Finite-Length and Asymptotic Analysis and Design of LDPC Codes for Binary Erasure and Fading Channels

by

Kaiann L. Fu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2007

Doctoral Committee:
Associate Professor Achilleas Anastasopoulos, Chair
Professor Peter L. Duren
Professor Wayne E. Stark
Assistant Professor Sandeep P. Sadanandarao

To my parents, family, and Michael

ACKNOWLEDGEMENTS

I would like to thank the many people whose support has made the completion of my Ph.D. possible.

First, I would like to thank the National Science Foundation, National Aeronautics and Space Administration, Horace H. Rackham School of Graduate Studies, and the Electrical Engineering and Computer Science Department for providing the funding for my Ph.D. education.

My advisor, Prof. Achilleas Anastasopoulos, has been invaluable in my journey through graduate school. I have gained much knowledge from his expertise and experience, and his guidance and support have greatly helped me to develop my research skills. An excellent advisor, he provided direction as needed while pushing me to think more independently. He also encouraged me to challenge myself, contributing to my professional and personal growth. Further, he has always been patient and helpful throughout my many years working with him. I am very grateful to have had such a great advisor and it has truly been an invaluable experience working closely with him.

I would also like to thank the other members of my Ph.D. committee, Professors Wayne Stark, Sandeep Pradhan, and Peter Duren, for dedicating their time and effort to be on my committee and offering suggestions and comments on my research.

I am extremely grateful to have had the opportunity to work with the Information Processing Group at the Jet Propulsion Laboratory as part of my NASA

Graduate Student Researchers Program Fellowship. With their limitless knowledge, expertise, and experience, the group was an invaluable resource, and I learned a tremendous amount from their advice, ideas, comments, and suggestions. In particular, I would like to thank Jon Hamkins, Dariush Divsalar, Sam Dolinar, Kenneth Andrews, Christopher Jones, and Michael Cheng for their advice and many fruitful discussions.

I would also like to thank Sarah Fogal. Discussing research with her always helped me to clarify issues and gain a better understanding of the material. Thank you for many productive as well as fun conversations.

To my parents and siblings, I cannot offer enough thanks for their eternal support. Throughout my life, they have always been there to help and support me in any way they can. I could not be where I am today without their guidance and advice through the years.

Last, but certainly not least, I would like to thank my fiancé, Michael Chang, for his loving encouragement and support. Through my long journey through graduate school, he has helped me to keep my sanity and to keep a healthy perspective on life. When I struggled with my research, he always provided support, encouragement, and belief in me when I needed it. I cannot adequately express in words how grateful I am for everything he has done for me. Thank you so much for everything!

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

APPENDIX

CHAPTER 1

Introduction

Today's world is flooded with communication systems, from systems that communicate between many small, wireless devices to systems that communicate with spacecraft throughout the solar system and beyond. Regardless of the application, the main goal is to transmit data reliably from one place to another, and communications theory has helped to greatly improve the capabilities of communication systems.

A typical digital communication system consists of several components, as shown in Fig. 1.1. At the transmitter, the digital source data is first compressed by the source coder to remove redundancy and then coded by the channel coder for error-correction capabilities. Next, the modulator places the digital data onto a carrier frequency to be sent over the channel, the medium for communication. At the receiver, the process is reversed: the received signal is first demodulated to recover the digital data, then decoded by the channel decoder which can correct some errors introduced by the channel, and finally uncompressed by the source decoder to obtain a reproduction of the original source data.

The ever-increasing demand for higher data rates, higher reliability, and lower complexity in communication systems drives research in all components of commu-

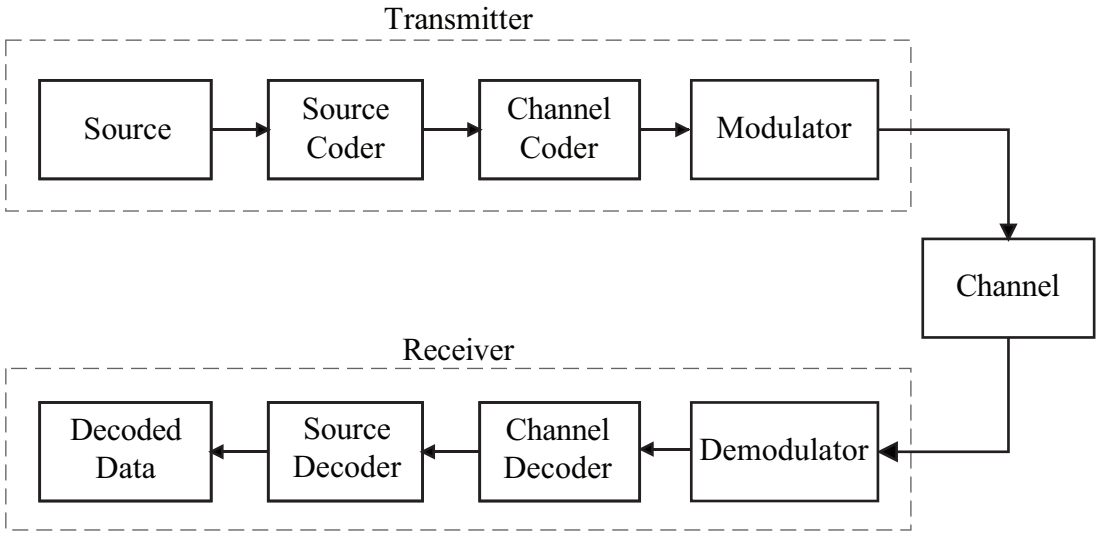


Figure 1.1: A typical digital communication system block diagram.

nication systems, including channel coding, which is the component under investigation in this dissertation. To increase the robustness of a communication system, channel coding strategically adds redundancy to the transmitted data stream such that the receiver can more accurately decode the data it receives, despite any errors or noise introduced by the channel.

In 1948, Shannon developed fundamental limits on the performance of communication systems in his seminal works [5, 6]. For channel coding, Shannon derived the channel capacity, which is the fundamental limit on how much data rate a channel can support with arbitrarily small error probability. While the fundamental limits have been known for many years, only fairly recent work on turbo codes [7] and low-density parity-check (LDPC) codes [8,9] has resulted in practical codes that can come close to capacity. However, these codes only approach capacity in the limit as code-length approaches infinity and only on particular channels. Thus, the channel-coding component still has much room for growth, particularly in the areas of finite-length codes and coding for other channels. Different scenarios present different challenges to channel coding, and this dissertation will address two such challenges.

We first consider communication over a time-selective, frequency-non-selective complex-fading channel where the fading is constant over a block of N symbols, where N is the channel coherence time and is independent from block to block. The fading models a variety of wireless communication channels where the transmitted signal reaches the receiver through multiple paths and thus creates multipath fading. In this case, both the amplitude and phase of the signal are altered by the channel, making it more difficult for the receiver to correctly determine what the transmitter sent. The challenge for communication over this channel is how to recover the data, without knowledge of the channel state, through channel estimation and coding techniques. The phase component of the fading is particularly damaging to the signal because it allows signals to be more easily confused. For example, consider binary phase-shift keying (BPSK) modulation where the signal constellation consists of $+\sqrt{E_s}$ and $-\sqrt{E_s}$ where E_s is the energy per transmitted symbol. If the channel changes the phase of the transmitted signal by 180°, then each symbol is mapped onto the other, and the receiver will decode every symbol incorrectly.

Next, we consider the challenge of designing practical, finite-length codes which can achieve extremely low error rates. Such high performance is desired in many scenarios. For example, in deep-space communications, since signals must travel between Earth and various spacecraft scattered around the solar system and beyond, the signal-to-noise ratios (SNR) are very small, making communication more difficult. The challenge for channel coding is to provide high data rates with extremely low bit-error-rates at small SNR values while still keeping complexity low. Due to the complexity of analysis, we will investigate performance on a simplified channel—the binary erasure channel (BEC), where bits are either received correctly or are erased with probability ϵ. The analysis here will provide insight into code behavior and code design for the BEC and may extend to more realistic channels as well.

To address these two challenges, we will study low-density parity-check (LDPC) codes. LDPC codes in conjunction with iterative decoding based on message-passing algorithms have been shown to achieve excellent performance over a variety of channels, e.g., the binary erasure channel (BEC) [8], the additive white Gaussian noise (AWGN) channel [10, 11], the Rayleigh fading channel where only amplitude is affected by the fading [4, 12, 13], and interference channels [13–15]. In the limit as codelength grows toward infinity, there exist sequences of LDPC codes which approach capacity for the BEC [8] and there exists a code which is within 0.0045 dB of capacity for the AWGN channel [9].

LDPC codes were originally developed by Gallager [16] in the 1960's. However, the true potential of these codes was not realized until the 1990's when turbo codes were introduced [7]. Turbo codes were a major breakthrough in channel coding because their performance could come close to capacity while all existing codes at the time were still several decibels (dB) away from capacity. Further, turbo codes only require low-complexity decoders that utilize iterative-decoding methods. The similarities between turbo codes and LDPC codes led to the rediscovery of LDPC codes. In fact, if the convolutional codes making up the turbo code are terminated to generate a linear block code, then the turbo code can be viewed as a type of LDPC code.

In this dissertation, we first investigate the use of LDPC codes for the time-selective, frequency-non-selective complex-fading channel where both the amplitude and the phase of the transmitted signal are altered by the channel. While LDPC codes have shown their prowess in a variety of applications, including the BEC and AWGN channels, their potential as capacity-achieving codes for more realistic wireless channels has not been established yet. However, experimental evidence as well as some preliminary analytical results [4] have led to the conjecture [17] that LDPC—

or in general, turbo-like codes—can achieve capacity for a wide range of channels. Thus, we analyze and design LDPC codes for the complex-fading channel. To combat the fading, we use a pilot-symbol-assisted scheme where known pilot symbols are added to the LDPC code and used to help estimate the channel at the receiver. Using infinite-length analysis, we investigate several iterative message-passing joint-estimation-and-decoding strategies, optimal energy distribution between pilot and data symbols, and LDPC code design. Several interesting results are obtained regarding unification of analysis and code design.

Next, we investigate the use of LDPC codes for very high reliability, i.e., very low error rates, over the BEC. In the limit as codelength approaches infinity, LDPC codes can approach capacity for the BEC. However, in practical applications, the codelength must be finite and finite-length codes suffer from error floors, which limit the achievable error rate. Error floors arise since the probability of error does not continue to drop dramatically as the erasure probability decreases to small values but instead tends to flatten out to what is known as the error floor, thus, making it very difficult to reach low error rates. To achieve very low error rates desired for high reliability while keeping power requirements low and codelengths relatively short, the error floor must be lowered. We will investigate factors affecting the error floor through finite-length analysis to gain insight into how finite-length LDPC codes can be designed to improve error-floor performance. Previous work has shown that error-floor performance of LDPC codes is determined by stopping sets [1]. Asymptotic analysis of weight and stopping-set enumerators, for codewords and stopping sets which grow linearly with codelength, has aided in designing LDPC codes with lower error floors but does not reflect the behavior of sublinearly-sized stopping sets, which can dominate the iterative-decoding error-floor performance [2, 18–20]. Thus, we provide a perspective on protograph-based and standard LDPC ensemble enu-

merators, based on analysis of stopping sets with *sublinear* growth, which brings new insight into sublinear stopping-set behavior, advantages of protograph structure, and effects of precoding. We show for stopping sets that grow at most logarithmically with codelength, the enumerators follow a polynomial relationship with codelength, unlike the exponential relationship for linearly-growing stopping sets, and this polynomial relationship can be approximately captured by a single parameter. Further, we begin to address the question, "Given *finite* stopping-set sizes and *finite* codelengths, do the stopping sets follow the behavior predicted by linear or sublinear analysis?"

This chapter provides the necessary background for LDPC codes and a brief introduction to our research work. In Chapter 2, the first main topic of performance analysis and code design of LDPC codes for the time-selective, frequency-non-selective complex-fading channel is investigated. Chapter 3 discusses the second main topic of finite-length analysis of LDPC codes for the binary erasure channel. Finally, Chapter 4 concludes the dissertation with a summary of the research and possible areas of future work.

1.1 LDPC Background

In this section, the relevant background on LDPC codes will be provided, including LDPC code structure and representation, decoding algorithms, performance analysis for infinite-length and finite-length code ensembles, and protograph-based structures.

1.1.1 Binary Linear Block Codes

First, we provide a very brief overview of binary linear block codes, since LDPC codes are binary linear block codes. A binary linear block code with codelength n consists of a codebook C which is a linear subspace of $\{0,1\}^n$ with dimension k. At the transmitter, the channel coder uniquely maps information sequences of length k to binary codewords of length n from the codebook C.

A binary linear block code can be described as the kernel of an $(n-k)\times n$ parity-check matrix $\mathbf{H}$ where each row in $\mathbf{H}$ represents a parity check on the codewords. Thus, an $n \times 1$ vector $\mathbf{c}$ is a codeword in C if and only if $\mathbf{Hc} = \mathbf{0}$.

1.1.2 LDPC Code Structure and Representation

An LDPC code is a binary linear block code characterized by a low density of 1's in its parity-check matrix $\mathbf{H}$. A regular (d_v,d_c) LDPC code has exactly d_v 1's in each column and exactly d_c 1's in each row of $\mathbf{H}$. For irregular LDPC codes, the number of 1's in each column or row is not constant, so the number of 1's in a column can vary from column to column and the number of 1's in a row can vary from row to row. These increased degrees of freedom allow irregular codes to achieve improved performance over regular codes.

Factor graphs [21] provide a useful, compact representation of LDPC codes and are particularly useful for describing low-complexity iterative-decoding algorithms for decoding LDPC codes. In a bipartite factor graph, each symbol (or bit) in the codeword is represented by a variable node while each parity-check equation, i.e., each row in $\mathbf{H}$, is represented by a code check node. An edge connects a variable node with a check node if the corresponding code symbol is present in the corresponding parity-check equation. For example, the factor graph for a regular (3,6) LDPC code is shown in Fig. 1.2.

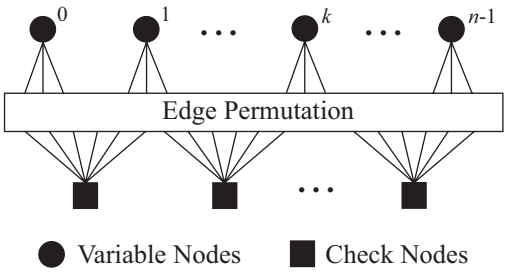


Figure 1.2: Factor graph for a regular $(3, 6)$ LDPC code.

The degree of a node in the LDPC graph denotes the number of edges connected to that node. Thus, for a regular (d_v,d_c) LDPC code, each variable node, which is connected to d_v check nodes via d_v edges, has degree d_v while each check node, which is connected to d_c variable nodes via d_c edges, has degree d_c. Irregular LDPC codes are described by the distribution of degrees across the nodes in the factor graph, and this degree distribution is specified by the degree polynomials

$$\lambda(x) = \sum_{i=1}^{d_v} \lambda_i x^{i-1} \qquad \rho(x) = \sum_{i=1}^{d_c} \rho_i x^{i-1}, \tag{1.1}$$

where λ_i (ρ_i) is the fraction of edges connected to variable (check) nodes with degree i and d_v (d_c) is the maximum degree of a variable (check) node [11]. The rate of an LDPC code is given by $R = 1 - \int_0^1 \rho(x)dx / \int_0^1 \lambda(x)dx$. For regular codes, this expression simplifies to $R = 1 - d_v/d_c$.

The degree polynomials can also be denoted using the node perspective where the fraction of nodes is considered instead of the fraction of edges. Specifically,

$$l(x) = \sum_{i=1}^{d_v} l_i x^i \qquad r(x) = \sum_{i=1}^{d_c} r_i x^i, \tag{1.2}$$

where l_i (r_i) is the fraction of variable (check) nodes with degree i. We will also represent the degree polynomials from the node perspective where the number of

nodes is considered instead of the fraction of nodes:

$$L(x) = \sum_{i=1}^{d_v} L_i x^i \qquad R(x) = \sum_{i=1}^{d_c} R_i x^i, \tag{1.3}$$

where L_i (R_i) is the number of variable (check) nodes with degree i.

The concept of a cycle in the graph is important for developing the infinite-length analysis of an LDPC code. A cycle in the factor graph exists if there is a collection of edges which connect a node back to itself. The girth of a factor graph is the length of the shortest cycle. If the factor graph is cycle-free, then the graph is a tree. These concepts are important in proving the optimality of the decoding algorithm, as will be discussed in Section 1.1.4.

Given degree polynomials $\lambda(x)$ and $\rho(x)$, the standard LDPC ensemble will denote the ensemble of all possible codes (or equivalently, graphs) with the given degree polynomials for a given codelength n. To generate the standard ensemble, first let $e = \frac{n}{\sum_{i=1}^{d_v} \lambda_i / i}$ denote the total number of edges in the graph, and let a variable-node or check-node socket represent a position that an edge can connect to on that particular node, so the total number of sockets associated with a node equals the degree of that node. Next, enumerate all variable-node sockets from 1 to e and all check-node sockets from 1 to e. Then, randomly permute the check-node-socket numbering. Finally, connect variable-node socket i with check-node socket i with an edge for each i from 1 to e. Completing this process, for all possible permutations of the check-node-socket numbering, generates the standard ensemble. Note that in the standard ensemble, there exists the possibility that a single variable node is connected to a single check node by multiple edges.

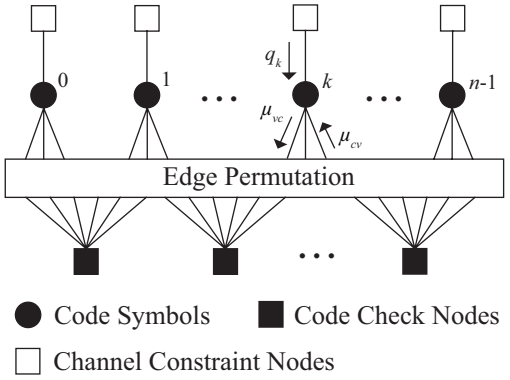


Figure 1.3: Factor graph for a regular $(3, 6)$ LDPC code extended to include channel constraint nodes for a memoryless channel.

1.1.3 Decoding

At the receiver, the optimal decoding strategy that minimizes the probability of codeword or symbol error is maximum-likelihood (ML) decoding, where the receiver chooses the codeword or symbol, respectively, with the highest probability (likelihood) of being sent, given the received signal. However, ML decoding has high complexity since it requires calculating the probability of each codeword or symbol being sent. For LDPC codes, iterative, message-passing algorithms on the LDPC factor graphs provide a more practical, low-complexity solution with faster decoding and without much performance loss despite its suboptimality.

To decode on an LDPC factor graph, the factor graph is first extended to include channel constraint nodes, which represent the information obtained from the received symbols, based on the channel model, about each variable node in the LDPC code. An example of such an extended factor graph is shown in Fig. 1.3 for a memoryless channel, where each received symbol from the channel is only dependent on the corresponding code symbol sent across the channel and is independent of all other received symbols and code symbols.

In this dissertation, we investigate practical, iterative-decoding algorithms, based on message-passing on factor graphs, for LDPC codes. Since decoding LDPC codes

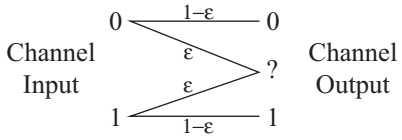


Figure 1.4: The binary erasure channel (BEC).

is simplest for the BEC, the decoding concepts will first be introduced for the BEC and then extended to more realistic channels via the sum-product algorithm, which can be applied to any channel.

Decoding for the BEC

On the BEC, symbols are either received correctly with probability $1-\epsilon$ or they are erased with probability ϵ, i.e., the receiver declares that the symbol is unknown, as shown in Fig. 1.4. The decoding algorithm, which decodes the codewords sent across the BEC, can be viewed as an iterative method of solving the linear system $\mathbf{Hc} = \mathbf{0}$ where $\mathbf{H}$ is the $n(1-R) \times n$ parity-check matrix and $\mathbf{c}$ is the $n \times 1$ codeword for which we are solving.

At the receiver, the codeword $\mathbf{c}$ is partially received: some symbols are erased and all other symbols are received correctly. Let $\mathbf{c}_e$ represent the vector of erased symbols and $\mathbf{H}_e$ be the matrix consisting of the corresponding columns of $\mathbf{H}$. Similarly, let $\mathbf{c}_{ne}$ represent the vector of correctly received symbols and $\mathbf{H}_{ne}$ be the matrix consisting of the corresponding columns of $\mathbf{H}$. Then, the linear system $\mathbf{Hc} = \mathbf{0}$ can be expressed as $\mathbf{H}_e\mathbf{c}_e + \mathbf{H}_{ne}\mathbf{c}_{ne} = \mathbf{0}$, and this equation can be rearranged to obtain

$$\mathbf{H}_e\mathbf{c}_e = \mathbf{H}_{ne}\mathbf{c}_{ne} \triangleq \mathbf{d} \tag{1.4}$$

where $\mathbf{d}$ is a known quantity since all non-erased symbols are known at the receiver.

To solve for $\mathbf{c}_e$, the iterative decoding algorithm first looks for a row i in $\mathbf{H}_e$ with

a single 1 in column j and zeros in all other positions, for some $j \in \{1, \ldots, n\}$. If such an i and j exist, then we can determine the value of the jth symbol of $\mathbf{c}_e$ to be d_j, the jth value of $\mathbf{d}$. Next, the ith row and jth column of $\mathbf{H}_e$ as well as the jth symbol of $\mathbf{c}_e$ can be moved to the known right-hand side of the linear system (1.4), and the process is iteratively repeated until there is no longer a row in $\mathbf{H}_e$ with only a single 1.

A graphical interpretation of this algorithm is shown in Fig. 1.5 where the variable nodes take values from $\{0, 1, ?\}$ where ? represents an erasure. At the beginning of the algorithm, the variable nodes are initialized to the corresponding value (0, 1, or ?) that the receiver received from the channel. This initialization is equivalent to the channel constraint nodes passing their received symbols along edges in the graph to the corresponding code symbols. (The channel constraint nodes are not explicitly shown in Fig. 1.5 to keep the figure simple.) Next, the binary number stored next to each check node is initialized to zero. This number will keep track of the partial binary sum of variable-node values for variable nodes connected to that particular check node. In the first step, the variable nodes pass their value along the edges to the check nodes. Next, the following two steps are iteratively processed.

Check-Node Step: Each check node computes the binary sum of its current stored number and all non-erased incoming messages. The check node now stores this new number. Each of the edges with non-erased messages are now removed from the graph since they have already passed along all the information that they can. This is equivalent to the removal of the jth column of $\mathbf{H}_e$. Now, we look for check nodes which have only one remaining edge connected to it. These are the check nodes which can be resolved. This step is equivalent to looking for a row in $\mathbf{H}_e$ with only a single 1. For each of these singly-connected check nodes, the binary sum stored next to the check node is now passed along the remaining edge.

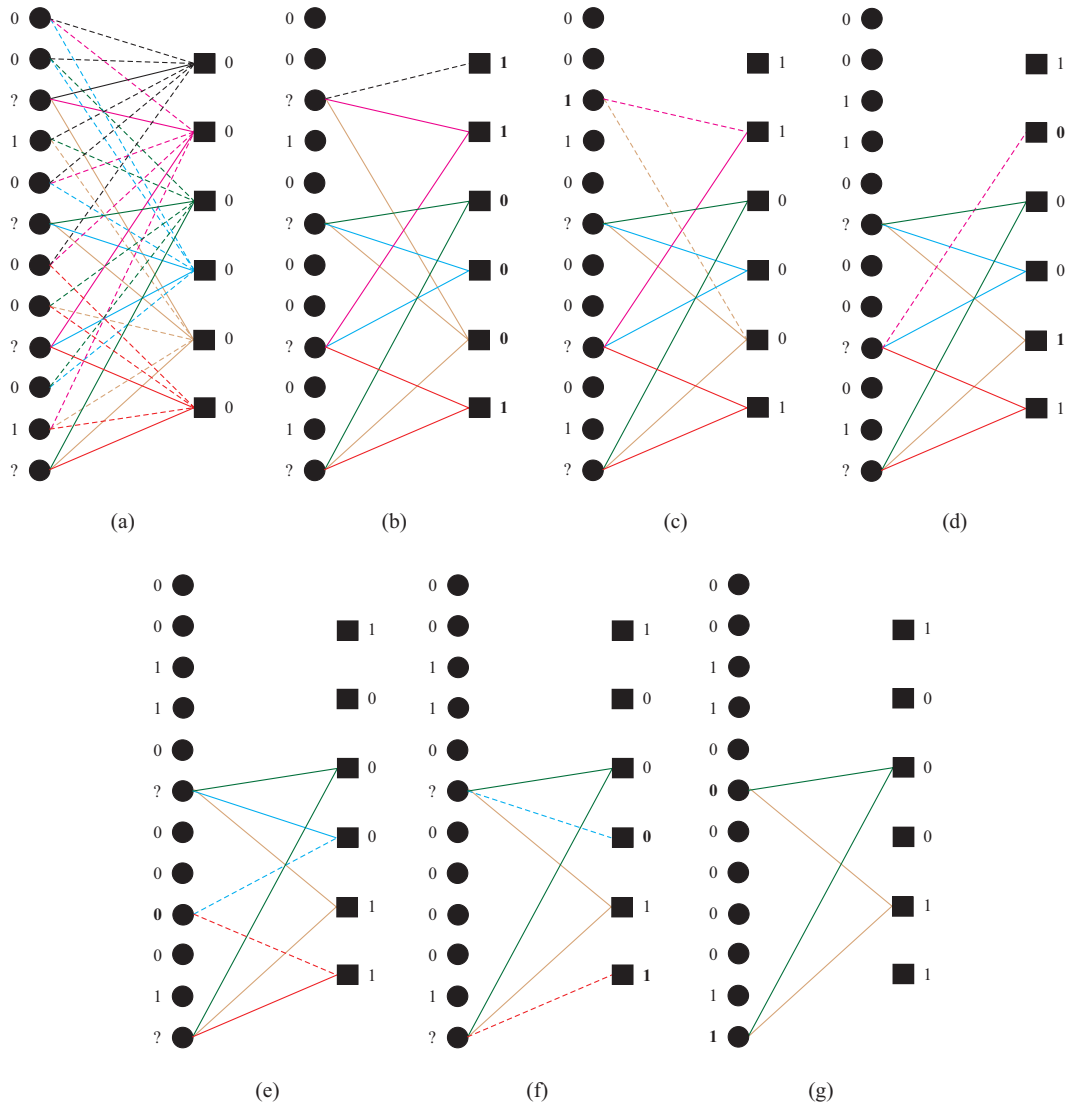


Figure 1.5: An example of iterative decoding on a regular (3,6) LDPC code for the BEC. Dashed edges represent edges on which new values are being passed. Bolded numbers represent numbers whose values have been updated. The LDPC decoding graph is shown (a) after initialization, (b) after the first check-node step, (c) after the first variable-node step, (d) after the second check-node step, (e) after the second variable node step, (f) after the third check-node step, and (g) after the third variable-node step. At this point, all erasures have been resolved, so the receiver has successfully decoded the codeword.

Variable-Node Step: For each variable node, if it is connected to at least one edge which now has a non-erased value, then the value of the variable node is now known and is set to this non-erased value passed along the edge(s) from the check node(s). This step is equivalent to setting the jth symbol of $\mathbf{c}_e$ to d_j. Note that the BEC can only produce erasures, not errors, so messages cannot be conflicting. Now, all edges with non-erased messages are removed from the graph. This step is equivalent to the removal of the ith row of $\mathbf{H}_e$. For each variable node which now has a non-erased value, that value is now passed along all remaining edges connected to that variable node.

The above two steps are iteratively repeated. The algorithm stops when either there are no more edges in the graph or when no more values can be resolved. The former case indicates that all the variable-node values are known, and thus, the receiver has successfully decoded the codeword. The latter case occurs when all the check nodes have either zero or at least two edges remaining. Thus, the algorithm can proceed now further.

An equivalent description of this algorithm is useful for understanding the generalization of this algorithm to the sum-product algorithm, which will be described in the next section. This equivalent description is as follows. The initialization step remains the same.

Check-Node Step: At each check node, the outgoing message on edge e is calculated as follows, for each edge connected to that particular check node. If at least one of the incoming messages to the check node from edges other than edge e is an erasure, then the outgoing message on edge e will also be an erasure. However, if all the incoming messages to the check node from edges other than edge e are not erasures, then the outgoing message on edge e is computed as the binary sum of all the incoming messages from edges other than edge e. Note that this step is simply

calculating the parity check at the check node.

Variable-Node Step: At each variable node, the outgoing message on edge e is calculated as follows, for each edge connected to that particular variable node. If at least one of the incoming messages to the variable node from edges other than edge e is not an erasure, then the value of the variable node is known to be the value of the non-erased incoming message(s), and the outgoing message on edge e takes that value. Note that on the BEC, there are no errors, only erasures, so there will never be a conflict between incoming messages. If all of the incoming messages to the variable node from edges other than edge e are erasures, then the outgoing message on edge e is also an erasure.

The algorithm stops when either all variable-node values are known or when no further progress can be made, i.e., no more erasures can be determined.

If the iterative decoder successfully decodes the entire codeword, as in Fig. 1.5, then the maximum-likelihood decoder will also succeed in decoding the codeword correctly. However, the converse is not true, i.e., there are cases where the iterative decoder fails but the maximum-likelihood decoder succeeds. For example, Fig. 1.6 provides a scenario where the iterative decoder fails to decode three variable nodes. However, by closer inspection, observe that the only possible way to satisfy all check-node constraints is for all three of these variable nodes to take the value 0. Thus, the maximum-likelihood decoder will be able to decode these three variable nodes, and hence decode the entire codeword correctly. This example illustrates the suboptimality of the iterative decoder.

The Sum-Product Algorithm

The sum-product algorithm is a general, low-complexity, iterative-decoding, belief-propagation algorithm described as a message-passing algorithm on a factor graph [22,

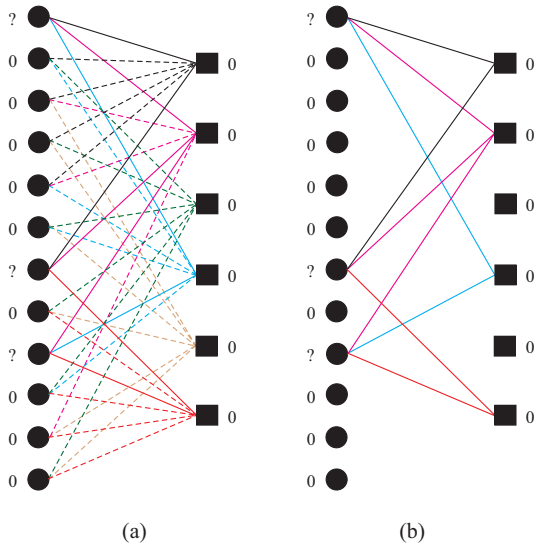


Figure 1.6: An example of iterative decoding on a regular (3,6) LDPC code for the BEC where iterative decoding fails but maximum-likelihood decoding succeeds. When the iterative decoder receives the channel values given in (a), then at the next iteration, the decoder stops at (b) with three variable nodes still erased and can decode no further.

