UNIVERSITY OF CALIFORNIA

Los Angeles

Non-Binary Protograph-Based LDPC Codes: Analysis, Enumerators and Designs

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Electrical Engineering

by

Yizeng Sun

2013

ABSTRACT OF THE THESIS

Non-Binary Protograph-Based LDPC Codes: Analysis, Enumerators and Designs

by

Yizeng Sun

Master of Science in Electrical Engineering

University of California, Los Angeles, 2013

Professor Lara Dolecek, Chair

Non-binary LDPC codes can outperform binary LDPC codes using sum-product algorithm with higher computation complexity. Non-binary LDPC codes based on protographs have the advantage of simple hardware architecture. In the first part of this thesis, we will use EXIT chart analysis to compute the thresholds of different protographs over GF(q). Based on threshold computation, some non-binary protograph-based LDPC codes are designed and their frame error rates are compared with binary LDPC codes. For maximum-likelihood decoder, weight enumerator can predict frame error rate of an LDPC code. In the second part of this thesis, we calculate weight enumerators of protograph-based non-binary LDPC code ensembles both for finite length case and asymptotic case. In addition, the trapping set and stopping set enumerators are presented.

The thesis of Yizeng Sun is approved.

Danijela Cabric

Richard Wesel

Lara Dolecek, Committee Chair

University of California, Los Angeles

2013

DEDICATION

To my parents.

Contents

Chapter 1

Introduction

1.1 Background

In 1948, Claude Shannon published his celebrated paper *A mathematical theory of communication* on reliable communication over noisy channels [1]. This work founded the fields of channel coding, source coding and information theory. Shannon's central theme was that we can communicate reliably with error probability tending to 0 over a noisy channel provided that the information rate for a given code did not exceed the capacity of the channel. However, in the proof of the theorem, Shannon assumed random generated codes and that the block lengths of codes must tend to infinity to achieve near zero error probability. The encoding and decoding algorithms for such codes have high complexity and are not practical in real world applications.

After Shannon's publication, researchers devised a large number of effective coding schemes over the following decades. Examples include Hamming codes [2], BCH codes [3], Reed-Solomon codes [4] and convolutional codes [5]. However, none of these codes can approach Shannon's theoretical limit in a practical channel. The breakthrough came in 1993 with the discovery of turbo codes [7]. Turbo codes applied the iterative decoding scheme and were the first class of codes shown to approach Shannon's capacity limit. The second breakthrough was the rediscovery of low-density parity-check(LDPC) codes [9, 10] which can also approach capacity in practical use.

1.2 LDPC codes

LDPC codes are a class of linear block codes invented by Gallager in his 1960 PhD thesis [8]. Gallager noticed the benefit of expressing a linear block code with low-density parity-check matrix. Gallager also invented the decoding algorithm which worked locally on graphical models. This algorithm is known as sum product algorithm. This algorithm was reinvented and generalized by Judea Pearl [11] in 1982 as a belief propagation algorithm and is widely used in Bayesian networks and Markov random fields. However, initially LDPC codes didn't attract much attention and were largely forgotten in the following thirty years. The reason is the hardware capabilities in the 1960s couldn't match the implementation needs of the sum-product algorithm. After their rediscovery in mid 1990s, LDPC codes attracted intense attention.

1.2.1 Matrix representation

Definition 1. *Assume F_q is the finite space with q elements and F_q^n is the vector space with of all the n-tuples over F_q. A block code C of length n with 2^k codewords is a linear block code if and only if all the codewords form a subspace B_c of F_q^n with dimension k. It is called a q-ary (n, k) linear block code. When q = 2, the code becomes a binary code. All vectors in C are called codewords.*

In particular, for an (n, k) linear block code C over finite field F_q, all the codewords can be represented be the combination of vectors in the subspace B_c. We can express a basis of the subspace B_c in a matrix G, called the generator matrix, such that

$$\forall c \in C, \exists u \in F_q^k, \ s.t. \ c = uG. \tag{1.1}$$

Note for the subspace B_c, there are many possible bases, so the generator matrix is not

unique. Equation (1.2) is the generator matrix of $(7, 4)$ Hamming code.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{1.2}$$

From linear algebra, the null space of B_c is a $n - k$ dimension subspace of F_q^n. We can express a basis of the null space of B_c as a matrix H, called the parity-check matrix, such that

$$\forall c \in C, cH^T = 0. \tag{1.3}$$

Note for the null space of B_c, there are many possible bases, so the parity-check matrix is not unique. Equation (1.4) is the parity-check matrix of $(7, 4)$ Hamming code.

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{1.4}$$

As a linear block code, LDPC code can be represented as the null space of a parity-check matrix. The density of non-zeros elements in the parity-check matrix H is sparse, i.e., most elements in H are zero. A regular LDPC code is a linear block code whose parity-check matrix H contains d_v non-zero elements in each column and d_c non-zero elements in each row. Assuming that the dimension of H is $m \times n$, we can get

$$d_v = d_c \frac{m}{n}. \tag{1.5}$$

If the number of non-zero elements in each row or column is not constant, then the code is an irregular LDPC code.

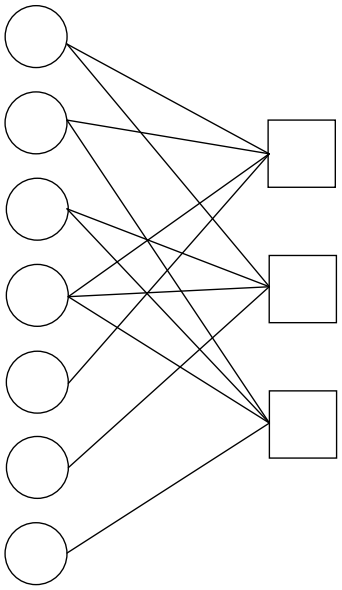

Figure 1.1: Tanner graph of Hamming (7,4) code.

1.2.2 Graphical representation

Tanner [10] invented the graphical representation of LDPC codes, now called Tanner graph. This bipartite graph representation is equivalent to the parity-check matrix representation. A bipartite graph is a graph whose vertices can be divided into two disjoint sets check nodes and variable nodes such that every edge connects a check node and a variable node. With the parity-check matrix, the associated Tanner graph can be obtained easily. Each row of the parity-check matrix H corresponds to a check node and each column of the parity-check matrix H corresponds to a variable node. Check node i is connected to variable node j whenever element H_{ij} in H is non-zero. For binary LDPC codes, the weight of each edge is always one and not indicated explicitly. For non-binary LDPC codes over GF(q), the weight of each edge is equal to $H_{ij} \in GF(q) \backslash 0$.

Figure 1.1 shows the associated Tanner graph of a Hamming $(7, 4)$ code with the parity-check matrix shown in (1.4).

The degree of a check node is equal to the number of non-zero elements in the corresponding row of H and the degree of a variable node is equal to the number of

non-zero elements in the corresponding column of H. Thus, for regular LDPC codes, we still use d_c to denote check node degree and use d_v to denote variable node degree. For irregular LDPC codes, the degrees are not constant and are usually specified by variable node and check node degree distribution polynomials, denoted by $\lambda(x)$ and $\rho(x)$ respectively. In the polynomial

$$\lambda(x) = \sum_{i=1}^{r_{max}} \lambda_i x^{i-1}, \tag{1.6}$$

λ_i denotes the fraction of all edges connected to degree-i variable nodes and r_{max} denotes the maximum variable node degree. Similarly, in the polynomial

$$\rho(x) = \sum_{i=1}^{c_{max}} \rho_i x^{i-1}, \tag{1.7}$$

ρ_i denotes the fraction of all edges connected to a degree-i check nodes and c_{max} denotes the maximum check node degree.

1.3 Protograph-based LDPC codes

A protograph $G = (V,C,E)$ [12] is a relatively small bipartite graph that consists of the set $V = \{v_1,v_2,\ldots,v_{n_v}\}$ of variable nodes, the set $C = \{c_1,c_2,\ldots,c_{n_c}\}$ of check nodes and the set $E = \{e_1,e_2,\ldots,e_{|E|}\}$ of edges connecting variable nodes and check nodes. In Figure 1.2, a regular-$(2,4)$ protograph is presented. In this protograph, there are 4 variable nodes each of degree 2 and 2 check nodes each of degree 4.

A protograph-based LDPC code is obtained from the protograph by a copy-permute-and-scale procedure. First, the protograph G is copied N times to get a large graph $G^N = (V^N, C^N, E^N)$. In this procedure, each variable node $v_i \in V$ (check node $c_i \in C$) in protograph G is copied N times to yield the set $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,N}\}$ of variable nodes (the set $C_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,N}\}$ of check nodes) in the large graph G^N. Likewise, each edge $e_i \in E$ in protograph G produces the set E_i of edges in the graph G^N where $E_i = \{e_{i,1}, \ldots, e_{i,N}\}$, and the edge $e_{i,j}$ for $1 \leq j \leq N$ connects the variable

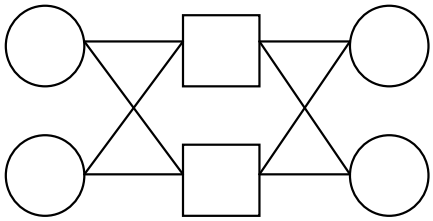

Figure 1.2: Regular-(2,4) protograph

node $v_{k,j}$ and the check node $c_{l,j}$ if the edge e_i connects the variable node v_k and the check node c_l in G. Thus, G^N contains N copies of G.

Second, in each set $E_i = \{e_{i,1}, \ldots, e_{i,N}\}$, the N edges are permuted by a $N \times N$ permutation matrix $\{\pi_i\}$. In this procedure, individual protographs in G^N are interconnected while local structure is preserved.

Third, each edge is given a non-zero element in the field $GF(q)$, called the edge weight. If the protograph code is over $GF(2)$, the edge weight can only be 1 and this step is omitted.

1.4 Thesis overview

In Chapter 2, we introduce the extrinsic information transfer (EXIT) chart for protograph-based non-binary LDPC codes. With this method, we calculate the thresholds of different protographs over $GF(q)$. Using protographs with low threshold, we then design some non-binary protograph-based LDPC codes and compare their performance with their binary counterparts.

