to [26], we know that the variable nodes of every (a, b) TS can span at most $\lfloor \frac{a}{2} \rfloor m + 1$ consecutive replicas. This implies that such a TS is repeated by $L - x$ in the Tanner graph of the corresponding SC code with coupling length L, where $0 \leq x \leq \lfloor \frac{a}{2} \rfloor m$. Using the assumption of $L \gg \lfloor \frac{a}{2} \rfloor m$, we have $L - x \approx L$. Thus, if the multiplicity of (a, b) TSs in the block code and the probability of existence of such TSs are given, we can find the asymptotic expected value for the multiplicity of these TSs in the corresponding SC code Tanner graph after performing the edge spreading process using a random spreading matrix B.

3.4.2 General bounds for TSs

In this section, we present an analysis of the distribution of TSs (in general) in SC codes. Let $\mathcal{S}$ be a TS, which has n cycles. By Lemma 2, we can assign an equation (similar to Equation (3.1)) to each of those n cycles to find the necessary and sufficient conditions for the existence of that TS. We call the set of those equations *cycle space*. Suppose that there are k cycles $C_{l_1}, C_{l_2}, \ldots, C_{l_k}$ such that the set of their corresponding equations can span all equations in the set cycle space. We call these cycles the fundamental cycles of LETS $\mathcal{S}$. By Lemma 2, the LETS $\mathcal{S}$ exists if and only if the equations in the cycle space corresponding to the fundamental cycles of $\mathcal{S}$ hold.

Theorem 7. *Let G be the Tanner graph of a protograph-based LDPC block code and suppose that we have a TS $\mathcal{S}$ in that Tanner graph. Consider memory m, and a large coupling length L to cover all the variable nodes in $\mathcal{S}$. Then, we apply the edge spreading process using a spreading matrix B whose non-negative entries are assigned independently and identically with a uniform distribution in $[0, m]$. Assume that the set of fundamental cycles of $\mathcal{S}$ consists of k cycles. Then, the probability that $\mathcal{S}$ remains unbroken in the corresponding SC code Tanner graph is $\mathcal{O}(\frac{1}{m^k})$.*

Proof. Let $C_{l_1}, C_{l_2}, \ldots, C_{l_k}$ be the set of fundamental cycles of $\mathcal{S}$. By Lemma 2, we have

$$\begin{aligned} &\Pr\left(\mathcal{S} \text{ in the block code remains in the SC code}\right) \\ &= \Pr\left(C_{l_1}\right) \Pr\left(C_{l_2} | C_{l_1}\right) \Pr\left(C_{l_3} | C_{l_1}, C_{l_2}\right) \ldots \Pr\left(C_{l_k} | C_{l_1} \ldots C_{l_{k-1}}\right). \end{aligned} \tag{3.34}$$

Corresponding to each cycle, like C_{l_t} of length ℓ, suppose that the differential parameters for the contributing edges in this cycle makes the set $\Delta B^{l_t} =$

$\{\Delta B_1, \Delta B_2, \ldots, \Delta B_{\ell/2}\}$. Thus, $\Pr\left(C_{l_t}|C_{l_1}, \ldots, C_{l_{t-1}}\right)$ in (3.34) becomes

$$\Pr\left(C_{l_t}|C_{l_1}, \ldots, C_{l_{t-1}}\right) = \Pr\Big(\sum_{i=1}^{\ell/2} \Delta B_i = 0 | C_{l_1}, \ldots C_{l_{t-1}}\Big). \tag{3.35}$$

We say that the differential parameter ΔB_i appears positive (negative, respectively) in the equation corresponding to a cycle if ΔB_i ($-\Delta B_i$, respectively) is in that equation. For instance, in the equation $\Delta B_4 + \Delta B_5 - \Delta B_3 = 0$, the differential parameter ΔB_4 and ΔB_5 appear positive and the differential parameter ΔB_3 appears negative. Let S be the set of edges (equivalently, differential parameters) that are in cycle C_{l_t} and also in at least one of the cycles $C_{l_1}, \ldots, C_{l_{t-1}}$ (positive or negative). We denote the size of set S by $|S|$. By (3.35), we have

$$\Pr\left(C_{l_t}|C_{l_1}, \ldots, C_{l_{t-1}}\right) = \Pr\Big(\sum_{\Delta B_i \in \Delta B^{l_t} - S} \Delta B_i + \sum_{\Delta B_i \in S} \Delta B_i = 0 | C_{l_1}, \ldots, C_{l_{t-1}}\Big). \tag{3.36}$$

By (3.36), and using the law of total probability for conditional probabilities we have

$$\begin{aligned}
&\Pr\left(C_{l_t}|C_{l_1}, \ldots, C_{l_{t-1}}\right) \\
&= \sum_{y=-|S|m}^{|S|m} \Big(\Pr\Big(\sum_{\Delta B_i \in \Delta B^{l_t} - S} \Delta B_i + \sum_{\Delta B_i \in S} \Delta B_i = 0 | \sum_{\Delta B_i \in S} \Delta B_i = y, C_{l_1}, \ldots C_{l_{t-1}}\Big) \\
&\times \Pr\Big(\sum_{\Delta B_i \in S} \Delta B_i = y | C_{l_1}, \ldots C_{l_{t-1}}\Big)\Big)
\end{aligned} \tag{3.37}$$

Since $\Delta B^{l_t} - S$ denotes the set of edges that are exclusively existed in $\mathcal{C}_{l_t}$ and they are not included in any other cycles from $\mathcal{C}_{l_1}, \ldots \mathcal{C}_{l_{t-1}}$, the random variable $\sum_{\Delta B_i \in \Delta B^{l_t} - S} \Delta B_i$ is independent from $\mathcal{C}_{l_1}, \ldots \mathcal{C}_{l_{t-1}}$. Thus, we can continue (3.37)

as follows:

$$\Pr(\mathcal{C}_{l_t}|\mathcal{C}_{l_1},\ldots,\mathcal{C}_{l_{t-1}}) = \sum_{y=-|S|m}^{|S|m} \Big(\Pr\big(\sum_{\Delta B_i \in \Delta B^{l_t}-S} \Delta B_i + \sum_{\Delta B_i \in S} \Delta B_i = 0 | \sum_{\Delta B_i \in S} \Delta B_i = y \big) \times \Pr\big(\sum_{\Delta B_i \in S} \Delta B_i = y | C_{l_1},\ldots C_{l_{t-1}} \big) \Big)$$
$$= \sum_{y=-|S|m}^{|S|m} \Big(\Pr\big(\sum_{\Delta B_i \in \Delta B^{l_t}-S} \Delta B_i = -y \big) \times \Pr\big(\sum_{\Delta B_i \in S} \Delta B_i = y | C_{l_1},\ldots C_{l_{t-1}} \big) \Big) \quad (3.38)$$

By Lemma 7, we know that $\Pr\big(\sum_{\Delta B_i \in \Delta B^{l_t}-S} \Delta B_i = -y\big) = \mathcal{O}(\frac{1}{m})$. So, by (3.38) we have

$$\Pr\big(C_{l_t}|C_{l_1},\ldots C_{l_{t-1}}\big)$$
$$= \sum_{y=-|S|m}^{|S|m} \mathcal{O}(\frac{1}{m}) \times \Pr\big(\sum_{\Delta B_i \in S} \Delta B_i = y | C_{l_1},\ldots C_{l_{t-1}} \big)$$
$$= \mathcal{O}(\frac{1}{m}) \sum_{y=-|S|m}^{|S|m} \Pr\big(\sum_{\Delta B_i \in S} \Delta B_i = y | C_{l_1},\ldots C_{l_{t-1}} \big) = \mathcal{O}(\frac{1}{m}). \quad (3.39)$$

Thus, by (3.39), and Lemma 7, the initial equation in (3.34) results in

$$\Pr\big(\mathcal{S} \text{ in the block code remains in the SC code}\big) = \mathcal{O}(\frac{1}{m^k}). \quad (3.40)$$

■

As a result of Theorem 7, following remark is obtained immediately.

Remark 4. *The more fundamental cycles contributing to a TS results in a larger probability of breakage for this TS during a random edge spreading process. As a result, this TS is less likely to exist in the corresponding SC code.*

Referring to [20], one can modify Theorem 7 for different categories of trapping sets. To this end, we start with the following corollary and then present the specific results for the category of ETSs. We should note that induced subgraph $\mathcal{S}$ of each TS in an LDPC code includes $|E(\mathcal{S})| - |F(\mathcal{S})| + 1$ fundamental cycles, where, $E(\mathcal{S})$ and $F(\mathcal{S})$ denote the number of edges and nodes included in $\mathcal{S}$. Thus, according to Theorem 7, we have the following results.

Corollary 2. *Suppose that we have a TS $\mathcal{S}$ in an LDPC block code Tanner graph, and we consider a random edge spreading process based on the memory m and a large coupling length L to create the corresponding SC code Tanner graph. The probability that $\mathcal{S}$ remains unbroken in the SC code is then equal to $\mathcal{O}(\frac{1}{m^{|E(\mathcal{S})|-|F(\mathcal{S})|+1}})$, where $E(\mathcal{S})$ and $F(\mathcal{S})$ denote the number of edges and nodes included in $\mathcal{S}$.*

Lemma 8. *Assume an induced subgraph $\mathcal{S}$ for an (a, b) ETS in a variable-regular block code with variable node degree d_v. Considering a large enough coupling length L, the structure $\mathcal{S}$ remains unbroken in the corresponding SC code with the following probability,*

$$\Pr\big((a,b)ETS\,\mathcal{S}\ in\ the\ block\ code\ exists\ in\ the\ SC\ code\big) = \mathcal{O}(\frac{1}{m^{\frac{a(d_v-2)-b}{2}+1}}). \quad (3.41)$$

Where m denotes the memory of SC code and the non-negative entries of B matrix are independently and identically distributed with a uniform distribution from the interval $[0, m]$.

Proof. In [20] it was shown that for an (a, b) ETS $\mathcal{S}$ in a variable-regular Tanner graph with variable node degree d_v, we have

$$|F(\mathcal{S})| - |E(\mathcal{S})| = a + \frac{b - ad_v}{2}. \quad (3.42)$$

Using Corollary 2 along with (3.42) we get

$$|E(\mathcal{S})| - |F(\mathcal{S})| + 1 = -a - \frac{b - ad_v}{2} + 1 = \frac{a(d_v - 2) - b}{2} + 1. \quad (3.43)$$

■

As a result of Lemma 8, the following corollary is achieved.

Corollary 3. *In a variable-regular LDPC block code with a fixed variable node degree d_v, those (a, b) ETSs with larger a and/or smaller b are more likely to break during a random edge spreading process.*

Corollary 3 can imply the improved error floor performance of SC codes compared with their underlying block codes. This is due to the fact that (a, b) ETSs with smaller b are mainly to blame for the performance degradation of LDPC codes in the error floor region. Thus, our analysis proved that such structures are more likely to break

during the edge spreading process and thus, leading to a better error floor performance for the corresponding SC codes.

3.5 Numerical Results

Our numerical results are divided into three parts. In the first part, we aim at investigating the probability distribution functions for the existence of cycles of different lengths in the SC codes. At the second part, we consider some practical QC-SC codes with $d_v = 3$ and compare their actual number of cycles of different lengths as well as $(4, 2)$ LETSs with our theoretical expected values. We also compare the average multiplicity of cycles and trapping sets using a random spreading matrix, with some of the existing designed codes in the literature. Finally, at the third part, we describe how our analysis could demonstrate some observations mentioned in the literature.

3.5.1 Probability of breakage of cycles

In this subsection, we assess the accuracy of our calculations for the cycle distribution in SC codes given in Section 3.2. First of all, we consider different values of γ and c and calculate the multiplicity of cycles (i.e., TBC walks) of length 4 and 6 in these fully-connected $\gamma \times c$ base graphs.[8] This value for a cycle of length k, which is called as $N_k^{\text{block base graph}}$, could be found using [18, Theorem 5] or through a computer search. Then, we generate a number of $\gamma \times c$ spreading matrices whose entries are assigned independently and identically with a uniform distribution in $[0, m]$. Next, we need to check Lemma 2 for each of these $N_k^{\text{block base graph}}$ cycles to count how many of them are not broken during the edge spreading process. For the cycle of length k, we call this number as *Actual* N_k^U, where U stands for the word *unbroken*. In comparison, multiplying the probability of existence of each cycle of length k ($k \in \{4, 6\}$) introduced in Theorems 1 and 2 by $N_k^{\text{block base graph}}$, we obtain $\mathcal{E}(N_k)^{\text{U}}$ as the expected value for the multiplicity of unbroken k-cycles in the corresponding SC code base graph. As it could be observed from Table 3.1, our theoretical calculations almost match the numerical results. It should be noticed that at this part, we only

[8] We only focus on 4-cycles and 6-cycles at this part because for these two types of cycles, TBC walks are the same as cycles, and we presented one single equation for the probability of existence of these cycles. Numerical results for larger cycles (cycles of length 8) along with the trapping sets are discussed in the next subsection.

count how many TBC walks (cycles) remain unbroken during the edge spreading process. In fact, each one of these unbroken cycles, with a zero permutation shift, could have a multiplicity of $L-x$ in the final SC code Tanner graph, as described in the next part of our numerical results.

3.5.2 Expected value for the multiplicity of cycles and LETSs in SC codes

In this part, we provide some numerical results to assess the theoretical analysis regarding the expected value for the multiplicity of cycles and LETSs in an SC code Tanner graph. As the first experiment, we consider $(3,p)$ AB-LDPC codes (with the parity check matrix given in (2.3)) with different values of p as the underlying block code. Then, we find the multiplicity of cycles of length 6 and 8 (i.e., block cycles of length 6 and 8) in addition to the number of $(4,2)$ LETSs in these block codes, and denote these numbers in Tables 3.2-3.3 by N_S^{block}, where S shows either the length of the cycle or the class of LETS. Then, using the assumption of large L (i.e., $L \gg \lfloor \frac{a}{2} \rfloor m+1$ for a TS in the (a,b) class), we find the expected value for each of these structures as a function of L and present them in the tables.[9] Considering different random B matrices whose entries have i.i.d. uniform distribution from $[0,m]$, one can find the actual multiplicity of cycles and LETSs in the corresponding SC code Tanner graph, which is denoted by "Actual N_S^{SC}".[10] As is evident from Tables 3.2-3.3, the theoretical values match the actual values either in a low rate AB-SC code ($p=5,7$, Table 3.2) or a high-rate AB-SC code ($p=31$, Table 3.3).

In order to enumerate the multiplicity of cycles and LETSs in the SC codes as a function of L, given in Table 3.2-3.3, we use the following approaches:

Enumeration of cycles in SC codes

Proposition 2 presents a closed form for the multiplicity of cycles of length ℓ in terms of L and m, which is stated as $\sum_{x=0}^{\lfloor \frac{\ell}{4} \rfloor m}(L-x) \times N_{\ell,block}^{x}$. Therefore, by only inspecting

[9] According to Theorem 5, these expected values are found through a multiplication of N_S^{block} by the calculated probability functions for different cycles and LETSs. For those cases with upper and lower bounds we also present the corresponding bounds for the expected values.

[10] For the numerical values denoted by "Actual N_S^{SC}" in Tables 3.2-3.3, we consider $L=300$.

Table 3.1: Multiplicities of cycles of different lengths in the base graph of a block code before and after performing the edge spreading process using a random spreading matrix B.

$\gamma \times c$	B matrix	m	$N_4^{\text{block base graph}}$	$N_6^{\text{block base graph}}$	$\mathcal{E}(N_4)^{\text{U}}$ / Actual N_4^{U}	$\mathcal{E}(N_6)^{\text{U}}$ / Actual N_6^{U}
3×4	0 0 1 0 1 0 0 0 1 0 0 1	1	18	24	6.75 / 5	7.5 / 6
3×5	0 0 1 0 0 0 1 0 1 0 0 0 1 1 1	1	30	60	11.25 / 8	18.75 / 19
3×5	1 1 2 2 2 2 0 2 0 1 1 1 0 0 2	2	30	60	7.03 / 6	11.60 / 9
4×6	2 2 1 1 2 1 1 2 2 1 1 1 0 0 1 0 2 0 0 0 0 1 0 2	2	90	480	21.11 / 19	92.84 / 99
3×4	3 0 3 1 0 1 2 1 1 3 2 2	3	18	24	3.09 / 1	3.39 / 3
4×8	3 2 2 0 3 2 1 2 1 0 0 0 0 0 1 0 3 3 1 1 1 3 1 0 3 1 0 1 0 0 3 3	3	168	1344	28.87 / 30	190.31 / 203
3×5	2 4 1 2 1 3 1 2 3 4 4 2 0 0 1	4	30	60	4.08 / 6	6.72 / 10
4×8	2 1 3 3 0 2 2 3 3 3 1 3 3 0 0 2 4 1 2 1 3 1 2 3 4 4 2 0 0 1 4 1	4	168	1344	22.84 / 26	150.61 / 154
5×10	2 0 4 4 2 0 1 2 2 1 3 3 1 0 1 1 2 2 0 1 4 0 4 3 2 2 1 2 4 2 2 1 2 3 3 1 1 4 0 4 4 3 0 1 1 3 0 3 0 3	4	450	7200	61.20 / 61	806.86 / 796

Table 3.2: Multiplicity of cycles and $(4,2)$ LETSs in the AB-SC codes and their underlying AB-LDPC block code with $L \gg 2m+1$.

p	m	B matrix	N_6^{block}	N_8^{block}	$N_{(4,2)}^{block}$	$\frac{\mathcal{E}(N_6^{SC})}{\text{Actual } N_6^{SC}}$	$\frac{\mathcal{E}(N_8^{SC})}{\text{Actual } N_8^{SC}}$	$\frac{\mathcal{E}(N_{(4,2)}^{SC})}{\text{Actual } N_{(4,2)}^{SC}}$
5	1	$\begin{matrix}1&0&0&0&1\\1&0&1&0&1\\1&1&0&1&0\end{matrix}$	100	600	150	$\frac{31.25L}{25L-20}$	$\frac{131.25L \le \mathcal{E}(N_8^{SC}) \le 225L}{140L-140}$	$\frac{14.06L \le \mathcal{E}(N_{(4,2)}^{SC}) \le 15.23L}{10L-10}$
5	1	$\begin{matrix}1&1&1&0&0\\1&1&1&0&1\\0&0&0&1&0\end{matrix}$	100	600	150	$\frac{31.25L}{35L-15}$	$\frac{131.25L \le \mathcal{E}(N_8^{SC}) \le 225L}{165L-75}$	$\frac{14.06L \le \mathcal{E}(N_{(4,2)}^{SC}) \le 15.23L}{15L-5}$
5	1	$\begin{matrix}0&1&1&1&1\\1&0&1&0&0\\1&1&0&1&1\end{matrix}$	100	600	150	$\frac{31.25L}{30L-15}$	$\frac{131.25L \le \mathcal{E}(N_8^{SC}) \le 225L}{120L-65}$	$\frac{14.06L \le \mathcal{E}(N_{(4,2)}^{SC}) \le 15.23L}{10L-5}$
5	2	$\begin{matrix}2&2&0&2&1\\0&0&1&2&2\\0&2&2&1&2\end{matrix}$	100	600	150	$\frac{19.34L}{20L-25}$	$\frac{81.48L \le \mathcal{E}(N_8^{SC}) \le 140.74L}{105L-165}$	$\frac{5.41L \le \mathcal{E}(N_{(4,2)}^{SC}) \le 5.86L}{5L-5}$
7	1	$\begin{matrix}1&0&1&0&0&0&0\\0&0&0&0&0&0&1\\1&1&0&0&0&1&0\end{matrix}$	294	3087	441	$\frac{91.87L}{112L-56}$	$\frac{675.28L \le \mathcal{E}(N_8^{SC}) \le 1157.6L}{798L-595}$	$\frac{41.34L \le \mathcal{E}(N_{(4,2)}^{SC}) \le 44.8L}{49L-35}$
7	1	$\begin{matrix}0&0&0&0&0&0&1\\0&1&1&0&0&1&0\\0&1&0&1&1&1&1\end{matrix}$	294	3087	441	$\frac{91.87L}{98L-35}$	$\frac{675.28L \le \mathcal{E}(N_8^{SC}) \le 1157.6L}{805L-574}$	$\frac{41.34L \le \mathcal{E}(N_{(4,2)}^{SC}) \le 44.8L}{49L-14}$
7	2	$\begin{matrix}0&1&2&2&2&1&0\\2&2&2&2&2&1&1\\0&2&0&0&0&0&2\end{matrix}$	294	3087	441	$\frac{56.86L}{49L-21}$	$\frac{419.22L \le \mathcal{E}(N_8^{SC}) \le 724.11L}{406L-476}$	$\frac{15.93L \le \mathcal{E}(N_{(4,2)}^{SC}) \le 17.22L}{14L-7}$

Table 3.3: Multiplicity of cycles and $(4, 2)$ LETSs in the AB-SC codes and their underlying AB-LDPC block code with $L \gg 2m + 1$ and $p = 31$.

m	B	N_6^{block}	N_8^{block}	$N_{(4,2)}^{block}$	$\frac{\mathcal{E}(N_6^{SC})}{\text{Actual } N_6^{SC}}$	$\frac{\mathcal{E}(N_8^{SC})}{\text{Actual } N_8^{SC}}$	$\frac{\mathcal{E}(N_{(4,2)}^{SC})}{\text{Actual } N_{(4,2)}^{SC}}$
1	B_1	28830	1859535	43245	$\frac{9009.4L}{8897L - 5642}$	$\frac{406773.3L \leq \mathcal{E}(N_8^{SC}) \leq 697325.6L}{514569L - 441812}$	$\frac{4054.2L \leq \mathcal{E}(N_{(4,2)}^{SC}) \leq 4392L}{4216L - 2945}$
1	B_2	28830	1859535	43245	$\frac{9009.4L}{9269L - 5766}$	$\frac{406773.3L \leq \mathcal{E}(N_8^{SC}) \leq 697325.6L}{525357L - 448787}$	$\frac{4054.2L \leq \mathcal{E}(N_{(4,2)}^{SC}) \leq 4392L}{4526L - 3317}$
1	B_3	28830	1859535	43245	$\frac{9009.4L}{9331L - 5921}$	$\frac{406773.3L \leq \mathcal{E}(N_8^{SC}) \leq 697325.6L}{509206L - 428792}$	$\frac{4054.2L \leq \mathcal{E}(N_{(4,2)}^{SC}) \leq 4392L}{4650L - 3317}$
B_1	1 0 1 1 0 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0						
B_2	0 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 0 1 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 1 1 0						
B_3	1 1 0 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 0 1 1 0 0 1 1 1 1 0 1 0 1 0 1 1 0 0 1 0 0 1 1 1 1 1 1 0 1 0 1 1 0 0 0 1 0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0						

the spreading matrix B for each block cycle of length ℓ, which is not broken according to Lemma 2, the corresponding value of x is obtained. Summing over all cycles of length ℓ with the multiplicity of $L-x$ in the SC code, we achieve a linear equation in terms of L for the number of ℓ-cycles in the SC code. Therefore, we use the closed form in Proposition 2 to express the actual number of cycles in terms of L for a given B matrix and an underlying block code.

Enumeration of LETSs in SC codes

Referring to [26], we know that the multiplicity of each class of LETS in an SC code is a linear function in terms of L. In this regard, assuming the multiplicity of (a,b) LETS as $\alpha L-\beta$, we need to look for the coefficients α and β (non-negative integers) to build the linear function. In fact, if we have two points of this line, we can find the corresponding linear equation. For this aim, first, we consider the minimum coupling length of SC code, which is required to cover all variable nodes of the target LETS in the SC code. This coupling length is $L_1=(\lambda-1)m+1$, where λ is the maximum number of variable nodes included at the shortest path connecting any two variable nodes of (a,b) LETS [24]. Using this coupling length, we have the parity check matrix for SC code C_1. Then, we use the efficient search algorithm of [32] to find the distribution of (a,b) LETS for this code. Denote the multiplicity of (a,b) LETS in C_1 as $N_{(a,b)}^{L_1}$. Increasing L_1 by one, we have $L_2=(\lambda-1)m+2$. Therefore, we construct the SC code C_2 with respect to L_2 and for the second time, we apply search algorithm over this code. Similarly, the multiplicity of (a,b) LETS in C_2 is represented by $N_{(a,b)}^{L_2}$. As a result of this two-step process, we obtain two distinct points of a linear equation. Therefore, it is easy to derive the following equation for the multiplicity of (a,b) LETS in the corresponding SC code:

$$N_{(a,b)\,LETS}=(N_{(a,b)}^{L_2}-N_{(a,b)}^{L_1})L-[N_{(a,b)}^{L_2}L_1-N_{(a,b)}^{L_1}(L_1+1)]=\alpha L-\beta. \tag{3.44}$$

Based on the above discussion and the linear equation in (3.44) we find the multiplicity of LETSs given in Table 3.2-3.3.

As another experiment, we refer to some existing designed SC-LDPC codes in the literature and compare the number of cycles and LETSs in those codes with the average numbers presented in this work. To this end, we consider SC-LDPC codes in [10] whose spreading matrices are optimized to minimize the number of 6-cycles in

Table 3.4: Multiplicity of cycles and LETSs in the optimized AB-SC codes in [10], compared with their corresponding average numbers. The underlying block codes are AB-LDPC codes with $\gamma = 3$ and $p = 17$.

m	N_6^{block}	N_8^{block}	$N_{(4,2)}^{block}$	$\frac{\mathcal{E}(N_6^{SC})}{\text{Actual } N_6^{SC}}$	$\frac{\mathcal{E}(N_8^{SC})}{\text{Actual } N_8^{SC}}$	$\frac{\mathcal{E}(N_{(4,2)}^{SC})}{\text{Actual } N_{(4,2)}^{SC}}$
1	4624	152592	6936	$\frac{1445L}{612L-476}$	$\frac{33387L \le \mathcal{E}(N_8^{SC}) \le 57222L}{37774L-30855}$	$\frac{650.25L \le \mathcal{E}(N_{(4,2)}^{SC}) \le 704.43L}{187L-153}$
2	4624	152592	6936	$\frac{894.35L}{51L-68}$	$\frac{20722L \le \mathcal{E}(N_8^{SC}) \le 35793L}{22117L-28152}$	$\frac{250.54L \le \mathcal{E}(N_{(4,2)}^{SC}) \le 270.98L}{17L-34}$

the resultant SC codes, where the underlying block code is an AB-LDPC code with $\gamma = 3$ and $p = 17$. Then, we use the approach introduced in Section 3.5.2 to find the actual multiplicity of cycles and LETSs for such SC codes, given the optimized spreading matrix in [10]. The results are shown in Table 3.4. For the average numbers reported in this table, we use our calculated probability distribution functions and the assumption of large L (e.g., $L = 300$), similar to Theorem 5, to find the expected values in terms of L.

As is evident from Table 3.4, the designed code in [10] has significantly smaller multiplicity of 6-cycles and $(4, 2)$ LETSs, compared to the corresponding average numbers using a random spreading matrix. This is expected because the codes in [10] are optimized to have small number of 6-cycles. Also, as $(4, 2)$ LETSs consist of two 6-cycles, the less number of 6-cycles leads to the smaller number of $(4, 2)$ LETSs. Moreover, since the design in [10] does not perform any optimization on the 8-cycles, we observe that the number of 8-cycles in such designed codes are not far from their expected values when using a random spreading matrix.

3.5.3 Some trends, inferences, and observations

In this part, we present some results and trends which stem from the analysis provided in the previous sections. As the first experiment, we evaluate the trend of changing the multiplicity of cycles and small LETSs in SC codes for different values of m. Referring to the previous sections, we find the probability distribution functions for

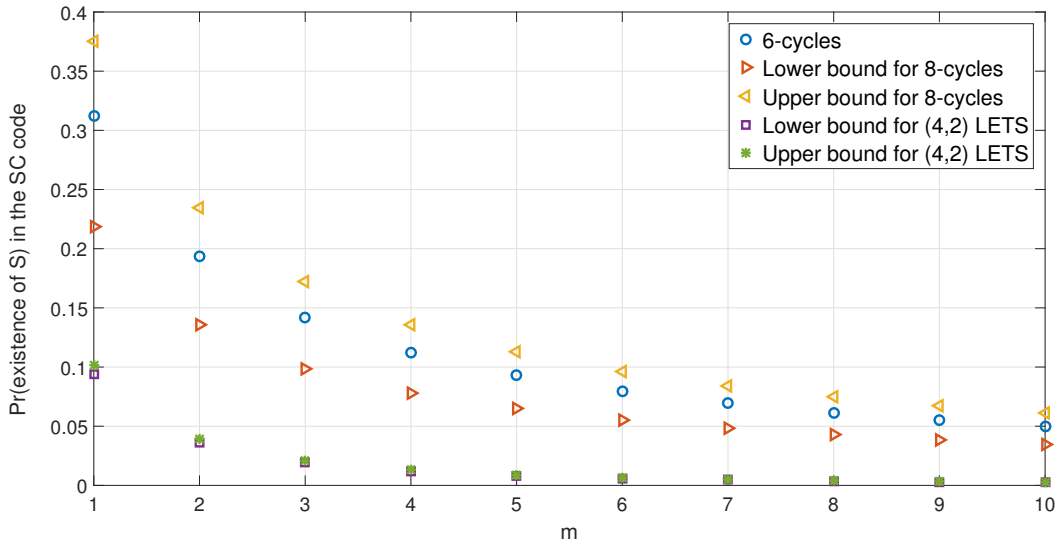


Figure 3.7: Impact of memory m on the probability of existence of cycles and $(4,2)$ LETSs in protograph-based SC codes. (The vertical axis refers to the probability of existence of cycles and $(4,2)$ LETSs in protograph-based SC-LDPC codes after a random edge spreading process.)

the existence of cycles and small LETSs as a function of m. Fig. 3.7 depicts the variation of Pr $\left(\text{existence of}\,\mathcal{S}\right)$ by changing m values from 1 all the way to 10, where $\mathcal{S}$ includes cycles of length 6 to 8 in addition to small LETSs in the $(4,2)$ class.

As is evident from Fig. 3.7, increasing memory results in the smaller probability of existence for LETSs/cycles in the SC code. In particular, going from $m=1$ to $m=2$ results in a significant reduction in the number of small cycles and small trapping sets which are to blame for performance degradation in the error floor region. On the other hand, we know that increasing memory results in a larger constraint length and thus, a higher decoding latency [7,65,78]. This, in fact, implies a trade off between the decoding latency and the performance of SC codes in the error floor region. However, one can see that after $m=5$ and 6, the slope of reduction, particularly for small trapping sets in the $(4,2)$ class, becomes too small, and we may not see a remarkable improvement by increasing m to values larger than $5,6$. Thus, one can suggest to choose m at most 6 in most cases, if the goal is to reduce the multiplicity of such small LETSs.

According to Fig. 3.7, one can observe that the probability of existence of cycles in the corresponding SC code is reduced by increasing the length of the cycles, i.e., cycles of length 6 has a larger probability of existence compared to 8-cycles (considering the

average of the lower and upper bounds for the probability of existence of 8-cycles). As a result of this observation, we present the following statement.

Corollary 4. *Since the probability of breakage of cycles with larger length (during a random edge spreading process) is higher, those trapping set structures containing larger cycles are more likely to break and disappear from the SC code, compared to other trapping sets including smaller cycles. In other words, a trapping set including larger cycles is broken with higher probability during a random edge spreading process.*

As the next part of our experiment, we want to evaluate the results in Fig. 3.7 by comparing them with some practical SC codes. With the assumption of large enough L, we already demonstrated that the multiplicity of different trapping sets and cycles in an SC code are linear functions in terms of L. Moreover, by changing the memory m, such multiplicities would also change. We also discussed the fact that every TS or a cycle in the block code could be broken during the edge spreading process, or they move to the corresponding SC code Tanner graph. As a result, we can develop the following equation

$$N_{\mathcal{S}}^{SC} = N_{\mathcal{S}}^{block} \times f(m, L), \tag{3.45}$$

where, $N_{\mathcal{S}}^{SC}$ and $N_{\mathcal{S}}^{block}$ refer to the multiplicity of a TS with induced subgraph $\mathcal{S}$ in the SC code and the corresponding underlying block code, respectively. And $f(m, L)$ is a function in terms of m and L. Here, we consider a fixed large value for L (e.g., $L = 300$), and normalize $f(m, L)$ over L to specifically investigate the impact of m on our numerical results (we call it as a multiplicative fraction). Thus, for an AB underlying block code with $\gamma = 3$ and two different values of p ($p = 7$ and $p = 13$), one can find the multiplicity of 6-cycles, 8-cycles and $(4, 2)$ LETSs, denoted by $N_{\mathcal{S}}^{block}$. Then, we randomly construct B matrices (with different values of m), whose entries are chosen independently and identically with a uniform distribution from $[0, m]$, and we create the corresponding SC codes with $L = 300$. Afterwards, we calculate the multiplicity of cycles of length 6 and 8 as well as $(4, 2)$ LETSs in such SC codes (denote this number by $N_{\mathcal{S}}^{SC}$), and consequently calculate $f(m, L)$ for these codes. Fig. 3.8(a)-(b) illustrate how $f(m, L)/L$ is varying with different values of $m \in [1, 10]$ for $p = 7$ and $p = 13$, respectively. The overall trend in these figures matches the decreasing trend shown in Fig. 3.7 for all the structures.

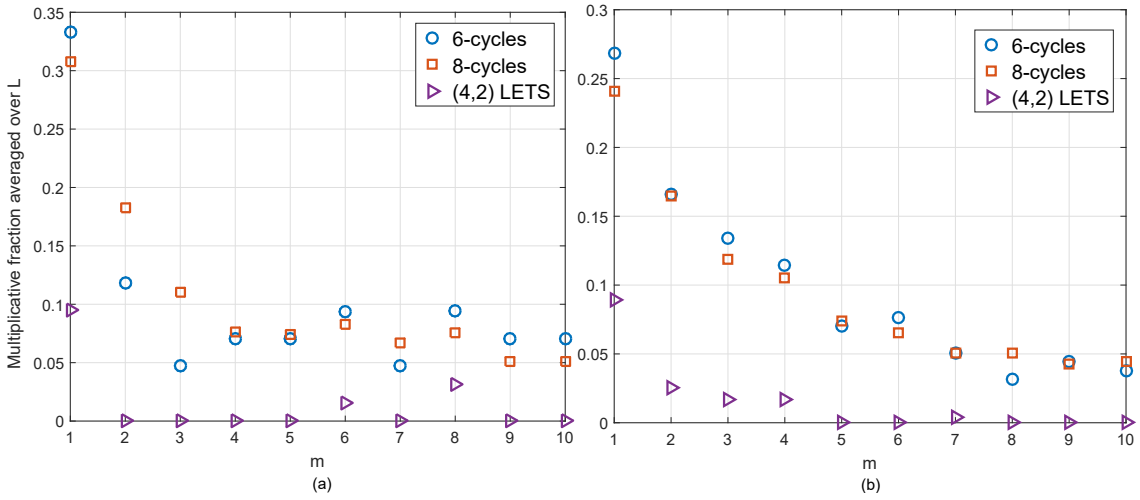


Figure 3.8: Impact of memory m on the normalized multiplicity of cycles and small LETSs in AB-SC codes with $\gamma = 3$ and (a) $p = 7$, (b) $p = 13$. (The vertical axis refers to $f(m, L)/L$ as given in (3.45).)

Chapter 4

Simple Conditions for the Construction of QC-LDPC Codes Free of Small Trapping Sets

4.1 Introduction

In this chapter, we discuss the design of QC-LDPC codes with $g = 8$ that are free of small LETSs. Our design technique is based on imposing simple constraints on the cycles of length 8 in the code's Tanner graph. In the existing literature described in Chapter 1, the most relevant work to ours in the concept of QC-LDPC codes is [80] in which QC-LDPC codes with $d_v = 3$ and $g = 8$ are constructed that are free of LETSs with size $a \leq 8$ and the number of odd-degree check nodes $b \leq 3$.

In this chapter, through a careful examination of the small trapping sets and their relationships with different types of 8-cycles, we impose a less restrictive constraint on 8-cycles compared to that of [80]. This, in general, allows us to find codes that are free of the same range of trapping sets but have a smaller block length compared to those of [80]. Moreover, while the results of [80] are limited to $d_v = 3$, in this work, we also devise a simple constraint on 8-cycles for codes with $d_v = 4$. For both cases of $d_v = 3$ and $d_v = 4$, we also derive a lower bound on the lifting degree and the block length of the codes that can satisfy the constraints imposed on 8-cycles. For $d_v = 3$, the derived lower bound is smaller than that of [80]. In fact, we find codes with $d_v = 3$ that are free of the same range of trapping sets as the codes designed in [80], but with block lengths that are smaller than the lower bound of [80].

The rest of this chapter is as follows: First we give a brief description on some

primary results and the main idea of our design approach in Section 4.2. Sections 4.3 and 4.4 are devoted to the design of QC-LDPC codes with variable node degrees 3 and 4, respectively. Finally, the numerical results are presented in Section 4.5 to demonstrate the superiority of our proposed design technique.

4.2 Definitions and Primary Results

In this chapter, we consider QC-LDPC codes whose base graphs are fully-connected with no parallel edges and are of size $3 \times c$ ($d_v = 3$) or $4 \times c$ ($d_v = 4$). The exponent matrix is based on the structure given in (2.2) with the condition of $0 \leq p_{2,2} \leq ... \leq p_{2,c}$.

Different characterizations of LETSs for variable-regular LDPC codes can be found in [32,45]. Most relevant to the discussions of this chapter is the *layered superset (LSS) characterization* of LETSs [31, 45]. In this characterization, each and every LETS is characterized as an embedded sequence of LETSs that starts from a simple cycle or a small *prime* structure [31] and grows one variable node at a time until it reaches the targeted structure. The added variable node at each step makes connections to m degree-1 check nodes of the parent structure, where $m \geq 2$, to generate the child structure. This expansion is dubbed dot_m in [32].

Consider a Tanner graph G with the set of variable and check nodes V and W, respectively, and let $\mathcal{S}$ be a LETS in G. Suppose that the set of degree-1 check nodes in the subgraph of $\mathcal{S}$ is denoted by $\Gamma_{\mathrm{o}}(\mathcal{S})$. We refer to $\mathcal{S}$ as *LSS non-expandable (LSS-nE)*, if every variable node in $V \backslash \mathcal{S}$ has at most one neighbor in $\Gamma_{\mathrm{o}}(\mathcal{S})$. (We note that for the case of $d_v = 3$, LETSs are the same as EASs, and LSS-nE LETSs are identical to fully EASs [23].) The following result then follows immediately from the definition of LSS-nE LETSs.

Lemma 9. *If $\mathcal{S}$ is an LSS-nE LETS in a Tanner graph G, then none of the children of $\mathcal{S}$ generated by dot_m expansions (following the LSS property) can exist in G.*

Referring to Lemma 1, we know that cycles in QC-LDPC codes (Tanner graphs) correspond to those *tailless backtrackless closed (TBC) walks* in the base graph whose permutation shifts are equal to zero and have no TBC subwalk with zero permutation shift. Having said that, we assume $g = 8$, and closely examine the structure of different types of 8-cycles in cyclic liftings of $3 \times c$ and $4 \times c$ fully-connected base

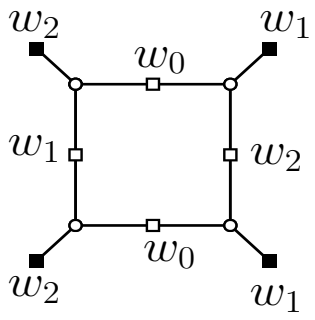


Figure 4.1: A configuration of Type-B 8-cycles in cyclic liftings of the fully-connected $3 \times c$ base graph.

graphs. In Fig. 3.2, we have shown all TBC walks of length 8 that can be mapped to 8-cycles in the lifted graph of a QC-LDPC code. Figs. 3.2(a) and (b) correspond to two 4-cycles that share one node (check and variable node, respectively). Figs. 3.2(d) and (e) present cases where two 4-cycles share two nodes but no edges. Figs. 3.2(f) and (g) are for two 4-cycles that share three nodes (two edges). Fig. 3.2(h) presents a 4-cycle that is traversed twice to form a TBC walk. Fig. 3.2(i) corresponds to two 4-cycles that share two nodes and one edge. Finally, Fig. 3.2(c) corresponds to an 8-cycle in the base graph.

4.3 Constraints on Eight-cycles for QC-LDPC Codes with $d_v = 3$

It is clear that the four check nodes of any 8-cycle in a protograph QC-LDPC code with $d_v = 3$ must belong to either two or three of the row blocks. (See Fig. 3.2(a), (e), (f), (g), (h) and (i).) Correspondingly, we distinguish the 8-cycles as being of *Type A* or *Type B*, respectively. One can see that TBC walks of Figs. 3.2(e), (g) and (h) are mapped to 8-cycles of Type A, whereas, Type-B 8-cycles in the lifted graph are resulted from the TBC walks of Figs. 3.2(a), (f) and (i).

Theorem 8. *In a protograph QC-LDPC code with $g = 8$ and $d_v = 3$, if there is no Type-A 8-cycle and if all Type-B 8-cycles are LSS-nE, then the code is free of LETSs within the range $\mathcal{R} : a \leq 8$ and $b \leq 3$.*

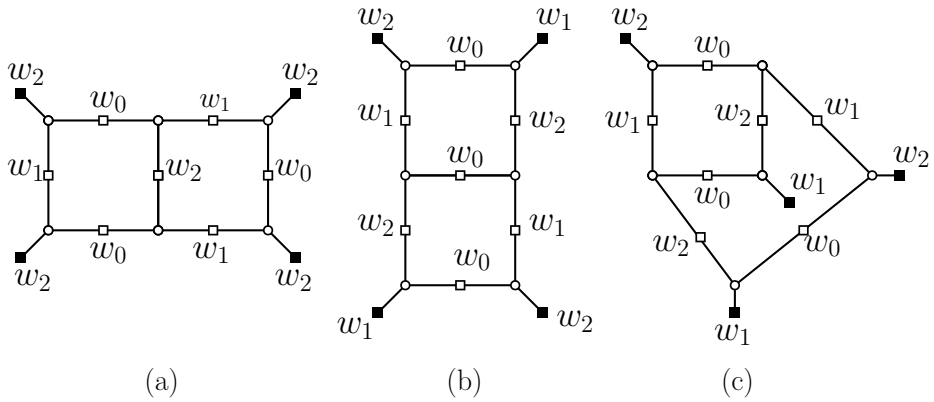


Figure 4.2: Different $(6, 4)$ LETS configurations that are children of the Type-B 8-cycle shown in Fig. 4.1.

Proof. There are a total of 14 non-isomorphic LETS structures for an LDPC code with $g = 8$ and $d_v = 3$ within the range $\mathcal{R}$ [45]. Examining these structures, one can see that they are all either LSSs of 8-cycles or LSSs of the two non-isomorphic structures in the $(6, 4)$ class. By the assumption of the theorem, none of the 8-cycles is LSS expandable. In the following, we also show that none of the two $(6, 4)$ structures is LSS expandable either. The two $(6, 4)$ structures are themselves the children of 8-cycles through a path expansion, pa_2 [32]. Since the code is free of Type-A 8-cycles, the only possible configurations of 8-cycles are limited to those of Type B corresponding to the TBC walks of Figs. 3.2(a), (f) and (i). There are three such configurations, each having one of the three check node indices repeated. For the purpose of this proof, all three configurations are identical, and we arbitrarily consider the one in Fig. 4.1. (Degree-2 and degree-1 check nodes are represented by hollow and solid squares, respectively.) The $(6, 4)$ configurations that can be obtained as a result of the expansion of the 8-cycle of Fig. 4.1 by pa_2 are all shown in Fig. 4.2. (Note that the row blocks of the 3 check nodes adjacent to any variable node must be different, and that since there are no Type-A 8-cycles, there must be three different check node indices within any 8-cycle. Moreover, the application of pa_2 to the two check nodes with indices w_2 and to the two check nodes with indices w_1 and w_2 at the top of Fig. 4.1 will result in configurations that are similar to those of Figs. 4.2(a) and (b), respectively.) One should note that the two configurations in Fig. 4.2 (a) and (b) both have the same topological structure. Now, we examine each of the three configurations in Fig. 4.2, and show that none is LSS expandable. All the degree-1 check nodes of Fig. 4.2(a) have the same index and thus no two of them

can be connected to the same variable node. For Fig. 4.2(b), the only way that the configuration can be expanded based on LSS property is to connect a variable node to indices that are different, i.e., w_1 and w_2. This however, creates either a 6-cycle or a Type-A 8-cycle, both assumed to be missing in the Tanner graph. Finally, for the configuration of Fig. 4.2(c), connecting any pair of degree-1 check nodes with indices w_1 and w_2 through a variable node creates either a 6-cycle or a Type-A 8-cycle, or a Type-B 8-cycle that appears to be LSS expandable, all in contradiction with the assumptions of the theorem. The last case occurs if one is to connect the two degree-1 check nodes connected to the square-shaped 8-cycle. ■

We note that if the conditions of Theorem 8 are satisfied, in addition to the LETSs in the range $\mathcal{R}$, all LETSs in the range $a \leq 9, b \leq 1$, and a large percentage of structures outside $\mathcal{R}$ will also be eliminated. In any construction process, checking the conditions of Theorem 8 is simple. For each 8-cycle, one needs to check (a) whether its check nodes are only from two row blocks (Type A), (and that if the cycle is Type B,) (b) whether there is any variable node outside the cycle that is common among the sets of neighbors of the cycle's degree-1 check nodes. If the answer to (a) or (b) is positive, then the conditions of the theorem are not met. In the following, we derive a lower bound on the lifting degree of the codes that can satisfy the conditions of Theorem 8.

