Weight Consistency Matrix Framework for
Non-Binary LDPC Code Optimization

A thesis submitted in partial satisfaction of
the requirements for the degree
Master of Science in Electrical Engineering

by

Chinmayi Radhakrishna Lanka

2016
ABSTRACT OF THE THESIS

Weight Consistency Matrix Framework for Non-Binary LDPC Code Optimization

by

Chinmayi Radhakrishna Lanka

Master of Science in Electrical Engineering
University of California, Los Angeles, 2016
Professor Lara Dolecek, Chair

Transmission channels underlying modern memory systems, e.g., Flash memories, possess a significant amount of asymmetry. While existing LDPC codes optimized for symmetric, AWGN-like channels are being actively considered for Flash applications, we demonstrate that, due to channel asymmetry, such approaches are fairly inadequate. We propose a new, general, combinatorial framework for the analysis and design of non-binary LDPC (NB-LDPC) codes for asymmetric channels. We introduce a refined definition of absorbing sets, which we call general absorbing sets (GASs), and an important subclass of GASs, which we refer to as general absorbing sets of type two (GASTs). Additionally, we study the combinatorial properties of GASTs. We then present the weight consistency matrix (WCM), which succinctly captures key properties in a GAST. Based on these new concepts, we then develop a general code optimization framework, and demonstrate its effectiveness on the realistic Flash channels. Our optimized designs enjoy over one order of magnitude performance gain in the uncorrectable BER (UBER) relative to the unoptimized codes.
The thesis of Chinmayi Radhakrishna Lanka is approved.

Richard Wesel
Abeer Alwan
Lara Dolecek, Committee Chair

University of California, Los Angeles
2016
To my father, Radhakrishna
## Contents

1 Introduction  
1.1 Linear codes ................................................. 1  
1.2 LDPC Codes ................................................. 2  
1.3 Construction of Structured LDPC codes .......................... 3  
1.4 Error Floor in LDPC codes ..................................... 5  
1.4.1 Absorbing sets .............................................. 6  
1.5 Previous Work .................................................. 7  
1.6 Contribution ................................................... 8  

2 General Absorbing Set  
2.1 Motivation ..................................................... 9  
2.2 General Absorbing Set ......................................... 10  
2.3 Matrix-Theoretic representation of GAS .......................... 11  
2.4 General Absorbing Set of Type Two .............................. 12  

3 Analysis of GAST  
3.1 Combinatorial properties of GAST .............................. 15  
3.2 Removal of GAST ............................................... 18  
3.3 Weight Consistency Matrix ..................................... 19  

4 WCM Framework  
4.1 Algorithm to compute WCMs .................................... 22
4.2 Size of WCMs .................................................. 25
  4.2.1 Symmetric GAST ........................................... 25
  4.2.2 Asymmetric GAST ........................................... 26
4.3 NB-LDPC Code Optimization Algorithm ....................... 28
4.4 WCM Customization .............................................. 30
  4.4.1 WCM for Elementary AS ................................. 31
  4.4.2 WCM for BAS .............................................. 31

5 Experimental Results .............................................. 34
  5.1 The Normal-Laplace Mixture Flash (NLMF) Channel ........... 34
    5.1.1 Simulation Results ........................................ 35
  5.2 Another Practical Flash Channel ................................ 38
    5.2.1 Simulation Results ........................................ 38

6 Conclusion ...................................................... 41

References .......................................................... 42
# List of Figures

1.1 Parity check matrix and Tanner graph for the length 5, GF(4) Hamming code ................................................. 2

2.1 Three different (6, 2) NB ASs (column weight 3) ................................................................. 10

2.2 Venn diagram showing different classes of ASs ................................................................. 13

3.1 A (4, 4, 4, 6, 0) GAST, $\gamma = 4$, (left) vs a (6, 0, 0, 9, 0) GAST, $\gamma = 3$, (right). ULGASTs are reached by setting all the weights to 1. ................................................................. 18

4.1 A (6,2,2,11,0) GAST, $\gamma = 4$; Example of a symmetric GAST ........................................ 25

4.2 A (6,2,2,5,2) GAST, $\gamma = 3$; Example of an asymmetric GAST ........................................ 27

4.3 A (4,2,2,5,0) GAST, $\gamma = 3$. Example of a BAS ................................................................. 33

5.1 Threshold voltage distribution of the NLMF channel without (left) and with (right) programming errors. ................................................................. 35

5.2 Simulation results for Code 1 (unoptimized), Code 3 (elementary removal), and Code 5 (WCM framework) over NLMF channel. All the three codes have $\gamma = 3$. ................................................................. 36

5.3 Simulation results for Code 2 (unoptimized), Code 4 (elementary removal), and Code 6 (WCM framework) over NLMF channel. All the three codes have $\gamma = 4$. ................................................................. 36

5.4 Threshold voltage distribution under various P/E cycles using nonparametric kernel density estimation ................................................................. 39
5.5 Simulation results for Code 1 (unoptimized), Code 7 (WCM framework)

over the channel in [27] .............................. 40
List of Tables

5.1 Error profile of Codes 1, 3, and 5 over the NLMF channel, RBER = 4.6e−4,
UBER (unoptimized) = 1.04e−11, and UBER (optimized) = 9.04e−13
over the NLMF channel. .............................................................. 37
5.2 Error profile of Codes 2, 4, and 6 over the NLMF channel, RBER = 1.2e−3,
UBER (unoptimized) = 1.94e−13, and UBER (optimized) = 1.87e−14. 38
5.3 Error profile of Codes 1 and 7 over the channel in [27], RBER = 6.33e-4,
UBER (unoptimized) = 9.86e-12, and UBER (optimized) = 8.12e-13 ... 39
Chapter 1

Introduction

1.1 Linear codes

Channel codes whose codewords create a linear vector space over a field $F$ are called linear codes. In particular, a code $C$ over the field $F$ is called linear if

$$\forall x_1, x_2 \in C; \forall \alpha_1, \alpha_2 \in F : \alpha_1 x_1 + \alpha_2 x_2 \in C \quad (1.1)$$

For any linear code, there exists a matrix called the parity-check matrix (PCM) such that its codewords are in the null space of the PCM. Hence we can define a linear code in terms of the PCM as follows: A linear code $C$ has a set of codewords $x = (x_1, x_2, ..., x_n)$ that satisfy the parity check equation

$$H x^T = 0 \quad (1.2)$$

where $H$ is the $(n - k) \times k$ parity check matrix.

**Definition 1.** The Tanner graph of a code associated with the $(n - k) \times n$ parity-check matrix $H$, is a bipartite graph consisting of $2n - k$ vertices such that each one of $n$ bit nodes correspond to one of the $n$ columns of the matrix $H$ and each one of $n - k$ check nodes correspond to one of the $n - k$ rows of the matrix $H$. There exists an edge between bit node $j$ and check node $i$ if and only if the entry in $i^{th}$ row and $j^{th}$ column of matrix $H$, $h_{ij}$, is nonzero.
The parity-check matrix of code is not unique, because there can be different Tanner graphs representing the same code. The rows of a parity-check matrix of a code define a set of constraints which each codeword of the code must satisfy. Thus, the parity-check matrix can be interpreted as a set of linear equations called the \textit{parity-check equations}. Since any linear combination of the parity-check equations generates another valid parity-check equation, different parity check matrices can represent the same code.

\textbf{Example 1.} Figure 1.1 shows the Tanner graph and the associated PCM of a length 5 Hamming code [1] defined over GF(4), where $\alpha$ is the primitive element of GF($q$).

\begin{align*}
H = c_1 & \begin{bmatrix} 1 & \alpha & \alpha^2 & 0 & 0 \end{bmatrix} \\
& c_2 \begin{bmatrix} 0 & 1 & \alpha & \alpha^2 & 0 \end{bmatrix} \\
& c_3 \begin{bmatrix} 0 & 0 & 1 & \alpha & \alpha^2 \end{bmatrix}
\end{align*}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{hamming_code_graph.png}
\caption{Parity check matrix and Tanner graph for the length 5, GF(4) Hamming code}
\end{figure}

\section{1.2 LDPC Codes}
LDPC codes were first introduced by Gallager in 1963 [2]. The construction of LDPC codes was then generalized by Tanner who introduced the bipartite graph for the graphical representation of these codes.
LDPC codes are linear codes that have a sparse parity-check matrix [3]. These codes, like any linear code, can be denoted by their corresponding Tanner graphs. In the Tanner graph, the \( n \) columns of the parity-check matrix correspond to the bit nodes and the \( n-k \) rows of the parity-check matrix to the check nodes. There is an edge between variable node \( j \) and check node \( i \) if \( h_{ij} \neq 0 \) and this edge is related with the constant \( h_{ij} \).