Chapter 3 gives the weight enumerator of protograph-based non-binary LDPC codes both for random edge weights and edge weights with some constraints. In addition to the weight enumerator, we also give the enumerator of stopping sets and trapping sets.

Finally, Chapter 4 gives the conclusion of this work.

Chapter 2

EXIT chart analysis and design of protograph-based non-binary LDPC codes

2.1 Introduction

LDPC codes are shown to approach Shannon capacity over noisy channels with sum-product algorithm. The effectiveness of sum-product algorithm depends on the structure of the Tanner graph and degree distribution of check nodes and variable nodes, i.e. d_v and d_c for regular codes and $\lambda(x)$ and $\rho(x)$ for irregular codes.

In order to choose good degree distributions, Richardson and Urbanke developed density evolution [14] to predict thresholds of LDPC codes. Another method is extrinsic information transfer (EXIT) chart [13] invented by S. Brink. Both method assume the length of codewords tends to infinity and no cycles in the Tanner graph, thus the input messages of each node are independent. The difference is that density evolution keeps track of the probability distribution of messages over edges and EXIT chart instead uses the mutual information (MI) between an edge message and the associated transmitted bit to dramatically simplify the threshold computation. In EXIT chart, the iterative decoding procedure becomes the evolution of MI. If the MI goes to 1, then the decoding

is successful.

In this chapter, the sum-product algorithm and Brink's original EXIT chart are first introduced. We then apply protograph EXIT(PEXIT) chart [16] to non-binary protograph-based LDPC codes.

2.2 Log-likelihood sum-product decoder

Consider the AWGN channel with power spectral density N_0 and noise variance $\sigma_n^2 = \frac{N_0}{2}$ using BPSK modulation. The normalized signal-to-noise ratio (SNR) is defined as

$$\frac{E_b}{N_0} = \frac{1}{2R\sigma_n^2}, \tag{2.1}$$

where E_b is the energy used per information bit and R is the rate of the code. Consider the log-domain sum-product algorithm as the decoding algorithm. Assume X is the transmitted bit and Y is the output of the channel. Then $Y = X+N$ and $N \sim \mathcal{N}(0, \sigma_n^2)$. The log-likelihood ratio (LLR) of the channel output becomes

$$L_{ch}(y) = log\frac{P(X=+1|y)}{P(X=-1|y)} = \frac{2}{\sigma_n^2}y. \tag{2.2}$$

From this equation, we know that $L_{ch}(y)$ is a Gaussian random variable subject to $\mathcal{N}(\pm\frac{2}{\sigma_n^2}, \frac{4}{\sigma_n^2})$.

At the variable node side, for a degree d_v variable node connected to check nodes $(1, 2, \ldots, d_v)$, assume $L^v_{j,in}$ is the input of this variable node from check node j. Then the output message from this variable node to the check node i is

$$L^v_{i,out} = L_{ch} + \sum_{j\in(1,2,\ldots,d_v),j\neq i} L^v_{j,in}. \tag{2.3}$$

At the check node side, the calculation is more complex. For a degree d_c check node connected to variable nodes $(1, 2, \ldots, d_c)$, assume $L^c_{j,in}$ is the input of this check node from variable node j. Then the output message from this check node to the variable

node i is

$$L^c_{i,out} = \sum_{j \neq i} \boxplus L^c_{j,in}. \tag{2.4}$$

The boxplus $\boxplus$ is defined as

$$L_1 \boxplus L_2 = \log(\frac{1 + e^{L_1+L_2}}{e^{L_1} + e^{L_2}}). \tag{2.5}$$

2.3 EXIT chart for regular and irregular binary LDPC codes

The original EXIT chart [13] is used to calculate the thresholds of regular and irregular binary LDPC codes.

2.3.1 Mutual information and EXIT chart

With equation (2.2), we can compute the MI between the LLR of channel output and the transmitted bit, i.e. $I(X, L_{ch}(Y))$. We use $J(\sigma_{ch})$ to denote it, where $\sigma^2_{ch} = \frac{4}{\sigma^2_n}$.

$$J(\sigma_{ch}) = H(X) - H(X|L_{ch}(Y)) = 1 - \int_{-\infty}^{\infty} \frac{e^{-(t-\sigma^2_{ch}/2)^2/2\sigma^2_{ch}}}{\sqrt{2\pi\sigma^2_{ch}}} log_2(1 + e^{-t})\, dt. \tag{2.6}$$

The function $J(\sigma)$ can be treated as the MI between a binary random variable $P \in \{\frac{\sigma^2}{2}, -\frac{\sigma^2}{2}\}$ and $Q = P + N$, where N is a zero mean Gaussian random variable with variance σ^2.

With equations (2.3) and (2.4), we can iteratively track the evolution of MI. In order to simplify the calculations, the probability distributions of $L^v_{j,in}$ and $L^c_{j,in}$ are approximated as the LLRs of the output of an AWGN channel with BPSK modulation. Then, for regular LDPC codes with variable node degree d_v and rate R, the extrinsic MI becomes

$$I_{Ev}(I_{Av}, d_v, \frac{E_b}{N_0}, R) = J(\sqrt{(d_v - 1)[J^{-1}(I_{Av})]^2 + \sigma^2_{ch}}), \tag{2.7}$$

where I_{Av} denotes the *a priori* MI between the input message to the variable node and the transmitted bit. I_{Ev} denotes the extrinsic MI between the output message of a variable node and the transmitted bit.

On the check node side, for regular LDPC codes with check node degree d_c, we use an approximation from [13],

$$I_{Ec}(I_{Ac}, d_c) \approx 1 - J(\sqrt{d_c - 1}J^{-1}(1 - I_{Ac})), \tag{2.8}$$

where I_{Ac} denotes the *a priori* MI between the input message of a check node and the transmitted bit. I_{Ec} denotes the extrinsic MI between the output message of a check node and the transmitted bit.

For irregular LDPC codes with edge degree distributions $\lambda(x)$ and $\rho(x)$, the EXIT functions are given by [13],

$$I_{Ev} = \sum_{i=1}^{r_{max}} \lambda_i I_{Ev}(I_{Av}, i, \frac{E_b}{N_0}, R), \tag{2.9}$$

and

$$I_{Ec} = \sum_{i=1}^{c_{max}} \rho_i I_{Ec}(I_{Ac}, i), \tag{2.10}$$

where λ_i denotes the fraction of all edges connected to degree-i variable nodes and r_{max} denotes the maximum variable node degree. Here, ρ_i denotes the fraction of all edges connected to a degree-i check nodes and c_{max} denotes the maximum check node degree. These definitions are the same as equations (1.6) and (1.7).

At the beginning of the decoding, $I_{Ev}^{(0)} = I_{ch}$ and $I_{Ec}^{(0)} = 0$, where the numbers in brackets denotes the iteration index. Since the output message of a variable node is the input message of a check node and vice versa, the MI can be computed iteratively using equations (2.3) and (2.4). It will either converge to 1 or stop at some value less than 1. The evolution of MI of a successful decoding is shown in Figure 2.1.

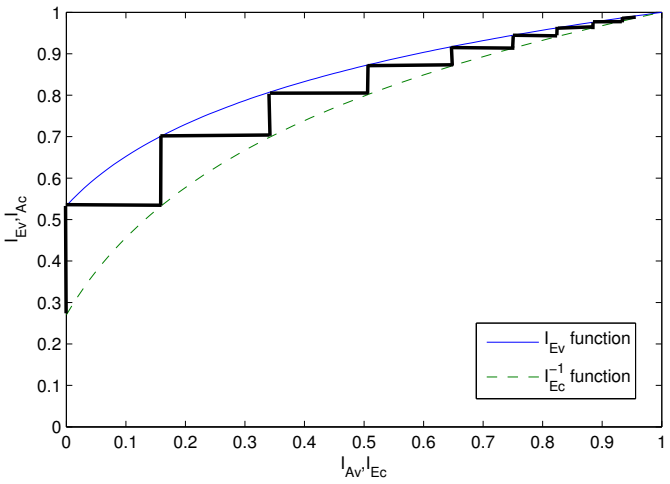

Figure 2.1: The evolution of mutual information in EXIT chart.

2.4 Exit chart analysis for protograph-based LDPC codes

2.4.1 Protograph-based binary LDPC codes

Unlike general regular or irregular LDPC codes, protograph codes have fixed structures which makes the original EXIT chart calculation less accurate. For example, different protograph codes with the same degree distribution may have different thresholds [16], but the original EXIT chart can not distinguish among them. For this reason, Liva [16] developed protograph EXIT (PEXIT) chart for binary protograph codes. In PEXIT chart, we compute the MI between messages over each edge in the protograph and the associate transmitted bit iteratively. If the *a posteriori* MI of each variable node converges to 1, the decoding is successful.

2.4.2 Probability distribution of messages for non-binary LDPC codes

For a non-binary LDPC code over GF(q), messages passed in each edge are LLR vectors of dimension $q-1$. Bennatan et al. [15] proved that check-to-variable messages transmitting across an edge can be well approximated by a Gaussian distribution with the message symmetry and permutation invariance properties. These properties can be achieved by two assumptions. First, all the edge weights are uniformly distributed over

$(1, q-1)$. Second, before transmitting through the channel, a random coset vector is added to the codeword. After receiving, the coset vector is subtracted. Then, the mean and variance of a check-to-variable message is can be represented by equations (2.11) and (2.12). We can see that this Gaussian distribution is determined only by one variable σ,

$$\mu = \begin{bmatrix} \sigma^2/2 \\ \sigma^2/2 \\ \vdots \\ \sigma^2/2 \end{bmatrix}_{(q-1)\times 1}, \tag{2.11}$$

and

$$\Sigma = \begin{bmatrix} \sigma^2 & & & \sigma^2/2 \\ & \sigma^2 & & \\ & & \ddots & \\ \sigma^2/2 & & & \sigma^2 \end{bmatrix}_{(q-1)\times(q-1)}. \tag{2.12}$$

For the variable-to-check message, we can get the probability distribution from equation (2.3), in which $L^v_{j,in}$ is the LLR of a Gaussian random vector with mean μ and covariance matrix Σ specified by equations (2.11) and (2.12) and L_{ch} depends on the channel and modulation.