Theorem 9. *Consider a protograph QC-LDPC code with a fully-connected* $3 \times c$ *base graph and* $g = 8$*. If this code has no Type-A* 8*-cycle, then the lifting degree* M *of this code must satisfy* $M \geq c(c-1)+1$*.*

Proof. Type-A 8-cycles can only appear in the Tanner graph of the lifted code as a result of one of the three TBC walks shown in Figs. 3.2(e), (g) or (h) in the base graph. The TBC walk of Fig. 3.2(h) involves a 4-cycle of the base graph. To ensure that the girth of the lifted graph is at least 8 and that no Type-A 8-cycle of this type exists in the lifted graph, the permutation shift p of any 4-cycle in the base graph must satisfy

$$p \neq 0 \mod M, \quad \text{and} \quad 2p \neq 0 \mod M\,. \tag{4.1}$$

The TBC walks of Figs. 3.2(e) and (g) involve 4-cycles that have two check nodes in common. In a fully-connected $3 \times c$ base graph, there are $\binom{c}{2}$ 4-cycles containing two (given) distinct check nodes. Let p_1 and p_2 denote the permutation shifts of two such

cycles. To ensure that no Type-A 8-cycle of the types in Figs. 3.2(e) or (g) exists in the lifted graph, in addition to (4.1), p_1 and p_2 must satisfy

$$p_1 + p_2 \neq 0 \mod M, \quad \text{and} \quad p_1 - p_2 \neq 0 \mod M\,. \tag{4.2}$$

The lower bound follows from the fact that the permutation shifts of the $\binom{c}{2}$ 4-cycles must satisfy (4.1) and (4.2), and by using the *pigeonhole principle*. ■

We note that in [80], to remove the LETSs in the range $a \leq 8, b \leq 3$, the authors focused on removing the $(5, 3)$ and $(7, 3)$ LETSs, and demonstrated that those LETSs will be removed if all the 8-cycles that contain two check nodes within a given row block are avoided. This condition is different, and appears to be more restrictive, than the one established in Theorem 8. In fact, the lower bound $M \geq 2c(c-1)+1$ of [80] is larger than our lower bound in Theorem 9, and as can be seen in Section 4.5, we are able to find codes whose block lengths are smaller than the lower bound of [80].

4.4 Constraints on Eight-cycles for QC-LDPC Codes with $d_v = 4$

For a QC-LDPC code lifted from a fully-connected $4 \times c$ base graph, any of the TBC walks shown in Fig. 3.2 can result in an 8-cycle in the lifted graph. An important observation, however, is that if we assume that $g = 8$ and that all 8-cycles of the code are LSS-nE, then LETSs within a large range of a and b values will be eliminated. This is demonstrated in Table 4.1, where for the range of values $a \leq 11$ and $b \leq 8$, each entry shows the percentage of the eliminated structures within the corresponding (a, b) class. (The dash notation in the table is used for classes that do not exist in codes with $d_v = 4$ and $g = 8$.) As can be seen from Table 4.1, all LETSs within the union of the following ranges are eliminated: $\{a \leq 11, b \leq 2\}$, $\{a \leq 10, b \leq 4\}$, $\{a \leq 9, b \leq 6\}$, $\{4 < a \leq 7, b \leq 8\}$. Next, we present a constraint on 8-cycles that can be easily implemented/verified, and results in all 8-cycles to be LSS-nE.

Lemma 10. *In a QC-LDPC code with $g = 8$, lifted from the fully-connected $4 \times c$ base graph, every 8-cycle is LSS-nE, if the four degree-2 check nodes in each 8-cycle of the code belong to four different row blocks of the parity-check matrix.*

Table 4.1: Percentage of eliminated non-isomorphic LETS structures for each (a, b) class, within the range $a \leq 11, b \leq 8$, when all instances of 8-cycles are LSS-nE ($d_v = 4$ and $g = 8$).

	$a = 4$	$a = 5$	$a = 6$	$a = 7$	$a = 8$	$a = 9$	$a = 10$	$a = 11$
b = 0	–	–	–	–	100.00	–	100.00	100.00
b = 1	–	–	–	–	–	–	–	–
b = 2	–	–	–	–	100.00	100.00	100.00	100.00
b = 3	–	–	–	–	–	–	–	–
b = 4	–	–	–	100.00	100.00	100.00	100.00	99.39
b = 5	–	–	–	–	–	–	–	–
b = 6	–	–	100.00	100.00	100.00	100.00	98.19	98.01
b = 7	–	–	–	–	–	–	–	–
b = 8	8-cycles	100.00	100.00	100.00	85.71	94.00	94.05	93.16

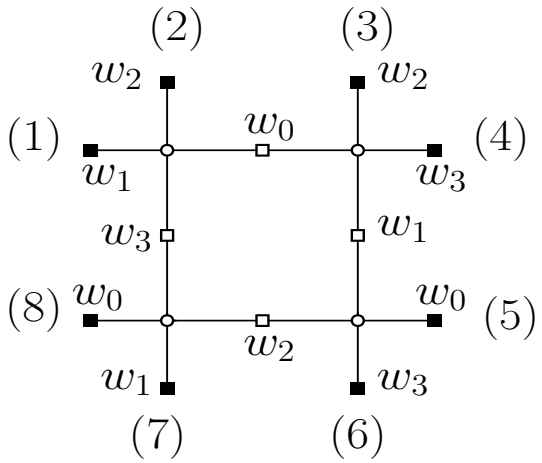


Figure 4.3: An 8-cycle with distinct degree-2 check node indices.

Proof. A general configuration of an 8-cycle with four distinct check node indices is shown in Fig. 4.3. One can see that connecting any pair of degree-1 check nodes through a new variable node either contradicts the girth requirement or implies the existence of an 8-cycle with at least two check nodes having the same (row block) index. To see this, without loss of generality, assume that one of the degree-1 check nodes is the one labeled "1". Since the new variable node can only be connected to check nodes with different indices, the possibilities for the other check node are all the other degree-1 check nodes except for the one labeled "7". Connecting the check node "1" to the one with label "2" creates a 4-cycle, while connecting it to the check nodes labeled "3", "4", or "8" creates a 6-cycle. Finally, connecting the check node

"1" to either one of the check nodes "5" or "6" results in 8-cycles with some repeated indices. ■

Theorem 10. *For any QC-LDPC code with $g = 8$ constructed by lifting the $4 \times c$ fully connected base graph, a necessary condition to avoid 8-cycles with repeated degree-2 check node indices is that the lifting degree M satisfies $M \geq 3c(c-1)+1$.*

Proof. Referring to Fig. 3.2, it is evident that when $d_v = 4$, none of the TBC walks of Figs. 3.2(b), (c) or (d) can result in an 8-cycle with repeated degree-2 check node indices. The common feature of all the other TBC walks in Fig. 3.2 is that they all involve two 4-cycles that share at least one check node. We first count the number of such 4-cycles. In a $\gamma \times c$ fully-connected base graph, there are $\binom{\gamma}{2}\binom{c}{2} = \frac{c\gamma(c-1)(\gamma-1)}{4}$ distinct 4-cycles. If we remove one check node from the base graph, there remain $\binom{\gamma-1}{2}\binom{c}{2} = \frac{c(\gamma-1)(c-1)(\gamma-2)}{4}$ 4-cycles. This implies that each check node is contained in $c(c-1)(\gamma-1)/2$, which is the difference between the two numbers, 4-cycles. This is the number of 4-cycles that share at least one check node. When $\gamma = 4$, this number is $3c(c-1)/2$. Now, to satisfy the condition of the theorem, the permutation shifts p_1 and p_2 of any pair of such 4-cycles must satisfy (4.2). In addition, to satisfy the girth constraint and to make sure that 8-cycles resulting from the double-traversal of 4-cycles do not occur, one needs to have the inequalities of (4.1) satisfied for the permutation shift p of any 4-cycle. Based on these and by using the pigeonhole principle, one can derive the lower bound. ■

4.5 Code Constructions and Numerical Results

Since [80] is the closest paper in the literature to our work in this chapter, we first construct codes for the same base graphs as those considered in [80], i.e., fully-connected 3×5, 3×6 and 3×10 base graphs. We note that based on the design approach of [80], to remove LETSs within the range $\mathcal{R} : a \leq 8, b \leq 3$, the minimum lifting degree is 41, 61 and 181, for the three base graphs, respectively. In [80], the authors found codes with lifting degrees 41, 63 and 251, for the three cases, respectively. In comparison, using the technique proposed in our work, for the same base graphs, to remove trapping sets in the same range $\mathcal{R}$, the lower bound of Theorem 9 is 21, 31 and 91, respectively. Using a greedy column by column search of the exponent matrix [81], we were able to find codes with lifting degrees 27, 41 and 165, for the

three base graphs, respectively. (We note that the gap between these values and the lower bounds of Theorem 9 is mainly due to the fact that the bound of Theorem 9 is derived only as a necessary condition for the elimination of Type-A 8-cycles. The derivation of the bound does not take into account the other constraint of the design, i.e., Type-B 8-cycles need to be LSS-nE.) These codes are significantly shorter than those of [80]. In particular, the block lengths of these codes surpass the lower bound of the design technique proposed in [80] by a large margin. The exponent matrices of the codes designed here are as follows:

$$P_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 14 & 21 & 22 & 25 \\ 0 & 1 & 5 & 7 & 19 \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 8 & 12 & 30 \\ 0 & 4 & 9 & 21 & 34 & 35 \end{bmatrix},$$

$$P_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 9 & 20 & 37 & 53 & 66 & 88 & 102 & 147 & 162 \\ 0 & 3 & 1 & 113 & 94 & 15 & 7 & 63 & 40 & 45 \end{bmatrix}. \tag{4.3}$$

As the second experiment, we consider the design of a girth-8 QC-LDPC code lifted from the fully-connected 4×5 base graph. If we use the design technique described in Section 4.4 for the removal of small trapping sets, based on the result of Theorem 10, the minimum possible lifting degree of such a code is $M = 61$. In fact, we were able to design a QC-LDPC code with this minimum lifting degree, and with the following exponent matrix:

$$P_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 3 & 21 & 55 \\ 0 & 5 & 15 & 44 & 31 \\ 0 & 57 & 49 & 38 & 24 \end{bmatrix}. \tag{4.4}$$

To design this code ($\mathcal{C}_4$), for the first three rows of the exponent matrix, we used the exponent matrix of a girth-10 QC-LDPC code with $M = 61$, designed in [81] by lifting the fully-connected 3×5 base graph. We then searched for the elements of the last row to satisfy the constraint that no 4-cycle or 6-cycle exists in the Tanner graph, and the constraint that there is no 8-cycle with repeated degree-2 check node indices.

The multiplicities of (a, b) LETSs of $\mathcal{C}_4$ in the range of $a \leq 11$ and $b \leq 8$ are shown in Table 4.2. As is evident from Table 4.2, all the LETSs in this range, except for the 8-cycles, are removed from the Tanner graph. For comparison, we have also provided in Table 4.2, the multiplicities of LETSs of a similar QC-LDPC code designed in [94]. This code also has girth 8 and is a cyclic lifting of the fully-connected 4×5 base graph with $M = 61$. The comparison shows a clear advantage of the designed code over the code of [94] in terms of the LETS distribution.

To demonstrate the superior performance of the designed codes, in Fig. 4.4, we have provided the frame error rate (FER) of $\mathcal{C}_1$-$\mathcal{C}_4$, decoded by a 5-bit min-sum algorithm [97] with clipping threshold 7.5 and maximum iteration number 50 over the binary-input (BI) AWGN channel. For comparison with $\mathcal{C}_1$-$\mathcal{C}_3$, we have also given the FER of QC-PEG codes [55] with similar base graphs and lifting degrees in Fig. 4.4. The FER of the code of [94] is provided for comparison with $\mathcal{C}_4$. As can be seen in Fig. 4.4, the codes constructed here handily outperform the existing codes in the error floor region.

To further evaluate the codes constructed in this work, one can compare their performance to finite-length capacity results of [70]. In [70] it has been demonstrated that the maximal coding rate achievable with error probability ϵ could be approximated by $C - \sqrt{\frac{V}{n}} Q^{-1}(\epsilon)$ where C is the channel capacity, V refers to the *channel dispersion* as a characteristic of the channel, and Q is the complementary Gaussian cumulative distribution function. As an example, we consider code $\mathcal{C}_3$ with block length 1650.

Table 4.2: Multiplicities of LETSs in the range of $a \leq 11$ and $b \leq 8$ for $\mathcal{C}_4$ and the code of [94].

LETS	$\mathcal{C}_4$	Code of [94]
$(4, 8)$	2135	2440
$(5, 8)$	0	610
$(10, 8)$	0	305

For such a block length and the block error rate of $\epsilon = 10^{-5}$, the smallest E_b/N_0 for the code rate of 0.7 is 2.65 dB. On the other hand, referring to Fig. 4.4 one can verify that C_3 has the code rate of 0.7 and obtains $FER = 10^{-5}$ at $E_b/N_0 = 3.3$ dB. Thus, our designed code C_3 has a gap of 0.65 dB to the finite-length capacity. It is worth noting that the gap to capacity could be further improved through an optimization over the degree distribution of the designed codes.

Finally, as an indication of the complexity of the proposed technique, we note that it took the search algorithm 2127, 7190, 24485, and 21173 seconds to find Codes $\mathcal{C}_1$-$\mathcal{C}_4$, respectively (given the corresponding lifting degrees). The search algorithm was implemented in MATLAB and ran on a desktop computer with 4 GHz CPU and 32 GB RAM. In general, the search complexity reduces with the increase in lifting degree. For example, for the 3×5 base graph, the run-time to find a code with $M = 100$ and free of LETSs in the range of $a \leq 8, b \leq 3$, was only about 2 sec. (compared to 2127 sec. for $M = 27$).

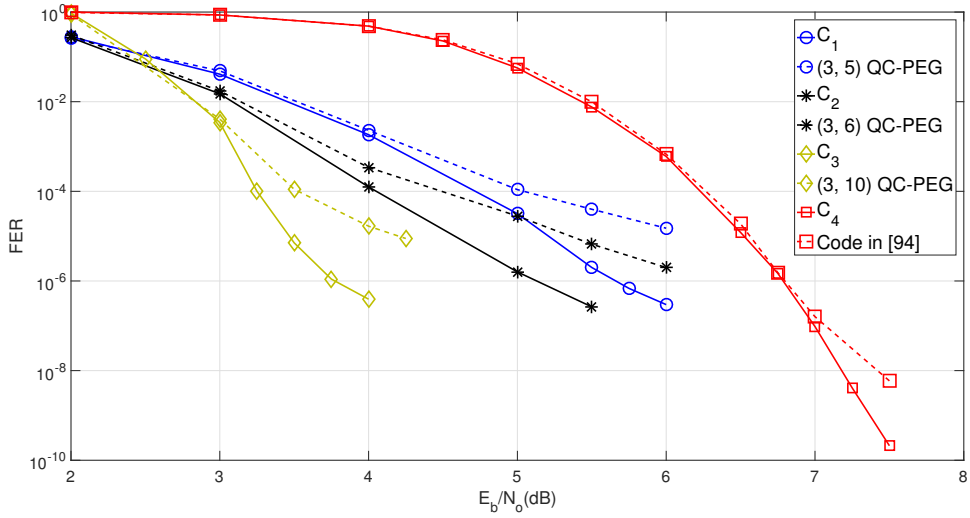


Figure 4.4: FER comparison of $\mathcal{C}_1$-$\mathcal{C}_4$ with some existing codes over the BI-AWGN channel.

Chapter 5

Simple Conditions for the Construction of Time-invariant SC-LDPC Codes Free of Small Trapping Sets

5.1 Introduction

So far, in Chapter 4, we developed a technique for the construction of QC-LDPC codes free of small LETSs. In this chapter, we extend our work for the design of variable-regular time-invariant spatially-coupled LDPC codes with small constraint length and low error floor. The proposed technique reduces the error floor by imposing simple constraints on the short cycles in the code's Tanner graph, which in turn, result in the elimination of the most dominant trapping sets of the code. We also derive lower bounds on the syndrome former memory for satisfying such constraints.

Most of the literature on the construction of SC-LDPC codes is devoted to codes with small variable node degrees such as $d_v = 3$ and $d_v = 4$ [78], [65], [26], [62], [24]. The advantage of lower node degrees is the lower decoding complexity and memory requirements. Moreover, a vast majority of the SC codes in the literature have girths 6 or 8 [24, 26, 62]. Such values of girth avoid cycles of length 4 that are known to be the most detrimental to the performance of iterative decoding algorithms. In this work, we consider codes with $d_v = 3, 4, 5$, and girth $g = 6, 8$. Thus, for a given degree distribution, rate and girth, we design variable-regular time-invariant SC-LDPC codes with a small constraint length and low error floor.

Our design technique has advantages over the existing work in terms of both complexity of the design and the performance of designed codes. In particular, compared

to the codes designed in the most recent publication [74], our designed codes have significantly lower constraint length for elimination of the same set or a larger set of targeted TSs. Furthermore, many of our designed codes with $g = 6$ and $d_v = 4$ and 5 have a smaller memory m and constraint length ν_s, and a better error floor performance compared to the codes in [7], [65], which have the same degree distribution, and rate but a larger girth of $g = 8$. Even for the case of $d_v = 3$, where the designed codes here have a larger memory than those in [7], [65], we demonstrate that for similar decoding complexity and latency, our designed codes can significantly outperform their counterparts, in the error floor region.

The rest of this chapter is organized as follows: In Section 5.2, we provide some preliminaries related to our design approach. The proposed techniques for the construction of SC-LDPC codes with $d_v = 3, 4$ and 5 are presented in Sections 5.3, 5.4 and 5.5, respectively. This is followed by design examples and extensive numerical results in Section 5.6.

5.2 Preliminaries

As already mentioned, we want to design time-invariant SC-LDPC codes with $d_v = 3, 4$, and $g = 6, 8$, having small constraint lengths. The parity check matrix for our designed code has the terminated form of (2.10), and thus, our goal is to design an optimized exponent matrix P based on our proposed simple conditions.

Corresponding to a $\gamma \times c$ exponent matrix P with $0 \leq p_{ij} \leq m$, for any two distinct columns $j_1, j_2 \in \{1, \ldots, c\}$ and any row $i \in \{1, \ldots, \gamma\}$ of P matrix, we define the *differential parameter* $\Delta P_i(j_1, j_2)$ as $p_{i,j_1} - p_{i,j_2}$. The following result is well-known, see, e.g., [7, 28]

Lemma 11. *Consider a $\gamma \times c$ exponent matrix P with the sequence of entries $p_{i_1 j_1}, p_{i_1 j_2}, p_{i_2 j_2}, p_{i_2 j_3}, \ldots, p_{i_{\ell/2} j_{\ell/2}}, p_{i_{\ell/2} j_1}$, where $i_1 \neq i_2, i_2 \neq i_3, \ldots, i_{\ell/2} \neq i_1$, and $j_1 \neq j_2, j_2 \neq j_3, \ldots, j_{\ell/2} \neq j_1$. This sequence corresponds to a cycle of length ℓ in the Tanner graph of the time-invariant SC-LDPC code if and only if* [1]

$$\Delta P_{i_1}(j_1, j_2) + \Delta P_{i_2}(j_2, j_3) + \ldots + \Delta P_{i_{\ell/2}}(j_{\ell/2}, j_1) = 0\,. \tag{5.1}$$

[1] One should note that some of the cycles corresponding to (5.1) are broken at the two boundaries of the Tanner graph, as a result of the termination. In particular, for cycles of length ℓ, the coupling length L needs to satisfy $L \geq \lfloor \ell/4 \rfloor m + 1$ to guarantee the existence of at least one such cycle [24].

We note that the sequence of entries in Lemma 11 corresponds to a closed walk with horizontal and vertical steps between the entries of P. For example, in Fig. 5.1, we have shown all such closed walks of length 8.[2] These closed walks are mapped to 8-cycles in the Tanner graph of the corresponding SC code if (5.1) is satisfied. In Fig. 5.1, we have only shown the submatrix of P that is involved in the closed walk. For each submatrix, the rows and columns are identified by i and j indices, respectively.

Definition 4. *Consider Fig. 5.1, and let $\mathcal{W}$ denote the closed walk in the exponent matrix P corresponding to an 8-cycle $\mathcal{C}$. If the four row indices of $\mathcal{W}$ belong to two (three, four, resp.) different rows of P, we refer to $\mathcal{C}$ as being* Type A *(*Type B*,* Type C*, resp.). In particular, the closed walks in Figs. 5.1(f), (g) and (i) correspond to 8-cycles of Type A, whereas, Type-B 8-cycles are resulted from the closed walks of Figs. 5.1(a), (c) and (h), and the closed walks of Figs. 5.1(b), (d), (e) correspond to Type-C 8-cycles.*

Example 5. *An example of a Type-A 8-cycle can be seen in Fig 5.2(a). This cycle consists of the nodes $u_0, w_1, u_2, w_2, u_1, w_1, u_3, w_2$. An example of a Type-B 8-cycle with nodes u_0, w_0, u_1, w_1, u_2, w_2, u_3, w_1 can be observed in Fig. 5.7(a).*

It should be noticed that in Chapter 4, we also distinguished different 8-cycles in terms of being Type-A or Type-B in the context of QC-LDPC codes.

5.3 Construction of SC-LDPC Codes with $d_v = 3$

5.3.1 SC-LDPC codes with $d_v = 3$ and $g = 6$

Proposition 3. *In an SC-LDPC code with $g = 6$ and $d_v = 3$, all LETSs within the range $\mathcal{R}_1 : \{a \leq 5, b \leq 2\}$, are eliminated if and only if all 6-cycles are LSS-nE.*

Proof. In a code with $d_v = 3$ and $g = 6$, there exist three different classes of LETS within the range $\mathcal{R}_1$: $(4, 0)$, $(4, 2)$ and $(5, 1)$ [32]. Classes $(4, 2)$ and $(4, 0)$, each has only one structure. These structures are LSS children of a 6-cycle, through the

[2]The same closed walks of length 8 were presented in Fig. 3 of [30] in the context of circulant-based SC-LDPC codes. They were also discussed earlier in [44] in the context of QC-LDPC block codes.

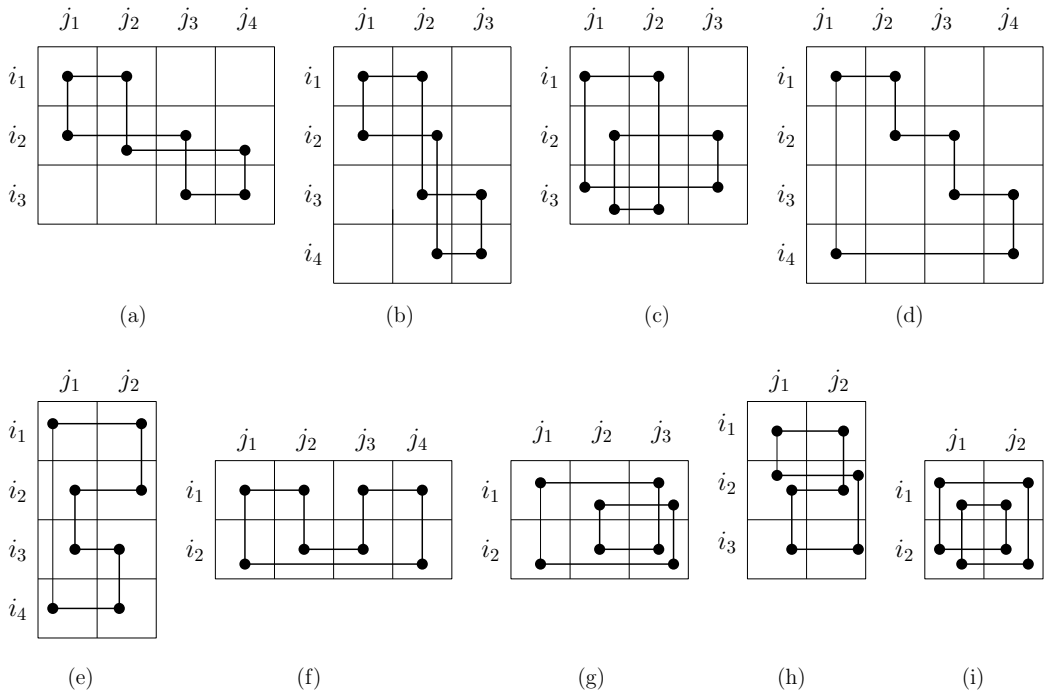


Figure 5.1: Closed walks of length 8 within the submatrices of P that can be mapped to 8-cycles in the SC-LDPC code's Tanner graph.

application of dot_2 and dot_3, respectively. There is also only one structure in the $(5,1)$ class, which is the LSS child of the $(4,2)$ LETS structure. The assumption of LSS-nE 6-cycles, thus, ensures the elimination of $(4,0)$, $(4,2)$ and $(5,1)$ LETSs. On the other hand, it is easy to show by contraposition that if all LETSs within the range $\mathcal{R}_1$ are eliminated, then all 6-cycles are LSS-nE. ■

To use the result of Proposition 3 in designing SC-LDPC codes that are free of LETSs in $\mathcal{R}_1$, one would need to create a parity-check matrix H_{SC}, corresponding to an exponent matrix P, with the minimum coupling length $L = m + 1$ [26], and first search for 6-cycles. Then, for each of the found cycles, one needs to check if the cycle can be expanded by either dot_2 or dot_3. In the following, we derive an equivalent constraint that can be tested by directly examining P, and is thus less complex.

Proposition 4. *In an SC-LDPC code with $g = 6$ and $d_v = 3$, all 6-cycles are LSS-nE if and only if all Type-A 8-cycles whose normal graphs have at least one chord, referred to as Type-A 8-cycles with chord, are avoided in the Tanner graph.*

Proof. We first assume Type-A 8-cycles with chord do not exist and prove that all 6-cycles are LSS-nE. Consider a 6-cycle that can be expanded by either dot_2 or dot_3

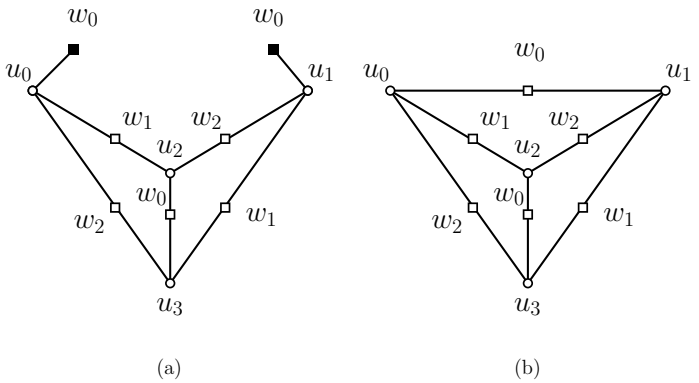


Figure 5.2: A 6-cycle $u_0, w_1, u_2, w_0, u_3, w_2$ expanded by (a) dot_2, and (b) dot_3, in a code with $d_v = 3$ and $g = 6$.

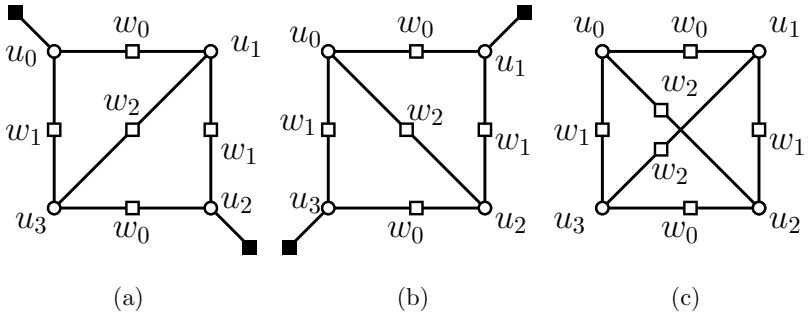


Figure 5.3: Different configurations of a Type-A 8-cycle $u_0, w_0, u_1, w_1, u_2, w_0, u_3, w_1$ with chord in a code with $d_v = 3$ and $g = 6$.

expansion. This implies that the Tanner graph has one of the two structures shown in Fig. 5.2. Both such structures contain a Type-A 8-cycle with chord, which is a contradiction. To prove if all 6-cycles are LSS-nE, then no Type-A 8-cycle with chord exists, we use contraposition. Suppose that a Type-A 8-cycle with chord exists and let $u_0, w_0, u_1, w_1, u_2, w_0, u_3, w_1$ be the sequence of the nodes of such a cycle. This cycle can thus have one of the configurations in Fig. 5.3(a)-(c). As can be seen, all such configurations have a 6-cycle that has been expanded by a *dot* expansion. ■

In the following, we derive a lower bound on m of an SC-LDPC code that is free of Type-A 8-cycles with chord.

Proposition 5. *Consider an SC-LDPC code with $d_v = 3$, $g = 6$, and memory m corresponding to a $3 \times c$ exponent matrix P. If this code has no Type-A 8-cycle with chord, then $m \geq \lceil \frac{(c-2)(c-3)+2}{8} \rceil$.*

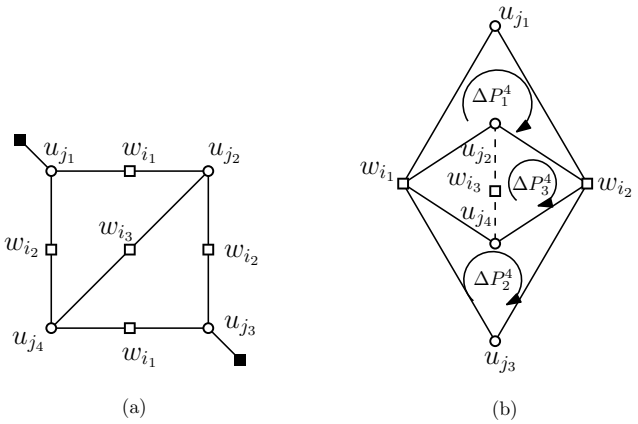


Figure 5.4: a) A possible structure for Type-A 8-cycles with chord based on the closed walk of Fig. 5.1(f). b) The closed walk of Fig. 5.1(f) corresponding to the 8-cycle of Fig. 5.4(a).

Proof. Referring to Fig. 5.1, one can see that for a code with $d_v = 3$, Type-A 8-cycles correspond to the closed walks in Figs. 5.1(f), (g) and (i). Furthermore, the existence of a chord requires the two variable nodes at the two ends of the chord to have different indices. This eliminates the closed walk of Fig. 5.1(i) as a possible candidate for a Type-A 8-cycle with chord. Now, consider the closed walk of Fig. 5.1(f). Fig. 5.4(a) shows one possible Type-A 8-cycle with chord that can be resulted from this closed walk. The closed walk of Fig. 5.1(f) corresponding to the 8-cycle of Fig. 5.4(a) is shown in Fig. 5.4(b) with its constituent closed walks of length 4. In this figure, the dashed line corresponds to the chord. Let ΔP^ℓ denote the sum of differential parameters given in (5.1) for a closed walk of length ℓ, and call it the differential parameter of the closed walk. In Fig. 5.4(b), three of the differential parameters for closed walks of length 4, denoted by ΔP_1^4, ΔP_2^4 and ΔP_3^4, are shown.

Consider a scenario where $\Delta P_1^4 + \Delta P_3^4 = 0$. This implies that the 6-cycle $u_{j_1}, w_{i_1}, u_{j_2}, w_{i_3}, u_{j_4}, w_{i_2}$ in Fig. 5.4(a) exists in the code's Tanner graph. (The assumption $\Delta P_1^4 + \Delta P_3^4 = 0$ is not imposing any additional constraint on the exponent matrix. This is because by the assumption $g = 6$ of the proposition, there is at least one cycle of length 6 in the Tanner graph.) With this constraint, the necessary condition for the Type-A 8-cycle with chord shown in Fig. 5.4(a) not to exist in the Tanner graph is to have $\Delta P_1^4 - \Delta P_2^4 \neq 0$. This is because if $\Delta P_1^4 = \Delta P_2^4$, then by $\Delta P_1^4 + \Delta P_3^4 = 0$, we have $\Delta P_2^4 + \Delta P_3^4 = 0$, which in turn implies that the 6-cycle $u_{j_2}, w_{i_2}, u_{j_3}, w_{i_1}, u_{j_4}, w_{i_3}$, and thus the Type-A 8-cycle with chord of Fig. 5.4(a) exist in the Tanner graph.

In general, since $p_{ij} \in \{0, 1, \ldots, m\}$, ΔP_1^4 has a value in the set $\{-2m, ..., -1, 0, 1, ..., 2m\}$. The constraint $g = 6$, however necessitates that $\Delta P_1^4 \neq 0$. This together with $\Delta P_1^4 \neq \Delta P_2^4$ limit the possible values of ΔP_1^4 to $4m - 1$ possibilities. On the other hand, for two fixed check node indices w_{i_1}, w_{i_2}, there are $\binom{c-2}{2}$ possible closed walks of length 4 that pass through these check nodes but do not pass through any of the two variable nodes involved in the closed walk corresponding to ΔP_2^4. Using the pigeonhole principle, to avoid Type-A 8-cycles with chord similar to the one shown in Fig. 5.4(a), we must thus have $4m - 1 \geq \binom{c-2}{2}$, or equivalently, $m \geq \lceil \frac{(c-2)(c-3)+2}{8} \rceil$. ■

In the following, we extend the range of elimination for LETSs.

Proposition 6. *In an SC-LDPC code with $g = 6$ and $d_v = 3$, a necessary and sufficient condition to eliminate all LETSs within the range $\mathcal{R}_2 : \{3 < a \leq 6, b \leq 3\} \cup \{a \leq 7, b \leq 2\}$ is to avoid all Type-A 8-cycles with chord and make all simple 8-cycles LSS-nE.*

Proof. There are 15 non-isomorphic structures within the range $\mathcal{R}_2$ for the Tanner graph of an (SC)-LDPC code with $d_v = 3$ and $g = 6$ [32]. The examination of these structures reveals that they are LSSs of either s_3 or s_4, where s_ℓ denotes a simple cycle of length 2ℓ. This together with Proposition 4 proves the sufficient condition. To prove the necessary condition, by contraposition, suppose that there exists at least one LSS expandable instance among s_4's or there exists at least one Type-A 8-cycle with chord. The latter, based on Proposition 4, implies that there exists at least one LSS expandable instance among s_3's. It is then easy to see that any LSS expansion of an s_4 or an s_3 will result in a LETS within the range $\mathcal{R}_2$. ■

In the following, we introduce a sufficient condition to further extend the range of LETS elimination in a code with $d_v = 3$ and $g = 6$. For this, we need the following definition.

Definition 5. *A TS $\mathcal{S}$ is twice-LSS expandable if $\mathcal{S}$ is expandable through two distinct variable nodes using dot expansions.*

As an example, Fig. 5.5 depicts a twice-LSS expandable 10-cycle, and two possible scenarios for its expansion through two variable nodes u_1 and u_2 using dot_2 expansions. Consider a TS $\mathcal{S}$ in a Tanner graph G, where G has a set of variable nodes U, and a set of check nodes W. In order to identify if $\mathcal{S}$ is twice-LSS expandable, one can

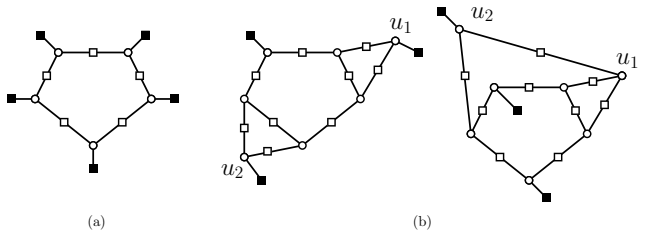
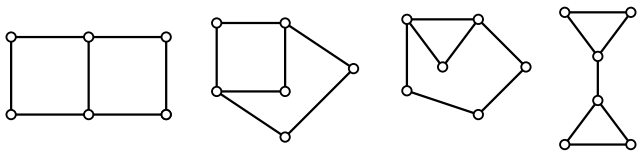


Figure 5.5: (a) A twice-LSS expandable 10-cycle in a code with $d_v = 3$ and $g = 6$, (b) two scenarios of the expansion through two variable nodes u_1 and u_2 using dot_2 expansions.

consider the multiset $\mathcal{M}$ whose elements are the variable nodes from the set $U \backslash \mathcal{S}$ that are connected to the odd-degree check nodes of $G(\mathcal{S})$, i.e., $\mathcal{M}$ is the union of the neighborhoods of the nodes in $\Gamma_o(\mathcal{S})$, excluding the variable nodes of $\mathcal{S}$, allowing for multiplicity. (Notation $\Gamma_o(\mathcal{S})$ is used to denote the odd-degree check nodes of $\mathcal{S}$.) If there are two variable nodes, each repeated at least twice in $\mathcal{M}$, or if there is one repeated variable node in $\mathcal{M}$, sharing a check node $w \in W \setminus \Gamma(\mathcal{S})$ with another variable node from $\mathcal{M}$, then $\mathcal{S}$ is twice-LSS expandable. (Notation $\Gamma(\mathcal{S})$ is used to denote the check nodes of $\mathcal{S}$.)

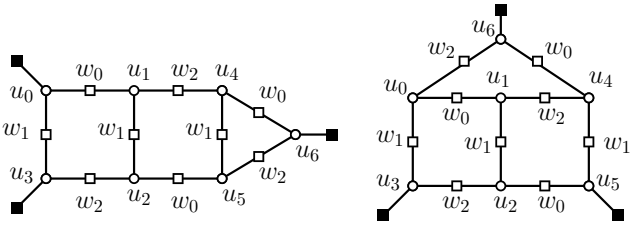
Proposition 7. *A sufficient condition for an SC-LDPC code with $g = 6$ and $d_v = 3$, to be free of all LETSs within the range $\mathcal{R}_3 : \{3 < a \leq 8, b \leq 3\}$ is to avoid all Type-A 8-cycles and to make all simple 8-cycles LSS-nE and all simple 10-cycles twice-LSS-nE.*

Proof. Comparing the range $\mathcal{R}_3$ with the smaller range $\mathcal{R}_2$, one can verify that $\mathcal{R}_3$ includes three extra classes of LETSs: $(7, 3), (8, 0)$, and $(8, 2)$. Since the constraints in this proposition satisfy all the constraints of Proposition 6, we know that all the LETSs within $\mathcal{R}_2$ are removed. We thus focus on the above three classes and show that, under the conditions of the proposition, none of the structures within these three classes can exist in the code. A careful inspection of the structures within the three classes reveals that they are all LSS children (direct or indirect) of $(6, 4)$ LETSs. In particular, all the structures within the $(7, 3)$ class are direct LSS children of $(6, 4)$ structures by dot_2 expansions, and those in $(8, 0)$ and $(8, 2)$ classes are in turn the LSS children of $(7, 3)$ structures. To prove the proposition, we thus show that all LETS structures in the $(6, 4)$ class are dot_2-nE. For a code with $d_v = 3$ and $g = 6$, we have 4 different non-isomorphic structures in the $(6, 4)$ class. We depict normal



(a) (b) (c) (d)

Figure 5.6: Normal graphs of all non-isomorphic LETS structures in the $(6, 4)$ class in a code with $d_v = 3$ and $g = 6$.



(a) (b)

Figure 5.7: Different configurations of LSS children for the $(6, 4)$ LETS structure in Fig. 5.6(a), in a code with $d_v = 3$ and $g = 6$.

graphs of these structures in Fig. 5.6. Considering all the possible dot_2 expansions of the structures of Fig. 5.6, one can verify that each one of them, regardless of variable and check node labelings, violates at least one of the conditions of the proposition. For example, Fig. 5.7 shows two non-isomorphic children of the structure of Fig. 5.6(a) by the dot_2 expansion. As can be seen, Fig. 5.7(a) includes an LSS-expandable simple 8-cycle, i.e., $u_1, w_2, u_4, w_1, u_5, w_0, u_2, w_1$, and Fig. 5.7(b) contains a Type-A 8-cycle, i.e., $u_0, w_2, u_6, w_0, u_4, w_2, u_1, w_0$.

As another example, three non-isomorphic dot_2 children of the $(6, 4)$ structure of Fig. 5.6(b) are shown in Fig. 5.8. One can verify that the structure in Fig. 5.8(a) has one Type-A 8-cycle, the structure in Fig. 5.8(b) contains an LSS-expandable 8-cycle, and the last structure in Fig. 5.8(c) contains a twice-LSS-expandable 10-cycle.

■

5.3.2 SC-LDPC codes with $d_v = 3$ and $g = 8$

The constraint $g = 8$ automatically removes all the LETSs within the range $\mathcal{R}_1$, described in Proposition 3. For the range $\mathcal{R}_2$, from Proposition 6, the following result

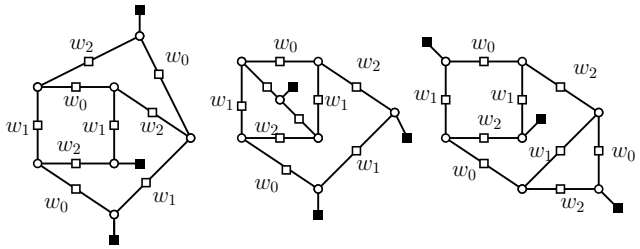


Figure 5.8: Different configurations of LSS children for the $(6, 4)$ LETS structure in Fig. 5.6(b), in a code with $d_v = 3$ and $g = 6$.

follows considering that for a code with $g = 8$, no Type-A 8-cycle with chord exists.

Corollary 5. *In an SC-LDPC code with $g = 8$ and $d_v = 3$, if all 8-cycles are LSS-nE, then the code is free of LETSs within the range $\mathcal{R}_2 \cup (3, 3)$.*[3]

Proposition 8. *In an SC-LDPC code with $g = 8$ and $d_v = 3$, if there is no Type-A 8-cycle and if all Type-B 8-cycles are LSS-nE, then the code is free of LETSs within the range $\mathcal{R}_3 : \{a \leq 8, b \leq 3\}$.*

Proof. The proof is similar to that of Theorem 8. ∎

In the following, we derive a lower bound on the memory size of an SC-LDPC code with no Type-A 8-cycles. This serves as a lower bound on m if one uses the result of Proposition 8 to design an SC-LDPC code.

Proposition 9. *Consider an SC-LDPC code with $g = 8$, $d_v = 3$, and memory m corresponding to a $3 \times c$ exponent matrix P. If this code has no Type-A 8-cycle, then $m \geq \left\lceil \frac{c(c-1)}{4} \right\rceil$.*

Proof. Type-A 8-cycles in the Tanner graph of an SC code with $d_v = 3$ correspond to the closed walks of Figs. 5.1(f), (g) or (i). Let ΔP^{ℓ} denote the sum of differential parameters given in (5.1) for a closed walk of length ℓ, and call it the differential parameter of the closed walk. For the code not to have any Type-A 8-cycle, we must have $\Delta P^8 \neq 0$, for all closed walks of Figs. 5.1(f), (g) or (i) in the code. For the walk of Fig. 5.1(i), we have $\Delta P^8 = 2\Delta P^4$, where ΔP^4 corresponds to the constituent closed walk of length 4. To satisfy the girth constraint and to eliminate 8-cycles of this type, we must then have $\Delta P^4 \neq 0$. Both closed walks of Figs. 5.1(f) and (g)

[3]The fact that the code is free of $(3, 3)$ LETSs follows from $g = 8$ and that $(3, 3)$ LETSs are 6-cycles.

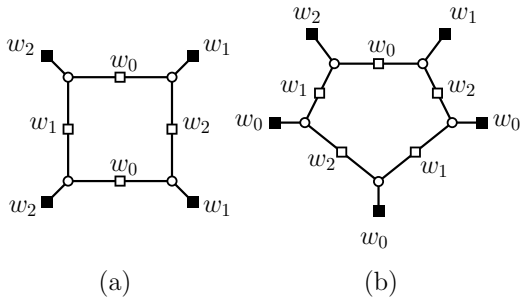


Figure 5.9: Configurations for (a) Type-B(w_0) 8-cycle, (b) 10-cycle(w_0) in a code with $d_v = 3$.

involve two closed walks of length 4 that have two rows of P in common. If we denote the differential parameters of these closed walks by ΔP_1^4 and ΔP_2^4, to satisfy the girth constraint and to eliminate such 8-cycles, we must have

$$\Delta P_1^4 \neq 0, \Delta P_2^4 \neq 0, \Delta P_1^4 \neq \Delta P_2^4, \Delta P_1^4 \neq -\Delta P_2^4 \,. \tag{5.2}$$

For a $3 \times c$ exponent matrix P, there exist $\binom{c}{2}$ possible closed walks of length 4 containing two (given) distinct rows. On the other hand, in general, since $p_{ij} \in [0, m]$, then ΔP^4 has a value in the set $\{-2m, ..., -1, 0, 1, ..., 2m\}$. Thus, to satisfy (5.2) for the $\binom{c}{2}$ closed walks, based on the pigeonhole principle, we need to have $2m \geq \binom{c}{2}$ or equivalently $m \geq \left\lceil \frac{c(c-1)}{4} \right\rceil$. ■

We note that the general method of proof used above is similar to the one used in [7] to prove Lemma 6 of [7]. Similar techniques were also used earlier in [44] to derive bounds on the lifting degree of protograph-based QC-LDPC codes with a certain girth.