In the case of binary LDPC codes, that is LDPC codes over GF(2), all nonzero elements of parity-check matrix are 1, so the edge does not have weights. Whereas for a Non-Binary LDPC code (NB-LDPC) defined over GF(\( q \)), the non zero elements of PCM are selected from the set of non-zero values in GF(\( q \)). The number of edges connected to each node in the Tanner graph is defined as the degree of the node.

**Definition 2.** A \((d_v,d_c)\)-regular LDPC code [4] is an LDPC code such that each check node has degree \( d_c \) and each bit node has degree \( d_v \). Similarly, we say that the parity-check matrix of a regular LDPC code has all column-weights equal to \( d_v \) and all row-weights equal to \( d_c \).

In contrast, an irregular LDPC code [5] has variable and check nodes of different degrees.

Nonbinary LDPC codes were first investigated by Davey and Mackay in 1998 [6]. It is shown that non-binary LDPC codes constructed over higher order Galois fields may obtain superior performance than their binary counterparts [6]. Since then, a lot of work has been devoted to the construction of various NB LDPC codes [5] and development of LDPC low complexity decoders. Contributing to the study of non-binary LDPC codes, our work - [7] - is focused on the optimization of NB-LDPC codes over several channels.

### 1.3 Construction of Structured LDPC codes

In this section, we introduced the constructions of non-binary array-based codes and non-binary quasi-cyclic codes.
Array-based LDPC codes are a class of quasi-cyclic LDPC codes, which were first introduced in [8]. These codes are parameterized by their blocklength $n$ and column weight $c$ such that $c \leq p = \sqrt{n}$, where $p$ is a prime number. Let $\sigma$ be the $p \times p$ circulant permutation matrix obtained by cyclically shifting the rows of the identity matrix $I$ to the right by 1 position.

$$\sigma = \begin{bmatrix} 0 & 0 & \ldots & 0 & 1 \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \end{bmatrix}$$ (1.3)

The parity check matrix $H(c,p)$ of an $(n, n - c \sqrt{n} + c - 1)$ array-based LDPC code is structured as shown in 1.4.

$$H(c, p) = \begin{bmatrix} I & I & I & \ldots & I \\ I & \sigma & \sigma^2 & \ldots & \sigma^{(p-1)} \\ I & \sigma^2 & \sigma^4 & \ldots & \sigma^{2(p-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & \sigma^{(c-1)} & \sigma^{2(c-1)} & \ldots & \sigma^{(c-1)(p-1)} \end{bmatrix}$$ (1.4)

To obtain an array-based NBLDPC code from its binary counterpart, we replace 1’s in the PCM with the non-zero elements of GF($q$). For an array based code, an unconstrained replacement [9] is performed, where each non-zero element is independently and uniformly generated.

Due to their implementation-friendly structure and superior performance, non-binary quasi-cyclic (NB-QC) LDPC codes [10] are well-suited for emerging data storage applications requiring very low error rates. The code construction of a NB-QC LDPC code is different from the array based construction such that a constrained replacement [11]
is performed. In the constrained replacement, the \( p \) non zero elements of a circulant permutation matrix \( \sigma \) are the same (chosen from \( \text{GF}(q) \)). Since there are \( q - 1 \) nonzero elements in \( \text{GF}(q) \) and there are exactly \( q - 1 \) different circulant permutation matrices over \( \text{GF}(2) \) of size \( (q-1) \times (q-1) \), there is a one-to-one correspondence between a nonzero element of \( \text{GF}(q) \) and a circulant permutation matrix of size \( (q - 1) \times (q - 1) \).

### 1.4 Error Floor in LDPC codes

LDPC codes are a class of capacity approaching codes when decoded with low capacity iterative decoders. Despite the fact that long LDPC codes can achieve excellent performance [12] in the low and medium signal-to-noise ratio (SNR), they exhibit a sudden saturation in the bit-error rate (BER) curve for high SNR and moderate block lengths - referred to as error floor [13]. Many practical applications such as storage and high speed digital communication often require extremely low error rates, in the order of \( 10^{-15} \). Therefore the study of error floors is of practical importance.

The error floor phenomenon is attributed to certain error prone substructures in the Tanner graph. In the Binary Erasure Channel (BEC), it was found that substructures called stopping sets dominate the performance in the error floor region [13]. For memoryless channels such as Binary Symmetric Channel (BSC) or Additive White Gaussian Channel (AWGN), it has been found that near-codewords, trapping sets [13] [14] [15], absorbing sets [16] and pseudocodewords are a few structures in the Tanner graph which determine the performance in the error floor region.

**Definition 3.** [17] A stopping set \( S \) is a subset of \( V \), the set of variable nodes, such that all neighbors of the variable nodes in \( S \) are connected to \( S \) at least twice. The size of the stopping set is the cardinality of \( S \).

Although stopping sets are useful in characterizing the errors in error floor region for BEC channels, they cannot be used to determine the performance of LDPC decoder
in other types of channels. Therefore, trapping sets were introduced in [13] to capture
decoding errors under iterative decoding algorithms for various types of channels.

Consider a subgraph of the Tanner graph of a code induced by a variable nodes,
given by the set \( V \). Let \( O \) (resp., \( E \)) be the set of check nodes connected to the set \( V \) an
odd (resp., even) number of times, i.e., the set \( E \) represents the set of satisfied check nodes
and the set \( O \) represents the set of unsatisfied check nodes. Note that the variable nodes
in set \( V \) are set to a non zero value (1 for a binary code and non zero value in \( GF(q) \) for
a NB-LDPC code defined over \( GF(q) \)) and all the other variable nodes are set to 0.

**Definition 4.** [14] The set \( V \) is an \((a,b)\) trapping set if \(|O| = b\).

1.4.1 Absorbing sets

Since not all trapping sets are problematic in practical iterative decoding algorithms,
it is useful to characterize a subclass of trapping sets that is the main cause of errors
under such decoders. Absorbing sets, first introduced in [16], are combinatorial objects
in Tanner graph that are guaranteed to be stable under a bit-flipping decoder.

**Definition 5.** The set \( V \) is an \((a,b)\) absorbing set if \(|O| = b\) and if each of the \( a \)
variable nodes have stricty more neighbors in \( E \) than in \( O \).

**Definition 6.** [14] An elementary absorbing set (resp., trapping set) is an absorbing
set (trapping set) with each of its neighboring satisfied check nodes having two edges
connected to the absorbing set (trapping set), and each of its neighbouring unsatisfied
checks having exactly one edge connected to the absorbing set (trapping set).

**Remark 1.** Since absorbing sets are a subclass of trapping sets, every absorbing set
is a trapping set, however the converse is not true.
1.5 Previous Work

It is well known that technology scaling makes flash memory cells subject to increasingly severe noise and distortion, which can largely degrade the NAND flash memory storage reliability and performance [18]. Therefore, increasingly powerful and sophisticated error correction and signal processing capabilities become indispensable. Due to their superior error correction capability and recent success in commercial hard disk drives, low-density paritycheck (LDPC) codes have attracted much attention and are seriously considered as the choice for the future storage applications.

While the operational asymmetry in Flash memories is well documented [19], the norm is still to directly apply LDPC codes, previously optimized for AWGN-like channels [20] [21] [22]. Authors of [23] proposed a framework to optimize the NB-LDPC code over the AWGN channel by the process of edge weight manipulation.

We note that previous work [15] [16] [23] studied less refined objects: an \((a, b)\) non-binary absorbing set (NB AS), with \(a\) variable nodes, \(b\) unsatisfied check nodes, and with each variable node having more satisfied than unsatisfied check nodes, but wherein no explicit classification with respect to satisfied/unsatisfied check nodes of different degrees was made (elementary ASs). Since the focus was on the AWGN channel, the techniques proposed by them focused on the removal of elementary ASs. This in itself was not an issue for symmetric channels, and in fact techniques focusing on the removal of elementary absorbing sets were demonstrated to be very effective [23].

The authors in [24] studied a subclass of ASs called the balanced AS (BAS) which were found to be the dominant object over the Partial Response (PR) channel [25]. Further, they were the first to introduce the presence of AS that were not elementary, and laid light to the fact that optimization techniques previously developed for the AWGN channel was not effective. They built over the framework proposed in [23] to remove detrimental objects from the error floor of the NB-LDPC code.
1.6 Contribution

In this work we re-visit the existing combinatorial definitions, such as absorbing sets (ASs) and elementary absorbing sets, that were proven to be useful in the error floor analysis of non-binary LDPC (NB-LDPC) codes over AWGN channels [16] [23]. By recognizing that the existing definitions are insufficient to describe the errors for asymmetric channels, we introduce a more finely specified combinatorial object: the general absorbing set (GAS). Additionally, we introduce an important subclass of GASs, which we call general absorbing sets of type two (GASTs). Our NB-LDPC code optimization objective then becomes the provable removal of GASTs. Through a succinct matrix-theoretic representation, we decompose a GAST into a set of submatrices, which we call weight consistency matrices (WCMs). By forcing the null spaces of the resultant WCMs to have a particular property, we provably remove detrimental GASTs from the graph representation of the code. Our approach systematically manipulates the edge weights in the graph representation of a non-binary code while maintaining all desirable structural properties, and can be applied to a wide variety of NB-LDPC codes. Most importantly, it offers the first theoretical framework for the design and analysis of NB-LDPC codes over realistic memory channels with asymmetry; as a case study, we demonstrate that on practical Flash channels [19] [27]. Our optimized codes outperform uninformed codes by more than 1 order of magnitude in uncorrectable bit error rate (UBER).