23]. The concept can be implemented for any channel, and for the BEC, it reduces down to the decoding algorithm given above. The sum-product algorithm can be considered a generalization of the BEC decoding algorithm. In this section, we describe a specific application of the sum-product algorithm to LDPC codes.

The sum-product algorithm uses message-passing on the LDPC graph to calculate maximum-likelihood symbol probabilities for each variable node in the LDPC codeword to determine the most likely value of that particular variable node. We describe below how the sum-product algorithm is implemented to decode the kth variable node in the codeword for each $k \in \{1, \ldots, n\}$ where n is the codelength.

First, consider a cycle-free factor graph. Then, the graph can be described as a tree with the kth variable node at the root. The general idea behind the sum-product algorithm is to split the evaluation and marginalization of a global function to local functions which are only dependent on a small neighborhood in the tree. Beginning with the leaf nodes, the local functions are calculated up the tree until the root node

is reached, resulting in the desired marginalized function for the kth variable node.

For LDPC codes, the sum-product algorithm is used to generate symbol probabilities $p(\mathbf{z}, a_k = a)$ for each variable node a_k and each possible symbol value a, where $\mathbf{z}$ is the vector of received symbols. The global function to be marginalized is $p(\mathbf{z}, \mathbf{a})$ where $\mathbf{a}$ is the vector $(a_1, \ldots, a_n)$. This global function can be factored as follows.

$$p(\mathbf{z}, \mathbf{a}) = p(\mathbf{z}|\mathbf{a})p(\mathbf{a}) = \prod_k p(z_k|a_k) \frac{1}{|C|} \prod_j I(j\text{th check node is satisfied}) \tag{1.5}$$

where the second equality follows for memoryless channels and equiprobable codewords, $|C|$ is the size of the codebook C, and $I(\cdot)$ is the indicator function which is used here to represent the parity-check equations of the parity-check matrix. The desired marginalized function is

$$p(\mathbf{z}, a_k = a) = \sum_{\mathbf{a}:a_k=a} p(\mathbf{z}, \mathbf{a}) = \sum_{\mathbf{a}:a_k=a} \prod_k p(z_k|a_k) \frac{1}{|C|} \prod_j I(j\text{th check node is satisfied}). \tag{1.6}$$

Thus, the symbol probability $p(\mathbf{z}, a_k = a)$ is split into functions with dependence only on local areas in the tree.

Fig. 1.7 shows the computation on a tree representing the neighborhood of a_k in the graph. Let $\mathcal{N}(c)$ denote the immediate neighbors of check node c and $\mathcal{N}(c)\backslash v$ denote the immediate neighbors of c excluding variable node v. Beginning at each leaf variable node a_i, $p(z_i|a_i)$ is calculated and sent up the tree to the adjacent check nodes c'. Each check node c' then calculates the following expression for the variable node a_j above c' in the tree.

$$p(\mathbf{z}_j, a_j = a) = \sum_{a_i \in \mathcal{N}(c')\backslash a_j} I(\mathcal{N}(c') \text{ satisfies parity check } c') \prod_{l:a_l \in \mathcal{N}(c')\backslash a_j} p(z_l|a_l) \tag{1.7}$$

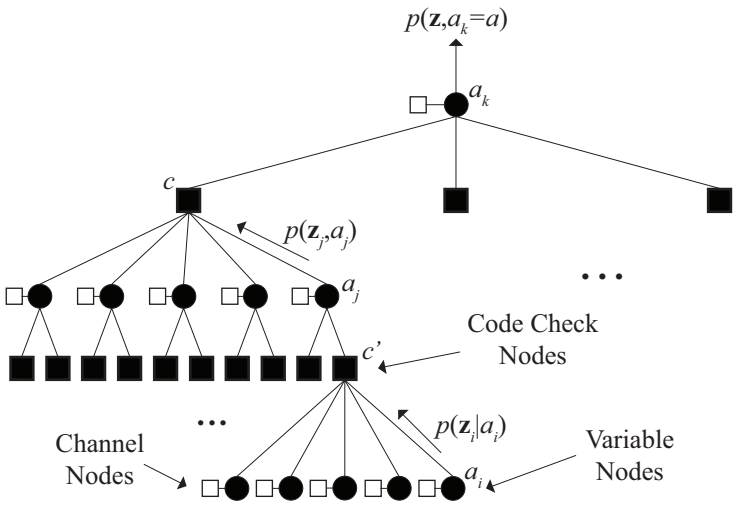


Figure 1.7: Sample tree for calculating $p(\mathbf{z}, a_k = a)$ for a regular $(3, 6)$ LDPC code. Typically, the trees will have greater depth than shown here.

where $\mathbf{z}_j$ represents all received symbols in the tree below a_j. The process is continued until the root node is reached and $p(\mathbf{z}, a_k = a)$ is calculated. Finally, the value for a_k is chosen to maximize $p(\mathbf{z}, a_k = a)$.

Note that the sum-product algorithm does not need separate trees and separate sets of computation to decode each variable node. The messages passed along the edges in the graph appear in the final marginalized function for many variable nodes a_k, so the computed messages can be reused for many $p(\mathbf{z}, a_k = a)$ calculations. By passing messages iteratively through the graph, the algorithm can simultaneously perform computations for all the variable nodes. Specifically, at each iteration, each node sends messages to its neighboring nodes based on the information it has received so far from the rest of the graph. Each additional iteration is equivalent to adding another layer to the computation tree for every variable node. This iterative process is successful because the messages are based on local functions whose contributions are passed along and processed through the graph just as they are passed along and processed up through the computation trees.

If the factor graph is cycle-free, the sum-product algorithm converges after a finite

number of iterations [22]. Thus, at the termination of the algorithm, $p(\mathbf{z}, a_k = a)$ is computed for each variable node a_k and each possible symbol value a. By choosing a_k to maximize $p(\mathbf{z}, a_k = a)$, the result is maximum a posteriori symbol detection. If the graph has cycles, then the sum-product algorithm is suboptimal but still provides a good low-complexity, iterative-decoding solution.

The particular implementation of the sum-product algorithm used in this dissertation is as follows. The message from a variable node to a code check node, the message from a code check node to a variable node, and the message from the channel check node to the kth variable node are denoted by μ_{vc}, μ_{cv}, and q_k respectively, for $k = 1, \ldots, n$, as shown in Fig. 1.3.

Since a_k is binary, instead of sending two messages $p(z_k|a_k = 0)$ and $p(z_k|a_k = 1)$, the message-passing can be simplified by using messages in the form of log-likelihood ratios, e.g., $\log \frac{p(z_k|a_k=0)}{p(z_k|a_k=1)}$. Thus, communication from one node to another can be completed with a single message.

Using messages in the form of log-likelihood ratios, the initial message from the kth channel node to the kth variable node is

$$q_k = \log \frac{P(z_k|a_k = 0)}{P(z_k|a_k = 1)}. \tag{1.8}$$

We can express the evolution of the log-likelihood messages through the factor graph as follows [24]. At the kth variable node v_k with degree d_k, let c_i for $i = 0, \cdots, d_k - 1$ represent the d_k code check nodes which are connected to the kth variable node. Then the outgoing message to the code check node c at the lth iteration is

$$\mu^l_{v_k c} = q_k + \sum_{i=0, c_i \neq c}^{d_k - 1} \mu^{l-1}_{c_i v_k}. \tag{1.9}$$

At the ith code check node c_i with degree d_i, let v_j for $j = 0, \cdots, d_i - 1$ represent the

d_i variable nodes which are connected to the ith code check node. Then the outgoing message to the variable node v at the lth iteration is

$$\mu_{c_i v}^{l} = 2 \tanh^{-1}[\prod_{j=0, v_i \neq v}^{d_i - 1} \tanh \frac{\mu_{v_j c_i}^{l}}{2}]. \tag{1.10}$$

At the lth iteration, the kth variable node can be decoded as follows. Compute

$$Q_k^l = q_k + \sum_{i=0}^{d_k - 1} \mu_{c_i v_k}^{l-1} = \mu_{c v_k}^{l-1} + \mu_{v_k c}^{l} \tag{1.11}$$

for any code check node c connected to v_k. Then, we decode $a_k = 0$ if $Q_k^l > 0$ and $a_k = 1$ if $Q_k^l < 0$.

Note that at each iteration, all the messages to all n variable nodes can be computed simultaneously, and all the messages to all check nodes can be computed simultaneously. Further, at each iteration, the algorithm produces a best-guess value for each of the n variable nodes; separate message-passing algorithms are not necessary to decode each variable node. So, the sum-product algorithm provides an efficient iterative-decoding method for decoding.

1.1.4 Performance Analysis

Analysis of LDPC code performance has shown that LDPC codes can approach capacity on a number of channels. For example, certain sequences of LDPC codes are proven to be capacity approaching on the binary erasure channel (BEC) [8]. Experimental evidence suggests that LDPC codes can also approach capacity on the AWGN channel [10, 11] as well as the Rayleigh fading channel where only amplitude is affected by the fading [4]. This capacity-approaching ability of LDPC codes is an asymptotic property where in the limit as codelength approaches infinity, arbitrarily

low error probability can be achieved.

In this dissertation, we will investigate the performance of LDPC codes with the iterative-decoding schemes discussed in Section 1.1.3. The analysis will be divided into two main categories: infinite-length analysis and finite-length analysis.

In infinite-length analysis, the performance is computed in the limit as the codelength approaches infinity. A method called density evolution, discussed in the next subsection, can be used to calculate this performance. As the codelength approaches infinity, the effect of cycles is eliminated since the probability of a finite cycle existing in the graph approaches zero. Thus, density evolution assumes a cycle-free code. The end result of density evolution is the threshold—the smallest SNR (or equivalently the largest erasure probability for the BEC) such that arbitrarily low error probability can be achieved with a particular LDPC code ensemble.

In all practical scenarios, the codelength must be finite. Thus, to capture the true achievable performance, finite-length analysis is needed. A typical probability of error versus erasure probability curve for iteratively-decoded LDPC ensembles for the BEC is provided in Fig. 1.8. The plot shows that performance of LDPC codes can be divided into two main regions: the threshold region and the error-floor region. In the threshold or waterfall region, seen at higher erasure probabilities, the bit error rate drops off quickly. In the error-floor region, seen at lower erasure probabilities, the bit error rate begins to level off to an error floor even with large codelengths. Empirical evidence shows a trade-off between the level of the error floor and the threshold for LDPC codes [25, 26] and turbo codes [27], but this tradeoff has not been proven to exist.

An upper bound on the maximum-likelihood decoding performance follows similar trends as the iterative-decoding performance, as shown in Fig. 1.9. Assuming this upper bound is fairly tight, Fig. 1.9 shows that the presence of an error floor

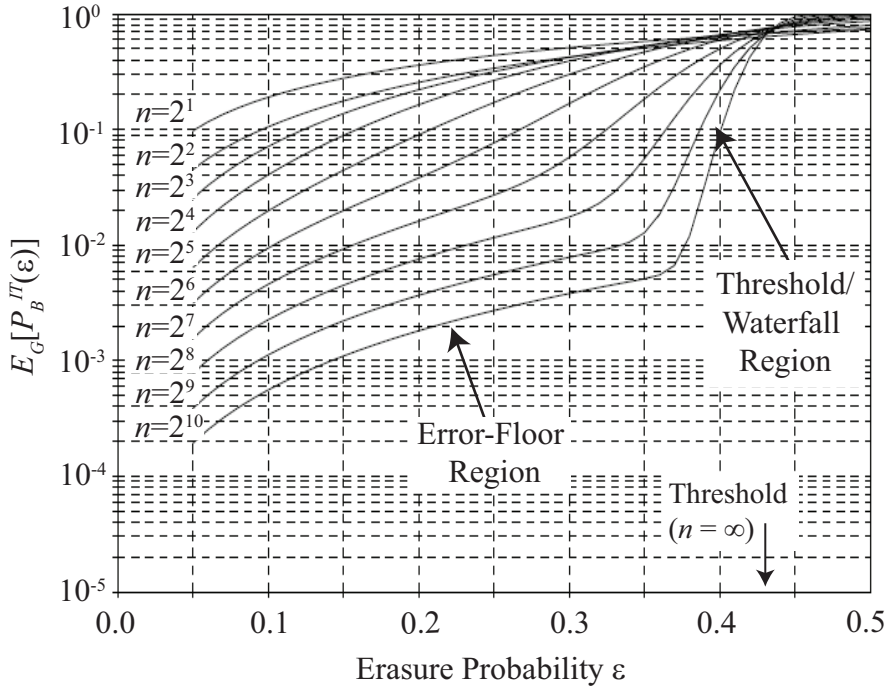


Figure 1.8: Ensemble block-error probability vs. erasure probability for iteratively-decoded regular (3,6) LDPC ensembles with codelength $n \in \{2^i : i = 1, \ldots, 10\}$ for the BEC [1]. In the ensemble block-error probability expression, $E_G[P_B^{IT}(\epsilon)]$, G is a particular LDPC graph realization, the expectation is taken over all possible graphs G in the ensemble, B denotes block-error probability, and IT denotes iterative decoding.

is a result of the structure of the LDPC ensembles and is not introduced by the suboptimal iterative-decoding algorithm. Using iterative decoding does result in performance degradation, but this degradation is fairly small in the error-floor region, and iterative decoding has the advantage of greatly reduced complexity compared to maximum-likelihood decoding. Determining the error-floor performance of these decoders relies on analyzing codewords and weight enumerators for maximum-likelihood decoding and stopping sets and stopping-set enumerators, described in Section 1.1.6, for iterative decoding. In this dissertation, the influence of stopping sets on error-floor performance will be analyzed since stopping sets describe the more practical iterative-decoding algorithm and since more tractable approximations and

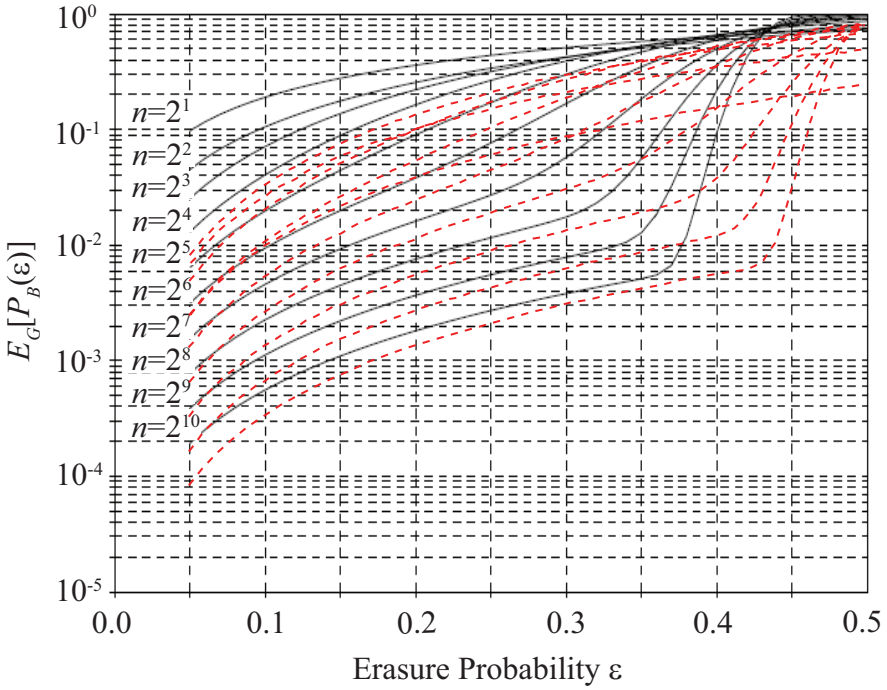


Figure 1.9: Ensemble block-error probability vs. erasure probability for regular (3,6) LDPC ensembles with codelength $n \in \{2^i : i = 1, \ldots, 10\}$ for iterative decoding and maximum-likelihood decoding for the BEC. The solid lines show the iterative-decoding performance while the dashed lines provide an upper bound on the maximum-likelihood decoding performance [1].

analysis can be obtained.

1.1.5 Infinite-Length Analysis with Density Evolution

For iterative decoding schemes, density evolution provides a practical method for analyzing the performance of regular and irregular LDPC code ensembles [10]. Analysis using density evolution involves evaluating the probability density functions (pdfs) of the messages exchanged between the nodes of the factor graph, given the pdfs of the initial messages.

Due to symmetry in the channel and in the message-passing algorithm, we can assume, in the performance analysis, that the all-zero codeword is transmitted. Thus, the pdfs of the messages from the channel constraint nodes are calculated using this

assumption. For example, consider the AWGN channel where the kth channel output is

$$y_k = \sqrt{E_s}(-1)^{x_k} + n \tag{1.12}$$

where E_s is the transmitted symbol energy, $x_k \in \{0, 1\}$ is the kth channel input, and n is the Gaussian noise with variance σ^2. The initial log-likelihood message for the $k$the code symbol is

$$q_k = \log \frac{P(y_k|x_k = 0)}{P(y_k|x_k = 1)} = \frac{2\sqrt{E_s}}{\sigma^2} y_k. \tag{1.13}$$

Thus, the initial message pdf from the kth channel constraint node to the kth variable node, given that the all-zero codeword was transmitted, is

$$f(q_k) = \frac{1}{\sqrt{2\pi\sigma_{q_k}^2}} \exp\left\{ -\frac{1}{2\sigma_{q_k}^2} (q_k - \frac{2E_s}{\sigma^2})^2 \right\}, \tag{1.14}$$

i.e., q_k is a Gaussian random variable with mean $2E_s/\sigma^2$ and variance $\sigma_{q_k}^2 = 4E_s/\sigma^2$. Note that the initial message pdf (1.14) is the same pdf for all k; thus, only one pdf is needed to describe all messages from channel constraint nodes to variable nodes.

Once the pdf of the messages from the channel constraint nodes, which is also the pdf of the initial outgoing messages from the variable nodes to the check nodes, is calculated, then standard pdf transformations described in [10] can be used to iteratively trace the pdfs of the messages as they are exchanged back and forth between the variable and check nodes. Note that the message pdf from the channel constraint nodes does not change with iterations. At each iteration, the density-evolution algorithm computes the average message pdf for all of the outgoing messages from all of the variable nodes, by averaging over all possible variable-node degrees. Thus, a single, average pdf describes all of the outgoing messages from all of the variable nodes. The same is true for the outgoing messages from the check nodes. Thus, density evolution only requires tracing a single pdf as it evolves through the graph.

Detailed description of this evolution can be found in [10].

Let $f^l(q)$ be the pdf associated with the log-likelihood ratio for a variable node at iteration l. For a given SNR, density evolution is used to determine if the error probability at iteration l, calculated as $\int_{-\infty}^{0} f^l(q)dq$, decreases to arbitrarily small values as l increases. By searching over SNR values and performing density evolution, we find the threshold—the smallest SNR value for which arbitrarily small error probability can be achieved. This threshold value provides a single metric for capturing the infinite-length performance of an LDPC code ensemble.

Under the standard assumptions of large girth (compared to the iteration number), the neighborhood of a graph is essentially a tree. In this case, all messages passed in the factor graph are independent and all calculated pdfs are exact. In the practical situation where cycles are present in the graph, the average behavior of the code converges to the cycle-free case as the length of the code increases [10].

1.1.6 Finite-Length Analysis

Analyzing the finite-length iterative-decoding performance of LDPC codes over the BEC requires investigation of stopping sets [1]:

Definition 1.1. For a given bipartite graph describing an LDPC code, a stopping set S is a subset of the variable nodes such that the check nodes connected to S are connected to S at least twice.

For example, Fig. 1.10 provides examples of how a set of variable nodes can be connected to the check nodes such that a stopping set is or is not formed. Fig. 1.10(a) shows an example of a stopping set while Fig. 1.10(b) is not a stopping set since one check node is connected exactly once to the set of variable nodes.

Defining a maximal stopping set on a subset $\mathcal{S}$ of the variable nodes to be the (unique) largest stopping set within $\mathcal{S}$, it can be shown that iterative decoding on

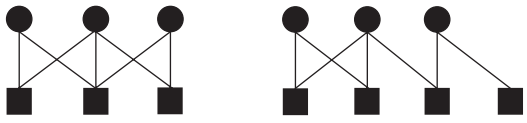


Figure 1.10: Examples showing how a set of variable nodes can be connected to the check nodes such that (a) it is a stopping set and (b) it is not a stopping set.

the graph fails exactly on the maximal stopping set within the set of erased variable nodes [1]. Thus, the stopping sets provide a method of calculating the probability of error of a code.

To calculate the average performance of an LDPC code ensemble, the average number of maximal stopping sets must be calculated and this calculation is largely a combinatorial problem. However, evaluating the number of *maximal* stopping sets requires a recursion, e.g., in [28, 29], which has high complexity—$O(n^3)$ in time and $O(n^2)$ in space. Therefore, an upper bound on performance is determined based on calculating the average number of stopping sets of a given size regardless of whether or not they are maximal.

As shown in [30], the upper bound on performance is found through evaluation of the ensemble stopping-set enumerators given by $s(n, v)$ where n denotes code-length and v denotes stopping-set size. The quantity $s(n, v)$ can be calculated as the expected number of stopping sets of size v in an LDPC ensemble with codelength n. Then, for iteratively-decoded LDPC ensembles, the expected probability of block error resulting from stopping sets of size v over a BEC with erasure probability ϵ can be upper bounded by

$$E_G[P_B^{IT}(n, v, \epsilon)] \leq \epsilon^v s(n, v) \tag{1.15}$$

where G is a particular LDPC graph realization, the expectation is taken over all possible graphs G in the ensemble, B denotes block-error probability, and IT denotes iterative decoding. Based on numerical evaluations in [30], this bound appears to be

quite tight in the error-floor region. Since this bound is much easier to evaluate than the exact expression, it can provide insight into the error-floor behavior of LDPC ensembles.

To see how to arrive at this upper bound, we first condition on the pattern of erased symbols U as follows. Let $V \subseteq U \subseteq W$ be a candidate stopping set, where W is the set of all variable nodes, G is a graph in the ensemble, and s.s. denotes stopping set(s). Then, for a given graph G,

$$\begin{aligned}
P_B^{IT}(n, v, \epsilon) &= \sum_{U \subseteq W} P(U \text{ erased}) P_B^{IT}(n, v, \epsilon | U \text{ erased}) \\
&= \sum_{U \subseteq W} P(U \text{ erased}) P(\text{the maximal s.s. } V \subseteq U \text{ in } G \text{ has size } v) \\
&\leq \sum_{U \subseteq W} P(U \text{ erased}) P(\exists \text{ a s.s. } V \subseteq U \text{ with size } v \text{ in } G) \\
&= \sum_{U \subseteq W} \epsilon^{|U|}(1-\epsilon)^{n-|U|} P\Big(\bigcup_{V \subseteq U : |V| = v} \{V \text{ is a s.s. in } G\} \Big) \\
&\leq \sum_{U \subseteq W} \epsilon^{|U|}(1-\epsilon)^{n-|U|} \sum_{V \subseteq U : |V| = v} P(V \text{ is a s.s. in } G) \\
&= \sum_{V \subseteq W : |V| = v} I(V \text{ is a s.s. in } G) \sum_{U : V \subseteq U \subseteq W} \epsilon^{|U|}(1-\epsilon)^{n-|U|} \qquad (1.16)
\end{aligned}$$

where $I(\cdot)$ is the indicator function. The last equality follows from a change in the order of the summations and since $P(V \text{ is a s.s. in } G)$ only takes values 0 or 1. Taking the expectation over all graphs G in the ensemble, we obtain

$$\begin{aligned}
E_G[P_B^{IT}(n, v, \epsilon)] &\leq E_G \left[\sum_{V \subseteq W : |V| = v} I(V \text{ is a s.s. in } G) \right] \sum_{u=v}^{n} \binom{n-v}{u-v} \epsilon^u (1-\epsilon)^{n-u} \\
&= E_G \left[\# \text{ s.s. with size } v \text{ in } G \right] \sum_{u=v}^{n} \binom{n-v}{u-v} \left(\frac{\epsilon}{1-\epsilon} \right)^u (1-\epsilon)^n \\
&= s(n, v)(1-\epsilon)^n \sum_{i=0}^{n-v} \binom{n-v}{i} \left(\frac{\epsilon}{1-\epsilon} \right)^{i+v} \qquad (1.17)
\end{aligned}$$

where the first expression follows since the second summation in (1.16) is only dependent on the sizes of the sets U, V, and W. Since the summation in (1.17) is a binomial expansion, the right-hand side of (1.17) simplifies as follows:

$$\begin{aligned} E_G[P_B^{IT}(n,v,\epsilon)] &\leq s(n,v)(1-\epsilon)^n \left(\frac{\epsilon}{1-\epsilon}\right)^v \left(1+\frac{\epsilon}{1-\epsilon}\right)^{n-v} \\ &= s(n,v)(1-\epsilon)^n \left(\frac{\epsilon}{1-\epsilon}\right)^v \left(\frac{1}{1-\epsilon}\right)^{n-v} \\ &= s(n,v)\epsilon^v. \end{aligned} \tag{1.18}$$

This derivation shows that the inequality in the bound arises from two main sources: (1) finding existence of all stopping sets of size v rather than finding only maximal stopping sets and (2) the union bound on $\{V \subseteq U : |V| = v\}$.

To see why this bound on block-error probability is useful for investigating error-floor performance, consider the union bound over $\{V \subseteq U : |V| = v\}$. When ϵ is small (a high-reliability channel), the most probable sets U of erased variable nodes are those with small size $|U|$. Since $|U|$ is small, the number of sets $V \subseteq U$ with $|V| = v$ is also small and thus, the union bound is tight. However, when ϵ is large (a low-reliability channel), then the most probable sets U of erased variable nodes are those with large size $|U|$. Since $|U|$ is large, there exists many different sets $V \subseteq U$, with $|V| = v$, with correlated probabilities $P(V \text{ is a s.s.})$. Thus, the union bound becomes loose. This shows that the upper bound in (1.15) is loose at large ϵ but is tighter at low ϵ. Thus, this bound is useful for studying error-floor behavior, which occurs at low ϵ.

For standard ensembles, the stopping set enumerator can be calculated through combinatorics as follows [30]:

$$s(n,v) = \sum_{e=0}^{L'(1)} \frac{\operatorname{coef}\{\prod_{i=1}^{d_v}(1+yx^i)^{L_i}, y^v x^e\} \operatorname{coef}\{\prod_{i=1}^{d_c}[(1+x)^i - ix]^{R_i}, x^e\}}{\binom{L'(1)}{e}} \tag{1.19}$$

where e is the number of edges in the stopping set and coef$\{p(x), x^k\}$ is the coefficient of the x^k term in the polynomial $p(x)$. The first term in the numerator represents the number of ways to choose v variable nodes such that there are exactly e edges emanating from them. The second term in the numerator represents the number of ways to connect e edges to a subset of the check nodes such that a stopping set is formed, i.e., such that all check nodes are connected to at least two of the e edges or to none at all. The denominator represents the number of ways to choose e edges out of the $L'(1)$ total edges in the graph.

1.1.7 Protograph-Based Structure

The introduction of protographs [31] produced low-complexity LDPC-decoder implementations without sacrificing performance. By introducing more structure into the LDPC graph, protographs provide a compact description of the LDPC code, resulting in a savings in the number of gates/transistors needed to implement the graph description in a field-programmable gate-array (FPGA) or other integrated circuit (IC).

A protograph is an LDPC code which typically consists of a small number of nodes. Each variable (check) node in the protograph will denote a variable (check) node type in the final LDPC code. Similarly, every edge in the protograph will denote an edge type in the final LDPC code.

To generate an LDPC ensemble based on a protograph structure, we first replicate the protograph to create a total of Z copies. Consider a particular edge type e which is connected to variable-node type i and check-node type j. We randomly permute the Z connections from the Z type-e edges to the Z type-j check nodes. This creates all possible combinations of connections between variable nodes of type i and check nodes of type j such that each node has one and only one connection.

This permutation is done on all edge types in the protograph. Fig. 1.11 provides an example of this protograph-expansion procedure.

Note that multiple connections between two nodes in the protograph are allowed but do not imply that individual nodes in the final code are connected multiple times. A simple way to guarantee that no multiple connections occur between individual nodes in the final code is to first expand the protograph by a small number Z such that no multiple connections exist in this intermediate protograph. Then, expand the intermediate protograph to the final desired code size.

Protographs enforce structure on an LDPC code, so protograph-based ensembles are a subset of the standard ensemble. Thus, enforcing protograph structure can be viewed as an expurgation technique.

Implementing a decoder for a protograph-based code only requires the storage of the original protograph; the entire LDPC code graph does not need to be stored. This results in a savings of a factor of roughly Z in the storage space needed on the decoder chip (often an FPGA). The permutation of the edges can be stored succinctly when circulant matrices can be used to describe the particular permutations used. Circulant matrices are simply square matrices which can be generated by summing shifted versions of the identity matrix.