In the following, we extend the elimination range of LETSs beyond $\mathcal{R}_3$, for codes with $d_v = 3$ and $g = 8$, by imposing further constraints on cycles of length 8 and 10. Based on the definition of Type-B 8-cycles, when $d_v = 3$, two of the degree-2 check nodes in the cycle correspond to the same row of P. If that row has index i, then we use the terminology Type-B(i) for the 8-cycle. Similarly, for a 10-cycle, when $d_v = 3$, one out of 5 check nodes corresponds to a unique row of P (each of the other 4 check nodes correspond to a row of P also shared by another check node). If that row has index i, then the 10-cycle is referred to as 10-cycle(i). Fig. 5.9 shows examples of a Type-B(w_0) 8-cycle and a 10-cycle(w_0).

Proposition 10. *In an SC-LDPC code with $d_v = 3$ and $g = 8$, the following conditions ensure the absence of all LETS structures within the range $a \leq 10, b \leq 3$, except one structure (out of 17) in the $(9, 3)$ class, one structure (out of 6) in the $(10, 0)$ class, and two non-isomorphic structures (out of 28) in the $(10, 2)$ class:*

- *i) Avoid all Type-A 8-cycles, and for $w_0 \in \{1, 2, 3\}$, avoid all Type-B(w_0) 8-cycles.*
- *ii) For $w_1, w_2 \in \{1, 2, 3\} \setminus \{w_0\}$ and $w_1 \neq w_2$, avoid all 10-cycle(w_1) and 10-cycle(w_2).*
- *iii) Make all 8-cycles and 10-cycles LSS-nE.*

Proof. The proof is given in Appendix D. ■

5.4 Construction of SC-LDPC Codes with $d_v = 4$

5.4.1 SC-LDPC codes with $d_v = 4$ and $g = 6$

Proposition 11. *An SC-LDPC code with $d_v = 4$ and $g = 6$ is free of all LETSs within the range $\mathcal{R}_4 : \{a \leq 6, b \leq 5\} \cup \{a \leq 7, b \leq 3\}$ if there exists no Type-A 8-cycle with chord and if all simple 8-cycles are dot_m-nE, where $m \in \{3, 4\}$.*

Proof. In general, there exist 27 non-isomorphic LETS structures within different classes in the range $\mathcal{R}_4$ for a code with $d_v = 4$ and $g = 6$. Normal graph representation of all these structures along with their parent/child relationships through dot_m expansions are shown in Fig. 5.10. In Fig. 5.10, we have also shown the parent structures in $(3, 6)$, $(4, 8)$ and $(5, 6)$ classes outside the range. From the parent/child relationships, it is clear that if the $(4, 4)$ structure is eliminated and the simple 8-cycles are dot_m-nE for $m = 3$ and $m = 4$, then all the structures in $\mathcal{R}_4$ will be eliminated. In the following, we demonstrate that the $(4, 4)$ structure includes a Type-A 8-cycle with chord and thus if the conditions of the proposition are met, all the LETSs in $\mathcal{R}_4$ are eliminated.

The $(4, 4)$ structure is a dot_3 expansion of 6-cycles. An arbitrary 6-cycle is shown in Fig. 5.11(a). We note that the three degree-2 check nodes must have different row labels, which are arbitrarily chosen to be w_0, w_1 and w_2 in Fig. 5.11(a). It is

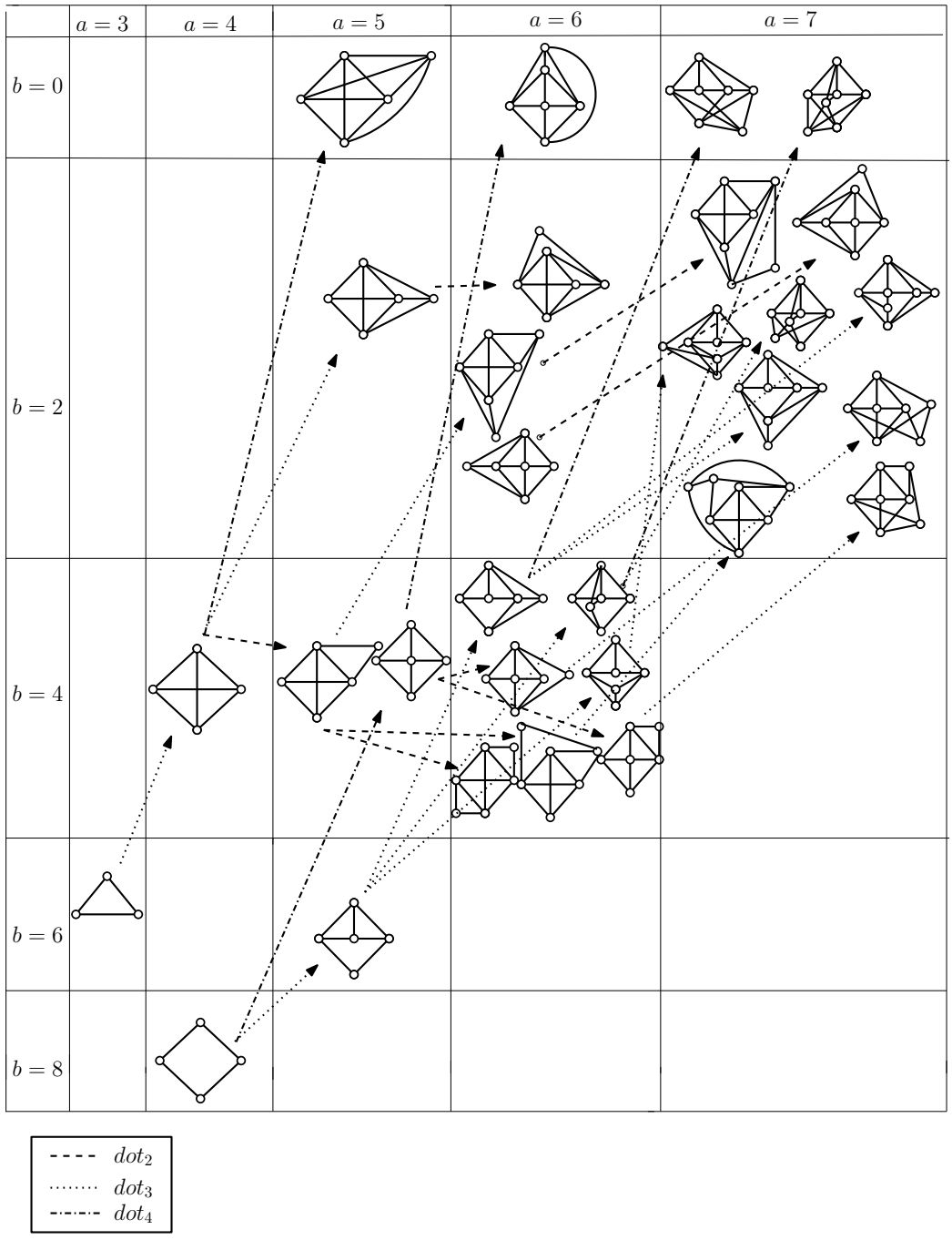


Figure 5.10: Normal graphs of all non-isomorphic LETS structures within the range $\mathcal{R}_4$ in a code with $d_v = 4$ and $g = 6$.

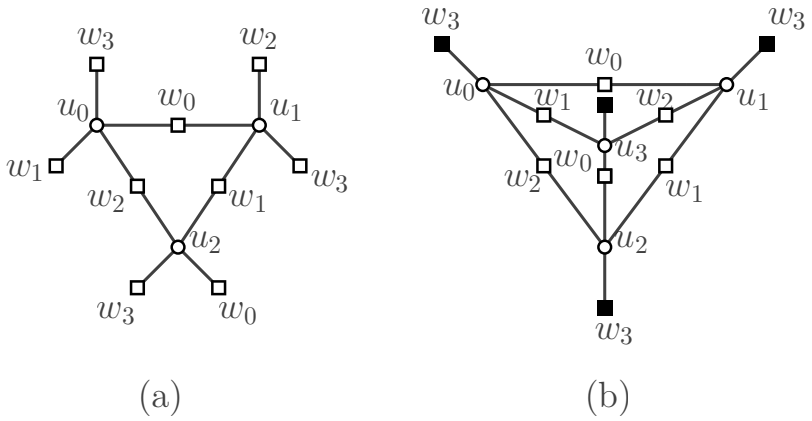


Figure 5.11: (a) A 6-cycle, and (b) one of its children through a dot_3 expansion in a code with $d_v = 4$ and $g = 6$.

easy to examine that any application of dot_3 expansion to the cycle of Fig. 5.11(a) results in at least one Type-A 8-cycle with chord. One such case is shown in Fig. 5.11(b), where there are three Type-A 8-cycles in the expanded graph, including $u_0, w_0, u_1, w_1, u_2, w_0, u_3, w_1$.

■

5.4.2 SC-LDPC codes with $d_v = 4$ and $g = 8$

For codes with $d_v = 4$, increasing the girth from 6 to 8 automatically eliminates all LETSs in the range $\mathcal{R}_4$. Moreover, for an (SC)-LDPC code with $d_v = 4$ and $g = 8$, if all 8-cycles are LSS-nE, then no LETS within the range $\mathcal{R}_5 : \{a \leq 11, b \leq 2\} \cup \{a \leq 10, b \leq 4\} \cup \{a \leq 9, b \leq 6\} \cup \{4 < a \leq 7, b \leq 8\}$ exists [64]. The following result provides a simple sufficient condition for all 8-cycles to be LSS-nE.

Proposition 12. *[64] In an SC-LDPC code with $d_v = 4$ and $g = 8$, every 8-cycle is LSS-nE if all 8-cycles of the code are Type-C.*

The following result provides a lower bound on the memory size of codes, for which all 8-cycles are Type-C.

Proposition 13. *For any SC-LDPC code with $d_v = 4$, $g = 8$, and memory m, constructed based on a $4 \times c$ exponent matrix, a necessary condition to avoid all Type-A and Type-B 8-cycles is to have $m \geq \left\lceil \frac{3c(c-1)}{4} \right\rceil$.*

Proof. Type-A and Type-B 8-cycles result from the closed walks shown in Figs. 5.1(a), (c), (f), (g), (h) and (i). An examination of the configurations in Figs. 5.1(a), (c), (f), (g), and (h) reveals that they all correspond to two closed walks of length 4 that share at least one row of the exponent matrix P. The number of such closed walks is equal to $\binom{\gamma}{2}\binom{c}{2} - \binom{\gamma-1}{2}\binom{c}{2}$, or $c(c-1)(\gamma-1)/2$, where the first and second terms in the subtraction are the number of possible closed walks of length 4 before and after excluding a row of P. For $\gamma = 4$, this number is $3c(c-1)/2$. Let ΔP_1^4 and ΔP_2^4 denote the differential parameters associated with the two closed walks just described. To avoid the formation of any Type-A and Type-B 8-cycles, ΔP_1^4 and ΔP_2^4 must satisfy (5.2), and thus $2m \geq \frac{3c(c-1)}{2}$, or equivalently $m \geq \left\lceil \frac{3c(c-1)}{4} \right\rceil$. ■

5.5 Construction of SC-LDPC Codes with $d_v = 5$

In this section, we discuss simple sufficient conditions for the design of SC-LDPC codes with $d_v = 5$ and $g = 6$ such that a range of potentially harmful trapping sets are eliminated.

Proposition 14. *An SC-LDPC code with $d_v = 5$ and $g = 6$ is free of all LETSs within the range $\mathcal{R}_6 : \{a \leq 8, b \leq 3\} \cup \{a \leq 7, b \leq 6\} \cup \{3 < a \leq 6, b \leq 9\}$ if there exists no Type-A 8-cycle, and no Type-B 8-cycle with chord.*

Proof. In general, in a code with $d_v = 5$ and $g = 6$, there exist 66 non-isomorphic LETS structures in the range $\mathcal{R}_6$. All such structures are generated as a result of dot_m expansions, $3 \leq m \leq 5$, of $(3, 9)$ and $(4, 10)$ LETS structures. In the following, we show that the sufficient conditions of the proposition make these structures dot_m-nE, for $3 \leq m \leq 5$.

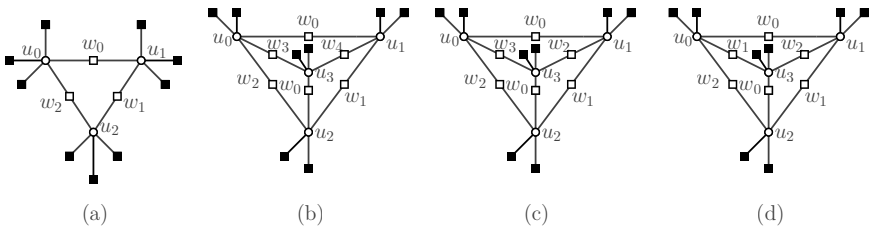


Figure 5.12: (a) A 6-cycle, and (b)-(d) three configurations of its children through a dot_3 expansion in a code with $d_v = 5$ and $g = 6$.

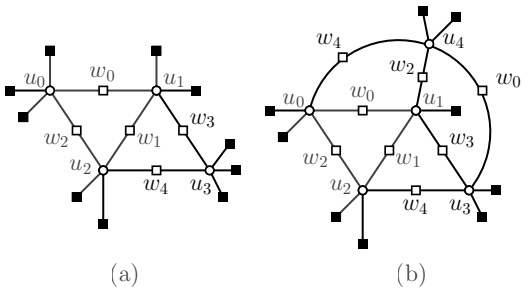


Figure 5.13: (a) A $(4, 10)$ LETS, and (b) one of its children through a dot_3 expansion in a code with $d_v = 5$ and $g = 6$.

The only structure in the $(3, 9)$ class is a 6-cycle in which all the check nodes have different row labels. One such cycle is shown in Fig. 5.12(a). The application of dot_4 or dot_5 to this structure creates 4-cycles, and is thus not possible in a code with $g = 6$. The application of dot_3 to this structure results in the creation of 3 additional check nodes of degree-2, where at least one has the same row label as one of the check nodes of the cycle (because there are only 5 possible row labels). Suppose that this repeated row label is w_0. There are then three possible configurations for the expanded structure, depending on whether one, two, or three row labels are repeated. Examples of such configurations are shown in Figs. 5.12(b)-(d), respectively. It is easy to see that each such configuration contains either a Type-A 8-cycle or a Type-B 8-cycle with chord, contradicting the conditions of the proposition.

The only structure of the $(4, 10)$ class is shown in Fig. 5.13(a). This structure contains an 8-cycle with a chord. We note that based on the conditions of the proposition, the 8-cycle must be of Type-C, and therefore, all the row labels of the degree-2 check nodes in this structure must be distinct. The application of a dot_3 expansion to this structure will create 3 more 8-cycles with chord. An example is shown in Fig. 5.13(b), where the three new 8-cycles with chord have the following variable nodes: $\{u_0, u_2, u_1, u_4\}$, $\{u_1, u_2, u_3, u_4\}$, and $\{u_0, u_1, u_3, u_4\}$. One can see that the three additional degree-2 check nodes must have different row labels, where each label is identical to one of the 5 already existing check node labels in the $(4, 10)$ structure. It then follows that the expanded structure contains either a Type-A 8-cycle or a Type-B 8-cycle with chord. To expand the $(4, 10)$ structure with a dot_4 expansion, while maintaining $g = 6$, one must have the new variable node connected to one degree-1 check node from each of the 4 variable nodes of the structure. The labels of these check nodes must be different, and thus each label will be identical to one

of the labels of the existing degree-2 check nodes of the $(4, 10)$ structure. It is then easy to see that the newly created structure contains either a Type-A 8-cycle or a Type-B 8-cycle with chord. Finally, the application of a dot_5 expansion to the $(4, 10)$ structure creates a 4-cycle and is thus not possible considering the girth constraint. ■

5.6 Code Constructions and Numerical Results

In this section, we construct SC-LDPC codes with small values of m satisfying the conditions of propositions in previous sections to eliminate different ranges of LETSs for a given degree distribution and girth g. For this, for a given value of m, we use a greedy column by column search of the exponent matrix such that at least one zero entry exists in each column [8]. For each column, if all possible values are exhausted with no success, we backtrack to the previous column and check the rest of eligible assignments for that column. If, for a given m, an exponent matrix satisfying the design constraints is not found within a certain period of time τ, we move on to the next m value. A pseudo-code of the search algorithm is given in Algorithm 1. In Algorithm 1, it is assumed that we start the search from a lower bound on m such as the one provided in Propositions 5, 9 or 13, and increase m until we find an exponent matrix that satisfies the girth constraint as well as the constraints imposed by the appropriate proposition. Algorithm 1 is implemented in MATLAB, and is run on a PC with 4-GHz CPU and 32-GB RAM. In all the reported results, τ is chosen to be 3 hours.

All the simulations reported in this section are performed over the binary-input AWGN channel, using a floating-point sliding window decoder based on belief propagation with a pair-wise check-node reduction (box-plus) and with maximum 50 iterations. When we compare the LETS distributions of different codes, for fairness, we normalize the multiplicity of LETS classes with respect to the block length N. This represents the average multiplicity of LETS classes per variable node (coded bit).

5.6.1 SC-LDPC codes with $d_v = 3$

In Table 5.1, we present the constructed codes with $d_v = 3, g = 6$, free of LETSs within the range $\mathcal{R}_1$ and $\mathcal{R}_2$, and compare them with similar codes in [74] that have

Table 5.1: Constructed SC-LDPC codes with $d_v = \gamma = 3$ and $g = 6$, free of LETSs within the range $\mathcal{R}_1$ and $\mathcal{R}_2$, in comparison with their counterparts in [74] that are free of LETSs in the range $\mathcal{R}_1$.

	range $\mathcal{R}_1$				range $\mathcal{R}_2$			Code of [74]
c	lower bound of m	m	P matrix	run-time (sec.)	m	P matrix	run-time (sec.)	m
6	2	4	2 0 0 4 2 4 1 4 1 0 0 4 0 1 4 4 1 0	840	9	1 1 0 9 8 0 0 9 6 0 0 9 8 0 5 3 1 4	404	17
7	3	5	2 3 0 4 0 0 5 5 0 1 2 4 5 0 0 5 4 0 5 1 2	247	12	0 4 0 12 1 2 8 11 0 7 3 11 0 10 2 9 12 0 0 11 0	1800	25
8	4	6	0 6 5 3 5 0 0 4 4 0 5 0 0 3 5 6 4 3 0 5 4 1 6 0	1560	18	16 15 0 13 18 17 0 0 6 0 8 13 0 0 16 18 0 8 7 0 13 18 13 14	3047	36
9	6	8	0 8 8 7 2 0 4 0 3 1 1 0 6 8 8 0 7 0 7 0 2 0 0 6 3 4 4	155	24	0 0 10 0 4 0 24 1 23 13 21 3 24 0 6 0 0 0 1 4 0 2 12 13 16 22 1	10760	56
10	8	10	8 3 0 7 0 6 1 0 8 5 1 9 8 5 3 0 8 10 0 0 0 0 8 0 10 9 0 0 4 10	21	29	5 28 16 2 0 29 0 21 0 14 0 24 23 0 11 0 15 15 27 6 4 0 0 25 21 20 26 0 29 0	9603	NA

the same degree distribution, and are free of LETSs within $\mathcal{R}_1$. In our search process, to find the codes that are free of LETSs in $\mathcal{R}_1$, we start the search from the lower bound of Proposition 5, also given in Table 5.1, and keep increasing m until we find a code that satisfies the constraints of Proposition 4. To find the codes that are free of LETSs in $\mathcal{R}_2$, we use the m values of the codes found in the first stage (that are free of LETSs in $\mathcal{R}_1$) as the starting point and increase m until the constraints of Proposition 6 are met.

The comparison of the codes designed here with those of [74] show that our constructed codes that are free of LETSs within the same range of $\mathcal{R}_1$ have a significantly lower m value and constraint length $\nu_s = (m+1)c$. Even when we increase the elimination range to $\mathcal{R}_2$, our designed codes are still superior to the codes of [74] in terms of m and ν_s.

In Table 5.2, we have presented the constructed codes with $d_v = 3, g = 8$, free of LETSs within the range $\mathcal{R}_3 \cup (3,3)$. To find the codes of Table 5.2, the initial value of m is selected to be the lower bound of Proposition 9, also given in Table 5.2. The value is increased until we find a code that satisfies the constraints of Proposition 8. Comparison with similar codes in [74], whose m values are given in Table 5.2, and are also designed to be free of LETSs within $\mathcal{R}_3 \cup (3,3)$, shows that the codes designed here have significantly lower m and ν_s compared to their counterparts in [74].

In both Tables 5.1 and 5.2, we have also provided the run-times of Algorithm 1 for

Table 5.2: Constructed SC-LDPC codes with $d_v = \gamma = 3$ and $g = 8$, free of LETSs within the range $\mathcal{R}_3 \cup (3, 3)$, in comparison with their counterparts in [74], that are also free of LETSs within $\mathcal{R}_3 \cup (3, 3)$.

c	lower bound of m	m	P matrix	run-time (sec.)	m in [74]
4	3	5	1 3 4 0 1 0 0 5 0 5 1 3	3	NA
5	5	8	3 4 7 0 8 2 0 4 7 0 0 8 0 2 4	10650	22
6	8	13	4 0 13 12 0 11 10 13 11 0 11 0 0 6 0 10 13 4	10448	35
7	11	21	0 0 0 0 14 21 19 13 1 19 2 0 0 7 2 10 1 20 12 17 0	10768	55

designing the exponent matrices given in the tables for the corresponding m values.

To compare the performance of our designed codes with those in [74], we consider codes with $d_v = \gamma = 3$, $d_c = c = 6$, and $g = 6$. The code designed in [74] with these parameters has $m = 17$. To have a fair comparison, we also consider the same value of m and using Proposition 6, construct $\mathcal{C}_1$, which is free of LETSs within $\mathcal{R}_2$, and has the following exponent matrix:

$$P_1 = \begin{bmatrix} 6 & 11 & 15 & 0 & 14 & 17 \\ 10 & 8 & 17 & 12 & 0 & 0 \\ 0 & 0 & 0 & 14 & 16 & 5 \end{bmatrix}. \tag{5.3}$$

Table 5.3 and Fig. 5.14 present the LETS distribution and BER performance of $\mathcal{C}_1$ in comparison with its counterpart from [74] for $L = 300$. Both codes have rate $R = 0.47$, and block length $N = 1800$, and are decoded by a decoder with a window size of $W = 5c(m + 1) = 540$ bits. The decoding complexity and latency for both codes are thus identical. As can be seen, $\mathcal{C}_1$ is significantly superior to its counterpart in terms of the LETS distribution and the error floor.

To compare our designed codes with the same degree distribution but different

Table 5.3: Normalized multiplicity of non-empty (a, b) LETS classes in the range of $a \leq 7$ and $b \leq 4$, for the designed code $\mathcal{C}_1$ compared to its counterpart in [74] ($L = 300$).

LETS	Code in [74]	$\mathcal{C}_1$
$(3, 3)$	0.96	0.79
$(4, 4)$	3.72	0.96
$(5, 3)$	2.41	0
$(6, 0)$	0.16	0
$(6, 2)$	0.48	0
$(6, 4)$	34.26	14.28
$(7, 3)$	6.36	0.45

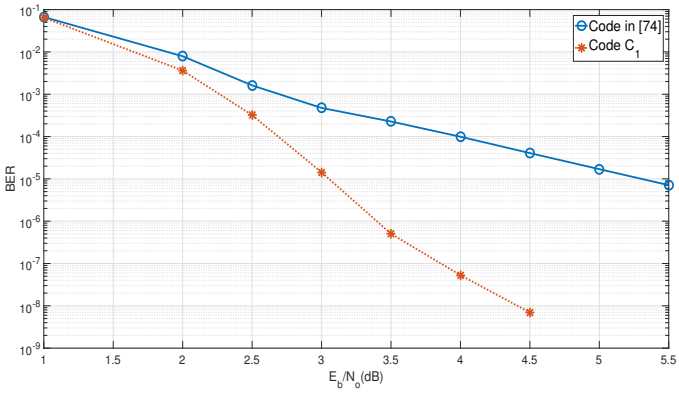


Figure 5.14: Performance comparison of designed code $\mathcal{C}_1$ with its counterpart in [74].

Table 5.4: SC-LDPC codes with $d_v = \gamma = 3$ and $g = 8$, free of LETSs within the range $\mathcal{R}_2 \cup (3, 3)$, constructed based on Corollary 5.

c	m	P matrix	run-time (sec.)
6	9	9 0 8 4 0 0 0 3 0 0 9 8 7 2 0 8 1 5	920
7	12	3 0 10 6 0 11 0 0 6 0 0 8 0 12 9 12 1 10 1 3 2	4516
8	18	2 0 12 13 18 0 4 0 0 6 12 0 0 17 0 18 15 7 16 0 2 9 18 12	10136
9	24	13 0 7 17 0 21 0 0 24 13 1 0 0 17 0 22 24 0 14 5 6 3 14 2 17 13 15	10184
10	29	0 25 0 25 10 22 0 0 0 0 26 25 3 0 0 0 7 5 29 24 9 6 25 27 8 16 24 18 4 14	4621

girth values, we consider the same LETS elimination ranges of $\mathcal{R}_2$ and $\mathcal{R}_3$, as considered in Tables 5.1 and 5.2, and design codes with $d_v = 3, g = 8$ and $d_v = 3, g = 6$, for those ranges, respectively. The designed codes are given in Tables 5.4 and 5.5, respectively. These codes are designed based on the constraints of Corollary 5 and Proposition 7, respectively. The run-times of Algorithm 1 to find the exponent matrices corresponding to different m values given in the tables are also reported in both tables.

The comparison between the constructed codes in Tables 5.1 and 5.4 shows that, for the same value of c, they all have the same value of m, and thus the same constraint length. We have listed the LETS distribution for each pair of the designed codes in Tables 5.1 and 5.4 (with the same degree distribution, rate and block length) in Table 5.6. The results demonstrate that, for all cases, the multiplicity of 8-cycles in the code with $g = 6$ is less than that of its counterpart with $g = 8$. Moreover, in all cases, except $c = 7$, the multiplicity of $(7, 3)$ LETSs is smaller in the code with $g = 6$ compared to the code with $g = 8$. The same applies to $(8, 2)$ LETSs except for the cases of $c = 7, 8$, and 10. These demonstrate the advantages of girth-6 codes in comparison with their girth-8 counterparts. Similar comparisons between the

Table 5.5: SC-LDPC codes with $d_v = \gamma = 3$ and $g = 6$, free of LETSs within the range $\mathcal{R}_3$, constructed based on Proposition 7.

c	m	P matrix	run-time (sec.)
4	6	4 6 0 1 5 1 6 0 0 0 4 6	2
5	9	5 0 3 4 9 0 9 7 9 1 9 7 0 0 0	1489
6	14	6 0 0 12 0 14 8 14 11 0 6 0 0 1 11 2 9 9	3610
7	23	19 13 0 0 18 20 0 23 0 10 23 0 9 13 0 1 19 17 21 0 23	9875

codes of Tables 5.2 and 5.5 show that girth-6 codes have slightly larger memory and constraint length compared to their girth-8 counterparts. This however, comes with the advantage of a superior LETS distribution, with the exception of $c = 6$, as shown in Table 5.7. For a fair comparison, in Table 5.7, the value of L in each case is selected such that the compared codes have the same rate. Table 5.7 also demonstrates that girth-6 codes have consistently a smaller number of 8-cycles compared to their girth-8 counterparts.

As an example for comparing the performances of similar codes with different girth values, in Fig. 5.15, we compare the BER of the two codes with $c = 4$ and $R = 0.21$ from Table 5.7. Both codes are decoded by a sliding window decoder with the window size of $5(m+1)c = 120$ bits, where $m = 5$ of the code with $g = 8$ is chosen in the equation. As can be seen, the girth-6 code outperforms its girth-8 counterpart in a wide range of SNR values, for the same decoding complexity and latency.

To further demonstrate the competitive performance of the codes with $g = 6$ designed here, we consider the code $\mathcal{C}_2$, with $d_v = 3, d_c = 6, g = 6, m = 9, L = 380$, and free of LETSs in $\mathcal{R}_2$, whose exponent matrix is given in the first row of Table 5.1. We then compare $\mathcal{C}_2$ with three codes with the same rate and degree distributions but with $g = 8$. The first two codes have $m = 7$, and have been designed in [7, 65].

Table 5.6: Normalized multiplicity of 8-cycles and non-empty (a, b) LETS classes in the range of $a \leq 8$ and $b \leq 3$, for the designed codes of the same rate in Table 5.1 (for $\mathcal{R}_2$) and Table 5.4. All codes have $L = 300$. (All LETSs in the $(4, 4)$ class are 8-cycles.)

	$c = 6, R = 0.485$		$c = 7, R = 0.55$		$c = 8, R = 0.60$		$c = 9, R = 0.53$		$c = 10, R = 0.67$	
LETS	$g = 8$	$g = 6$	$g = 8$	$g = 6$	$g = 8$	$g = 6$	$g = 8$	$g = 6$	$g = 8$	$g = 6$
$(3, 3)$	0	0.65	0	0.41	0	0.47	0	0.10	0	0.36
$(4, 4)$	7.95	4.88	9.55	8.43	11.33	9.52	14.72	11.97	15.05	13.72
$(7, 3)$	5.63	4.03	3.41	6.96	5.23	4.41	9.04	5.88	7.54	6.52
$(8, 2)$	1.45	0.96	0.40	1.91	0.57	0.58	1.33	0.92	1.10	1.26

Table 5.7: Normalized multiplicity of 8-cycles and non-empty (a, b) LETS classes in the range of $a \leq 10$ and $b \leq 3$, for the designed codes of the same rate in Tables 5.2 and 5.5. (All LETSs in the $(4, 4)$ class are 8-cycles.)

	$c = 4, R = 0.21$		$c = 5, R = 0.37$		$c = 6, R = 0.48$		$c = 7, R = 0.53$	
	$g = 8$	$g = 6$	$g = 8$	$g = 6$	$g = 8$	$g = 6$	$g = 8$	$g = 6$
LETS	$L = 90$	$L = 108$	$L = 180$	$L = 200$	$L = 300$	$L = 323$	$L = 210$	$L = 230$
	$m = 5$	$m = 6$	$m = 8$	$m = 9$	$m = 13$	$m = 14$	$m = 21$	$m = 23$
$(3, 3)$	0	0.47	0	0.19	0	0.16	0	0.12
$(4, 4)$	2.85	0.46	4.04	2.51	4.16	3.22	3.96	2.60
$(9, 3)$	4.13	0	6.93	1.28	2.51	4.11	2.25	1.33
$(10, 2)$	1.82	0	0.94	0	0	0.94	0.25	0

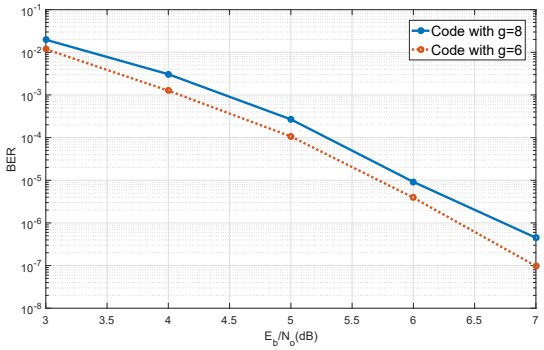


Figure 5.15: Performance comparison of two codes with $c = 4$ and different girths from Table 5.7.

Table 5.8: Normalized multiplicity of (a, b) LETSs in the range of $a \leq 8$ and $b \leq 3$, for the designed codes $\mathcal{C}_2$ (with $g = 6$) and $\mathcal{C}_3$ (with $g = 8$), compared to the codes of [7] and [65] with $g = 8$ (all codes have $\gamma = 3, c = 6, R \approx 0.49$). Codes of [7] and [65] have $L = 300$, and the designed codes $\mathcal{C}_2$ and $\mathcal{C}_3$ have $L = 380$.

LETS	Code of [7]	Code of [65]	$\mathcal{C}_2$	$\mathcal{C}_3$
$(3, 3)$	0	0	0.65	0
$(5, 3)$	1.96	1.14	0	0
$(6, 2)$	1.30	0.16	0	0
$(7, 3)$	15.23	7.77	4.06	5.68
$(8, 0)$	0.16	0	0	0
$(8, 2)$	4.21	2.43	0.97	1.46

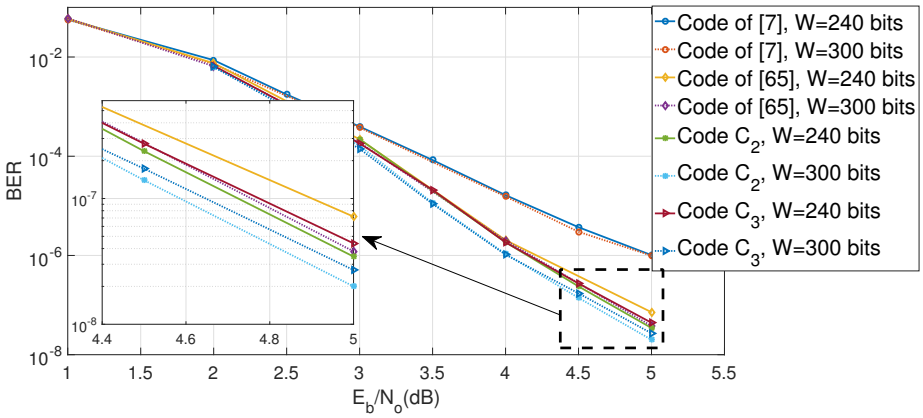


Figure 5.16: Performance comparison of designed codes $\mathcal{C}_2$ and $\mathcal{C}_3$ with their counterparts in [7, 65].

The code of [7] is designed to have the smallest m value for $g = 8$. The code of [65] is designed to have the same smallest m value as that of the code of [7], but to be free of some of the dominant TSs of that code and to have a larger minimum distance. For both these codes, we use $L = 300$, such that they have the same rate of $R \approx 0.49$ as $\mathcal{C}_2$. The third code with $g = 8$ is our designed code $\mathcal{C}_3$ from Table 5.4 with $m = 9$. In Table 5.8, we have presented the LETS distributions of the four codes within the range of $a \leq 8$ and $b \leq 3$. We have also given the BER of the four codes in Fig. 5.16 for two window sizes of $W_1 = 5(7 + 1) \times 6 = 240$ bits and $W_2 = 5(9 + 1) \times 6 = 300$ bits. These results show the superior LETS distribution and error floor performance of $\mathcal{C}_2$ (for the same decoding complexity and latency) in comparison with its girth-8 counterparts.

Table 5.9: SC-LDPC codes with $d_v = \gamma = 4$, $g = 6$, constructed based on Proposition 11, and free of LETSs within the range $\mathcal{R}_4$.

c	m	P matrix	run-time (sec.)	m in [7] for $g = 8$
5	5	5 1 5 3 0 5 4 0 0 2 1 5 4 0 5 0 0 1 4 5	1160	10
6	8	5 4 0 2 8 0 0 4 4 0 5 8 5 0 8 8 5 1 2 5 2 7 0 4	630	11
7	10	4 0 6 4 9 2 5 6 4 9 10 3 0 0 0 10 4 8 3 9 10 9 4 0 0 0 10 5	2509	14
8	14	9 12 2 3 5 1 3 11 8 14 9 1 0 12 12 3 0 12 0 11 9 14 0 0 14 0 6 0 12 0 4 11	1359	17

As a final design example for $d_v = 3$, using Proposition 10, we design SC codes with $c = 4$ and $c = 5$. The designed codes have $m = 15$ and $m = 33$, and the following exponent matrices:

$$P' = \begin{bmatrix} 0 & 0 & 13 & 15 \\ 15 & 3 & 0 & 0 \\ 15 & 9 & 9 & 1 \end{bmatrix}, \; P'' = \begin{bmatrix} 0 & 30 & 0 & 33 & 0 \\ 26 & 0 & 14 & 0 & 5 \\ 16 & 6 & 9 & 3 & 31 \end{bmatrix}, \tag{5.4}$$

respectively. Within the range $a \leq 12, b \leq 3$ and considering $L = 300$, the code with $c = 4$ has 276 instances of the $(9, 3)$ LETS structure depicted in Fig. D.1(e), and no LETS in classes $(10, 0)$ and $(10, 2)$. The code with $c = 5$, however, is free of all (a, b) LETSs within the range $a \leq 12, b \leq 3$.

5.6.2 SC-LDPC codes with $d_v = 4$ and $d_v = 5$

Table 5.9 presents the codes with $d_v = 4$ and $g = 6$, designed based on Proposition 11, and with the smallest m that we could find. These codes are free of LETSs within the range $\mathcal{R}_4$. For comparison, we have also listed in the table, the smallest value of m given in [7] for similar codes with $g = 8$. It is evident from the table that the designed codes have smaller memory and constraint length compared to their girth-8

Table 5.10: Multiplicity of non-empty (a, b) LETS classes in the range of $a \leq 11$ and $b \leq 10$, for the designed code C_4 ($L = 300$).

LETS	Code C_4
$(4, 8)$	3584
$(5, 10)$	57061
$(6, 10)$	13022
$(7, 10)$	798

counterparts in [7].

As another example of codes with $d_v = 4$, we construct a code $\mathcal{C}_4$ with $d_v = 4$, $d_c = 5$, $m = 38$, and $g = 8$, based on Proposition 12, and with the following exponent matrix:

$$P_4 = \begin{bmatrix} 1 & 12 & 0 & 0 & 31 \\ 9 & 22 & 34 & 25 & 0 \\ 0 & 27 & 33 & 3 & 27 \\ 38 & 0 & 8 & 2 & 11 \end{bmatrix}. \tag{5.5}$$

To construct this code, we started from the lower bound of Proposition 13, i.e., $m = 15$, and kept increasing m until we reached $m = 38$ and found the above exponent matrix within the time limit of 3 hours. Code $\mathcal{C}_4$ is free of LETSs in the range $\mathcal{R}_5$. The LETS distribution of the code for $L = 300$ is given in Table 5.10 for the range of $a \leq 11$ and $b \leq 10$.

Examples of designed codes with $d_v = 5$, $g = 6$, and free of LETSs within the range $\mathcal{R}_6$ are given in Table 5.11. These codes are designed based on Proposition 14 and have the smallest m values that we could find. The run-times of Algorithm 1 to design the exponent matrices for these m values are also given in the table. In Table 5.11, we have also given the smallest m values for codes with the same degree distributions but with $g = 8$. One can see that the designed codes have the same or smaller memory and constraint length compared to their counterparts in [7].

We note that in addition to having a smaller memory and constraint length, our designed codes with $g = 6$ have in general a superior LETS distribution and a better error floor compared to their girth-8 counterparts in [7]. As an example, we consider

Table 5.11: SC-LDPC codes with $d_v = \gamma = 5$, $g = 6$, constructed based on Proposition 14, and free of LETSs within the range $\mathcal{R}_6$.

c	m	P matrix	run-time (sec.)	m in [7] for $g = 8$
6	14	2 8 2 13 9 10 0 0 14 7 0 14 9 6 7 8 8 0 10 0 11 3 14 7 14 10 0 0 2 2	9774	14
7	19	7 18 0 19 11 0 3 3 3 10 10 0 19 0 0 1 13 1 17 4 8 13 11 10 0 6 0 16 19 0 9 2 0 15 1	10052	22
8	28	17 7 9 20 5 2 23 0 28 2 18 2 20 0 4 5 0 0 12 26 0 28 27 15 17 4 0 22 1 8 8 27 17 28 12 0 3 18 0 22	8840	31
9	30	0 22 14 25 10 0 26 12 6 2 30 9 1 4 25 30 29 29 14 0 12 25 0 1 28 0 30 0 5 0 22 17 10 8 26 0 12 30 16 0 23 20 0 2 21	4115	39
10	40	17 35 0 0 36 16 10 7 17 21 0 25 22 39 3 5 3 21 27 37 4 1 0 38 8 14 0 23 39 28 38 0 9 22 40 29 5 0 0 15 6 7 10 5 0 0 35 13 20 0	1002	49

$d_v = 5$ and $c = 7$, and compare our designed code from Table 5.11 with its counterpart in [7]. We select $L = 380$ and $L = 440$ for the two codes, respectively, so that both have the same rate $R = 0.25$. The LETS distribution and BER performance of both codes are given in Table 5.12 and Fig. 5.17, respectively. The window size for decoding is selected $W = 700$ bits for both codes, and thus both codes have the same decoding complexity and latency. One can see from Table 5.12 and Fig. 5.17 that the designed code has a superior LETS distribution and error floor compared to its counterpart in [7].

5.6.3 Comparison with codes of larger girth

In this part, we consider an SC-LDPC code designed in [6] with $d_v = 4, c = 5, g = 10$, and $m = 50$. For comparison, we construct a code $\mathcal{C}_5$ with the exact same degree distribution and memory but with $g = 8$, based on Proposition 12. The exponent

Table 5.12: Normalized multiplicity of 8-cycles and non-empty (a, b) LETS classes in the range of $a \leq 8$ and $b \leq 9$, for the designed code with $c = 7$ from Table 5.11 (with $L = 380$ and $m = 19$) compared with its counterpart from [7] with $L = 440$ and $m = 22$.

LETS	Designed code with $g = 6$	Code of [7] with $g = 8$
$(3, 9)$	3.74	0
$(4, 12)$	0.41	106.67
$(8, 8)$	0	0.13

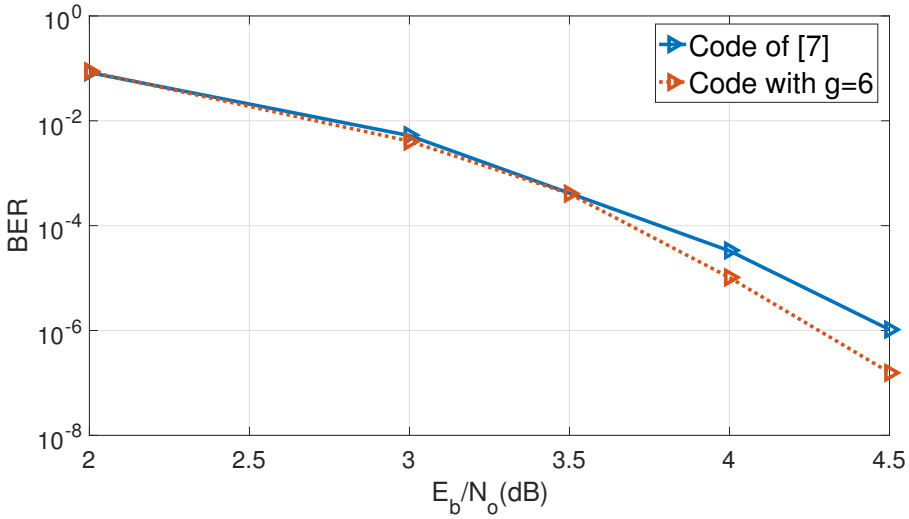


Figure 5.17: Performance comparison of our designed code with $g = 6$ and its counterpart with $g = 8$ from [7].

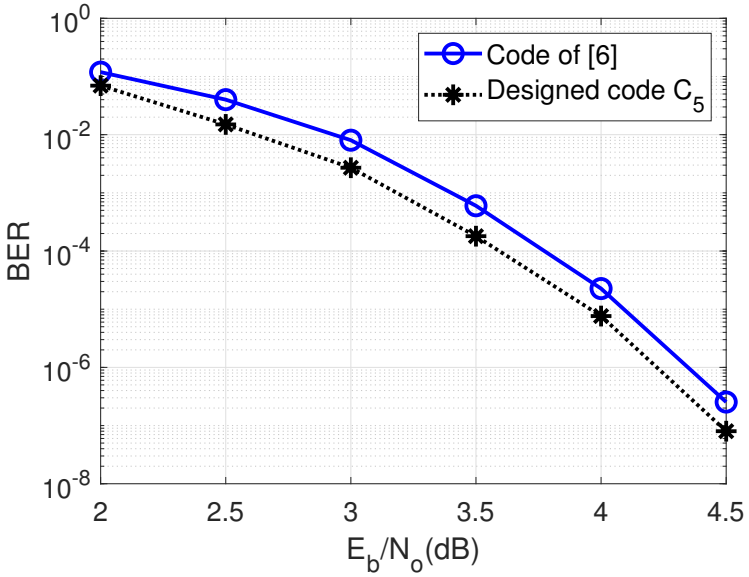


Figure 5.18: Performance comparison of the designed code $\mathcal{C}_5$ with $g = 8$ with its counterpart in [6] with $g = 10$.

matrix of $\mathcal{C}_5$ is given by

$$P_5 = \begin{bmatrix} 9 & 44 & 50 & 49 & 0 \\ 45 & 40 & 37 & 28 & 26 \\ 17 & 18 & 20 & 42 & 41 \\ 43 & 0 & 49 & 11 & 50 \end{bmatrix}. \tag{5.6}$$

For both codes, we use $L = 1000$, which corresponds to $R = 0.16$, and use a window size of $W = 5(50 + 1) \times c = 1275$ bits for decoding. The LETS distribution and BER of the two codes are compared in Table 5.13 and Fig. 5.18, respectively. These results demonstrate the superiority of $\mathcal{C}_5$ compared to its counterpart with larger girth.