2.4.3 EXIT chart for protograph-based non-binary LDPC codes

With the probability distribution of messages for non-binary LDPC codes, Chang et al. [21] extended the binary PEXIT chart to the non-binary case. Assume that B is the base matrix of a protograph and that b_{ij} is the (i, j)-entry of the matrix B. This entry denotes the number of edges connecting the check node *i* and the variable node *j* in the protograph. The mutual information between the message from variable node j to check node i and the transmitted symbol v_j is formulated as [21]:

$$I_{Ev}(ij) = \begin{cases} J(\sigma_{v-to-c}) & \text{if node } j \text{ is punctured,} \\ J_R(\sigma_{v-to-c}) & \text{otherwise.} \end{cases} \tag{2.13}$$

Here

$$J(\sigma_{v-to-c}) = I(S; W = X), \tag{2.14}$$

$$J_R(\sigma_{v-to-c}) = I(S; W = X + Y), \tag{2.15}$$

and

$$\sigma^2_{v-to-c} = \sum_{s \in N(j), s \neq i} b_{sj}[J^{-1}(I_{Av}(s, j))]^2 + \\ +(b_{ij} - 1)[J^{-1}(I_{Av}(i, j))]^2, \tag{2.16}$$

where X is a Gaussian random vector with parameter σ_{v-to-c}, Y is the random vector of the initial messages, S is the transmitted symbol and $I_{Ev}(i, j) = 0$ if $b_{ij} = 0$.

The MI between the message from the check node i to the variable node j and the transmitted symbol v_j is formulated as a Gaussian random vector with the parameter σ_{c-to-v}:

$$I_{Ec}(ij) = 1 - J(\sigma_{c-to-v}), \tag{2.17}$$

with

$$\sigma^2_{c-to-v} = \sum_{s \in N(i), s \neq j} b_{is}[J^{-1}(1 - I_{Ac}(i, s))]^2 + \\ (b_{ij} - 1)[J^{-1}(1 - I_{Ac}(i, j))]^2. \tag{2.18}$$

The *a posteriori* MI of variable node j and the transmitted bit can be calculated by,

$$I_{AP}(j) = \begin{cases} J(\sigma_{AP}) & \text{if node } j \text{ is punctured,} \\ J_R(\sigma_{AP}) & \text{otherwise,} \end{cases} \tag{2.19}$$

with

$$\sigma^2_{AP} = \sum_{s \in N(j)} b_{sj}[J^{-1}(I_{Av}(s, j))]^2. \tag{2.20}$$

The evaluation process ends when either $I_{AP}(j) = 1$ for all variable nodes which means that the decoding is successful or when the maximum number of iterations is reached which means that the decoding fails.

With the non-binary PEXIT chart, we can calculate the thresholds of protographs

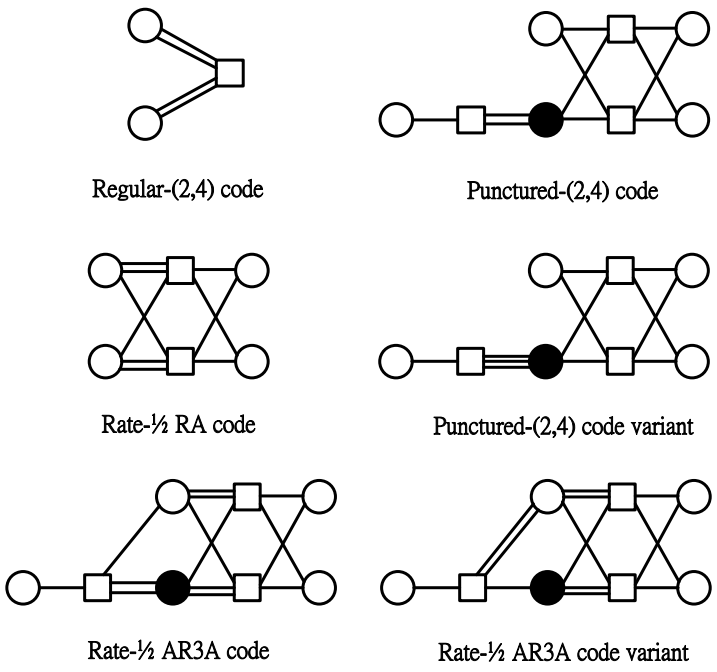

Figure 2.2: The structures of the protographs, the filled nodes are punctured.

in Figure 2.2 for different q. The thresholds of the binary image of the protographs with BPSK modulation over AWGN channel is shown in Figure 2.3. Note for $q \leq 32$, the results were already derived in [21]. In Figure 2.3, we can observe that protographs with small average variable node degree(≤ 2.5) tends to have higher threshold for small q and lower thresholds for large q. This observation can help us to choose protograph when we want to design a protograph code over $GF(q)$.

Similar to the analysis for the binary image of non-binary protograph codes over AWGN channel, we apply the Monte Carlo method to approximate $J(\sigma)$ and $J_R(\sigma)$ for PAM and PSK modulations over AWGN channel. For PAM we apply a technique in [27], which suggests a non-uniform constellation for PAM. Figure 2.4 illustrates the average bit SNR thresholds over PAM for some protographs in figure 2.2. Figure 2.5 illustrates the results with QPSK modulation over AWGN for some protographs in Figure 2.2. Similar to the BPSK modulation over AWGN channel, regular-(2,4) code has higher threshold when q is small and lower threshold when q is large.

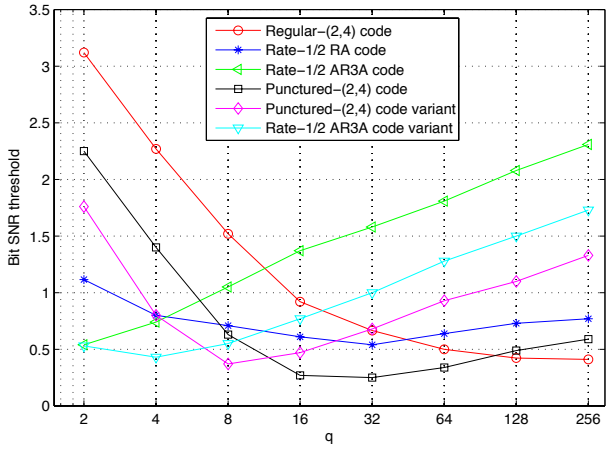

Figure 2.3: Bit SNR thresholds of binary image of PB NB LDPC codes with BPSK over AWGN channel

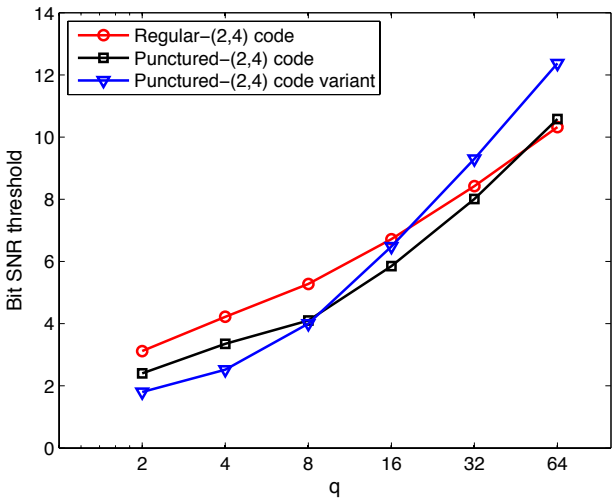

Figure 2.4: Bit SNR thresholds of PB NB LDPC codes with PAM over AWGN channel.

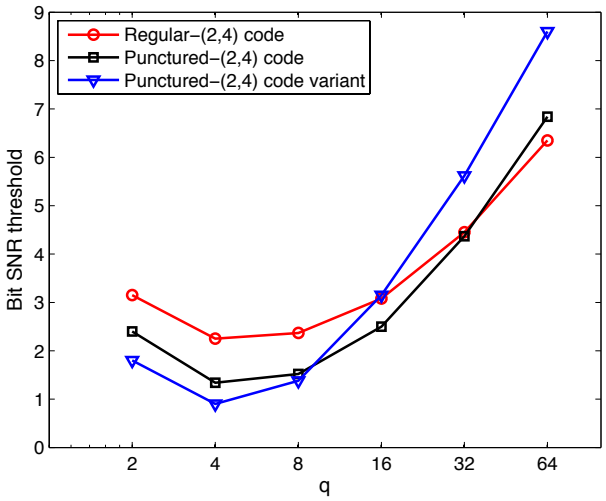

Figure 2.5: Bit SNR thresholds of PB NB LDPC codes with QPSK over AWGN channel.

2.4.4 Design examples

In this part, we try to design non-binary prootgraph code with large q. In Figure 2.3, the regular-$(2, 4)$ code is shown to have the lowest threshold for $q = 256$ and we use this protograph to design LDPC codes over $GF(256)$. It is known that short cycles in the Tanner graph will deteriorate the performance of LDPC codes. We use progressive edge-growth (PEG) [17] algorithm to generate Tanner graphs with large girth and fewer short cycles. Table 2.1 shows the number of different length cycles of the three designed codes. The edges weights are assigned by two rules. First, similar to [24] [25], the edge weights are chosen to maximize the minimum distance of a check node. Second, for $d_v = 2$, each cycle can be represented by a square matrix. If the square matrix is not full rank, there will be codewords associated with this cycle, which is not desirable. We try to choose edge weights to minimize the number of cycles associated with codewords. In Table 2.1, the numbers in brackets are the number of not full rank cycles. For comparison, we also generate a regular-$(2, 4)$ protograph code with random permutations and edge weights. The number of cycles is shown in Table 2.2.

When $N = 4, 8, 16$, the transmitted bits of corresponding codes are 128, 256 and 512. The simulation results of (128,64), (256,128) and (512,256) non-binary LDPC codes are shown in Figure 2.6. For comparison, performance of binary LDPC codes

n(bits)	N / cycle size	8	12	16
128	4	36(0)	96(0)	72(0)
256	8	20(0)	160(0)	634(4)
512	16	0	208(2)	788(0)

Table 2.1: Number of cycles for different designed codes.

n(bits)	N / cycle size	4	8	12	16
128	4	3(0)	21(0)	71(0)	66(0)
256	8	4(0)	25(0)	76(0)	316(1)
512	16	11(0)	30(0)	49(0)	340(2)

Table 2.2: Number of cycles for different random protograph codes.

and random regular non-binary LDPC codes over $GF(256)$ are presented. As we can see, our designed codes perform better than random codes and binary LDPC codes.