The rest of the thesis is organized as follows. In Chapter 2 we provide a motivation and define a new, more general class of absorbing sets called the general absorbing sets (GAS). In Chapter 3, we study the combinatorial properties of GAST and propose a technique to remove them from the Tanner graph. In Chapter 4, we delve deeper into the implementation of the GAST removal framework. Chapter 5 contains the experimental results obtained by applying our framework on two practical Flash channels, followed by the conclusion in Chapter 6.
Chapter 2

General Absorbing Set

2.1 Motivation

We recall that an elementary AS is an AS with all satisfied check nodes having degree 2 and all unsatisfied check nodes having degree 1. The elementary ASs were found to be the problematic objects [23] in symmetric channels such as AWGN channel. However in asymmetric channels, configurations that are not necessarily elementary can also be problematic. For example, consider two variable nodes with a shared check node. Suppose that, due to asymmetry, the two variable nodes both have excessively wrong values. The shared check node will then be unable to overcome these incorrect values, resulting in an AS with a degree 2 unsatisfied check node (non-elementary AS). Therefore grouping the objects of interest by using the parameters $a$ and $b$ may not be sufficient to make a distinction between the elementary and non-elementary objects. The below example further demonstrates the need for a new, more generalized definition of an absorbing set.

Example 2. Consider the three non-binary absorbing sets shown in the Figure 2.1. Using the $(a, b)$ definition of AS, we see that all the three structures are $(6,2)$ AS, however it is clear that they possess different connectivity properties. The first one is an elementary AS while the other two are not and are in turn different from each other. This motivates us to introduce a new definition for AS.
2.2 General Absorbing Set

**Definition 7.** Consider a subset $V$ of variable nodes in the Tanner graph of the NB-LDPC code. Set all the variable nodes in $V$ to values $\in GF(q) \setminus 0$ and set all other variable nodes to 0. The set $V$ is said to be an $(a, b, b_2, d_1, d_2, d_3)$ general absorbing set (GAS) over $GF(q)$ if $a$ is the size of $V$, $b$ (resp., $b_2$) is the number of unsatisfied (resp., degree 2 unsatisfied) check nodes connected to $V$, $d_1$ (resp., $d_2$ and $d_3$) is the number of degree 1 (resp., 2 and $> 2$) check nodes connected to $V$, and each variable node in $V$ is connected to strictly more satisfied than unsatisfied neighboring check nodes.

An elementary absorbing set, previously studied [23], contains only the degree 1 and degree 2 check nodes, of which, the degree 1 check nodes are unsatisfied and the degree 2 check nodes are always satisfied. No explicit classification with respect to the degree or with respect to satisfied/unsatisfied check nodes was made. The GAS is a more generalized object where the satisfied check nodes are always of degree 2 or higher and the unsatisfied check nodes are of degree 1 or higher ($b \geq d_1 + b_2$).

The GAS definition introduced above makes an explicit distinction between the three configurations in Example 2. The three configurations are $(6,2,0,2,8,0)$, $(6,2,2,0,9,0)$ and $(6,2,0,2,5,2)$ respectively.

The description of a GAS depends on having appropriate non-binary values (labels)
associated with its edge weights. It is useful to view the induced subgraph in terms of its unlabeled variant. Thus, we define the following graph-theoretic object.

**Definition 8.** Let $\mathcal{V}$ be a subset of variable nodes in the Tanner graph of a code. Let $\mathcal{O}$ (resp., $\mathcal{T}$ and $\mathcal{H}$) be the set of degree 1 (resp., 2 and $> 2$) check nodes connected to variable nodes in $\mathcal{V}$. This graphical configuration is an $(a, d_1, d_2, d_3)$ unlabeled GAS (UGAS) if it satisfies the following two conditions:

- $|\mathcal{O}| = d_1$, $|\mathcal{T}| = d_2$, and $|\mathcal{H}| = d_3$.
- Each variable node in the UGAS is connected to more neighbors $\in \{\mathcal{T} \cup \mathcal{H}\}$ than $\in \mathcal{O}$.

Note that the UGAS is a purely topological object. We can drop the parameters $b$ and $b_2$, since they are no longer relevant. Further, only the degree 1 check nodes are guaranteed to be unsatisfied irrespective of the edge weights. This explains the second condition of Definition 8 - for the unlabelled configuration to be an absorbing set, the number of degree 1 check nodes should be smaller than the number of check nodes of degree 2 and degree $>2$ combined.

### 2.3 Matrix-Theoretic representation of GAS

Let $\mathbf{H}$ denote the parity check matrix of an NB-LDPC code defined over GF($q$). Consider an $(a, b, b_2, d_1, d_2, d_3)$ GAS in the Tanner graph of this code. Let $\mathbf{A}$ be the $\ell \times a$ submatrix of $\mathbf{H}$ that consists of $\ell = d_1 + d_2 + d_3$ rows of $\mathbf{H}$, corresponding to the check nodes participating in this GAS, and $a$ columns, corresponding to the variable nodes participating in this GAS.

An $(a, b, b_2, d_1, d_2, d_3)$ GAS must satisfy:

- **Topological conditions:** Its unlabeled configuration must satisfy the UGAS conditions stated in Definition 8.
• **Weight conditions:** The set is an \((a, b, b_2, d_1, d_2, d_3)\) GAS over \(\text{GF}(q)\) if and only if there exists an \((\ell - b) \times a\) submatrix \(W\) of column rank \(r_W < a\), with elements \(w_{ij}, 1 \leq i \leq (\ell - b), 1 \leq j \leq a\), of the matrix \(A\), that satisfies the following two conditions:

1. Let \(\mathcal{N}(W)\) be the null space of the submatrix \(W\), and let \(d^T_k, 1 \leq k \leq b\), be the \(k\)th row of the matrix \(D\) obtained by excluding \(W\) from \(A\). Then, 
   \[
   \exists \ v = [v_1 v_2 ... v_a]^T \in \mathcal{N}(W) \text{ s.t. } v_j \neq 0 \ \forall j
   \in \{1, 2, \ldots, a\} \text{ and } d^T_k v \neq 0 \ \forall k \in \{1, 2, \ldots, b\},
   \]
   i.e., \(Av = \begin{bmatrix} W_{(\ell-b)\times a} \\ D_{b\times a} \end{bmatrix} v = \begin{bmatrix} 0_{(\ell-b)\times 1} \\ \neq 0 \end{bmatrix} \). \hfill (2.1)

2. Let \(d_{kj}, 1 \leq k \leq b, 1 \leq j \leq a\), be the elements of the matrix \(D\). Then, 
   \[
   \forall j \in \{1, 2, \ldots, a\}, \quad \left( \sum_{i=1}^{\ell-b} F(w_{ij}) \right) > \left( \sum_{k=1}^{b} F(d_{kj}) \right), \hfill (2.2)
   \]
   where \(F(y) = 0\) if \(y = 0\), and \(F(y) = 1\) otherwise.

Computations are performed over \(\text{GF}(q)\).

Here, \(W\) is the submatrix of satisfied check nodes and \(D\) is the submatrix of unsatisfied check nodes. For the object to be an absorbing set, the number of satisfied check nodes connected to it should be greater than the number of unsatisfied check nodes, which is explained by the second condition.

### 2.4 General Absorbing Set of Type Two

This work is focused on a subset of GAS, defined as follows:

**Definition 9.** A GAS that has \(d_2 > d_3\) and all the unsatisfied check nodes connected to it (if any) \(\in \{O \cup T\}\) (having either degree 1 or degree 2), is defined as a \((a, b, d_1, d_2, d_3)\) general absorbing set of type two (GAST).
The word ”two” refers to the maximum degree of any unsatisfied check node connected to the set. Similar to the UGAS definition (Definition 8), we call the unlabeled GAST configuration **UGAST**.

**Remark 2.** The focus on GAST over GAS is justified because:

- The existence of an unsatisfied check node of degree $> 2$ in the configuration significantly increases the likelihood that the object is not an absorbing set.

- Our simulation results over several channels has shown that the percentage of ASs containing an unsatisfied check node of $> 2$ is less than one percent.

- GASTs are more general than any previously introduced type of ASs (e.g., Elementary ASs, Balanced ASs [24]), as can be seen in Figure 2.2

![Figure 2.2: Venn diagram showing different classes of ASs](image)

**Remark 3.** Assuming that the degree of any unsatisfied check node is $\leq 2$, different types of known ASs become special cases of GASTs. We summarize this in the following lines:

- A GAST with $b = d_1$ and $d_3 = 0$ is an elementary AS.