Table 5.13: Normalized multiplicity of non-empty (a, b) LETS classes in the range of $a \leq 11$ and $b \leq 12$, for the designed code $\mathcal{C}_5$ with $g = 8$ and its counterpart with $g = 10$ from [6].

LETS	Code of [6]	Designed code $\mathcal{C}_5$
$(4, 8)$	0	1.54
$(5, 10)$	30.22	29.32
$(6, 10)$	0	2.83
$(6, 12)$	233.22	1.52
$(7, 12)$	60.54	0
$(8, 12)$	3.39	0

5.6.4 Comparison with circulant-based (CB) SC-LDPC codes

To further demonstrate the competitive performance of our designed codes, we compare them with similar circulant-based (CB) codes (or equivalently quasi-cyclic codes) from [26] and [62], lifted based on the lifting degree M. As the first example, we consider $d_v = 3, c = 7$, and compare the performance of codes designed in [26], with our designed code $\mathcal{C}_6$ with the same degree distribution and constraint length, given in the last row of Table 5.5. The codes of [26] have $m = 2, M = 8$ and $m = 3, M = 6$, and are designed to minimize the multiplicity of 6-cycles so that the girth of both codes is 8. We select the coupling length of $L = 86$ and $L = 115$ for the first and second code of [26], respectively, and $L = 690$ for $\mathcal{C}_6$. This means all codes have the same rate $R = 0.56$. To have similar decoding latency and complexity for all codes, we consider window size of $5(2 + 1) \times 8 \times c = 5(3 + 1) \times 6 \times c = 5(23 + 1) \times c = 840$ bits for all codes. From Table 5.14 and Fig. 5.19, one can see that $\mathcal{C}_6$ has a superior LETS distribution and BER performance compared to its counterparts.

As the second example of this subsection, we consider an array-based CB SC-LDPC code constructed in [62] with $\gamma = d_v = 3, c = d_c = 13, m = 2$, and lifting degree $M = 13$, optimized to be free of 6-cycles ($g = 8$). For comparison, we construct a code $\mathcal{C}_7$ with the same degree distribution, but a smaller girth of 6, based on Proposition 6. The constructed code has $m = 52$, is free of all LETSs within the range $\mathcal{R}_2$, and has

Table 5.14: Normalized multiplicity of non-empty (a, b) LETS classes in the range of $a \leq 10$ and $b \leq 3$, for our designed code C_6 compared to its counterparts from [26] (All codes have $d_v = 3, c = 7$ and $R = 0.56$).

LETS class	Code of [26], $m = 2, M = 8, L = 86, g = 8$	Code of [26], $m = 3, M = 6, L = 115, g = 8$	Designed code C_6, $L = 690, g = 6$
$(3, 3)$	0	0	0.13
$(5, 3)$	0.08	0.13	0
$(6, 2)$	0.05	0	0
$(7, 3)$	1.22	0.20	0
$(8, 2)$	0.51	0	0
$(9, 3)$	2.42	2.57	1.49
$(10, 2)$	1.08	0.39	0

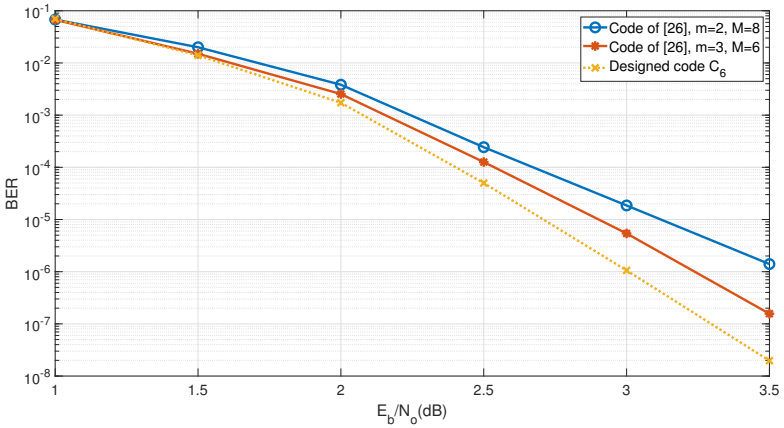


Figure 5.19: Performance comparison of our designed code $\mathcal{C}_6$ and its counterparts from [26].

Table 5.15: Normalized multiplicity of non-empty (a, b) LETS classes in the range $a \leq 8, b \leq 3$, for our designed code $\mathcal{C}_7$ compared to its counterpart from [62] (Both codes have $d_v = 3, d_c = 13$, and $R = 0.7583$).

LETS class	Code of [62], $L = 30$, $N = 5070$	Code $\mathcal{C}_7$, $L = 780$, $N = 10140$
$(3, 3)$	0	0.29
$(5, 3)$	3.10	0
$(6, 0)$	0.22	0
$(6, 2)$	0.22	0
$(7, 3)$	66.43	8.61
$(8, 0)$	1.75	0
$(8, 2)$	14.99	1.30

the following exponent matrix:

$$P_7 = \begin{bmatrix} 0 & 45 & 0 & 49 & 10 & 29 & 0 & 25 & 0 & 0 & 31 & 41 & 44 \\ 20 & 43 & 28 & 0 & 0 & 17 & 17 & 0 & 36 & 10 & 34 & 0 & 0 \\ 52 & 0 & 38 & 47 & 22 & 0 & 28 & 19 & 5 & 44 & 0 & 33 & 9 \end{bmatrix}. \tag{5.7}$$

To have the same rate $R = 0.75$, we select $L = 780$ and $L = 30$ for $\mathcal{C}_7$ and its counterpart, respectively. The decoding window sizes are selected to be $4(52 + 1) \times 13 = 2756$ bits, and $5(2 + 1) \times 13 \times 13 = 2535$ bits, respectively, resulting in similar decoding latency and complexity. In Table 5.15, we compare the LETS distributions of the two codes. As can be seen, although $\mathcal{C}_7$ has a smaller girth, its LETS distribution is far superior to that of the code of [62].[4] In particular, while the minimum distance of the latter code is six, the minimum distance of $\mathcal{C}_7$ is strictly larger than eight. In [62], it is noted that there is a trade-off between the multiplicity of 6-cycles and the minimum distance of an SC code. Our results demonstrate that if we relax the constraint on the multiplicity of 6-cycles, we can instead impose stricter conditions on the interactions of such cycles, such as those given in Proposition 3, to effectively eliminate the harmful trapping sets including low-weight codewords. Performance comparison of the two codes, which demonstrates significantly lower error floor of $\mathcal{C}_7$, is provided in Fig. 5.20.

[4] We note that this improvement comes at the expense of an increase in the constraint length of our designed code $\mathcal{C}_7$.

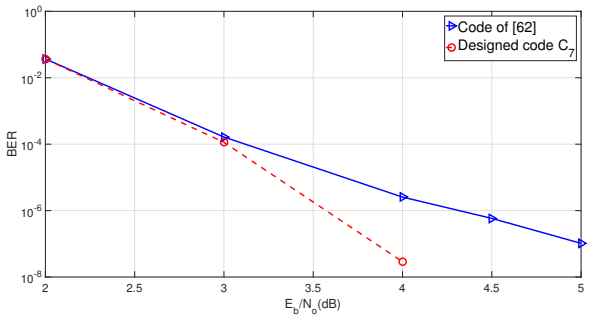


Figure 5.20: Performance comparison of our designed code $\mathcal{C}_7$ and its array-based CB SC-LDPC counterpart from [62].

5.6.5 Design of high-rate codes

To demonstrate that our techniques can be applied to design high-rate codes with large values of d_c, in this subsection, we design two high-rate codes and compare them with similar codes in the literature. As the first example, we consider $d_v = 3, c = 31$, and design a code $\mathcal{C}_8$ with $m = 92$ based on Proposition 4, which has the following exponent matrix:

$$P_8 = \begin{bmatrix} 84 & 7 & 21 & 0 & 37 & 0 & 88 & 83 & 35 & 66 & 43 & 13 & 58 & 74 & 31 & 29 & 77 & 61 & 65 & 70 & 28 & 50 & 0 & 1 & 78 & 86 & 90 & 72 & 22 & 32 & 4 \\ 0 & 0 & 0 & 28 & 0 & 29 & 0 & 0 & 20 & 0 & 0 & 0 & 76 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 85 & 38 & 41 & 0 & 0 & 0 & 9 & 0 & 0 & 0 \\ 80 & 58 & 54 & 81 & 81 & 56 & 52 & 62 & 0 & 24 & 83 & 92 & 0 & 64 & 45 & 23 & 47 & 17 & 11 & 75 & 63 & 0 & 13 & 0 & 90 & 0 & 9 & 0 & 89 & 85 & 70 \end{bmatrix}.$$

It took Algorithm 1 less than one minute to construct P_8. By selecting $L = 552$, this code has a rate of $R = 0.88$. For comparison, we use the array-based SC-LDPC code given in [62, Table I] with parameters $d_v = 3, c = 31, m = 2, g = 6$, and lifting degree $M = 31$. This code, which is designed by a joint optimization of the multiplicity of 6-cycles and the minimum distance, has the same degree distribution and constraint length as those of $\mathcal{C}_8$. To have the same rate $R = 0.88$ as $\mathcal{C}_8$, we select $L = 12$ for the code of [62]. Table 5.16 presents the LETS distribution for the two codes within the range $a \leq 8, b \leq 3$. From Table 5.16, it can be seen that $\mathcal{C}_8$ has a larger minimum distance and smaller multiplicity of LETSs in most of the LETS classes. This is despite the fact that the multiplicity of 6-cycles is larger in $\mathcal{C}_8$.

As another example, we consider $d_v = 5, d_c = 25, g = 6$, and using Proposition 14,

Table 5.16: Normalized multiplicity of non-empty (a, b) LETS classes in the range of $a \leq 8, b \leq 3$, for our designed code $\mathcal{C}_8$ compared to its counterpart from [62] (Both codes have $d_v = 3, d_c = 31, g = 6, R = 0.88$).

LETS class	Code of [62], $L = 12, N = 11532$	$\mathcal{C}_8$, $L = 552, N = 17112$
$(3, 3)$	1.68	2.98
$(4, 2)$	0.6	0
$(5, 3)$	33.62	47.28
$(6, 0)$	0.6	0
$(6, 2)$	16.33	9.73
$(7, 3)$	1609.73	1384.6
$(8, 0)$	11.40	0.11
$(8, 2)$	652.24	338.68

we design a code $\mathcal{C}_9$ with $m = 250$, rate $R \approx 0.8$ and the following exponent matrix:

$$P_9 = \begin{bmatrix} 46 & 0 & 40 & 120 & 145 & 76 & 226 & 56 & 101 & 243 & 67 & 181 & 198 & 143 & 118 & 233 & 97 & 225 & 80 & 17 & 102 & 193 & 71 & 166 & 54 \\ 170 & 98 & 234 & 31 & 186 & 240 & 240 & 242 & 105 & 198 & 164 & 42 & 69 & 170 & 234 & 42 & 31 & 8 & 186 & 227 & 158 & 24 & 137 & 137 & 190 \\ 20 & 193 & 86 & 0 & 12 & 0 & 100 & 230 & 0 & 84 & 0 & 236 & 43 & 33 & 0 & 88 & 78 & 203 & 134 & 0 & 0 & 236 & 250 & 216 & 0 \\ 0 & 42 & 0 & 122 & 39 & 13 & 0 & 0 & 148 & 115 & 136 & 0 & 87 & 99 & 76 & 121 & 0 & 0 & 0 & 129 & 235 & 227 & 175 & 0 & 202 \\ 19 & 73 & 60 & 199 & 0 & 50 & 121 & 134 & 173 & 0 & 51 & 216 & 0 & 0 & 123 & 0 & 213 & 237 & 22 & 61 & 244 & 0 & 0 & 242 & 131 \end{bmatrix}.$$

It took Algorithm 1 5380 seconds to design this code for $m = 250$. We note that this code has a significantly smaller memory in comparison with $m = 458$ of the code designed in [7] that has the exact same degree distribution but $g = 8$.

Algorithm 1 Proposed search algorithm for the construction of SC-LDPC codes with a $\gamma \times c$ exponent matrix P, girth g and free of LETSs in the range $\mathcal{R}$.

Input: γ, c, g, $\mathcal{R}$, m_0 (lower bound on m), τ (maximum search time per m).

1: $F \leftarrow 1, \quad m \leftarrow m_0$
2: **while** $F = 1$ **do**
3: **Initialization:** $j = 1$
4: $Index = \text{ones}(1, c+1)$ ▷ $Index$ is set to an all-one row vector of length $c+1$
5: $\mathcal{D} \leftarrow$ Matrix of size $\gamma \times \left((m+1)^\gamma - m^\gamma\right)$ consisting of all possible column vectors of length γ with the elements from $\{0, 1, \ldots, m\}$ and with at least one zero element in each column vector.
6: **while** $1 \leq j \leq c$ **do**
7: **for** $k = Index(j)$ to $(m+1)^\gamma - m^\gamma$ **do**
8: **if** run-time exceeds τ **then** go to Line 27.
9: **end if**
10: $P(:, j) \leftarrow k$-th column of $\mathcal{D}$ ▷ Column j of P is assigned.
11: **if** the current P matrix satisfies the girth constraint **then**
12: for the current P matrix, check the design constraints based on $\gamma, g, \mathcal{R}$.
13: **if** all the constraints are satisfied **then**
14: $Index(j) \leftarrow k+1$ ▷ Save the next column index, in case the algorithm backtracks to column j.
15: $j \leftarrow j+1$
16: $Index(j) \leftarrow 1$
17: Go to Line 21.
18: **end if**
19: **end if**
20: **end for**
21: **if** $k = (m+1)^\gamma - m^\gamma$ AND $j \leq c$ **then**
22: $j \leftarrow j-1$
23: **else if** $j > c$ **then**
24: $F \leftarrow 0$
25: **end if**
26: **end while**
27: $m \leftarrow m+1$
28: **end while**
29: **Output:** P

Chapter 6

Spatially Coupled LDPC Codes with Small Constraint Length and Low Error Floor

6.1 Introduction

It is known that in finite-length (practical) scenarios, SC-LDPC codes suffer from error floor [61], as a result of some small substructures of the code's Tanner graph, referred to as *trapping sets* (TSs) [72]. Another important design aspect for SC-LDPC codes is their encoding and decoding complexity, which is related to the constraint length (ν_s) of such codes. In this regard, the work of [7], [78] was about the design of time-invariant SC-LDPC codes with small constraint length ν_s by minimizing the memory m for a given girth. Due to the small constraint length for such codes, they are called as *compact codes*.

So far, in Chapter 5 we introduced a design technique based on simple conditions on the short cycles that results in the elimination of most harmful LETS structures in the Tanner graph of SC codes. We also attempted to find the smallest m satisfying our conditions to maintain the small constraint length in our designed codes. In this chapter, again the two main objectives of low error floor and small constraint length are targeted. Thus, following the same approach as in [7], [78], we directly design the $\gamma \times c$ *symbolic matrix* of SC-LDPC codes (as given in Equation (2.11)), and in each case maintain the same degree distribution, girth and constraint length as the codes designed in [7], [78]. This guarantees the same advantages, as the codes of [7], [78], in terms of complexity and latency. In addition, by careful design of the

exponent matrix, we significantly reduce the error floor. For this, we first investigate the parent/child relationship between the dominant TSs of the code, including non-zero codewords, and to reduce the multiplicity of the TSs effectively and efficiently, we devise a technique based on the parent/child relationship between TSs that aims at successively minimizing the multiplicity of TSs depending on their harmfulness.

In this chapter, first, we briefly explain the main idea for our approach in Section 6.2. Then, a comprehensive description of our design approach is presented in Section 6.3. In Section 6.4, we present the simulation results to compare our designed codes (using the proposed technique) with the stat-of-the-art.

6.2 Main Idea

Leafless ETSs (LETS) are known to be the most harmful TSs in the error floor region for variable-regular LDPC codes over the additive white Gaussian noise (AWGN) channel, see, [32] and the references therein. Finding LETSs, however, is in general a difficult problem [19]. Nevertheless, different characterizations of LETSs are provided in the literature [45], [32]. In particular, the so-called *dpl characterization* of [32] characterizes each LETS as a nested sequence of LETSs that starts from a simple cycle and grows in multiple stages through three expansions *dot*, *path* and *lollipop*, until it reaches the target LETS. In this work also, we use the same three expansions to describe the parent/child relationships between LETSs. The main distinguishing factor however is that while the goal in [32] is to exhaustively characterize all the (a, b) LETS structures within a rectangular region of $a \leq a_{\max}$ and $b \leq b_{\max}$, in this work, we have no such limitation. In addition, we tailor our parent/child characterization of the target LETS structures to lend itself to a successive search/minimization of the structures in accordance to their harmfulness.

When a parent-child relationship exists between two structures $\mathcal{S} \subset \mathcal{S}'$ through an expansion $\mathcal{E}$, we can avoid having the structure $\mathcal{S}'$ in the Tanner graph G if we make sure that all the instances of structure $\mathcal{S}$ in G cannot be expanded through $\mathcal{E}$. In this case, we say that the instances of $\mathcal{S}$ are *non-expandable* (nE) with respect to $\mathcal{E}$, or $\mathcal{E}$-nE. As an example, we say a LETS S is dot_m-nE, if there is no variable node outside S that is connected to m degree-1 check nodes of S.

6.3 Design Approach

Our goal here is to design time-invariant SC-LDPC codes with the same values of γ, c and m as those in [7], [78] but with lower error floors. In this regard, consider a code $\mathcal{C}$ designed in [7] or [78] with given values of γ, c and girth g, and suppose that the (minimal) memory of this code is m. Also suppose that the dominant TS structures of $\mathcal{C}$ are provided in the set $\mathcal{L}$. These TS structures may be obtained by Monte Carlo simulations or through characterization tables such as those of [45], [32], where (a, b) classes (each with multiple non-isomorphic structures) with small a and b values are considered to be dominant.

The main idea of the proposed technique is to minimize the multiplicity of these structures in the constructed code. For this, we sort the classes of structures in $\mathcal{L}$ according to their harmfulness, and then aim to successively minimize them with giving higher priority to the more harmful structures. We also closely investigate the parent/child relationships among these structures and devise a low-complexity algorithm to search for these structures in the constructed codes. For this, we organize the TS classes in a forest (collection of rooted trees), where the classes in $\mathcal{L}$ are the top nodes of the trees and the roots are either short simple cycles or structures that can be easily expanded from short simple cycles. Any branch of the tree represents a parent-child relationship between the end nodes of the branch, where the child (parent) is the head (tail) of the branch.

In this work, since our focus is on LETSs and since each LETS structure has at least one cycle, our goal is to characterize each LETS structure of interest as a nested sequence of LETSs which starts from short simple cycles (cycles of length g or at most $g+2$) and then is expanded successively through the three expansions *dot*, *path* and *lollipop* to reach the target LETS. With regards to the short cycles, to minimize the search complexity, we always use cycles of length g as the starting point, unless the target LETS structure cannot be generated by successive expansions of the g-cycles, i.e., the girth g' of the subgraph of the target LETS is larger than g. In that case, we choose g' as the length of the initial cycle structure. For the expansions, since *dot* has the lowest complexity, followed by *path* and *lollipop*, respectively [32], we use them in the same order of priority, i.e., we use as many dot_m expansions as possible, and only if an structure cannot be generated only by *dot* expansions, then we use the other two expansions with the first priority given to pa_m before any lo^c_m is used.

An example of the forest for the (a, b) LETSs, in the range $a \leq 12, b \leq 3$, of a

code with variable node degree $d_v = \gamma = 3$ and $g = 8$ is shown in Fig. 6.1. The tree roots in this case are $(4, 4)$ (8-cycles), $(6, 4)$, $(8, 4)$ and $(10, 4)$ classes, from left to right, respectively. These structures are at the top level of parent structures, denoted by $\mathcal{P}$ in Fig. 6.1. As can be seen, all the structures of $\mathcal{L}$ in the forest are generated by (successive) applications of dot_m expansion. In the first tree, the initial structure is only the root of the tree, while in the second, third and fourh trees, *dot* expansions have to be also applied to some structures in $\mathcal{P}$ other than the roots. For example, in the first tree, there is only one structure in the $(5, 3)$ class and this structure is a dot_2 expansion of 8-cycles. On the same tree, $(6, 0)$ and $(6, 2)$ classes, each having only one structure, are children of the $(5, 3)$ structure through dot_3 and dot_2 expansions, respectively. Finally, the last node on the first tree is the $(7, 1)$ class also with only one structure, which is the dot_2 expansion of the $(6, 2)$ structure. As another example, in the third tree, there are 17 non-isomorphic structures in the $(9, 3)$ class. Although 16 of these structures can be generated from the structures in the $(8, 4)$ class using the dot_2 expansion, the one remaining structure cannot. It however, can be generated from structures in the $(8, 6)$ class through dot_3.

The forest, whose construction was just described, is then used as a road-map for our design. The general idea is to organize the TS structures of $\mathcal{L}$ in the order of their decreasing harmfulness and then try to minimize their multiplicities successively in that order. If, at a given stage of the successive minimization, the goal is to minimize the multiplicity of the LETS structure $\mathcal{S}$ which is an $\mathcal{E}$-expansion of the parent structure $\mathcal{S}'$, where $\mathcal{E}$ can be a *dot*, a *path* or a *lollipop* expansion, we achieve this by minimizing the number of instances of $\mathcal{S}'$ that are $\mathcal{E}$-expandable, or equivalently, maximizing $\mathcal{E}$-nE instances. Note that if $\mathcal{S}'$ is not a target LETS itself, i.e., it does not belong to $\mathcal{L}$, this goal is much less restrictive than trying to eliminate all the instances of $\mathcal{S}'$ in the code, and therefore results in better designs.

In the design process, we first find some candidate exponent matrices Φ with the same values of γ, c, m and g as those in [7, 78], whose corresponding codes can potentially outperform the codes of [7, 78] in the error floor region. We then search within Φ to find exponent matrices that in fact have superior TS distribution and error floor. In the search process within Φ, maximizing $\mathcal{E}$-nE instances at every step of successive minimization proves to be too restrictive, i.e., the pool of candidates shrinks very quickly as we go through the iterative process, with the final solution only minimizing one or two of the targeted TSs. This implies that in the design process, no

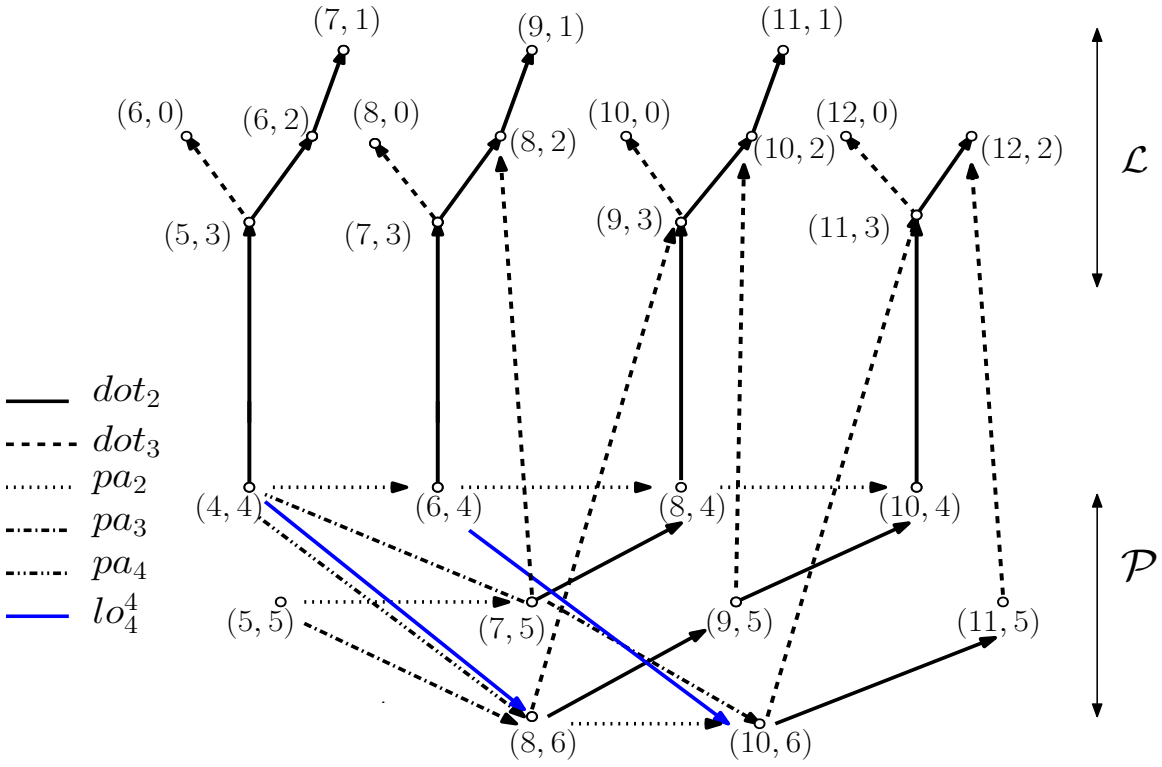


Figure 6.1: The (a, b) LETS classes of codes with $d_v = \gamma = 3$ and $g = 8$, for $a \leq 12, b \leq 3$, organized in a forest with the required parent classes and the corresponding expansions.

control can be exerted over the multiplicity of any but one or two of the most harmful TSs. To alleviate this problem, at each step of minimization (iteration), we only reject a fraction of the candidates that are the worst in terms of the multiplicity of $\mathcal{E}$-nE instances, and keep the rest, which of course include all those with the maximum number of $\mathcal{E}$-nE instances. As the iterations go on, and the complexity of the search algorithm increases (as more TSs will have to be searched for), this pool of candidates shrinks further, thus making the total complexity of the search tractable. Eventually, we are left with one or just a few candidate exponent matrices, all with attractive TS distribution and error floor.

Finally, we note that the idea of searching for LETSs as successive expansions of simple cycles was also used in [46] to eliminate certain LETSs in the PEG construction of block LDPC codes with low error floors.

6.4 Designed Codes and Simulation Results

We present our designs for $g = 8$ and $d_v = \gamma = 3$ and 4, as well as $g = 10$ and $\gamma = 3$.

Consider the scenario of an SC-LDPC code with $\gamma = 3$ and $g = 8$, whose LETS structures were discussed in Fig. 6.1. Clearly, based on the parent/child relationships shown in Fig. 6.1, the multiplicity of many of the dominant TSs of this code are affected by the multiplicity of 8-cycles, denoted by η here.[1] To reduce the error floor, one thus would be interested in codes with small η values. Moreover, since the most problematic TSs of the code are located on the first tree, rooted at 8-cycles, one would be interested in codes with a small number of $(5, 3)$ structures, denoted by ξ here, (to calculate ξ from the exponent matrix, one needs to count the number of cases where two 8-cycles have 4 edges in common.). The same discussion is applicable to the case of $\gamma = 4$, except that the 8-cycles and their dot_2 expansions are in $(4, 8)$ and $(5, 8)$ classes, respectively. To obtain the set Φ of exponent matrices, we first start by the exponent matrix P_0 of a code designed in [7]. All the candidate exponent matrices in Φ are then generated based on P_0 (in one or multiple steps) by maintaining the same values of γ, c, m and g. Suppose that P_0 (or a modified version of P_0) has η_0 8-cycles and ξ_0 instances of the dot_2 expansion of 8-cycles. To generate the candidate exponent matrices, we modify the columns of P_0 (or a modified version of P_0) such

[1] To calculate η from the exponent matrix P, one needs to count the number of cases where the equation $\sum_{k=1}^{4} p_{i_k,j_k} - p_{i_k,j_{k+1}} = 0$ is satisfied for indices $1 \leq i_k \leq \gamma$ and $1 \leq j_k \leq c$, such that $i_k \neq i_{k+1}, j_k \neq j_{k+1}$, for $k = 1, 2, 3$, and $i_4 \neq i_1, j_4 \neq j_1$.

that the girth is maintained, and at least one of the two values η and ξ is smaller than η_0 and ξ_0, respectively. (For the case of $g = 10$, we only consider the number of 10-cycles.) We do this by first modifying the columns of P_0 (or a modified version of P_0) that contribute to the largest number of g-cycles. The choices of the elements within the selected columns are made by random from the set $\{0, 1, \ldots, m\}$.

After the set Φ is generated, we examine the candidate exponent matrices in Φ for their LETS distributions, and in a number of iterations, trim the set Φ to converge to one or just a few exponent matrices with the best TS distribution. At each iteration of the trimming process, we focus on one of the branches in the forest that includes the dominant TSs of the code. We start from the root of the first tree and move up the branches of the tree, and then move to the rest of the trees in order of their harmfulness. For branches that are in the same level of a tree, we first consider the ones whose heads are more harmful. For each branch under consideration (in a specific iteration of the trimming process), we count the multiplicity of the LETSs at the head of the branch for all the remaining candidate exponent matrices, and trim a fraction of candidates with the largest such multiplicity. Our experiments show that for many cases trimming about half of the candidates for the first few iterations is a proper choice to provide a good trade-off between the complexity and the quality of the designed codes. To examine the TS distribution of candidate exponent matrices, as described above, we construct the terminated version of parity-check matrix given in (2.10) with $L = \left\lfloor \frac{a_{\max}}{2} \right\rfloor m + 1$, where $a_{\max}$ is the size of the largest TS of interest. This choice is the smallest value of L that guarantees any structure of size $a_{\max}$ will be covered within the Tanner graph of the SC-LDPC code [24]. The pseudo-code of the proposed approach is given in Algorithm 2. Parameters J and k, respectively, determine the maximum size of Φ and the number of columns of P_0 that are modified simultaneously. In general, increasing the value of J improves the final design at the expense of increased complexity. Increasing k can also be beneficial in improving the final result. In the following, we design a number of SC-LDPC codes using the approach just explained. For the simulations, we use a floating-point sliding window decoder based on belief propagation algorithm with a pair-wise check node reduction, known as box-plus [14]. The maximum number of iterations per window is 50, and the simulations are performed over the binary-input AWGN channel. The decoding latency (in bits) of the sliding window decoder is $T = c \times W$, where W denotes the window size in blocks, and is selected to be in the range of $5(m+1) \leq W \leq 10(m+1)$,

Algorithm 2 Proposed algorithm for the construction of SC-LDPC codes with low error floor.

1: **Input:** P_0, g, $\mathcal{L}$ and the corresponding forest, J, k.
2: $\eta_0 \leftarrow$ no. of g-cycles corresponding to P_0
3: **if** $g \leq 8$ **then**
4: $\quad \xi_0 \leftarrow$ no. of dot_2 expansions of g-cycles for P_0
5: **end if**
6: Let $\pi_1, \ldots, \pi_c$ be the columns of P_0 ordered according to their contributions to g-cycles
7: $j \leftarrow 1,\ \ i \leftarrow 0,\quad n \leftarrow 1$
8: **while** $j \leq J$ AND $i \leq c - k$ **do**
9: $\quad$ Let Π^* be the set of all possible choices of $k-1$ distinct columns from $\pi_{i+2}, \ldots, \pi_c$
10: $\quad \Pi \leftarrow$ combination of π_{i+1} with every $(k-1)$-ary collection in Π^*
11: $\quad$ **for** each collection t of k columns in Π selected in lexicographic order **do**
12: $\quad\quad$ Modify columns in t of P_0 randomly and store the result in B until all (or most of) the possible choices are exhausted ▷ Repetitions of B are avoided.
13: $\quad\quad$ **if** $\{B$ has girth $g\}$ AND $\{\eta(B) \leq \eta_0, \xi(B) < \xi_0$ or $\eta(B) < \eta_0, \xi(B) \leq \xi_0\}$ **then** ▷ For $g > 8$, the second condition is changed to $\eta(B) < \eta_0$, for $n = 1$, and $\eta(B) \leq \eta_0$, for $n > 1$.
14: $\quad\quad\quad \Phi\{j\} \leftarrow B,\ j \leftarrow j + 1$
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: $\quad i \leftarrow i + 1$
18: $\quad$ **if** $i > c - k$ **then**
19: $\quad\quad P_0 \leftarrow \Phi\{n\},\ i \leftarrow 0,\ n \leftarrow n + 1$
20: $\quad\quad$ Execute Lines 2–6, and then go to Line 9
21: $\quad$ **end if**
22: **end while**
23: $a_{max} \leftarrow$ size of the largest LETS structure in $\mathcal{L}$
24: Let $\mathcal{S}_1, \ldots \mathcal{S}_{|\mathcal{L}|}$ denote the LETS structures in $\mathcal{L}$ in the order of reduced harmfulness
25: Create $H_{SC}(\gamma, c, m, L = \left\lfloor \frac{a_{\max}}{2} \right\rfloor m + 1)$ for each exponent matrix in Φ
26: $i \leftarrow 1$
27: **while** $|\Phi| > 1$ AND $i \leq |\mathcal{L}|$ **do**
28: $\quad$ **for** each structure s (or class of structures) in the forest leading up to $\mathcal{S}_i$ (that has not been visited before) and finally $\mathcal{S}_i$ itself **do**
29: $\quad\quad$ Find the multiplicity of s ($|s|$) in all the codes (H_{SC}) corresponding to all the exponent matrices in Φ, and discard a fraction $p(s)$ of those exponent matrices that have the largest $|s|$
30: $\quad$ **end for**
31: $\quad i \leftarrow i + 1$
32: **end while**
33: **Output:** Φ.

to minimize the performance degradation compared to the flooding schedule [17]. The minimum distance values $d_{\min}$ of the codes are obtained using the algorithm of [33].

As the first example, we consider the exponent matrix with $\gamma = 3$, $c = 6$, and $m = 7$ designed in [7]. The multiplicities of LETSs of the SC-LDPC code corresponding to this exponent matrix with $L = 300$ within the range $a \leq 8, b \leq 3$ are given in Table 6.1. Only the classes with nonzero multiplicity are shown in the table. Our simulations show that in the error floor region the vast majority of the errors of this code are undetected and belong to the $(8, 0)$ class. In fact, this code has a minimum distance $d_{\min} = 8$. For our design, we first consider the forest of Fig. 6.2(a), where the leafs are ordered in terms of their decreasing priority from left to right. By applying the successive minimization to this forest with $J = 600$ and $k = 3$, we obtain the following exponent matrix:

$$P_1 = \begin{bmatrix} 1 & 0 & 0 & 7 & 7 & 5 \\ 7 & 4 & 7 & 0 & 4 & 3 \\ 0 & 7 & 6 & 5 & 0 & 0 \end{bmatrix},$$

The code C_1 corresponding to P_1 with $L = 300$ has $d_{\min} = 14$, and a much better LETS distribution compared to the code of [7], as can be seen in Table 6.1. The bit error rate (BER) of C_1 along with that of the code of [7] are given in Fig. 6.3(a). Although, one can see the significant improvement in the error floor, there is still a slight change in the slope of the BER curve of C_1 at around 10^{-7}. Simulations show that LETSs in the class of $(6, 2)$ are responsible for this. We thus design a new code C_2, this time based on the forest of Fig. 6.2(b), which has the following exponent matrix:

$$P_2 = \begin{bmatrix} 2 & 5 & 7 & 7 & 0 & 5 \\ 6 & 0 & 0 & 4 & 7 & 7 \\ 7 & 7 & 6 & 0 & 6 & 0 \end{bmatrix}.$$

The LETS distribution and BER of C_2 are given in Table 6.1 and Fig. 6.3(a), respectively. As can be seen, C_2 has a superior error floor compared to C_1. This is despite

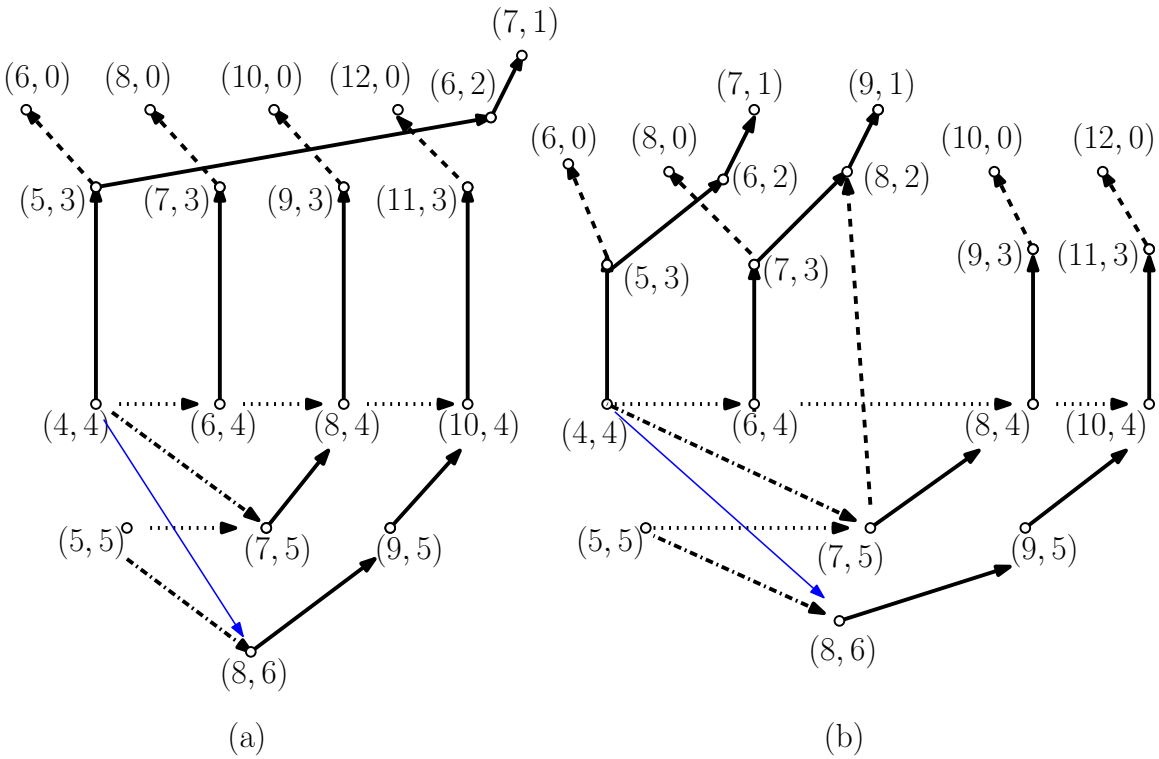


Figure 6.2: The forest used for the design of (a) C_1 and (b) C_2.

the fact that $d_{\min}$ of C_2 (12) is smaller than that of C_1 (14).

To demonstrate the applicability of our approach to design codes with large values of c and γ, we also design exponent matrices for $c = 17$ ($\gamma = 3$), and $\gamma = 4$ ($c = 5$):

$$P_3 = \begin{bmatrix} 5 & 17 & 6 & 12 & 30 & 0 & 7 & 37 & 30 & 20 & 1 & 33 & 0 & 16 & 0 & 0 & 21 \\ 29 & 0 & 21 & 24 & 15 & 34 & 0 & 0 & 19 & 0 & 7 & 9 & 14 & 0 & 36 & 33 & 7 \\ 0 & 28 & 0 & 0 & 0 & 32 & 30 & 29 & 19 & 21 & 31 & 0 & 37 & 36 & 28 & 2 & 0 \end{bmatrix},$$

$$P_4 = \begin{bmatrix} 10 & 3 & 6 & 1 & 0 \\ 10 & 0 & 0 & 0 & 8 \\ 0 & 5 & 0 & 7 & 9 \\ 0 & 8 & 9 & 2 & 0 \end{bmatrix}.$$

Table 6.1: Distribution of 8-cycles and (a, b) LETSs in the range of $a \leq 8$ and $b \leq 3$ for the designed codes C_1 and C_2 compared to the code of [7] (all codes have $\gamma = 3, c = 6, m = 7, L = 300, g = 8, N = 1800, R = 0.49$).

LETS	Code in [7]	Code C_1	Code C_2
8-cycles	22647	17350	17354
$(5, 3)$	3531	2352	2057
$(6, 2)$	2350	880	298
$(7, 3)$	27426	15749	13991
$(8, 0)$	293	0	0
$(8, 2)$	7590	3794	4380

For P_3 and P_4, in Algorithm 2, we use parameters $\{J = 70, k = 2\}$ and $\{J = 200, k = 2\}$, respectively. Similar to their counterparts in [7], P_3 and P_4 both have $g = 8$, and $m = 37$ and 10, respectively. The BER curves of the corresponding codes C_3 and C_4 ($L = 300$) in Fig. 6.3(b) show improvements over their counterparts from [7] in the error floor region. The dominant TS classes of the codes of [7] are $\{(8, 0)\}$ and $\{(13, 4), (14, 2)\}$, respectively. These classes are, however, absent in C_3 and C_4, respectively.

As the last example, we consider two codes C_5 and C_6 both having $g = 10$ and $\gamma = 3$, and with $c = 4$ and $c = 5$, respectively. Similar to their counterparts in [78], the codes have $m = 11$ and $m = 19$, respectively. Simulation results show that the dominant TSs of the codes of [78] are in $(13, 3)$ and $(a, 4)$ classes with $a \leq 12$. We use the forest of Fig. 6.4 and Algorithm 2 with parameters $\{J = 200, k = 2\}$ and $\{J = 80, k = 1\}$, respectively, to design C_5 and C_6 with the following exponent matrices:

$$P_5 = \begin{bmatrix} 10 & 0 & 0 & 9 \\ 0 & 8 & 0 & 11 \\ 8 & 2 & 11 & 0 \end{bmatrix}, \quad P_6 = \begin{bmatrix} 19 & 14 & 2 & 0 & 1 \\ 0 & 4 & 19 & 9 & 12 \\ 0 & 0 & 3 & 19 & 19 \end{bmatrix}. \tag{6.1}$$

The LETS distribution of C_5 and C_6 for $L = 300$ along with that of their counterparts in [78] are given in Table 6.2. Table 6.2 shows the superior LETS distribution of C_5 and C_6 compared to the codes of [78]. In particular both codes are free of LETSs

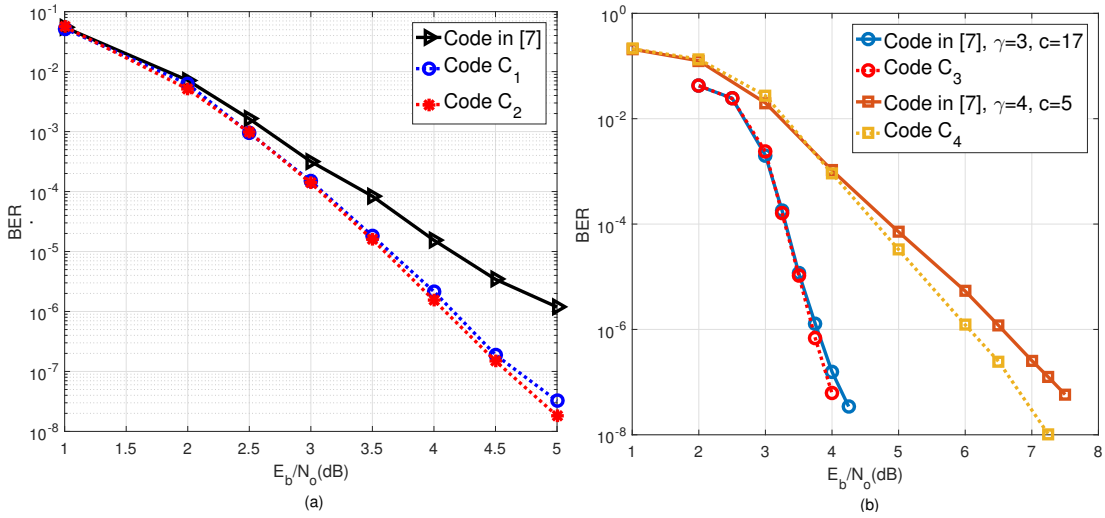


Figure 6.3: (a): Performance comparison of (a) designed codes C_1 and C_2 with the similar code of [7] ($W = 75$), (b) designed codes C_3 ($R = 0.8, N = 5100, W = 190$) and C_4 ($R = 0.17, N = 1500, W = 88$) with the similar codes of [7].

within the union of the two ranges $a \leq 14, b \leq 3$ and $a < 12, b \leq 4$.

To obtain our results, we implemented Algorithm 2 in MATLAB and ran it on a desktop computer with 4-GHz CPU and 32-GB RAM. The run-time to find the initial set Φ (Lines 2–22 of Algorithm 2) for $C_1 - C_6$ was about 8, 8, 20, 2, 6 and 0.5 hours, respectively. To obtain the final exponent matrix (Lines 23–32 of Algorithm 2), the run-time for $C_1 - C_6$ was about 95, 90, 289, 80, 45 and 40 minutes, respectively.