To see how circulant matrices can be used to describe a protograph-based LDPC code, consider a protograph with M variable-node types and J check-node types connected via E edge types. The parity-check matrix $\mathbf{H}$ is subdivided into MJ submatrices of size $Z \times Z$ where each submatrix represents the connections between variable nodes of a type i and check nodes of type j for some $i \in \{1, \ldots, M\}$ and $j \in \{1, \ldots, J\}$. For each edge type connecting variable-node type i and check-node type j in the protograph, generate a unique shifted version of the identity matrix. After summing these shifted identity matrices, the result is the submatrix of $\mathbf{H}$

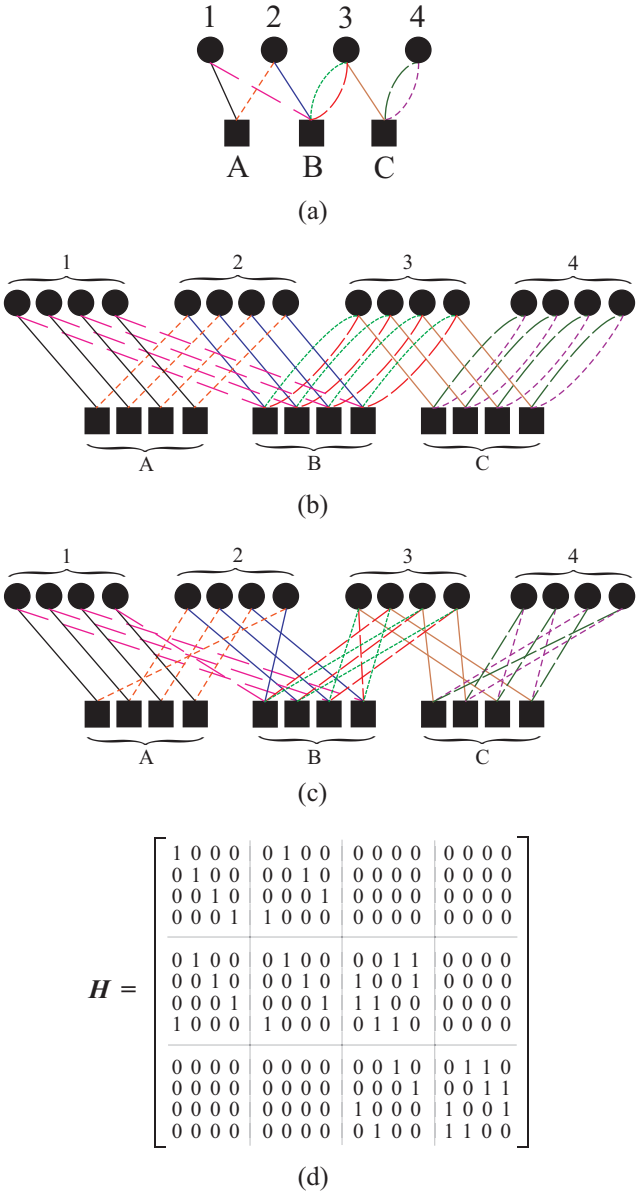


Figure 1.11: Example of a protograph expansion using circulant matrices: (a) the protograph, (b) Z copies of the protograph, (c) the final code generated from (b) through permutation of edges within each edge type, and (d) the parity-check matrix $\mathbf{H}$ for the code represented in (c).

corresponding to connections between type-i and type-j nodes. An example of this type of code generation is given in Fig. 1.11. To represent protograph-based LDPC codes built from circulants, one only needs to know the protograph structure, the number of copies Z in the expansion, and the E shift values corresponding to the amount of shift from an identity matrix for each edge type.

In a decoder implementation, message-passing can be accomplished with Z serial passes through the stored protograph structure. Each of the Z passes corresponds to one of the Z protograph copies in the full LDPC code. Specifically, to complete one set of message transfers from the variable nodes to the check nodes or vice versa (see steps in Section 1.1.3), Z passes through the protograph structure are completed where in each pass, messages are passed in one direction along each edge in one of the Z protograph copies. In other words, message transfer within one protograph copy is computed simultaneously and each protograph copy is processed serially.

Flarion Technologies implements another version of protograph decoding where each edge type is processed serially [32]. All messages traveling in a given direction on edges of a particular type are computed simultaneously, and the decoder moves serially through each edge type. An advantage of this method is that there are no wasted computation cycles since all nodes which are processed simultaneously have the same degree. In the previous method, processing one protograph copy requires waiting for the highest degree node to finish processing before moving on to the next protograph copy.

Experimental evidence suggests that enforcing protograph structure on an LDPC code does not compromise the threshold (infinite-length) or error-floor (finite-length) performance significantly. In fact, the additional structure may help to improve both the threshold and the error floor. This idea will be investigated in this dissertation.

1.2 Dissertation Outline

The research presented in this dissertation consists of two main components: LDPC coding for time-selective complex-fading channels and finite-length analysis for the BEC.

1.2.1 LDPC Codes for Time-Selective Complex-Fading Channels

The model considered in this dissertation is a block complex-fading model where the fading is constant throughout a single block and independent from block to block. This block-independent fading model closely models several wireless communication environments. For example, in a frequency-hopped spread-spectrum modulation scheme, each time the system hops to a different frequency band, a new fading channel realization is encountered. Further, if the fading is slow enough, then we can model the fading as being constant throughout an entire block. Applications include cellular communications and partial-band jamming channels in satellite communications.

Fading arises in wireless communications because the transmitted signal propagates through the medium to the receiver via multiple paths due to effects such as reflection and refraction. Due to differences in path length, the signal arrives at the receiver with a different phase and amplitude for each path. At the receiver, the combination of these multiple paths, each with its own phase delay and amplitude level, results in amplitude and phase changes in the signal, i.e., fading. In the practical case when the channel fading is unknown to the receiver, the fading effects, particularly phase changes, can be quite significant.

In order to combat unknown fading in the channel, we use a simple pilot-symbol-

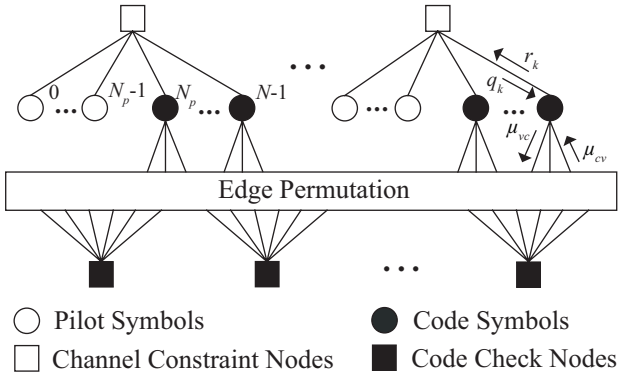


Figure 1.12: Factor graph for a pilot-symbol-assisted LDPC code in a flat, block-independent fading channel.

assisted (PSA) scheme where known pilot symbols are periodically sent by the transmitter to help the receiver estimate the channel. Specifically, in each block of size N, where N is the channel coherence time, known pilot symbols are transmitted in addition to the code symbols from the channel code. The channel codes used in this dissertation are LDPC codes since they can achieve excellent performance even when decoded with low-complexity, iterative decoders. Further, the factor graph representation of LDPC codes along with the iterative-decoding algorithm can easily be extended to include the unknown fading. Thus, the receiver can iteratively perform joint channel estimation and decoding. Fig. 1.12 provides the complete factor graph, including pilot symbols, code symbols from the LDPC code, and channel constraint nodes representing the fading parameters.

Optimal decoding on the factor graph in Fig. 1.12 is obtained by implementing the sum-product algorithm. However, computation at the channel constraint nodes is exponentially complex with N. Thus, we investigate several simpler but suboptimal decoding strategies as well as some ideal decoders which will provide bounds on performance:

- Perfect channel state information (CSI): In this ideal case, the receiver knows the channel state information exactly. Thus, the performance of this decoder

provides a bound on the performance of all other decoders.

- Pilot and data correct decision feedback (PDCDF): This is a hypothetical decoder which cannot actually be implemented but provides bounds on the performance of other decoders. In this decoding algorithm, the channel estimates are calculated from the pilot symbols as well as the coded symbols where the correct values of the coded symbols are provided by a genie. Any practical decoding scheme, including the sum-product algorithm, cannot perform better than the PDCDF decoder.

- Pilot-only (PO): This is a simple, practical algorithm where only the pilot symbols are used to estimate the channel realizations.

- Quantized decision feedback (QDF): This is an ad-hoc, practical algorithm in which channel estimates are calculated based on the pilot symbols as well as some of the code symbols. At each iteration, the code symbols which participate in the channel estimate as those which have a high reliability of being a 0 or a 1. Those code symbols whose values are less certain are not considered in the channel estimation. This algorithm is more complex than the PO algorithm, but it provides improved performance since the channel estimation makes use of additional information provided by the code symbols.

For each of these decoding algorithms, the energy distribution between pilot and code symbols is optimized and the design of optimal LDPC codes is investigated.

Some interesting results, which will be discussed in Chapter 2 but are mentioned briefly here, are obtained regarding unification of analysis and code design in certain scenarios. For the perfect-CSI, PO, and PDCDF receivers, we will show the following:

- Density evolution only needs to be numerically evaluated once over a single parameter b. Then, for any of these receiver types, any channel coherence

time, and any energy allocation, the infinite-length analysis can be directly obtained from the single density-evolution evaluation by mapping channel and receiver parameters to b appropriately.

- Finding the optimal energy distribution between pilot and code symbols only requires a simple closed-form calculation from b.

- The optimal LDPC code for one of these receivers will also be optimal for the other receivers and for any system parameters.

These results significantly simplify the analysis and code design for these three decoder types.

In Chapter 2, density evolution and simulation results are presented for the PSA decoding algorithms using regular and irregular LDPC codes. For the examples provided, we will show the following:

- Optimizing the allocation of power to the pilots and the code symbols results in 1 to 2 dB improvement in performance.

- By utilizing the iterative, joint decoding/estimation provided by the QDF algorithm, the performance improves over the best PO receiver by 0.9 dB with BPSK modulation and 0.2 dB with QPSK modulation.

- Although the best proposed practical scheme, the optimized QDF receiver, provides significant gains over the non-optimized PO receiver, its performance is still 2.4 dB away from capacity.

Details of our work with LDPC codes on the time-selective, frequency-non-selective complex-fading channel are provided in Chapter 2, and this work has also been published in [33] and [34].

1.2.2 Finite-Length Analysis for the BEC

In the world of communications, there exists a constant demand for communication systems capable of delivering higher data rates with low power, low error rate, and low complexity. LDPC codes, with their capacity-achieving capabilities and low complexity, are excellent candidates for high-performance communications. The research community has developed LDPC codes with very good finite-length threshold performance, i.e., with sharper waterfall regions. However, current LDPC codes are still plagued by error floors. Due to limited power available at transmitters, SNR cannot be indefinitely increased to obtain lower and lower error probabilities, particularly in the error-floor region where small improvements in error rate come at a high cost, i.e., large increases in SNR. To be able to achieve extremely low error rates without high-power requirements, the error floor must be lowered.

We will analyze error-floor performance of LDPC codes over the BEC. Due to the simplicity of this channel, analysis is much simpler and trends are more easily recognizable. Although we would ideally study more realistic channels, such as the AWGN channel, analysis for the AWGN channel is very difficult and no closed form solution exists [35]. Some simulation is still required to obtain analytical results, and obtaining a tractable theoretical analysis is not promising. However, analysis for the BEC channel can still provide insights into code behavior for the AWGN channel since the effects of the code structure on performance are related for both channels.

As shown in Section 1.1.6, the error-floor performance of LDPC codes over the BEC is determined by stopping sets. Asymptotic analysis of weight and stopping-set enumerators, for codewords and stopping sets which grow linearly with codelength, has aided in designing LDPC codes with lower error floors but does not reflect the behavior of sublinearly-sized stopping sets, which can dominate the iterative-decoding error-floor performance [2, 18–20]. Thus, we provide a perspective on protograph-

based and standard LDPC ensemble enumerators, based on analysis of stopping sets with *sublinear* growth, which brings new insight into sublinear stopping-set behavior, advantages of protograph structure, and effects of precoding.

In Chapter 3, we present our finite-length analysis of protograph-based and standard LDPC code ensembles based on sublinear stopping-set enumerators. By approximating the enumerator expressions, the following results and derivations are obtained for sublinearly-growing stopping sets:

- For stopping sets which grow at most logarithmically with codelength, the stopping-set enumerators follow a polynomial relationship with codelength. This behavior is contrasted with the exponential relationship with codelength characteristic of linearly growing stopping sets, shown in [2, 19].

- Bounds are derived for the region of validity of the approximations to help address the question, "Given a finite stopping-set size and a finite codelength, do the stopping sets follow the behavior predicted by the analysis of linear or sublinear stopping sets?"

- Protograph-based LDPC ensembles always perform at least as well as standard LDPC ensembles, in terms of sublinear stopping-set behavior, and can, in fact, perform strictly better.

- Using linear and integer programming, the dominating enumerator exponents can be easily evaluated.

- The dominating enumerator exponent generally follows a linear trend with stopping-set size, and the slope of this linear trend can provide a single metric for comparing the error-floor performance of different LDPC ensembles.

- The technique of precoding, which provides improved threshold performance, is analyzed to see its affect on error-floor performance. A worst-case bound

is derived for the change in the dominating enumerator exponent when a protograph-based LDPC ensemble is precoded.

Illustrative examples, including (3,6) LDPC ensembles, are provided to verify the analytical results and illustrate the insights and concepts discussed.

The analysis and results in this dissertation help to provide a better understanding of how LDPC code structure influences error-floor performance. Details of our work on finite-length analysis of LDPC ensembles for the BEC are provided in Chapter 3 and are partially published in [36].

CHAPTER 2

LDPC Codes for Time-Selective Complex-Fading Channels

As the demands on wireless communication systems continually increase, the design of better coding schemes is necessary to achieve high data rates with low error probabilities and low complexity. A major challenge in many wireless environments is multipath fading. Multipath fading occurs when a signal propagates from the transmitter to the receiver via multiple paths, e.g., after reflections off objects in the environment, refraction, or other miscellaneous effects. Since each path results in a different received amplitude and phase, the combination of the different paths results in multipath fading where both the amplitude and the phase of the received signal is altered by the channel. The unknown phase shift introduced by the fading channel can be detrimental to the performance since the phase shift may cause the transmitted signals to be mapped onto other points in the signal constellation and hence, causing all signals to be decoded incorrectly.

In this chapter, we consider an independent block-fading channel model. Not only is this model easier to analyze, but it also closely represents several real-world scenarios. For example, for applications using frequency-hopped spread-spectrum (FHSS),

the transmitter changes the carrier frequency of the transmitted signal every block and hence, each block has independent fading. Various applications include cellular phone communications to satellite communications facing partial-band jamming.

To combat the unknown phase shift and amplitude degradation of the signal at the receiver introduced by the fading, known pilot symbols will be transmitted for each block. The techniques and results presented in this chapter can be used to approximate behavior for continuously changing fading channels, as well, by choosing block sizes on the order of the channel coherence time.

For the channel coder, low-density parity-check (LDPC) codes are an excellent candidate for use on the fading channel. These codes in conjunction with iterative decoding based on message-passing algorithms have been shown to achieve excellent performance over the AWGN channel [10, 11]. Their potential as capacity-achieving codes for more realistic wireless channels has not been established yet. However, experimental evidence as well as some preliminary analytical results [4] have led to the conjecture [17] that LDPC—or in general, turbo-like codes—can achieve capacity for a wide range of channels[1].

Recently, it was demonstrated that LDPC codes show very good performance over the memoryless frequency-non-selective (i.e., flat) Rayleigh fading channel [4] and for the noncoherent AWGN channel [37, 38]. A more realistic channel is considered in this chapter. Specifically, a time-selective, frequency-non-selective complex fading channel is considered, where both the effects of amplitude and phase variation are taken into account. This is certainly a more realistic model than the one assuming only amplitude [4] or only phase [37, 38] variations of the transmitted signal. Furthermore, in this work, the memoryless assumption of [4] is raised and channel dynamics are explicitly taken into account by considering a block-independent fading

[1]Although the focus of this chapter is on LDPC codes, the decoding algorithms and analysis techniques can also be applied to other turbo-like codes.

model. In particular, the complex fading is considered constant for a block of length N (which can be thought of as the channel coherence time) and independent from block to block. This model is quite accurate for frequency-hopping or time-division multiple-access schemes. It is also a good model for more general channels since it simplifies analysis by modeling the dynamics of the fading process through a single parameter N.

Although coding for this channel is generally complicated as evidenced in [39][2], in this chapter we consider a simple coding scheme, namely pilot-symbol-assisted (PSA) LDPC codes[3]. Pilot symbol(s) of specified energy are added in the beginning of each block of length N to establish a reference for the phase of the symbols and to aid (implicitly) the estimation/decoding process. We consider a general scenario where both the number of pilots as well as the pilot energy are design parameters[4]. Clearly, the quality of channel estimation improves with increased energy in the pilots, while the quality of the decoded symbols depends on the energy spent on the code symbols. Thus, for a fixed energy per information bit, a trade-off between allocation of energy to the pilot and code symbols exists. Using density evolution [10, 11], this trade-off is studied without resorting to simulations, and the optimal power allocation is obtained for several PSA receiver structures first suggested and analyzed in [38] for the block-independent noncoherent AWGN channel. The *optimized* PSA LDPC codes are shown to have significantly improved performance over the non-optimized codes, and the optimal energy allocation to pilot and code symbols depends both on the channel coherence time and the particular receiver used. Thus a quantitative

[2]It is noted that the work in [39] refers to the more interesting case of multiple-input/multiple-output complex-fading channels.

[3]When the channel state information is known perfectly at the receiver (i.e., for the perfect-CSI receiver), the PSA scheme is not needed and hence, it is not used.

[4]One can consider an equivalent—or actually a more bandwidth efficient—system having a single pilot per block of length N with a specified energy. However, multiple pilots per block are useful when the total pilot energy needs to be spread in time due to amplifier dynamic-range constraints at the transmitter.

answer, which is closely related to a particularly simple family of codes and decoders, is obtained to the question of "how much pilot energy is required for transmission in the noncoherent fading channel?" [40, 41] for the single-antenna scenario.

In this chapter, two classes of receivers are considered. In the first one, channel estimation is performed once, followed by iterative decoding. In the second one, estimation and decoding are performed in an iterative fashion, resulting in performance closer to that of belief propagation (i.e., the sum-product algorithm) but with only a fraction of its complexity. As mentioned earlier, these receiver structures have been discussed before in the context of LDPC decoding in the noncoherent AWGN channel [38], and similar receivers have also been proposed in the context of decoding turbo codes in the presence of time-selective fading [42, 43]. The presented formulation, however, leads to a number of surprising results regarding the analysis and design of these codes.

First, it is shown that for a given (regular or irregular) LDPC code, and for the first class of receivers described above, performance analysis in the form of density evolution only needs to be performed once regardless of the channel dynamics N, the number of pilots, and the energy allocation to pilot and code symbols. Moreover, this analysis is exactly the same as if the code operated in an equivalent fading channel with perfect channel-state information (CSI) available at the receiver and with a smaller signal-to-noise ratio.

Second, design and optimization of irregular LDPC codes for the first class of receivers discussed above is greatly simplified. In particular, the design process can be decomposed into two steps. The first one, which is the most time consuming, involves degree polynomial optimization and is usually performed by computer search using differential evolution techniques [4, 44]. It is shown here that this step need only be performed once regardless of whether the code operates with perfect-CSI or

with no CSI at the receiver and regardless of the channel dynamics, N. Thus, the best codes for perfect-CSI are also the best codes for no CSI and for any channel dynamics, for the receivers of the first class. The second step involves optimization of the allocation of energy to the pilot and code symbols and can be performed using closed-form expressions.

The above two results are very different from the corresponding results observed in [37, 38] for the noncoherent AWGN channel. In particular, in the noncoherent AWGN channel, analysis and code design cannot be unified as discussed above for the complex-fading channel and must be conducted separately for each receiver, for each value of the channel coherence time N, and for each value describing the energy allocation to pilot and code symbols.

When additional complexity can be afforded, one can utilize the receiver structures of the second class that perform estimation and decoding in an iterative fashion. When these receivers are used, the two results mentioned above are no longer valid. We optimize PSA LDPC codes for these receivers and show performance gains over the receivers of the first class.

The rest of this chapter is structured as follows. In Section 2.1, we discuss, in detail, the system and channel model under consideration. The specific message-passing estimation/decoding algorithms are described in Section 2.2 while performance analysis using density evolution and code design are presented in Section 2.3. Finally, section 2.4 presents numerical results and conclusions.

2.1 System and Channel Model

For the channel model, we consider a block-independent fading model where the fading is constant for a block of N symbols and is independent from block to block. To facilitate channel estimation in each block, we use the PSA scheme depicted in

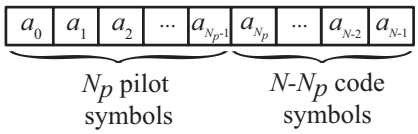


Figure 2.1: Block diagram for the pilot-symbol-assisted scheme in a block of length N.

Fig. 2.1 where the first N_p transmitted symbols are pilot symbols, each with energy E_p, followed by $N - N_p$ code symbols, each with energy E_s. For the block-fading model, the exact placement of the pilot symbols in the block will not affect the performance.

An LDPC code is used as the underlying code for the system. For each block of length N, we transmit symbols $(-1)^{a_k}$ for all $k = 0, \ldots, N-1$ with pilot symbols $a_k = 0$ for $k = 0, \ldots, N_p - 1$ and with code symbols from the LDPC code $a_k \in \{0, 1\}$ for $k = N_p, \ldots, N-1$. The analysis presented in this chapter is applied to general irregular LDPC codes with maximum variable (check) node degree of d_v (d_c) and degree polynomials $\lambda(x)$ and $\rho(x)$ as defined in [11], but they can also be applied to other codes which can be represented by a factor graph.

The received symbols for each block of length N can be expressed as

$$z_k = c\sqrt{E_k}(-1)^{a_k} + n_k \qquad k = 0, \ldots, N-1, \tag{2.1}$$

where

$$E_k = \begin{cases} E_p & k = 0, \ldots, N_p - 1 \\ E_s & k = N_p, \ldots, N-1. \end{cases} \tag{2.2}$$

In the above equations, the fading coefficient c is modeled as a zero-mean, circular complex Gaussian random variable with $E[|c|^2] = 1$. Thus, the fading amplitude has a Rayleigh density while the fading phase has a uniform density in $[0, 2\pi)$. The additive noise is modeled by independent zero-mean, circular complex Gaussian random

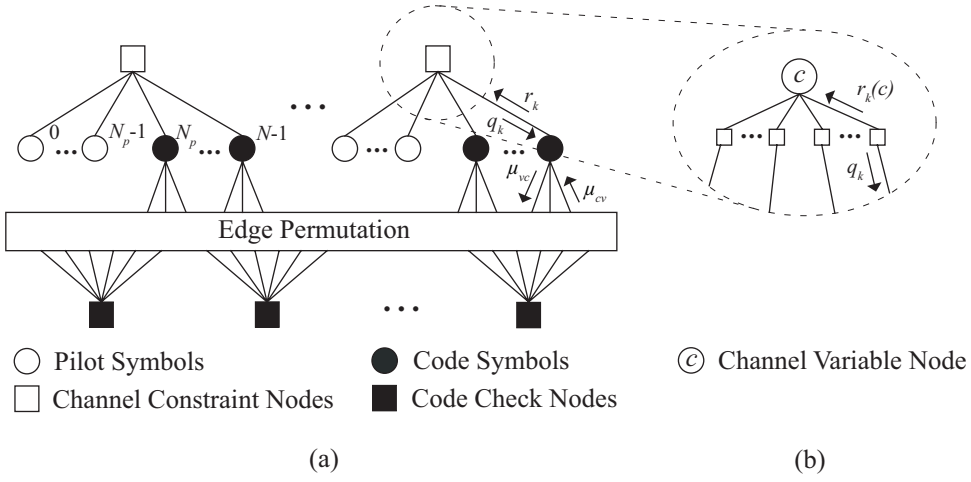


Figure 2.2: (a) Factor graph for a pilot-symbol-assisted LDPC code in a block-independent flat-fading channel. (b) Equivalent representation of the channel constraint node in (a). The channel variable node represents the fading coefficient c.

variables n_k with $E[|n_k|^2] = N_0$.

The effective energy per information bit is

$$E_b = \frac{1}{R}(E_s + \frac{N_p E_p}{N - N_p}) \tag{2.3}$$

where R is the rate of the LDPC code. Due to the pilot transmission, the overall rate (throughput) of the code is reduced to $R_c = R(N-N_p)/N$ (bits/complex dimension). For the channel model presented above, if no constraints on the pilot energy E_p exist, then a PSA scheme with $N_p > 1$ is suboptimal since we can simply put the total energy $N_p E_p$ into a single pilot and increase the total throughput by a factor of $(N-1)/(N-N_p)$. However, if the peak power is limited due to linearity constraints at the transmitter amplifier, then multiple pilots per block might be necessary to achieve higher effective pilot energies at the cost of lowering the overall throughput.

We consider a regular PSA scheme where the number of pilot symbols per block, N_p, and the energy per pilot symbol, E_p, are the same for all blocks. The factor graph [21] of this system is shown in Fig. 2.2(a). Each transmitted symbol, i.e., each

pilot symbol and each code symbol, is represented by a variable node while each parity check is represented by a code check node. In addition, the variable nodes are connected to channel constraint nodes which represent the constraints imposed by the fading channel. In an equivalent representation, each channel constraint node can be decomposed as in Fig. 2.2(b) to explicitly express the dependence on the fading coefficient c. In this case, a variable node describes c while each of the channel constraint nodes here represents the channel constraint, based on c, on a single transmitted symbol. It will be shown in Section 2.2 that the two factor graph representations are indeed equivalent. Although a channel interleaver is usually inserted between the encoder and the channel, this device is not necessary for LDPC codes due to the fact that interleaving is inherent since the order of the code symbols is irrelevant.

2.2 Decoding Algorithms

The PSA LDPC codes can be iteratively decoded by message-passing algorithms operating on the factor graph of the system. Since message exchange between variable and check nodes for LDPC codes is well understood (see Section 1.1.3), we concentrate here on the messages generated at the channel constraint node in Fig. 2.2(a). In the following, we describe several receiver algorithms which rely on different options for message generation at the channel constraint node. The receivers discussed here were proposed in [38] for the noncoherent AWGN channel. Thus, in the following, we concentrate on the details of these receivers that are specific to the channel under consideration. In particular, (i) analytical results are easier to obtain here compared to [38] since closed-form expressions exist for the message densities, as will be shown in Section 2.3, and (ii) analysis and code design for different system and channel parameters are unified for the first class of receivers in the complex-fading channel

considered here, which was not the case for the noncoherent AWGN channel in [38].

2.2.1 Perfect Channel-State Information

When perfect-CSI is available at the receiver, i.e., when the fading coefficient c is exactly known at the receiver side, the message q_k from the channel constraint node to the kth variable node can be evaluated as in [4]

$$q_k = \log \frac{f(z_k|a_k = 0, c)}{f(z_k|a_k = 1, c)} = \frac{4\sqrt{E_s}}{N_0} Re\{z_k c^*\}. \tag{2.4}$$

Since the fading coefficient is known, no pilot symbol is necessary. In addition, since q_k is independent of all incoming messages r_i (as shown in Fig. 2.2(a)) for $i \in \{0, \ldots, N-1\} \backslash \{k\}$, the message q_k is evaluated once at the beginning of the iterative algorithm and does not change in subsequent iterations.

2.2.2 Sum-Product Algorithm

When CSI is not available at the receiver, iterative detection and estimation can be performed using the sum-product algorithm. In particular, the channel constraint node generates a log-likelihood ratio for the kth variable node based on the information it receives from the $N-1$ other variable nodes and from the N channel output values. Assuming that the incoming message from the ith variable node to the channel constraint node is a log-likelihood ratio of the form $r_i = \log(p_i(a_i = 0)/p_i(a_i = 1))$,[5] the outgoing message to the kth variable node can be expressed as

$$q_k = \log \frac{f(\mathbf{z}|a_k = 0)}{f(\mathbf{z}|a_k = 1)} = \log \frac{\sum\limits_{\mathbf{a}:a_k=0} \mathcal{CN}_N(\mathbf{z}; \mathbf{0}, \mathbf{K}(\mathbf{a})) \prod\limits_{i=0, i\neq k}^{N-1} p_i(a_i)}{\sum\limits_{\mathbf{a}:a_k=1} \mathcal{CN}_N(\mathbf{z}; \mathbf{0}, \mathbf{K}(\mathbf{a})) \prod\limits_{i=0, i\neq k}^{N-1} p_i(a_i)}, \tag{2.5}$$

[5]It is assumed that the incoming messages corresponding to the pilot symbols are $r_k = +\infty$ for $k = 0, \ldots, N_p - 1$.

where $\mathbf{z} = (z_0, \ldots, z_{N-1})^T$, the summations are over all possible vectors $\mathbf{a} \in \{0,1\}^N$ with the kth element given as $a_k = 0$ or $a_k = 1$, and $\mathcal{CN}_N(\mathbf{z}; \mathbf{0}, K(\mathbf{a}))$ denotes an N-dimensional, zero-mean, complex Gaussian probability density function with covariance matrix

$$\mathbf{K}(\mathbf{a}) = E[\mathbf{z}\mathbf{z}^H|\mathbf{a}] = \boldsymbol{\mu}\boldsymbol{\mu}^H + N_0\mathbf{I}_N, \tag{2.6}$$

where $\boldsymbol{\mu} = (\mu_0, \ldots, \mu_{N-1})^T$, $\mu_i = \sqrt{E_i}(-1)^{a_i}$ for all $i \in \{0, \ldots, N-1\}$, and $\mathbf{I}_N$ is the $N \times N$ identity matrix.

The expression in (2.5) does not explicitly involve the unknown channel parameter c since it has been implicitly integrated out of the expression. The message q_k can also be expressed following the equivalent factor graph representation in Fig. 2.2(b) where the dependence on the complex fading coefficient c is explicit:

$$\begin{aligned} q_k &= \log \frac{\int\limits_{\mathbb{C}} f(\mathbf{z}|a_k = 0, c) f(c) dc}{\int\limits_{\mathbb{C}} f(\mathbf{z}|a_k = 1, c) f(c) dc} \\ &= \log \frac{\int\limits_{\mathbb{C}} \mathcal{CN}(z_k; \sqrt{E_k}c, N_0)\mathcal{CN}(c; 0, 1) \prod\limits_{i=0, i\neq k}^{N-1} r_i(c) dc}{\int\limits_{\mathbb{C}} \mathcal{CN}(z_k; -\sqrt{E_k}c, N_0)\mathcal{CN}(c; 0, 1) \prod\limits_{i=0, i\neq k}^{N-1} r_i(c) dc} \end{aligned} \tag{2.7}$$

where $\mathbb{C}$ is the complex plane and

$$r_i(c) = \sum_{a=0}^{1} \mathcal{CN}(z_i; \sqrt{E_i}c(-1)^a; N_0) p_i(a_k = a) \tag{2.8}$$

for $i = 0, \ldots, N-1$. By explicitly integrating over c, it can be shown that the messages in (2.5) and (2.7) are equal, thus establishing the equivalence between the factor graphs in Figs. 2.2(a) and 2.2(b).

It is a well-known fact that if the factor graph is cycle-free, then at the termination of the sum-product algorithm, the maximum a posteriori (MAP) estimate

of each code symbol is obtained [21]. However, evaluating the channel-to-variable-node message in (2.5) has exponential complexity in N. Similarly, evaluating the complex integrals in (2.7) has—at least theoretically—infinite complexity. The integration can be approximated by quantizing the amplitude and phase of c, which is the method commonly used in practice [45]. Motivated by the high complexity of the exact sum-product algorithm in evaluating the channel-to-variable-node message, suboptimal implementations of this operation are suggested in Sections 2.2.3 and 2.2.5. In addition, the hypothetical receiver described in Section 2.2.4 provides a bound on the performance of the exact sum-product algorithm.

2.2.3 Pilot-Only Detection

In the pilot-only (PO) receiver, only the pilot symbol(s) are used by the channel constraint node to obtain information about the channel. The message q_k for the kth variable node is computed by

$$\begin{aligned} q_k &= \log \frac{f(z_k, z_0, \ldots, z_{N_p-1}|a_k = 0)}{f(z_k, z_0, \ldots, z_{N_p-1}|a_k = 1)} \\ &= \frac{4\sqrt{E_s} Re\{z_k \sqrt{E_p} \sum_{i=0}^{N_p-1} z_i^*\}}{N_0(N_0 + N_p E_p + E_s)}. \end{aligned} \tag{2.9}$$

Similar to the perfect-CSI case, the message q_k is independent of all messages r_i for $i \in \{0, \ldots, N-1\}\backslash\{k\}$, so it is evaluated once at the beginning of the iterative process and remains the same at each iteration.