- A GAST with $b > d_1$ or $d_3 > 0$ is a non-elementary AS.

- A GAST with $0 \leq b \leq \left\lfloor \frac{np}{2} \right\rfloor$ is a balanced AS (BAS).
A GAST with $\lceil \frac{n g}{2} \rceil < b \leq a g$ is an unbalanced AS (UBAS), where $g = \lfloor \frac{2^{-1}}{2} \rfloor$, and $\gamma$ is the column weight.

Balanced and unbalanced AS were previously introduced in [24]. In particular, balanced AS play a critical role in the context of channels with memory such as those encountered in magnetic recording applications.
Chapter 3

Analysis of GAST

In Chapter 2, we have seen that focussing on elementary AS is not sufficient and code optimization techniques focusing on elementary ASs are ineffective as they are agnostic to a finer classification of the important configurations. In this chapter we establish a series of properties of GASTs, using which we can remove the problematic structures.

3.1 Combinatorial properties of GAST

The following theorem provides a condition for when a degree 2 check node can be unsatisfied in a GAST configuration after proper edge weight labelling of a UGAST, operating on the same set of variable nodes and check nodes. By an edge weight labeling we mean an assignment of edge weights from the set $\mathbb{GF}(q) \setminus \{0\}$ for a GAST associated with a code defined over $\mathbb{GF}(q)$.

Theorem 1. Consider an $(a, d_1, d_2, d_3)$ UGAST with $\mathcal{T}$ denoting the set of $d_2$ degree 2 check nodes and with $\mathcal{H}$ denoting the set of $d_3$ check nodes of degree $\geq 2$ in this configuration. This UGAST can result in an $(a, b, d_1, d_2, d_3)$ GAST (with proper edge weight labeling) with $b > d_1$ if and only if there exists a check node $c$ in $\mathcal{T}$ such that the two neighbors of $c$ (with respect to this UGAST) each have the property that at least $\lceil \frac{\gamma + 1}{2} \rceil$ of their neighboring check nodes belong to $\{\mathcal{T} \cup \mathcal{H}\}$. 

15
Proof. We prove Theorem 1 by contradiction. We assume that there exists no check nodes $\in \mathcal{T}$ connecting pairs of variable nodes that have strictly $> \lceil \frac{\gamma+1}{2} \rceil$ check nodes $\in \{\mathcal{T} \cup \mathcal{H}\}$ connected to each of them. This means, for each degree 2 check node, at least one of the connected variable nodes has exactly $\lceil \frac{\gamma+1}{2} \rceil$ connected check nodes $\in \{\mathcal{T} \cup \mathcal{H}\}$. Note that the number of satisfied check nodes connected to any variable node in any AS cannot be $< \lceil \frac{\gamma+1}{2} \rceil$. Also note that these check nodes $\in \{\mathcal{T} \cup \mathcal{H}\}$ are the only check nodes that can be satisfied because the others $\in \mathcal{O}$ are always unsatisfied.

As a result, if any degree 2 check node is forced to be unsatisfied in the GAST (because of the edge weights), at least one variable node of the two will have the following bound on the number of satisfied check nodes connected to it.

$$s_{vn} \leq \left\lfloor \frac{\gamma+1}{2} \right\rfloor - 1 \leq \left\lfloor \frac{\gamma-1}{2} \right\rfloor. \quad (3.1)$$

Thus, and since an AS condition is to have $\geq \lceil \frac{\gamma+1}{2} \rceil$ satisfied check nodes connected to each variable node in the configuration, our configuration under analysis will not become an AS if any degree 2 check node is unsatisfied. In other words, given our assumption, all the unsatisfied check nodes must be $\in \mathcal{O}$ ($\notin \mathcal{T}$), like elementary ASs, in order that the object is an AS. This contradiction proves Theorem 1.

Remark 4. The importance of Theorem 1 is that it provides the necessary topological condition on a GAST to have unsatisfied check nodes of degree 2. If this condition is not satisfied, then all the unsatisfied check nodes of this GAST are of degree 1 (elementary ASs), which makes the removal process of the GAST much easier as we shall see later.

Theorem 2: Given an $(a, d_1, d_2, d_3)$ UGAST, the maximum number of unsatisfied check nodes, $b_{max}$, in the resulting GAST after edge weight labeling is upper bounded by:

$$b_{max} \leq d_1 + b_{ut}, \quad \text{where}$$

$$(3.2)$$

$$b_{ut} = \frac{1}{2} \left( a \left\lfloor \frac{\gamma-1}{2} \right\rfloor - d_1 \right). \quad (3.3)$$
Proof. Since degree 1 check nodes are always unsatisfied, we can write the bound on $b_{\text{max}}$ as follows:

$$b_{\text{max}} \leq d_1 + b_{\text{ut}},$$

where $b_{\text{ut}}$ is the upper bound on the maximum number of degree 2 unsatisfied check nodes ($\in \mathcal{T}$). In the beginning, we assume that all the degree 2 check nodes are satisfied.

To compute $b_{\text{ut}}$, we span the variable nodes in the GAST configuration one by one to make the maximum number of connected degree 2 check nodes unsatisfied. Similar to any AS, the maximum number of unsatisfied check nodes connected to a variable node in the GAST is $\left\lfloor \frac{\gamma - 1}{2} \right\rfloor$. Thus, the maximum number of additional check nodes connected to variable node $i$, $1 \leq i \leq a$, that can be made unsatisfied is $b_{\text{ut},i} = \left\lfloor \frac{\gamma - 1}{2} \right\rfloor - b_{\text{up},i}$, where $b_{\text{up},i}$ is the number of already-unsatisfied check nodes connected to variable node $i$ updated by what has been done for all the variable nodes processed prior to $i$.

For some $a$, $d_1$, and $\gamma$, the upper bound $b_{\text{ut}}$ is achieved if for each variable node that has $> \left\lceil \frac{\gamma + 1}{2} \right\rceil$ connected check nodes $\in \{\mathcal{T} \cup \mathcal{H}\}$ there exists another variable node connected to it, through some degree 2 check node, which satisfies the same condition. In other words, the degree 2 check node connecting these two variable nodes can be made unsatisfied while the object remains an AS. Thus,

$$b_{\text{ut}} = \sum_{i=1}^{a} b_{\text{ut},i} = \sum_{i=1}^{a} \left[ \left\lfloor \frac{\gamma - 1}{2} \right\rfloor - b_{\text{up},i} \right]$$

$$= a \left\lfloor \frac{\gamma - 1}{2} \right\rfloor - \sum_{i=1}^{a} b_{\text{up},i}. \quad (3.4)$$

Since $\sum_{i=1}^{a} b_{\text{up},i}$ represents the final number of unsatisfied check nodes we will end up with after spanning all the variable nodes in the GAST, it can be concluded that:

$$\sum_{i=1}^{a} b_{\text{up},i} = d_1 + b_{\text{ut}}. \quad (3.5)$$

Substituting from (3.5) into (3.4) results in a recursive equation where $b_{\text{ut}}$ appears in both the RHS and the LHS. The solution of this equation is (3.3), which completes the proof.
Figure 3.1: A $(4,4,4,6,0)$ GAST, $\gamma = 4$, (left) vs a $(6,0,9,0)$ GAST, $\gamma = 3$, (right). ULGASTs are reached by setting all the weights to 1.

**Example 3.** Consider the $(4,4,6,0)$ UGAST ($\gamma = 4$) shown in Fig. 3.1, left panel. From Theorem 1, irrespective of the edge weight labeling that converts this UGAST to a GAST, the resultant GAST cannot have unsatisfied check nodes with degree 2 (Else it is not an AS). From (3.3), $b_{at} = 0$ and thus, from (3.2), $b_{max} = b = d_1 = 4$. Hence, this GAST can only be an elementary AS.

In contrast, the $(6,0,9,0)$ UGAST ($\gamma = 3$), shown in the right panel of Fig. 3.1, has the following property. When this UGAST is converted to a GAST for some GF($q$), it is possible to have degree 2 unsatisfied checks node (from Theorem 1). Also from (3.3), $b_{at} = 3$, and thus, $b_{max} = b_{at} = 3 \neq d_1 = 0$.

### 3.2 Removal of GAST

We now propose a technique, using Theorems 1 and 2, to remove a GAST from the Tanner graph of an LDPC code. The objective is to remove a GAST such that it does not result in another GAST for the same UGAST configuration after the edge weight labelling.

For a given $(a,b,d_1,d_2,d_3)$ GAST, let $Z$ be the set of all $(a,b',d_1,d_2,d_3)$ GASTs with
$d_1 \leq b' \leq b_{\text{max}}$ which have the same UGAST configuration as the original $(a, b, d_1, d_2, d_3)$ GAST. Here, $b_{\text{max}}$ is the largest allowable number of unsatisfied check nodes for these configurations (This value can be computed using Theorem 2).