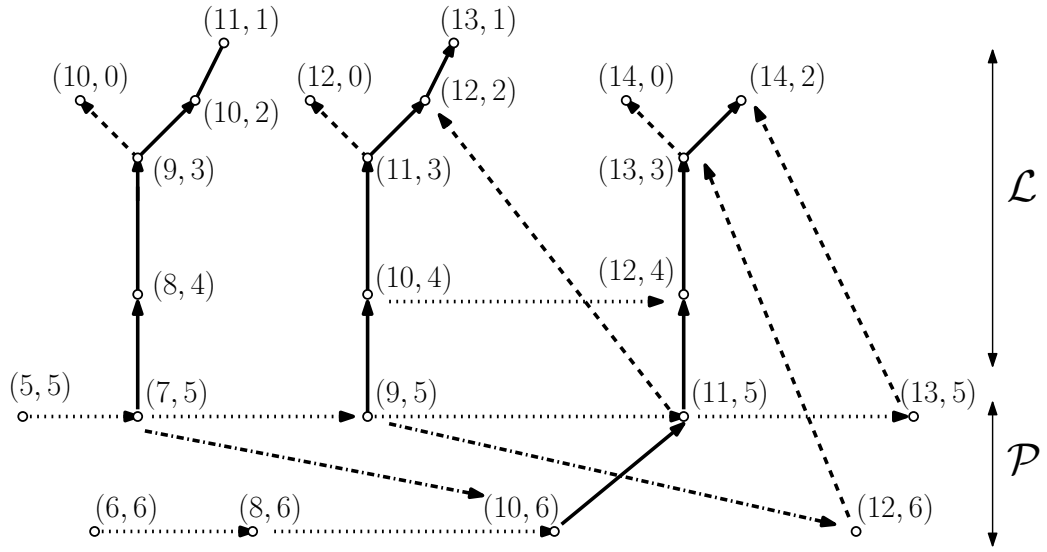


Figure 6.4: The forest used for the design of C_5 and C_6.

Table 6.2: Distribution of (a, b) LETSs in the range of $a \leq 14$ and $b \leq 5$ for C_5 ($N = 1200, R = 0.22$) and C_6 ($N = 1500$, $R = 0.362$) compared to that of their counterparts in [78].

LETS	Code in [78]	Code C_5	Code in [78]	Code C_6
$(5, 5)$	4606	3726	11777	10331
$(7, 5)$	2003	1142	5491	3310
$(8, 4)$	0	0	263	0
$(9, 5)$	5919	1391	16316	8191
$(10, 4)$	0	0	542	0
$(11, 5)$	16538	3891	42235	19941
$(12, 4)$	3341	282	3488	0
$(13, 3)$	821	0	791	0
$(13, 5)$	17816	9650	91459	59967
$(14, 4)$	3330	828	8280	3201

Chapter 7

Derivation of Bounds on the Memory and Lifting Degree of QC SC-LDPC Codes

7.1 Introduction

In Chapter 3 we studied the average distribution of cycles and TSs in a protograph-based SC-LDPC code corresponding to a $\gamma \times c$ spreading matrix B. In this chapter, again we consider protograph-based SC-LDPC codes with a $\gamma \times c$ spreading matrix. And, by considering either a separate design of spreading matrix and exponent matrix or a joint design of both, we derive bounds on the memory m and lifting degree M to achieve QC SC-LDPC codes with $g = 6$ or 8. Since the constraint length for QC SC-LDPC codes is defined as $\nu_s = (m+1)cM$, both m and M are important for the applications that the small constraint length is needed. In this respect, our theoretical analysis could give an insight into the impact of m and M for the design of QC SC-LDPC codes with a specific girth.

In this chapter, we start with brief preliminaries in Section 7.2. In Section 7.3, we describe the proposed scheme for the derivation of bounds in a joint assignment of memory and lifting degree. This is followed by Sections 7.4 and 7.5 where we fix either m or M and derive bounds for the other parameter. Some numerical results are presented in Section 7.6.

7.2 Preliminaries

Schmalen *et al.* in [76] introduced a number of design options to construct SC-LDPC codes and assessed how fast they are decodable by optimizing the contributing

parameters. As a part of their research, they considered underlying block code with parity check matrix H_{block} of size $d_1 \times d_2$ (d_1 and d_2 are in bits and the block code could be quasi-cyclic or not). The edge spreading process was performed to generate the corresponding SC-LDPC code, and they showed that the minimum distance in such SC codes with memory m and a given rate is upper bounded as

$$d_{min} \leq (d_1 + 1)(md_1 + 1). \tag{7.1}$$

The authors then proposed to construct SC codes made from either a larger size underlying block code (i.e., larger d_1) with smaller memory, known as *weakly coupled* or smaller size underlying block code (i.e., smaller d_1) with larger memory, referred to as *strongly coupled*. The reason behind their idea is to achieve a large minimum distance, according to (7.1), and maintain the decoding complexity low. This is because both the decoding complexity and the minimum distance increase by increasing either the value of d_1 or m. Therefore, they proposed that while one of these parameters (d_1 or m) increases, the other one decreases and thus, the decoding complexity does not increase.

Conducting simulations, it has been shown in [76] that weakly coupled codes (with smaller m and larger d_1) have better error floor performance compared to the strongly coupled codes, and thus they are more promising to be used in optical communications.[1] This is due to the fact that the upper bound in (7.1) increases quadratically with the parameter d_1, but only linearly with m, and thus increasing d_1 is a more effective way to increase the minimum distance. This observation motivated us to perform the analysis in this chapter by looking into the design of QC SC-LDPC codes. Moreover, as the decoding complexity and latency need to be small, it is vital to study the impact of memory m and lifting degree M on the constraint length of corresponding QC-SC codes. In this respect, we derive bounds on m and M to discuss the smallest required constraint length needed for achieving specific girths in the final QC SC-LDPC codes. Furthermore, the numerical results in this chapter also demonstrate the superiority of weakly coupled SC-LDPC codes compared to their strongly coupled counterparts, in terms of the error floor performance.

According to the above discussion, we need to derive bounds on m and M in order to achieve a specific girth of 6 or 8 in the constructed QC SC-LDPC code. Based on

[1] Note that the improvement in the error floor region comes at the expense of an inferior waterfall performance for such codes compared to the strongly coupled codes.

the preliminaries in Section 2.3.1, a given TBC walk W of length l in the base graph of a block code will correspond to a cycle of the same length in the resultant QC-SC LDPC code if and only if we have both Equation (2.5) of Lemma 1, and Equation (3.1) of Lemma 2 satisfied. In this respect, there exist two matrices, i.e., the spreading matrix B and the exponent matrix P associated with QC SC-LDPC codes. Thus, we need to assign entries to either the B matrix (with respect to m) or the P matrix (with respect to M) in different scenarios. In the first scenario, described in Section 7.3, we consider the joint design of P and B matrices (M and m). Then, in the other two scenarios, in Sections 7.4 and 7.5, either B matrix (and m) or P matrix (and M) is fixed and we give bounds on the other parameter that is not fixed.

7.3 Upper Bounds on Lifting Degree and Memory in QC-SC Codes

In this section, we present some upper bounds[2] for lifting degree and memory of QC-SC codes where the girth is 6. Note that our proposed approach in this section simultaneously utilizes two steps of the optimization algorithm. In other words, at the same round, we choose two pairs (M, P) and (m, B) and by introducing sufficient conditions, we discuss the values of M and m that could guarantee the design of QC-SC codes with girth 6.

The starting point for this construction is a block code base graph which is fully-connected with no parallel edges. Then, referring to the discussions in Section 2.3.1, the graph lifting process (based on the P matrix) and the edge spreading process (based on the B matrix) are applied over such a base graph. The ultimate goal is to achieve $g = 6$ in the corresponding QC SC-LDPC code.

7.3.1 Sufficient condition for girth 6

Theorem 11. *For any QC SC-LDPC code constructed with respect to a (γ, c) fully-connected block code base graph, with no parallel edges, a sufficient condition for having a girth of at least 6 is $(m+1)M \geq e(2\gamma-3)(2c-3)$, where e is Euler's number.*

[2]Note that as the approach proposed in Section 7.3 is based on considering the worst case scenario with respect to the probabilistic study of random bad events, the bounds are in fact considered as the upper bounds.

Proof. Consider a $\gamma \times c$ matrix D. In order to construct the QC-SC code it is enough to choose M, m and then assign a pair (x, y) to each cell of D, where $x \in [0, M-1]$ and $y \in [0, m]$. Assume that we picked m and M such that $(m+1)M \geq e(2\gamma-3)(2c-3)$, where e is Euler's number. For each cell of D independently pick two integers x and y uniformly at random from the intervals $[0, M-1]$ and $[0, m]$, respectively. Next, we show that with positive probability there exists a QC-SC code with girth at least 6.

For a base graph with no parallel edges, all the TBC walks of length less than or equal to 6 are cycles. So, in order to check Equation (2.5) of Lemma 1, and Equation (3.1) of Lemma 2, we only need to check all 4-cycles in the base graph. Each 4-cycle corresponds to a 4-tuple (i_1, i_2, j_1, j_2), where $i_1, i_2 \in [1, \gamma]$, $j_1, j_2 \in [1, c]$, $i_1 \neq i_2$ and $j_1 \neq j_2$. Call that 4-cycle $\mathcal{C}$. In fact the 4-cycle $\mathcal{C}$ uses the cells (i_1, j_1), (i_1, j_2), (i_2, j_1), and (i_2, j_2). The 4-cycle $\mathcal{C}$ will be mapped to a 4-cycle in the resultant QC-SC code if and only if we have both Equation (2.5) of Lemma 1, and Equation (3.1) of Lemma 2 satisfied for that cycle. For each 4-tuple (i_1, i_2, j_1, j_2) define the random bad event $A_{(i_1,i_2,j_1,j_2)}$ as having both of Lemma 1, and Lemma 2 satisfied for the cycle corresponding to that 4-tuple.

In [18], it was shown that under the assumption of i.i.d permutation shifts with uniform distribution, the probability of having Equation (2.5) of Lemma 1 satisfied for a 4-cycle is $1/M$. With the same argument we can see that the probability of having Equation (3.1) of Lemma 2 satisfied for a 4-cycle is upper bounded by $1/(m+1)$. Thus,

$$\Pr(A_{(i_1,i_2,j_1,j_2)}) \leq \frac{1}{(m+1)M}. \tag{7.2}$$

We say that two events $A_{(i_1,i_2,j_1,j_2)}$ and $A_{(i'_1,i'_2,j'_1,j'_2)}$ are dependent if and only if the corresponding cycles use a common cell of D. Next, we calculate that each event depends on at most how many other events. Consider the event $A_{(i_1,i_2,j_1,j_2)}$. This event uses the cells (i_1, j_1), (i_1, j_2), (i_2, j_1) and (i_2, j_2). Then, we have:

The number of events that use the cell (i_1, j_1): If we choose a row i' (other than row i_1) and a column j' (other than column j_1), we find an event. We have $\gamma - 1$ options for choosing the row and $c-1$ options for choosing the column. So, the total number of events that use the cell (i_1, j_1) is $(\gamma-1)(c-1)$. Noting that we also counted the event $A_{(i_1,i_2,j_1,j_2)}$, the number of events (other than $A_{(i_1,i_2,j_1,j_2)}$) that use the cell (i_1, j_1) is equal to

$$(\gamma-1)(c-1)-1. \tag{7.3}$$

The number of events that use the cell (i_1, j_2) and do not use the cell (i_1, j_1): In this case we should pick a row i' (other than row i_1) and a column j' (other than columns j_1 and j_2). So, we have $\gamma - 1$ options for choosing the row and $c - 2$ options for choosing the column. Thus, the total number is

$$(\gamma - 1)(c - 2). \tag{7.4}$$

The number of events that use the cell (i_2, j_1) and do not use the cells (i_1, j_1) and (i_1, j_2): In this case we should pick a row i' (other than rows i_1 and i_2) and a column j' (other than column j_1). So, we have $\gamma - 2$ options for choosing the row and $c - 1$ options for choosing the column. Thus, the total number is

$$(\gamma - 2)(c - 1). \tag{7.5}$$

The number of events that use the cell (i_2, j_2) and do not use the cells (i_1, j_1), (i_1, j_2) and (i_2, j_1): In this case we should pick a row i' (other than rows i_1 and i_2) and a column j' (other than columns j_1 and j_2). Therefore, we have $\gamma - 2$ options for choosing the row and $c - 2$ options for choosing the column. Thus, the total number is

$$(\gamma - 2)(c - 2). \tag{7.6}$$

By summing (7.3), (7.4), (7.5) and (7.6), we conclude that the event $A_{(i_1,i_2,j_1,j_2)}$ depends on

$$(2\gamma - 3)(2c - 3) - 1 \tag{7.7}$$

other events.

Now, assume that the base graph has t cycles of length four and for each 4-cycle C_i consider an event A_i. Next, we prove that $\Pr[\cap_{i=1}^{t} \bar{A}_i] > 0$. We use the Lovàsz Local Lemma that is a fundamental tool of the probabilistic method to prove the theorem.

Lemma 12. *[Symmetric Lovàsz Local Lemma [91]] Suppose $p \in (0, 1)$, $d \geq 1$, and $A_1, \ldots, A_t$ are events such that for each i, $\Pr(A_i) \leq p$. If each A_i is mutually independent of all but d other events A_j, and $ep(d + 1) \leq 1$, where $e = 2.71828$ is Euler's number, then $\Pr[\cap_{i=1}^{t} \bar{A}_i] > 0$.*

By (7.2), we have $p = 1/((m+1)M)$. Also, by (7.7), we have $d = (2\gamma-3)(2c-3)-1$.

By noting that $(m+1)M \geq e(2\gamma-3)(2c-3)$, we have $ep(d+1) \leq 1$, where e is Euler's number. Thus, by symmetric Lovàsz Local Lemma we have $\Pr[\cap_{i=1}^{t} \bar{A}_i] > 0$. This completes the proof. ■

7.4 Bounds on Lifting Degree in QC-SC Codes with a Fixed Memory

In this section, we study the construction process of QC SC-LDPC codes having a specific girth of 6 or 8 in two separate steps. Suppose that a protograph of an SC code with a fixed memory m and a given $\gamma \times c$ spreading matrix B has been designed. This base graph may have $g = 4$ or $g = 6$. Thus, we derive bounds on the size of the lifting degree M for such a base graph so that the corresponding QC SC-LDPC code has the girth of 6 or 8.

7.4.1 Necessary condition for girth 6

It is clear that when we obtain an SC base graph at the first stage of optimization such that it has no 4-cycle, the protograph has already $g = 6$. In this case, it is trivial that $M = 1$ is the smallest lifting degree to obtain a lifted graph with $g = 6$. Therefore, suppose that the optimized SC protograph, obtained from the first step of algorithm, has some 4-cycles. There is a well-known lower bound for the lifting degree of QC-LDPC codes to obtain the girth equals to six [44]. This lower bound works in the context of QC SC-LDPC codes as follows.

Consider the protograph of an SC code that we obtained in step 1 and let $\alpha(c_i, c_j)$ denote the number of positions at which both columns c_i and c_j have non-zero entries. Similarly, let $\alpha(r_i, r_j)$ be the number of positions at which both rows r_i and r_j have non-zero entries. In this respect, we have the following theorem:

Theorem 12. *[44] Consider a $\gamma \times c$ base matrix with memory m. Corresponding to this base matrix we can create an $x \times y$ SC code base graph with $x = (L+m)\gamma$ and $y = Lc$, where $L = m+1$. For any QC SC-LDPC code lifted from such a base graph with no parallel edges, a necessary condition for having a girth of at least 6 is that*

$$M \geq \alpha_{max} = max\{\alpha_{r,max}, \alpha_{c,max}\} + 1, \tag{7.8}$$

where $\alpha_{r,max} = max\{\alpha(r_i, r_j) - 1 : 1 \leq i, j \leq x, i \neq j\}$ *and* $\alpha_{c,max} = max\{\alpha(c_i, c_j) - 1 : 1 \leq i, j \leq y, i \neq j\}$.

Since the protograph of SC code is not a fully-connected graph, we cannot derive a closed form expression to calculate α_{max}. Instead, we find this parameter for each given SC protograph. Thus, given a spreading matrix B (leading to an SC base graph with $g = 4$), we construct the protograph of corresponding SC code using this B matrix and $L = m + 1$. Then, for every pair of r_i, r_j $(1 \leq i, j \leq (L + m)\gamma, i \neq j)$ and c_i, c_j $(1 \leq i, j \leq Lc, i \neq j)$ we calculate the parameter $\alpha(r_i, r_j)$ and $\alpha(c_i, c_j)$, respectively. Therefore, $\alpha_{r,max}$ and $\alpha_{c,max}$ are calculated and α_{max} is obtained accordingly. Using this algorithm we can find a lower bound for lifting degree M corresponding to each SC protograph to achieve $g \geq 6$ after performing the lifting process.

7.4.2 Necessary condition for girth 8

In this part, we discuss theoretical bounds for lifting degree to achieve $g \geq 8$ in the lifted graph of QC SC-LDPC codes, when an SC base graph is given. Depending on the values of m, γ and c, we may end up with an SC protograph having 4-cycle or being free of any 4-cycle. Thus, our approach here, is presented in two categories: SC protographs with or without 4-cycles.

SC protograph without 4-cycles

The protograph is free of any 4-cycle, so the smallest possible cycles in the protograph are 6-cycles. Note that as the girth of the base graph is 6, any TBC walk of length 6, 8 or 10 in the base graph is also a cycle [44]. Thus, we need to study the dependency between these small cycles in the base graph giving rise to the appearance of 6-cycles in the lifted graph. For this aim, we refer to Fig. 7.1, where the dependency between two cycles of the same length or different lengths sharing some edges to make a new 6-cycles, is shown. Thus, for each structure in Fig. 7.1 we decompose it in a way that a TBC walk of length 6 is identified, and then we discuss the relationship between permutation shifts of the contributing TBC walks giving rise to the appearance of a 6-cycle in the lifted graph.

• For the first case, let us describe the structure in Fig. 7.1(a). In this figure, there exist two 6-cycles with permutation shifts p_1 and p_2, having three edges (consisting

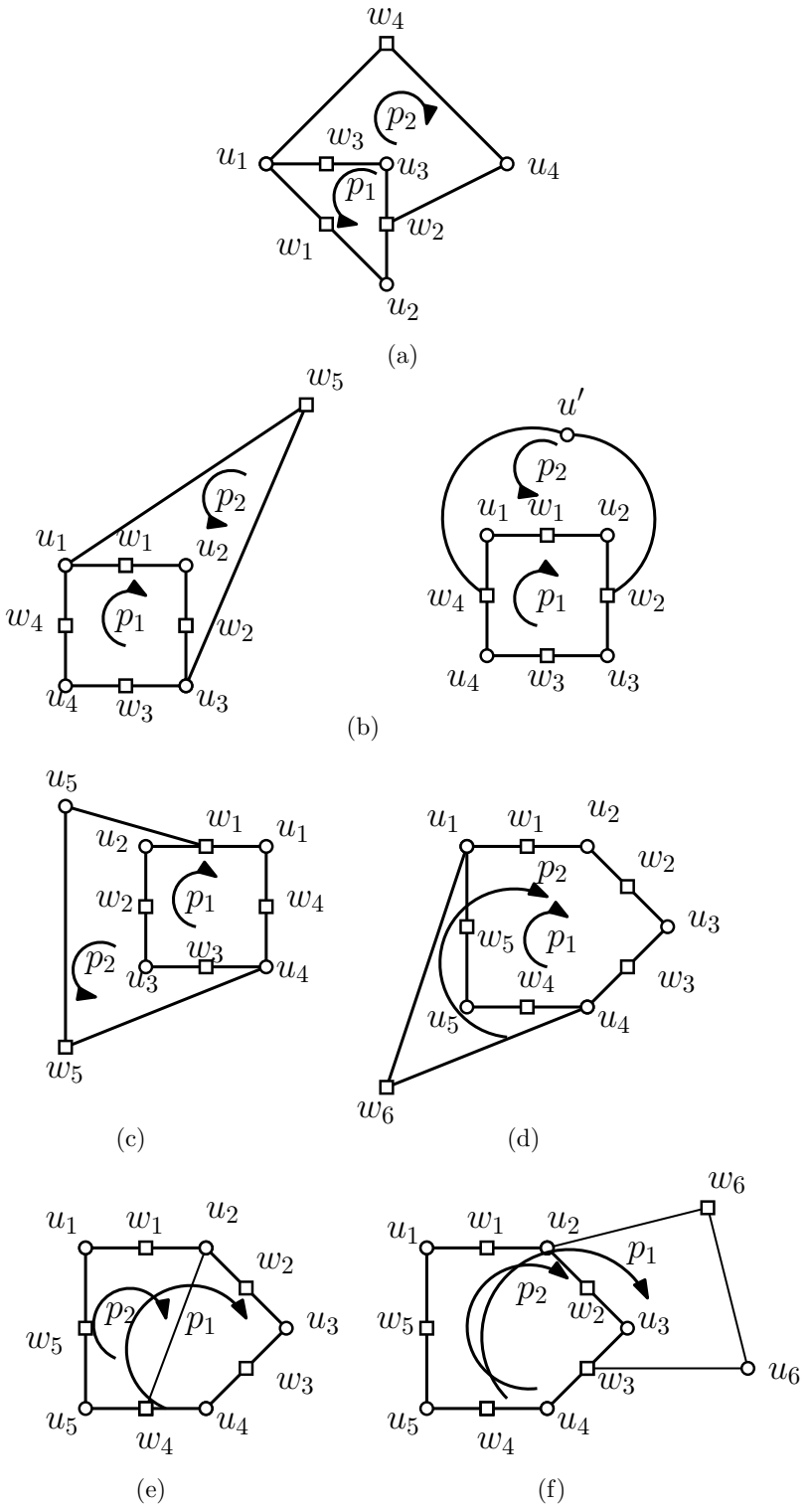


Figure 7.1: Structures in a base graph with $g = 6$ that could be mapped to cycles of length 6 in the lifted graph (a) Dependencies between two 6-cycles making another 6-cycle. (b) Dependencies between a pair of 6-cycle and 8-cycle making a new 6-cycle. (c) Dependencies between two 8-cycles making a 6-cycle. (d) Dependencies between a pair of 8-cycle and 10-cycle creating a 6-cycle. (e) Dependencies between a pair of 6-cycle and 10-cycle creating a new 6-cycle. (f) Dependencies between two 10-cycles creating a 6-cycle.

of u_1, w_3, u_3, w_2) in common. As a result of these common edges, another 6-cycle $u_1, w_1, u_2, w_2, u_4, w_4, u_1$ with $p_W \overset{M}{=} p_1 - p_2$ is created. In order to avoid 6-cycles made from this structure we must have $p_1 \neq 0$, $p_2 \neq 0$ and $p_1 \neq p_2$. If we denote every set of three edges (including a set of two variable nodes and two check nodes such as u_1, w_3, u_3, w_2) as e_3^k and the multiplicity of 6-cycles sharing e_3^k is referred to as num_{6_k}, we define $num_6^{max} = \max_k\{num_{6_k}\}$ as the maximum number of 6-cycles sharing three specific consecutive edges like e_3^k. As a result, lifting degree M must satisfy $M \geq num_6^{max} + 1$ to avoid 6-cycles.

Remark 5. *It is worth mentioning that for the base graph with $d_v = \gamma = 3$ and $g = 6$, the structure of Fig. 7.1(a) cannot exist in the base graph. This is because, $w_2 \neq \{w_1, w_3, w_4\}$, $w_1 \neq \{w_2, w_3, w_4\}$, $w_3 \neq \{w_1, w_2, w_4\}$ and $w_4 \neq \{w_1, w_2, w_3\}$. Thus, the indices for all four check nodes w_1, w_2, w_3 and w_4 must be distinct. This requires to have check nodes with four different row indices, which is not possible in a graph with $d_v = \gamma = 3$.*

• Now, consider the structure in Fig. 7.1(b) (left one), which illustrates an 8-cycle with permutation shift p_1 and a 6-cycle with permutation shift p_2 such that they share four consecutive edges including vertices u_1, w_1, u_2, w_2, u_3. One can verify that the combination of these two cycles makes another 6-cycle $u_1, w_4, u_4, w_3, u_3, w_5, u_1$ with permutation shift $p_W \overset{M}{=} p_2 - p_1$. As we need to avoid any cycle of length 6, we must have $p_2 \neq 0$ and $p_1 \neq p_2$. This implies that $p_1 \in [0, M-1]$ and $p_2 \in [1, M-1] \backslash p_1$, and the lower bound for M to satisfy these conditions is derived as follows. According to the discussion above, let us consider four consecutive edges such as the one in Fig. 7.1(b), which is made of nodes u_1, w_1, u_2, w_2, u_3 and denote this set of edges by e_4^k. (equivalently, this set of edges is identified by four distinct entries of the B matrix, including two rows and three columns of B.) Corresponding to e_4^k, we can find if there exists a pair of 6-cycle (with permutation shift p_2) and 8-cycle (with permutation shift p_1) sharing this set of four edges. In case that there exists at least one set of edges like e_4^k, which is in common with a pair of 6 and 8-cycle to create this structure, we must have $M \geq 2$. The last inequality comes from the fact that $p_2 \in [1, M-1] \backslash p_1$. So, if p_1 takes any value in $[0, M-1]$, there exist totally $M-2$ eligible values to be assigned to p_2 if $p_1 \neq 0$, or there exist $M-1$ eligible values to be assigned to p_2 if $p_1 = 0$. Thus, as the union of two ranges $M - 2 \geq 1$ and $M - 1 \geq 1$, we conclude that $M \geq 2$. Therefore, the existence of at least one instance of the left structure of Fig. 7.1(b) in the SC protograph results in the lifting degree $M \geq 2$ as the necessary

condition to achieve $g \geq 8$ in the lifted graph. Regarding the structure on the right side of Fig. 7.1(b), one can see that two cycles of length 8 and 6, with permutation shifts p_1 and p_2, respectively, share four consecutive edges including w_2, u_2, w_1, u_1, w_4. As a result, we must have $p_2 \neq 0$ and $p_1 \neq p_2$, which means that p_1 can take M different values and p_2 takes $M-2$ (or $M-1$) different values, if $p_1 \neq 0$ (or $p_1 = 0$). Thus, if there exists at least one instance of Fig. 7.1(b) in a graph, it necessitates that $M \geq 2$.

In the following Remark we describe how the structure on the right side of Fig. 7.1(b) could appear in an SC code with $d_v = \gamma = 3$.

Remark 6. *It is clear that when $d_v = \gamma = 3$, there are at most three distinct row indices for check nodes in the graph. On the other hand, for the structure on the right side of Fig. 7.1(b) we must have $w_1 \neq w_2$, $w_1 \neq w_4$ and $w_2 \neq w_4$. Nevertheless, as two check nodes w_1 and w_3 do not share any common neighboring variable nodes, we could have $w_1 \overset{\gamma}{=} w_3$ (i.e., w_1 and w_3 have the same remainder after division by γ) and it implies that these two check nodes stem from the same row in the underlying block code base matrix (or equivalently in the spreading matrix B), and during the edge spreading process they are located in two different components H_i and H_j ($i \neq j$). To clarify it better, we refer to Fig. 7.2 at which one possible configuration of an 8-cycle in Fig. 7.1(b), when $\gamma = 3$, is depicted. One can observe that check nodes w_1 and w_3 belong to the same row of the block code base matrix, but they move to two different rows (and two different components H_0 and H_2) in the corresponding SC base matrix. Thus, we see that although there exist at most 3 different check node indices in the block code base graph, when converting it to the SC code, we have more number of check nodes in the SC base graph such that they originally stem from these 3 row indices in the block code base graph.*

• Next, we consider Fig. 7.1(c), at which two 8-cycles C_1 and C_2 with permutation shifts p_1 and p_2 share five consecutive edges including $w_1, u_2, w_2, u_3, w_3, u_4$. As a result of this combination we find a new 6-cycle $w_1, u_1, w_4, u_4, w_5, u_5, w_1$ with permutation shift $p_W \overset{M}{=} p_1 - p_2$. Thus, if $p_1 \neq p_2$, we avoid such a 6-cycle. It implies that $p_1 \in [0, M-1]$ and $p_2 \in [0, M-1] \backslash p_1$. Therefore, similar to the five consecutive edges in Fig. 7.1(c) containing nodes $w_1, u_2, w_2, u_3, w_3, u_4$, we consider every set of five consecutive edges in the graph and denote it by e_5^k. If we define num_{8_k} as the multiplicity of all distinct 8-cycles sharing e_5^k with each other, the maximum value of

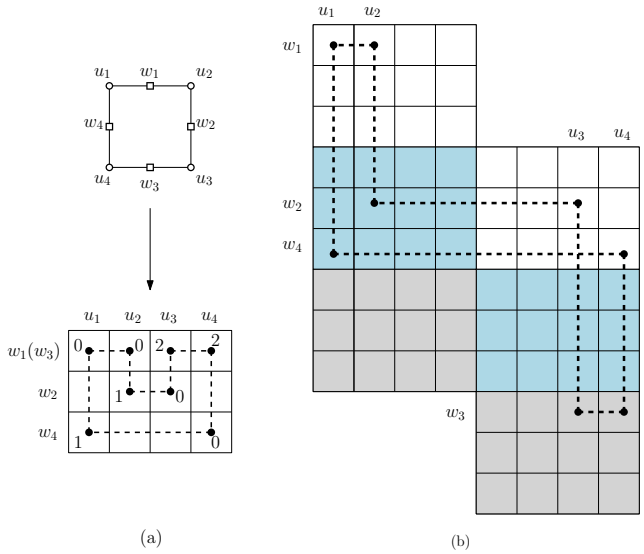


Figure 7.2: a) An 8-cycle mapped to the block base matrix with the entries of spreading matrix is given for each edge. (b) The configuration of 8-cycle (given in Fig. 7.2(a)) in the SC code base matrix, after applying the edge spreading process with $m = 2$. Components H_0, H_1 and H_2 are colored white, blue and gray, respectively.

num_{8_k} over all possible sets of five edges (e_5^k) is denoted by $num_8^{max} = \max_k\{num_{8_k}\}$. Thus, we have at most num_8^{max} number of distinct 8-cycles sharing five consecutive edges like e_5^k. Each of these cycles can take M different permutation shifts in $[0, M-1]$. In order to avoid 6-cycles made through the combination of these 8-cycles, we must have $M \geq num_8^{max}$.

In order to calculate num_8^{max} we need to find 8-cycles and check them according to their common edges. For this, we can create the protograph for the corresponding SC code with the given B matrix, and $L = 2m + 1$. Then, we count 8-cycles in the SC protograph. Having the set of all nodes involved in these counted 8-cycles, it is easy to investigate how many 8-cycles share a specific set of consecutive edges such as e_5^k in a given SC base graph. The maximum of these numbers are then considered as num_8^{max}.

• As the next structure, consider Fig. 7.1(d) at which a pair of cycles, including a 10-cycle and an 8-cycle, share six specific consecutive edges including vertices $u_1, w_1, u_2, w_2, u_3, w_3, u_4$. If we denote the permutation shift of contributing 10-cycle s_5 by p_1 and that of the corresponding 8-cycle s_4 by p_2, we end up with a new 6-cycle $u_1, w_5, u_5, w_4, u_4, w_6, u_1$ with permutation shift $p_1 - p_2$. Thus, we must have $p_1 \neq p_2$ to avoid the appearance of such a 6-cycle in the lifted graph. Therefore, if $p_1 \in [0, M-1]$, we must have $p_2 \in [0, M-1] \backslash p_1$. Referring to what we discussed for Fig. 7.1(b), once

a pair of 8-cycle and 10-cycle sharing six edges (similar to Fig. 7.1(d)) is found for a given SC protograph, we conclude that $M \geq 2$.

• The next candidate is Fig. 7.1(e), which illustrates the combination of a 10-cycle and a 6-cycle generating a new 6-cycle with permutation shift $p_W \overset{M}{=} p_1 - p_2$. Here, we need to have $p_2 \neq 0$ and $p_1 \neq p_2$. Thus, $p_1 \in [0, M-1]$ and $p_2 \in [1, M-1] \backslash p_1$. This implies that once a pair of 10-cycle and 6-cycle is found in a graph to create the structure in Fig. 7.1(e), we have $M \geq 2$. This comes with assigning $p_1 = 0$ and $p_2 = 1$, which leads to the smallest possible lifting degree as a necessary condition to satisfy the girth $g \geq 8$.

• Finally we consider Fig. 7.1(f). In this structure we have a pair of 10-cycles sharing 7 edges to create a new 6-cycle with permutation shift $p_W = p_1 - p_2$. In order to eliminate such a 6-cycle in the lifted SC graph we must have $p_1 \neq p_2$. Therefore, having at most num_{10}^{max} number of 10-cycles sharing a specific set of 7 edges requires that $M \geq num_{10}^{max}$. To find num_{10}^{max} we can create the base graph of the corresponding SC code with respect to the given B matrix and $L = 2m + 1$.[3] Then, we detect 10-cycles in this base graph, and now it is easy to find the maximum number of such 10-cycles sharing specific set of 7 edges.

In conclusion, we have the following theorem.

Theorem 13. *Consider a QC SC-LDPC code which is constructed based on a $\gamma \times c$ block code base graph with no parallel edges through a 2-step construction approach, where in the first step we choose the memory m and the spreading matrix B to design the SC protograph. Suppose that the resultant SC protograph is free of any 4-cycle and we want to have girth 8 in the corresponding QC-SC code. Then for $d_v = \gamma > 3$, the lifting degree M should meet the inequality $M \geq \max\{2, num_6^{max} + 1, num_8^{max}, num_{10}^{max}\}$. Also, if $d_v = \gamma = 3$, we have $M \geq \max\{2, num_8^{max}, num_{10}^{max}\}$, where num_6^{max}, num_8^{max} and num_{10}^{max} denote the maximum number of distinct 6-cycles sharing three edges as in Fig. 7.1(a), maximum number of 8-cycles sharing five edges as in Fig. 7.1(c), and the maximum number of 10-cycles sharing seven edges as shown in Fig. 7.1(f), respectively.*

Proof. The proof is clear based on the above discussion. ■

The lower bound derived above generally depends on num_6^{max}, num_8^{max}, num_{10}^{max} and the existence of structures in Fig. 7.1(b), 7.1(d) and 7.1(e). This implies that

[3] According to Lemma 3, the value of L is chosen such that the corresponding base graph covers all the variable nodes involved in a 10-cycle.

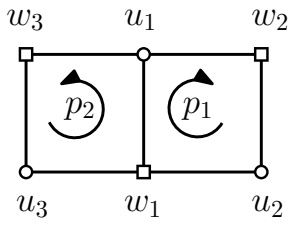


Figure 7.3: Two 4-cycles in the base graph sharing one edge [44]. Circles and squares correspond to the variable nodes and check nodes, respectively.

such a lower bound is graph-dependent and for every given SC base graph, with $g = 6$, we need to obtain a lower bound for lifting degree to achieve $g = 8$ in the corresponding lifted SC graph. Thus, if we have a B matrix eliminating all 4-cycles in the corresponding SC protograph, we need to find the lower bound for lifting degree using the above approach. Starting with this lower bound, we can assign permutation shifts accordingly in the second step of construction to achieve a QC SC-LDPC code with $g \geq 8$.

SC protograph with 4-cycles

In this case again we use a well-known lower bound that was presented for QC-LDPC codes in [44]. This approach is based on the fact that in Fig. 7.3, we must have $p_1 \neq 0$, $p_2 \neq 0$ and $p_1 \neq p_2$ to avoid any 4-cycle or 6-cycle in the corresponding lifted graph. The lower bound needs to be modified for base graphs that are not fully-connected (such as SC base graphs). Then it will be used as the starting point to search for an exponent matrix P with a small M satisfying the condition $g \geq 8$ in the corresponding QC SC-LDPC code.

Theorem 14. *[44] For any QC protograph code constructed using a base graph with no parallel edges, a necessary condition for having a girth of at least 8 is that the lifting degree M must satisfy $M \geq \max_i\{N_4^{e_i}\} + 1$, where the maximum is taken over all the edges of the base graph and $N_4^{e_i}$ is the number of distinct 4-cycles containing the edge e_i.*

In order to use Theorem 14 for SC codes we need to consider every edge like e_i in the SC code base graph with $L = m + 1$ (as the smallest coupling length to cover all the variable nodes of every 4-cycle in the graph). Associated with this base graph and referring to the edge spreading process discussed in Chapter 2, we have a $\gamma \times c$ spreading matrix B, where every edge like e_i corresponds to an entry of this matrix.

Then, we need to inspect the B matrix and find the number of unbroken 4-cycles including this specific entry of B (or equivalently the edge e_i in SC base graph). If we denote this number by $N_4^{e_i}$, it is easy to find the maximum of all these $N_4^{e_i}$ values for all different γc entries of B. And, the lower bound is achieved as $\max\{N_4^{e_i}\} + 1$.

7.4.3 Sufficient condition for girth 8

Here, we discuss how to find an upper bound for the lifting degree to satisfy $g \geq 8$ when there is no cycle of length 4 in the SC protograph. In this regard, we present the following proposition [95].

Proposition 15. *Suppose that we are in the process of choosing a proper non-negative value for the i-th row and j-th column ($P_{i,j}$) of a $\gamma \times c$ exponent matrix P corresponding to an SC base graph with $g = 6$. If for any $P_{i,j}$, the number of TBC walks of length smaller than 8 in the block code base graph having $p_W \overset{M}{=} 0$ with respect to exponent matrix P is smaller than M, then for each $P_{i,j}$ we can find at least one permutation shift in $\{0, 1, ..., M-1\}$ to be assigned to it such that no relation for $l < 8$ as given in (2.4) leads to $p_W \overset{M}{=} 0$, and thus we achieve $g \geq 8$ in the lifted SC graph.*

Now, we describe how to utilize Proposition 15 for the derivation of upper bounds on lifting degree to achieve QC SC-LDPC codes with $g \geq 8$, having an SC base graph with $g = 6$. Associated with this base graph, we have a $\gamma \times c$ spreading matrix B, and we need to assign the permutation shifts $P_{i,j}$ for the corresponding $\gamma \times c$ exponent matrix P. For each entry $e_{i,j}$ in the B matrix we count the number of 6-cycles involving that entry.[4] Suppose that e_{max} is an entry (edge) with the largest contribution to creating 6-cycles and it appears in n_{max} 6-cycles. Thus, according to Proposition 15, it suffices to consider $M \geq n_{max} + 1$ to guarantee that there exists at least one permutation shift for each $P_{i,j}$ such that all 6-cycles are removed from the lifted graph. This could be interpreted as an upper bound for the lifting degree as we consider the sufficient condition for the removal of all 6-cycles.

In conclusion, in order to find the upper bound for each case, we need to examine the given spreading matrix B and find the contribution of each entry (or each edge

[4]This could be interpreted as the number of 6-cycles that are not broken according to the given spreading matrix B and include the entry $e_{i,j}$ in the corresponding $\gamma \times c$ block code base matrix. In fact, such an entry then appears as an edge in the SC base graph considering the given B matrix and $L \geq m + 1$.

of the block code base graph) to creating 6-cycles. Then, according to the value of n_{max}, an upper bound for lifting degree to achieve $g \geq 8$ is proposed

7.5 Bounds on Memory in QC-SC Codes with a Fixed Lifting Degree

In this part, we fix the lifting degree to be M, so that for a given QC-LDPC block code, lifted from a fully-connected base graph with lifting degree M, we derive lower bounds on the memory m to obtain a specific girth in the corresponding QC SC-LDPC code after performing the edge spreading process.[5] In fact, we extend the results of [7] on the derivation of bounds for m in Type-1 SC codes to obtain a specific girth $g = 6$ or $g = 8$. One can verify that the bounds in [7] are a special class of our proposed bounds when $M = 1$ and thus, the block code is assumed to have a $\gamma \times c$ all-one parity check matrix (i.e., having $M = 1$, the exponent matrix of a fully-connected $\gamma \times c$ base graph is a $\gamma \times c$ all-zero matrix.).

Definition 6. *Corresponding to an exponent matrix P, difference matrix ΔP of size $\binom{\gamma}{2} \times c$ is defined as*

$$\Delta P = \begin{bmatrix} \Delta p^1_{1,2} & \Delta p^2_{1,2} & \dots & \Delta p^c_{1,2} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta p^1_{1,\gamma} & \Delta p^2_{1,\gamma} & \dots & \Delta p^c_{1,\gamma} \\ \Delta p^1_{2,3} & \Delta p^2_{2,3} & \dots & \Delta p^c_{2,3} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta p^1_{2,\gamma} & \Delta p^2_{2,\gamma} & \dots & \Delta p^c_{2,\gamma} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta p^1_{\gamma-1,\gamma} & \Delta p^2_{\gamma-1,\gamma} & \dots & \Delta p^c_{\gamma-1,\gamma} \end{bmatrix}, \tag{7.9}$$

where $\Delta p^j_{i_k,i_t} \overset{M}{=} P_{i_k,j} - P_{i_t,j}, \forall j \in [1, c], \forall i_k, i_t \in [1, \gamma]$, $i_k < i_t$.[6]

[5] Note that the edge spreading based on cutting vectors is a special case of this work.

[6] We use the notation $P_{i_k,j}$ for the entry of P matrix located at the i_k-th row and j-th column.

It should be noted that the concept of difference matrix is also introduced in the literature to study the girth properties of QC-LDPC codes, e.g., see [2].

7.5.1 Bound derivation for $g \geq 6$

Lemma 13. *A QC SC-LDPC code made from a QC-LDPC code with parity check matrix $H(\gamma, c)$ must satisfy*

$$m \geq \left\lceil \frac{c' - 1}{2} \right\rceil, \tag{7.10}$$

to have $g \geq 6$. Here $c' = \max_i\{R_i\}$, where R_i corresponds to the maximum number of times that an element appears in the i-th row of matrix ΔP, and the maximum is taken over all rows of ΔP such that $1 \leq i \leq \binom{\gamma}{2}$.

Proof. In order to satisfy $g \geq 6$ in a QC SC-LDPC code, for any 4-cycle existed in the underlying QC-LDPC code, (3.1) should not hold true (i.e., the summation of differential parameters must be non-zero). Considering all rows of ΔP matrix, assume that there exist at most c' repetitions over all rows of ΔP, where $c' \leq c$. This implies that we have at most c' distinct 4-cycles rising from every two pairs of rows in P matrix. We can associate a vertical differential parameter $\Delta B^v_{j_1}(i_1, i_2) = B_{i_1,j_1} - B_{i_2,j_1} \in [-m, m]$ with every entry $\Delta p^{j_1}_{i_1,i_2}$ of difference matrix ΔP. Thus, referring to Lemma 2 and the pigeonhole principle, a necessary condition to avoid any 4-cycle in the corresponding QC SC-LDPC code is that $2m + 1 \geq c'$, or equivalently, $m \geq \lceil \frac{c'-1}{2} \rceil$. ∎

Corollary 6. *Considering $M = 1$ as the special case of Lemma 13. In this case, ΔP is an all-zero $\binom{\gamma}{2} \times c$ matrix and we conclude that $c' = c$. Thus, the same result as that of [7] is obtained and $m \geq \lceil \frac{c-1}{2} \rceil$.*

From Lemma 13 it is evident that the lower bound is graph-based. Thus, for a given QC-LDPC code with a fixed lifting degree M and an exponent matrix P, we can find the lower bound on m to achieve $g \geq 6$ in the corresponding QC SC-LDPC code. In the following example, we describe how to find the lower bound based on Lemma 13.

Example 6. *Suppose that we have a QC-LDPC block code with lifting degree $M = 3$*

and the following P matrix

$$P = \begin{bmatrix} 0 & 1 & 1 & 2 & 1 & 2 \\ 0 & 1 & 2 & 0 & 0 & 1 \\ 1 & 0 & 2 & 2 & 2 & 2 \end{bmatrix}. \tag{7.11}$$

Corresponding to (7.11) we have the following difference matrix

$$\Delta P = \begin{bmatrix} 0 & 0 & 2 & 2 & 1 & 1 \\ 2 & 1 & 2 & 0 & 2 & 0 \\ 2 & 1 & 0 & 1 & 1 & 2 \end{bmatrix}. \tag{7.12}$$

From (7.12) one can see that $R_1 = 2, R_2 = 3$ and $R_3 = 3$. Thus, $c' = 3$ (maximum number of repetitions occurs at the second and third row with $R_3 = 3$), and we must have $m \geq \left\lceil \frac{c'-1}{2} \right\rceil = 1$ to achieve $g \geq 6$ in the corresponding QC SC-LDPC code.

7.5.2 Bound derivation for $g \geq 8$

Lemma 14. *A necessary condition for a QC SC-LDPC code made from a QC-LDPC block code with parity check matrix $H(\gamma, c)$ and $\gamma = 3$,[7] to have $g \geq 8$ is that*

$$m \geq \left\lceil \frac{t}{4} \right\rceil. \tag{7.13}$$

Here, t denotes the maximum number of different summations such as

$$\Delta p_{1,3}^{j_1} = \Delta p_{1,2}^{j_2} + \Delta p_{2,3}^{j_3}, \tag{7.14}$$

over different choices of $j_1 \in [1, c]$ and $j_2 \neq j_3 \neq j_1$.