2.2.4 Pilot and Data Correct Decision Feedback

The pilot and data correct decision feedback (PDCDF) receiver is a hypothetical receiver which cannot be implemented in practice. However, it can be used to derive

a lower bound on the minimum E_b/N_0 required to achieve error-free transmission using the exact sum-product algorithm. In the PDCDF receiver, the output of the channel constraint node for the kth symbol is determined by the N_p pilot symbols as well as the other $N - N_p - 1$ code symbols in the block. It is assumed that due to the presence of a genie, the correct values of these $N - N_p - 1$ code symbols $\mathbf{a}'_k = (a_{N_p}, \ldots, a_{k-1}, a_{k+1}, \ldots, a_{N-1})$ are available at each iteration. The resulting message is of the form

$$\begin{aligned} q_k &= \log \frac{f(\mathbf{z}|\mathbf{a}'_k, a_k = 0)}{f(\mathbf{z}|\mathbf{a}'_k, a_k = 1)} \\ &= \frac{4\sqrt{E_s} Re\{z_k(\sqrt{E_p} \sum_{i=0}^{N_p-1} z_i + \sqrt{E_s} \sum_{i=N_p, i\neq k}^{N-1} (-1)^{a_i} z_i)^*\}}{N_0(N_0 + N_p E_p + E_s(N - N_p - 1) + E_s)}. \end{aligned} \tag{2.10}$$

Once again, it is observed that the messages q_k remain the same at each iteration. In addition, (2.10) is equivalent to (2.9) with $(N - N_p - 1)$ additional pilot symbols, each with energy E_s. Thus, as will be verified in Section 2.3.1, the PDCDF receiver is equivalent to a PO receiver with an effective pilot energy of $N_p E_p + (N - N_p - 1)E_s$, since the other $N - N_p - 1$ code symbols in the block also act as pilots for the kth variable node. Following an argument similar to the one used in [38], one can show rigorously that the PDCDF receiver cannot perform worse than the sum-product algorithm.

2.2.5 Quantized Decision Feedback

In the PDCDF receiver, $N - N_p - 1$ code symbols in a block act as pilots for calculating the log-likelihood ratio for the kth variable node. This is a hypothetical scenario where the receiver knows exactly the values of these $N - N_p - 1$ symbols. In practical scenarios, when the r_i messages entering the channel constraint node

are strongly biased towards $\pm\infty$, the corresponding symbols can act as pilots as well. Motivated by this observation, we propose an ad-hoc algorithm, the quantized decision feedback (QDF) receiver, which operates as follows. The incoming messages are first quantized according to the following rule

$$\hat{r}_i = \begin{cases} +\infty & \text{if } r_i > T \\ 0 & \text{if } -T \leq r_i \leq T \\ -\infty & \text{if } r_i < -T, \end{cases} \tag{2.11}$$

for $i \in \{0, \ldots, N-1\} \backslash \{k\}$ where T is a predetermined threshold value. Using these quantized messages $\hat{r}_i$ in evaluating the message q_k is equivalent to assuming that symbols for which $|r_i| > T$ act as pilots, while those for which $|r_i| \leq T$ do not contribute to the channel-estimation process. For notational simplicity, let $\hat{s}_i = sgn(\hat{r}_i)$ where $sgn(x)$ equals $+1$ if $x > 0$, 0 if $x = 0$, and -1 if $x < 0$. When $\hat{s}_i \neq 0$, $\hat{s}_i$ represents an estimate of the transmitted symbol $(-1)^{a_i}$. Also, let $\mathbf{z}'_k$ and $\hat{\mathbf{r}}'_k$ be the vectors of z_i's and $\hat{r}_i$'s, respectively, for all indices i such that $i \in \{0, \ldots, N-1\}$, $i \neq k$, and $\hat{r}_i \neq 0$. The resulting expression for the message q_k is

$$\begin{aligned} q_k &= \log \frac{f(\mathbf{z}'_k | \hat{\mathbf{r}}'_k, a_k = 0)}{f(\mathbf{z}'_k | \hat{\mathbf{r}}'_k, a_k = 1)} \\ &= \frac{4\sqrt{E_s} Re\{z_k (\sqrt{E_p} \sum_{i=0}^{N_p-1} z_i + \sqrt{E_s} \sum_{i=N_p, i\neq k}^{N-1} \hat{s}_i z_i)^*\}}{N_0(N_0 + N_p E_p + N'_k E_s + E_s)}, \end{aligned} \tag{2.12}$$

where $N'_k = \sum_{i=N_p, i\neq k}^{N-1} |\hat{s}_i|$ is the number of non-zero quantized messages $\hat{r}_i$ excluding messages from pilot symbols and the message corresponding to the kth variable node. It is emphasized that unlike the previous cases, in the QDF scheme, the channel constraint node must recalculate the message q_k at each iteration, resulting in an iterative joint-detection/estimation technique.

2.3 Performance Analysis and Code Design

The performance of the decoding algorithms in Section 2.2 is analyzed using density evolution (Section 1.1.4), which involves evaluating the probability density functions (pdfs) of the messages exchanged between the nodes of the factor graph [10], given the pdfs of the initial messages. For an LDPC code ensemble and a particular decoding algorithm, density-evolution analysis produces the threshold—the smallest SNR value for which arbitrarily small error probability can be achieved.

To apply density evolution, we assume that the all-zero codeword is transmitted. This assumption is not restrictive since the fading channel considered herein satisfies the channel symmetry condition $f(z_k|a_k = 0) = f(-z_k|a_k = 1)$. In addition, for these decoding algorithms, the channel constraint node preserves symmetry since a flip in sign of z_k results in a flip in sign of q_k at each iteration.

Under the standard assumptions of large girth (compared to the iteration number), the neighborhood of a graph is essentially a tree. In this case, all messages passed in the factor graph are independent and all calculated pdfs are exact. In the practical situation where cycles are present in the graph, it was shown in [10] that the average behavior of the code converges to the cycle-free case as the length of the code increases.

Using density-evolution analysis techniques, codes can be designed by optimizing irregular LDPC-code degree polynomials and energy distributions E_p/E_s to minimize the E_b/N_0 threshold—the value of E_b/N_0 required to achieve arbitrarily small error probability—obtained from density evolution. In practical, numerical optimization schemes, the optimization is completed subject to an upper bound on the maximum degrees d_v and d_c of the variable and check degree polynomials, respectively.

In this section, performance analysis and code design will be discussed for two sets of receivers: a) the perfect-CSI, PO, and PDCDF receivers which belong to

the class of receivers where channel estimation is only performed once and b) the QDF receiver which belongs to the class of receivers where channel estimation and decoding are performed in an iterative fashion.

2.3.1 The Perfect-CSI, PO, and PDCDF Receivers

For all the receivers described in Section 2.2 except the QDF receiver, the messages from the channel constraint nodes do not change with iterations. Thus, in order to perform density evolution for these receivers, it suffices to evaluate the initial message pdf passed from the channel constraint nodes to the variable nodes and then, follow the standard pdf transformations described in [10] to trace the pdfs of the messages exchanged in the code portion of the factor graph. Furthermore, it can be observed that all q_k messages described previously are of the form $q = CRe\{xy^*\}$ where C is a constant, x and y are zero-mean, complex Gaussian variables, and the subscript k is dropped for notational simplicity. The pdf of q can be expressed as [46, Appendix B]

$$f(q) = \frac{v_1 v_2}{v_1 + v_2}[e^{v_2 q}u(-q) + e^{-v_1 q}u(q)] \tag{2.13}$$

where $u(q)$ is the unit step function, and v_1 and v_2 are given by

$$\begin{aligned} v_1 &= \frac{2}{C(\sqrt{E[xx^*]E[yy^*]} + E[xy^*])} \\ v_2 &= \frac{2}{C(\sqrt{E[xx^*]E[yy^*]} - E[xy^*])}. \end{aligned} \tag{2.14}$$

For the perfect-CSI, PO, and PDCDF receivers, $v_2 = v_1 + 1$ and thus, the initial pdfs can all be expressed with the same equation dependent on a single parameter,

b:

$$f(q) = \frac{b(1+b)}{1+2b}[e^{(1+b)q}u(-q) + e^{-bq}u(q)] \tag{2.15}$$

where $b = v_1$. For the perfect-CSI receiver, $C = 4\sqrt{E_s}/N_0$, $x = z_k$, $y = c$, $E[xx^*] = E_s + N_0$, $E[yy^*] = 1$, and $E[xy^*] = \sqrt{E_s}$, which results in the following simplified expression for b:

$$b = \frac{1}{2}\sqrt{1 + \frac{N_0}{E_s}} - \frac{1}{2}. \tag{2.16}$$

Since no pilots are necessary, b and $f(q)$ are independent of N_p and E_p. For the PO receiver, $C = 4\sqrt{E_s}/(N_0(N_0 + N_pE_p + E_s))$, $x = z_k$, $y = \sqrt{E_p}\sum_{i=0}^{N_p-1} z_i$, $E[xx^*] = E_s + N_0$, $E[yy^*] = N_pE_p(N_pE_p + N_0)$, and $E[xy^*] = N_pE_p\sqrt{E_s}$, which results in the following simplified expression for b:

$$b = \frac{1}{2}\sqrt{(1 + \frac{N_0}{E_s})(1 + \frac{E_s}{E_{p,eff}}\frac{N_0}{E_s})} - \frac{1}{2} \tag{2.17}$$

where $E_{p,eff} = N_pE_p$. Clearly the performance of this receiver is only dependent on E_s/N_0 and $E_{p,eff}/E_s$. For the PDCDF receiver, b is given by (2.17) with $E_{p,eff} = N_pE_p + (N - N_p - 1)E_s$.

In light of the fact that the performance of the perfect-CSI, PO, and PDCDF receivers depends on the system parameters (i.e., N, N_p, E_s/N_0, E_p/E_s) only through a single parameter b, performance analysis using density evolution for a given LDPC code ensemble can be performed easily for all of these receivers as follows.

First, using density evolution with initial message pdfs given in (2.15), the threshold value b^* is obtained such that for all $b < b^*$, the iterative-decoding algorithm converges to zero probability of bit error as the number of iterations and the codelength increases, while for all $b > b^*$, the iterative-decoding algorithm does not converge to zero probability of bit error. Such a b^* exists because the performance of the receivers is monotonically decreasing with b. A proof of this monotonicity is pro-

vided in Appendix A. Since this step is independent of the particular receiver type and independent of the system parameters, this single number b^* characterizes the performance of all of these systems.

Once b^* is found, the E_b/N_0 threshold can be obtained through closed-form expressions for each receiver type. To derive these closed-form expressions, we first find $(E_b/N_0)'$, the minimum E_b/N_0 required to achieve a given value of b. For the perfect-CSI receiver, one can solve (2.16) to obtain $(E_s/N_0)' = (B-1)^{-1}$ where $B = (1+2b)^2$ and hence,

$$\left(\frac{E_b}{N_0}\right)' = \left(\frac{E_s}{N_0}\right)' \frac{1}{R} = \frac{1}{(B-1)R} = \frac{1}{4b(b+1)R}. \tag{2.18}$$

For the PO receiver with system parameters N, N_p, and R, equations (2.17) and (2.3) involve the quantities E_s/N_0, $E_{p,eff}/E_s$, and E_b/N_0. Solving this system of two equations and three unknowns for E_b/N_0 by eliminating E_s/N_0, we obtain

$$\frac{E_b}{N_0} = \frac{1}{R}\left(1 + \frac{\rho}{N-N_p}\right)\frac{\rho + 1 + \sqrt{(\rho+1)^2 + 4(B-1)\rho}}{2(B-1)\rho} \tag{2.19}$$

where $B = (1+2b)^2$ and $\rho = E_{p,eff}/E_s$. The optimal $E_{p,eff}/E_s$ that minimizes E_b/N_0 is given by the following closed-form expression:

$$\left(\frac{E_{p,eff}}{E_s}\right)' = \frac{B(N-N_p) + \sqrt{B(N-N_p)}}{B + \sqrt{B(N-N_p)}}. \tag{2.20a}$$

This results in the minimum E_b/N_0 given by

$$\left(\frac{E_b}{N_0}\right)' = \frac{1}{R}\left[\left(\frac{N-N_p+1}{N-N_p}\right)\frac{1}{4b(b+1)} + \frac{1}{2\sqrt{N-N_p}}\left(\frac{1}{b} + \frac{1}{b+1}\right)\right]. \tag{2.20b}$$

Similarly, for the PDCDF receiver when $N \geq N_p + 2$,[6] combining (2.17) and (2.3)

[6]When $N = N_p + 1$, the PDCDF receiver is a PO receiver and hence, the equations in (2.20)

along with $E_{p,eff} = N_p E_p + (N - N_p - 1)E_s$ results in

$$\frac{E_b}{N_0} = \frac{1}{R}\left(\frac{\rho+1}{N-N_p}\right)\frac{\rho+1+\sqrt{(\rho+1)^2+4(B-1)\rho}}{2(B-1)\rho}. \tag{2.21}$$

The optimal $E_{p,eff}/E_s$ in this case satisfies the boundary condition

$$\left(\frac{E_{p,eff}}{E_s}\right)' = N - N_p - 1 \tag{2.22a}$$

which results in the minimum E_b/N_0 given by

$$\left(\frac{E_b}{N_0}\right)' = \frac{\sqrt{(N-N_p)^2+4(B-1)(N-N_p-1)}}{2R(B-1)(N-N_p-1)} + \frac{N-N_p}{2R(B-1)(N-N_p-1)}. \tag{2.22b}$$

For all of the receivers here, $(E_b/N_0)'$ is a monotonically decreasing function of b. Thus, since arbitrarily small error probability is possible if and only if $b < b^*$, $(E_b/N_0)'$ evaluated at $b = b^*$ is the E_b/N_0 threshold.

At this point, a comparison with the analysis in [38] is in order. For the case of the noncoherent AWGN channel, finding the minimum E_b/N_0 value for the PO receiver requires running density evolution for each value of E_s/N_0 and $E_{p,eff}/E_s$. Thus, searching over a two-dimensional space is required. However, in this work, due to the established equivalence among the first class of receivers, the search need only be conducted over a single parameter, b, to obtain analytical results for the perfect-CSI, PO, and PDCDF receivers and for all system and channel parameters.

Code design is also unified for all three receivers and all system and channel parameters. In particular, code design can be divided into two steps. The first step relies on the fact that since the E_b/N_0 threshold is monotonically decreasing with b^* as shown above, b^* provides an ordering for LDPC codes for these receivers over

apply in this case.

the complex-fading channel, i.e., LDPC codes with larger values of b^* will have better performance. Thus, in the first step, the irregular LDPC degree polynomials are optimized using differential evolution [4, 44], subject to an upper bound on the maximum node degrees, to produce an LDPC code with the largest b^*. The process of degree-polynomial optimization is usually aided by imposing a constraint on the degree polynomials known as the stability condition [11], which is presented in Section 2.3.3. In the case of perfect-CSI receivers, this step concludes the code design. For PO and PDCDF receivers, the second step is optimizing the energy distribution between pilot and code symbols for the LDPC code found in the first step. This optimization can be determined through the closed-form expressions given in (2.20) and (2.22).

The above discussion is essentially a constructive proof of the statement that the optimal LDPC codes for the perfect-CSI receiver coincide with the optimal codes for the PO and PDCDF receivers for arbitrary channel dynamics N and pilots per block N_p. More precisely, the optimal irregular LDPC code for all of these cases should be the same. The only difference is the allocation of power to pilot and code symbols, which depends on the particular receiver used and the channel dynamics through the closed form expressions (2.20) and (2.22). We emphasize that this equivalence results in a tremendous complexity reduction in designing good codes since the degree optimization is usually the most time consuming portion of the design process. Furthermore, even if design complexity is not an issue, the above statement guarantees that only a single binary LDPC code needs to be designed and utilized, even if a system is supposed to operate in an environment where the channel dynamics are not known a priori. For instance, the LDPC codes optimized for the perfect-CSI receiver in [4] are also optimal codes for the PO and PDCDF receivers for arbitrary channel dynamics (although this was not apparent to the authors of [4]).

2.3.2 The QDF Receiver

Analysis for the QDF receiver is more complicated since the pdf describing the outgoing messages from a channel constraint node at the lth iteration is dependent on the pdf $f^{(l-1)}(r_i)$ of the incoming messages r_i for $i \in \{0, \ldots, N-1\}$ from the previous iteration.

To derive the pdf for the outgoing messages from the channel constraint nodes at iteration l, we first consider when the code symbols act as effective pilots in the channel estimation at a particular channel constraint node. For a particular outgoing message from the channel constraint node, there are $N - N_p - 1$ code symbols (other than the one corresponding to the outgoing message) which can take part in the channel estimation. If the quantized message $\hat{r}_i = +\infty$, then the decoder correctly guesses the value of the corresponding code symbol and this code symbol acts as a *correct* effective pilot in the channel estimation. Let c be the number of code symbols, out of the $N - N_p - 1$ possibilities, that act as *correct* effective pilots, i.e., the number of code symbols for which $\hat{r}_i = +\infty$. If the quantized message $\hat{r}_i = -\infty$, then the decoder incorrectly guesses the value of the corresponding code symbol and this code symbol acts as an *erroneous* effective pilot in the channel estimation. Let e is the number of code symbols that act as *erroneous* effective pilots, i.e., the number of code symbols for which $\hat{r}_i = -\infty$. Then, the joint probability mass function $p^{(l)}(c, e)$, describing the probability that c correct and e erroneous effective pilots are used by the channel constraint node to help estimate the channel at the lth iteration, is given by a multinomial distribution:

$$p^{(l)}(c, e) = \binom{N - N_p - 1}{c \ , \ e} p_c^c p_e^e (1 - p_c - p_e)^{N - N_p - 1 - c - e} \tag{2.23}$$

where p_c, the probability that a code symbol acts as a correct effective pilot, is given

by

$$p_c = P(\hat{r}_i = +\infty) = \int_T^{\infty} f^{(l-1)}(r_i) dr_i \tag{2.24}$$

and p_e, the probability that a code symbol acts as an erroneous effective pilot, is given by

$$p_e = P(\hat{r}_i = -\infty) = \int_{-\infty}^{-T} f^{(l-1)}(r_i) dr_i. \tag{2.25}$$

Since c code symbols act as correct effective pilots, e code symbols act as incorrect effective pilots, and $N - N_p - c - e - 1$ code symbols do not contribute at all to the message evaluation, the pdf $f(q|c, e)$ of the outgoing messages from the channel constraint node, conditioned on c and e, has the form of (2.13) with the constants v_1 and v_2 determined using (2.14) with

$$E[xx^*] = E_s + N_0 \tag{2.26a}$$

$$E[yy^*] = (N_p E_p + E_s(c - e))^2 + N_0(N_p E_p + E_s(c + e)) \tag{2.26b}$$

$$E[xy^*] = \sqrt{E_s}(N_p E_p + E_s(c - e)). \tag{2.26c}$$

Finally, the message pdf from the channel constraint nodes at the lth iteration can be evaluated using total probability as

$$f^{(l)}(q) = \sum_{c=0}^{N-N_p-1} \sum_{e=0}^{N-N_p-1-c} f(q|c, e) p^{(l)}(c, e). \tag{2.27}$$

Unlike the simplified analysis in Section 2.3.1, the analysis here requires running separate density evolutions for each set of values of E_p/E_s and E_s/N_0 in order to find the optimal set of values which minimizes the E_b/N_0 threshold for the given LDPC code. Further, this analysis must be completed separately for each set of values of N and N_p.

Code design is also much more difficult for the QDF receiver and is dependent

on the values of N and N_p. The procedure for code design described here can, in general, be applied for any receiver type. Using differential evolution [4,44], a search over degree polynomials, subject to an upper bound on the maximum node degrees, can be completed to find the best LDPC code and energy distribution, given N and N_p. The performance of each LDPC code is determined in this process by using density evolution and searching over E_p/E_s and E_s/N_0 to find the E_b/N_0 threshold.

Since this optimization process is computationally expensive, we simplify the process slightly by using the following ad-hoc iterative procedure. First, we initialize[7] $E_p/E_s = 1$. A target value is set for the E_b/N_0 threshold, and differential evolution is used to find a code, with the given value of E_p/E_s, that can achieve arbitrarily small error probability for that value of E_b/N_0. The target E_b/N_0 threshold is decreased until no such code can be found. With the best LDPC code found in the previous step, we find the optimal E_p/E_s. Then, the procedure is repeated for this new value of E_p/E_s. The iterative procedure ends when no improvement in E_b/N_0 is obtained from the last iteration. This ad-hoc procedure is not necessarily optimal and may not converge to the globally-optimal solution. However, we repeated this process for several other initial E_p/E_s values, and no significant improvement in performance was observed.

2.3.3 Symmetry and Stability Conditions

The stability condition in [11] provides necessary and sufficient conditions for density evolution on an LDPC graph to converge to zero probability of error, for binary-input/scalar-output memoryless channels. In this section, we extend the stability condition for the perfect-CSI, PO, and PDCDF receivers. Due to the complexity of

[7]This initial value was chosen based on extensive experiments and experience gained from [38], which showed that for LDPC degree polynomials with better performance, the optimal value of E_p/E_s is close to 1 for the range of values of N examined.

the message generation at the channel constraint node, conditions for convergence for the QDF receiver were not attainable.

The basic result in this area was derived in [11] for LDPC codes operating on binary-input/scalar-output memoryless channels. In particular, it was shown that if the channel input/output description is symmetric, *i.e.*, if the pdf of the scalar output y, conditioned on the binary input a satisfies

$$f(y|a=0) = f(-y|a=1), \tag{2.28}$$

then the pdf $f(q)$ of the initial message $q = \log \frac{f(y|a=0)}{f(y|a=1)}$ satisfies an exponential symmetry of the form

$$f(-q) = e^{-q} f(q) \qquad q > 0. \tag{2.29}$$

Furthermore, it was shown that this exponential symmetry is invariable to message transformations in the sum-product algorithm for messages exchanged between variable nodes and check nodes in the LDPC graph. Based on this result, necessary and sufficient conditions, involving the degree polynomials and the channel parameters, that guarantee the convergence of density evolution (with the sum-product algorithm) to zero probability of error were derived. Specifically, the general stability condition stated in [11] is as follows. For an exponentially-symmetric initial-message density $f(q)$, define

$$s = -\log \int_{-\infty}^{\infty} f(q) e^{-q/2} dq. \tag{2.30}$$

If $\lambda'(0)\rho'(1) > e^s$, then density evolution will be bounded away from zero probability of error at all iterations. If $\lambda'(0)\rho'(1) < e^s$, then there exists an $\epsilon > 0$ such that $\int_{-\infty}^{0} f(q)dq < \epsilon$ implies that density evolution converges to zero probability of error as the number of iterations increases to infinity.

There are two main obstacles in applying the above results to the algorithms

presented in Section 2.2: (i) the channels under consideration are not memoryless, and (ii) the channel output is not a scalar, but a complex vector. Regarding the first obstacle, the requirement of a memoryless channel is not actually necessary if the decoding algorithm and the channel satisfy the following conditions: the messages from the channel constraint node do not change with iterations, and all messages arriving at a given variable or code check node are independent, at each iteration. It can be shown for the PO receiver that the channel is equivalent to a binary-input vector-output channel where the input a_k results in a channel output (z_k, z_0), where z_0 is the received symbol corresponding to the pilot symbol. This case satisfies the above conditions as long as the code is long enough and the interleaver is chosen suitably.

In the following, we present a method that overcomes the second obstacle by transforming this vector channel to an equivalent scalar channel with input/output symmetry as in (2.28). In particular, consider a new channel which takes the output of the original channel, (z_k, z_0), and creates a new output

$$y_k = \log \frac{f(z_k, z_0|a_k = 0)}{f(z_k, z_0|a_k = 1)}. \tag{2.31}$$

It should be clear that y_k is a sufficient statistic for a_k since no information is lost in the new channel. Thus, the concatenation of the original and new channels forms an equivalent binary-input scalar-output channel (from a_k to y_k).

Furthermore, the density $f(y_k|a_k = 0)$ is in the form of (2.13) where v_1 and v_2 are given by

$$v_1 = \frac{N_0(E_p + E_s + N_0)}{2\sqrt{E_p E_s}(\sqrt{(E_p + N_0)(E_s + N_0)} + \sqrt{E_p E_s})} \tag{2.32a}$$

$$v_2 = \frac{N_0(E_p + E_s + N_0)}{2\sqrt{E_p E_s}(\sqrt{(E_p + N_0)(E_s + N_0)} - \sqrt{E_p E_s})} \tag{2.32b}$$

for the PO receiver, and the density $f(y_k|a_k = 1)$ is the same as $f(y_k|a_k = 0)$ except that v_1 and v_2 are switched. Thus, the equivalent channel satisfies the input/output symmetry of (2.28). As a result, the corresponding initial message q'_k used by the sum-product algorithm,

$$\begin{aligned} q'_k &= \log \frac{\frac{v_1 v_2}{v_1+v_2}[e^{v_2 y_k} u(-y_k) + e^{-v_1 y_k} u(y_k)]}{\frac{v_1 v_2}{v_1+v_2}[e^{v_1 y_k} u(-y_k) + e^{-v_2 y_k} u(y_k)]} \\ &= (v_2 - v_1) y_k = (v_2 - v_1) \frac{4\sqrt{E_s} Re\{z_k \sqrt{E_p} z_0^*\}}{N_0(N_0 + E_p + E_s)}, \end{aligned} \tag{2.33}$$

will satisfy the exponential symmetry of (2.29).

Based on these results, the stability condition of [11] holds for the PO receiver with the quantity e^s evaluated as

$$e^s = 1 + \frac{E_p E_s}{N_0(E_p + E_s + N_0)}, \tag{2.34}$$

which implicitly depends on the channel coherence time N. Furthermore, since the PDCDF receiver is equivalent to the PO receiver with an effective pilot energy of $E_{p,eff} = E_p + (N-2)E_s$, the above symmetry and stability results, with this effective pilot energy used in place of E_p in (2.34), also apply to the PDCDF receiver. Finally, for the perfect-CSI receiver, which can be viewed here as a PO receiver with $E_p = \infty$, the corresponding result can be found to be $e^s = 1 + E_s/N_0$, which agrees with the result of [4].

2.4 Numerical Results

In this section, density evolution results are generated using discretized density evolution [9]. In particular, each message is quantized using an 8-bit uniform quantizer with $2^8 - 1$ quantization levels. The range of the quantizer is roughly optimized

to obtain low E_b/N_0 thresholds. Since messages are quantized, only probability mass functions need to be evaluated. Thus, the resulting E_b/N_0 thresholds can be considered as upper bounds for belief propagation with continuous messages. However, discretized density evolution is exact for a practical receiver that quantizes the messages before iterative processing.

The results presented here, for all receiver types, use a single pilot symbol in each block, i.e., $N_p = 1$, since this is the optimal choice for N_p. For the perfect-CSI, PO, and PDCDF receivers, a search over b with a resolution of 0.001 is conducted to find b^*, the largest b such that the bit error rate is less than 10^{-8} in at most 1000 iterations. Results for the QDF receiver are obtained by searching over E_s/N_0 and E_p/E_s values to find the minimum possible E_b/N_0 (with a resolution of 0.001 dB) required to obtain a bit error rate less than 10^{-8} in at most 1000 iterations. For the QDF receiver, a separate search must be completed for each value of the channel coherence time N. At each iteration during discretized density evolution, the value of the threshold T for the QDF receiver is numerically optimized to minimize the probability of bit error at that iteration.

Fig. 2.3 displays a summary of the density-evolution results for the regular $(3, 6)$ LDPC code ensemble with a quantizer range of $(-35, 35)$. Discretized density evolution results in $b^* = 0.203$. Thus, the perfect-CSI receiver has an E_b/N_0 threshold of 3.06 dB, which agrees with the results in [4]. We note that for the perfect-CSI receiver, the overall code rate $R_c = R = 0.5$. However, for the other receivers where a pilot symbol was transmitted, $R_c = R(N-1)/N = 0.5(N-1)/N$. Thus, the comparison between these receivers with the perfect-CSI receiver is not exactly fair, especially for small values of N. This inconsistency can be corrected by using irregular LDPC codes with appropriate degree polynomials such that the overall rates of the compared systems are equal.

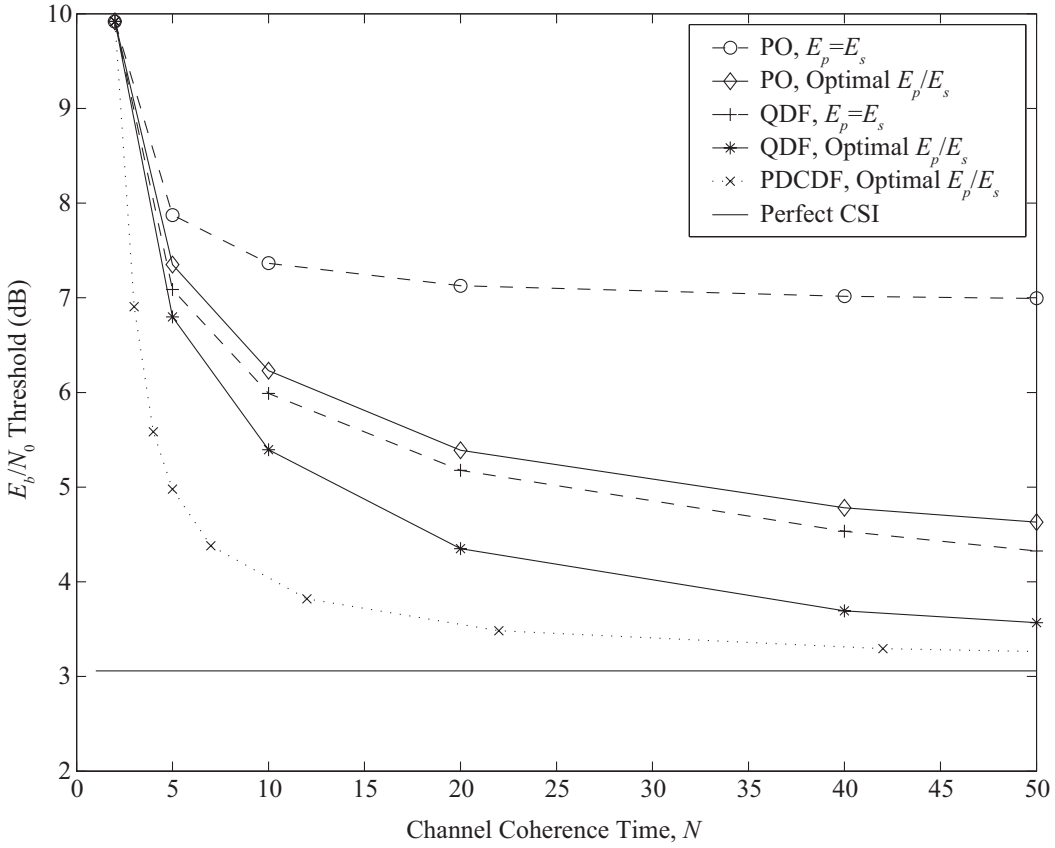


Figure 2.3: Discretized density-evolution results for the regular (3,6) LDPC code over a complex Gaussian flat-fading channel.