**Definition 10.** An $(a, b, d_1, d_2, d_3)$ GAST is said to be removed from the Tanner graph of an NB-LDPC code if and only if the resulting object (after edge weight processing) $\not\in \mathcal{Z}$.

Consider the two configurations mentioned in Example 3. For the $(4, 4, 4, 6, 0)$ GAST ($\gamma = 4$), $b_{\text{max}} = b = d_1 = 4$, which means $\mathcal{Z}$ contains only one object that is the $(4, 4, 4, 6, 0)$ GAST itself. To remove this GAST, it would be sufficient to increase the number of unsatisfied check nodes by 1 i.e., $(4, 5, 4, 6, 0)$.

On the other hand, for the $(6, 0, 0, 9, 0)$ GAST ($\gamma = 3$), $b_{\text{max}} = b_{ul} = 3$, which means $\mathcal{Z} = \{(6, 0, 0, 9, 0), (6, 1, 0, 9, 0), (6, 2, 0, 9, 0), (6, 3, 0, 9, 0)\}$. Therefore to remove the $(6, 0, 0, 9, 0)$ GAST, it is not sufficient to convert it to $(6, 1, 0, 9, 0)$ as it is still a problematic configuration.

### 3.3 Weight Consistency Matrix

For a given GAST, define a matrix $\mathbf{W}^z$ to be the matrix obtained by removing $b'$, $d_1 \leq b' \leq b_{\text{max}}$, rows corresponding to check nodes $\in \{\mathcal{O} \cup \mathcal{T}\}$ from the matrix $\mathbf{A}$. These $b'$ check nodes can simultaneously be unsatisfied under some edge weight labeling that produces another GAST which has the same UGAST as the given GAST. Let $\mathcal{U}$ be the set of all matrices $\mathbf{W}^z$. Each element $\in \mathcal{Z}$ has one or more matrices $\in \mathcal{U}$. In principle GASTs can be removed by manipulating the associated matrices $\mathbf{W}^z$ of the set $\mathcal{U}$. However, a more efficient approach is to work with matrices that are each a submatrix of multiple $\mathbf{W}^z$ matrices. In this way, we can remove problematic GASTs while only focusing on a smaller collection of matrices. These new matrices are referred to as the weight consistency matrices (WCMs).
Definition 11. For a given \((a, b, d_1, d_2, d_3)\) GAST and its associated adjacency matrix \(A\) and its associated set \(Z\), we construct a set of \(t\) matrices as follows:

1. Each \(W_{cm}^h\), \(1 \leq h \leq t\), in this set is an \((\ell - b_h^{cm}) \times a\) submatrix, \(d_1 \leq b_h^{cm} \leq b_{\text{max}}\), formed by removing different \(b_h^{cm}\) rows corresponding to check nodes \(\in \{O \cup T\}\) from the \(\ell \times a\) matrix \(A\) of the GAST. These \(b_h^{cm}\) check nodes can simultaneously be unsatisfied under some edge weight labeling that produces an absorbing set which has the same UGAST as the given GAST.

2. Each matrix \(W^z \in \mathcal{U}\), for every element \(\in Z\), contains at least one element of the resultant set as its submatrix.

3. The cardinality \(t\) of this set is the smallest cardinality of all sets satisfying conditions 1 and 2.

We refer to the matrices in this set as weight consistency matrices (WCMs) and to this set as \(\mathcal{W}\).

The below theorem makes use of the WCMs to remove the GAST

**Theorem 3.** The necessary and sufficient processing needed to remove an \((a, b, d_1, d_2, d_3)\) GAST, according to Definition 10, is to change the edge weights such that for every WCM \(W_{cm}^h\), there does not exist any vector with all its entries \(\neq 0\) in the null space of that WCM. Mathematically:

\[
\text{If } N(W_{cm}^h) = \text{span}\{x_1, x_2, \ldots, x_k\}, \text{ then } \\
\# r = [r_1 \ r_2 \ \ldots \ r_k]^T, \ v = r_1x_1 + r_2x_2 + \cdots + r_kx_k \\
= [v_1 \ v_2 \ \ldots \ v_a]^T \text{ s.t. } v_j \neq 0 \ \forall j \in \{1, 2, \ldots, a\}. \quad (3.6)
\]

Computations are performed over GF\((q)\).

**Proof.** We divide our proof into two parts. First, we prove that breaking the weight conditions stated in (2.1) and (2.2) for any submatrix \(W^z\) can only be done as stated in
The weight conditions are broken if:

\[ \exists \mathbf{v} = [v_1 \ v_2 \ \ldots \ v_a]^T \in \mathcal{N}(\mathbf{W}_z) \text{ s.t. } v_j \neq 0 \quad \forall j \in \{1, 2, \ldots, a\}. \tag{3.7} \]

Since the set of vectors \( \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\} \) is the basis of \( \mathcal{N}(\mathbf{W}_z) \), then if there is no linear combination of them over GF\( (q) \) that results in \( \mathbf{v}, v_j \neq 0 \quad \forall j \in \{1, 2, \ldots, a\} \), (3.7) is automatically satisfied. In other words, if (3.6) is satisfied, the weight conditions of \( \mathbf{W}_z \) are broken.

Second, we prove that breaking such weight conditions for all the WCMs (the smallest AS submatrices \( \in \mathcal{W} \)), guarantees the GAST removal. By the definition of WCMs (Definition 11), \( \exists \mathbf{W}_{cm}^h \in \mathcal{W} \), which is a submatrix of any matrix \( \mathbf{W}_z \in \mathcal{U} \) of size \((\ell - b') \times a, d_1 \leq b' < b_{\text{max}}\).

Now, recall the following linear algebraic lemma:

If we have a matrix \( \mathbf{F} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix} \),

then, \( \mathcal{N}(\mathbf{F}) \subseteq \mathcal{N}(\mathbf{F}_1) \) and \( \mathcal{N}(\mathbf{F}) \subseteq \mathcal{N}(\mathbf{F}_2) \). \tag{3.8}

Applying this lemma to our case, if \( \exists \mathbf{v} = [v_1 \ v_2 \ \ldots \ v_a]^T \in \mathcal{N}(\mathbf{W}_{cm}^h) \text{ s.t. } v_j \neq 0 \quad \forall j \), for every WCM \( \in \mathcal{W} \), this implies that \( \exists \mathbf{v} = [v_1 \ v_2 \ \ldots \ v_a]^T \in \mathcal{N}(\mathbf{W}_z) \text{ s.t. } v_j \neq 0 \quad \forall j \), for any matrix \( \mathbf{W}_z \) of size \((\ell - b') \times a, d_1 \leq b' \leq b_{\text{max}}\). This implies the removal of the GAST according to Definition 10, which completes the proof of Theorem 3.

**Remark 5.** The concepts proposed by Theorem 3 are not only useful for GAST removal, but also for GAST detection. In other words, using Theorem 3, we can detect if a certain configuration can be an \((a, b', d_1, d_2, d_3)\) GAST, \(d_1 \leq b' \leq b_{\text{max}}\), by checking the null spaces of all the WCMs \( \in \mathcal{W} \).
Chapter 4

WCM Framework

We deploy all the illustrated definitions and theorems to develop the new NB-LDPC code optimization framework. The objective of the new framework is to remove GASTs from the Tanner graph of the NB-LDPC code using their WCMs and via minimum number of edge weight manipulations.

4.1 Algorithm to compute WCMs

As mentioned in Chapter 3, the GAST removal is performed in two steps. The first stage is the WCM extraction which is explained in Algorithm 1. Algorithm 1 operates on the UGAST to obtain the WCMs.

**Algorithm 1** Finding the WCMs for a given GAST

1: **Input**: Tanner graph of the GAST $G'$ with edge weights over GF($q$), from which, the matrix $A$ is formed.

2: Set the maximum number of nested for loops, loop_max.

3: Mark all the check nodes $\in \{T \cup H\}$ as satisfied.

4: Check if $\exists$ at least one degree 2 check node connecting two variable nodes, each is connected to $> \left\lceil \frac{n+1}{2} \right\rceil$ check nodes that are marked as satisfied.

5: if $\nexists$ any of them then
\[ \exists \text{ only one } (\ell - d_1) \times a \text{ WCM. Extract it by removing all the rows of degree 1 check nodes from the matrix } A. \]

7: Go to 27.

8: else

9: Count such check nodes (that satisfy the condition in 4), save the number in \( u^0 \), and save their indices in \( y^0 = [y^0(1) \; y^0(2) \; \ldots \; y^0(u^0)]^T \).

10: end if

11: Compute \( b_{ut} \) from (3.3) in Theorem 2. If \( b_{ut} = 1 \), go to 26.

12: for \( i_1 \in \{1, 2, \ldots, u^0\} \) do

13: Mark all the check nodes \( \in \{T \cup H\} \) as satisfied.