Proof. The proof is similar to the proof of [7, Lemma 6], such that we need to consider different cases leading to the creation of 4-cycles or 6-cycles in the corresponding QC SC-LDPC code. This depends on the difference matrix ΔP. Therefore, we need to find the maximum number of summations such as (7.14) that exist in ΔP for the

[7] It should be noted that the lower bounds for m proposed in [7] for the construction of SC codes with $g = 8$ was also limited to SC codes with variable node degree 3.

given QC-LDPC block code, and denote this number by t. With every two entries of ΔP such as $\Delta p_{1,2}^{j_2}$ and $\Delta p_{2,3}^{j_3}$, we can associate the vertical differential parameters $\Delta B_{j_2}^v(1,2)$ and $\Delta B_{j_3}^v(2,3)$ where it is known that $\Delta B_{j_2}^v(1,2)+\Delta B_{j_3}^v(2,3) \in [-2m, 2m]$. As we need to satisfy $\Delta B_{j_1}^v(1,3) \neq \Delta B_{j_2}^v(1,2) + \Delta B_{j_3}^v(2,3)$ to avoid 6-cycles, and $\Delta B_{j_1}^v(1,3) \neq \Delta B_{j'}^v(1,3)$, where $j' \in [1,c]\backslash j_1$, to avoid 4-cycles,[8] these t different sums (such as the one in (7.14)) can assume $4m$ values. Thus, we conclude that $4m \geq t$, or equivalently, $m \geq \lceil \frac{t}{4} \rceil$. ■

Corollary 7. *Considering lifting degree $M = 1$ implies that $\Delta p_{i,i'}^{j} = 0, \forall i, i' \in [1,3], j \in [1,c]$, and thus, the summation of every two pairs of ΔP such as $\Delta p_{1,2}^{j_2}$ and $\Delta p_{2,3}^{j_3}$ is zero and equals to $\Delta p_{1,3}^{j_1}$. Thus, there exist totally $\binom{c}{2}$ different summations such as (7.14), corresponding to the choices of j_1 and j_2 among the total number of c columns. Thus, $t = \binom{c}{2}$, and $m \geq \left\lceil \frac{c(c-1)}{8} \right\rceil$, which is the same as the result of [7, Lemma 6] for SC codes with $\gamma = 3$ having $g = 8$.*

Example 6 (Continued)**.** *Referring to the P matrix and ΔP matrix given in (7.11) and (7.12), one can see that the maximum number of summations in the form of (7.14) is equal to 8, which corresponds to $j_1 = 3$ and the following 8 pairs of (j_2, j_3):*

$$(j_2, j_3) \in \{(1,6),(2,1),(2,6),(5,2),(5,4),(6,2),(6,4),(6,5)\}. \tag{7.15}$$

As an example, for $j_1 = 3$ and $(j_2, j_3) = (1,6)$, we have $\Delta p_{1,3}^{j_1} = 2, \Delta p_{1,2}^{j_2} = 0$ and $\Delta p_{2,3}^{j_3} = 2$ which results in $\Delta p_{1,3}^{j_1} = \Delta p_{1,2}^{j_2} + \Delta p_{2,3}^{j_3}$. The same results would also hold for the rest of the pairs in (7.15). As a result of the above discussion and the fact that $t = 8$, we have $m \geq \lceil \frac{8}{4} \rceil = 2$ to achieve the corresponding QC SC-LDPC code with $g \geq 8$.

7.6 Numerical Results and Comparison with Existing Codes

In this part, we provide some numerical results to check our theoretical bounds presented in Sections 7.3-7.5. According to each set of bounds, we present some numerical

[8] Here we assumed that there exists at least one entry like $\Delta p_{1,3}^{j'}$ in the difference matrix ΔP such that $\Delta p_{1,3}^{j_1} = \Delta p_{1,3}^{j'}$ and thus, we need to have $\Delta B_{j_1}^v(1,3) \neq \Delta B_{j'}^v(1,3)$. Otherwise, these t different summations could have $4m+1$ different values.

results for the values of lifting degree and memory of the corresponding QC-SC codes. Our numerical results in this section all correspond to the fully-connected block base graphs and compare the theoretical results (bounds on m and M) provided in Sections 7.3-7.5 with some practical results based on a computer search. The heuristic search algorithms were implemented in MATLAB and ran on a desktop computer with 4 GHz CPU and 32 GB RAM. We also note that the bounds provided in Sections 7.3-7.5 work based on the application. In other words, if the base matrix of SC code (and accordingly, the spreading matrix) is given, we need to use the bounds on M introduced in Section 7.4. Otherwise, for a given QC-LDPC code, the bounds on m given in Section 7.5 would be applicable to reach a certain girth in the final QC SC-LDPC code. In general, all of these bounds target to achieve a small constraint length in the corresponding QC SC-LDPC code.

7.6.1 QC-SC codes with a fixed memory and varying lifting degree

We consider two types of SC base graphs including base graphs with $g = 4$ and $g = 6$. According to the approach given for each of these two cases, we may end up with different lower and/or upper bounds for the lifting degree based on Theorems 14 or 13. Also, as the proposed bounds are graph-based, different SC protographs could have different lower and upper bounds for lifting degree to achieve a specific girth.

Table 7.1: Properties of constructed SC codes with different values of m and the corresponding bounds for lifting degree M to obtain a girth ($g = 6$ or 8). All spreading matrices are optimized to have a minimum number of 4-cycles.

	Code C_1	Code C_2	Code C_3	Code C_4	Code C_5	Code C_6
$\gamma \times c$	3×4	3×4	3×5	3×6	4×6	3×9
m	1	2	2	4	5	5
B matrix	1 0 1 1 0 1 0 1 0 1 1 0	1 0 2 2 0 2 0 2 2 2 1 0	0 0 2 1 2 2 1 0 0 2 1 2 2 0 0	0 4 0 3 1 4 1 3 4 0 4 0 4 0 3 4 0 1	0 5 4 3 4 1 4 1 1 5 3 2 3 5 3 0 0 3 3 1 2 3 5 5	4 4 5 2 0 0 2 3 0 1 0 0 3 5 2 0 2 4 0 5 0 0 0 4 4 0 5
Girth of the base graph	4	6	6	6	6	6
Lower bound of lifting degree to achieve $g = 8$ (Theorems 13 or 14)	3	2	2	2	3	2
Upper bound of lifting degree to achieve $g = 8$ (Proposition 15)	(6+1)=7	$(3 + 1) = 4$	$(7 + 1) = 8$	(2+1)=3	(13+1)=14	$(11+1) = 12$
Minimum lifting degree to achieve $g = 8$ (in practice)	3	2	3	3	4	3
Minimum constraint length (ν_s) for girth $g = 8$ (theory)	24	24	30	60	108	108
Minimum constraint length (ν_s) for girth $g = 8$ (in practice)	24	24	45	90	144	162

Table 7.1 presents the corresponding results for a variety of spreading matrices. It

should be noted that the base graph for code C_1 has $g = 4$, while, for other codes C_2-C_6, the SC base graph has $g = 6$. In all these codes, an attempt was made to obtain a spreading matrix which minimizes the multiplicity of 4-cycles in the corresponding SC protograph.[9] In Table 7.2, however, the spreading matrices are just random and no optimization is performed on them to reduce the number of short cycles in their corresponding SC protographs. For the codes in Table 7.2, we consider arbitrary m values and randomly create B matrices. The corresponding SC protographs have $g = 4$, and so, we present the lower bounds for lifting degree to achieve $g = 6$ and $g = 8$. The theoretical bounds in both tables are then compared with the values of lifting degree found in practice through a heuristic computer search (with at most one hour search-time). From Tables 7.1-7.2 it is evident that our theoretical bounds on lifting degree in most cases are close to the practical values found through a heuristic search, so the bounds are almost tight.[10] The search process is performed by starting from the lower bounds on M, proposed in Section 7.4. Then, if no P matrix is found within 1 hour of search, we increase M by one and keep searching. As a result of this computer search we end up with the smallest values of M associated with a P matrix resulting in QC SC-LDPC codes with $g \geq 6$ or $g \geq 8$. We also calculate the constraint length for our designed codes in the tables, based on the equation $\nu_s = (m+1)cM$. Comparing Tables 7.1 and 7.2 it is clear that optimizing one step of the design process (i.e., the first step for the design of B matrix in Table 7.1) results in a smaller constraint length to achieve a certain girth of 8 in the corresponding QC SC-LDPC code.

As another example, we refer to the code with $\gamma = 3, c = 7, M = 7, m = 1$ and $g = 6$ designed in [26]. This code was optimized to have the minimum number of 6-cycles in the lifted graph. In this regard, first the spreading matrix was designed to minimize the multiplicity of 6-cycles in the corresponding SC base graph, while the girth of the base graph is still 4 and no consideration was performed over the

[9] In order to minimize the multiplicity of 4-cycles we start from the lower bound on m proposed in [7] for achieving an SC base graph with $g = 6$. Then, a heuristic search for a limited period of time is performed over each value of m and if no code is found, we increase m by one and keep searching until an SC base graph with $g = 6$ and a small m is obtained.

[10] As is evident from Table 7.1, base graph for C_1 has $g = 4$, and we also need to count the contribution of every edge to the appearance of 4-cycles in order to derive an upper bound on M for this code based on Proposition 15. Since the contribution of edges to 4-cycles are smaller compared to the contribution of edges to the appearance of 6-cycles, we consider the latter for the derivation of upper bound for this case.

Table 7.2: Properties of constructed SC codes with different values of m and the corresponding bounds for lifting degree M to obtain a girth ($g = 6$ or 8). Spreading matrices are randomly constructed with a given m.

	Code C_7	Code C_8	Code C_9
$\gamma \times c$	4×6	3×9	3×10
m	3	4	4
B matrix	3 2 2 1 0 2 3 0 1 1 0 2 1 3 1 2 0 1 0 3 3 1 2 1	3 3 1 3 0 3 0 1 0 0 4 3 1 4 0 2 1 3 3 0 2 2 3 3 3 1 3	4 4 0 4 3 0 1 2 4 4 0 4 4 2 4 0 2 4 3 4 3 0 4 4 3 3 3 1 3 0
Girth of the base graph	4	4	4
Lower bound of lifting degree to achieve $g = 6$ (Theorem 12)	4	3	3
Lower bound of lifting degree to achieve $g = 8$ (Theorem 14)	6	4	4
Minimum lifting degree to achieve $g = 6$ in practice	4	3	3
Minimum lifting degree to achieve $g = 8$ in practice	6	6	6
Constraint length (ν_s) to achieve $g = 8$ in theory	144	180	200
Constraint length (ν_s) to achieve $g = 8$ in practice	168	270	300

multiplicity of 4-cycles in the SC base graph. Then, at the second step, the authors proposed an exponent matrix with optimized permutation shifts to further minimize the multiplicity of 6-cycles and remove all the existing 4-cycles in the corresponding QC SC-LDPC code. Given the base graph of this code, we calculate the lower bound of $M \geq 7$ for the lifting degree to achieve $g \geq 8$ in the corresponding QC-SC code. However, with $M = 7$, the base graph proposed in [26] does not achieve $g \geq 8$.

On the other hand, we also set $m = 1$ (same as that of [26]) and we can find a spreading matrix B with the small number of 4-cycles and 6-cycles in the corresponding SC base graph.[11] For such a base graph, lower bound for lifting degree to obtain $g \geq 8$ is also $M \geq 7$, and we could construct code C_{10} corresponding to this spreading matrix to achieve $g = 8$ with $M = 7$. Table 7.3 contains the information for code C_{10} compared to its counterpart from [26].

As is evident from Table 7.3, according to the analysis in Section 7.4.2, one can see that both our optimized spreading matrix and that of [26] have $M \geq 7$ as their lower bound to achieve $g = 8$. However, in Appendix E we prove that this bound is not achievable for the base graph of [26]. In this respect, we present the following result.

Corollary 8. *In an SC base graph with $g = 4$, depending on the multiplicity of 4-cycles and the distribution of those 4-cycles sharing a specific edge, the lower bound*

[11] According to the small value of m, we cannot eliminate all 4-cycles at this step, thus, we just minimize them while the multiplicity of 6-cycles are also reduced. The number of unbroken 4-cycles associated with our designed B matrix is smaller (80 vs 84) and the number of unbroken 6-cycles is a little larger (276 vs 240) than those of the B matrix in [26].

Table 7.3: Constructed SC code C_{10} with $g = 8$ compared with an SC code in [26], with the same memory (m), lifting degree (M) and degree distribution.

	Code C_{10}	Code in [26]
$\gamma \times c$	3×7	3×7
m	1	1
B	1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1 1 0 0	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 1
M	7	7
Lower bound of M to achieve $g = 8$	7	7
P	6 0 2 3 0 0 3 0 0 0 0 3 0 6 6 0 5 2 2 5 0	0 1 3 5 2 4 1 0 1 2 3 4 5 6 0 5 0 2 4 6 2
Girth (g) of the designed QC-SC code	8	6

$M \geq \max_i\{N_4^{e_i}\} + 1$, *as proposed in Theorem 14, could be achievable.*

In fact, Corollary 8 gives more information if it is studied for a given SC base graph with a specific distribution of 4-cycles that share an edge.

According to Corollary 8 and as a general idea, for the construction of QC SC-LDPC codes with girth g, such that the corresponding SC protograph has girth $g' < g$, it is more beneficial to minimize the multiplicity of cycles of length g' (rather than the larger cycles) in the SC protograph. This is because the smallest cycles (of length g') in the base graph are responsible for the existence of larger cycles, as they form larger cycles when sharing one or a few edges. This is, in fact, the opposite of the approach in [26], at which the authors claimed that 4-cycles are easy to be all removed from the lifted graph by a careful choice of permutation shifts, and thus the optimization for B is done only in terms of 6-cycle minimization. Here, we showed that this statement could not be suitable in general. Particularly, for those cases with a small value of M, which is close to the lower bound for $g \geq 8$, we may be able to minimize (or even remove all) the 6-cycles in the lifted graph, if the number of 4-cycles is minimized in the base graph. It is worth noting that the approach of [63] for the design of QC

SC-LDPC codes with $g \geq 8$ is also based on minimizing the number of 4-cycles in the corresponding SC base graph.

7.6.2 QC-SC codes with a fixed lifting degree and varying memory

According to what we described in Section 7.5, we assume that the exponent matrix P with a fixed lifting degree M is given. Thus, our aim is to search for the smallest m to achieve a specific girth $g = 6$ or $g = 8$ in the corresponding QC SC-LDPC code. Starting from the lower bound of m, we search for at most 1 hour to find a spreading matrix B which results in a code with girth 6 or 8. If no code is found within this time, we increase m by one and keep searching with the new value of m. This process continues until we find a value for m satisfying the girth condition. We present the numerical results of this part for two sets of QC-SC codes. The first set of codes, given in Table 7.4, have random exponent matrices, while those presented in Table 7.5 enjoy optimized exponent matrices. The optimization process for these codes is performed to obtain a P matrix with the smallest number of 4-cycles in the corresponding QC-LDPC block code, after searching for a limited time.[12] In all the cases, however, the starting QC-LDPC block codes have $g = 4$. From Table 7.4 it is evident that our lower bounds for m, as well as our proposed values of m found through a computer search, are smaller compared to those of the counterpart codes in [7] with the same girth and degree distribution.[13] Also, as we expected, the constraint length of our codes are larger compared to those of [7]. This is due to the fact that the direct construction of SC-LDPC codes (i.e., with $M = 1$) always achieves a smaller constraint length to obtain a specific girth, compared to the situation that we start from an underlying QC-LDPC block code. Nevertheless, in Section 7.6.4 we show that such an increase in the constraint length usually comes with an advantage of a better error floor performance. We also note that code C_{22} given in Table 7.5 is the one described in Example 6 so that the lower bounds of memory for this code to achieve $g = 6$ and $g = 8$ are $m \geq 1$ and $m \geq 2$, respectively.

[12] In order to avoid the large complexity imposed on this optimization step, we limit the search time to at most half an hour.

[13] Note that the results of [7] are for the special case of $M = 1$.

Table 7.4: Properties of constructed SC codes with different values of M and the corresponding bounds for memory m to obtain a girth ($g = 6$ or 8). Exponent matrices of our constructed codes are randomly constructed with a given M.

	Code C_{11}	Code C_{12}	Code C_{13}	Code C_{14}	Code C_{15}	Code C_{16}	Code C_{17}	Code C_{18}	Code C_{19}
$\gamma \times c$	3×4	3×5	3×6	3×10	3×10	3×10	3×10	3×10	3×10
Lifting degree (M)	2	2	2	3	4	4	4	4	3
Lower bound of memory to achieve $g = 6$ (Lemma 13)	1	2	2	3	3	2	2	3	2
Lower bound of memory to achieve $g = 6$ (theory) in [7]	2	2	3	5	5	5	5	5	5
Lower bound of memory to achieve $g = 8$ (Lemma 14)	1	3	3	7	5	6	5	8	7
Lower bound of memory to achieve $g = 8$ (theory) in [7]	2	3	4	12	12	12	12	12	12
Constraint length (ν_s) in theory to achieve $g = 8$	16	40	48	240	240	280	240	360	240
Constraint length (ν_s) in theory to achieve $g = 8$ in [7]	12	20	30	130	130	130	130	130	130
Smallest memory to achieve $g = 6$ (in practice)	1	2	2	3	3	3	2	4	3
Smallest memory to achieve $g = 8$ (in practice)	2	3	4	10	7	8	8	11	11
Smallest memory to achieve $g = 8$ (in practice) in [7]	3	6	7	15	15	15	15	15	15
Constraint length (ν_s) (in practice) to achieve $g = 8$	24	40	60	330	320	360	360	480	360
Constraint length (ν_s) (in practice) to achieve $g = 8$ in [7]	16	35	48	160	160	160	160	160	160

Table 7.5: Properties of constructed SC codes with different values of M and the corresponding bounds for memory m to obtain a girth ($g = 6$ or 8). Exponent matrices of our constructed codes are optimized to achieve small number of 4-cycles corresponding to a given M.

	Code C_{20}	Code C_{21}	Code C_{22}	Code C_{23}	Code C_{24}
$\gamma \times c$	3×4	3×5	3×6	3×10	3×10
Lifting degree (M)	3	3	3	5	4
Lower bound of memory m to achieve $g = 6$ (Lemma 13)	1	1	1	1	1
Lower bound of memory m to achieve $g = 8$ (Lemma 14)	1	2	2	4	5
Constraint length (ν_s) in theory to achieve $g = 8$	24	45	54	250	240
Smallest memory m to achieve $g = 6$ (in practice)	1	1	1	1	1
Smallest memory m to achieve $g = 8$ (in practice)	2	2	3	6	7
Constraint length (ν_s) (in practice) to achieve $g = 8$	36	45	72	350	320

7.6.3 QC-SC codes with varying lifting degree and varying memory

In this part, we fix neither the spreading matrix B, nor the exponent matrix P. Thus, what we do is to simultaneously assign the entries of P matrix and B matrix to satisfy the girth condition to be $g = 6$ or $g = 8$ in the corresponding QC SC-LDPC code. Therefore, we choose different values of m and M and jointly construct the corresponding P matrix and B matrix satisfying a desired girth ($g = 6$ or 8). An effort is made to find the smallest values of the pair (m, M) satisfying the girth condition. The algorithm is implemented such that we start with an arbitrary pair of (m, M) that are generally larger than the values of (m, M) proposed by the 2-step construction techniques, presented in Tables 7.1-7.5 for a given degree distribution and girth 6. Then, for this chosen pair of (m, M), we continuously assign entries of B and P matrices in random (from the set of all eligible assignments for each column of B and P matrices) and check the girth of the corresponding QC SC-LDPC code using Lemma 2 and Lemma 1. Once a code with $g = 6$ is found, we reduce m by one, and keep searching with the new m and the previous M.[14] Again, if the search was successful, this time we reduce M by one unit, and keep m as is. This process goes on until 3 hours of the whole search over all selected pairs of (m, M) is completed. At the end, we report the latest achievable pair of (m, M) in Table 7.6 for the corresponding degree distribution and girth 6. The same process is also performed to obtain the smallest possible pairs of (m, M) satisfying the girth of 8 in the corresponding QC SC-LDPC codes. Comparing Table 7.6 with Tables 7.1-7.2 and Table 7.4 reveals the fact that when there is a degree of freedom on both contributing parameters m and M (as it is the case for the codes in Table 7.6), we are generally able to find the smaller values of (m, M) and smaller constraint length for a given girth and degree distribution. However, this comes at the expense of larger implementation complexity for obtaining the numerical results of Table 7.6.[15] Furthermore, comparing the values of (m, M) for the codes in Tables 7.1 and 7.5 with those in Table 7.6 shows that if we conduct an optimization at the first step of the construction (as it is the case for

[14] Here we prefer to first change the values of m and then we change M. It is also possible to start with changing M and then m.

[15] This is because we simultaneously assign entries to both the P matrix and B matrix and that is why we consider a larger run time for the joint construction method. Only, for the codes $C_2 - C_6$ for which we remove all the 4-cycles, at the first step, the run-time for the separate construction technique is larger than that of the joint design.

the codes in Table 7.1 and Table 7.5) we could approach the smallest (m, M) pairs (and also the constraint lengths) obtained through a joint construction in Table 7.6.

It is worth noting that as the upper bound proposed in Section 7.3 (Theorem 11) is based on considering the worst-case scenario, it proposes significantly larger pairs of (m, M) for QC SC-LDPC codes with $g \geq 6$. On the other hand, we observe that in practice, we are able to achieve smaller pairs of (m, M) for our constructed QC SC-LDPC codes using a joint design approach.

Table 7.6: Properties of constructed SC codes when neither of M nor m is fixed.

	Code C_{25}	Code C_{26}	Code C_{27}	Code C_{28}
$\gamma \times c$	3×4	3×5	3×6	3×10
Upper bound on $(m+1)M$ to achieve $g \geq 6$ from Theorem 11	41	57	73	138
Smallest $(m+1)M$ to achieve $g = 6$ in practice	4	4	6	12
Smallest (m, M) to achieve $g = 6$ (in practice)	(1,2)	(1,2)	(2,2)	(3,3)
Smallest (m, M) to achieve $g = 8$ (in practice)	(2,2)	(2,3)	(3,3)	(5,5)
Constraint length $(\nu_s = (m+1)cM)$ to achieve $g = 8$ in practice	24	45	72	300

7.6.4 Performance comparison

In this section, the BER performance for two of our QC SC-LDPC codes with $g = 8$ made through the 2-step construction technique of Section 7.5 and the joint construction technique are compared with a counterpart code from [7] with $g = 8$ and the same degree distribution. It should be noted that the codes in [7] are based on the direct design of SC-LDPC codes, and thus no lifting stage is introduced there. We consider our designed code C_{15} from Table 7.4 with $\gamma = 3, c = 10, m = 7, M = 4$ and $g = 8$, in addition to code C_{28} from Table 7.6 with $\gamma = 3, c = 10, m = 5, M = 5$ and $g = 8$. The counterpart code from [7] with $m = 15$ has the same degree distribution and girth as our codes. We choose $L = 100, L = 75$ and $L = 214$ for our designed codes C_{15}, C_{28} and for the code in [7], respectively, so that all codes have the same rate $R = 0.68$. Also, in order to have a fair comparison, we choose $W = 160c = 1600$ bits for all codes. LETS distribution for these codes is given in Table 7.7, where we present the normalized multiplicity (i.e., multiplicity divided by block length N) to find the contribution of each variable node to the generation of different classes of LETS. From this table, it is clear that our designed codes have

Table 7.7: Normalized multiplicity of non-empty LETS classes in the range of $a \leq 8, b \leq 3$ for our designed code C_{15} (with block length $N = 4000$), designed code C_{28} (with block length $N = 3750$) and their counterpart from [7] (with block length $N = 2140$). All codes have $R = 0.68$ and $g = 8$.

LETS class	Designed code C_{15}	Designed code C_{28}	Code in [7]
$(5, 3)$	1.5190	1.5253	5.3659
$(6, 2)$	0.4660	0.3787	2.7033
$(7, 3)$	12.7520	19.7773	71.7168
$(8, 0)$	0	0	0.1874
$(8, 2)$	2.8670	3.3840	25.9500

significantly smaller number of LETSs in all classes within the range, compared to their counterpart from [7].

As a result of LETS distribution given in Table 7.7, we observe a superior error floor performance for our designed codes as shown in Fig. 7.4.[16] It should be noticed that although we did not conduct any optimization on the distribution of trapping sets in our designed codes and the optimization was only performed with respect to the girth (i.e., we only impose the condition of having $g = 8$), the LETS distribution and the error floor performance of our designed codes are significantly improved compared to the code in [7].

Furthermore, our designed codes have larger minimum distance compared to the code of [7].[17] This is consistent with the result of [76] regarding the superiority of weakly coupled SC-LDPC codes, compared to the strongly coupled codes in terms of the minimum distance and error floor performance. Referring to the discussion presented in Section 7.2, our codes C_{15} and C_{28} have $d_1 = 3 \times M = 12, m = 7$ and $d_1 = 3 \times M = 15, m = 5$, respectively. While for the code of [7] we have $d_1 = 3$ and $m = 15$. Therefore, one can verify that the former (code C_{15} and C_{28}) belong to the class of weakly coupled, while the latter is a strongly coupled code. Considering the same decoding latency and complexity (as the window size in bits and the number of iterations is equal for all cases) a better error floor performance and larger minimum distance are evident for our designed weakly coupled codes compared to their strongly coupled counterpart.

[16] We use a floating-point sliding window decoder based on belief propagation algorithm. The maximum number of iterations is 50 and the simulations are performed over the binary-input AWGN channel.

[17] From Table 7.7 it is clear that the minimum distance for the code in [7] is 8. While code C_{28} has $d_{min} = 12$ and C_{15} has $d_{min} > 12$.

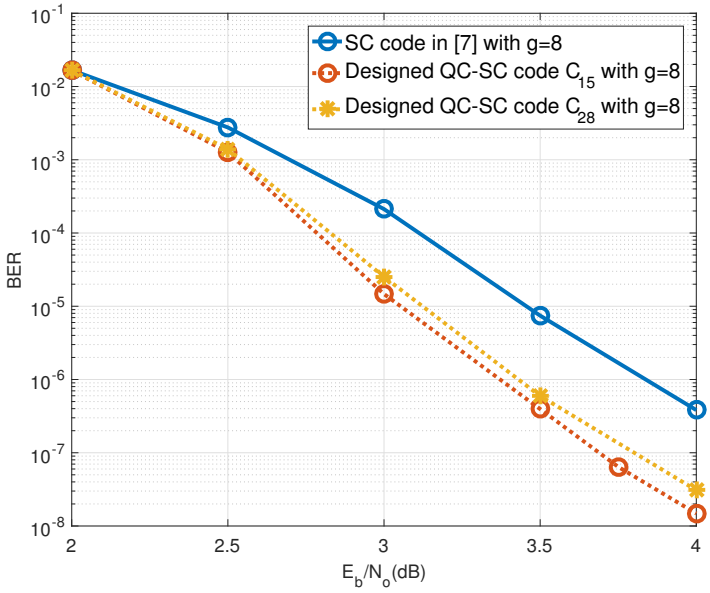


Figure 7.4: Performance comparison of designed QC-SC codes $\mathcal{C}_{15}$ and C_{28} from Table 7.4 and Table 7.6, respectively, with their counterpart from [7] having the same girth $g = 8$ and rate $R = 0.68$.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this thesis, we studied the error floor analysis of SC-LDPC codes, and proposed a number of design techniques for the construction of either QC-LDPC codes or SC-LDPC codes with low error floor, and small implementation complexity.

First, we conducted a mathematical analysis on the distribution of cycles and small TSs in protograph-based SC-LDPC codes. In this respect, we considered general edge spreading process and we calculated the probability distribution function for the existence of cycles of different lengths and small TSs during a random edge spreading process. These functions are in terms of memory m and they are, in fact, monotonically decreasing as m grows. Furthermore, we proved that the probability distribution function for the existence of a TS containing k fundamental cycles during a random edge spreading process, is monotonically decreasing with m^k. From these two results, we conclude that the larger memory and/or the larger number of fundamental cycles included in a TS results in a higher probability of breakage for such cycles and TSs during the edge spreading process. This is translated into a reduction on the number of harmful structures in the error floor region, and thus a better error floor performance for SC-LDPC codes. To complement our probabilistic study, we also performed a deterministic study on the distribution of cycles in a protograph-based SC-LDPC codes. For this, we considered a deterministic spreading matrix and we proposed a method to find the exact multiplicity of cycles of different lengths in terms of the coupling length L by only inspecting the spreading matrix. This method, compared to the search-based approach, is simpler for implementation. This study gives an insight into the error floor performance of finite-length protograph-based

SC-LDPC codes with respect to the memory m and the structure of the cycles or TSs.

In the second part of this thesis, we proposed various design approaches for the construction of high-performance QC-LDPC and SC-LDPC codes. The first type of design technique is based on imposing simple conditions on the short cycles of code's Tanner graph. This results in the elimination of most harmful TS structures. In this approach, we, in fact, only focus on those short cycles that are capable to create some harmful TS structures. This is against the common approach of reducing the multiplicity of (or eliminating) all cycles of a specific length. As a result, our proposed approach offers a better error floor performance (due to the elimination of harmful TS structures) and small implementation complexity. Generally speaking, one of the main advantages of this design approach is the simplicity of implementation. In other words, our proposed conditions are easily implemented and verified in a code, and this makes the search algorithm more efficient compared to the other search-based algorithms in the literature. As a result of the low implementation complexity, one could use our proposed technique for the design of high-rate codes within a reasonable period of time. Our designed codes using this method are presented in the context of QC-LDPC codes and time-invariant SC-LDPC codes. Our extensive simulation results demonstrated the superiority of our designed codes particularly in terms of the error floor performance and the decoding complexity and latency, compared to the state-of-the-art.

Moreover, we proposed a systematic efficient search algorithm for the construction of time-invariant SC-LDPC codes with low error floor and small constraint length. This approach is based on the successive minimization of the multiplicity of harmful TS structures in an SC code's Tanner graph. In this regard, we developed a tree-based search method by giving a higher priority to the TSs with higher contribution to the error floor performance (i.e, the most harmful TSs in the error floor region). As a result of this approach, we could construct time-invariant SC-LDPC codes with low error floor and the minimum constraint length to achieve a specific girth.

To complement all the above research directions, we performed an analysis of the design of QC SC-LDPC codes with small constraint length and a specific girth. It is known that the constraint length of a QC SC-LDPC code depends linearly on the memory and lifting degree of such codes. Thus, we look at the problem of QC-SC code design from three different perspectives. First, we fixed the memory and

we derived bounds on lifting degree to achieve a certain girth for the corresponding QC SC-LDPC code. Secondly, we performed the same analysis by fixing the lifting degree and calculating bounds on the memory to obtain a specific girth in the final QC SC-LDPC code. Finally, as the third scenario, we considered a joint optimization of memory and lifting degree and we proposed bounds on the multiplication of these two parameters to obtain a specific girth in the QC SC-LDPC code. Then, we compared achievable values of memory and lifting degree in these scenarios and discussed the pros and cons of each of them.

8.2 Future Work

Insights gained from this work can enable further research on the design of spatially coupled LDPC codes with different properties. As the first step, it would be an interesting topic of future research to apply the techniques proposed in Chapter 5 to design QC SC-LDPC codes based on general edge spreading process. Due to the simplicity of our proposed technique, the designed QC SC-LDPC codes would benefit from a low error floor as well as low implementation complexity.

Another possible direction for a future study is to extend our design approach for the construction of irregular SC-LDPC codes with an improved error floor performance and small constraint length. In addition to the low error floor gained by our proposed techniques, such codes would offer more advantages in terms of the waterfall performance by optimizing the degree distribution.

In addition to it, it is possible to look at the analysis of cycles and TS distribution for an SC protograph which is not of Type-1. In other words, one can assume parallel edges in the base graph of protograph-based SC-LDPC codes and perform almost similar analysis to find the average distribution of cycles and TSs in such codes. Further modification on our proposed design techniques to make it applicable to base graphs with parallel edges is another promising research direction. Moreover, it is feasible to propose design techniques for the construction of time-varying SC-LDPC codes, as these codes offer a more degree of freedom to further eliminate a set of harmful TS structures.

In addition to above recommendations, it is interesting to come up with a different construction technique, such as *symmetric construction* (by applying some modifications to the technique proposed in [82]) that could be utilized for the design

of SC-LDPC codes. Such design techniques can offer a remarkable gain in terms of the implementation complexity and makes the search process much simpler.

List of References

[1] B. Amiri, A. H. Reisizadehmobarakeh, H. Esfahanizadeh, J. Kliewer, and L. Dolecek, "Optimized design of finite-length separable circulantbased spatially-coupled codes: An absorbing set-based analysis," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4029–4043, Oct. 2016.

[2] F. Amirzade and M. Sadeghi, "Lower bounds on the lifting degree of QC-LDPC codes by difference matrices," *IEEE Access*, vol. 6, pp. 23688–23700, Apr. 2018.

[3] 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.

[4] M. Baldi and M. Bianchi and G. Cancellieri and F. Chiaraluce, "Progressive differences convolutional low-density parity-check codes," *IEEE Commun. Lett.*, vol. 16,no. 11, pp. 1848–1851, Nov. 2012.

[5] M. Baldi and M. Battaglioni and F. Chiaraluce and G. Cancellieri, "Time-invariant spatially coupled low-density parity-check codes with small constraint length," in *Proc. of IEEE Int. Black Sea Conf. on Commun. and Networking (BlackSeaCom)*, 2016, pp. 1–5.

[6] M. Battaglioni, A. Tasdighi, M. Baldi, M. H. Tadayon and F. Chiaraluce, "Compact QC-LDPC block and SC-LDPC convolutional codes for low-latency communications," in *Proc. of IEEE 29th Annual Int. Symp. Personal, Indoor and Mobile Radio Commun. (PIMRC)*, Bologna, 2018, pp. 1–5.

[7] M. Battaglioni, A. Tasdighi, G. Cancellieri, F. Chiaraluce and M. Baldi, "Design and analysis of time-invariant SC-LDPC convolutional codes with small constraint length," *IEEE Trans. Commun.*, vol. 66, no. 3, pp. 918–931, Mar. 2018.

[8] M. Battaglioni, F. Chiaraluce, M. Baldi, and D. Mitchell, "Efficient search and elimination of harmful objects in optimized QC SC-LDPC codes," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019.

[9] A. Beemer and C. A. Kelley, "Avoiding trapping sets in SC-LDPC codes under windowed decoding," in *Proc. of IEEE Int. Symp. Inf. Theory and its Appl. (ISITA)*, Oct. 2016, pp. 206–210.

[10] A. Beemer, S. Habib, C. A. Kelley, and J. Kliewer, "A generalized algebraic approach to optimizing SC-LDPC codes," in *Proc. Annu. Allerton Conf. Commun., Control Comput.*, Oct. 2017, pp. 672–679.

[11] A. Beemer, "Design and analysis of graph-based codes using algebraic lifts and decoding networks," PhD dissertation, University of Nebraska- Lincoln, 2018.

[12] I. E. Bocharova, F. Hug, R. Johannesson, B. D. Kudryashov and R. V. Satyukov, "Searching for voltage graph-based LDPC tailbiting codes with large girth," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2265–2279, Apr. 2012

[13] D. Chang, F. Yu, Z. Xiao, N. Stojanovic, F.N. Hauske, Y. Cai, C. Xie, L. Li, X. Xu, Q. Xiong "LDPC convolutional codes using layered decoding algorithm for high speed coherent optical transmission," in *Proc. Opt. Fiber Commun. Conf. (OFC)*, 2012. pp. 1–3.

[14] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.

[15] L. Chen, S. Mo, D. J. Costello, D. G. M. Mitchell, and R. Smarandache, "A protograph-based design of quasi-cyclic spatially coupled LDPC codes," in *Proc. of IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 1683–1687.

[16] J. Cho and L. Schmalen, "Construction of protographs for large-girth structured LDPC convolutional codes," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, Jun. 2015, pp. 4412–4417.

[17] D. J. Costello, L. Dolecek, T. E. Fuja, J. Kliewer, D. G. M. Mitchell and R. Smarandache, "Spatially coupled sparse codes on graphs: Theory and practice," *IEEE Commun. Mag.*, vol. 52, pp. 168–176, Jul. 2014.

[18] A. Dehghan and A. H. Banihashemi, "On the Tanner graph cycle distribution of random LDPC, random protograph-based LDPC, and random quasi-cyclic LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 64, no. 6, pp. 4438–4451, Jun. 2018.

[19] A. Dehghan and A. H. Banihashemi, "Hardness results on finding leafless elementary trapping sets and elementary absorbing sets of LDPC codes," *IEEE Trans. Inf. Theory*, vol. 65, no. 7, pp. 4307–4315, Jul. 2019.

[20] A. Dehghan and A. H. Banihashemi, "Asymptotic average multiplicity of structures within different categories of trapping sets, absorbing sets, and stopping sets in random regular and irregular LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 65, no. 10, pp. 6022–6043, Oct. 2019.

[21] 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, Jun. 2002.

[22] 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 (ISIT)*, Jun. 2015, pp. 1079–1083.

[23] 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.

[24] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "A novel combinatorial framework to construct spatially-coupled codes: Minimum overlap partitioning," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2017, pp. 1693–1697.

[25] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "Spatially coupled codes optimized for magnetic recording applications," *IEEE Trans. Magn.*, vol. 53, no. 2, pp. 1–11, Feb. 2017.

[26] H. Esfahanizadeh, A. Hareedy, and L. Dolecek, "Finite-length construction of high performance spatially-coupled codes via optimized partitioning and lifting," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 3–16, Jan. 2019.

[27] A. Farsiabi and A. H. Banihashemi, "Error floor estimation of LDPC decoders — A code independent approach to measuring the harmfulness of trapping Sets," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 2667–2679, May 2020.

[28] 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.

[29] R. G. Gallager, "Low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[30] A. Hareedy, R. Wu, and L. Dolecek, "A channel-aware combinatorial approach to design high performance spatially-coupled codes," *IEEE Trans. Inf. Theory*, vol. 66, no. 8, pp. 4834–4852, Aug. 2020.

[31] 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, Mar. 2015.

[32] 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.

[33] Y. Hashemi and A. H. Banihashemi, "Tight lower and upper bounds on the minimum distance of LDPC codes," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 33–36, Jan. 2018.

[34] 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.

[35] H. Hatami, D. G. M. 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 (ISIT)*, Vail, CO, 2018, pp. 826–830.

[36] H. Hatami, D. G. M. Mitchell, D. J. Costello and T. E. Fuja, "Performance bounds and estimates for quantized LDPC decoders," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 683-696, Feb. 2020.

[37] X.-Y. 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.

[38] M. Ivkovic, S. K. Chilappagari, and B. Vasic, "Eliminating trapping sets in low-density parity-check codes by using Tanner graph covers," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3763–3768, Aug. 2008.

[39] A. R. Iyengar, M. Papaleo, P. H. Siegel, J. K. Wolf, A. Vanelli-Coralli and G. E. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2303–2320, Apr. 2012.

[40] A. R. Iyengar, P. H. Siegel, R. L. Urbanke and J. K. Wolf, "Windowed decoding of spatially coupled codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2277–2292, Apr. 2013.

[41] A. Jimenez Felstrom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2181–2191, Sep. 1999.

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

[43] 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.

[44] M. Karimi and A. H. Banihashemi, "On the girth of quasi-cyclic protograph LDPC codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4542–4552, Jul. 2013.

[45] 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.

[46] 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, pp. 1288–1291, Aug. 2012.

[47] S. Kim, J. S. No, H. Chung and D. J. Shin, "Quasi-cyclic low-density parity-check codes with girth larger than 12," *IEEE Trans. Inf. Theory*, vol. 53, no. 8, pp. 2885–2891, Aug. 2007.

[48] D. E. Knuth, "Johann Faulhaber and sums of powers," *Mathematics of Computation*, vol. 61, pp. 277–294, Jul. 1993.

[49] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[50] S. Kudekar, T. Richardson and R. Urbanke, "Spatially-coupled ensembles universally achieve capacity under belief propagation," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 7761–7813, Dec. 2013.

[51] 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.

[52] S. Landner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. Int. Conf. on Wireless Networks, Commun. and Mobile Computing*, Maui, HI, 2005, pp. 630-635.

[53] M. Lentmaier, G. P. Fettweis, K. S. Zigangirov and D. J. Costello, "Approaching capacity with asymptotically regular LDPC codes," in *Proc. Inform. Theory Appl. Workshop,*, Feb. 2009, pp. 173–177.

[54] M. Lentmaier, A. Sridharan, D. J. Costello, and K. S. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5274–5289, Oct. 2010.

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

[56] D. Mackay, R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996.

[57] Y. Mao and A. H. Banihashemi, "A heuristic search for good low-density parity-check codes at short block lengths," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Helsinki, Finland, Jun. 2001, pp. 41–44.

[58] D. G. M. Mitchell, A. E. Pusane, N. Goertz and D. J. Costello, "Free distance bounds for protograph-based regular LDPC convolutional codes," in *Proc. International Symp. Turbo Codes and Related Topics*, 2008, pp. 408–413.

[59] D. G. M. Mitchell, A. E. Pusane, and D. J. Costello, Jr., "Minimum distance and trapping set analysis of protograph-based LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 254–281, Jan. 2013.

[60] D. G. M. Mitchell, L. Dolecek and D. J. Costello, "Absorbing set characterization of array-based spatially coupled LDPC codes," *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Honolulu, HI, 2014, pp. 886–890.

[61] D. G. M. Mitchell, M. Lentmaier and D. J. Costello, "Spatially coupled LDPC codes constructed from protographs," *IEEE Trans. Inf. Theory*, vol. 61, no. 9, pp. 4866–4889, Sep. 2015.

[62] D. G. M. Mitchell and E. Rosnes, "Edge spreading design of high rate array-based SC-LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2940–2944.

[63] S. Mo, L. Chen, D. J. Costello, D. G. M. Mitchell, R. Smarandache and J. Qiu, "Designing protograph-based quasi-cyclic spatially coupled LDPC codes with large girth," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5326–5337, Sep. 2020.

[64] 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.

[65] S. Naseri and A. H. Banihashemi, "Spatially coupled LDPC codes with small constraint length and low error floor," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 254–258, Feb. 2020.

[66] S. Naseri and A. H. Banihashemi, "Construction of time invariant spatially coupled LDPC codes free of small trapping sets," submitted to *IEEE Trans. Commun.*, Jun. 2020.

[67] S. Naseri, A. Dehghan and A. H. Banihashemi, "Theoretical enumeration and average number of cycles and trapping sets in finite-length spatially coupled LDPC codes," to be submitted.

[68] S. Naseri, A. Dehghan and A. H. Banihashemi, "On the girth of quasi-cyclic spatially coupled LDPC codes," to be submitted.

[69] D. V. Nguyen, S. K. Chilappagari, M. W. 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.

[70] Y. Polyanskiy, H. V. Poor and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

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

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

[73] T. Richardson and S. Kudekar, "Design of low-density parity check codes for 5G new radio," *IEEE Commun. Mag.*, vol. 56, pp. 28–34, Mar. 2018.

[74] M. Sadeghi and F. Amirzade, "Edge-coloring technique to analyze elementary trapping sets of spatially-coupled LDPC convolutional codes," *IEEE Commun. Lett.*, vol. 24, no. 4, pp. 711–715, Apr. 2020.

[75] A. Sariduman, A. E. Pusane, and Z. C. Taskn, "On the construction of regular QC-LDPC codes with low error floor," *IEEE Commun. Lett.*, vol. 24, no. 1, pp. 25–28, Jan. 2020.

[76] L. Schmalen, V. Aref, J. Cho, D. Suikat, D. Rsener, and A. Leven, "Spatially coupled soft-decision error correction for future lightwave systems," *Journal of Lightwave Technology*, vol. 33, no. 5, pp. 1109–1116, Mar. 2015.

[77] L. Schmalen, D. Suikat, D. Rösener, V. Aref, A. Leven, S. ten Brink "Spatially coupled codes and optical fiber communications: An ideal match?" in *Proc. IEEE Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2015, pp. 460–464.

[78] M. H. Tadayon, A. Tasdighi, M. Battaglioni, M. Baldi, F. Chiaraluce, "Efficient search of compact QC-LDPC and SC-LDPC convolutional codes with large girth," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1156–1159, Jun. 2018.

[79] R. M. Tanner, "A recursive approach to low-complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[80] 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.

[81] 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.

[82] A. Tasdighi, A. H. Banihashemi and M. R. Sadeghi, "Symmetrical constructions for regular girth-8 QC-LDPC codes," *IEEE Trans. Commun.*, vol. 65, no. 1, pp. 14–22, Jan. 2017.

[83] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protograph," Jet Propulsion Lab., IPN Progress Rep., vol. 42, no. 154, pp. 42–154, Aug. 2003.

[84] T. Tian, C. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, vol. 52, pp. 1242–1247, Aug. 2004.

[85] 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.

[86] 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.