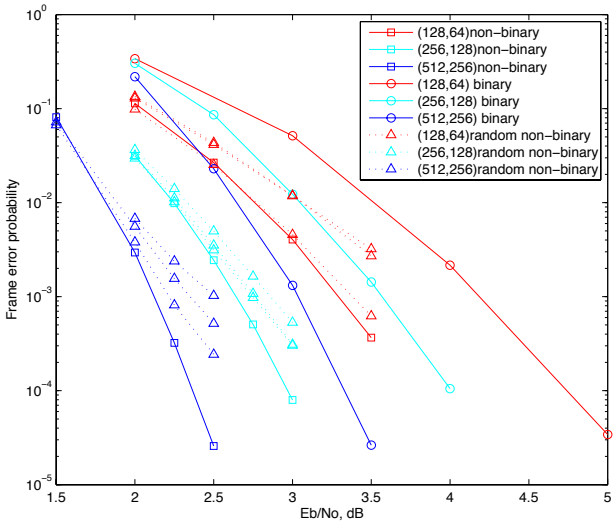

Figure 2.6: Frame error rates of rate 1/2 LDPC codes.

By puncturing the regular$-(2,4)$ protograph, we can get a rate $2/3$ protograph shown in Figure 2.7. In Figure 2.8, the NB-PEXIT analysis shows that the rate $2/3$ protograph has the lowest threshold when $q = 256$ (1.15dB), only 0.1dB higher than the channel capacity of 1.059dB. Using the same Tanner graphs and labels with the three regular-$(2,4)$ codes, we construct non-binary LDPC codes with rate $2/3$ and block-lengths 96, 192 and 384 in bits by taking the binary image of constructed non-binary protograph codes over GF(256). Performance simulations of these codes are shown in Fig. 2.9. For comparison, we plot simulation results of binary LDPC codes with rate

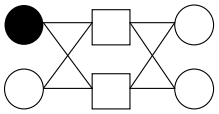

Figure 2.7: Punctured regular-(2,4) code protograph.

$2/3$ and length 192 bits in [28]. Our non-binary code over GF(256) outperforms the binary code by 1.0dB. We also compare our 384 bits code with binary rate compatible code in [29] with lower rate $R = 0.6$ and more than doubled in length. Our code shows similar performance in terms of bit error rate at only half the code length.

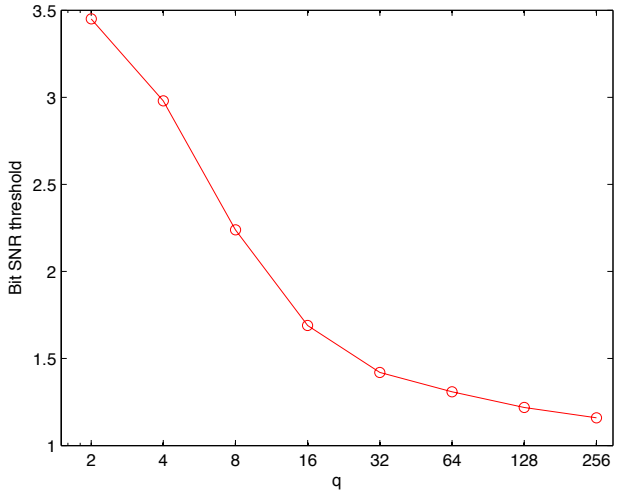

Figure 2.8: Punctured regular-(2,4) code thresholds.

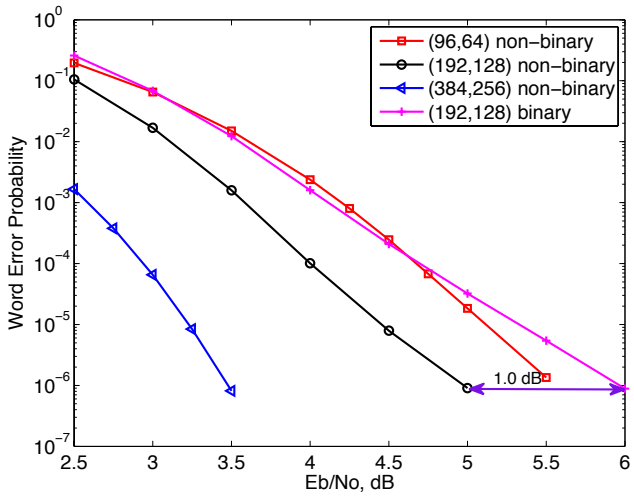

Figure 2.9: Frame error rates of rate 2/3 LDPC codes.

Chapter 3

Weight enumerators of protograph-based non-binary LDPC code ensembles

3.1 Introduction

Protograph-based LDPC codes have the advantages of a simple design procedure and highly structured encoders and decoders. In order to choose good protographs to design codes, we can use the method in Chapter 2 to calculate the threshold. In this chapter, we calculate weight enumerators (WEs) of protograph codes.

For a given linear code, the codeword WE is a polynomial that specifies the number of codewords of each possible weight. Codeword WEs can be used to estimate the code error rate with maximum likelihood (ML) decoding. For belief propagation decoding, because the existence of structures like trapping sets and stopping sets, the code error rate cannot be determined by the WEs, but it is still important to avoid low weight codewords. However, it is generally very difficult to calculate the WE of a specific code. Given this, the average WEs for code ensembles is often calculated.

WEs for ensemble of protograph-based binary LDPC codes and generalized LDPC codes are derived in [19] for finite and asymptotic case. In the paper, the authors also

gave the method to enumerate non-codeword objects that can cause decoding failure including trapping sets, stopping sets. In [22] [23], Divsalar et al. extended the derivation to non-binary case. In this chapter, we follow the methods used in [22] [23] to get WE for non-binary protograph codes.

3.2 Symbol weight enumerator for ensemble of protograph-based non-binary LDPC code

3.2.1 Definitions

Following the protograph-based non-binary (NB PB) LDPC codes definition in Chapter 1, we define the protograph-based code ensemble and its WE.

Definition 2 (Protograph-based code ensemble)**.** *Given a protograph $G = (V, C, E)$, we repeat it N times to get a large graph $G^N = (V^N, C^N, E^N)$. The (G, N, q) protograph-based code ensemble is the collection of codes with all possible permutations of $E_i = \{e_{i,1}, \ldots, e_{i,N}\}$ for every $e_i \in E$ and all possible edge weights assignments.*

Definition 3 (Protograph-based code ensemble WE)**.** *For a (G, N, q) protograph-based code ensemble, the WE is the average number of codewords with each Hamming weight for all codes in the ensemble.*

3.2.2 Symbol weight enumerator for finite length ensemble

The symbol weight enumerator for finite length protograph code ensemble is derived in [22]. First, for the simplest case, consider a protograph with a single check node c and m variable nodes connected to it by single edge. This protograph can be treated as a $(m, m-1)$ linear block code C over $GF(q)$. For this code, there exists $K = q^{m-1}$ codewords. Then we place all the K codewords in a $K \times m$ matrix M, such that each row of M is a different codeword of C.

From M, we can get a binary $K \times m$ matrix M_b which is obtained by converting all non-zero elements of M to 1. Note some rows of M_b will be the same. Define a $K_r \times m$ binary matrix $M_{b,r}$ as the submatrix of M_b that consists of all distinct rows of M_b. For a $(m, m-1)$ linear block code over $GF(q)$, the weight of non-zero codeword are between 2 and m. Then, the number of rows of $M_{b,r}$ is equal to $K_r = 1 + \sum_{i=2}^{m} \binom{m}{i}$.

When the check node is repeated N times, we get a Tanner graph with N check nodes and mN variable nodes. It can be treated as an $(mN, (m-1)N)$ linear block code C^N over $GF(q)$. These long codewords can be treated as the concatenation of N codewords $\in C$. We can use n_k to denote the number of occurrences of the k^{th} codeword among these N short codewords. Then collect them in a $1 \times K$ vector $\mathbf{n}$, where $\mathbf{n} = [n_1, n_2, \ldots, n_K]$. It is convenient to use a vector $\mathbf{w} = [w_1, w_2, \ldots, w_m]$ to denote the weight vector of a codeword in C^N, where w_i denotes the number of non-zero elements of variable nodes $\{v_{i,1}, v_{i,2}, \ldots, v_{i,N}\}$.

Let us use $A^{C^N}(\mathbf{w})$ to denote the average number of codewords with weight-vector $\mathbf{w}$ of the (C, N, q) ensemble. The calculation of $A^{C^N}(\mathbf{w})$ is given in [22].

Theorem 1. *The weight-vector enumerator $A^{\mathcal{C}^N}(\mathbf{w})$ of (C, N, q) ensemble is given by,*

$$A^{\mathcal{C}^N}(\mathbf{w}) = \sum_{\{\mathbf{n}\}} C\left(N; n_1, n_2, \ldots, n_{K_r}\right) e^{\mathbf{n} \cdot \mathbf{f}_q^T}, \tag{3.1}$$

where $C\left(N; n_1, n_2, \ldots, n_{K_r}\right)$ is the multinomial coefficient

$$C\left(N; n_1, n_2, \ldots, n_{K_r}\right) = \frac{N!}{n_1! n_2! \ldots n_{K_r}!}, \tag{3.2}$$

and $\{\mathbf{n}\}$ is the set of integer-vector solutions to $\mathbf{w} = \mathbf{n} \cdot \mathbf{M}_{b,r}^{\mathcal{C}}$, with $n_1, n_2, \ldots, n_{K_r} \geq 0$, and $\sum_{k=1}^{K_r} n_k = N$. The vector $\mathbf{f}_q = [f_{q,1}, f_{q,2} \ldots, f_{q,K_r}]$ has entries $f_{q,k} = \ln g(q, |\mathbf{x}_k|)$, where $\mathbf{x}_k$ is the k-th row of $\mathcal{M}_{b,r}^{\mathcal{C}}$, $|\mathbf{x}_k|$ is the weight of $\mathbf{x}_k$, and $g(q,i) = \frac{q-1}{q}[(q-1)^{i-1} + (-1)^i]$.