14: Mark the selected check node \( c_{y^0(i_1)} \) as unsatisfied.

15: Redo 9 but save in \( u^1_{i_1} (< u^0) \) and \( y^1_{i_1} \).

16: If \( b_{ut} = 2 \), go to 12.

17: for \( i_2 \in \{1, 2, \ldots, u^1_{i_1}\} \) do

18: Mark the selected check node \( c_{y^1_{i_1}(i_2)} \) as unsatisfied.

19: Redo 9 but save in \( u^2_{i_1,i_2} (< u^1_{i_1}) \) and \( y^2_{i_1,i_2} \).

20: If \( b_{ut} = 3 \), go to 17.

21: ...

22: The lines from 17 to 20 are properly repeated as many times as \( \text{loop}_\text{max} - 2 \).

23: ...

24: end for

25: end for

26: Obtain the WCMs via the indices in the \( y \) arrays. In particular, by removing permutations of the rows corresponding to \( c_{y^0(i_1)}, c_{y^1_{i_1}(i_2)}, \ldots, c_{y^{b_{ut}-1}_{i_1,i_2}} \), and the degree 1 check nodes from \( A \), we can reach any WCM.

27: Eliminate any repeated WCM to reach the final set of WCMs, \( W \), where \( |W| = t \).

28: Output: The set \( W \) of all WCMs of the GAST.
Note that $b_{ul}$ is an upper bound for the maximum number of degree 2 check nodes that can be removed. Thus, it can happen in some cases that $u_{i_1i_2\cdots i_{b_{ul}}-1} = 0 \forall i_1, i_2, \ldots, i_{b_{ul}}$. In such cases, the exact maximum number of degree 2 check nodes that can be removed is the number of levels (nested loops in Algorithm 1) $b_{el}$ after which $u_{i_1i_2\cdots i_{b_{el}}} = 0 \forall i_1, i_2, \ldots, i_{b_{el}}$. Note also that because WCMs are not necessarily of the same length, Algorithm 1 may stop before $b_{el}$ levels starting from some $c_{\rho'_{(i_1)}}$, which results in an $(\ell - b_{cm}) \times a$ WCM with $b_{cm} < b_{\text{max}} = d_1 + b_{el}$.

To illustrate Algorithm 1, we contrast the two configurations in Fig.3.1 one more time. The set $W$ contains only one WCM of size 6 × 4 for the $(4, 4, 4, 6, 0)$ GAST ($\gamma = 4$) (see steps 4, 5, 6, and 7 of Algorithm 1). Having a single WCM is the case for all GASTs that cannot have unsatisfied check nodes $\in T$ (which exemplifies the ease in removing them).

On the contrary, for the $(6, 0, 0, 9, 0)$ GAST ($\gamma = 3$), following Algorithm 1 gives $u^0 = 9$ (i.e., all the check nodes are connecting pairs satisfying the condition in Theorem 1), $y^0 = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9]^T$, and $b_{\text{max}} = b_{ul} = b_{el} = 3$. The matrix $A$ is:

$$A = \begin{bmatrix}
    c_1 & w_{11} & w_{12} & 0 & 0 & 0 & 0 \\
    c_2 & 0 & w_{22} & w_{23} & 0 & 0 & 0 \\
    c_3 & 0 & 0 & w_{33} & w_{34} & 0 & 0 \\
    c_4 & 0 & 0 & 0 & w_{44} & w_{45} & 0 \\
    c_5 & 0 & 0 & 0 & 0 & w_{55} & w_{56} \\
    c_6 & w_{61} & 0 & 0 & 0 & 0 & w_{66} \\
    c_7 & 0 & w_{72} & 0 & 0 & w_{75} & 0 \\
    c_8 & w_{81} & 0 & 0 & w_{84} & 0 & 0 \\
    c_9 & 0 & 0 & w_{93} & 0 & 0 & w_{96}
\end{bmatrix}$$

If Algorithm 1 selects $c_1$ first, it will be marked as unsatisfied. As a result, $c_2$, $c_7$, $c_6$, and $c_8$ cannot be selected with $c_1$ (otherwise the configuration will not be an AS). On level 2, the algorithm can select one of $c_3$, $c_4$, $c_5$, and $c_9$. If the algorithm selects
$c_3$, it will be marked as unsatisfied. As a result, $c_4$ and $c_9$ cannot be selected. The only choice remaining on level 3 is then $c_5$, which means one WCM is extracted by removing $(c_1, c_3, c_5)$ together from $A$. Applying the steps from 12 to 25 on all the 9 check nodes in $y^0$, results in that the set $W$ contains 6 WCMs of size $6 \times 6$. They are formed by removing the following groups of 3 rows from $A$: \{(c_1, c_3, c_5), (c_1, c_4, c_9), (c_2, c_4, c_6), (c_2, c_5, c_8), (c_3, c_6, c_7), (c_7, c_8, c_9)\}.

### 4.2 Size of WCMs

The size of the WCMs $\in W$ depends on the UGAST configuration. If the UGAST configuration is symmetric, then all the WCMs $\in W$ are of the same size. On the other hand if the UGAST configuration is asymmetric, then all the WCMs are not guaranteed to be of the same size. By symmetric, we mean that for all possible selections of check nodes in $y^0$, the number of choices in the subsequent levels will be the same.

#### 4.2.1 Symmetric GAST

A symmetric GAST configuration, has all the WCMs $\in W$ of size $(l - b_{max}) \times a$.

![Figure 4.1: A (6,2,2,11,0) GAST, $\gamma = 4$; Example of a symmetric GAST](image-url)
Example 4. Consider the \((6, 2, 2, 11, 0) \text{ GAST} \gamma = 4\), shown in Figure 4.1. According to Algorithm 1, \(b_{\text{max}} = 4\), \(d_1 = 2\), \(b_{ut} = 2\), \(b_{ct} = 2\), \(u^0 = 6\) and \(y^0 = [1 4 7 8 10 11]^T\). Since \(b_{ut} = b_{ct}\), we say that the maximum bound is achievable. Suppose we choose \(c_1\) on level one and mark it as unsatisfied, the only choice we are left with is to select \(c_4\) on level 2 (choosing any other check node on level 2 is not possible since it will no longer be an AS). Similarly, when we start with any check node \(\in y^0\) on level 1, we will always have exactly one possible selection on level 2. This confirms that it is a symmetric configuration.

A total of 3 WCMs will be extracted by the algorithm. The 3 WCMs are obtained by simultaneously removing rows \((c_1, c_4)\), \((c_7, c_8)\), \((c_{10}, c_{11})\) from the adjacency matrix. Note that the degree 1 checks \((c_{12} \text{ and } c_{13})\) are removed from the adjacency matrix by default in all cases. Therefore all 3 WCMs are of size \(6 \times 9\).

Remark 6. Always for a symmetric configuration, \(b_{ut} = b_{ct}\). In other words, the upper bound (for the number of degree 2 unsatisfied check nodes) is always achieved for a symmetric configuration.

4.2.2 Asymmetric GAST

An asymmetric GAST will have WCMs of size \((l - b_t) \times a\), where \(d_1 \geq b_t \geq b_{\text{max}}\). Depending on the GAST under investigation, for certain configurations, we may have all WCMs of the same size.

Consider the \((6, 2, 2, 5, 2) \text{ GAST}, \gamma = 3\), shown in Figure 4.2.
According to Definition 10, to remove the (6, 2, 2, 5, 2), the resulting object after edge weight changes $\notin \mathbb{Z}$; $\mathbb{Z} = \{(6, 2, 2, 5, 2), (6, 3, 2, 5, 2), (6, 4, 2, 5, 2)\}$. If we trace Algorithm 1 for the (6, 2, 5, 2) UGAST, we get $u^0 = 3$, $y^0 = [1 \ 2 \ 3]^T$, and $d_1 = 2$, $b_{\text{max}} = 4$, $b_{\text{ult}} = 2$. The adjacency matrix, $A$, of the GAST is:

$$A = \begin{bmatrix}
  v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\
  c_1 & w_{11} & 0 & 0 & 0 & 0 & w_{16} \\
  c_2 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\
  c_3 & 0 & w_{32} & w_{33} & 0 & 0 & 0 \\
  c_4 & 0 & 0 & w_{43} & w_{44} & 0 & 0 \\
  c_5 & w_{51} & 0 & w_{53} & 0 & w_{55} & 0 \\
  c_6 & 0 & w_{62} & 0 & w_{64} & 0 & w_{66} \\
  c_7 & 0 & 0 & 0 & 0 & w_{75} & w_{76} \\
  c_8 & 0 & 0 & 0 & 0 & w_{85} & 0 \\
  c_9 & 0 & 0 & 0 & 0 & w_{94} & 0 \\
\end{bmatrix}$$

Figure 4.2: A (6,2,5,2) GAST, $\gamma = 3$; Example of an asymmetric GAST
Since $b_{at} = 2$, the algorithm 1 iterates for 2 levels. If the algorithm selects $c_1$ on the first level, $c_2$ cannot be selected and the only choice on level 2 is $c_3$. Therefore one WCM is extracted by removing rows $c_1$ and $c_3$ from the adjacency matrix. For the next WCM, on level 1, if the algorithm selects $c_2$ and marks it as unsatisfied, both $c_1$ and $c_3$ are not available for selection on level 2. This means that the second WCM is extracted by removing the row corresponding to $c_2$. On completion of the algorithm, the set $W$ consists of two WCMs of size $7 \times 6$ and $8 \times 6$ by removing rows $(c_1, c_3)$ and $(c_2)$ respectively.