[87] D. Truhachev, K. S. Zigangirov and D. J. Costello, "Distance bounds for periodically time-varying and tail-biting LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4301–4308, Sep. 2010.

[88] B. Vasic, S. K. Chilappagari, D. V. Nguyen, and S. K. Planjery, "Trapping set ontology," in *Proc. 47th Annu. Allerton Conf. Commun., Control Comput.*, Monticello, IL, USA, Sep. 2009, pp. 1–7.

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

[90] L. Wei, D. G. M. Mitchell, T. E. Fuja and D. J. Costello, "Design of spatially coupled LDPC codes over GF (q) for windowed decoding," *IEEE Trans. Inf. Theory*, vol. 62, no. 9, pp. 4781–4800, Sep. 2016.

[91] D. B. West, *Introduction to graph theory*, Prentice Hall. Inc., Upper Saddle River, NJ, 1996.

[92] H. Xiao and A. H. Banihashemi, "Improved progressive-edge-growth (PEG) construction of irregular LDPC codes," *IEEE Commun. Lett.*, vol. 8, no. 12, pp. 715–717, Dec. 2004.

[93] 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.

[94] H. Xu, H. Li, B. Bai, M. Zhu, and B. Zhang, "Tanner (J,L) quasi-cyclic LDPC codes: Girth analysis and derived codes," *IEEE Access*, vol. 7, no. 4, pp. 944–957, Jan. 2019.

[95] G. Zhang, Y. Hu, Y. Fang and J. Wang, "Constructions of Type-II QC-LDPC codes with girth eight from sidon sequence," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 3865–3878, Jun. 2019.

[96] Z. Zhang, L. Dolecek, B. Nikolic, V. Anantharam and M. J. Wainwright, "Design of LDPC decoders for improved low error rate performance: quantization and algorithm choices," *IEEE Trans. Commun.*, vol. 57, no. 11, pp. 3258–3268, Nov. 2009.

[97] 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.

[98] H. Zhou and N. Goertz, "Cycle analysis of time-invariant LDPC convolutional codes," in *Proc. Int. Conf. Telecommun.*, Apr. 2010, pp. 23–28.

[99] *IEEE* standard for information technology local and metropolitan area networks specific requirements part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 5: enhancements for higher throughput, *IEEE* Std. 802.11n-2009, Oct. 29, 2009.

[100] *IEEE* standard for local and metropolitan area networks Part 16: air interface for fixed and mobile broadband wireless access systems amendment 2: physical and medium access control layers for combined fixed and mobile operation in licensed bands and corrigendum 1, *IEEE* Std. 802.16e-2005 and 802.16-2004/Cor 1-2005, Feb. 28, 2006.

[101] *IEEE* P802.22 Draft Standard for Wireless Regional Area Networks Part 22: Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: Policies and Procedures for Operation in the TV Bands, *IEEE P802.22 Draft Standard*, D0.3, May 2007.

[102] *IEEE* Standard for Information Technology-Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks-Specific Requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification, *IEEE Std. 802.3an*, Sep. 2006.

[103] *ETSI* Standard TS 102 005: Digital Video Broadcasting (DVB): frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2), ETSI Std. TS 102 005, Mar. 2010.

[104] *ETSI* Standard TR 102 376 V1.1.1: Digital Video Broadcasting (DVB) User guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVBS2), ETSI Std. TR 102 376, Feb. 2005.

[105] *ETSI* TS 138 212 v15.8.0: 5G NR multiplexing and channel coding (3GPP TS 38.212 version 15.8.0 release 15), ETSI TS 138 212, pp. 1–105, Jan. 2020.

Appendix A

A.1 Faulhaber's Formula

We present the *Faulhaber's formula* [48], which expresses the sum of p-th powers of the first n positive integers, and we use it in the calculations of Chapter 3. For some values of p we have the following equation:

$$
\begin{aligned}
F_1(n) &= \sum_{i=1}^{n} i^1 = \frac{1}{2}n + \frac{1}{2}n^2 \\
F_2(n) &= \sum_{i=1}^{n} i^2 = \frac{1}{6}n + \frac{1}{2}n^2 + \frac{1}{3}n^3 \\
F_3(n) &= \sum_{i=1}^{n} i^3 = \frac{1}{4}n^2 + \frac{1}{2}n^3 + \frac{1}{4}n^4 \\
F_4(n) &= \sum_{i=1}^{n} i^4 = -\frac{1}{30}n + \frac{1}{3}n^3 + \frac{1}{2}n^4 + \frac{1}{5}n^5 \\
F_5(n) &= \sum_{i=1}^{n} i^5 = -\frac{1}{12}n^2 + \frac{5}{12}n^4 + \frac{1}{2}n^5 + \frac{1}{6}n^6 \\
F_6(n) &= \sum_{i=1}^{n} i^6 = \frac{1}{42}n - \frac{1}{6}n^3 + \frac{1}{2}n^5 + \frac{1}{2}n^6 + \frac{1}{7}n^7 \\
F_7(n) &= \sum_{i=1}^{n} i^7 = \frac{1}{12}n^2 - \frac{7}{24}n^4 + \frac{7}{12}n^6 + \frac{1}{2}n^7 + \frac{1}{8}n^8 \\
F_8(n) &= \sum_{i=1}^{n} i^8 = \frac{-1}{30}n + \frac{2}{9}n^3 - \frac{7}{15}n^5 + \frac{2}{3}n^7 + \frac{1}{2}n^8 + \frac{1}{9}n^9 \\
F_9(n) &= \sum_{i=1}^{n} i^9 = \frac{-1}{12}n^2 + \frac{1}{2}n^4 - \frac{7}{10}n^6 + \frac{3}{4}n^8 + \frac{1}{2}n^9 + \frac{1}{10}n^{10}.
\end{aligned}
\tag{A.1}
$$

A.2 Proof of Lemma 6

Considering two random variables $(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2')$ which are assumed to be independent of each other, we calculate $\Pr(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2') = k)$. By (3.4), we have $k \in [-2m, 2m]$. For the value of k four cases can be considered.
Case 1. If $-2m \le k < -m$. Then we have

$$\Pr(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2') = k) = \sum_{x=-m}^{k+m} \Pr(\Delta B_{i_1}(j_1, j_2) = x) \Pr(\Delta B_{i_2}(j_1', j_2') = k - x). \tag{A.2}$$

In (A.2) the upper limit and lower limit on x are obtained by noting that we should have $k - x, x \in [-m, m]$, thus $-m \le x - k \le m$, so $-m + k \le x \le m + k$. Since $-2m \le k < -m$, the limit $-m \le x \le m + k$ satisfy all above conditions. By using (3.4) in (A.2), we have

$$\begin{aligned}
&\Pr(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2') = k) \\
&= \sum_{x=-m}^{k+m} \frac{(m+1) - |x|}{(m+1)^2} \times \frac{(m+1) - |k-x|}{(m+1)^2} \\
&= \sum_{x=-m}^{k+m} \frac{(m+1) + x}{(m+1)^2} \times \frac{(m+1) + (k-x)}{(m+1)^2} \\
&= \sum_{x=-m}^{k+m} \frac{(m+1)^2 + (m+1)k + x(k-x)}{(m+1)^4} \\
&= \frac{(m+1)^2(k+2m+1)}{(m+1)^4} + \frac{(m+1)k(k+2m+1)}{(m+1)^4} + \sum_{x=-m}^{k+m} \frac{xk - x^2}{(m+1)^4}
\end{aligned}$$

$$= \frac{\Big((m+1)^2(k+2m+1) + (m+1)k(k+2m+1) + \sum_{x=1}^{m}(-xk - x^2) + \sum_{x=1}^{m+k}(xk - x^2)\Big)}{(m+1)^4}. \tag{A.3}$$

Next, we use Faulhaber's formula (A.1), to simplify (A.3), then we have

$$\begin{aligned}
&\Pr(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2') = k) = \\
&\frac{1}{(m+1)^4}\Big(k^3/6 + k^2 m + k^2 + 2km^2 + 4km + 11\frac{k}{6} + 4\frac{m^3}{3} + 4m^2 + 11\frac{m}{3} + 1\Big).
\end{aligned} \tag{A.4}$$

This completes the proof of Case 1.

Case 2. If $-m \leq k < 0$. We have

$$\Pr(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2') = k) = \sum_{x=-m}^{k+m} \Pr(\Delta B_{i_1}(j_1, j_2) = x) \Pr(\Delta B_{i_2}(j_1', j_2') = k - x). \tag{A.5}$$

In (A.5) the upper bound and lower bound on x are obtained by noting that we should have $k - x, x \in [-m, m]$, thus $-m + k \leq x \leq m + k$. We also have $-m \leq k < 0$, thus the limit $-m \leq x \leq m + k$ has all conditions. By using (3.4) in (A.5), we have

$$\begin{aligned}
&\Pr(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2') = k) \\
&= \sum_{x=-m}^{k+m} \frac{(m+1) - |x|}{(m+1)^2} \times \frac{(m+1) - |k-x|}{(m+1)^2} \\
&= \sum_{x=-m}^{k-1} \frac{\Big((m+1) + x\Big)\Big((m+1) - (k-x)\Big)}{(m+1)^4} \qquad \text{(A.6)} \\
&+ \sum_{x=k}^{-1} \frac{\Big((m+1) + x\Big)\Big((m+1) + (k-x)\Big)}{(m+1)^4} \\
&+ \sum_{x=0}^{k+m} \frac{\Big((m+1) - x\Big)\Big((m+1) + (k-x)\Big)}{(m+1)^4} \qquad \text{(A.7)}
\end{aligned}$$

Next, we use Faulhaber's formula (A.1), to simplify (A.7), then we have

$$\begin{aligned}
&\Pr(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2') = k) = \\
&\frac{1}{6(m+1)^4}\Big(- 3k^3 - 6k^2 m - 6k^2 + 3k + 4m^3 + 12m^2 + 14m + 6\Big).
\end{aligned}$$

Case 3. If $0 \leq k < m$. We have

$$\Pr(\Delta B_{i_1}(j_1, j_2) + \Delta B_{i_2}(j_1', j_2') = k) = \sum_{x=-m+k}^{m} \Pr(\Delta B_{i_1}(j_1, j_2) = x) \Pr(\Delta B_{i_2}(j_1', j_2') = k - x). \tag{A.8}$$

In (A.8) the upper bound and lower bound on x are obtained by noting that we should have $k - x, x \in [-m, m]$, thus $-m + k \leq x \leq m + k$. We also have $0 \leq k < m$,

thus the limit $-m+k \leq x \leq m$ has all conditions. By using (3.4) in (A.8), we have

$$\begin{aligned}
&\Pr(\Delta B_{i_1}(j_1,j_2)+\Delta B_{i_2}(j_1',j_2')=k)\\
&=\sum_{x=-m+k}^{m}\frac{(m+1)-|x|}{(m+1)^2}\times\frac{(m+1)-|k-x|}{(m+1)^2}\\
&=\sum_{x=-m+k}^{-1}\frac{\Big((m+1)+x\Big)\Big((m+1)-(k-x)\Big)}{(m+1)^4}\\
&+\sum_{x=0}^{k-1}\frac{\Big((m+1)-x\Big)\Big((m+1)-(k-x)\Big)}{(m+1)^4}\\
&+\sum_{x=k}^{m}\frac{\Big((m+1)-x\Big)\Big((m+1)+(k-x)\Big)}{(m+1)^4}
\end{aligned}\tag{A.9}$$

Next, we use Faulhaber's formula to simplify (A.9), then we have

$$\begin{aligned}
&\Pr(\Delta B_{i_1}(j_1,j_2)+\Delta B_{i_2}(j_1',j_2')=k)=\\
&\frac{1}{6(m+1)^4}\Big(3k^3-6k^2m-6k^2-3k+4m^3+12m^2+14m+6\Big).
\end{aligned}$$

Case 4. If $m \leq k \leq 2m$. We have

$$\begin{aligned}
&\Pr(\Delta B_{i_1}(j_1,j_2)+\Delta B_{i_2}(j_1',j_2')=k)\\
&=\sum_{x=-m+k}^{m}\frac{(m+1)-|x|}{(m+1)^2}\times\frac{(m+1)-|k-x|}{(m+1)^2}\\
&=\sum_{x=-m+k}^{m}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)-(k-x)|}{(m+1)^2}
\end{aligned}\tag{A.10}$$

Finally, we use Faulhaber's formula to simplify (A.10), then we have

$$\begin{aligned}
&\Pr(\Delta B_{i_1}(j_1,j_2)+\Delta B_{i_2}(j_1',j_2')=k)=\\
&\frac{1}{(m+1)^4}\Big(-k^3/6+k^2m+k^2-2km^2-4km-11\frac{k}{6}+4\frac{m^3}{3}+4m^2+11\frac{m}{3}+1\Big).
\end{aligned}$$

Appendix B

B.1 Proof of Theorem 3

In the following, we describe how to find the probability that each of the structures in Fig. 3.2 remains unbroken during the edge spreading process using a random spreading matrix. The assumption is that the non-negative entries of B matrix are independent and identically distributed uniform random variables from $[0, m]$.

- Fig. 3.2(a)

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\
&= \Pr\left(B_{w_0,u_0} - B_{w_1,u_0} + B_{w_1,u_2} - B_{w_2,u_2} + B_{w_2,u_3} - B_{w_1,u_3} + B_{w_1,u_1} - B_{w_0,u_1} = 0\right) \\
&= \Pr\left(\Delta B_{w_0}(u_0,u_1) + \Delta B_{w_1}(u_2,u_0) + \Delta B_{w_2}(u_3,u_2) + \Delta B_{w_1}(u_1,u_3) = 0\right) \\
&= \sum_{y=-2m}^{2m} \Pr(\Delta B_{w_0}(u_0,u_1) + \Delta B_{w_1}(u_2,u_0) = y) \times \\
&\Pr(\Delta B_{w_2}(u_3,u_2) + \Delta B_{w_1}(u_1,u_3) = -y | \Delta B_{w_0}(u_0,u_1) + \Delta B_{w_1}(u_2,u_0) = y)
\end{aligned} \tag{B.1}$$

For the sake of simplicity, we define two random variables $[\Delta B_{w_0}(u_0,u_1) + \Delta B_{w_1}(u_2,u_0)] = X$ and $[\Delta B_{w_2}(u_3,u_2) + \Delta B_{w_1}(u_1,u_3)] = Z$. It is easy to verify that these two random variables are independent, because the contributing differential parameters in X and Z share no common edges. Thus,

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) = \\
&\sum_{y=-2m}^{2m} \Pr(\Delta B_{w_0}(u_0,u_1) + \Delta B_{w_1}(u_2,u_0) = y) \Pr(\Delta B_{w_2}(u_3,u_2) + \Delta B_{w_1}(u_1,u_3) = -y)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{y=-2m}^{2m} \Pr(X=y)\Pr(Z=-y) \\
&= \sum_{y=-2m}^{-m-1} \Pr(X=y)\Pr(Z=-y) + \sum_{y=-m}^{-1} \Pr(X=y)\Pr(Z=-y) \\
&+ \sum_{y=0}^{m-1} \Pr(X=y)\Pr(Z=-y) + \sum_{y=m}^{2m} \Pr(X=y)\Pr(Z=-y) \qquad \text{(B.2)}
\end{aligned}
$$

Next, we calculate (B.2)

$$
\begin{aligned}
&\sum_{y=-2m}^{2m} \Pr(X=y)\Pr(Z=-y) = \\
&= \sum_{y=-2m}^{-m-1} \big[\frac{1}{(m+1)^4}\big(y^3/6 + y^2 m + y^2 + 2ym^2 + 4ym + 11\frac{y}{6} + 4\frac{m^3}{3} + 4m^2 + 11\frac{m}{3} + 1\big)\big]^2 \\
&= \frac{1}{(m+1)^8}\big[\sum_{y=-2m}^{-m-1} \frac{y^6}{36} + \sum_{y=-2m}^{-m-1} \frac{y^5}{3}(m+1) + \sum_{y=-2m}^{-m-1} \frac{y^4}{3}\big((2m^2+4m+\frac{11}{6}) + 3(m+1)^2\big) \\
&+ \sum_{y=-2m}^{-m-1} (y^3)\big(\frac{\frac{4m^3}{3} + 4m^2 + 11\frac{m}{3} + 1}{3} + 2(m+1)(2m^2+4m+\frac{11}{6})\big) \\
&+ \sum_{y=-2m}^{-m-1} (y^2)\big(2(m+1)(\frac{4m^3}{3} + 4m^2 + 11\frac{m}{3} + 1) + (2m^2+4m+\frac{11}{6})^2\big) \\
&+ \sum_{y=-2m}^{-m-1} y\big(2(2m^2+4m+\frac{11}{6})(\frac{4m^3}{3} + 4m^2 + 11\frac{m}{3} + 1)\big) \\
&+ \sum_{y=-2m}^{-m-1} (\frac{4m^3}{3} + 4m^2 + 11\frac{m}{3} + 1)^2\big] \\
&= \frac{1}{(m+1)^8}\big[\sum_{y=m+1}^{2m} \frac{y^6}{36} - \sum_{y=m+1}^{2m} \frac{y^5}{3}(m+1) + \sum_{y=m+1}^{2m} \frac{y^4}{3}\big((2m^2+4m+\frac{11}{6}) + 3(m+1)^2\big) \\
&- \sum_{y=m+1}^{2m} (y^3)\big(\frac{\frac{4m^3}{3} + 4m^2 + 11\frac{m}{3} + 1}{3} + 2(m+1)(2m^2+4m+\frac{11}{6})\big) \\
&+ \sum_{y=m+1}^{2m} (y^2)\big(2(m+1)(\frac{4m^3}{3} + 4m^2 + 11\frac{m}{3} + 1) + (2m^2+4m+\frac{11}{6})^2\big) \\
&- \sum_{y=m+1}^{2m} y\big(2(2m^2+4m+\frac{11}{6})(\frac{4m^3}{3} + 4m^2 + 11\frac{m}{3} + 1)\big) \\
&+ \sum_{y=m+1}^{2m} (\frac{4m^3}{3} + 4m^2 + 11\frac{m}{3} + 1)^2\big] \\
&= \frac{m}{2520(m+1)^8}\big(10m^6 + 105m^5 + 427m^4 + 840m^3 + 805m^2 + 315m + 18\big). \qquad \text{(B.3)}
\end{aligned}
$$

Also, we have

$$\begin{aligned}
&\sum_{y=-m}^{-1} \Pr(X=y)\Pr(Z=-y) \\
&= \sum_{y=-m}^{-1} \big[\frac{1}{6(m+1)^4}\big(-3y^3 - 6y^2m - 6y^2 + 3y + 4m^3 + 12m^2 + 14m + 6\big)\big]^2 \\
&= \frac{1}{36(m+1)^8}\big[\sum_{y=1}^{m}(3y^3)^2 - \sum_{y=1}^{m} 36y^5(m+1) + \sum_{y=1}^{m} y^4(-18 + 36(m+1)^2) \\
&- \sum_{y=1}^{m} y^3\big(-6(4m^3 + 12m^2 + 14m + 6) - 36(m+1)\big) \\
&+ \sum_{y=1}^{m} y^2\big(-12(m+1)(4m^3 + 12m^2 + 14m + 6) + 9\big) \\
&- \sum_{y=1}^{m} y\big(6(4m^3 + 12m^2 + 14m + 6)\big) + \sum_{y=1}^{m}(4m^3 + 12m^2 + 14m + 6)^2\big] \\
&= \frac{m}{70(36(m+1)^8)}\big(594m^6 + 3563m^5 + 9177m^4 + 12740m^3 + 9891m^2 + 3857m + 498\big).
\end{aligned} \tag{B.4}$$

Furthermore,

$$\begin{aligned}
&\sum_{y=0}^{m-1} \Pr(X=y)\Pr(Z=-y) \\
&= \sum_{y=0}^{m-1} \big[\frac{1}{6(m+1)^4}\big(3y^3 - 6y^2m - 6y^2 - 3y + 4m^3 + 12m^2 + 14m + 6\big)\big]^2 \\
&= \frac{m}{70(36(m+1)^8)}\big(594m^6 + 4613m^5 + 15057m^4 + 26600m^3 + 26691m^2 \\
&+ 14147m + 3018\big).
\end{aligned} \tag{B.5}$$

Finally, we have

$$\begin{aligned}
&\sum_{y=m}^{2m} \Pr(X=y)\Pr(Z=-y) = \sum_{y=m}^{2m} \big[\frac{1}{(m+1)^4}\big(-y^3/6 + y^2m + y^2 - 2ym^2 - 4ym \\
&- 11\frac{y}{6} + 4\frac{m^3}{3} + 4m^2 + 11\frac{m}{3} + 1\big)\big]^2
\end{aligned}$$

$$= \frac{1}{(m+1)^8}\Big[\frac{m^7}{252} + \frac{5m^6}{72} + \frac{181m^5}{360} + \frac{35m^4}{18} + \frac{311m^3}{72} + \frac{395m^2}{72} + \frac{1543m}{420} + 1\Big]. \tag{B.6}$$

By substituting the equations (B.3), (B.4), (B.5) and (B.6) into (B.2) the calculations will be completed.

The Probability calculations for other TBC walks in Fig. 3.2(b)-(e) are the same as above. Thus, in the followings we only present the beginning of calculation process for these structures and skip the rest as they all conclude the same result as that of Fig. 3.2(a)

- Fig. 3.2(b)
  This TBC walk only exists in base graphs with $d_v > 3$. So for such base graphs, the probability that this structure remains a TBC walk in the corresponding SC base graph is

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\
&= \Pr\big(B_{w_0,u_1} - B_{w_2,u_1} + B_{w_2,u_2} - B_{w_3,u_2} + B_{w_3,u_1} - B_{w_1,u_1} + B_{w_1,u_0} - B_{w_0,u_0} = 0\big) \\
&= \Pr\big(\Delta B_{w_0}(u_1,u_0) + \Delta B_{w_2}(u_2,u_1) + \Delta B_{w_3}(u_1,u_2) + \Delta B_{w_1}(u_0,u_1) = 0\big) \\
&= \sum_{y=-2m}^{2m} \Pr\big(\Delta B_{w_0}(u_1,u_0) + \Delta B_{w_2}(u_2,u_1) = y\big) \\
&\times \Pr(\Delta B_{w_3}(u_1,u_2) + \Delta B_{w_1}(u_0,u_1) = -y\big).
\end{aligned} \tag{B.7}$$

- Fig. 3.2(c)

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\
&= \Pr\big(B_{w_0,u_1} - B_{w_1,u_1} + B_{w_1,u_2} - B_{w_2,u_2} + B_{w_2,u_3} - B_{w_3,u_3} + B_{w_3,u_0} - B_{w_0,u_0} = 0\big) \\
&= \Pr\big(\Delta B_{w_0}(u_1,u_0) + \Delta B_{w_1}(u_2,u_1) + \Delta B_{w_2}(u_3,u_2) + \Delta B_{w_3}(u_0,u_3) = 0\big) \\
&= \sum_{y=-2m}^{2m} \Pr\big(\Delta B_{w_0}(u_1,u_0) + \Delta B_{w_1}(u_2,u_1) = y)\big) \\
&\times \Pr\big(\Delta B_{w_2}(u_3,u_2) + \Delta B_{w_3}(u_0,u_3) = -y\big).
\end{aligned} \tag{B.8}$$

- Fig. 3.2(d)

This structure only exists in base graphs with $d_v > 3$:

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\
&= \Pr\left(B_{w_0,u_1} - B_{w_1,u_1} + B_{w_1,u_0} - B_{w_0,u_0} + B_{w_2,u_1} - B_{w_3,u_1} + B_{w_3,u_0} - B_{w_2,u_0} = 0\right) \\
&= \Pr\left(\Delta B_{w_0}(u_1,u_0) + \Delta B_{w_1}(u_0,u_1) + \Delta B_{w_2}(u_1,u_0) + \Delta B_{w_3}(u_0,u_1) = 0\right) \\
&= \sum_{y=-2m}^{2m} \Pr\left(\Delta B_{w_0}(u_1,u_0) + \Delta B_{w_1}(u_0,u_1) = y\right) \\
&\times \Pr\left(\Delta B_{w_2}(u_1,u_0) + \Delta B_{w_3}(u_0,u_1) = -y\right). \qquad \text{(B.9)}
\end{aligned}$$

- Fig. 3.2(e)

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\
&= \Pr\left(B_{w_0,u_0} - B_{w_1,u_0} + B_{w_1,u_1} - B_{w_0,u_1} + B_{w_0,u_2} - B_{w_1,u_2} + B_{w_1,u_3} - B_{w_0,u_3} = 0\right) \\
&= \Pr\left(\Delta B_{w_0}(u_0,u_1) + \Delta B_{w_1}(u_1,u_0) + \Delta B_{w_0}(u_2,u_3) + \Delta B_{w_1}(u_3,u_2) = 0\right) \\
&= \sum_{y=-2m}^{2m} \Pr\left(\Delta B_{w_0}(u_0,u_1) + \Delta B_{w_1}(u_1,u_0) = y\right) \\
&\times \Pr\left(\Delta B_{w_0}(u_2,u_3) + \Delta B_{w_1}(u_3,u_2) = -y\right). \qquad \text{(B.10)}
\end{aligned}$$

Now, we proceed the proof with the rest of TBC walk structures in Fig. 3.2(f)-(i) as follows:

- Fig. 3.2(f)

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\
&= \Pr\left(B_{w_0,u_1} - B_{w_1,u_1} + B_{w_1,u_0} - B_{w_2,u_0} + B_{w_2,u_1} - B_{w_1,u_1} + B_{w_1,u_0} - B_{w_0,u_0} = 0\right) \\
&= \Pr\left(\Delta B_{w_0}(u_1,u_0) + \Delta B_{w_1}(u_0,u_1) + \Delta B_{w_2}(u_1,u_0) + \Delta B_{w_1}(u_0,u_1) = 0\right). \qquad \text{(B.11)}
\end{aligned}$$

One can see that in this case, the term $\Delta B_{w_1}(u_0,u_1)$ is repeated twice in (B.11). Let us denote $X = \Delta B_{w_1}(u_0,u_1) + \Delta B_{w_1}(u_0,u_1)$ and $Z = \Delta B_{w_0}(u_1,u_0) +$

$\Delta B_{w_2}(u_1, u_0)$. One could see that the random variables X and Z are independent from each other. Then,

$$\sum_{t=-2m}^{2m} \Pr(X=-t)\Pr(Z=t|X=-t) = \sum_{t=-2m}^{2m} \Pr(X=-t)\Pr(Z=t) =$$
$$\sum_{t=-2m}^{2m} \Pr\big(2\Delta B_{w_1}(u_1,u_0) = -t\big)\Pr\big(\Delta B_{w_2}(u_1,u_0) + \Delta B_{w_0}(u_1,u_0) = t\big)$$
$$\overset{\text{define } y=\frac{t}{2}}{=} \sum_{y=-m}^{m} \Pr\big(\Delta B_{w_1}(u_1,u_0) = -y\big)\Pr\big(\Delta B_{w_2}(u_1,u_0) + \Delta B_{w_0}(u_1,u_0) = 2y\big). \tag{B.12}$$

Now, depending on the value of m being even or odd we proceed the calculations in either of the following two ways. Also, for the sake of simplicity, here we reduce the notation $\Delta B_{w_1}(u_1, u_0)$, $\Delta B_{w_2}(u_1, u_0)$ and $\Delta B_{w_0}(u_1, u_0)$ to ΔB_{w_1}, ΔB_{w_2} and ΔB_{w_0} , respectively.

– Odd m

$$\begin{aligned}
&\Pr(\text{The TBC walk}\,\mathcal{C}_8\,\text{remains in SC protograph})\\
&= \sum_{y=-m}^{\frac{-m-1}{2}} \Pr(\Delta B_{w_1} = -y)\Pr(\Delta B_{w_2} + \Delta B_{w_0} = 2y)\\
&+ \sum_{y=\frac{-m+1}{2}}^{-1} \Pr(\Delta B_{w_1} = -y)\Pr(\Delta B_{w_2} + \Delta B_{w_0} = 2y)\\
&+ \sum_{y=0}^{\frac{m-1}{2}} \Pr(\Delta B_{w_1} = -y)\Pr(\Delta B_{w_2} + \Delta B_{w_0} = 2y)\\
&+ \sum_{y=\frac{m+1}{2}}^{m} \Pr(\Delta B_{w_1} = -y)\Pr(\Delta B_{w_2} + \Delta B_{w_0} = 2y)\\
&= \sum_{y=-m}^{\frac{-m-1}{2}} \frac{(m+1)+y}{(m+1)^2} T_1(2y) + \sum_{y=\frac{-m+1}{2}}^{-1} \frac{(m+1)+y}{(m+1)^2} T_2(2y)\\
&+ \sum_{y=0}^{\frac{m-1}{2}} \frac{(m+1)-y}{(m+1)^2} T_3(2y) + \sum_{y=\frac{m+1}{2}}^{m} \frac{(m+1)-y}{(m+1)^2} T_4(2y)
\end{aligned}$$

$$= \frac{23m^4 + 92m^3 + 148m^2 + 112m + 45}{60(m+1)^5}. \tag{B.13}$$

Where T_i denotes the i-th term in the equation (3.10) in Lemma 6.

– Even m

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\
&= \sum_{y=-m}^{\frac{-m}{2}-1} \Pr(\Delta B_{w_1} = -y) \Pr(\Delta B_{w_2} + \Delta B_{w_0} = 2y) \\
&+ \sum_{y=\frac{-m}{2}}^{-1} \Pr(\Delta B_{w_1} = -y) \Pr(\Delta B_{w_2} + \Delta B_{w_0} = 2y) \\
&= \sum_{y=0}^{\frac{m}{2}-1} \Pr(\Delta B_{w_1} = -y) \Pr(\Delta B_{w_2} + \Delta B_{w_0} = 2y) \\
&+ \sum_{y=\frac{m}{2}}^{m} \Pr(\Delta B_{w_1} = -y) \Pr(\Delta B_{w_2} + \Delta B_{w_0} = 2y) \\
&= \frac{23m^4 + 92m^3 + 148m^2 + 112m + 60}{60(m+1)^5}.
\end{aligned} \tag{B.14}$$

- Fig. 3.2(g)

$$\begin{aligned}
&\Pr\big(\text{The TBC walk } \mathcal{C}_8 \text{ exists in SC protograph}\big) \\
&= \Pr\big(B_{w_0,u_0} - B_{w_1,u_0} + B_{w_1,u_2} - B_{w_0,u_2} + B_{w_0,u_1} - B_{w_1,u_1} + B_{w_1,u_2} - B_{w_0,u_2} = 0\big) \\
&= \Pr\big(2(B_{w_1,u_2} - B_{w_0,u_2}) + B_{w_0,u_0} - B_{w_1,u_0} + B_{w_0,u_1} - B_{w_1,u_1} = 0\big) \\
&= \Pr\big(2\Delta B^v_{u_2}(w_1, w_0) + \Delta B^v_{u_0}(w_0, w_1) + \Delta B^v_{u_1}(w_0, w_1) = 0\big).
\end{aligned} \tag{B.15}$$

The notation $\Delta B^v_{u_j}(w_i, w_{i'})$ used above refers to the *vertical differential parameter*, with the probability function similar to (3.4). Now, we consider $X = \Delta B^v_{u_2}(w_1, w_0)$ and $Z = \Delta B^v_{u_0}(w_0, w_1) + \Delta B^v_{u_1}(w_0, w_1)$ and the rest of the calculations is similar to what we presented for Fig. 3.2(f). Thus, we have

$$\begin{aligned}
&\Pr\big(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}\big) \\
&= \sum_{t=-2m}^{2m} \Pr(X = -t) \Pr(Z = t | X = -t) = \sum_{t=-2m}^{2m} \Pr(X = -t) \Pr(Z = t)
\end{aligned}$$

$$= \begin{cases} \frac{23m^4+92m^3+148m^2+112m+45}{60(m+1)^5} & \text{odd } m \\ \frac{23m^4+92m^3+148m^2+112m+60}{60(m+1)^5} & \text{even } m \end{cases} \tag{B.16}$$

- Fig. 3.2(h)

$$\begin{aligned} &\Pr\big(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}\big) \\ &= P\big(\Delta B_{w_0}(u_1,u_0) + \Delta B_{w_1}(u_0,u_1) + \Delta B_{w_0}(u_1,u_0) + \Delta B_{w_1}(u_0,u_1) = 0\big) \\ &= \Pr\big(2[\Delta B_{w_0}(u_1,u_0) + \Delta B_{w_1}(u_0,u_1)] = 0\big) \\ &= \Pr\big(\Delta B_{w_0}(u_1,u_0) + \Delta B_{w_1}(u_0,u_1) = 0\big) = \frac{2m^2+4m+3}{3(m+1)^3}. \end{aligned} \tag{B.17}$$

Which is the same as the probability of existence of 4-cycles, given in Theorem 1.

- Fig. 3.2(i)

$$\begin{aligned} &\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\ &= \Pr\big(B_{w_0,u_1} - B_{w_2,u_1} + B_{w_2,u_0} - B_{w_1,u_0} + B_{w_1,u_2} - B_{w_2,u_2} + B_{w_2,u_0} - B_{w_0,u_0} = 0\big) \\ &= \sum_{y=0}^{m} \Pr(B_{w_2,u_0} = y) \times \\ &\Pr\big(B_{w_0,u_1} - B_{w_2,u_1} + 2B_{w_2,u_0} - B_{w_1,u_0} + B_{w_1,u_2} - B_{w_2,u_2} - B_{w_0,u_0} = 0 | B_{w_2,u_0} = y\big) \\ &= \sum_{y=0}^{m} \Pr(B_{w_2,u_0} = y) \Pr\big((B_{w_0,u_1} - B_{w_2,u_1}) + (B_{w_1,u_2} - B_{w_2,u_2}) - (B_{w_1,u_0} + B_{w_0,u_0}) = -2y\big) \\ &= \sum_{y=0}^{m} \Pr(B_{w_2,u_0} = y) \Pr\big(\Delta B^v_{u_1}(w_0,w_2) + \Delta B^v_{u_2}(w_1,w_2) - (B_{w_1,u_0} + B_{w_0,u_0}) = -2y\big). \end{aligned} \tag{B.18}$$

Now, consider $S = (B_{w_1,u_0} + B_{w_0,u_0})$. This random variable has the pmf given in (3.8). According to (3.8) we can continue (B.18) as follows:

$$\begin{aligned} &\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\ &= \sum_{y=0}^{m} \Pr(B_{w_2,u_0} = y) \Pr(\Delta B^v_{u_1}(w_0,w_2) + \Delta B^v_{u_2}(w_1,w_2) - S = -2y) \\ &= \frac{1}{m+1} \sum_{y=0}^{m} \sum_{z=-2m}^{2m} \Pr(\Delta B^v_{u_1} + \Delta B^v_{u_2} = z) \Pr(S = z + 2y). \end{aligned} \tag{B.19}$$

Where we used the fact that two random variables S and $\Delta B^v_{u_1} + \Delta B^v_{u_2}$ are independent as they do not share any common edge. Here, we need to break down the sigma function according to the equations in (3.10) and (3.8). Given that $0 \leq y \leq m$ and $-2m \leq z \leq 2m$, we have $-2m \leq z + 2y \leq 4m$. On the other hand, referring to (3.8) we know that $0 \leq z + 2y \leq 2m$, and so $-2y \leq z \leq 2m - 2y$. Therefore, we can expand (B.19) as follows based on the values of m:

– Odd m

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\
&= \frac{1}{m+1} \sum_{y=0}^{\frac{m-1}{2}} \Big(\sum_{z=-2y}^{0} T_2(z) J_1(z+2y) + \sum_{z=1}^{m-2y} T_3(z) J_1(z+2y) \\
&+ \sum_{z=m-2y+1}^{m} T_3(z) J_2(z+2y) + \sum_{z=m+1}^{2m-2y} T_4(z) J_2(z+2y) \Big) \\
&+ \frac{1}{m+1} \sum_{y=\frac{m+1}{2}}^{m} \Big(\sum_{z=-2y}^{-m} T_1(z) J_1(z+2y) + \sum_{z=-m+1}^{m-2y} T_2(z) J_1(z+2y) \\
&+ \sum_{z=m-2y+1}^{-1} T_2(z) J_2(z+2y) + \sum_{z=0}^{2m-2y} T_3(z) J_2(z+2y) \Big), \qquad \text{(B.20)}
\end{aligned}$$

where, T_i and J_i refer to the i-th term in the equations (3.10) and (3.8), respectively. Substituting the corresponding functions in (B.20) we get

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\
&= \sum_{y=0}^{\frac{m-1}{2}} \Big(\frac{1}{60(m+1)^7} (13m^5 + 50m^4 y + 90m^4 + 40m^3 y^2 + 240m^3 y + 235m^3 - \\
&80m^2 y^3 + 390m^2 y + 300m^2 - 160my^4 - 480my^3 - 240my^2 + 300my + 202m \\
&+ 160y^5 + 240y^4 - 200y^3 - 300y^2 + 40y + 60) \Big) \\
&+ \sum_{y=\frac{m+1}{2}}^{m} \Big(\frac{1}{60(m+1)^7} (23m^5 - 50m^4 y + 90m^4 + 440m^3 y^2 + 24m^3 y + 185m^3 \\
&- 880m^2 y^3 + 690m^2 y + 300m^2 + 640my^4 - 480my^3 - 840my^2 + 300my + 242m \\
&- 160y^5 + 240y^4 + 200y^3 - 300y^2 - 40y + 60) \Big)
\end{aligned}$$

$$= \frac{302m^4 + 1208m^3 + 1937m^2 + 1458m + 495}{720(m+1)^5}. \tag{B.21}$$

– Even m

$$\begin{aligned}
&\Pr(\text{The TBC walk } \mathcal{C}_8 \text{ remains in SC protograph}) \\
&= \frac{1}{m+1} \sum_{y=0}^{\frac{m}{2}-1} \Big(\sum_{z=-2y}^{0} T_2(z) J_1(z+2y) + \sum_{z=1}^{m-2y} T_3(z) J_1(z+2y) \\
&+ \sum_{z=m-2y+1}^{m} T_3(z) J_2(z+2y) + \sum_{z=m+1}^{2m-2y} T_4(z) J_2(z+2y) \Big) \\
&+ \frac{1}{m+1} \sum_{y=\frac{m}{2}}^{m} \Big(\sum_{z=-2y}^{-m} T_1(z) J_1(z+2y) + \sum_{z=-m+1}^{m-2y} T_2(z) J_1(z+2y) \\
&+ \sum_{z=m-2y+1}^{-1} T_2(z) J_2(z+2y) + \sum_{z=0}^{2m-2y} T_3(z) J_2(z+2y) \Big) \\
&= \frac{302m^6 + 1812m^5 + 4655m^4 + 6540m^3 + 5348m^2 + 2448m + 720}{720(m+1)^7}.
\end{aligned} \tag{B.22}$$

Appendix C

C.1 Proof of Theorem 6

Suppose that in the Tanner graph of a protograph-based LDPC code we have a $(4,2)$ LETS, as depicted in Fig. 3.6.[1] Consider differential parameters in Fig. 3.6 as follows:

$$\begin{aligned}&\Delta B_1 = B_{w_0,u_1} - B_{w_0,u_0}, \quad \Delta B_2 = B_{w_1,u_2} - B_{w_1,u_1}, \quad \Delta B_3 = B_{w_2,u_0} - B_{w_2,u_2},\\ &\Delta B_4 = B_{w_0,u_3} - B_{w_0,u_2}, \quad \Delta B_5 = B_{w_1,u_0} - B_{w_1,u_3}.\end{aligned} \tag{C.1}$$

This TS consists of two 6-cycles C_{6_1}, C_{6_2} and one 8-cycle C_8. The $(4,2)$ LETS remains unbroken during the edge spreading process and moves to the SC code Tanner graph if and only if $\Delta B_1 + \Delta B_2 + \Delta B_3 = 0$, $\Delta B_4 + \Delta B_5 - \Delta B_3 = 0$, and $\Delta B_1 + \Delta B_2 + \Delta B_4 + \Delta B_5 = 0$. We note that the third equation can be derived from the sum of the first two equations. Thus, the (4,2) LETS exists in the SC code if and only if $\Delta B_1 + \Delta B_2 + \Delta B_3 = 0$, $\Delta B_4 + \Delta B_5 - \Delta B_3 = 0$. Thus, two 6-cycles C_{6_1} and C_{6_2} are the fundamental cycles for this $(4,2)$ LETS. Consequently, we have

$$\begin{aligned}&\Pr\big((4,2)\,\text{LETS in block code remains in SC code}\big)\\ &= \Pr(C_{6_1}\,\text{is unbroken}) \times \Pr(C_{6_2}\,\text{is unbroken} \mid C_{6_1}\,\text{is unbroken})\end{aligned} \tag{C.2}$$

Thus,

$$\begin{aligned}&\Pr\big((4,2)\,\text{remains in SC code}\big)\\ &= \Pr(\Delta B_1 + \Delta B_2 + \Delta B_3 = 0)\Pr(\Delta B_4 + \Delta B_5 - \Delta B_3 = 0 \mid \Delta B_1 + \Delta B_2 + \Delta B_3 = 0)\end{aligned} \tag{C.3}$$

[1] Check node indices for the LETS structure in Fig. 3.6 are assigned such that two check nodes with a common adjacent variable nodes have different indices.

Thus,

$$\begin{aligned}\Pr\big((4,2)\,\text{remains in SC code}\big) &= \Pr(\Delta B_1+\Delta B_2+\Delta B_3=0)\times\\ &\Pr(\Delta B_4+\Delta B_5-\Delta B_3=0\mid \Delta B_1+\Delta B_2+\Delta B_3=0).\end{aligned}\tag{C.4}$$

Next, we calculate $\Pr(\Delta B_4+\Delta B_5-\Delta B_3=0\mid \Delta B_1+\Delta B_2+\Delta B_3=0)$. We use the law of total probability for conditional probabilities which says that $\Pr(A|C)=\sum_n \Pr(A|C\cap B_n)\Pr(B_n|C)$. Therefore, we have

$$\begin{aligned}&\Pr(\Delta B_4+\Delta B_5-\Delta B_3=0\mid \Delta B_1+\Delta B_2+\Delta B_3=0)\\ &=\sum_{y=-m}^{m}\Big(\Pr(\Delta B_4+\Delta B_5-\Delta B_3=0\mid \Delta B_1+\Delta B_2+\Delta B_3=0,\Delta B_3=y)\\ &\times\Pr(\Delta B_3=y\mid \Delta B_1+\Delta B_2+\Delta B_3=0)\Big)\end{aligned}\tag{C.5}$$

Now, in (C.5) we need to discuss the dependence of $\Delta B_4+\Delta B_5-\Delta B_3$ and $\Delta B_1+\Delta B_2+\Delta B_3$. Let us expand $\Delta B_4+\Delta B_5$ and $\Delta B_1+\Delta B_2$ as follows:

$$\begin{aligned}\Delta B_4+\Delta B_5 &= B_{w_0,u_3}-B_{w_1,u_3}+B_{w_1,u_0}-B_{w_0,u_2},\\ \Delta B_1+\Delta B_2 &= B_{w_0,u_1}-B_{w_1,u_1}+B_{w_1,u_2}-B_{w_0,u_0}.\end{aligned}\tag{C.6}$$

Now, based on the fact that $u_0\neq u_1$, $u_0\neq u_2$, $u_1\neq u_2$, $u_0\neq u_3$ and $u_2\neq u_3$,[2] we could have two different scenarios as follows:

$$\begin{cases}\text{Case 1: if } u_1=u_3 \quad , \ \Delta B_4+\Delta B_5 \text{ and } \Delta B_1+\Delta B_2 \text{ are dependent}\\ \text{Case 2: if } u_1\neq u_3 \quad , \ \Delta B_4+\Delta B_5 \text{ and } \Delta B_1+\Delta B_2 \text{ are independent}\end{cases}\tag{C.7}$$

On the other hand, according to the check nodes and variable nodes labeling in Fig. 3.6, one can verify that the 8-cycle C_8 could be an inverse image of the TBC walks in Fig. 3.2(e) or (g), existing in the block code base graph. The structure in Fig. 3.2(g) has three different variable node indices, while the structure in Fig. 3.2(e) consists of four different variable node indices. Thus, the former belongs to Case 1 while the latter is in the Case 2 category. Referring to Theorem 3 and Fig. 3.3, it is evident

[2]Variable nodes that are adjacent to a common check node should have different indices.

that the probability of existence for the structure of Fig. 3.2(e) is larger than that of Fig. 3.2(g). So, we can assume the scenario in Case 1 as a lower bound and the scenario in Case 2 as an upper bound for the probability of existence of a $(4, 2)$ LETS in a protograph-based SC code.