We follow the combinatorial calculation of [19] to calculate the symbol WE of an ensemble. For binary LDPC codes, a length-L uniform interleaver is a probabilistic device that map each Hamming weight-w interleaver input into the $\binom{L}{w}$ distinct permu-

tations of it with equal probability. A non-binary uniform interleaver over $GF(q)$ is similar to the binary case, except for each edge there are $(q-1)$ possible edge weights. For symbol weight-w input, the interleaver gives $(q-1)^w\binom{L}{w}$ possible outputs with equal probability. With the uniform interleaver of non-binary code, we have [22]

Lemma 1. *Consider two serially concatenated linear block codes $\mathcal{C}_1$ and $\mathcal{C}_2$, connected by a length N non-binary uniform interleaver. Then, the average number of codewords over all possible interleavers with symbol weight-d inputs and symbol weight-f outputs is given by*

$$A_{f,d}^{SCC} = \sum_{w} \frac{A_{f,w}^{\mathcal{C}_1} A_{w,d}^{\mathcal{C}_2}}{(q-1)^w \binom{K_2}{w}}, \tag{3.3}$$

where $A_{f,w}^{\mathcal{C}_1}$ is the number of codewords in $\mathcal{C}_1$ of Hamming weight w corresponding to $\mathcal{C}_1$-encoder inputs of Hamming weight f, and $A_{w,d}^{\mathcal{C}_2}$ is the number of codewords in $\mathcal{C}_2$ of Hamming weight d corresponding to C_2-encoder inputs of Hamming weight w.

Similar to the binary protograph ensemble in [19], the weight enumerator of non-binary protograph ensemble can be computed use the serial concatenated code scheme. In this case, each variable node v_i(check node c_i) in the protograph G is repeated to get a group of N variable nodes $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,N}\}$ (check nodes $C_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,N}\}$). The group of N variable nodes V_i is considered to be a constituent repetition code with a weight-d_i input of length N. Further, the group of N check nodes C_i is considered to be a constituent code with a fictitious output of weight zero. In this way, a protograph code is treated as a serial concatenated code. Now, we can apply lemma 1 to get the average number of codewords with weight-vector $\mathbf{w}$ of an ensemble [22].

Theorem 2. *The weight-vector enumerator of a ensemble of protograph-based non-binary LDPC code is*

$$A(\mathbf{d}) \;=\; \frac{\prod_{j=1}^{n_c} A^{\mathcal{C}_j^N}(\mathbf{d}_j)}{\prod_{i=1}^{n_v} (q-1)^{d_i(t_i-1)} \binom{N}{d_i}^{t_i-1}}, \tag{3.4}$$

where $A^{\mathcal{C}_j^N}(\mathbf{d}_j)$ is the weight-vector enumerator of the code $\mathcal{C}_j^N$ induced by the N copies of the constraint node c_j. The elements of $\mathbf{d}_j$ comprise a subset of the elements of $\mathbf{d} = [d_1, d_2, ..., d_{n_v}]$.

Theorem 3. *For a PB NB LDPC code ensemble, let us represent the average number of codewords with symbol weight d by A_d. This quantity is equal to the sum of $A(\mathbf{d})$ over all $\mathbf{d}$ such that $\sum_{i=1}^{n_v} d_i = d$. Then $A_d = \sum A(\mathbf{d})$.*

Example 1. *In this example we calculate the average symbol weight enumerator for regular-$(2, 4)$ code, punctured-$(2, 4)$ code and punctured-$(2, 4)$ code variant in Figure 2.2. Note that the last two protographs are obtained by adding a different accumulator on regular-$(2, 4)$ code.*

The three protographs are first repeated 16 times, 8 times and 8 times respectively to get ensembles. In this way, each ensemble contains 32 transmitted variable nodes, i.e. 32 transmitted symbols (96 bits). The symbol WEs for the three ensembles are shown in Figure 3.1 for codeword symbol weights less than 10.

To further illustrate the enumeration technique, we plot the symbol WEs for the three protographs with code length 80 symbols (240 bits), i.e. $N = 40$ for first code and $N = 20$ for the second and third codes. The results are shown in Figure 3.2, for the symbol weights less than 10. We note that, relative to the regular-$(2, 4)$ code, the punctured code and its variant both have on average fewer low symbol weight codewords, and that the variant has the best distribution of the three codes for small codeword weights. This relative ordering of codes is consistent with the threshold calculations in Figures 2.3, 2.4, and 2.5 for AWGN channel with different modulations.

Example 2. *Continuing on with the baseline regular-$(2, 4)$ protograph repeated $N = 20$ times (40 symbols), in Figure 3.3 we plot the average number of codewords A_d as a function of the field order q. As expected, the average number of codewords increases with q.*

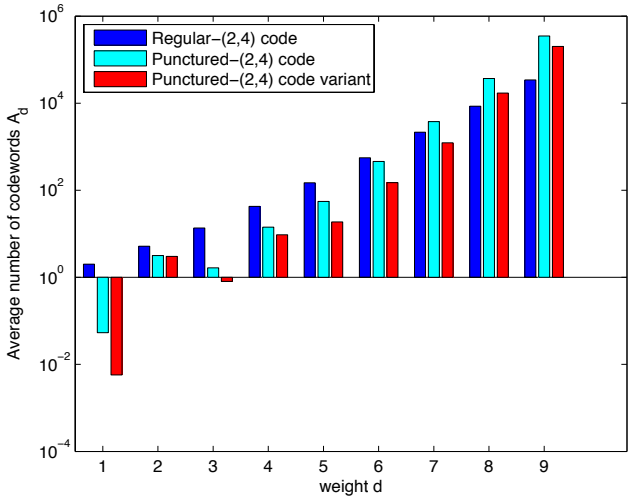

Figure 3.1: WEs of 32-symbol ensembles over GF(8) with random edge weights.

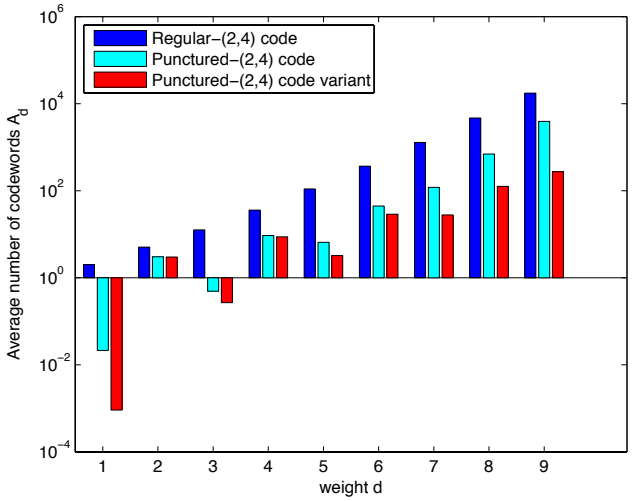

Figure 3.2: WEs of 80-symbol ensembles over GF(8) with random edge weights.

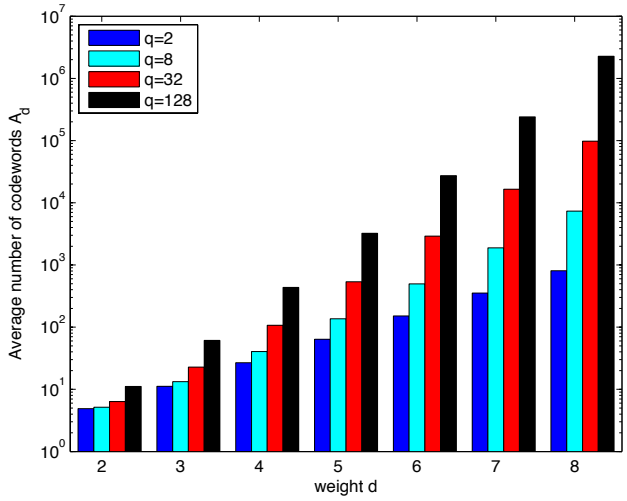

Figure 3.3: WEs for regular-$(2, 4)$ ensembles with different q.

3.2.3 Asymptotic weight enumerator

Given that the formulas in the previous subsection involve the number of copies N, the normalized logarithmic asymptotic weight is defined in [22]

$$r(\delta) = \limsup_{N\to\infty} \frac{\ln A_d}{N}, \tag{3.5}$$

where $\delta = d/N$. Note that $n = n_v \cdot N$, so the asymptotic weight enumerator in terms of n can be expressed as

$$\tilde{r}(\tilde{\delta}) = \limsup_{n\to\infty} \frac{\ln A_d}{n}, \tag{3.6}$$

where $\tilde{r}(\tilde{\delta}) = \frac{1}{n_v} r(\tilde{\delta} n_v)$. Assume t_i is the variable node degree of variable node i. In [22], the authors give an approximation to calculate $r(\delta)$,

$$r(\delta) = \max_{\{\delta_l : v_l \in V\}} \left\{ \sum_{j=1}^{n_c} a^{c_j}(\omega_j) - \sum_{i=1}^{n_v} (t_i - 1)[H_q(\delta_i)] \right\}, \tag{3.7}$$

where $\delta_i = d_i/N$, $\sum_{\{\delta_i : v_i \in V\}} \delta_i = \delta$ and $H_q(\delta_i) \triangleq \delta_i \ln(q-1) + H(\delta_i)$. $H(\delta_i) = -(1-\delta_i) ln(1-\delta_i) - \delta_i ln \delta_i$.