In Fig. 2.3, the PDCDF curve provides a lower bound on the E_b/N_0 threshold for practical receivers. It is observed that for large N, the performance of both the PO and PDCDF receivers approaches the perfect-CSI performance, which can also be proved using (2.16), and (2.17).

The performance gain when the energy distribution between E_p and E_s is optimized can also be observed in Fig. 2.3. For $N = 20$, the threshold for the PO receiver is about 1.7 dB lower with the energy distribution optimized compared to the case when $E_p = E_s$. A 0.8 dB improvement is seen in the equivalent scenario for the QDF receiver. The optimal values of E_p/E_s for $N = 10$ and $N = 20$ are 5.4 dB and 7.2 dB, respectively, for the PO receiver and 3.5 dB and 3.3 dB for the QDF

Table 2.1: Rate-1/2 Irregular LDPC Degree Polynomials Optimized for the Perfect-CSI Receiver in [4].

$\boldsymbol{\lambda}(x)$		$\boldsymbol{\rho}(x)$	
λ_1	0	ρ_1	0
λ_2	0.292439	ρ_2	0
λ_3	0.253636	ρ_3	0
λ_4	0.060454	ρ_4	0
λ_5	0	ρ_5	0
λ_6	0	ρ_6	0.007254
λ_7	0	ρ_7	0.979220
λ_8	0	ρ_8	0.013526
λ_9	0.031610		
λ_{10}	0.361861		

receiver.

The performance of the QDF receiver is significantly better than the PO receiver. For $N = 20$, the E_b/N_0 threshold for the optimized QDF receiver is about 4.4 dB, which is roughly 1.0 dB lower than the optimized PO threshold and 0.8 dB higher than the PDCDF lower bound. For $N = 40$, the QDF threshold is 3.7 dB, which is about 1.1 dB lower than the PO receiver and 0.4 dB away from the PDCDF lower bound. As N increases, the gap between the QDF and PDCDF performance decreases.

By using optimized irregular LDPC codes, improved performance over the regular (3,6) LDPC code is achieved. The irregular LDPC code described by the parameters given in Table 2.1 is a rate-1/2 code, with maximum degrees $(d_v, d_c) = (10, 8)$, optimized for the perfect-CSI receiver [4][8]. This code is also optimal for the PO and PDCDF receivers and for all values of the channel coherence time N, as shown

[8]Although the parameters $\lambda_1, \lambda_5, \lambda_6, \lambda_7, \lambda_8, \rho_1, \rho_2, \rho_3, \rho_4, \rho_5$ were set to zero in [4], optimization over these values showed no noticeable improvement.

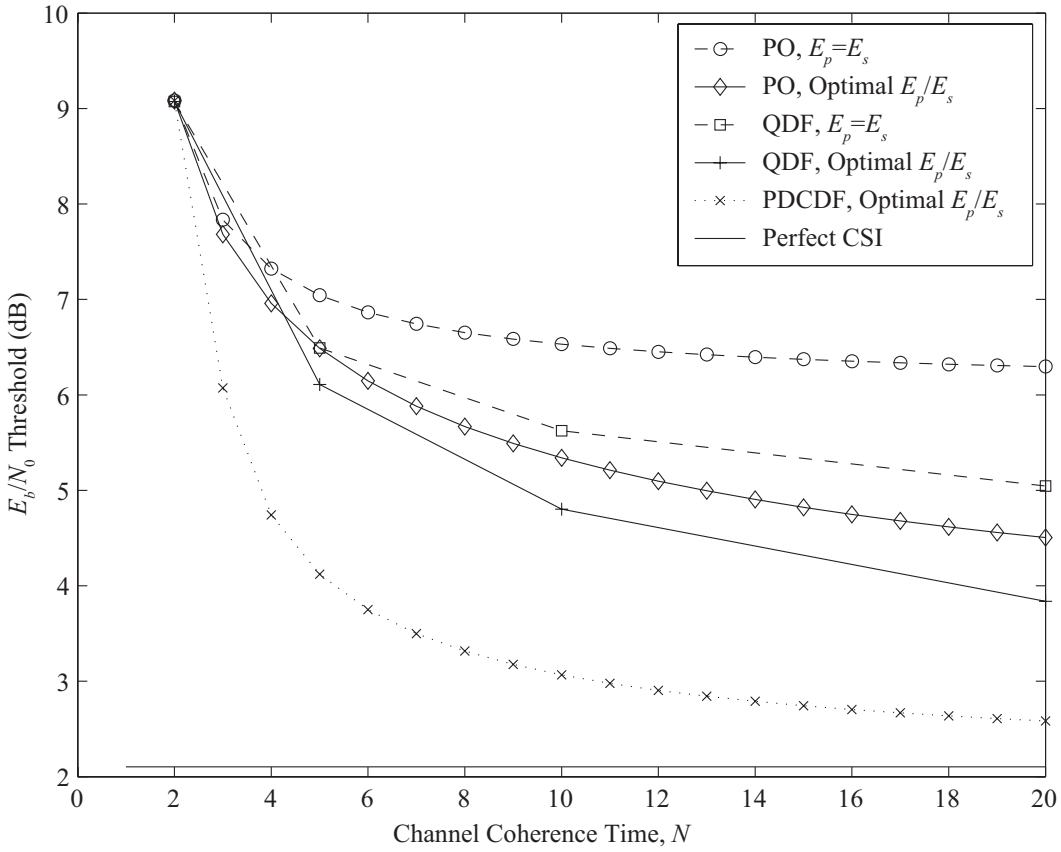


Figure 2.4: Discretized density-evolution results for the irregular LDPC code in Table 2.1 over a complex Gaussian flat-fading channel.

in Section 2.3. Based on the results of [4], only minor performance improvement can be expected by using irregular LDPC codes with higher maximum degrees. For this code, discretized density evolution with a quantizer range of $(-17, 17)$ results in $b^* = 0.247$.

Fig. 2.4 displays a summary of the density-evolution results for the irregular LDPC code given in Table 2.1. For the perfect-CSI receiver, density evolution resulted in an E_b/N_0 threshold of 2.09 dB, which agrees with the results in [4]. Similar to the regular $(3, 6)$ LDPC code, the PO and PDCDF performances with optimized energy allocations approach the perfect-CSI performance for large N. When E_p/E_s is optimized, a performance gain of about 1.2 dB for the PO receiver and about

0.8 dB for the QDF receiver is achieved over the $E_p = E_s$ case at $N = 10$.

From Fig. 2.4, observe that the PO receiver with optimal E_p/E_s values performs better than the QDF receiver with $E_p = E_s$. Thus, simply optimizing the energy allocation in the PO receiver can be more beneficial than using the more complex QDF receiver. When E_p/E_s is also optimized for the QDF receiver, the gain of the QDF receiver over the PO receiver is 0.5 dB at $N = 10$, which is comparable to the corresponding gain seen in [38] for the noncoherent AWGN channel. When $E_p = E_s$, the QDF receiver has a more significant benefit over the PO receiver, e.g., 0.9 dB at $N = 10$ and 1.3 dB at $N = 20$. Comparing this result to the corresponding gains in [43] for turbo codes with a Jake's complex-fading channel model, the benefit of iterative decoding and estimation over separate estimation and decoding is approximately 1 dB smaller here. This can be attributed to several causes: (1) the QDF algorithm is an ad-hoc algorithm which is not very efficient for iterative joint-detection/estimation, (2) in the Jake's fading channel model, the fading varies continuously, so greater performance gain can be achieved by utilizing more symbols (in addition to the pilot symbols) for channel estimation than in the block-independent model, where the fading is independent from block to block. More precisely, the channel estimation filter in [43] uses 61 symbols whereas the QDF receiver can only use at most $N - 1$ symbols in the channel estimation.

Fig. 2.5 shows the improvement in performance when the irregular LDPC code in Table 2.1 is used, compared to the regular (3,6) LDPC code performance. The E_b/N_0 threshold is about 0.9 dB lower in the PO case and about 0.6 dB lower in the QDF case compared to the regular $(3, 6)$ LDPC code at $N = 10$ when E_p/E_s is optimized.

By optimizing an irregular LDPC code with maximal degrees $(d_v, d_c) = (10, 8)$ for the QDF receiver at $N = 10$ following the procedure in Section 2.3b, we obtain

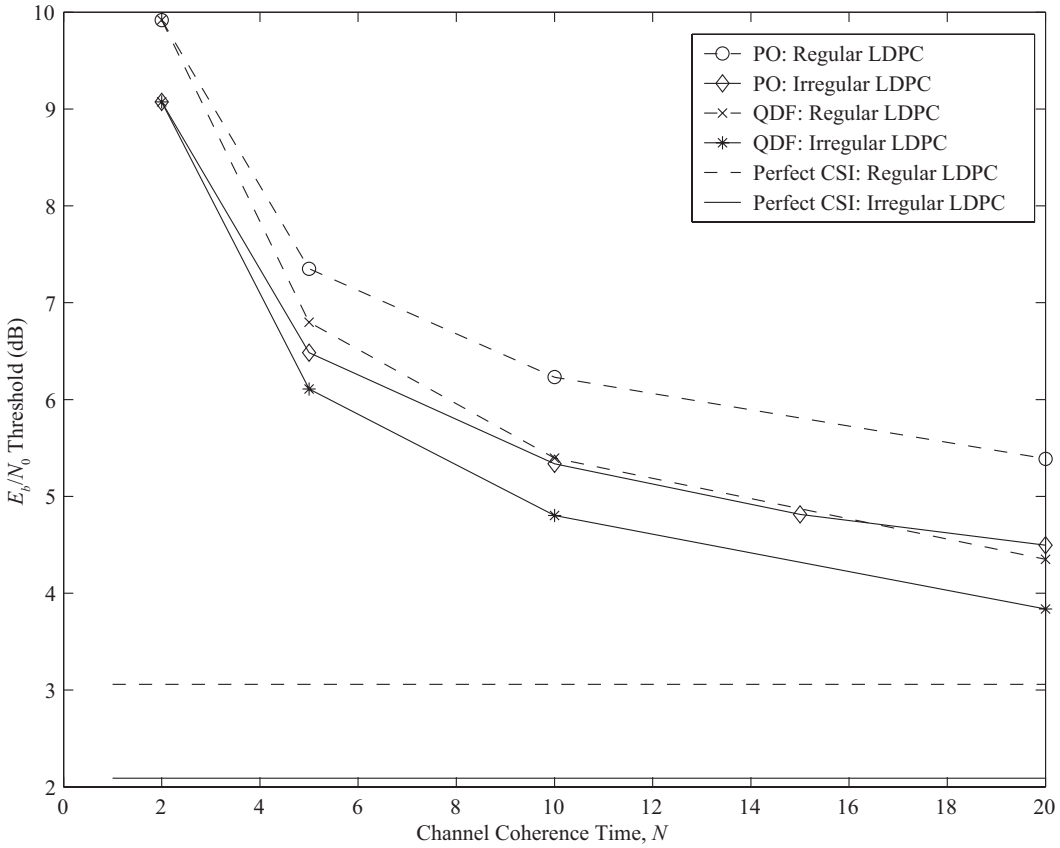


Figure 2.5: Comparison of density-evolution results for the regular (3,6) LDPC code and the irregular LDPC code in Table 2.1 over a complex Gaussian flat-fading channel.

the code in Table 2.2, and this optimized code results in some improvement in performance, as shown in Fig. 2.6. A significant gain of 1.1 dB is achieved when $E_p = E_s$ while a small gain of 0.4 dB is achieved when E_p/E_s is optimized. Thus, if adjusting E_p/E_s is a viable option in the transmitter architecture, then not much performance is lost by simply using the code optimized for the perfect-CSI receiver.

The optimal energy distributions E_p/E_s are shown in Fig. 2.7. The QDF receiver requires less pilot energy than the PO receiver since the code symbols can also contribute to the channel estimate. For the PO receiver, as N increases, more energy can be used for the pilot symbol to obtain a better channel estimate since the E_b/N_0 penalty due to pilots is reduced as N increases. The previous statement is also true

Table 2.2: Rate-1/2 Irregular LDPC Degree Polynomials Optimized for the QDF Receiver at $N = 10$.

$\boldsymbol{\lambda}(x)$		$\boldsymbol{\rho}(x)$	
λ_1	0.000000	ρ_1	0.000000
λ_2	0.373027	ρ_2	0.003367
λ_3	0.304234	ρ_3	0.026410
λ_4	0.066596	ρ_4	0.138425
λ_5	0.082820	ρ_5	0.164885
λ_6	0.003623	ρ_6	0.149070
λ_7	0.009525	ρ_7	0.117388
λ_8	0.000279	ρ_8	0.400455
λ_9	0.032068		
λ_{10}	0.127828		

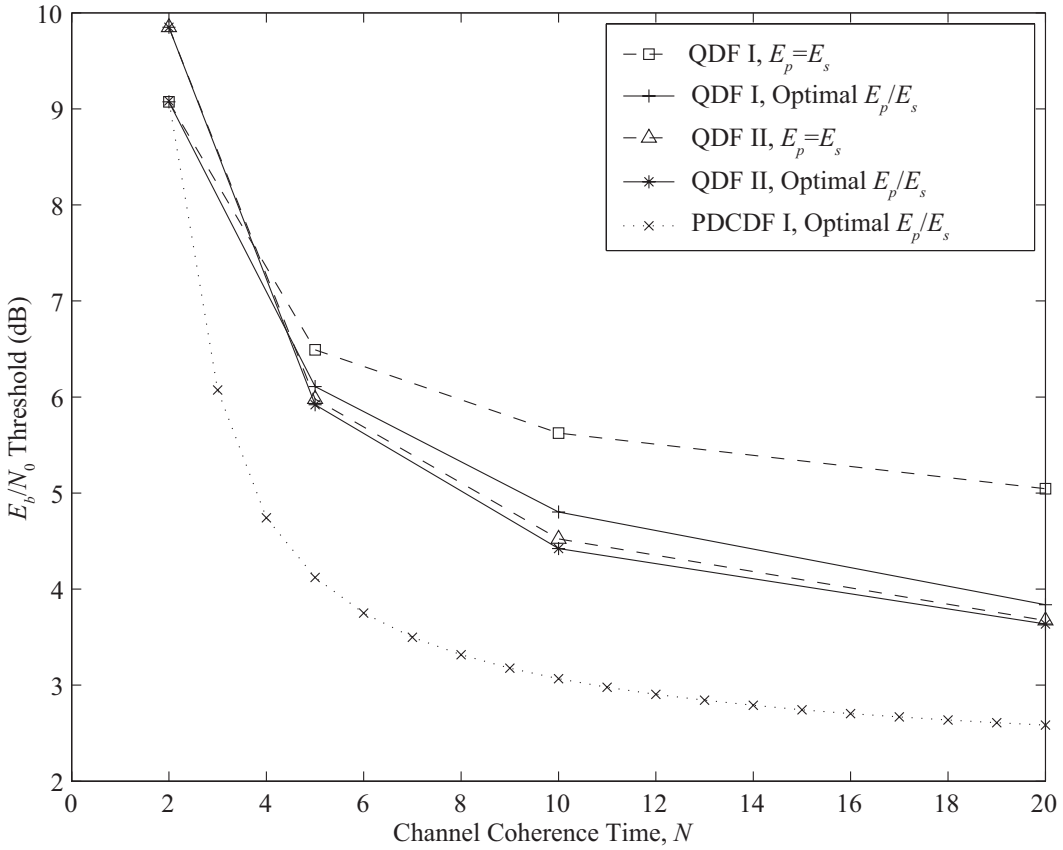


Figure 2.6: Performance comparison of the irregular LDPC codes in Tables 2.1 and 2.2, denoted by the labels "I" and "II", respectively.

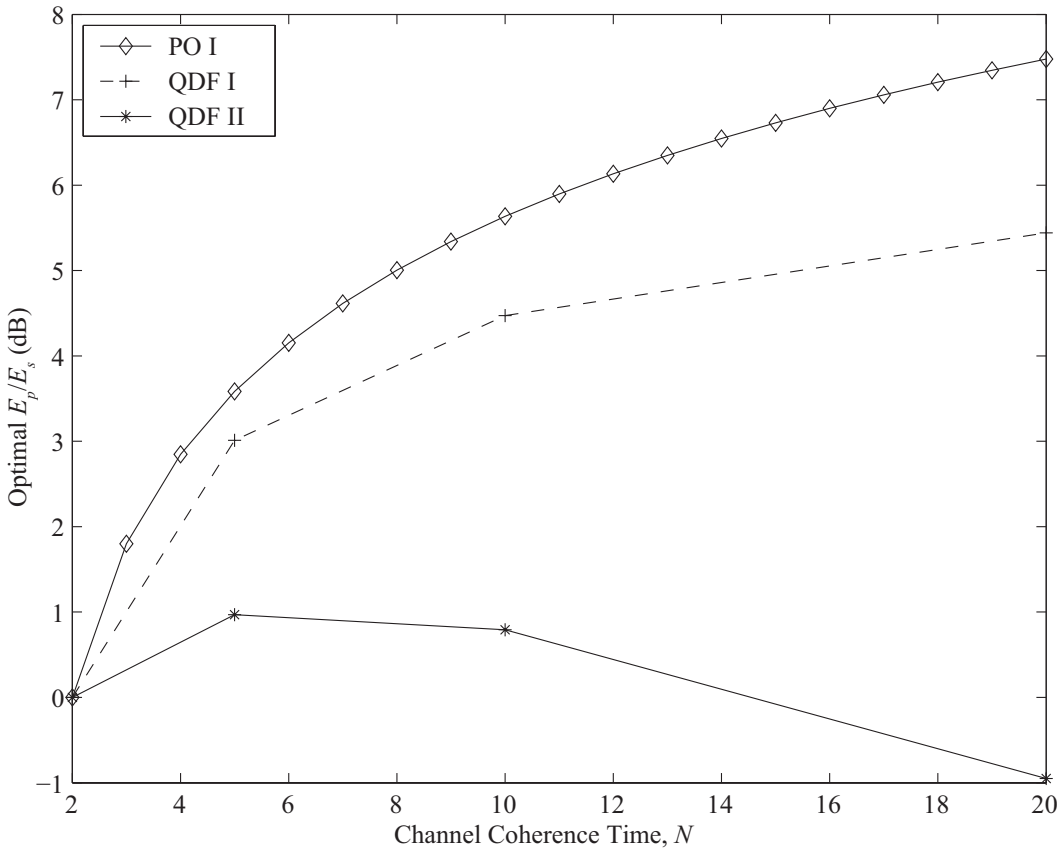


Figure 2.7: Optimal E_p/E_s for PO and QDF receivers for the irregular LDPC codes in Tables 2.1 and 2.2, denoted by the labels "I" and "II", respectively.

for the QDF receiver when N is small. However, for large N in the QDF receiver, more code symbols can contribute to the channel estimate and hence, less energy is needed in the pilot symbol[9]. Although the performance of the QDF receiver with optimized energy allocations is similar for the codes in Table 2.1 and Table 2.2, the energy allocation that achieves this performance is quite different, as evidenced by the two corresponding curves in Fig. 2.7. If the transmitter circuitry does not allow for E_p/E_s values much different than 1 (due to peak-power and/or power-added-efficiency constraints), then it is beneficial to use the code in Table 2.2 which was optimized for the QDF receiver. Thus, for the QDF receiver, good performance can

[9]This latter trend is also observed in the QDF I curve in Fig. 2.7 for larger values of N than depicted.

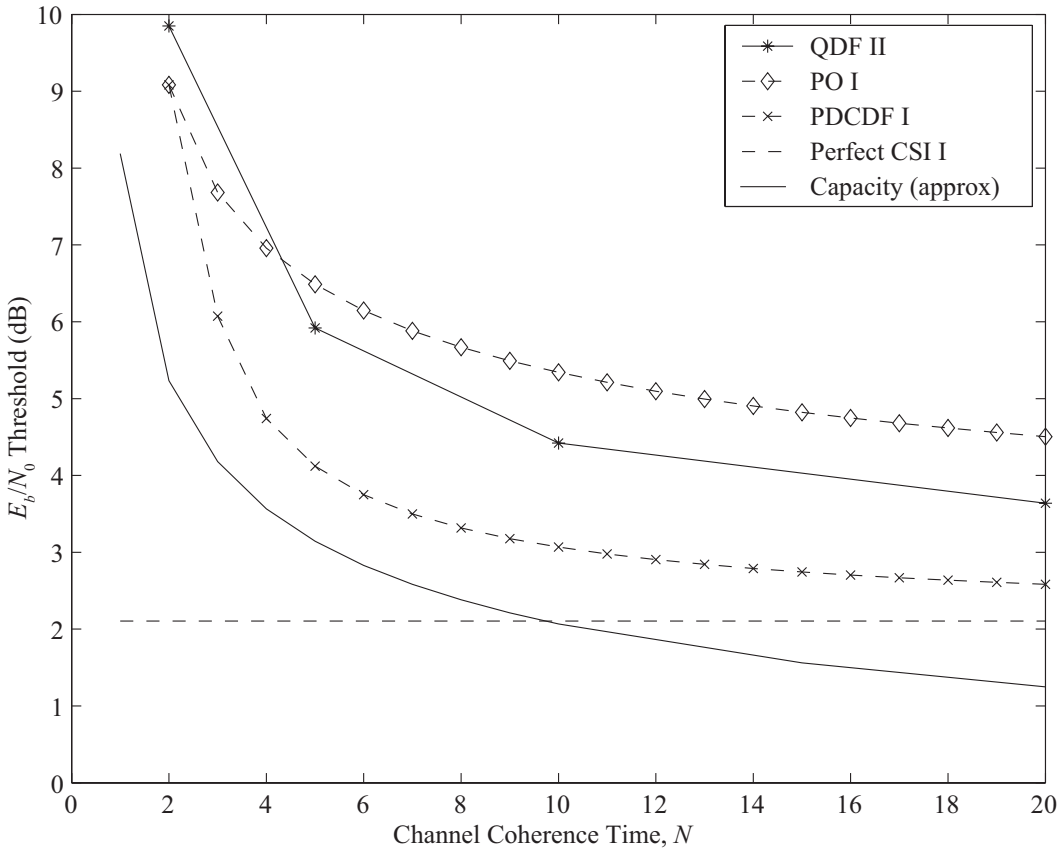


Figure 2.8: Performance of the optimized PSA LDPC codes compared to an approximation of the capacity for the complex Gaussian flat-fading channel. The labels "I" and "II" denote the irregular LDPC codes in Tables 2.1 and 2.2, respectively.

be achieved by optimizing the energy distribution E_p/E_s and/or by optimizing the LDPC code.

Fig. 2.8 compares the best performance of the optimized PSA LDPC codes for the perfect-CSI, PO, PDCDF, and QDF receivers, where optimization has been completed over the LDPC-code degree polynomials and the energy distribution in the PSA scheme, to an approximation of the capacity for the complex-fading channel. The approximation for capacity is obtained by using a two-point approximation with capacity-achieving isotropically-distributed inputs [47].

Since the code in Table 2.1 is an optimal code for the PDCDF receiver, the PDCDF curve provides a lower bound on the performance of any LDPC code with

maximal degrees $(d_v, d_c) = (10, 8)$ in a PSA scheme with BPSK modulation. Thus, Fig. 2.8 shows that the performance of such LDPC codes must be at least 1 dB from capacity.

The best QDF performance provides an improvement of 0.9 dB over the best PO performance at $N = 10$. However, the best QDF performance is still 1.4 dB away from the PDCDF bound and 2.4 dB away from capacity at $N = 10$. The gap between the optimized QDF receiver and capacity may be due to limitations of the pilot-symbol-assisted scheme, the suboptimality of the QDF receiver, and/or the specific modulation scheme (i.e., BPSK) utilized.

Simulations were conducted to verify the density-evolution results and to further investigate the performance of a more practical system utilizing quadrature phase-shift keying (QPSK) modulation. For the QDF simulations, a QPSK symbol was used as an effective pilot symbol in the message generation at the channel constraint node only if $|r_i| > T$ for both the LDPC code bits that make up the QPSK symbol. The simulation results for this system are presented in Fig. 2.9. The simulations were conducted for a regular (3,6) LDPC code and for the rate-1/2 irregular LDPC codes in Table 2.1 and Table 2.2 with a codelength of 10008, channel coherence time $N = 10$, $N_p = 1$, and 100 decoding iterations. For the QDF receiver, the threshold T was set to a fixed value for all iterations, and this fixed value was numerically optimized for each simulation point, resulting in values of T between 1 and 2.5. The E_p/E_s values used in Fig. 2.9 were roughly optimized through simulations for the QPSK-modulation scheme. Compared to the optimal E_p/E_s values obtained through density-evolution analysis with BPSK modulation, the optimal values here are roughly the same for the PO receiver and 1.4 dB larger for the QDF receiver. The increase in the optimal E_p/E_s values for the QDF receiver is expected since the code bits cannot contribute as much information to the channel estimate which

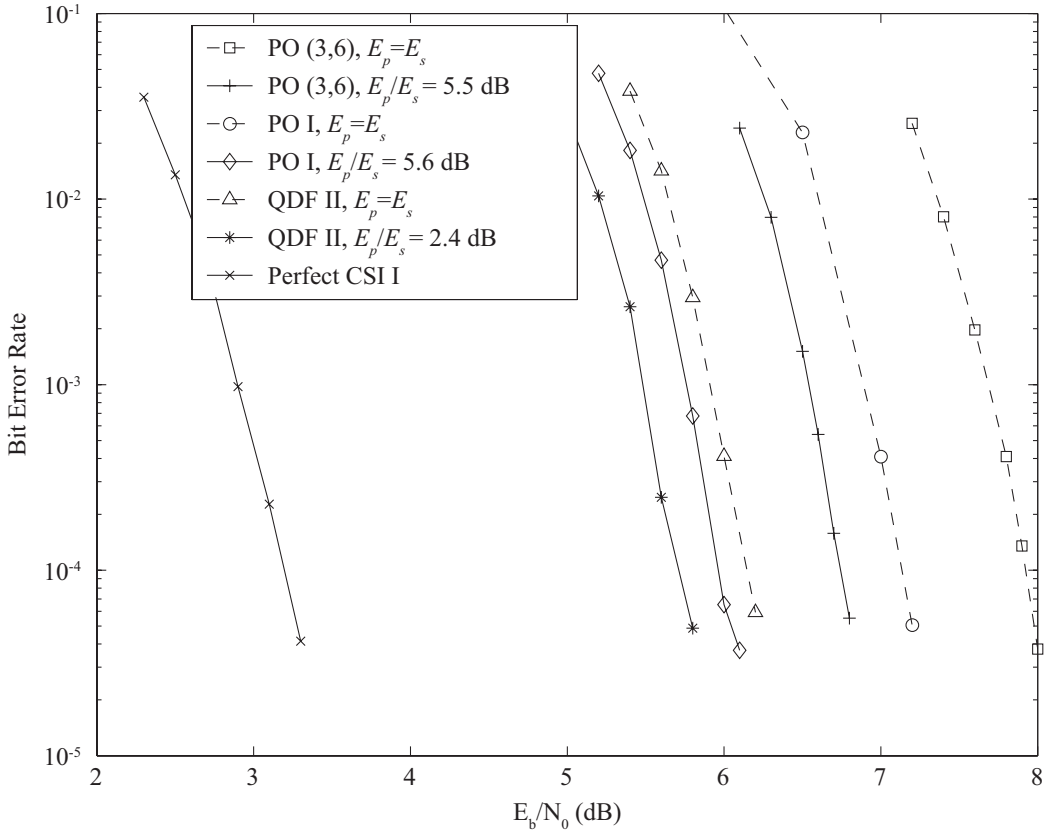


Figure 2.9: Simulations of a regular $(3, 6)$ LDPC code and the irregular LDPC codes in Table 2.1 and Table 2.2, denoted by the labels "$(3, 6)$", "I", and "II", respectively, with a codelength of 10008, channel coherence time of 10, QPSK modulation, and 100 decoding iterations.

QPSK modulation is used.

The simulation results show that the proposed iterative algorithms with QPSK modulation have performance roughly comparable to the simulated performance with BPSK modulation. For the PO receiver, performance improves by 0.1 to 0.2 dB with QPSK modulation over BPSK modulation since the penalty due to pilot insertion per information bit is smaller for QPSK modulation, and thus, a larger pilot can be used for the same E_b/N_0, resulting in better channel estimates and better performance. For the QDF receiver, performance degrades by 0.3 to 0.7 dB when QPSK modulation is used, due in part to the more stringent requirement that two code bits have

to be strongly biased in order for the corresponding QPSK symbol to act as a pilot. With QPSK modulation, the best PO performance is only slightly worse (0.2 dB) than the performance of the optimized QDF receiver whereas with BPSK modulation, the optimized QDF receiver provides a 0.9 dB improvement over the optimized PO receiver in analysis and simulation. Based on this observation, one can argue that well-designed PO schemes present an excellent performance/complexity tradeoff, compared to designs based on the QDF and other iterative decoding/estimation techniques.

At this point, a comparison with the performance of receivers in [48] is in order. In [48], turbo codes, convolutional codes, and RA codes in conjunction with differential QPSK modulation are used over a block complex-fading channel. The channel estimation portion of the iterative channel-estimation/decoding algorithm is completed through an averaging estimator for the fading amplitude and trellis decoding with quantized phases for the fading phase. Simulation results for the QDF receiver, with optimized rate 1/4 LDPC codes, 100 iterations, and all other conditions equivalent to [48], results in about 0.4 to 0.9 dB worse performance for $N = 10, 20$, and 50 compared to the best results reported in [48]. This loss is due largely to the suboptimality of the QDF receiver. However, the complexity of the QDF receiver is much lower than that of the receivers in [48], where the channel estimation requires calculations with 20 quantization levels for the phase and four possibilities for the transmitted QPSK symbol, for each of the N sections in the trellis. Thus, the QDF receiver provides a low-complexity solution at a cost of about 0.4 to 0.9 dB in performance.

CHAPTER 3

Finite-Length Analysis for Binary Erasure Channels

Infinite-length analysis of LDPC codes with density-evolution techniques has provided a method for determining the performance of a code ensemble in the asymptotic case when codelength becomes infinite. However, practical limitations on codelength prevent practical codes from achieving this asymptotic threshold performance. Thus, the ability to analyze finite-length performance is key to understanding factors affecting the performance of practical codes and to designing practical codes with good finite-length performance.