**Remark 7.** If the WCMs of a GAST are of different sizes, then the GAST is an asymmetric configuration. However, the converse is not true.

### 4.3 NB-LDPC Code Optimization Algorithm

Once the WCMs are extracted (using Algorithm 1), we then remove the GAST by systematically manipulating the edge weights (while other structural properties of the LDPC code is not altered) according to Theorem 3. Algorithm 2 demonstrates how the GAST removal process can be achieved. It takes the unoptimized Tanner graph as the input and gives the optimized Tanner graph as the output.

**Algorithm 2** Optimizing NB-LDPC codes by reducing the number of GASTs

1: **Input:** Tanner graph $G$ of the NB-LDPC code with edge weights over GF($q$).

2: Determine $G$, the set of all GASTs to be removed.

3: Let $X$ be the set of GASTs in $G$ that cannot be removed and initialize it with $\emptyset$.

4: Let $P$ be the set of GASTs in $G$ that have been processed and initialize it with $\emptyset$.

5: Sort the GASTs in $G$ from the smallest to the largest.

6: Start from the smallest GAST (smallest index).

7: **for** $\forall$ GAST $s \in G \setminus P$ **do**

8: Double check that the unlabeled configuration of $s$ satisfies the UGAST conditions in Definitions 8 and 9.
If it does, determine the minimum number of necessary edge weight changes to remove the GAST, $E_{GAST,min}$, (using the first part of Lemma 2 in [23]).

Extract the subgraph $G'$ of the GAST $s$, from $G$.

Determine the set $W$ of all WCMs ($|W| = t$).

for $h \in \{1, 2, \ldots, t\}$ do

Find the null space $\mathcal{N}(W^cm)$ of the $h$th WCM.

if (3.6) is satisfied then

Go to 12.

else

Try to stay within $E_{GAST,min}$, by keeping track of the changes already performed in $G'$.

Perform the minimum number of edge weight changes in $G'$ to achieve (3.6) for the $h$th WCM.

If these edge weight changes undo the removal of any GAST $\in \mathcal{P} \setminus \mathcal{X}$, cancel them and go to 18.

if $\not\exists$ edge weights to execute 18 and 19 then

Add GAST $s$ to the set $\mathcal{X}$.

Go to 27.

end if.

end if.

end for.

Update $G$ by the changes performed in $G'$.

Add GAST $s$ to the set $\mathcal{P}$.

If $\mathcal{P} \neq \mathcal{G}$, go to 7 to pick the next smallest GAST.

end for
If $\mathcal{X} = \emptyset$, then all the GASTs have been removed. Else, only the remaining GASTs in $\mathcal{X}$ cannot be removed.

Output: Optimized Tanner graph $G$ of the NB-LDPC code with edge weights over $\text{GF}(q)$.

**Remark 8.** For elementary NB ASs, the unsatisfied check nodes are always $\in \mathcal{O}$, which means there exists only one $(\ell - d_1) \times a$ WCM. The problem of resolving the null space of this WCM according to Algorithm 2 reduces to breaking the weight condition stated in Lemma 1 in [23].

### 4.4 WCM Customization

As long as the maximum degree of any unsatisfied check node in the AS is 2, Algorithm 2 can be used to remove any type of ASs from the NB-LDPC code. The proposed WCM optimization framework is suitable for optimizing NB-LDPC codes to be used over many channels since we can remove objects such as elementary ASs, BAS and GAST using the WCM framework.

The WCM framework, in its current form can be used to remove elementary ASs (AWGN channel) and BASs (PR channel). However, removing elementary ASs or BASs, as restricted subclasses of GASTs, requires less steps and fewer edge weight changes compared to removing GASTs according to Definition 10. In other words, it can be enough for an elementary AS to be converted into a non-elementary AS, and for a BAS to be converted into a UBAS. In order to customize the WCM framework for removing such simpler objects (compared to GASTs), the definition of WCMs and how to find them in the matrix $\mathbf{A}$ (Algorithm 1) should be customized to only capture the objects of interest (which depend on the channel).
4.4.1 WCM for Elementary AS

We have seen multiple times that in an elementary AS, only the degree 1 check nodes are unsatisfied and all the degree 2 check nodes are satisfied. This means that, for an $(a, b, d_1, d_2, d_3)$ set, according to Theorem 2, $d_1 = b$ and $b_{ut} = 0$. It implies that by converting the object to an $(a, b + 1, d_1, d_2, d_3)$ set, the elementary AS can be removed. To achieve this, operating on only one WCM is sufficient. This WCM is obtained by removing only the degree 1 check nodes from the adjacency matrix $A$. The WCM obtained will always be of size $(l - d_1) \times a$. By applying our optimization framework (Algorithm 2) on the single WCM, elementary absorbing sets can be successfully removed.

Remark 9. Since only one WCM is involved in the optimization process, the removal process becomes much easier and was illustrated in Example 3.

4.4.2 WCM for BAS

Balanced ASs [24], were found to be the key contributors to the error floor in channels with memory such as the Partial Response (PR) Channel [25]. A BAS can be defined as follows:

**Definition 12.** An absorbing set that has $0 \leq b \leq \lfloor \frac{ag}{2} \rfloor$ is a balanced absorbing set (BAS) and an absorbing set that has $\lfloor \frac{ag}{2} \rfloor < b \leq ag$ is an unbalanced absorbing set (UBAS).

It was shown in [24] that it suffices to convert a BAS to UBAS (UBAS is not a problematic object in the PR channel). According to Theorem 2, for PR channel,

$$b^{pr}_{max} = \lfloor \frac{ag}{2} \rfloor$$

$$b^{pr}_{ut} = b^{pr}_{max} - d_1$$

(4.1)

where, $b^{pr}_{ut}$ is the upper bound on the number of degree 2 check nodes that can be unsatisfied, while the configuration remains as an AS. For the PR channel, the WCMs are obtained by simultaneously removing $b^{pr}_{max}$ number of possible combinations of check
nodes (according to Algorithm 1) from the adjacency matrix. By applying the proposed framework on the obtained set of WCMs, we can achieve BAS removal. The BAS removal process is similar to the GAST removal process with $b_{\text{max}} = \lfloor \frac{a_2}{2} \rfloor$. The below example illustrates the gain we can achieve by going for a customized definition.

**Example 5.** Consider the $(4,2,2,5,0)$ GAST, $\gamma = 3$, shown in Figure 4.3, to be a problematic object over the PR channel. The $(4,2,2,5,0)$ is a BAS. To remove this BAS from the Tanner graph, converting it to $(4,3,2,5,0)$ GAST does the job since $(4,3,2,5,0)$ is an UBAS. Therefore we will extract one WCM of size $5 \times 4$. This means that the number of edge weights available for modification is $10$ ($w_{11}, w_{12}, w_{22}, w_{23}, w_{33}, w_{34}, w_{44}, w_{41}, w_{52}, w_{54}$). By changing any of these available edge weights, over GF($q$), the BAS can be converted to a UBAS.

In contrast, if the object appeared on an asymmetric channel, the GAST will have to be removed completely (i.e., converting to a UBAS is not sufficient). In other words, according to Theorem 1 and Theorem 2 the check node $c_5$ can be unsatisfied such that the configuration remains as a GAST. To completely remove the GAST, we operate on one WCM of size $4 \times 4$. In order to remove the object with minimum number of edge weight changes, the weights associated with check node $c_5$ should not be modified as changing them alone cannot remove the GAST. This implies that the number of possible choices of edge weights that can be modified are fewer in number (8 in this example).

The issue of number of edge weights available for modification is crucial because in most cases, several absorbing sets may share a certain edge. Using too many edges to remove one GAST, may lead to a situation where no edge weight is available for the removal of another GAST. With more number of choices for edge weights available for modification, there is a lower probability of running out of edge-weights that can be changed. It means that we can remove many more GASTs from the Tanner graph and obtain an optimized code with a better performance. The performance improvement is mainly due to the additional degree of freedom provided by targeting BASs (and not the
larger class of ASs), which makes it more likely that appropriate edge weight choices can be made during the object removal process.