In (3.7), $a^{c_j}(\omega_j)$ is the asymptotic weight-vector enumerator of the constraint node c_j, it can be calculated as

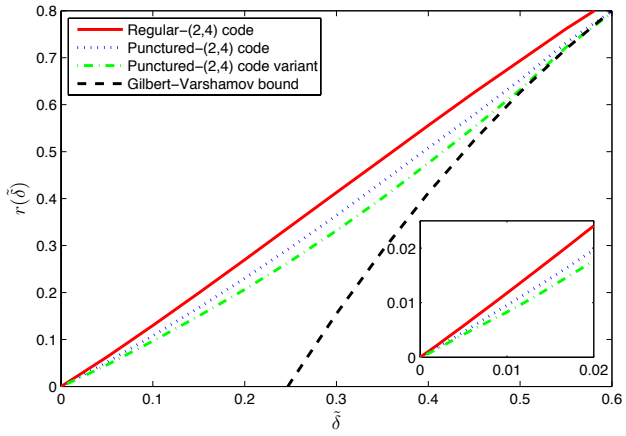

Figure 3.4: Asymptotic symbol weight enumerator with random edge weights.

$$a^c(\omega) = \max_{\{P_\omega\}} \left\{ H(P_\omega) + P_\omega \cdot \mathbf{f}_q^T \right\}, \tag{3.8}$$

under the constraint that $\{P_\omega\}$ is the set of solutions to $\omega = P_\omega \cdot \mathbf{M}_{b,r}^{\mathcal{C}}$, with $p_1, p_2, \ldots, p_{K_r} \geq 0$ and $\sum_{k=1}^{K_r} p_k = 1$.

For the calculation of asymptotic ensemble weight enumerators, notice in equation (3.8), that $H(P_\omega)$ is a concave function and $P_\omega \cdot \mathbf{f}_q^T$ is a linear function. The constraints, $p_1, p_2, \ldots, p_{K_r} \geq 0$ and $\sum_{k=1}^{K_r} p_k = 1$ are also linear. Then, the search for $a^c(\omega)$ becomes a convex optimization problem and can be solved very quickly. In the calculation of equation (3.7), first, we use Monte Carlo method to generate random vectors $(\delta_1, \delta_2, \cdots, \delta_{n_v})$. We then calculate each $a^{c_j}(\omega_j)$ using a convex optimization method to get the value of $\sum_{j=1}^{n_c} a^{c_j}(\omega_j) - \sum_{i=1}^{n_v}(t_i - 1)[H_q(\delta_i)]$. The largest value of all these samples is $r(\delta)$.

Note that the ensemble of all rate-R, q-ary ("random") linear codes (whose parity-check matrix entries are i.i.d. uniform) has the weight enumerator $A^{\mathcal{C}}(w) = (q-1)^w \binom{n}{w} e^{-n(1-R)\ln(q)}$ and the asymptotic weight enumerator [8],

$$\tilde{r}(\tilde{\delta}) = H_q(\tilde{\delta}) - (1-R)\ln(q), \tag{3.9}$$

which corresponds to the asymptotic Gilbert-Varshamov bound for the non-binary case.

Example 3. *Continuing with the protographs discussed in Example 1, we compute the*

asymptotic symbol weight enumerators for the three protographs for q = 8, as shown in Figure 3.7. As we can see, in the asymptotic case, the punctured-(2,4) code and the punctured-(2,4) code variant both have fewer low symbol weight codewords than the regular code. This result is in agreement with the finite length calculation and threshold calculation. Similar to the binary case studied in [19], the asymptotic symbol weight enumerators converge to the Gilbert-Varshamov bound as $\tilde{\delta}$ gets larger. Here, again, of the three candidate protographs, the punctured-$(2, 4)$ variant offers the enumerator closest to the Gilbert-Varshamov bound.

Example 4. *In this example, we provide the asymptotic WE for the $(3, 6)$ regular LDPC code ensemble over $GF(q)$, as shown in Figure 3.5. We also note that our result for $GF(2)$ is in agreement with [19]. From the figure, we can see that as q gets larger, $r(\tilde{\delta})$ tends to be smaller for small $\tilde{\delta}$ and increases faster as $\tilde{\delta}$ grows. Similar to [19], we use $\tilde{\delta}_{min}$ to denote the second zero crossing of $r(\tilde{\delta})$(the first zero crossing is $r(0) = 0$). We recall that the second zero crossing, if it exists, is called the typical relative minimum distance.*

Figure 3.6 shows how the typical relative minimum distance $\tilde{\delta}_{min}$ changes with varying q. Consistent with [26], while $\tilde{\delta}_{min}$ for the Gilbert-Varshamov bound grows monotonically with q, $\tilde{\delta}_{min}$ is in fact non-monotonic. In particular, $\tilde{\delta}_{min}$ attains maximum value for $q = 64$ and 128.

3.3 Enumerator of graph cover non-binary protograph codes

3.3.1 Graph cover non-binary protograph codes

Graph cover non-binary protograph (GC-NBP) code [23] is a special class of non-binary protograph codes. General protograph codes are built from a small Tanner graph called protograph. In the graph cover case, each edge in the protograph has a weight associated with it. The protograph can be represented by $G = (V,C,E,S)$, where V, C, E are the

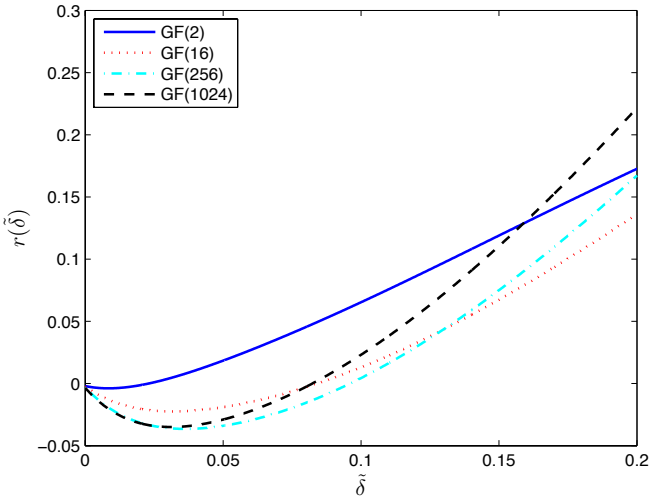

Figure 3.5: Asymptotic symbol weight enumerator of $(3, 6)$ regular code ensemble for different q.

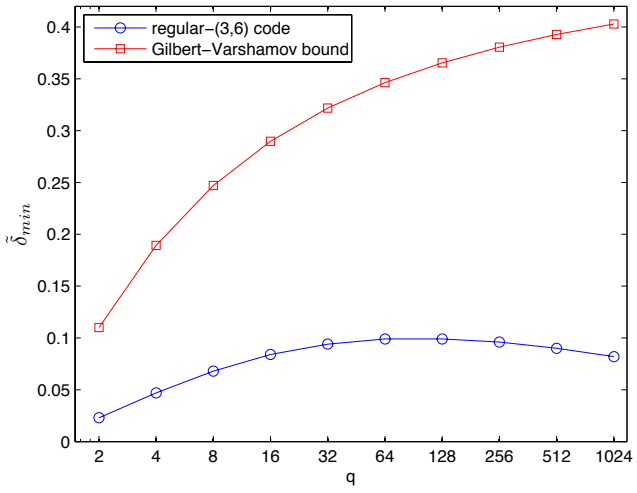

Figure 3.6: Typical minimum distance of $(3, 6)$ regular code ensemble for different q.

sets of variable nodes, check nodes and edges as before and S is the set of weights associated with all the edges. The construction of GC-NBP code is just a copy-permute procedure instead of the general copy-permute-and-scale procedure because the weight of edges are determined by S. Then, we can define the GC-NBP code ensemble as follows.

Definition 4 (GC-NBP code ensemble)**.** *Given a protograph $G = (V, C, E, S)$, we repeat it N times to get a large graph $G^N = (V^N, C^N, E^N, S^N)$. The (G, N, q) GC-NBP ensemble is the collection of codes with all possible permutations of $E_i = \{e_{i,1}, \ldots, e_{i,N}\}$ for every $e_i \in E$.*

3.3.2 GC-NBP ensemble finite length enumerators

The weight enumerator of GC-NBP ensemble is derived in [23]. Consider a degree-m constraint node c in scaled protograph G defined over $GF(q)$. We can view the node c with specified edge weights as a (m_j, $m_j - 1$) single parity check, linear code $\mathcal{C}$ over $GF(q)$. By copying $\mathcal{C}$ N times and permuting, we get an $N-$cover code ensemble C^N. A codeword $\hat{\mathbf{x}}_\mathbf{N}$ of C^N can be represented as $(x_{1,1} \cdots x_{1,N},\ x_{2,1} \cdots x_{2,N},\ \cdots,\ x_{n_v,1} \cdots x_{n_v,N})$. For each codeword, the weight can be represented by a $(q-1) \times n_v$ matrix $\mathbf{d} = \mathbf{d}(\hat{\mathbf{x}}_N)$. Entry $\mathbf{d}(\ell, i)$ of matrix $\mathbf{d}$ is equal to the number of occurrences of symbol ℓ in positions $x_{i,k}$, $1 \leq k \leq N$. The weight enumerator of GC-NBP ensemble C^N is given in [23].

Theorem 4. *The frequency weight matrix enumerator $A^{\mathcal{C}^N}(\mathbf{d})$ of $\mathcal{C}^N$ is given by,*

$$A^{\mathcal{C}^N}(\mathbf{d}) = \sum_{\{\mathbf{n}\}} C\left(N; n_1, n_2, \ldots, n_K\right), \tag{3.10}$$

where $C\left(N; n_1, n_2, \ldots, n_K\right)$ is the multinomial coefficient and $\{\mathbf{n}\}$ is the set of integer-vector solutions to $\mathbf{d} = \mathbf{n} \cdot \mathbf{M}_b^{\mathcal{C}}$, with $n_1, n_2, \ldots, n_K \geq 0$, and $\sum_{k=1}^{K} n_k = N$, and n_k the number of occurrences of the k^{th} codeword among these N copies of c.