As discussed in Section 1.1.4, practical LDPC codes with finite lengths suffer from error floors, which limit the achievable error rate. In order for high-performance LDPC codes to achieve very low error rates desired for high reliability while keeping power requirements low and codelengths relatively short, this error floor must be lowered. Thus, we investigate factors affecting the error floor through finite-length analysis to gain insight into how finite-length LDPC codes can be designed to improve error-floor performance.

The analysis in this chapter is completed for the binary erasure channel (BEC)

with erasure probability ϵ since finite-length analysis is much more tractable on this channel. For the BEC, stopping sets [1] determine the iterative-decoding LDPC performance, and ensemble stopping-set enumerators capture the average performance of an LDPC ensemble, as shown in Section 1.1.6. These stopping-set enumerators behave similarly to weight enumerators, and both enumerators are useful in analyzing error-floor performance, e.g., in [2, 19, 20, 49, 50].

Asymptotic analysis of weight and stopping-set enumerators for standard and protograph-based LDPC ensembles has been completed for the case when codeword weight and stopping-set size grow linearly with codelength n [19, 20, 49, 50]. This analysis has aided in the design of LDPC codes with linear minimum distance and with good threshold and error-floor performance, e.g., [18,50]. However, the analysis does not capture the behavior of stopping sets that grow sublinearly with n, and these sublinear stopping sets can dominate the iterative-decoding error-floor performance.

As shown in [20,51], when $\lambda'(0)\rho'(1) > 1$, then with high probability there exists a stopping set with size less than or equal to $\beta \log n$ for some nonzero constant β, so the smallest, nonzero stopping-set size cannot grow faster than $\log n$. More rigorously, for an LDPC standard ensemble satisfying the condition $\lambda'(0)\rho'(1) > 1$, there exists $\delta > 0$ such that $P(v^* \leq \delta \log n) \geq 1 - c^{\sqrt{n}}$ where v^* is the smallest, nonzero stopping-set size in the ensemble and $c \in (0, 1)$ is an appropriate constant [20, 51].

Thus, the stopping sets that dominate performance in this case have size less than or equal to $\beta \log n$, motivating the study of sublinearly-sized stopping sets. Further details are provided in Section 3.1.

In this chapter, we provide a perspective on enumerators for protograph-based and standard LDPC ensembles, based on analysis of stopping sets with *sublinear* growth with codelength, which brings new insight into (1) how sublinear stopping-set enumerators behave, (2) how this behavior impacts code design, (3) how protograph

structure can improve performance over standard ensembles, and (4) how precoding affects error-floor performance. We first obtain tractable approximations to the stopping-set enumerators. Then, we address the question of when the approximations are valid, i.e., for what stopping-set sizes and codelengths do the approximations apply. The resulting analysis shows that for stopping sets that grow at most logarithmically with codelength, the enumerators follow a *polynomial* relationship with codelength, unlike the exponential relationship for linearly-growing stopping sets. Using linear and integer programming, we find a simple method for determining the dominating exponent of this polynomial relationship, and this evaluation leads to a single metric, based on the dominating exponent, for characterizing the sublinear stopping-set enumerator behavior. We also begin to address the question, "For a *finite* stopping-set size and a *finite* codelength, do the stopping sets follow the behavior predicted by the linear or sublinear analysis?" Answering this question is beneficial for gaining insight into the design of practical, finite-length codes with low error floors.

3.1 A Discussion on Stopping-Set Sizes: Motivation for Studying Sublinear Stopping Sets

In Section 1.1.6, the effect of stopping-sets of size v on error-floor performance was presented. Specifically, for any given positive integer $v \in \{1, \ldots, n\}$, we can tightly upper bound the probability of error in the error-floor region resulting from stopping sets of size v in a code ensemble of codelength n using (1.15). In order to obtain insights into code design, we will need to determine for a given code ensemble with codelength n, what stopping-set size v dominates and hence limits the error-floor performance and how this dominating v scales with n.

In the error-floor region, the error probability, which is closely upper bounded by (1.15), is dominated by the smallest stopping-set in the ensemble. Intuitively, the smallest stopping set has the most influence since the probability of all variable nodes in a small stopping set being erased by the channel is larger than the corresponding probability for larger stopping sets. Thus, an important metric for an LDPC ensemble is the **stopping number** [20], v^*, defined to be the size of the smallest, nonempty stopping set in the code ensemble.

Since the stopping number, v^*, in the code is a dominant factor in the error-floor performance, the question arises of how fast v^* can grow with n. We discuss here prior work on stopping-set enumerators for linearly-growing stopping sets and on analysis of stopping numbers. Using these results, we will motivate the study of sublinear stopping sets.

Much of the work on stopping-set enumerators has focused on stopping sets that grow linearly with n, i.e., $v = \delta_s n$ for some constant δ_s [18–20, 49, 50]. For these linearly-growing stopping sets, the enumerator has been shown to follow an exponential relationship with n, i.e.,

$$s(n, \delta_s n) \sim e^{E(\delta_s)n} \tag{3.1}$$

where the exponent is defined as

$$E(\delta_s) = \lim_{n\to\infty} \frac{\log s(n, \delta_s n)}{n}. \tag{3.2}$$

A metric $\delta_{s,min}$ is defined to be the exponent's first zero crossing from negative to positive, i.e.,

$$\delta_{s,min} = \inf\{\delta_s > 0 : E(\delta_s) > 0\}. \tag{3.3}$$

Thus, for all $\delta_s \in (0, \delta_{s,min})$, the exponent is negative and hence, the enumerator

decays exponentially with n. To design codes with this analysis in mind, they seek code ensembles with the largest possible $\delta_{s,min}$ [18, 50]. This design goal stems from the idea of maximizing minimum distance.

However, even if an LDPC ensemble has the property that $\delta_{s,min} > 0$, this does not guarantee that the stopping number grows linearly with codelength. The analysis of linearly-growing stopping sets shows that for a *particular* stopping-set size $v = \delta_s n$ for any $\delta_s \in (0, \delta_{s,min})$, the probability that stopping sets of size v exist in the ensemble is small for large n. However, this does not guarantee that there is a high probability that a *particular* LDPC code from the ensemble does not contain *any* stopping sets of size $v < \delta_{s,min} n$. Sublinearly-sized stopping sets, which are not considered in this linear stopping-set analysis, may have a significant probability of occurrence, and hence, the stopping number may be sublinear with codelength.

In the case when the minimum variable-node degree is strictly greater than two, Burshtein and Miller have shown that standard ensembles have linear stopping number with high probability [49]. Specifically, they show that with probability $1 - o(1)$, an LDPC code drawn from a standard ensemble defined by degree distributions $\lambda(x)$ and $\rho(x)$ does not have stopping sets of size γN or less for an appropriately chosen value $\gamma > 0$. However, many good codes require the presence of degree-2 nodes, e.g., the codes in [8, 11, 18, 52, 53].

Orlitsky and others have shown a stronger statement that considers standard ensembles with degree-2 variable nodes [20]. They show that if $\lambda'(0)\rho'(1) < 1$, then the probability that the stopping number is linear with codelength is $\sqrt{1 - \lambda'(0)\rho'(1)} > 0$. However, if $\lambda'(0)\rho'(1) > 1$, then with high probability, the stopping number is at most logarithmic with n [51].

Based on the flatness condition for the BEC, capacity-achieving sequences of LDPC standard ensembles must approach the flatness condition $\lambda'(0)\rho'(1) < 1/\epsilon$

with equality [54]. Since $\lambda'(0)\rho'(1) \approx 1/\epsilon > 1$, then the stopping number for capacity-approaching sequences must be sublinear [51].

These results show that the stopping number, which dominates performance in the error-floor region, grows only sublinearly with codelength for many codes, including all capacity-achieving sequences of LDPC standard ensembles for the BEC. Thus, in many cases, sublinear stopping sets dominate error-floor performance. This motivates the study of sublinear stopping sets, which will be presented in this chapter.

A few notes are in order. First, although some analysis of sublinear stopping sets is completed in [20] for determining the stopping number, the analysis there focuses on asymptotic behavior in the limit as codelength approaches infinity. The analysis does not provide insight into the sublinear stopping-set behavior in the practical, finite-length case.

Second, the analysis of sublinear stopping sets in [49] only applies to the case when the minimum variable-node degree is strictly greater than two.

Lastly, note that the analysis and conclusions regarding stopping number in [20, 49, 51] apply to *standard* LDPC ensembles. When protograph-based structure is enforced, the results may or may not be valid. In this case, the analysis of sublinear stopping sets is still important to characterize the sublinear stopping-set behavior and because the stopping number may still be sublinear with codelength for many protograph-based codes.

3.2 Ensemble Stopping-Set Enumerators

First, we present expressions for the stopping-set enumerators for both standard ensembles and protograph-based ensembles.

3.2.1 Standard Ensembles

First, consider the standard LDPC ensemble with maximum variable-node degree d_v, maximum check-node degree d_c, and rate R. Additional quantities describing properties of the code are defined in Section 1.1.2.

The expected number of stopping sets of size v in the standard ensemble of LDPC codes with codelength n is given by [30]

$$s(n,v) = \sum_{e=0}^{L'(1)} \frac{\text{coef}\{\prod_{i=1}^{d_v}(1+yx^i)^{L_i}, y^v x^e\}\,\text{coef}\{\prod_{i=1}^{d_c}[(1+x)^i - ix]^{R_i}, x^e\}}{\binom{L'(1)}{e}} \tag{3.4}$$

where e is the number of edges in the stopping set.

The first factor in the numerator represents the number of ways to choose v variable nodes such that there are exactly e edges emanating from them. Evaluating the first term in the numerator results in

$$A_v = \text{coef}\left\{\prod_{i=1}^{d_v}(1+yx^i)^{L_i}, y^v x^e\right\} = \sum_{\mathbf{k}\in\mathcal{S}_v}\prod_{i=1}^{d_v}\binom{nl_i}{k_i} \tag{3.5}$$

where

$$\mathcal{S}_v = \left\{(k_1,\ldots,k_{d_v}):\ 0\le k_i\le nl_i,\ \forall\, 1\le i\le d_v;\ \sum_{i=1}^{d_v}k_i = v;\ \sum_{i=1}^{d_v} ik_i = e\right\} \tag{3.6}$$

and k_i is the number of variable nodes in the stopping set with degree i.

The second factor in the numerator represents the number of ways to connect e edges to a subset of the check nodes such that a stopping set is formed, i.e., such that all check nodes are connected to at least two of the e edges or to none at all.

Evaluating the second term in the numerator results in

$$A_c = \text{coef}\left\{\prod_{i=1}^{d_c}[(1+x)^i - ix]^{R_i}, x^e\right\} = \sum_{\mathbf{n}\in\mathcal{S}_c}\prod_{i=1}^{d_c}\left\{\frac{(n(1-R)r_i)!}{\prod_{j=0,j\neq 1}^{i} n_{i,j}!}\prod_{\substack{j=0\\j\neq 1}}^{i}\binom{i}{j}^{n_{i,j}}\right\} \quad (3.7)$$

where

$$\mathcal{S}_c = \{\mathbf{n} : n_{i,j} \geq 0,\ \forall\, 2 \leq j \leq i \leq d_c; \quad (3.8)$$
$$\sum_{j=0}^{i} n_{i,j} = n(1-R)r_i,\ \forall\, 2 \leq i \leq d_c; \sum_{i=2}^{d_c}\sum_{j=2}^{i} j n_{i,j} = e,\ \forall\, 2 \leq i \leq d_c\},$$

$\mathbf{n}$ is the vector of all $n_{i,j}$ for $2 \leq j \leq i \leq d_c$, and $n_{i,j}$ represents the number of degree-i check nodes which are connected to the stopping set via j edges. Note that $n_{i,1}$ must always be zero in order to form a stopping set.

The denominator represents the number of ways to choose e edges out of the $L'(1)$ total edges in the graph.

$$A_d = \binom{L'(1)}{e} = \binom{n\sum_{i=1}^{d_v} i l_i}{e} \quad (3.9)$$

3.2.2 Protograph-Based Ensembles

Consider a protograph with M variable-node types and J check-node types connected via E edge types. This protograph is repeatedly replicated to create a total of Z copies of the protograph. Finally, for each edge-type e in the base protograph, the endpoints of the Z type-e edges are randomly permuted between the Z variable nodes and Z check nodes to which the Z edges are connected. The ensemble consisting of all such permutations is the protograph-based LDPC ensemble, and the codelength $n = ZM$.

To derive an expression for $s(n, v)$, we will first define several quantities for $i = 1, \ldots, M$; $j = 1, \ldots, J$; and $k = 1, \ldots, d_{c,j}$:

$$
\begin{aligned}
n_i &= \text{number of type-}i\text{ variable nodes in a set of } v \text{ variable nodes} \\
\mathbf{n} &= (n_1, n_2, \ldots, n_M) = \text{ vector of all } n_i\text{'s for } i = 1, \ldots, M \\
d_{v,i} &= \text{degree of the } i\text{th variable-node type} \\
d_{c,j} &= \text{degree of the } j\text{th check-node type} \\
\nu_{j,k} &= \text{variable-node type connected to the } k\text{th edge emanating from the} \\
&\qquad j\text{th check-node type in the protograph.}
\end{aligned}
$$

Note that $\nu_{j,k}$ and $\nu_{j,l}$ can be the same variable-node type for $k \neq l$ if a variable-node type and check-node type are connected by multiple edges in the protograph.

Theorem 3.1. *For a protograph-based LDPC ensemble generated from Z copies of a protograph with M variable-node types and J check-node types connected via E edge types, the expected number of stopping sets of size v is*

$$
s(n, v) = \sum_{\mathbf{n} \in \mathcal{S}_p} \left[\prod_{i=1}^{M} \binom{Z}{n_i}^{1-d_{v,i}} \times \prod_{j=1}^{J} \operatorname{coef} \left\{ \left[\prod_{k=1}^{d_{c,j}} (1 + x_k) - \sum_{k=1}^{d_{c,j}} x_k \right]^{Z}, \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\} \right] \tag{3.10}
$$

where the codelength $n = ZM$ and

$$
\mathcal{S}_p = \left\{ \mathbf{n} \in \mathbb{Z}^M : \, 0 \leq n_i \leq Z, \, \forall i \in \{1, \ldots M\}; \, \sum_{i=1}^{M} n_i = v \right\}. \tag{3.11}
$$

The set $\mathcal{S}_p$ contains all possible distributions of the variable-node types within a set of v variable nodes. In (3.10), the term $\prod_{i=1}^{M} \binom{Z}{n_i}$ represents the number of

ways to choose v variable nodes out of the Z protograph copies such that they follow the distribution in $\mathbf{n}$. The term $\prod_{i=1}^{M} \binom{Z}{n_i}^{d_{v,i}}$ represents the total number of edge permutations possible for the edges emanating from these v variable nodes. Finally, the coefficient term in (3.10) represents the number of ways to connect the v variable nodes to check nodes of type j such that a stopping set is formed.

Proof. See Appendix B. □

Equation (3.10) is an exact expression and can be shown to be equivalent to the expression in [50]. By expressing the enumerator with the coefficient term in (3.10), we are able to describe a closed-form combinatorial expression which is then approximated in Section 3.3. In [50], the equivalent term is calculated recursively using multinomial z-transforms.

3.3 Enumerator Approximations for Sublinear-Sized Stopping Sets

From the expressions in Section 3.2, the ensemble stopping-set enumerators for sublinear-sized stopping sets will be approximated for large n. The approximations will provide insight into how the enumerators scale with codelength. Results will be presented for both standard ensembles and protograph-based ensembles.

3.3.1 Standard Ensembles

For standard ensembles, the combinatorial expressions in (3.4) can be approximated as n approaches infinity to obtain the following theorem.

Theorem 3.2. *For a standard LDPC ensemble with $d_c > 2$, codelength n, and stopping-set size v such that*

$$v \leq xn \min_{\substack{1 \leq i \leq d_v : l_i \neq 0 \\ 1 \leq j \leq d_c : r_j \neq 0}} \left\{ l_i, \frac{2}{d_v}(1-R)r_j \right\}, \tag{3.12}$$

for any constant $x \in [0, 1/d_v)$, the expected number of stopping sets of size v is approximated by

$$s(n, v) = \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v - \lceil e/2 \rceil \pm \frac{O(v \log v)}{\log n}}. \tag{3.13}$$

Proof. See Appendix C. □

The requirement that $d_c > 2$ is not very restrictive since it is typically met by good LDPC codes. The condition on v in (3.12) is also not restrictive since we are only investigating sublinearly-sized stopping sets here. Although the condition (3.12) says that the expression in (3.13) is valid for stopping-set sizes up to a linear proportion of n, the approximation in (3.13) is only useful up to a sublinear proportion of n for which the big-O term is insignificant compared to the rest of the exponent. For example, the big-O term will be insignificant for stopping-set sizes that grow at most logarithmically with n but not for stopping-sizes that grow proportionally to n^x for any $x > 0$.

When the stopping-set size v is a finite, fixed constant, then the theorem simplifies.

Corollary 3.1. *For a standard LDPC ensemble with a finite, fixed stopping-set size v, $d_c > 2$, and codelength n satisfying*

$$n > \frac{d_v v}{\min_{\substack{1 \leq i \leq d_v : l_i \neq 0 \\ 1 \leq j \leq d_c : r_j \neq 0}} \left\{ l_i, \frac{2}{d_v}(1-R)r_j \right\}}, \tag{3.14}$$

the expected number of stopping sets of size v is approximated by

$$s(n,v) = \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} O(1). \tag{3.15}$$

Proof. Since v is a finite, fixed constant, we can find an n large enough to satisfy the condition in (3.12). Formally, for all n such that (3.14) is satisfied,

$$v < \frac{1}{d_v} n \min_{\substack{1 \le i \le d_v : l_i \neq 0 \\ 1 \le j \le d_c : r_j \neq 0}} \left\{ l_i, \frac{2}{d_v}(1-R) r_j \right\} \tag{3.16}$$

and hence,

$$v \le xn \min_{\substack{1 \le i \le d_v : l_i \neq 0 \\ 1 \le j \le d_c : r_j \neq 0}} \left\{ l_i, \frac{2}{d_v}(1-R) r_j \right\}$$

for some $x \in [0, 1/d_v)$. Thus, we can apply Theorem 3.2 to find

$$s(n,v) = \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} n^{\pm \frac{O(v \log(v))}{\log n}}. \tag{3.17}$$

The big-O notation in the last term indicates that there exists a positive constant α and an integer n_0 such that for all $n > n_0$,

$$0 \le O(v \log v) \le \alpha v \log v. \tag{3.18}$$

Thus, we can bound that last term of (3.17) as follows:

$$n^{O(v \log v)/\log n} \le n^{\alpha v \log v / \log n} = n^{\log_n v^{\alpha v}} = v^{\alpha v} \tag{3.19}$$

and

$$n^{-O(v \log v)/\log n} \ge n^{-\alpha v \log v / \log n} = n^{\log_n v^{-\alpha v}} = v^{-\alpha v}. \tag{3.20}$$

Thus,

$$v^{-\alpha v} \leq n^{\pm O(v \log v)/\log n} \leq v^{\alpha v}. \tag{3.21}$$

Since v is a finite, fixed constant, these upper and lower bounds are positive constants, so $n^{\pm O(v \log v/\log n)} = O(1)$. Hence,

$$s(n, v) = \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} O(1). \tag{3.22}$$

□

Applying Theorem 3.2 to stopping sets which grow at most logarithmically with codelength yields the following corollary.

Corollary 3.2. *For a standard LDPC ensemble with $d_c > 2$, stopping-set size $v \leq \beta \log n$ for any constant $\beta > 0$, and codelength n large enough such that*

$$\frac{\log n}{n} < \frac{1}{\beta d_v} \min_{\substack{1 \leq i \leq d_v : l_i \neq 0 \\ 1 \leq j \leq d_c : r_j \neq 0}} \left\{ l_i, \frac{2}{d_v}(1 - R) r_j \right\}, \tag{3.23}$$

the expected number of stopping sets of size v is approximated by

$$s(n, v) = \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil \pm O(\log [\log n])}. \tag{3.24}$$

When $v = \beta \log n$, the big-O term in (3.24) grows proportionally with $\log v$ and hence is insignificant compared to the rest of the exponent for large n. Also, note that there exists a $\beta > 0$ small enough such that the condition in (3.23) is satisfied for all $n \geq 1$ and hence, (3.24) holds for all $n \geq 1$ for this β value.

Proof. Rearranging the condition in (3.23), we obtain

$$\beta \log n < \frac{n}{d_v} \min_{\substack{1 \le i \le d_v : l_i \neq 0 \\ 1 \le j \le d_c : r_j \neq 0}} \left\{ l_i, \frac{2}{d_v}(1-R) r_j \right\}. \tag{3.25}$$

Thus,

$$v \le \beta \log n \le xn \min_{\substack{1 \le i \le d_v : l_i \neq 0 \\ 1 \le j \le d_c : r_j \neq 0}} \left\{ l_i, \frac{2}{d_v}(1-R) r_j \right\} \tag{3.26}$$

for some constant $x \in [0, 1/d_v)$. Applying Theorem 3.2 and noting that (3.26) gives the relation

$$\begin{aligned} \frac{O(v \log v)}{\log n} &= \frac{O(\beta \log n \log [\beta \log n])}{\log n} = O(\beta \log [\beta \log n]) \\ &= O(\beta \log \beta + \beta \log[\log n]) = O(\log[\log n]) \end{aligned} \tag{3.27}$$

yields the final result:

$$s(n, v) = \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v - \lceil e/2 \rceil \pm O(\log [\log n])}. \tag{3.28}$$

□

3.3.2 Protograph-Based Ensembles

Using similar techniques as for standard ensembles, $s(n, v)$ will be approximated for protograph-based ensembles as follows. Let $\mathcal{V}$ be the set of all variable nodes in the graph and fix a subset $\mathcal{V}_v \in \mathcal{V}$ such that the variable-node types in $\mathcal{V}_v$ are distributed according to $\mathbf{n}$, where $\mathbf{n}$ is a vector in $\mathcal{S}_p$. Now, several quantities will be

defined for a particular check-node type j. Let

$$\mathcal{B} = \left\{ (b_1, b_2, \ldots, b_{d_{c,j}}) \in \{0,1\}^{d_{c,j}} : \sum_{k=1}^{d_{c,j}} b_k \neq 1 \right\}. \tag{3.29}$$

This set represents the possible connections a single check node of type j can have to $\mathcal{V}_v$ such that it is connected at least twice or not at all. Specifically, for each $k \in \{1, \ldots, d_{c,j}\}$, b_k is 1 if the kth edge emanating from the check node is connected to $\mathcal{V}_v$ and is 0 otherwise. The size of the set $\mathcal{B}$ is

$$|\mathcal{B}| = 2^{d_{c,j}} - d_{c,j}. \tag{3.30}$$

Enumerate the elements in $\mathcal{B}$ as $\{\beta_0, \beta_1, \ldots, \beta_{|\mathcal{B}|-1}\}$ where β_0 is the all-zero vector and all other vectors are enumerated in any fashion. Let $\beta_{h,k}$ be the kth element of β_h. Let m_h be the number of check nodes of type j whose connections are described by β_h and let $\mathbf{m} = (m_0, \ldots, m_{|\mathcal{B}|-1})$ be the vector of all such quantities. Finally, for a given $\mathbf{m}$, let w_j be the number of check nodes of type j which are connected to $\mathcal{V}_v$. With these definitions, we can now express an approximation to $s(n, v)$.

Theorem 3.3. *For an LDPC ensemble based on Z copies of a protograph with M variable-node types, J check-node types, E edge types, codelength $n = ZM$, and stopping-set size $v < n$, the expected number of stopping sets of size v is approximated by*

$$s(n, v) = \sum_{\mathbf{n} \in \mathcal{S}_p} Z^{v - e + w^* \pm \frac{O(v \log v)}{\log n}} \tag{3.31}$$

where e is the number of edges emanating from the set of variable nodes, $\mathcal{V}_v$, which is distributed according to $\mathbf{n}$, and $\mathcal{S}_p$ is defined in (3.11)*. The quantity $w^* = \sum_{j=1}^{J} w_j^*$*

where $w_j^* = \max_{\mathbf{m}\in\mathcal{S}_m}\{w_j\}$ *and*

$$\mathcal{S}_m = \Bigg\{ \mathbf{m} \in \mathbb{Z}^{|\mathcal{B}|} : \; 0 \le m_h \le Z, \; \forall h = 0, \ldots, |\mathcal{B}| - 1; \; \sum_{h=0}^{|\mathcal{B}|-1} m_h = Z; \\ \sum_{h=0}^{|\mathcal{B}|-1} \beta_{h,k} m_h = n_{\nu_{j,k}}, \; \forall k = 1, \ldots, d_{c,j} \Bigg\} \tag{3.32}$$

for each $j \in \{1, \ldots, J\}$.

Proof. See Appendix D. □

The quantity w^* in Theorem 3.3 represents the largest number of check nodes possible in a subgraph induced by a stopping set distributed according to $\mathbf{n}$. The difficulty in applying Theorem 3.3 is in determining w^* for a given v and n.

Similar to the standard-ensemble case in Section 3.3.1, Theorem 3.3 for protograph-based codes simplifies when the stopping-set size is a finite, fixed constant or grows at most logarithmically with codelength, as shown in the next two corollaries.

Corollary 3.3. *For a protograph-based LDPC ensemble with a finite, fixed stopping-set size* v *and codelength* $n > v$*, the expected number of stopping sets of size* v *is approximated by*

$$s(n, v) = \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} O(1). \tag{3.33}$$

Corollary 3.4. *For a protograph-based LDPC ensemble with codelength* n*, stopping-set size* $v \le \beta \log n$ *for any constant* $\beta > 0$*, and* n *large enough such that* $\log n/n < 1/\beta$*, the expected number of stopping sets of size* v *is approximated by*

$$s(n, v) = \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^* \pm O(\log[\log n])}. \tag{3.34}$$

When $v = \beta \log n$, the big-O term in (3.34) grows proportionally with $\log v$ and

hence is insignificant compared to the rest of the exponent for large n. Also, note that there exists a $\beta > 0$ small enough such that the condition $\log n/n < 1/\beta$ is satisfied for all $n \geq 1$ and hence, (3.34) holds for all $n \geq 1$ for the β value.

The proofs of Corollary 3.3 and Corollary 3.4 parallel the proofs of Corollary 3.1 and Corollary 3.2, respectively, for the standard ensemble (see Section 3.3.1), and hence the proofs are omitted here.

3.4 Insights into Sublinear Stopping-Set Enumerator Behavior

In this section, we discuss the insights provided by the enumerator approximations and compare standard ensembles with protograph-based ensembles.

Theorems 3.2 and 3.3 show that the enumerator approximations, for stopping sets that grow at most logarithmically with codelength, have a polynomial relationship with the codelength n. In contrast, the asymptotic analysis in [19, 50] shows that the enumerator follows an exponential behavior with codelength, for linearly-growing stopping sets. This difference indicates that our analysis captures sublinear behavior of the stopping sets which could not be captured in [19, 50]. Thus, the results here help to more fully capture stopping-set behavior and may be more useful in gaining insight for finite codelengths, particularly for short codes where the dominating stopping-set size $v = \delta_{s,min} n$ in the linear analysis may be too small (e.g., small enough to be avoided by an intelligent permutation algorithm) to provide meaningful results.

We define two categories of behavior: category P contains stopping sets which follow the polynomial behavior in Theorems 3.2 and 3.3, which includes finite stopping sets and stopping sets which grow at most logarithmically with codelength.

Category E contains stopping sets which follow the exponential behavior in [19, 50], which includes stopping sets which grow linearly with codelength.

For standard ensembles with stopping sets in category P, Theorems 3.1 and 3.2 show that the exponent of n is dominated by $v - \lceil e/2 \rceil$ and this exponent decreases with increasing e, where e is the number of edges in the stopping set. Thus, the dominating term in the summation will be the one with the smallest value of e such that a stopping set can be formed with v variable nodes and e edges. This value of e will be denoted by e_{min} and the corresponding dominating exponent will be denoted by

$$E_S = \lceil e_{min}/2 \rceil - v \tag{3.35}$$

such that

$$s(n, v) \approx c n^{-E_S} \tag{3.36}$$

for some constant c and for large n. These results show that $s(n, v)$ grows proportionally with n^{-E_S} and hence, is dependent on n polynomially. Calculations on various code ensembles agree with this result (for example, see Section 3.6).

For standard ensembles with finite v and $n \geq v/l_{d_{v,min}}$, then $e_{min} = v d_{v,min}$ where $d_{v,min}$ is the minimum variable-node degree in the graph. Hence, $E_S = \lceil d_{v,min} v/2 \rceil - v$. This indicates that improved error-floor performance can be obtained by increasing the minimum variable-node degree and hence increasing E_S. Thus, $d_{v,min}$ is the *one* code characteristic which dominates the error-floor performance of $s(n, v)$ for all $v \leq \beta \log n$ satisfying the conditions given in Corollary 3.2.

A special case occurs when $d_{v,min} = 2$. In this case, $E_S = 0$ and thus the expected number of stopping sets of size v does not decay polynomially with n for any value of v. This implies that there always exists a significant probability of small stopping sets containing degree-2 variable nodes.

Another special case occurs when $v = 1$. In this case, we are examining the trivial case when a single variable node has multiple connections to each check node to which it is connected. In practice, this case will always be avoided in any practical design. Taking the dominating term of the probability of block error for the LDPC code ensemble to be the term corresponding to the dominating term in the $s(n, v)$ expression for $v = 1$ (since this is the most probable scenario), the resulting expression agrees with the expression given in Theorem 16 in [20]. Here, we were able to provide a more general expression and intuition behind that expression.

If a capacity-achieving sequence of LDPC code ensembles satisfies the flatness condition [54], then $\lambda'(0)\rho'(1) \to 1/\epsilon$. It has been conjectured that all capacity-achieving sequences must satisfy the flatness condition, but this has yet to be proven. Since $\lambda'(0) = \lambda_2$, the flatness condition says that λ_2 must approach $1/(\epsilon\rho'(1)) \neq 0$ in order to obtain good threshold performance. Thus, codes with good thresholds need degree-2 variable nodes. On the other hand, $\lambda'(0)\rho'(1) \to 1/\epsilon > 1$, so Theorem 3.2 applies and thus, good error-floor performance required high $d_{v,min}$. These two requirements are in direct conflict with each other, indicating that there may indeed be a fundamental tradeoff between threshold and error-floor performance, at least for standard ensembles (if the flatness condition is required for capacity-achieving sequences).