Figure 4.3: A (4,2,2,5,0) GAST, $\gamma = 3$. Example of a BAS

**Remark 10.** By going for a tailored WCM definition (specific to the channel), we can gain a lot in terms of algorithm complexity and the number of edge weights that need to be modified. As we can see in case of both elementary AS and BAS, the number of edge weight changes and algorithm complexity either remains the same or is significantly reduced without a compromise on the code optimization. This demonstrates the completeness of the proposed algorithm.
Chapter 5

Experimental Results

In this chapter, the effectiveness of the WCM framework is demonstrated by optimizing and simulating the performance of NB-LDPC codes over realistic Flash channels. Using a realistic, asymmetric channel—as opposed to a symmetric channel such as AWGN channel—allows us to better characterize objects that dominate the error-floor of NB-LDPC codes in practice.

5.1 The Normal-Laplace Mixture Flash (NLMF) Channel

In a recent work, Parnell et al. accurately modeled the voltage threshold distribution of sub-20nm MLC (2-bit) NAND Flash memory [19]. Each of the four states are modeled as normal-laplace mixture distributions, taking into account various sources of error due to wear-out effects. Through device testing, Parnell et al. provided accurate fitting results of the model for program/erase cycles up to ten times the manufacturer’s endurance specification. Figure 5.1 shows the threshold voltage distribution of the channel without and with programming errors (significant asymmetry).
5.1.1 Simulation Results

The NLMF channel with the parameters given in [19] was implemented for the 4 level Flash system with 3 reads (hard decision). The sector size is 512 bytes. A finite-precision Fase Fourier transform based $q$-ary sum product algorithm (FFT-QSPA) LDPC decoder [26] was used.

In the plots, Code 1 refers to a NB-QC-LDPC code with block length = 3996 bits, rate $\approx 0.89$, $\gamma = 3$, defined over GF(4). Code 2 is NB-QC-LDPC code defined over GF(4) with block length = 3280 bits, rate $\approx 0.8$ and $\gamma = 4$. Code 1 and Code 2 (unoptimized codes) are high performance NB-QC-LDPC codes designed according to [10], and do not have cycles of length 4 in their Tanner graphs. Code 3 (resp., Code 4) is obtained by optimizing Code 1 (resp., Code 2) by attempting to remove only the elementary ASs in its error profile table - 5.1 (resp., 5.2). Code 5 (resp., Code 6) is the result of optimizing Code 1 (resp., Code 2) for the NLMF channel by attempting to remove all the GASTs in 5.1 (resp., 5.2) using the WCM framework.
Figure 5.2: Simulation results for Code 1 (unoptimized), Code 3 (elementary removal),
and Code 5 (WCM framework) over NLMF channel. All the three codes have $\gamma = 3$.

Figure 5.3: Simulation results for Code 2 (unoptimized), Code 4 (elementary removal),
and Code 6 (WCM framework) over NLMF channel. All the three codes have $\gamma = 4$.

Figures 5.2 and 5.3 show that even though only 3 reads are used, the codes optimized
by removing GASTs using our WCM framework (Codes 5 and 6) outperform the unin-
formed codes (Codes 1 and 2) by more than 1 order of magnitude. More importantly,
the two figures show that conventional code optimization techniques which assume channel symmetry (e.g., techniques that would focus on the removal of elementary ASs) are ineffective for realistic memory channels.

<table>
<thead>
<tr>
<th>Error type</th>
<th>Count Code 1</th>
<th>Count Code 3</th>
<th>Count Code 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4, 2, 2, 5, 0)</td>
<td>45</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(4, 3, 2, 5, 0)</td>
<td>15</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>(4, 4, 4, 4, 0)</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(6, 0, 0, 9, 0)</td>
<td>12</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(6, 1, 0, 9, 0)</td>
<td>7</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>(6, 2, 0, 9, 0)</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>(6, 2, 2, 5, 2)</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>(7, 1, 0, 10, 1)</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Other</td>
<td>9</td>
<td>13</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 5.1: Error profile of Codes 1, 3, and 5 over the NLMF channel, RBER = 4.6e-4, UBER (unoptimized) = 1.04e-11, and UBER (optimized) = 9.04e-13 over the NLMF channel.

Table 5.1 (resp., 5.2) shows the error profiles of Codes 1, 3, and 5 (resp., 2, 4, and 6). The tables reveal the effectiveness of the WCM framework in removing detrimental GASTs. The tables further illustrate that optimizing the codes by removing only elementary ASs (as one would do for the case of symmetric channels) will not work for the NLMF channel. This is justified because, the unoptimized codes naturally have a high percentage of non-elementary ASs ($b > d_1$ or/and $d_3 \neq 0$) in their error profiles over the NLMF channel (31% for Code 1, and 26% for Code 2) and secondly, any technique that primarily focuses on the elimination of elementary ASs will not be effective as it would convert most of the elementary ASs into non-elementary ASs, which themselves are still problematic configurations. For example, Code 3 has many more (4, 3, 2, 5, 0) GASTs in its error profile compared to the Code 1 (see Table 5.1) mainly because many of these
(4, 3, 2, 5, 0) GASTs were originally (4, 2, 2, 5, 0) GASTs (elementary) and the optimization procedure of removing elementary ASs has converted them into (4, 3, 2, 5, 0) GASTs (non-elementary).

<table>
<thead>
<tr>
<th>Error type</th>
<th>Count</th>
<th>Code 2</th>
<th>Code 4</th>
<th>Code 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4, 4, 4, 6, 0)</td>
<td>47</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(6, 2, 2, 11, 0)</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(6, 4, 2, 11, 0)</td>
<td>14</td>
<td>14</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(6, 4, 4, 7, 2)</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(8, 3, 2, 15, 0)</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(8, 4, 4, 14, 0)</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>11</td>
<td>18</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Error profile of Codes 2, 4, and 6 over the NLMF channel, RBER = 1.2e−3, UBER (unoptimized) = 1.94e−13, and UBER (optimized) = 1.87e−14.

5.2 Another Practical Flash Channel

To strengthen our results, we have optimized and simulated the performance of NB-LDPC codes on another realistic Flash channel. The authors of [27] and [28] have developed a channel which models the threshold voltage distributions, for an MLC 2Y-nm (20-24nm) NAND Flash memory, by placing a Gaussian kernel function over each data point and then adding up the contributions over the whole data set. The threshold voltage distribution of the channel at different PE cycles is shown in Figure 5.4.

5.2.1 Simulation Results

Figure 5.5 shows the plots obtained when NB-LDPC codes were simulated over the channel of focus [27]. In the plot, Code 1 is the QC-NB-LDPC code (the same code was simulated over the NLMF channel, mentioned in section 5.1). Code 7 is the optimized
Figure 5.4: Threshold voltage distribution under various P/E cycles using nonparametric kernel density estimation code (over the channel of focus [27]) using the WCM framework. We observe in Figure 5.5 and in Table 5.2.1 that by using the proposed framework, an improvement of over one order of magnitude is achieved.

<table>
<thead>
<tr>
<th>Error type</th>
<th>Code 1</th>
<th>Code 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4, 2, 2, 5, 0)</td>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>(4, 3, 2, 5, 0)</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>(5, 3, 3, 6, 0)</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>(6, 0, 0, 9, 0)</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>(6, 1, 0, 9, 0)</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>(6, 2, 2, 8, 0)</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>(6, 2, 2, 5, 2)</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>(8, 0, 0, 12, 0)</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Other</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5.3: Error profile of Codes 1 and 7 over the channel in [27], RBER = 6.33e-4, UBER (unoptimized) = 9.86e-12, and UBER (optimized) = 8.12e-13
Figure 5.5: Simulation results for Code 1 (unoptimized), Code 7 (WCM framework) over the channel in [27], \( \gamma = 3 \).

Remark 11. Based on our simulations, the percentage of the non-elementary GASTs among the objects that dominate the error floor over the channel in [27] (\( \approx 19\% \)) is less than the percentage of non-elementary GASTs (\( \approx 31\% \)) over the NLMF channel, for the same QC-NB-LDPC code (Code 1). The intuition behind this phenomenon is that more the aggressive asymmetric nature of the channel, more the percentage of non-elementary GASTs. The NLMF channel is highly asymmetric compared to the channel in [27].
Chapter 6

Conclusion

In this work, we studied in detail the error floor performance of NB-QC-LDPC codes over practical Flash (asymmetric) channels. We demonstrated that the error profile over realistic Flash channels is different than that over symmetric channels such as the AWGN channel; the existing code optimization techniques previously developed for the AWGN channel transmission are thus not effective. We introduced novel combinatorial definitions and mathematical tools needed for a proper study of NB-LDPC codes in asymmetric memory channels. The WCM framework was applied to the realistic NLMF channel where clear benefits of the proposed technique relative to the existing approaches were demonstrated. As many emerging memory devices exhibit an increased level of asymmetry, the presented framework can be a valuable code design and optimization tool, that will enable memory engineers to use LDPC codes with confidence.
References