Theorem 5. *Let $A^{\mathcal{C}_j^N}(\mathbf{d}_j)$ be the frequency weight matrix enumerator of the code $\mathcal{C}_j^N$ induced by the N copies of the constraint node c_j with the associated scaling $\mathbf{s}_j$. Then, the frequency weight matrix enumerator of the GC-NBP ensemble is*

$$A(\mathbf{d}) \;=\; \frac{\prod_{j=1}^{n_c} A^{\mathcal{C}_j^N}(\mathbf{d}_j)}{\prod_{i=1}^{n_v} C(N;\, d_{i,0},\, d_{i,1},\, \ldots,\, d_{i,(q-1)})^{t_i-1}}. \tag{3.11}$$

In binary channels, the weight of a codeword is the weight of its binary image. With (3.11), we can get the weight enumerator of binary image of GC-NBP ensembles as,

$$A_d = \sum_{\{\mathbf{n}\}} A(\mathbf{d}), \tag{3.12}$$

where n is the set that $d_b(\mathbf{n}) = d$.

Example 5. *Let us consider the binary image of a LDPC code over $GF(q)$. We want to choose good edge weighs to maximize the minimum distance of the binary image. In [24, 25], the authors proposed a method to choose good rows of H. Each row is treated as a linear code with single check node. The minimum distance of the binary image of this code is denoted by d_{min}. The number of codewords with minimum distance is denoted by $W(d_{min})$. The author chose good rows with large d_{min} and small $W(d_{min})$. We also design codes based on this rule in Section. In this example, we use GC-NBP ensemble finite length enumerators to check the effectiveness of this approach.*

Let's consider the regular-$(2, 4)$ code in Figure 2.2 over $GF(16)$ and set $N = 4$. In [25], the author specified $(\alpha^0, \alpha^3, \alpha^7, \alpha^{11})$ as good weights of a degree 4 check node over $GF(16)$ with $d_{min} = 2$, $W(d_{min}) = 1$. We compare GC-NBP ensemble enumerators with this good edge weights and random generated edge weights $(\alpha^6, \alpha^7, \alpha^9, \alpha^{10})$ and $(\alpha^1, \alpha^2, \alpha^7, \alpha^{14})$. As we can see, the GC-NBP with good edge weights has fewer low weight codewords.

3.3.3 GC-NBP ensemble asymptotic weight enumerators

Equipped with the new weight enumerator, the authors in [23] also derived the asymptotic weight enumerator when N tends to infinity,

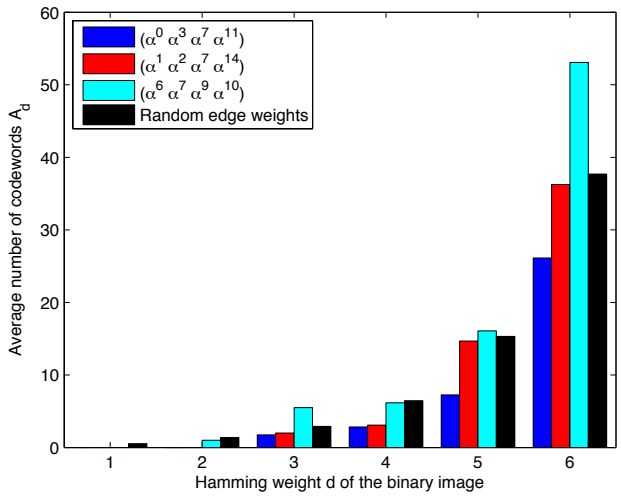

Figure 3.7: Weight enumerator for the binary image of GC-NBP LDPC code ensemble.

$$r_H(\delta) = \limsup_{N\to\infty} \frac{\ln A_{d_H}}{N}, \tag{3.13}$$

or in terms of the codeword length n (where $n = n_v \cdot N$)

$$\tilde{r_H}(\tilde{\delta}) = \limsup_{n\to\infty} \frac{\ln A_{d_H}}{n}, \tag{3.14}$$

where $\tilde{r_H}(\tilde{\delta}) = \frac{1}{n_v} r_H(\tilde{\delta} n_v)$. $r_H(\delta)$ can be calculated in a similar way,

$$\begin{aligned} r_H(\delta) = \quad & \max_{\{\boldsymbol{\delta}\}} \Big\{ \textstyle\sum_{j=1}^{n_c} a^{\mathcal{C}_j}(\boldsymbol{\delta}_j) \\ & - \textstyle\sum_{i=1}^{n_v} (t_i - 1) H\left([\delta_{i,0},\ \delta_{i,1},\ \ldots,\ \delta_{i,(q-1)}] \right) \Big\}. \end{aligned} \tag{3.15}$$

The term $a^{\mathcal{C}_j}(\boldsymbol{\delta}_j)$ stands for the asymptotic frequency weight matrix enumerator of the constraint node c_j, and it is computed as $a^{\mathcal{C}_j^{\infty}}(\boldsymbol{\delta}_j) = \max_{\{P_{\boldsymbol{\delta}_j}\}} \left\{ H(P_{\boldsymbol{\delta}_j}) \right\}$. The collection $\{P_{\boldsymbol{\delta}_j}\}$ represents the set of solutions to $\boldsymbol{\delta}_j = P_{\boldsymbol{\delta}_j} \cdot M_b^{\mathcal{C}_j}$, with $P_{\boldsymbol{\delta}_j} = [p_1, p_2, \ldots, p_K]$, $p_1,\ p_2,\ \ldots,\ p_K \geq 0$ and $\sum_{k=1}^{K} p_k = 1$.

The asymptotic weight enumerators with parameters same as in Example 5 are also calculated. The result shows that A_d of good edge weights assignments is only about 2% smaller than random edge weights protographs.

3.3.4 Complexity of computing weight enumerators of PB NB and GC-NBP codes

Both the finite and asymptotic enumerators of PB NB and GC-NBP are based on enumerator of a linear code with single check node. Let's compare the computation complexity of this basic code for PB NB and GC-NBP case. Assume the single check node is degree 4 over $GF(16)$. For the finite case, assume the check node is repeated N times.

(a) For the PB NB case, $\mathbf{w} = \mathbf{n} \cdot \mathbf{M}_{b,r}^{\mathcal{C}}$, the dimension of matrix $\mathbf{M}_{b,r}^{\mathcal{C}_j}$ is $K_{j,r} \times m_j$, where $K_{j,r} = 1 + \sum_{i=2}^{m_j} \binom{m_j}{i} = 12$ and $m_j = 4$. For the finite case, there are $\binom{N+K_{j,r}-1}{K_{j,r}-1} = \binom{N+11}{11}$ possible choices of $\mathbf{n}$. We need to enumerate all possible $\mathbf{n}$'s to get the symbol weight enumerator. For the asymptotic case, with fixed δ, we need to find the length m_j vector $\omega = [\delta_1, \delta_2, \delta_3, \delta_4]$ under the constraint $\sum_{i=1}^{m_j} \delta_i = \delta$ to maximize $a^c(\omega)$. This is a search in dimension 4 space.

(b) For the GC-NBP case, the dimension of matrix $\mathbf{M}_b^{\mathcal{C}_j}$ is $K_j \times m_j(q-1)$, while $K_j = q^{(m_j-1)} = 16^3 = 4096$ and $m_j(q-1) = 60$. For the finite case, there are $\binom{N+K_j-1}{K_j-1} = \binom{N+4095}{4095}$ possible choices of $\mathbf{n}$. The possible choices of $\mathbf{n}$ are much larger than the PB NB case, which makes the enumeration much more difficult. For the asymptotic case, with fixed δ, we need to find the length $m_j(q-1) = 60$ vector $\boldsymbol{\delta}$= $[\delta_1, \delta_2, \ldots, \delta_{59}, \delta_{60}]$ under constraints to maximize $a^c(\boldsymbol{\delta})$. This is a search in dimension 60 space. The dimension is much higher than the PB NB case, so the computation complexity is also higher.

3.4 Enumerator for stopping sets

In the iterative decoding of LDPC codes over binary erasure channel, an (a, b) stopping set $\mathcal{S}(a, b)$ is a set of a variable nodes and b check nodes such that all the a check nodes are connected to at least two of the b variable nodes. If all of the b variable nodes of $\mathcal{S}(a, b)$ are erased, the iterative decoder will be stuck. The decoding procedure of binary and non-binary LDPC codes over binary erasure channel are very similar and

the definition of stopping set does not depend on q. In this part, we enumerate the structure of stopping set $\mathcal{S}(a, b)$ and do not consider the order of the Galois Field.

The way to enumerate stopping sets is derived in [19], [22]. The ensemble stopping set enumerator of a protograph G is the same as weight enumerator of a protograph G' which formulated by adding a new $degree - 1$ variable node to each check node in G. The modification of the stopping set check nodes - when these are viewed as individual binary codes is expressed by appending the column $[0, 1, 1, \cdots, 1]^T$ to the stopping set constraint matrix of the check nodes in G. This matrix determines the parameter a, and together with the support of added degree- 1 nodes (corresponding to the appended column) determines the parameter b in $\mathcal{S}(a, b)$. All non-binary scale values on each edge are converted to 1.

Example 6. *Let us consider the (3,6) protograph code again. With the method described in this section, the enumerator for stopping sets is similar to the weight enumerator for binary protograph LDPC codes. In Figure 3.8, $\mathcal{S}(a, \Delta \cdot a)$ is evaluated for several values of Δ. For (3,6) protograph code, each variable node is connected to three check nodes and each check node is connected to six variable nodes. Thus, for a erasure variable nodes, there are $3a$ edges connected to erasure variable nodes and $\frac{3a}{6} \leq b \leq \frac{3a}{2}$, i.e., $0.5 \leq \Delta \leq 1.5$. From Figure 3.8, we can see that with fixed a, there tends to be more stopping sets for larger Δ.*

3.5 Enumerator for trapping sets

An (a, b) trapping set $\mathcal{T}(a, b)$ is a set of a variable nodes and b check nodes. Among all the variable nodes in a Tanner graph, a variable nodes are non-zero and b check nodes are not satisfied. The way to enumerate trapping sets is similar to the weight enumerator and is derived in [19], [22].