Fig. 3.1 shows an example of this tradeoff for rate 1/2 LDPC standard ensembles. As discussed above, the error-floor behavior is characterized by $d_{v,min}$. Ideally, an ensemble will have both large $d_{v,min}$ and large threshold erasure probability ϵ. For the regular codes, the tradeoff is clearly seen in Fig. 3.1. This leads to the question, "Can irregular LDPC code ensembles do better?" Except when $d_{v,min} = 2$, attempts at irregular LDPC code optimization (using differential evolution on the degree polynomials) with a fixed $d_{v,min}$ resulted in no improvement over the regular codes. The

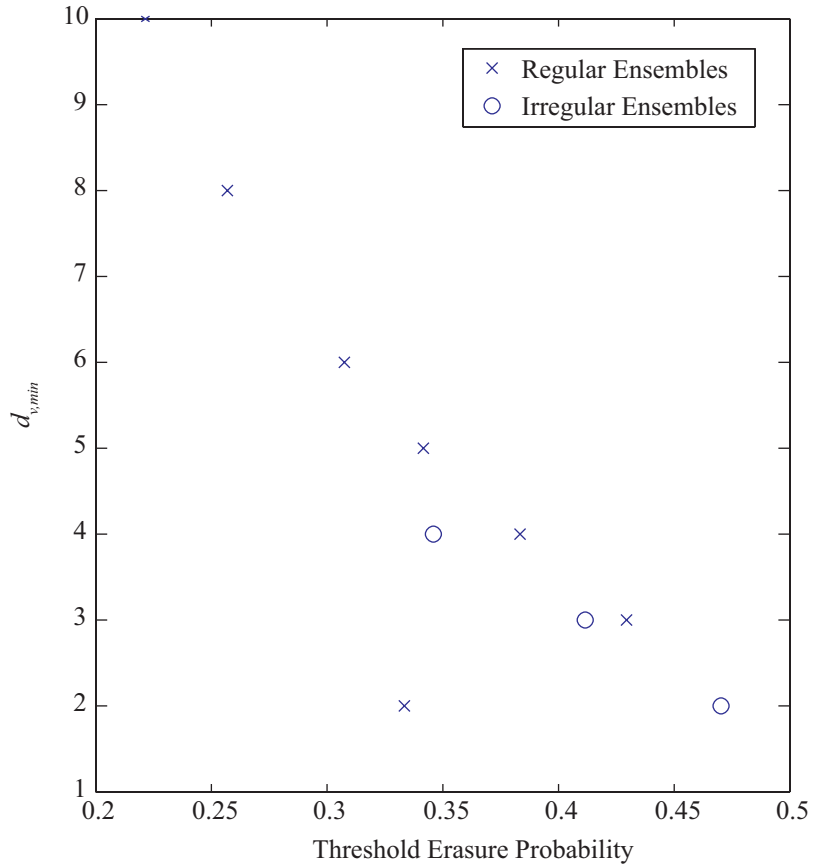


Figure 3.1: Error-floor vs. threshold performance for several rate-1/2 LDPC standard ensembles. Error-floor performance is characterized by $d_{v,min}$ and threshold performance is characterized by the largest erasure probability ϵ such that density evolution converges to arbitrarily small error probability.

case when $d_{v,min} = 2$ is unique since the regular (2,4) ensemble behaves poorly. The best irregular ensemble with $d_{v,min} = 2$ displayed in Fig. 3.1 is the optimized code from [11].

This example agrees with the conjecture that degree-2 variable nodes are necessary to achieve high thresholds. To simultaneously improve the error-floor performance, the quantity e_{min}/v must increase above the value of $d_{v,min}$ for a given v. This is where protograph-induced structure is beneficial, as will be shown below.

For protograph-based ensembles, the dominating term in the approximation for

$s(n, v)$ in Theorem 3.3 is n^{-E_P} where

$$E_P = e^* - v \tag{3.37}$$

and

$$e^* = \min_{\mathbf{n} \in \mathcal{S}_p} \{e - w^*\}. \tag{3.38}$$

Thus,

$$\lim_{n \to \infty} s(n, v) = n^{-E_P}. \tag{3.39}$$

Observe that $w^* \leq \lfloor e/2 \rfloor$ since the number of check nodes connected to a stopping set is at most the number of edges emanating from the stopping set divided by 2, i.e., when each check node is connected via exactly two edges (with one check node connected via three edges if e is odd). Thus,

$$\begin{aligned} e^* &= \min_{\mathbf{n} \in \mathcal{S}_p} \{e - w^*\} \geq \min_{\mathbf{n} \in \mathcal{S}_p} \{e - \lfloor e/2 \rfloor\} = \min_{\mathbf{n} \in \mathcal{S}_p} \{\lceil e/2 \rceil\} \\ &= \min_{\mathbf{n} \in \mathcal{S}_p} \left\lceil \sum_{i=1}^{M} n_i d_{v,i}/2 \right\rceil \geq \min_{\mathbf{n} \in \mathcal{S}_p} \left\lceil \sum_{i=1}^{M} \left[n_i d_{v,min} \right]/2 \right\rceil = \lceil d_{v,min} v/2 \rceil \,, \end{aligned}$$

and hence, a direct comparison between the exponents E_P and E_S can be made:

$$E_P = e^* - v \geq \lceil d_{v,min} v/2 \rceil - v = E_S. \tag{3.40}$$

Since large exponents E_P and E_S are desirable to make the exponent of n more negative, this result shows that protograph-based ensembles perform at least as well as standard ensembles. Further, protograph-based ensembles have the possibility for achieving exponents which are strictly more negative. To maximize E_P in protograph-based ensembles, we need to minimize w^*—the maximum number of check nodes in a subgraph induced by a stopping set $\mathcal{V}_v$—which is dependent solely

on the structure of the protograph. Intuitively, the structure of the protograph enforces structure so that we may not be able to simply connect exactly two edges to each of the $\lfloor e/2 \rfloor$ check nodes (with one check node connected to 3 edges if e is odd). Since the protograph structure limits the possible check nodes which the variable nodes in $\mathcal{V}_v$ can connect to, such a scheme using $\lfloor e/2 \rfloor$ check nodes may not result in a valid induced subgraph of $\mathcal{V}_v$. Thus, the number of check nodes needed to form a stopping set with $\mathcal{V}_v$ can be strictly smaller than for standard ensembles and hence, protograph-based ensembles can have strictly more negative exponents, resulting in better error-floor performance.

Additional insight provided by Theorems 3.2 and 3.3 includes the following. For finite stopping sets,

- if $(v - e + w^*) < 0$, then the expected number of such stopping sets can be made arbitrarily small by choosing a large enough n, or
- if $(v - e + w^*) \geq 0$, then the expected number of such stopping sets is lower bounded by a positive finite number for all n.

If $d_{v,min} \geq 3$, as in [55], then $(v - e + w^*) < 0$ for all stopping sets in the ensemble. Thus, the expected number of all finite-sized stopping sets can be made arbitrarily small by choosing a large enough n, and such codes are guaranteed to have good error-floor performance for category-P stopping sets.

The above results on stopping-set enumerators for category-P stopping sets lead to several questions. First, can we expurgate the ensemble or pick a specific code from the ensemble such that the dominating stopping-set size is increased? Further, can we find codes where the stopping number grows linearly with codelength, i.e., $v \geq \delta n$ for all stopping sets in the code for a given constant δ?

In [28], expurgation is accomplished by fixing the value of certain variable nodes in a code ensemble to achieve a target stopping number. However, is it possible to

strategically design ensembles through protographs to increase the stopping number? Intuitively, we can see that the additional structure imposed by the protograph can render certain sized stopping sets impossible. For example, by simply avoiding any multiple connections in the protograph, i.e., connecting each edge from a given variable-node type to different check-node types, all stopping sets of size 1 are eliminated.

In Section 3.6, we will discuss an example in detail to illustrate the various insights provided by the ensemble stopping-set enumerators. First, we examine when the approximations of Section 3.3 are valid.

3.5 Region of Approximation Validity

To help determine whether stopping sets behave polynomially or exponentially, we examine when the approximations of Section 3.3 are valid. Specifically, for what values of stopping-set size v and codelength n do the approximations apply? We will only present the analysis for protograph-based ensembles for which at most one edge type connects any variable-node type to any check-node type in the protograph. For all other protographs, one can simply expand the protograph to meet this condition and then apply the analysis given here.

Given $\mathbf{n} \in \mathcal{S}_p$ and $\mathbf{m} \in \mathcal{S}_m$, the corresponding term in the stopping-set enumerator in Theorem 3.1 can be expressed as

$$\ln s_{\mathbf{n},\mathbf{m}}(n, v) = \ln c + (v - e + w^*) \ln n + A \tag{3.41}$$

where

$$c = \prod_{i=1}^{M} \left[\frac{1}{n_i!}\right]^{1-d_{v,i}} \prod_{j=1}^{J} \frac{1}{\prod_{h=1}^{|\beta|-1} m_h!} M^{-(v-e*w^*)} \tag{3.42}$$

is independent of n and

$$A = \sum_{i=1}^{M} \left[\sum_{k=1}^{n_i - 1} \ln \left(1 - \frac{k}{Z} \right)^{(1-d_{v,i})} \right] + \sum_{j=1}^{J} \sum_{k=1}^{w_j - 1} \ln \left(1 - \frac{k}{Z} \right). \tag{3.43}$$

Comparing the error term $|A|$ to $|(v-e+w^*)\ln n|$ will show when the approximation in Theorem 3.3 is valid.

Theorem 3.4. *Consider a protograph-based ensemble for which at most one edge type connects any variable-node type to any check-node type in the protograph. Given a small fraction $\gamma > 0$ and any constant $a \in (0,1)$, let N_0 be the solution of n in the following equation:*

$$n \ln n = \frac{Mv}{\gamma} \left(1 + \frac{a}{2(1-a)^2} \right) \left(1 + \frac{1}{d_{v,avg} - 2} \right) \tag{3.44}$$

where $d_{v,avg}$ is the smallest average variable-node degree in stopping sets of size v. Then, for all v and n satisfying the conditions $v/n \leq a/M$ and $n > N_0$, the error term is upper bounded by

$$|A| \leq \gamma |(v - e + w^*) \ln n| \tag{3.45}$$

and hence, for small γ, the stopping-set enumerator can be approximated by

$$\ln s_{\mathbf{n},\mathbf{m}}(n, v) \approx \ln c + (v - e + w^*) \ln n. \tag{3.46}$$

Proof. See Appendix E. □

Since v is proportional to $N_0 \ln N_0$, the ratio of v/N_0 monotonically increases as v increases. Also, there may exist values of $n < N_0$ such that the upper bound on the error term still holds. Additional work to tighten the bounds used to generate the

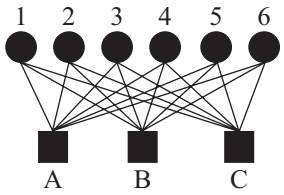


Figure 3.2: Protograph P analyzed in Section 3.6. The protograph contains 6 variable-node types (circles) and 3 check-node types (squares).

approximations are necessary to capture a larger portion of the region of validity.

3.6 A Detailed Example Illustrating Insights Provided by Enumerator Approximations

Comparing the regular (3,6) LDPC standard ensemble to the ensemble generated by protograph P in Fig. 3.2 provides an illustrative example of insights provided by the sublinear analysis in Sections 3.2-3.5. The protograph-P ensemble is a subset of the regular (3,6) standard ensemble. Among other restrictions, protograph P's structure prevents size-1 stopping sets from forming.

Examining the behavior of linearly-growing stopping sets (those that grow linearly with codelength) as in [19], both ensembles have the same $\delta_{s,min}$ value. However, the category-P stopping-set behavior is different for the two ensembles, as will be shown next.

From Theorems 3.1 and 3.3, the exponents E_S and E_P, respectively, can be calculated for category-P stopping sets as follows. First, note that these exponents correspond directly to the negative slope of a log-log plot of $s(n, v)$ versus n. For a given stopping-set size v, the total number of edges in the stopping set is $3v$. In order to form a stopping set with the largest number of check nodes, we need to spread out the edges as much as possible among the check nodes while still satisfying the

constraint that all check nodes must have either zero or at least two connections to the stopping set.

In protograph P, there are three types of check nodes: type A, type B, and type C. First, consider check nodes of type A. Since each of the v variable nodes in the stopping set has a single connection to a type-A check node, the type-A check nodes must be connected to the stopping set via v edges. The maximum number of type-A check nodes which can connect to the stopping set via v edges, with each check node connected via zero or at least two edges, is $\lfloor v/2 \rfloor$. The same analysis applies to check nodes of type B and C, as well. Thus, the total maximum number of check nodes connected to a stopping set of size v is $w^* = 3\lfloor v/2 \rfloor$, and the resulting exponent calculated from Theorem 3.3 is

$$E_P(v) = -v + 3v - 3\left\lfloor \frac{v}{2} \right\rfloor = 2v - 3\left\lfloor \frac{v}{2} \right\rfloor. \tag{3.47}$$

For the regular (3,6) standard ensemble, the exponent can be calculated directly from Theorem 3.1:

$$E_S(v) = -v + \left\lceil \frac{3v}{2} \right\rceil = \left\lceil \frac{v}{2} \right\rceil. \tag{3.48}$$

This can also be seen by observing that all check nodes in the standard ensemble are of the same type. Since there is no distinction between the check nodes, the $3v$ edges can be connected to any of the check nodes, as long as each check node is connected zero or at least two times. Thus, the maximum number of check nodes connected to the stopping set is $w^* = \lfloor 3v/2 \rfloor$. Applying (3.37) and (3.38) gives the same exponent as in (3.48).

Comparing the two results, we see that when v is even, the exponents are the

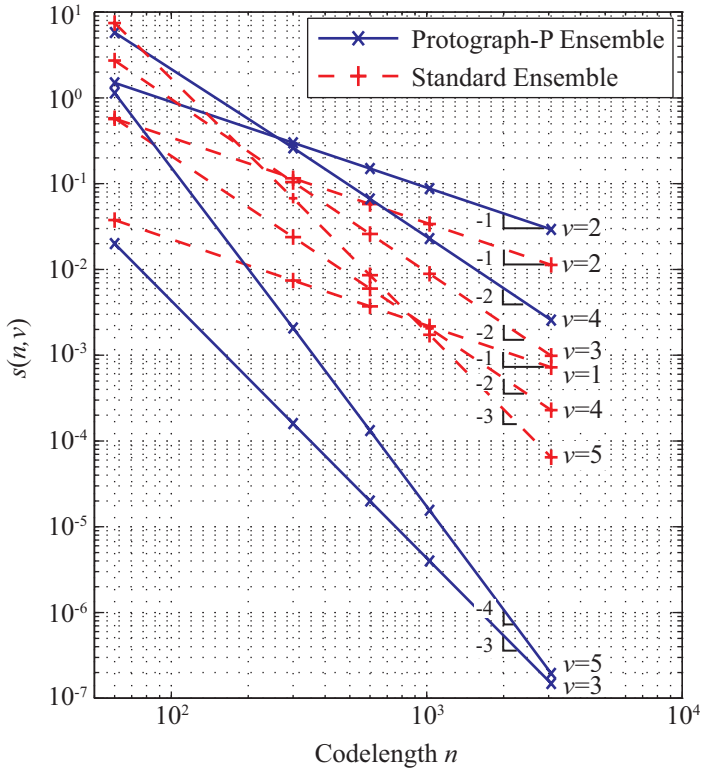


Figure 3.3: Ensemble stopping-set enumerator vs. codelength for the regular (3,6) standard ensemble and the protograph-P ensemble.

same for both ensembles. However, when v is odd, E_P is exactly one more than E_S:

$$E_P(v) - E_S(v) = 2v - 3\left\lfloor \frac{v}{2} \right\rfloor - \left\lceil \frac{v}{2} \right\rceil = 2v - 3\frac{v-1}{2} - \frac{v+1}{2} = 1. \tag{3.49}$$

Thus, in terms of the exponents for odd v, protograph P is strictly better than the standard ensemble.

The ensemble stopping-set enumerators for the regular (3,6) standard ensemble and the protograph-P ensemble are shown in Fig. 3.3 as a function of codelength n for stopping-set sizes $v = 1$ to 5. The values plotted are exact, numerically evaluated values calculated from Equation (3.4) and Theorem 3.1. Note that protograph P has no stopping sets of size one. Fig. 3.3 provides some key observations.

First, the stopping-set enumerators follow the category-P behavior: the log-log

plot is linear and the slope agrees with the exponent values E_P and E_S calculated above.

Second, protograph P greatly reduces the number of stopping sets of odd size. For odd v, the log-log slope was both analytically calculated and numerically evaluated to be one less for the protograph-P ensemble than for the standard ensemble. For even v, the protograph-P ensemble has the same log-log slope but higher offset than the standard ensemble. This leads to several questions: (1) is there some conservation of stopping-set numbers and (2) are there regions where the effect of the offset will dominate over the effect of the slope and vice versa?

Third, the linear behavior in the log-log domain predicted for category-P stopping sets seems to hold even for very small codelengths, e.g., $n \approx 10v$. Since the v/n ratio of 0.1 is significantly (one order of magnitude) larger than the $\delta_{s,min}$ values of 0.01-0.03 in [19, 50], this suggests that category-P behavior may extend to stopping-set sizes which grow faster than logarithmically with codelength. Further, the smallest stopping set in the ensemble may fall into the range where category-P behavior is applicable, which would mean that investigations into category-E stopping sets may not be the critical analysis. However, these are simply conjectures based on a couple ensemble examples with low v and n values. It is unclear whether this pattern will hold for larger v and n values and for all ensembles.

Lastly, if we fix $\delta = v/n$ and trace the behavior as n grows, we begin to see different trends in the plot, as shown in Fig. 3.4. The different behavior for even and odd stopping set sizes is apparent from this figure. Also, there appears to be an increase in the enumerators when $v = 0.1n$ and a decrease when $v = 0.01n$. This generally agrees with the trends predicted for category-E stopping sets in [19, 50].

Applying Theorem 3.4 to protograph P provides an illustrative example of when the approximation in Theorem 3.3 holds. Specifically, using $a = 0.25$ and $\gamma = 0.02$,

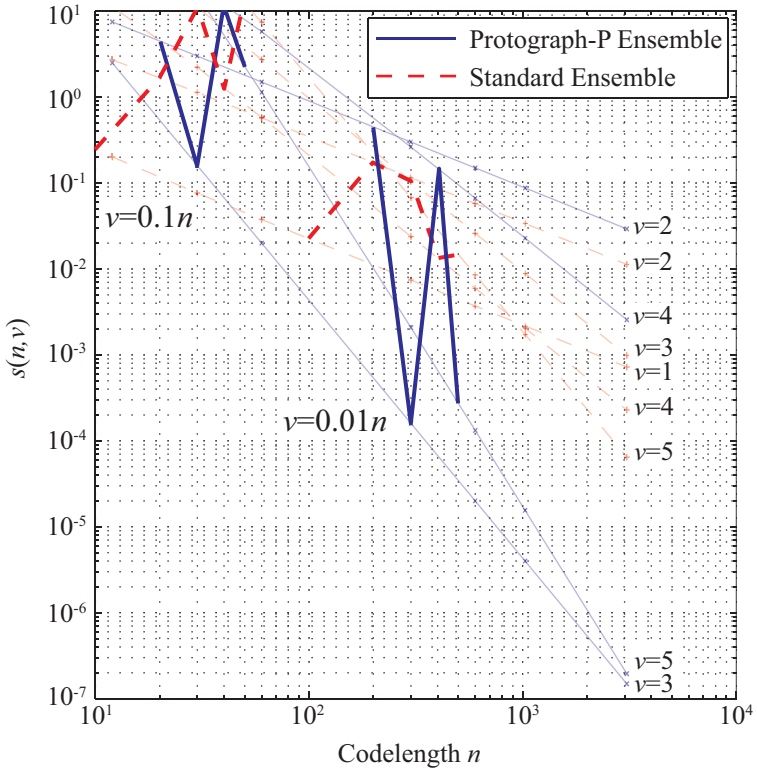


Figure 3.4: Trace of stopping-set enumerators in Fig. 3.3 for stopping sets growing linearly with codelength. The stopping sets have size $v = 0.1n$ and $v = 0.01n$.

we find the values of stopping-set size v and codelength n for which the approximation is accurate within 2%. Applying Theorem 3.4, we obtain values for N_0 in Table 3.1. This table shows, for example, that for stopping sets of size $v = 2$, the stopping sets follow category-P behavior for all codelengths $n \geq N_0 = 104$, i.e., for all ratios $v/n \leq v/N_0 = 0.019$. Since v/N_0 increases monotonically with v, all stopping sets of size $v \leq 0.019n$ follow category-P behavior for all $n \geq 104$.

Based on the analysis of category-E stopping sets in [19,50], the largest v/n ratio resulting in a negative stopping-set enumerator exponent is $\delta_{s,min} = 0.018$. Thus, according to this category-E analysis, the stopping-set sizes of interest are around $0.018n$. However, Table 3.1 shows that for all $n \geq 104$, these stopping sets actually follow category-P behavior.

This example has shown that category-P analysis is necessary to more fully cap-

Table 3.1: Bounds on the Region of Enumerator-Approximation Validity for Stopping Sets of Size v in the Protograph-P LDPC Ensemble

v	N_0	v/N_0
1	59	0.017
2	104	0.019
3	145	0.021
4	184	0.022
5	222	0.023
10	399	0.025
20	726	0.028
50	1617	0.031
100	2986	0.034
200	5543	0.036
500	12647	0.040

ture the stopping-set enumerator behavior.

3.7 Calculating Enumerator Exponents for Protograph-Based Ensembles

Theorem 3.3 showed that the polynomial enumerator exponent for protograph-based ensembles is $v - e + w^*$ where w^* represents the largest possible number of check nodes connected to a stopping set characterized by the variable-node type distribution $\mathbf{n}$. The quantity w^* is dependent on the structure of the protograph and hence, in general, it is difficult to evaluate exactly. In this section, we will show that an approximation to the enumerator exponent can be easily evaluated with linear/integer programming.

3.7.1 Evaluating w^*

First, we provide a method for calculating w^*, the maximum number of check nodes in a subgraph induced by a stopping set characterized by the variable-node

distribution $\mathbf{n}$, based on the stopping-set and protograph parameters.

Due to the protograph structure, connections to check nodes of type j are independent of check nodes of all other types. Thus, we can consider each check-node type j independently, and for each j, we evaluate w_j^*, the maximum number of type-j check nodes in a subgraph induced by a stopping set characterized by $\mathbf{n}$.

For the analysis, we assume that the maximum edge multiplicity in the protograph is one, i.e., each variable-node type can be connected at most once to each check-node type in the protograph. If this assumption does not hold, then we can simply expand the protograph, by the maximum edge multiplicity, such that the assumption is satisfied and then, apply the analysis.

Let $\nu_{i,j} \in \{0, 1\}$ be the number of edge connections between variable-node type i and check-node type j in the protograph.

Theorem 3.5. *Consider a set $\mathcal{V}$ of variable nodes with distribution $\mathbf{n} = (n_1, \ldots, n_M)$ where n_i is the number of type-i variable nodes in the set.*

For each check-node type j, if

$$2 \max_{1 \le i \le M} n_i \nu_{i,j} \le \sum_{i=1}^{M} n_i \nu_{i,j}, \tag{3.50}$$

then we can connect type-j check nodes to $\mathcal{V}$ such that $\mathcal{V}$ is a stopping set. Further,

$$w_j^* = \lfloor e_j/2 \rfloor \tag{3.51}$$

where $e_j = \sum_{i=1}^{M} n_i \nu_{i,j}$ is the total number of edges connected to type-j check nodes.

If (3.50) is not satisfied for any check-node type j, then $\mathcal{V}$ cannot be a stopping set.

Proof. See Appendix F. □

To see the intuition behind the role of (3.50) in Theorem 3.5, first rearrange the inequality (3.50), which is expressed in a form for easy evaluation, into a more intuitive form:

$$n_k \nu_{k,j} \leq \sum_{i=1, i \neq k}^{M} n_i \nu_{i,j} \tag{3.52}$$

where $k = \arg\max_{1 \leq i \leq M} n_i \nu_{i,j}$, i.e., the number of connections from type-k variable nodes to type-j check nodes must be less than or equal to the number of connections from all other variable-node types to type-j check nodes. Next, recall from Section 1.1.6 that in order to form a valid stopping set, each check node must be connected to $\mathcal{V}$ at least twice or not at all. Since each edge type in the protograph is permuted independently of all other edge types, the connections to each check-node type j can be considered separately from all other check-node types. Thus, fix a check-node type j. Then, for each variable-node type k which is connected to check-node type j in the protograph, i.e., $\nu_{k,j} = 1$, the n_k type-k variables nodes in $\mathcal{V}$ are connected to n_k unique type-j check nodes due to the protograph expansion. In order for these n_k type-j check nodes to be connected to $\mathcal{V}$ at least twice, there must be at least n_k other variable nodes in $\mathcal{V}$ of any type $i \neq k$ with $\nu_{i,j} = 1$ which can also connect to these n_k type-j check nodes. This results in the inequality (3.52). It suffices to require the inequality to be satisfied for the variable-node type k with the largest $n_k \nu_{k,j}$ value since the inequality will subsequently be satisfied for all other variable-node types as well. On the other hand, if (3.52) is not satisfied, then there are not enough variable nodes in $\mathcal{V}$ of type $i \neq k$ which can connect to the n_k type-j check nodes. Thus, there exists at least one singly-connected type-j check node and hence, a stopping set cannot be formed.

To summarize the main intuition behind (3.52), a valid stopping set can be formed if and only if each type-k variable node in $\mathcal{V}$ can be matched with a unique variable node in $\mathcal{V}$ of any other type $i \neq k$ such that $\nu_{i,j} = 1$. Fig. 3.5(a) shows an example

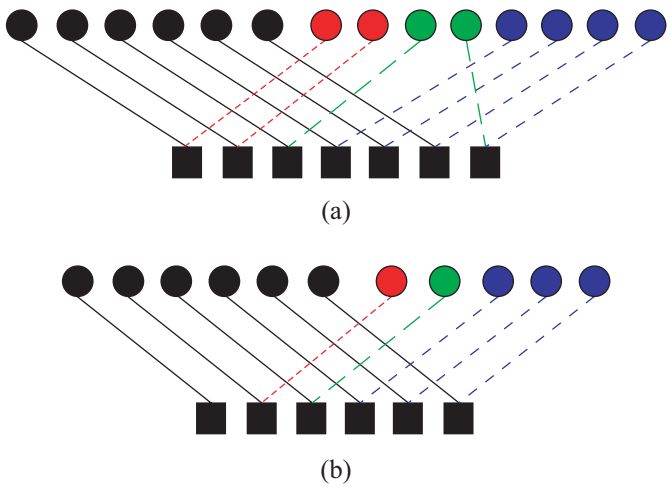


Figure 3.5: Two examples illustrating the intuition behind the role of (3.50) in Theorem 3.5. Each color denotes a different variable-node type. In example (a), (3.50) is satisfied. In example (b), (3.50) is not satisfied, leaving a singly-connected check node and hence keeping a stopping set from forming.

where (3.52) is satisfied while Fig. 3.5(b) shows an example where (3.52) is not satisfied and hence a stopping set cannot be formed.

To see the intuition behind (3.51) in Theorem 3.5, first observe that e_j is defined to be the number of edges connecting the set $\mathcal{V}$ to type-j check nodes. If (3.50) is satisfied, then these e_j edges can be spread out such that the type-j check nodes are all connected to $\mathcal{V}$ by exactly two edges, except one check node will be connected via three edges when e_j is odd. This configuration maximizes the number of type-j check nodes connected to $\mathcal{V}$, resulting in $w_j^* = \lfloor e_j/2 \rfloor$.

Using the evaluation of w_j^* provided by Theorem 3.5 and the fact that $w^* = \sum_{j=1}^{J} w_j^*$, we can revise Theorem 3.3 to more explicitly represent the enumerator exponent, resulting in an expression for the enumerator approximation which is easier to analyze.

Theorem 3.6 (Theorem 3.3 Version 2)**.** *For an LDPC ensemble based on Z copies of a protograph with M variable-node types, J check-node types, E edge types, codelength $n = ZM$, and stopping set size $v < n$, the expected number of stopping sets of size v*

is approximated by

$$s(n,v) = \sum_{\mathbf{n}\in\mathcal{S}_d} Z^{v-\sum_{j=1}^{J}\lceil e_j/2\rceil \pm \frac{O(v\log v)}{\log n}} \tag{3.53}$$

where

$$\mathcal{S}_d = \left\{ \mathbf{n}\in\mathcal{S}_p : \; 2\max_{1\le i\le M} n_i\nu_{i,j} \le \sum_{i=1}^{M} n_i\nu_{i,j},\; \forall j \right\}, \tag{3.54}$$

e_j is the number of edges connecting type-j check nodes to the stopping set $\mathcal{V}_v$ distributed according to $\mathbf{n}$, and $\mathcal{S}_p$ is defined in (3.11).

Theorem 3.6 differs from the original Theorem 3.3 in several ways. First, the summation in (3.53) of Theorem 3.6 is restricted to include only those variable-node distributions $\mathbf{n}$ which can form valid stopping sets, as determined by Theorem 3.5. Thus, all terms in the summation will be nonzero. Further, for these distributions $\mathbf{n}\in\mathcal{S}_d$, the enumerator exponent can be easily calculated with a simple algebraic expression. The maximization function for determining w^* in Theorem 3.3 is no longer needed. Thus, the enumerator approximation is now easier to analyze.

3.7.2 Evaluating Exponents with Linear/Integer Programming

To find the dominating enumerator exponent for a given code ensemble, one needs to search for the largest exponent in the polynomial described by (3.53) where the search is completed over the set $\mathcal{S}_d$ defined in (3.54). The set $\mathcal{S}_d$ depends on the protograph structure and thus, the dominant exponent is, in general, difficult to compute explicitly. However, a tight upper bound can be found using integer programming.

By removing the ceiling functions, we obtain an upper bound on the exponent:

$$v - \sum_{j=1}^{J} \left\lceil \frac{e_j}{2} \right\rceil \leq v - \left\lceil \frac{e}{2} \right\rceil = v - \left\lceil \frac{v d_{v,avg}}{2} \right\rceil \tag{3.55}$$

where $d_{v,avg}$ is the average variable-node degree in the stopping set. Using (3.55) to produce an objective function and describing $\mathcal{S}_d$ as a set of linear constraints, integer programming can produce the maximum enumerator exponent over $\mathbf{n} \in \mathcal{S}_d$. To develop the integer program, first observe that the criteria

$$2 \max_{1 \leq i \leq M} n_i \nu_{i,j} \leq \sum_{i=1}^{M} n_i \nu_{i,j} \qquad \forall j \in \{1, \ldots, J\} \tag{3.56}$$

can be written as MJ linear/integer programming constraints:

$$2 n_i \nu_{i,j} \leq \sum_{i=1}^{M} n_i \nu_{i,j} \qquad \forall i \in \{1, \ldots, M\},\ \forall j \in \{1, \ldots, J\}. \tag{3.57}$$

Thus, the integer program can be specified as follows.