Let us start with the Case 1, where it is assumed that $u_1 = u_3$. In this case, $\Pr(\Delta B_4 + \Delta B_5 - \Delta B_3 = 0 | \Delta B_1 + \Delta B_2 + \Delta B_3 = 0, \Delta B_3 = y)$ in (C.5) is expanded as

$$\sum_{y=-m}^{m} \Pr(B_{w_0,u_3} - B_{w_1,u_3} + B_{w_1,u_0} - B_{w_0,u_2} = y | B_{w_0,u_1} - B_{w_1,u_1} + B_{w_1,u_2} - B_{w_0,u_0} = -y). \tag{C.8}$$

For the sake of simplicity, denote $X = B_{w_0,u_3} - B_{w_1,u_3}$, $t_1 = B_{w_1,u_0} - B_{w_0,u_2}$ and $t_2 = B_{w_1,u_2} - B_{w_0,u_0}$. Then, according to the fact that $u_1 = u_3$, (C.8) becomes

$$\begin{aligned}
&\sum_{y=-m}^{m} \Pr(X + t_1 = y | X + t_2 = -y) \\
&= \sum_{y=-m}^{m} \sum_{x=-m}^{m} \Pr(X + t_1 = y | X + t_2 = -y, X = x) \Pr(X = x \mid X + t_2 = -y) \\
&= \sum_{y=-m}^{m} \sum_{x=-m}^{m} \Pr(X + t_1 = y | t_2 = -X - y, X = x) \frac{\Pr(X = x, x + t_2 = -y)}{\Pr(X + t_2 = -y)} \\
&= \sum_{y=-m}^{m} \sum_{x=-m}^{m} \Pr(t_1 = y - X | t_2 = -X - y, X = x) \frac{\Pr(X = x) \Pr(x + t_2 = -y)}{\Pr(X + t_2 = -y)} \\
&= \sum_{y=-m}^{m} \sum_{x=-m}^{m} \Pr(t_1 = y - x) \frac{\Pr(X = x) \Pr(t_2 = -x - y)}{\Pr(X + t_2 = -y)}
\end{aligned} \tag{C.9}$$

Where, in the last equality we used the fact that t_1 and t_2 are two independent random variables, as they do not share any edge with each other. Substituting (C.9) into (C.5), the initial equation (C.4) turns into

$$\begin{aligned}
&\Pr\big((4, 2) \text{ TS remains in SC code}\big) \\
&= \Pr(\Delta B_1 + \Delta B_2 + \Delta B_3 = 0) \times \\
&\sum_{y=-m}^{m} \sum_{x=-m}^{m} \Pr(t_1 = y - x) \frac{\Pr(X = x) \Pr(t_2 = -x - y)}{\Pr(X + t_2 = -y)} \Pr(\Delta B_3 = y \mid \Delta B_1 + \Delta B_2 + \Delta B_3 = 0)\Big)
\end{aligned}$$

$$
\begin{aligned}
&= \Pr(\Delta B_1 + \Delta B_2 + \Delta B_3 = 0)\times \\
&\sum_{y=-m}^{m}\sum_{x=-m}^{m} \Pr(t_1 = y - x)\frac{\Pr(X = x)\Pr(t_2 = -x - y)}{\Pr(X + t_2 = -y)}\frac{\Pr(\Delta B_3 = y, \Delta B_1 + \Delta B_2 = -y)}{\Pr(\Delta B_1 + \Delta B_2 + \Delta B_3 = 0)} \\
&= \Pr(\Delta B_1 + \Delta B_2 + \Delta B_3 = 0)\times \\
&\sum_{y=-m}^{m}\sum_{x=-m}^{m} \Pr(t_1 = y - x)\frac{\Pr(X = x)\Pr(t_2 = -x - y)}{\Pr(X + t_2 = -y)}\frac{\Pr(\Delta B_3 = y, X + t_2 = -y)}{\Pr(\Delta B_1 + \Delta B_2 + \Delta B_3 = 0)} \\
&= \sum_{y=-m}^{m}\sum_{x=-m}^{m} \Pr(t_1 = y - x)\frac{\Pr(X = x)\Pr(t_2 = -x - y)}{\Pr(X + t_2 = -y)}\Pr(\Delta B_3 = y, X + t_2 = -y) \\
&= \sum_{y=-m}^{m}\sum_{x=-m}^{m} \Pr(t_1 = y - x)\frac{\Pr(X = x)\Pr(t_2 = -x - y)}{\Pr(X + t_2 = -y)}\Pr(\Delta B_3 = y)\Pr(X + t_2 = -y)
\end{aligned}
\tag{C.10}
$$

Now, we need to discuss the eligible values for x and y according to the probability distribution function given in (3.4). Consider $-m \le x \le m$, and the fact that $|t_1 = -x + y| \le m$ and $|t_2 = -x - y| \le m$ in (C.10), we have

$$
\begin{cases}
|-x + y| \le m \rightarrow -m + y \le x \le m + y \\
|-x - y| \le m \rightarrow -m - y \le x \le m - y
\end{cases}
\tag{C.11}
$$

Depending on the values of y, the following conclusion is achieved:

$$
\begin{cases}
-m - y \le x \le m + y & \text{if } y < 0 \\
-m + y \le x \le m - y & \text{if } y \ge 0
\end{cases}
\tag{C.12}
$$

Thus, we can expand (C.10) as follows:

$$
\begin{aligned}
&\sum_{y=-m}^{m}\sum_{x=-m}^{m} \Pr(t_1 = y - x)\frac{\Pr(X = x)\Pr(t_2 = -x - y)}{\Pr(X + t_2 = -y)}\Pr(\Delta B_3 = y)\Pr(X + t_2 = -y) \\
&= \sum_{y=-m}^{-1}\frac{(m+1)+y}{(m+1)^2}\sum_{x=-m-y}^{m+y} \Pr(t_1 = y - x)\Pr(X = x)\Pr(t_2 = -x - y) \\
&+ \sum_{y=0}^{m}\frac{(m+1)-y}{(m+1)^2}\sum_{x=-m+y}^{m-y} \Pr(t_1 = y - x)\Pr(X = x)\Pr(t_2 = -x - y)
\end{aligned}
\tag{C.13}
$$

Now, we use the probability density function given in (3.4) to proceed the calculations.[3] For two cases of odd and even values of m we present the following results:

- Even m

$$
\begin{aligned}
&\Pr\big((4,2)\,\text{LETS remains in SC code Tanner graph}\big)\\
&=\sum_{y=-m}^{-\frac{m}{2}-1}\frac{(m+1)+y}{(m+1)^2}\Big(\sum_{x=-m-y}^{-1}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\\
&+\sum_{x=0}^{m+y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\Big)\\
&+\sum_{y=-\frac{m}{2}}^{-1}\frac{(m+1)+y}{(m+1)^2}\Big(\sum_{x=-m-y}^{y}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\\
&+\sum_{x=y+1}^{-1}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\\
&+\sum_{x=0}^{-y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\\
&+\sum_{x=-y+1}^{m+y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\Big)\\
&+\sum_{y=0}^{\frac{m}{2}}\frac{(m+1)-y}{(m+1)^2}\Big(\sum_{x=-m+y}^{-y}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\\
&+\sum_{x=-y+1}^{-1}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\\
&+\sum_{x=0}^{y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\\
&+\sum_{x=y+1}^{m-y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\Big)\\
&+\sum_{y=\frac{m}{2}+1}^{m}\frac{(m+1)-y}{(m+1)^2}\Big(\sum_{x=-m+y}^{-1}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\\
&+\sum_{x=0}^{m-y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\Big)\\
&=\frac{7m^4+28m^3+48m^2+40m+24}{24(m+1)^6}.
\end{aligned}
\tag{C.14}
$$

Similarly, for odd values of m we have

[3]It should be noticed that the probability distribution function for differential parameter is valid for the difference of any pair of B_{i_1,j_1} and B_{i_2,j_2} such that $i_1 \neq i_2$ or $j_1 \neq j_2$.

- Odd m

$$
\begin{aligned}
&\Pr\big((4,2)\,\text{LETS remains in SC code Tanner graph}\big)\\
&=\sum_{y=-m}^{\frac{-m+1}{2}-1}\frac{(m+1)+y}{(m+1)^2}\Big(\sum_{x=-m-y}^{-1}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\\
&+\sum_{x=0}^{m+y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\Big)\\
&+\sum_{y=\frac{-m+1}{2}}^{-1}\frac{(m+1)+y}{(m+1)^2}\Big(\sum_{x=-m-y}^{y}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\\
&+\sum_{x=y+1}^{-1}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\\
&+\sum_{x=0}^{-y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\\
&+\sum_{x=-y+1}^{m+y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\Big)\\
&+\sum_{y=0}^{\frac{m-1}{2}}\frac{(m+1)-y}{(m+1)^2}\Big(\sum_{x=-m+y}^{-y}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)-(-x-y)}{(m+1)^2}\\
&+\sum_{x=-y+1}^{-1}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\\
&+\sum_{x=0}^{y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\\
&+\sum_{x=y+1}^{m-y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)+(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\Big)\\
&+\sum_{y=\frac{m-1}{2}+1}^{m}\frac{(m+1)-y}{(m+1)^2}\Big(\sum_{x=-m+y}^{-1}\frac{(m+1)+x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\\
&+\sum_{x=0}^{m-y}\frac{(m+1)-x}{(m+1)^2}\times\frac{(m+1)-(y-x)}{(m+1)^2}\times\frac{(m+1)+(-x-y)}{(m+1)^2}\Big)\\
&=\frac{7m^4+28m^3+48m^2+40m+21}{24(m+1)^6}.
\end{aligned}
\tag{C.15}
$$

Now consider Case 2, where all variable node indices $u_0,\ldots,u_3$ are distinct, $\Delta B_4+\Delta B_5$ and $\Delta B_1+\Delta B_2$ have no common element and thus, these two random variables are independent. In this case, the calculation of $\Pr(\Delta B_4+\Delta B_5-\Delta B_3=0\mid\Delta B_1+\Delta B_2+\Delta B_3=$

0) is as follows

$$
\begin{aligned}
&\Pr(\Delta B_4+\Delta B_5-\Delta B_3=0 \mid \Delta B_1+\Delta B_2+\Delta B_3=0)= \\
&\sum_{y=-m}^{m}\Big(\Pr(\Delta B_4+\Delta B_5-\Delta B_3=0|\Delta B_1+\Delta B_2+\Delta B_3=0,\Delta B_3=y) \\
&\times \Pr(\Delta B_3=y|\Delta B_1+\Delta B_2+\Delta B_3=0)\Big) \\
&=\sum_{y=-m}^{m}\Big(\Pr(\Delta B_4+\Delta B_5-\Delta B_3=0|\Delta B_3=y)\Pr(\Delta B_3=y|\Delta B_1+\Delta B_2+\Delta B_3=0)\Big) \\
&=\sum_{y=-m}^{m}\Big(\Pr(\Delta B_4+\Delta B_5=y)\Pr(\Delta B_3=y|\Delta B_1+\Delta B_2+\Delta B_3=0)\Big) \\
&=\sum_{y=-m}^{m}\Big(\Pr(\Delta B_4+\Delta B_5=y)\frac{\Pr(\Delta B_3=y,\Delta B_1+\Delta B_2=-y)}{\Pr(\Delta B_1+\Delta B_2+\Delta B_3=0)} \\
&=\frac{\sum_{y=-m}^{m}\Big(\Pr(\Delta B_4+\Delta B_5=y)\Pr(\Delta B_3=y)\Pr(\Delta B_1+\Delta B_2=-y)\Big)}{\Pr(\Delta B_1+\Delta B_2+\Delta B_3=0)}. \qquad \text{(C.16)}
\end{aligned}
$$

Consequently, by (C.4) and (C.16), we have

$$
\begin{aligned}
&\Pr\big((4,2)\,\text{LETS remains in SC code Tanner graph}\big) \\
&=\sum_{y=-m}^{m}\Big(\Pr(\Delta B_4+\Delta B_5=y)\Pr(\Delta B_3=y)\Pr(\Delta B_1+\Delta B_2=-y)\Big) \\
&=\sum_{y=-m}^{-1}\Big(\frac{(m+1)+y}{(m+1)^2}\Big[\frac{1}{6(m+1)^4}\big(-3y^3-6y^2m-6y^2+3y+4m^3+12m^2+14m+6\big)\Big] \\
&\times\Big[\frac{1}{6(m+1)^4}\big(3(-y)^3-6y^2m-6y^2-3(-y)+4m^3+12m^2+14m+6\big)\Big]\Big) \\
&+\sum_{y=0}^{m}\Big(\frac{(m+1)-y}{(m+1)^2}\Big[\frac{1}{6(m+1)^4}\big(-3y^3-6y^2m-6y^2+3y+4m^3+12m^2+14m+6\big)\Big] \\
&\times\Big[\frac{1}{6(m+1)^4}\big(3(-y)^3-6y^2m-6y^2-3(-y)+4m^3+12m^2+14m+6\big)\Big]\Big) \\
&=\frac{1597m^6+9582m^5+25369m^4+37596m^3+33832m^2+18024m+5040}{5040(m+1)^8}. \qquad \text{(C.17)}
\end{aligned}
$$

In the above calculations we used (3.10) and Faulhaber's formula in (A.1).

Appendix D

D.1 Proof of Proposition 10

Avoiding all Type-A 8-cycles and making all 8-cycles and 10-cycles LSS-nE (to fulfill condition (iii) and a part of condition (i)) result in the elimination of all LETSs within the range $a \leq 10, b \leq 3$, except 5 non-isomorphic LETS structures in the $(9,3)$ class (given in Fig. D.1), 3 structures in the $(10,0)$ class, and 8 structures in the $(10,2)$ class. The 11 structures of size $a = 10$, are all LSS children of the five $(9,3)$ structures. In the following, we show that the rest of the sufficient conditions of the proposition eliminate 4 (out of 5) of the $(9,3)$ structures, and the only structure that can potentially remain in the $(9,3)$ class is the one in Fig. D.1(e). This implies that out of the 11 structures with $a = 10$, also, only those that are the LSS children of this remaining $(9,3)$ structure can potentially exist in the Tanner graph.

The $(9,3)$ LETS structure in Fig. D.1(a) contains a $(6,4)$ LETS as a substructure. The $(6,4)$ structure consists of two 8-cycles, and since the Tanner graph is free of Type-A

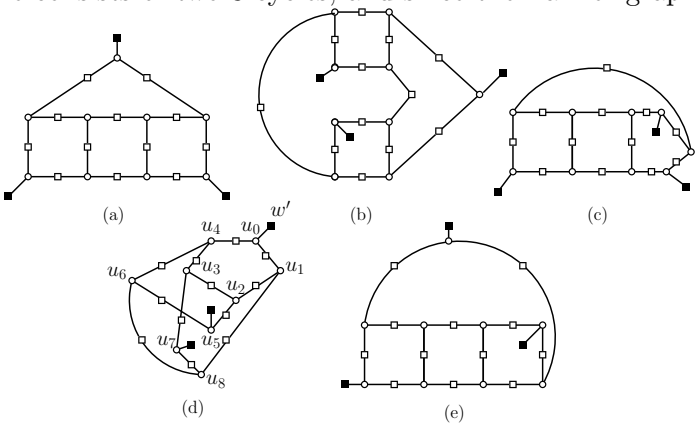


Figure D.1: Five different $(9,3)$ LETS structures in a code with $d_v = 3$ and $g = 8$.

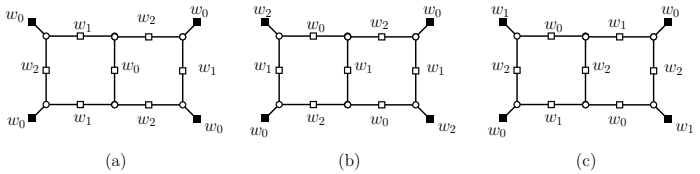


Figure D.2: Three possible labelings for the $(6, 4)$ LETS structure in a code with $d_v = 3$ and $g = 8$. Structure (a) consists of one Type-B(w_1) 8-cycle and one Type-B(w_2) 8-cycle, structures (b) and (c) contain two Type-B(w_1) 8-cycles and two Type-B(w_2) 8-cycles, respectively.

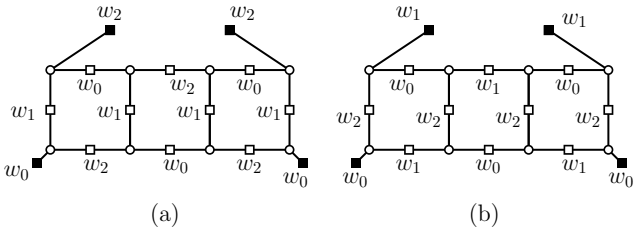


Figure D.3: Two configurations of an $(8, 4)$ LETS structure by the application of pa_2 expansion to (a) $(6, 4)$ LETS configuration in Fig. D.2(b), and (b) $(6, 4)$ LETS configuration in Fig. D.2(c).

8-cycles and Type-B(w_0) 8-cycles, these 8-cycles can only be either Type-B(w_1) or Type-B(w_2). The $(6, 4)$ structure can thus only have one of the three different labelings shown in Fig. D.2(a)-(c), corresponding to one 8-cycle of Type-B(w_1), one 8-cycle of Type-B(w_2); two 8-cycles of Type-B(w_1); or two 8-cycles of Type-B(w_2), respectively. Now to reach the $(9, 3)$ structure of Fig. D.1(a) from the $(6, 4)$ substructure, we consider a sequence of pa_2 and dot_2 expansions, where the intermediate structure (after the application of pa_2) is in the $(8, 4)$ class. The application of pa_2 from either the right side or the left side of Fig. D.2(a) results in a Type-B(w_0) 8-cycle, and is thus not possible. The application of pa_2 to the right side of Fig. D.2(b) and Fig. D.2(c) is shown in Figs. D.3 (a) and (b), respectively (same discussion applies if the pa_2 expansion is applied to the left side). As can be seen, the degree-1 check nodes either at the top or at the bottom of both figures have the same index and thus cannot accommodate the application of a dot_2 expansion. This means that the $(9, 3)$ LETS structure in Fig. D.1(a) cannot exist in the graph.

Now, consider the $(9, 3)$ LETS structure in Fig. D.1(b). This structure contains a 10-cycle at the right side. According to condition (ii), this 10-cycle must be of Type(w_0), which implies only one check node of the cycle has index w_0. Depending on the location of this check node, we thus have one of the scenarios shown in Fig. D.4 for the labeling of the check nodes within the 10-cycle and the two adjacent 8-cycles. One can see that the first four labelings in Figs. D.4(a)-(d) contain at least one Type-B(w_0) 8-cycle in their structures, denoted by a cross "$\times$" sign. Thus, none of these labelings can exist. In addition, the

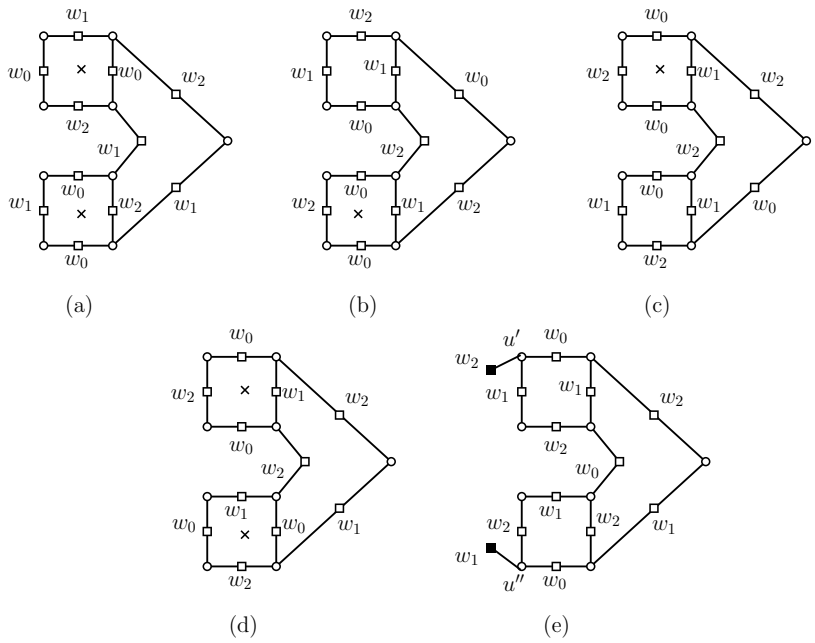


Figure D.4: Different check node lebelings for a substructure of the $(9, 3)$ LETS structure in Fig. D.1(b). Type-B(w_0) 8-cycles are marked by "$\times$" sign.

labeling of Fig. D.4(e) cannot create the $(9, 3)$ structure of Fig. D.1(b) because the two variable nodes u' and u'' are connected to two check nodes with different indices of w_2 and w_1, respectively, in Fig. D.4(e). As a result, the $(9, 3)$ structure of Fig. D.1(b) cannot exist in the graph.

Now, consider the next $(9, 3)$ structure in Fig. D.1(c), which contains the same $(6, 4)$ LETS structure as the one in the structure of Fig. D.1(a). As discussed before, considering the constraints of the proposition, the $(6, 4)$ structure can have one of the three labelings shown in Fig. D.2. Any one of these labelings results in the existence of two check nodes with row index w_0 in the 10-cycle at the top of Fig. D.1(c) (i.e., it creates a 10-cycle(w_1) or a 10-cycle(w_2)). This however, contradicts condition (ii) of the proposition. It is thus not possible to have the $(9, 3)$ structure of Fig. D.1(c) in the graph.

Regarding the $(9, 3)$ structure of Fig. D.1(d), one can verify that this structure contains five 10-cycles sharing some edges with each other. The sequence of variable nodes for these 10-cycles are "u_0, u_1, u_2, u_3, u_4", "u_4, u_3, u_2, u_5, u_6", "u_1, u_2, u_3, u_7, u_8", "u_4, u_3, u_7, u_8, u_6", and "u_1, u_2, u_5, u_6, u_8". Consider one of these constituent 10-cycles. Based on the condition (ii) of the proposition, this cycle must be a 10-cycle(w_0). This implies that three of the degree-1 check nodes connected to the variable nodes of this cycle have index w_0. As a result, it can then be seen that at least one of the other constituent 10-cycles in the structure will have two check nodes with index w_0, violating condition (ii) of the proposition. This is explained in Fig. D.5, where the initial cycle, which is assumed to be a 10-cycle(w_0), is

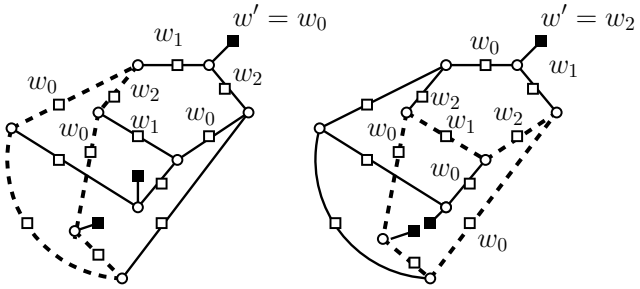


Figure D.5: Possible labelings for the $(9, 3)$ LETS structure of Fig. D.1(d).

shown at the top, with the two possibilities that for the degree-1 check node at the top with index w', we have either $w' = w_0$ or $w' \neq w_0$. As can be seen, in both cases, there is at least one 10-cycle(w_0) in the structure. This cycle is shown by dashed line in Figs. D.5(a) and (b). We thus conclude that the $(9, 3)$ structure of Fig. D.1(d) is also absent from the Tanner graph.

The only $(9, 3)$ structure that can potentially exist in a Tanner graph under the conditions of the proposition is the structure in Fig. D.1(e). The application of dot_3 and dot_2 expansions to this structure results in one structure in the $(10, 0)$ class and two non-isomorphic structures in the $(10, 2)$ class, respectively.

Appendix E

In this part, we prove that the lower bound of $M \geq 7$ to obtain $g = 8$ is not achievable for the SC protograph corresponding to the B matrix in [26], given in Table 7.3.

Considering the values of m, γ and c corresponding to the codes in Table 7.3 and referring to [7], one could see that the lower bound on m to obtain SC base graph with $g \geq 6$ is $m \geq 3$. Thus, $m = 1$ gives an SC protograph with $g = 4$. In our proposed spreading matrix we attempted to minimize the number of 4-cycles, but for the spreading matrix of [26], the minimization was done over the number of 6-cycles. According to the discussion in Section 7.4.2 to find the lower bound for lifting degree to achieve $g = 8$, corresponding to 21 entries in a 3×7 spreading matrix (i.e, 21 edges in the block code base matrix), we need to find the value of $N_4^{e_i}$ as the number of 4-cycles containing edge e_i in the protograph of the corresponding SC code with $L = m + 1$. The set of all these values (sorted in an increasing order) for the B matrix in [26] is

$$N_4^e = \{2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6\}, \tag{E.1}$$

and, corresponding to our optimized B matrix we have the following set

$$N_4^e = \{2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6\}. \tag{E.2}$$

The edges with maximum value of $N_4^{e_i} = 6$ in (E.1) are $e_{2,3}$ and $e_{2,5}$.[1] In Fig. E.1 we illustrate all of these 4-cycles including either $e_{2,3}$ or $e_{2,5}$, according to the B matrix of [26]. Now, we prove that no exponent matrix with $M = 7$ exists to create QC-SC code with $g = 8$ corresponding to this B matrix. Said differently, the following proposition for the B

[1] e_{w_i,u_j} refers to an edge connecting check node with row index w_i to the variable node with column index u_j. So, $e_{2,3}$ denotes an edge connecting check node with row index 2 to the variable node with column index 3.

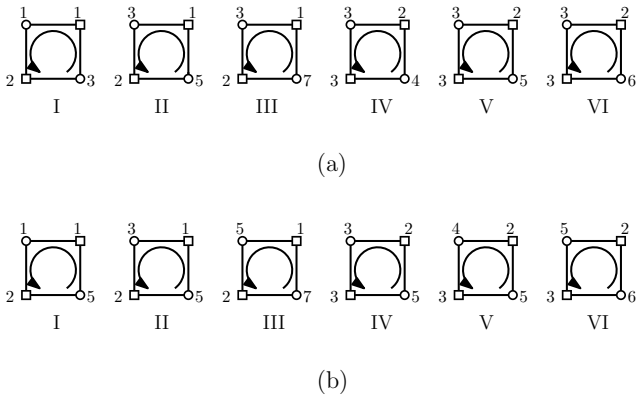


Figure E.1: All 4-cycles in the base graph of a block code that are not broken during the edge spreading process based on the 3×7 B matrix of [26], and contain (a) edge $e_{2,3}$, and (b) edge $e_{2,5}$.

matrix in [26] must be proved:

$$\text{If the QC-SC code has } g = 8 \quad \text{then} \quad M > 7. \tag{E.3}$$

By contraposition, we assume that $M = 7$, and then we show that using this lifting degree it is impossible to achieve $g = 8$ with the spreading matrix in [26] (i.e., the largest girth to achieve is $g = 6$). Thus, we start with the 4-cycles in Fig. E.1 and we prove that it is not possible to assign permutation shifts such that all these 4-cycles and all 6-cycles (made as a combination of every pair of such 4-cycles sharing one edge), have non-zero permutation shifts. In this regard, let us start by assigning 6 different permutation shifts to each 4-cycle in Fig. E.1(a) from the set $\{1, 2, ..., 6\}$.[2] Without loss of generality, let us assume that the permutation shifts for Fig. E.1(a-I) up to Fig. E.1(a-VI) are assigned to be $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ and α_6, respectively, where $1 \leq \alpha_i \leq 6$ and $\alpha_i \neq \alpha_j, \forall i \neq j$. According to the permutation shifts assigned to the cycles in Fig. E.1(a), we need to discuss different possible assignments for the structures in Fig. E.1(b) to check if $g = 8$ could be achieved or not. Since the structures in Fig. E.1(b-II) and Fig. E.1(b-IV) are the same as Fig. E.1(a-II) and Fig. E.1(a-V), respectively, we assign $p_W = \alpha_2$ to the 4-cycle in Fig. E.1(b-II) and $p_W = \alpha_5$ to the cycle in Fig. E.1(b-IV), which are the same permutation shifts as that of their counterparts in Fig. E.1(a).

Now, based on the already assigned permutation sifts to the 4-cycles in Fig. E.1(a), we need to find the permutation shifts for the other 4-cycles in Fig. E.1(b) to check if $g = 8$ is met for the given $M = 7$. According to the permutation shift α_3 assigned to the cycle in

[2] As we assume $M = 7$, every TBC walk of length 4 can have a permutation shift from the set $\{1, ..., M\}$. Note that for avoiding 4-cycles in the lifted graph, we exclude 0 from the set of eligible assignments.

Fig. E.1(a-III), we have

$$\begin{aligned} p_W &= p_{1,3} - p_{1,7} + p_{2,7} - p_{2,3} = \alpha_3 \\ &\rightarrow p_{2,7} - p_{1,7} = \alpha_3 - (p_{1,3} - p_{2,3}). \end{aligned} \tag{E.4}$$

Now, let us find the permutation shift for the 4-cycle in Fig. E.1(b-III) as follows:

$$\begin{aligned} p_W &= p_{1,5} - p_{1,7} + p_{2,7} - p_{2,5} \overset{\text{refer to } (E.4)}{=} \alpha_3 - (p_{1,3} - p_{2,3}) + p_{1,5} - p_{2,5} \\ &= \alpha_3 - (p_{1,3} - p_{1,5} + p_{2,5} - p_{2,3}) = \alpha_3 - \alpha_2, \end{aligned} \tag{E.5}$$

where, the last equality comes from the fact that $p_{1,3} - p_{1,5} + p_{2,5} - p_{2,3}$ defines p_W for the structure in Fig. E.1(a-II), which is assumed to be α_2.

Now, consider another structure in Fig. E.1(b-V). Before starting the calculation for permutation shift of this structure, let us give the following results corresponding to the structure in Fig. E.1(a-V).

$$\begin{aligned} p_W &= p_{2,3} - p_{2,5} + p_{3,5} - p_{3,3} = \alpha_5 \\ &\rightarrow p_{3,5} - p_{2,5} = \alpha_5 - (p_{2,3} - p_{3,3}). \end{aligned} \tag{E.6}$$

According to the result in (E.6), we can find p_W for the structure in Fig. E.1(b-V) as

$$\begin{aligned} p_W &= p_{2,4} - p_{2,5} + p_{3,5} - p_{3,4} = p_{2,4} - p_{3,4} + \underbrace{\big(\alpha_5 - (p_{2,3} - p_{3,3})\big)}_{\text{refer to } (E.6)} \\ &= \alpha_5 - \underbrace{(p_{2,3} - p_{2,4} + p_{3,4} - p_{3,3})}_{\text{refer to Fig. } E.1(a-IV)} = \alpha_5 - \alpha_4. \end{aligned} \tag{E.7}$$

So far we have found the permutation shifts for the 4-cycles in Fig. E.1(b-II)-Fig. E.1(b-V). Now, let us find the permutation shifts for two remaining structures in Fig. E.1(b-I) and Fig. E.1(b-VI). We start with Fig. E.1(b-I). The permutation shift for this structure is expanded as follows:

$$\begin{aligned} p_W &= p_{1,1} - p_{1,5} + p_{2,5} - p_{2,1} \\ &= \underbrace{(p_{1,1} - p_{1,3} + p_{2,3} - p_{2,1})}_{\text{refer to Fig. } E.1(a-I)} + \underbrace{(p_{1,3} - p_{1,5} + p_{2,5} - p_{2,3})}_{\text{refer to Fig. } E.1(a-II)} \\ &= \alpha_1 + \alpha_2. \end{aligned} \tag{E.8}$$

Similarly, for the structure in Fig. E.1(b-VI), one can find the permutation shift as

$$\begin{aligned} p_W &= p_{2,5} - p_{2,6} + p_{3,6} - p_{3,5} \\ &= \underbrace{(p_{2,3} - p_{2,6} + p_{3,6} - p_{3,3})}_{\text{refer to Fig. } E.1(a-VI)} - \underbrace{(p_{2,3} - p_{2,5} + p_{3,5} - p_{3,3})}_{\text{refer to Fig. } E.1(a-V)} \\ &= \alpha_6 - \alpha_5. \end{aligned} \tag{E.9}$$

Considering all structures in Fig. E.1(b), we found the following permutation shifts for each of them:

$$\begin{aligned} &(b-I) : p_W \overset{7}{=} \alpha_1 + \alpha_2 \quad (b-II) : p_W \overset{7}{=} \alpha_2 \\ &(b-III) : p_W \overset{7}{=} \alpha_3 - \alpha_2 \quad (b-IV) : p_W \overset{7}{=} \alpha_5 \\ &(b-V) : p_W \overset{7}{=} \alpha_5 - \alpha_4 \quad (b-VI) : p_W \overset{7}{=} \alpha_6 - \alpha_5 \end{aligned} \tag{E.10}$$

According to (E.10) let us discuss the possibility of permutation shift assignments in Fig. E.1(b) such that no pair of 4-cycles in this figure have the same permutation shift to avoid the generation of 6-cycles in the lifted graph made from these structures. It is clear that Fig. E.1(b-II) and Fig. E.1(b-IV) have the given permutation shifts α_2 and α_5, respectively. Thus, the rest of the structures in Fig. E.1(b) must have different value for permutation shift from the set $\{\alpha_1, \alpha_3, \alpha_4, \alpha_6\}$. One can verify that Fig. E.1(b-I) and Fig. E.1(b-III) can take permutation shifts from $\{\alpha_3, \alpha_4, \alpha_6\}$ and $\{\alpha_1, \alpha_4, \alpha_6\}$, respectively.[3] Likewise, the permutations shifts for Fig. E.1(b-V) and Fig. E.1(b-VI) could be chosen from the set $\{\alpha_1, \alpha_3, \alpha_4, \alpha_6\}$ and $\{\alpha_1, \alpha_3, \alpha_4\}$, respectively. In Fig. E.2 we discuss all these possibilities to check whether we conclude distinct non-zero permutation shifts for all these 4-cycles or not. In this figure, each layer corresponds to one structure in Fig. E.1(b), excluding those in Fig. E.1(b-II) and Fig. E.1(b-IV), as their permutation shifts are fixed. So, the first layer shows all possible p_W assignments for Fig. E.1(b-I), which is selected to be α_3, α_4 or α_6. According to the choice of p_W for this structure, we define the possible p_W for the remaining figures such that there is no pair of 4-cycles in Fig. E.1(b) with the same p_W. At the last layer, corresponding to the permutation shifts for Fig. E.1(b-VI), one can see that

[3]The 4-cycle in Fig. E.1(b-I) has $p_W = \alpha_1 + \alpha_2$. This implies that $p_W \neq \alpha_1$ and $p_W \neq \alpha_2$, because if $p_W = \alpha_1$, then $\alpha_2 = 0$ and if $p_W = \alpha_2$, then $\alpha_2 = 0$. And, we know that neither of α_1 nor α_2 could be zero to avoid 4-cycles in the lifted graph. Also, since p_W must be different than α_5, as it has already been assigned to Fig. E.1(b-IV), the only remaining possible choices for the permutation shift of Fig. E.1(b-I) are $\{\alpha_3, \alpha_4, \alpha_6\}$. Similar discussion is also applicable to Fig. E.1(b-III).

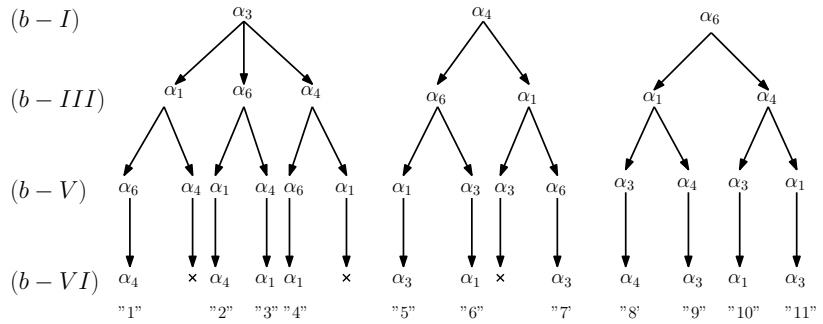


Figure E.2: Different scenarios for assigning permutation shifts to the 4-cycles in Fig. E.1(b-I), Fig. E.1(b-III), Fig. E.1(b-V) and Fig. E.1(b-VI). The "×" sign means that no valid permutation shift, which is different than the permutation shifts of other structures in Fig. E.1(b-I)-Fig. E.1(b-V) could be assigned to Fig. E.1(b-VI).

sometimes no valid assignment is found and thus we show those cases by a cross sign "×".[4] Regarding the other 11 different ways of assignment, where we could achieve an assignment for the last layer, here we show that none of them is valid as they all result in the creation of 4-cycles or 6-cycles in the lifted graph, which contradicts the assumption in (E.3).

We start from assignment "1" in Fig. E.2. This assignment states that Fig. E.1(b-I) has $p_W = \alpha_3$, Fig. E.1(b-III) has $p_W = \alpha_1$, Fig. E.1(b-V) has $p_W = \alpha_6$ and Fig. E.1(b-VI) has $p_W = \alpha_4$. Referring to (E.10) we have

$$\begin{cases} i)\ \alpha_1 + \alpha_2 = \alpha_3 \\ ii)\ \alpha_3 - \alpha_2 = \alpha_1 \\ iii)\ \alpha_5 - \alpha_4 = \alpha_6 \\ iv)\ \alpha_6 - \alpha_5 = \alpha_4 \end{cases} \xrightarrow{\text{substitute } (iii) \text{ in } (iv)} \alpha_5 - \alpha_4 - \alpha_5 = \alpha_4 \to \alpha_4 \overset{7}{=} 0, \tag{E.11}$$

which implies that the 4-cycle in Fig. E.1(a-IV) has a zero permutation shift and thus such a 4-cycle appears in the lifted graph. This contradicts the assumption of $g = 8$.

Now, consider assignment "2". Going through the permutation shifts given in Fig. E.2

[4]This happens when there is no eligible assignment for the last layer (4-cycles in Fig.E.1(b-VI)).

we have

$$\begin{cases} i)\,\alpha_1+\alpha_2=\alpha_3 \\ ii)\,\alpha_3-\alpha_2=\alpha_6 \\ iii)\,\alpha_5-\alpha_4=\alpha_1 \\ iv)\,\alpha_6-\alpha_5=\alpha_4 \end{cases} \overset{(i),(ii)}{\rightarrow} \alpha_1+\alpha_2-\alpha_2=\alpha_6 \rightarrow \alpha_1 \overset{7}{=} \alpha_6. \tag{E.12}$$

From (E.12) it can be seen that two 4-cycles in Fig. E.1(a-I) and Fig. E.1(a-VI) with a common edge $e_{2,3}$ have identical permutation shift, which makes a 6-cycle as a result of their combination in the lifted graph and thus, this assignment is not valid to get $g = 8$.

Regarding assignment "3" and "4" we have the following results, respectively.

$$\begin{cases} i)\,\alpha_1+\alpha_2=\alpha_3 \\ ii)\,\alpha_3-\alpha_2=\alpha_6 \\ iii)\,\alpha_5-\alpha_4=\alpha_4 \\ iv)\,\alpha_6-\alpha_5=\alpha_1 \end{cases} \overset{(i),(ii)}{\rightarrow} \alpha_1+\alpha_2-\alpha_2=\alpha_6 \rightarrow \alpha_1 \overset{7}{=} \alpha_6,$$

$$\begin{cases} i)\,\alpha_1+\alpha_2=\alpha_3 \\ ii)\,\alpha_3-\alpha_2=\alpha_4 \\ iii)\,\alpha_5-\alpha_4=\alpha_6 \\ iv)\,\alpha_6-\alpha_5=\alpha_1 \end{cases} \overset{(i),(ii)}{\rightarrow} \alpha_1+\alpha_2-\alpha_2=\alpha_4 \rightarrow \alpha_1 \overset{7}{=} \alpha_4, \tag{E.13}$$

which both imply the existence of a 6-cycle in the lifted graph.

Now, consider the following results for assignment "5":

$$\begin{cases} i)\,\alpha_1+\alpha_2=\alpha_4 \quad ii)\,\alpha_3-\alpha_2=\alpha_6 \\ iii)\,\alpha_5-\alpha_4=\alpha_1 \quad iv)\,\alpha_6-\alpha_5=\alpha_3 \end{cases}$$
$$\overset{(i),(ii)}{\rightarrow} \alpha_1+\alpha_3=\alpha_6+\alpha_4$$
$$\overset{(iii),(iv)}{\rightarrow} \alpha_1+\alpha_3=\alpha_6-\alpha_4$$
$$\rightarrow \alpha_6+\alpha_4=\alpha_6-\alpha_4 \rightarrow 2\alpha_4 \overset{7}{=} 0 \rightarrow \alpha_4=0. \tag{E.14}$$

The same result is also achieved for the assignment "6", which both contradict the assumption of $g=8$ and thus, neither of them is an accepted assignment.

Now, we move on to assignment "7" with the following conclusions:

$$\begin{cases} i)\,\alpha_1+\alpha_2=\alpha_4 \quad ii)\,\alpha_3-\alpha_2=\alpha_1 \\ iii)\,\alpha_5-\alpha_4=\alpha_6 \quad iv)\,\alpha_6-\alpha_5=\alpha_3 \end{cases}$$
$$\overset{(i),(ii),(iii),(iv)}{\rightarrow} \alpha_1+\alpha_3-\alpha_4+\alpha_6=\alpha_4+\alpha_1+\alpha_6+\alpha_3$$
$$\rightarrow -\alpha_4 \overset{7}{=} \alpha_4 \rightarrow \alpha_4 \overset{7}{=} 0, \tag{E.15}$$

which results in a creation of a 4-cycle in the lifted graph.

For assignment "8" and "9", one may end up with

$$\begin{cases} i)\,\alpha_1+\alpha_2=\alpha_6 \quad ii)\,\alpha_3-\alpha_2=\alpha_1 \\ iii)\,\alpha_5-\alpha_4=\alpha_3 \quad iv)\,\alpha_6-\alpha_5=\alpha_4 \end{cases} \tag{E.16}$$

and,

$$\begin{cases} i)\,\alpha_1+\alpha_2=\alpha_6 \quad ii)\,\alpha_3-\alpha_2=\alpha_1 \\ iii)\,\alpha_5-\alpha_4=\alpha_4 \quad iv)\,\alpha_6-\alpha_5=\alpha_3 \end{cases} \tag{E.17}$$

respectively.

And, regarding the assignment "10" we have

$$\begin{cases} i)\,\alpha_1+\alpha_2=\alpha_6 \quad ii)\,\alpha_3-\alpha_2=\alpha_4 \\ iii)\,\alpha_5-\alpha_4=\alpha_3 \quad iv)\,\alpha_6-\alpha_5=\alpha_1 \end{cases} \tag{E.18}$$

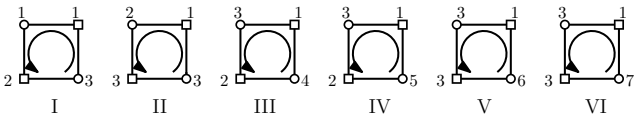


Figure E.3: All 4-cycles in the base graph of a block code, which contain the edge $e_{1,3}$ and are not broken during the edge spreading process using our optimized 3×7 B matrix .

Summing up two sides of equations in (E.16), (E.17) and (E.18), we obtain the same result as in (E.15). And, thus none of them is a valid assignment.

And, finally we reach the following result for assignment "11"

$$\begin{cases} i)\,\alpha_1 + \alpha_2 = \alpha_6 \quad ii)\,\alpha_3 - \alpha_2 = \alpha_4 \\ iii)\,\alpha_5 - \alpha_4 = \alpha_1 \quad iv)\,\alpha_6 - \alpha_5 = \alpha_3 \end{cases}$$
$$\overset{(i),(ii)}{\rightarrow} \alpha_1 + \alpha_3 = \alpha_6 + \alpha_4$$
$$\overset{(iii),(iv)}{\rightarrow} \alpha_1 + \alpha_3 = \alpha_6 - \alpha_4$$
$$\rightarrow \alpha_6 + \alpha_4 = \alpha_6 - \alpha_4 \rightarrow \alpha_4 \overset{7}{=} 0. \tag{E.19}$$

All the conclusions given in (E.11)- (E.19) disrupt the assumption of having $g = 8$ in the lifted graph. Thus, based on the contraposition, we demonstrated the fact that lower bound 7 is not achievable for the SC code with the given spreading matrix in [26].

On the other hand, here we show that for the protograph corresponding to our optimized spreading matrix given in Table 7.3, the lower bound 7 is achievable. To this end, in Fig. E.3, we depict all 6 4-cycles containing $e_{1,3}$ as the only most contributed edge to creating 4-cycles. Then, we show that the exponent matrix P for code C_{10} with the smallest lifting degree $M = 7$ is able to eliminate any 6-cycle that could be made as a combinations of these 4-cycles.

Referring to the P matrix for code C_{10}, the permutation shift values for the 4-cycles in Fig. E.3(I) - Fig. E.3(VI) as the maximum number of 4-cycles sharing a common edge are as follows:

$$(I) : p_W \overset{7}{=} 4 \quad (II) : p_W \overset{7}{=} 3 \quad (III) : p_W \overset{7}{=} 6$$
$$(IV) : p_W \overset{7}{=} 5 \quad (V) : p_W \overset{7}{=} 2 \quad (VI) : p_W \overset{7}{=} 1. \tag{E.20}$$

From (E.20) one can easily verify that none of the 4-cycles in Fig. E.3 results in a 6-cycle in the lifted graph. Considering other 20 edges with smaller contribution in creating 6-cycles, no 6-cycle (and apparently no 4-cycle) appears in the lifted graph using the proposed

exponent matrix for code C_{10}. Thus, $M = 7$ as a lower bound is achieved in this case.