The ensemble trapping set enumerator of a protograph G is the same as weight enumerator of a protograph G' which is formulated by adding a new $degree-1$ variable node to each check node in G. After the same repeat-permute-and-scale procedure, we

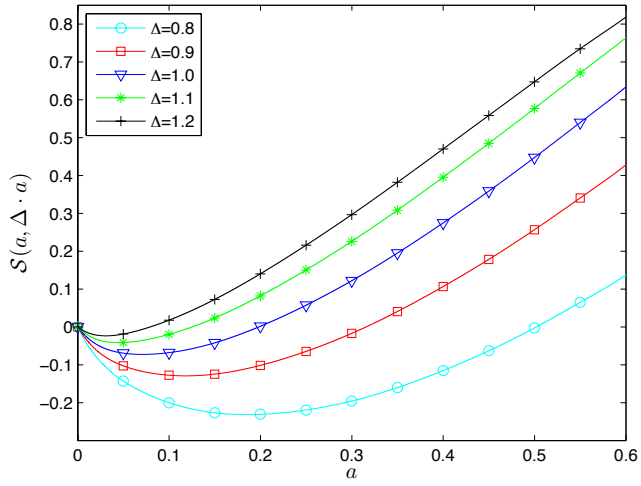

Figure 3.8: Asymptotic stopping set enumerators of regular-$(3, 6)$ protograph ensemble.

get a Tanner graph G_L from G and $G_L{}'$ from G'. Let us use V_n to represent the set of all new degree-1 variable nodes in $G^{L\prime}$ and use V_o to represent the set of all other variable nodes. Then an (a, b) trapping set in G_L is the same as a weight $(a + b)$ codeword in $G_L{}'$, where a non-zero variable nodes are in the set V_o and b non-zero variable nodes are in the set V_n. Let us define the (a, b) trapping set enumerator as $A_{a,b}$. Similar to the WE case, we define the normalized logarithmic asymptotic trapping set enumerator $\tilde{r}(\tilde{\alpha}, \tilde{\beta})$, as

$$\tilde{r}(\tilde{\alpha}, \tilde{\beta}) = \limsup_{n\to\infty} \frac{\ln A_{a,b}}{n}, \tag{3.16}$$

where $\tilde{\alpha} = a/n$ and $\tilde{\beta} = b/n$.

Example 7. *Let us consider regular-(3,6) protograph code ensemble. The asymptotic trapping set enumerators are plotted for different $\tilde{\beta}$ with $q = 16$ in Figure 3.9. Note when $\tilde{\beta} = 0$, by our definition, the curve corresponds to the asymptotic symbol weight enumerator. In the figure, when $\tilde{\alpha}$ is fixed, $\tilde{r}(\tilde{\alpha}, \tilde{\beta})$ increases with increasing $\tilde{\beta}$. This is the same as trapping set enumerator for binary protograph-based LDPC codes in [19].*

Example 8. *In this example, we consider regular-(3,6) protograph ensemble for different q's with fixed $\tilde{\beta} = 0.0002$. The simulation result is shown in Figure 3.10. In the figure, we can see that when $\tilde{\beta} = 0.0002$, there exist second zero-crossings, i.e. there exist typical relative $\tilde{r}(\tilde{\alpha}, 0.0002)$ trapping sets for different q's. Also, when $\tilde{\alpha}$ is fixed,*

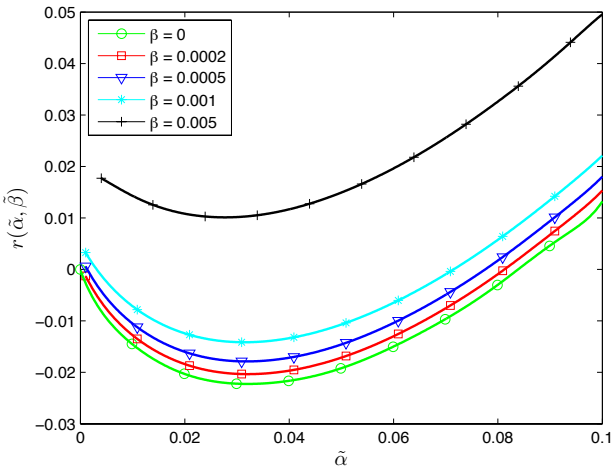

Figure 3.9: Asymptotic trapping set enumerators of regular-$(3, 6)$ code ensemble for $q = 16$.

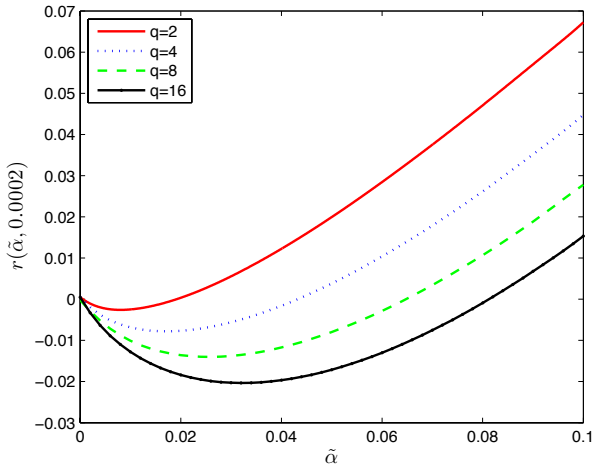

Figure 3.10: Asymptotic trapping set enumerators of the regular-$(3, 6)$ code ensemble for different q with $\tilde{\beta} = 0.0002$.

$\tilde{r}(\tilde{\alpha}, 0.0002)$ is smaller for larger q's. This indicates that for $\tilde{\beta} = 0.0002$, codes over larger q will have less trapping sets.

Chapter 4

Conclusion

In this thesis, the non-binary PEXIT is presented and thresholds of different protographs over GF(q) are calculated. In the calculation, regular-$(2, 4)$ protograph is shown to have lowest threshold for large q. Using this protograph, we designed several codes over GF(256) with rate 1/2 and length 128, 256 and 512 in bits with much smaller frame error rate than their binary counterparts. With puncturing of the regular-$(2, 4)$ protograph, we also designed several codes over GF(256) with rate 2/3 and length 96, 192 and 384 in bits which also show much lower frame error rate than their binary counterparts.

We also give examples of weight enumerator for protograph-based NBLDPC code ensemble both for finite and asymptotic case and compare the asymptotic weight enumerator with Gilbert-Varshamov bound. Weight enumerator is very useful to predict the frame error rate of linear codes with maximum likelihood decoder. Moreover, using similar method, the trapping set and stopping set enumerator are presented which help describe the error performance of LDPC codes using a belief propagation decoder.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication,"*The Bell Systems Technical Journal*, vol. 27, pp. 379 - 423, 623 - 656, Oct. 1948.

[2] R.W. Hamming, "Error detecting and error correcting codes," *The Bell Systems Technical Journal*, vol. 29, no. 2, pp. 147 - 150, Apr. 1950.

[3] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, pp. 68 - 79, Mar. 1960.

[4] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, pp. 300 - 304, Jun. 1960.

[5] P. Elias, "Coding for noisy channels," Proc. IRE Conv. Rec., part 4 3746, 1955.

[6] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," IEEE Trans. on Inform. Theory, Apr. 1967.

[7] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding," in *Proceedings of the IEEE International Conference on Communications*, pp. 1064 - 1070, May 1993.

[8] R. G. Gallager. *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[9] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronic Letters*, vol. 32, p. 1645, Aug. 1996.

[10] R. M. Tanner, “A recursive approach to low complexity codes,” *IEEE Transactions on Information Theory*, vol. 27, pp. 533 - 547, Sep. 1981.

[11] J. Pearl, ‘Reverend Bayes on inference engines: A distributed hierarchical approach,” *Proceedings of the Second National Conference on Artificial Intelligence*, 1982.

[12] J. Thorpe,”Low-density parity-check (LDPC) codes constructed from protographs,” IPN Progress Report, Aug. 2003.

[13] S. Brink, G. Kramer, A. Ashikhmin “Design of low-density parity-check codes for modulation and detection”, IEEE Trans. on Comm., Apr. 2004.

[14] T.Richardson and R.Urbanke, *The capacity of low-density parity-check codes under message-passing decoding*, IEEE Trans. Inform. Theory, Feb. 2001.

[15] A. Bennatan, D. Burshtein, *Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels*, IEEE Trans. Inform. Theory, Feb. 2006.

[16] G. Liva, M. Chiani, *Protograph LDPC codes design based on EXIT analysis*, Global Telecommunications Conference, 2007.

[17] X.-Y. Hu and D.M. Arnold, *Regular and irregular progressive edge-growth tanner graphs*, IEEE Tran. Inf. Theory, Jan. 2005.

[18] A. Abbasfar, D. Divsalar, K. Yao, *Accumulate-repeat-accumulate codes*, IEEE Trans. Comm., Apr. 2007.

[19] S. Abu-Surra, D. Divsalar, W. E. Ryan “Enumerators for protograph-based ensembles of LDPC and generalized LDPC codes,”*IEEE Trans.*, Feb. 2011.

[20] B. Chang, D. Divsalar, L. Dolecek “Non-binary protograph-based LDPC codes for short block-lengths”

[21] B. Chang, L. Dolecek, D. Divsalar “EXIT chart analysis and design of non-binary protograph-based LDPC codes”

[22] D. Divsalar, L. Dolecek "Enumerators for Protograph-Based Ensembles of Non-binary LDPC Codes," 2011 ISIT.

[23] D. Divsalar, L. Dolecek "Graph cover ensembles of non-binary protograph LDPC codes," 2012 ISIT.

[24] M.C.Davey, D.MacKay,"Low-density parity-check codes over $GF(q)$", IEEE Comm. Lett., June 1998.

[25] C. Poulliat, M. Fossorier, D. Declercq,"Design of regular $(2, d_c)$-LDPC codes over $GF(q)$ using their binary images," IEEE Trans. Comm., Oct. 2008.

[26] D. Divsalar, L. Dolecek, "On the typical minimum distance of protograph-based non-binary LDPC codes,"*ITA*, Feb, 2012.

[27] F. Sun and H. van Tilborg, "Approaching Capacity by Equiprobable Signaling on the Gaussian Channel," *Trans. on Info. Theory*, vol. 39, no. 5, pp. 1714-1716, Sep. 1993.

[28] M. Shin, J. Kim, H. Song, "Generalizaion of Tanner's minimum distance bounds for LDPC codes," *IEEE Communication Leters*, March, 2005.

[29] J. Ha, J. Kim, D. Klinc, S. McLaughlin, "Rate-compatible punctured low-density parity-check codes with short block lengths," *Trans. on Info. Theory*, February 2006.