The Integer Program

The objective function to be minimized is

$$e = \sum_{i=1}^{M} d_{v,i} n_i \tag{3.58}$$

subject to the following constraints:

- $2 n_i \nu_{i,j} \leq \sum_{i=1}^{M} n_i \nu_{i,j} \quad \forall i, j$
- n_i is an integer $\geq 0 \quad \forall i$
- $\sum_{i=1}^{M} n_i = v$

By making a simple modification to the optimization function and the last constraint, the integer program can analyze protographs with punctured nodes. First, let $I_i(punc) = 1$ if the ith variable-node type is punctured and $I_i(punc) = 0$ otherwise. Then, the modified integer program can be specified as follows.

The Integer Program with Punctured Nodes

The objective function to be minimized is

$$e_{eff} = \sum_{i=1}^{M} d_{v,i} - 2I_i(punc)n_i \tag{3.59}$$

subject to the following constraints:

- $2n_i\nu_{i,j} \leq \sum_{i=1}^{M} n_i\nu_{i,j} \quad \forall i, j$
- n_i is an integer $\geq 0 \quad \forall i$
- $\sum_{i=1}^{M} n_i(1 - I_i(punc)) = v$

The minimum objective value, $e_{eff,min}$, found by the integer program provides the following upper bound on the enumerator exponent:

$$v - \sum_{j=1}^{J} \left\lceil \frac{e_j}{2} \right\rceil \leq v - \left\lceil \frac{e_{eff,min}}{2} \right\rceil, \tag{3.60}$$

and this upper bound is achievable if e_j is even for all j. Further, since the solution vector $\mathbf{n}_{soln}$ found by the integer program can form a valid stopping set, evaluating the enumerator exponent in Theorem 3.6 for $\mathbf{n} = \mathbf{n}_{soln}$ provides a lower bound on the worst-case enumerator exponent.

Using the integer program, we analyze several rate-1/2 code ensembles: the regular (3,6) LDPC ensemble, the precoded (3,6) ensemble [2], and JPL's R4JA, AR4JA,

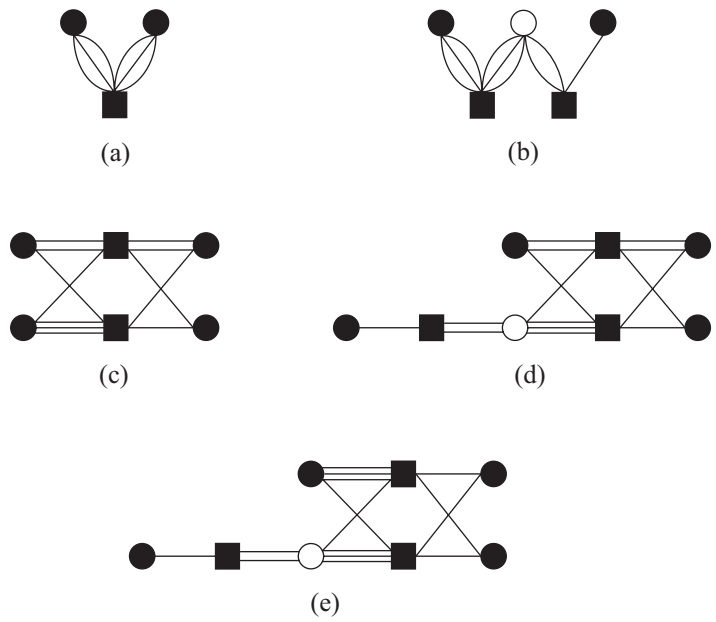


Figure 3.6: Compact representation of the protographs for the rate-1/2 LDPC ensembles analyzed in Section 3.7.2: (a) the regular (3,6) ensemble, (b) the precoded (3,6) ensemble [2], (c) the R4JA ensemble [2], (d) the AR4JA ensemble [2], and (e) the AR4A ensemble [3]. The black circles represent transmitted variable nodes while the white circles represent punctured variable nodes.

and AR4A code ensembles [2, 3]. The protographs for these code ensembles are presented in Fig. 3.6 in compact form. Since Theorem 3.5 and the subsequent analysis require protographs to have no edge multiplicities, i.e., each variable-node type can only be connected at most once to each check-node type in the protograph, the compact protographs in Fig. 3.6 are expanded to the protographs in Fig. 3.7 for analysis.

Using the integer program results, Fig. 3.8 plots bounds on the worst-case exponent versus stopping-set size v for the regular (3,6), AR4A, and AR4JA ensembles, whose protographs are given in Fig. 3.7. The solid lines correspond to the minimum objective values from the integer program and hence, provide an upper bound on the enumerator exponent. The dashed lines correspond to the exponent calculated for the integer program solution $\mathbf{n}_{soln}$ and hence, provide a lower bound on the worst-case

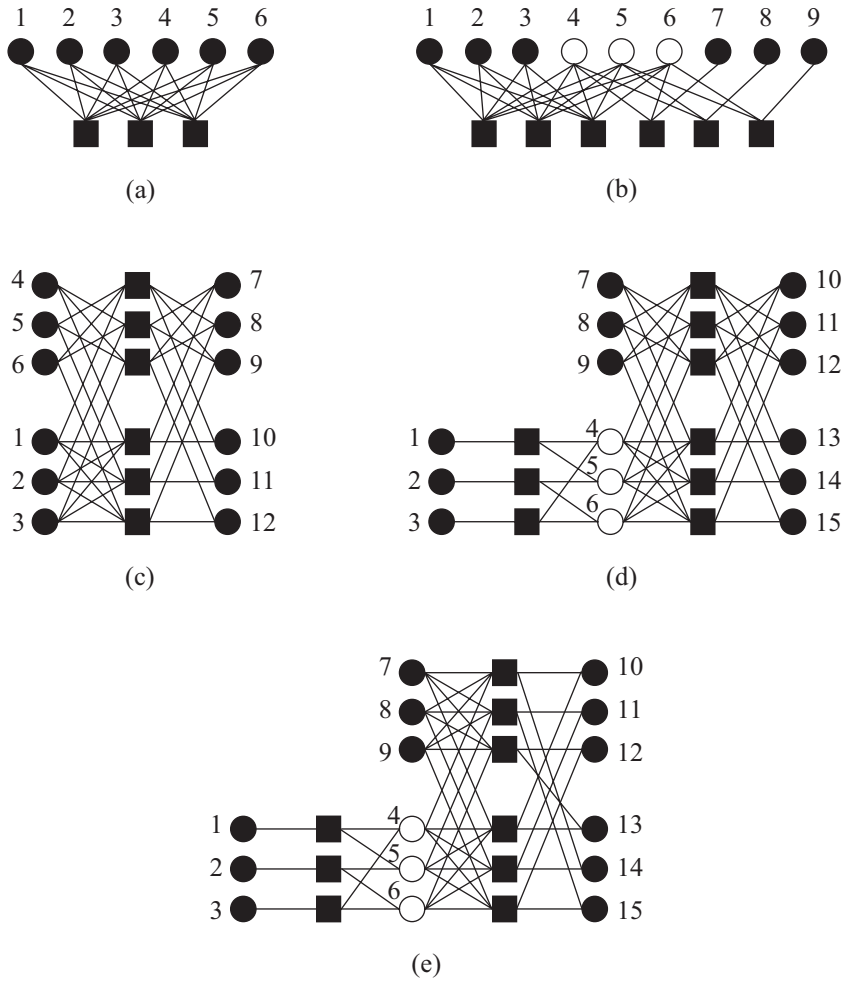


Figure 3.7: The protographs for the rate-1/2 LDPC ensembles analyzed in Section 3.7.2: (a) the regular (3,6) ensemble, (b) the precoded (3,6) ensemble [2], (c) the R4JA ensemble [2], (d) the AR4JA ensemble [2], and (e) the AR4A ensemble [3]. The black circles represent transmitted variable nodes while the white circles represent punctured variable nodes.

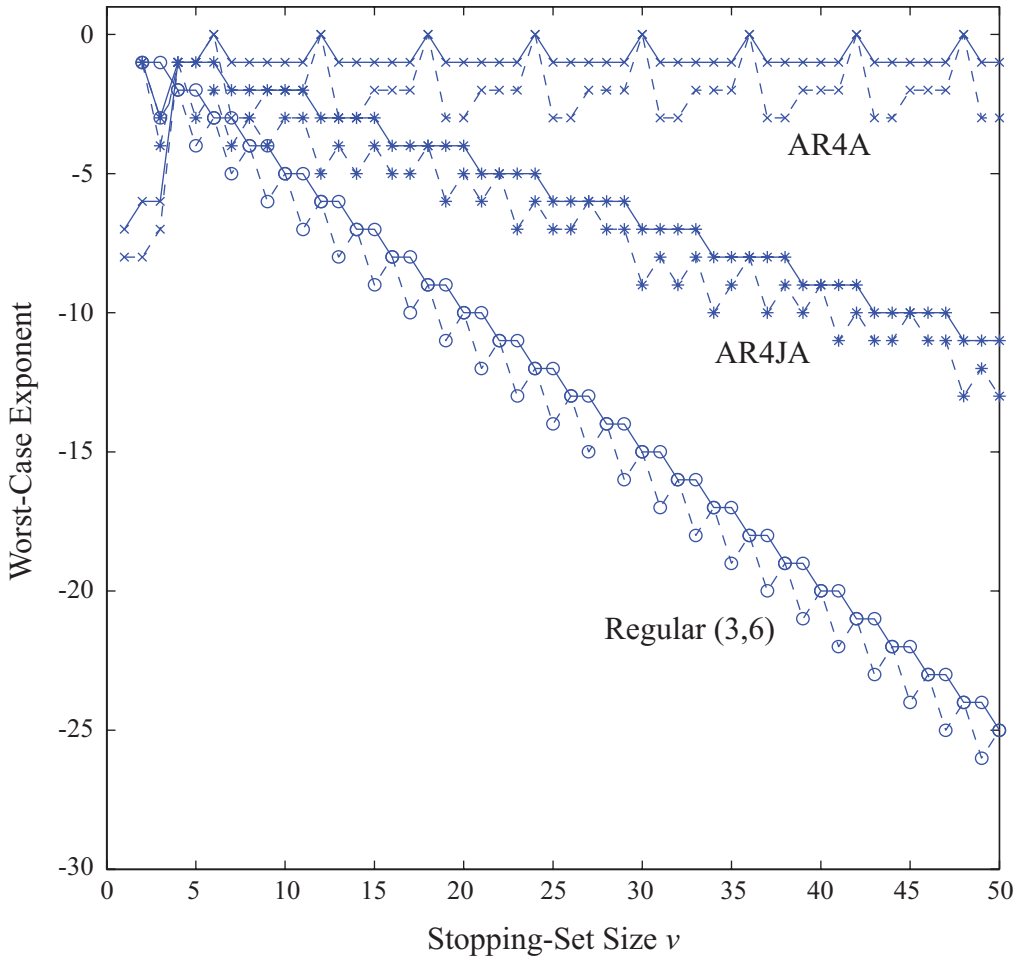


Figure 3.8: Upper (solid) and lower (dashed) bounds on the enumerator exponent determined through the integer program for the regular (3,6), AR4A, and AR4JA ensembles.

enumerator exponent.

From Fig. 3.8, observe that the upper and lower bounds only differ by a small number. Thus, the upper bound provided by the integer program is fairly tight. It may be possible to find tighter lower bounds by finding different $\mathbf{n}$ vectors which also produce the optimal objective value from the integer program.

The AR4A code ensemble has poor enumerator exponents, as shown in Fig. 3.8, indicating poor error-floor performance, which is consistent with the fact that ARA-type code ensembles have *sublinear* stopping number and minimum distance [18, 50].

Fig. 3.8 shows that the worst-case enumerator exponents generally follow a linear trend with stopping-set size v. Slight deviations from this linear trend occur due to the discretization of the exponent arising from the presence of ceiling functions in the exponent expression. The slope of this enumerator to stopping-set size relationship can be evaluated by modifying the integer program to a linear program, by normalizing the expressions in the integer program by v.

First, let $\delta_i = n_i/v$ where δ_i does not need to be an integer. Then, divide the objective function and linear constraints by v. The resulting linear program is specified as follows.

The Linear Program

The objective function to be minimized is

$$d_{v,avg} = \sum_{i=1}^{M} (d_{v,i} - 2I_i(punc))\delta_i \tag{3.61}$$

subject to the following constraints:

- $2\delta_i \nu_{i,j} \leq \sum_{i=1}^{M} \delta_i \nu_{i,j} \quad \forall i, j$
- $0 \leq \delta_i \leq 1 \quad \forall i$
- $\sum_{i=1}^{M} \delta_i (1 - I_i(punc)) = 1.$

Based on the optimal objective value $\min\{d_{v,avg}\}$, i.e., the smallest possible average variable-node degree in a stopping set, the worst-case enumerator exponent can be approximated by

$$E(v) \approx v(1 - \min_{\mathbf{n} \in \mathcal{S}_d} d_{v,avg}/2) = \alpha v \tag{3.62}$$

where

$$\alpha = 1 - \min_{\mathbf{n} \in \mathcal{S}_d} d_{v,avg}/2. \tag{3.63}$$

Table 3.2: Enumerator Exponent Factors Calculated Using Linear Programming

Code Ensemble	$\min_{\mathbf{n}\in\mathcal{S}_d} d_{v,avg}$	α
Regular (3,6)	3	-1/2
Precoded (3,6)	2.5	-1/4
R4JA	2.5	-1/4
AR4JA	$2.\bar{4}$	-2/9
AR4A	2	0

Table 3.3: Optimizing Variable-Node Distributions from the Linear Program

Code Ensemble	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7	δ_8	δ_9	δ_{10}	δ_{11}	δ_{12}	δ_{13}	δ_{14}	δ_{15}
Regular (3,6)	$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	0									
Precoded (3,6)	0	0	0	$\frac{1}{4}$	$\frac{1}{4}$	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$						
R4JA	0	0	0	0	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	$\frac{1}{4}$	$\frac{1}{4}$			
AR4JA	$\frac{1}{9}$	$\frac{1}{9}$	0	0	$\frac{1}{9}$	0	$\frac{1}{18}$	$\frac{1}{9}$	$\frac{1}{18}$	0	0	0	$\frac{1}{6}$	$\frac{2}{9}$	$\frac{1}{6}$
AR4A	0	0	0	0	0	0	0	0	0	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$

For a given stopping-set size v, the output of the linear program agrees exactly with the output of the integer program when the solution of the linear program has δ_i values such that $\delta_i v$ is an integer for all i. This analysis shows that the dominating enumerator exponent can be roughly characterized by a *single* parameter α. Thus, α is a *single* metric that captures the sublinear stopping-set behavior in the error-floor region and can be used to easily compare different protograph-based LDPC ensembles.

Table 3.2 provides α values for the LDPC ensembles whose protographs are depicted in Fig. 3.7. The corresponding optimizing variable-node distributions $\boldsymbol{\delta}$ from the linear program output are given in Table 3.3, where the numbering of the variable-node types follows the numbering in Fig. 3.7.

The values provided in Table 3.2 agree with the enumerator behavior shown in

Fig. 3.8. For a few of the smallest values of stopping-set size v, the linear program cannot accurately predict the exponent value since its solution requires non-integer $\delta_i v$ values that are close to zero. Since v is small, the v variable nodes in the stopping set cannot be distributed to the variable-node types in such a way to closely approximate the $\delta_i v$ values from the linear program solution. This reflects the protograph structure's ability to eliminate or reduce the number of very small stopping sets. For larger v values, the linear program is able to closely predict the exponent value since the $\delta_i v$ values in its solution can be closely approximated with nonzero integer values, i.e., the v variable nodes in the stopping set can be distributed to the variable-node types closely following the distribution provided by the $\delta_i v$ values.

Since α represents the slope of the enumerator exponent, the code ensembles with smaller, i.e., more negative, values of α have better sublinear stopping-set behavior in the error-floor region. The regular (3,6) ensemble has the best α value while the AR4A code has $\alpha = 0$, indicating that the enumerator exponent is not decreasing with stopping-set size.

Table 3.4 compares the α values from Table 3.4 to the infinite-length threshold values, which characterize the waterfall/threshold region of performance, obtained through density evolution. The threshold values in Table 3.4 are given for the AWGN channel since these values were already available from [2,3]. From Table 3.4, observe that as α increases, the threshold decreases. Thus, the code ensembles analyzed here follow the conjecture that there is a tradeoff between threshold and error-floor performance. However, since Table 3.4 only provides a few examples, no conclusive statement can be made regarding this conjecture.

Although α provides a useful metric for characterizing sublinear stopping-set behavior, α does not capture all of the factors affecting error-floor performance. For example, the stopping number cannot be determined by α. Recall from Section 3.1

Table 3.4: Enumerator Exponent Factors vs. Threshold

Code Ensemble	α	Threshold (dB)
Regular (3,6)	-1/2	1.1
R4JA	-1/4	1.0
Precoded (3,6)	-1/4	0.87
AR4JA	-2/9	0.63
AR4A	0	0.56

that the stopping number is the smallest nonzero stopping-set size in the code ensemble and is an important factor affecting error-floor performance. Another issue that the parameter α cannot address is whether or not one can easily expurgate an ensemble such that all stopping sets with size up to v are eliminated, for some integer v. Some code ensembles may be easier to expurgate than others. Basically, for good error-floor performance, a code needs to have a large stopping number and a small α, and α does not provide information on the stopping number of a code ensemble, an expurgated code ensemble, or a particular code in the ensemble. Thus, while α is a useful metric that can summarize sublinear stopping-set behavior in a single parameter, other factors such as stopping number are also important in determining the overall error-floor performance.

3.8 Precoding

Precoding [2] is a technique used in code design to improve threshold performance. This technique utilizes punctured nodes, i.e., variable nodes in the LDPC graph that are not transmitted. Punctured nodes typically arise from two main sources: puncturing existing nodes in a graph or adding state nodes to a graph. Puncturing can have many advantages including increased flexibility in how a code is represented on a graph which can result in improved iterative-decoding performance; the ability

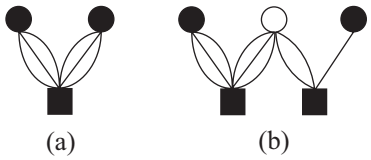


Figure 3.9: (a) Protograph P for a regular (3,6) LDPC ensemble. (b) Protograph P' which is derived from protograph P by precoding one of the variable nodes.

to build a family of rate-compatible codes; and the ability to build capacity-achieving sequences with *bounded* complexity per information bit, e.g., [56]. Precoding, which involves the addition of low-degree variable nodes while puncturing existing (typically high-degree) variable nodes, uses puncturing to allow low-degree variable nodes in the graph to improve the threshold performance while maintaining the same code rate and limiting the degradation to error-floor performance, as will be shown in this section.

We define precoding as follows:

Definition 3.1. Let P be a protograph with M variable nodes and J check nodes. To **precode** [2] a variable node v_i for some $i \in \{1, \ldots, M\}$, first add a check node c_{J+1} which is connected to v_i via two edges. Then, add a variable node v_{M+1} which is connected to c_{J+1} via one edge. Finally, puncture v_i. The resulting protograph P' has $M+1$ variable nodes (one of which is punctured), $J+1$ check nodes, and $E+3$ edges.

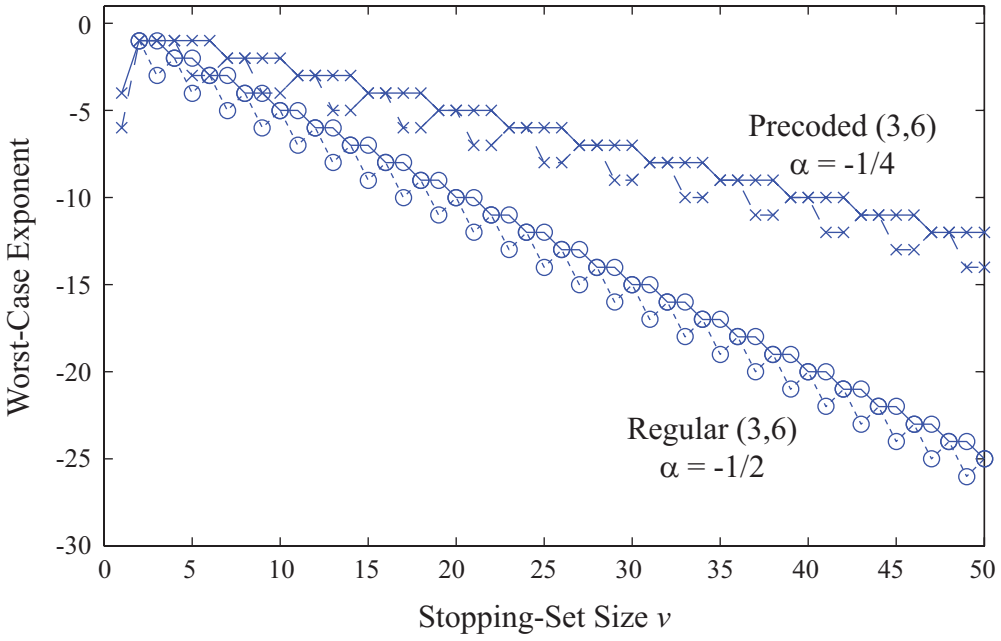
For example, Fig. 3.9 shows how a regular (3,6) protograph can be precoded.

The technique of precoding has been shown experimentally and numerically to improve error-floor performance, in terms of weight and stopping-set enumerator behavior for codewords and stopping sets which grow linearly with codelength [2]. Divsalar provides specific examples where precoding increases the δ_{min} metric for weight enumerators and the $\delta_{s,min}$ metric for stopping-set enumerators (see Section 3.1 for a discussion on the $\delta_{s,min}$ metric). For example, precoding a regular (3,6) LDPC

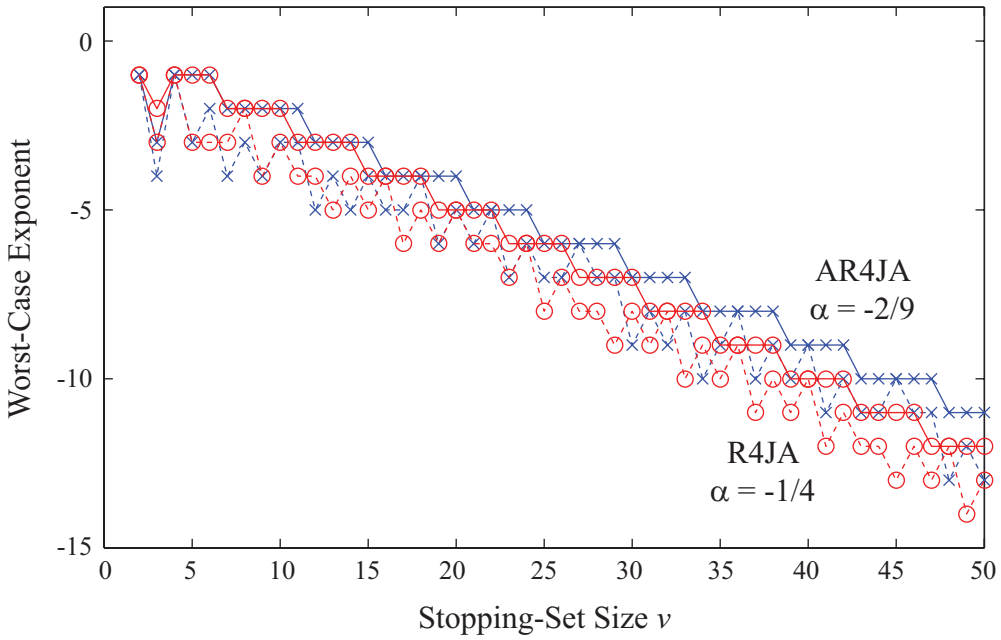
ensemble increases δ_{min} from 0.023 to 0.033 [2] and precoding the R4JA ensemble to the AR4JA ensemble increases $\delta_{s,min}$ from 0.011 to 0.012 [50]. Further, Divsalar has partially proven the benefits of precoding by comparing weight-enumerator exponents, for codewords that grow linearly with codelength [2], and similar results apply for the stopping-set-enumerator exponents, for stopping sets that grow linearly with codelength, as well. However, the partial proof is unable to show when and if precoding improves error-floor performance of the ensemble as a whole.

The sublinear stopping-set enumerator analysis presented earlier in this chapter is used here to gain additional insight into precoding. First, using the analysis in Section 3.7, we compare the stopping-set enumerator exponents for precoded ensembles to the exponents for the corresponding original, un-precoded ensembles. Fig. 3.10 plots the enumerator exponents versus stopping-set size v for (a) the regular (3,6) ensemble and its precoded version, the precoded (3,6) ensemble and (b) the R4JA ensemble and its precoded version, the AR4JA ensemble. As in Fig. 3.8, the solid lines are the upper bounds on the enumerator exponent obtained from the minimum objective values from the integer program. The dashed lines are the lower bounds on the worst-case enumerator exponent obtained from the integer program solution $\mathbf{n}_{soln}$.

As predicted by Table 3.2, which shows that the precoded ensembles have larger α values than the corresponding original ensembles, Fig. 3.10 shows that the precoded ensembles have larger exponents than the corresponding original ensembles for the same effective stopping-set size v, although the difference is fairly small for the R4JA and AR4JA ensembles. Thus, given a certain number of erasures v, the precoded ensemble has more stopping sets on which iterative decoding will fail. Hence, one expects that the precoded ensemble has worse error-floor performance, in terms of sublinear stopping-set behavior. However, one must also consider the stopping num-



(a)



(b)

Figure 3.10: Upper (solid) and lower (dashed) bounds on the enumerator exponent for (a) the regular (3,6) ensemble and its precoded version, the precoded (3,6) ensemble and (b) the R4JA ensemble (red with o markers) and its precoded version, the AR4JA ensemble (blue with x markers).

ber, i.e., the smallest nonzero stopping-set size in the code ensemble. Particularly after expurgation, it may be possible that one ensemble has a significantly larger stopping number than the other ensemble, and this may have a larger impact on the error-floor performance than the differences in α values between the two ensembles. We will prove in Theorem 3.7 that the dominant exponent of Z for the precoded ensemble is no worse than one plus the dominant exponent for the original ensemble, even after expurgation of small stopping sets in both ensembles.

Before introducing Theorem 3.7, first recall that for a BEC with erasure probability ϵ, the average block-error probability of an LDPC ensemble can be upper bounded by

$$E_G[P_B^{IT}(n,\epsilon)] \leq P_e^{ub} = \sum_{v=1}^{n} \epsilon^v s(n,v) \tag{3.64}$$

where the bound is tight in the error-floor region (see Section 1.1.6). From Theorem 3.1, a single term in the summation in the upper bound given by (3.64) can be expressed as

$$P_e^{ub}(v) = \epsilon^v \sum_{\mathbf{n}\in\mathcal{S}_p} \left[\prod_{i=1}^{M} \binom{Z}{n_i}^{1-d_{v,i}} \times \prod_{j=1}^{J} \text{coef} \left\{ \left[\prod_{k=1}^{d_{c,j}} (1+x_k) - \sum_{k=1}^{d_{c,j}} x_k \right]^Z , \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\} \right]. \tag{3.65}$$

For category-P stopping sets, this expression can be approximated by applying Theorems 3.3 and 3.4 to obtain

$$P_e^{ub}(v) \approx P_e^{ub'}(v) = \epsilon^v \sum_{\mathbf{n}\in\mathcal{S}_p} \frac{Z^{v-e+w^*}}{\prod_{i=1}^{M} n_i!^{1-d_{v,i}} \prod_{j=1}^{J} \prod_{h=1}^{|\beta|-1} m_h!} \tag{3.66}$$

where $\mathbf{m} \in \mathcal{S}_m$ is chosen to maximize w_j^* for each check-node type j. It can easily be shown that for a code ensemble with punctured nodes, a single term in the summation

over v in (3.64) is

$$P_{e,punc}^{ub}(v) = \sum_{\mathbf{n}\in\mathcal{S}_p} \epsilon^{v_{eff}} \left[\prod_{i=1}^{M} \binom{Z}{n_i}^{1-d_{v,i}} \times \prod_{j=1}^{J} \text{coef}\left\{ \left[\prod_{k=1}^{d_{c,j}} (1+x_k) - \sum_{k=1}^{d_{c,j}} x_k \right]^{Z}, \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\} \right] \quad (3.67)$$

where v is the stopping-set size including punctured nodes and v_{eff} is the number of variable nodes in the stopping set excluding punctured nodes. When nodes are punctured, they are automatically erased and hence, the channel only needs to erase v_{eff} variable nodes. If stopping sets of size v follow category-P behavior, then $P_{e,punc}^{ub}(v)$ can be approximated by applying Theorems 3.3 and 3.4 to obtain

$$P_{e,punc}^{ub}(v) \approx P_{e,punc}^{ub'}(v) = \sum_{\mathbf{n}\in\mathcal{S}_p} \frac{\epsilon^{v_{eff}} Z^{v-e+w^*}}{\prod_{i=1}^{M} n_i!^{1-d_{v,i}} \prod_{j=1}^{J} \prod_{h=1}^{|\beta|-1} m_h!} \quad (3.68)$$

where $\mathbf{m} \in \mathcal{S}_m$ is chosen to maximize w_j^* for each check-node type j.

Each stopping set $\mathcal{V}'$ in the precoded ensemble consists of a stopping set $\mathcal{V}$ from the original ensemble plus some number of type-$(M+1)$ variable nodes. For the precoded ensemble, the number of erasures the channel must introduce in order to generate a decoding failure on $\mathcal{V}'$ is increased by the presence of n_{pre} type-$(M+1)$ variable nodes but is decreased by the presence of n_{punc} type-i punctured nodes. Depending on the balance of these two contributions, the resulting error-floor performance may be better or worse than that of the original ensemble. The following theorem provides a bound on the worst-case error-floor performance with precoding.

Theorem 3.7. *Consider a protograph P whose type-i variable node is precoded to generate the protograph P'. Let V_P and $V_{P'}$ be the sets of stopping-set sizes which dominate the performance of the ensemble based on protograph P and P', respectively,*

and let $V_U = \{V_P \cup V_{P'}\}$. Assume that for all $v \in V_U$, stopping sets of size v follow category-P behavior for both protograph-based ensembles. Let Z be large enough such that Theorem 3.4 holds and $Z \geq 1/\epsilon$. Then,

$$P_{e,P'}^{ub} \approx \sum_{v \in V_U} P_{e,P'}^{ub'}(v) \leq (Z+1) \sum_{v \in V_U} P_{e,P}^{ub'}(v) \approx Z \cdot P_{e,P}^{ub}. \tag{3.69}$$

Thus, the largest exponent of Z in the error-probability approximation for protograph-P' ensembles is at most one greater than the largest exponent for protograph-P ensembles, in the error-floor region.

Proof. A sketch of the proof is provided here. (See Appendix G for a full proof.) Consider a finite stopping set $\mathcal{V}'$ from the ensemble generated from protograph P', with v' variable nodes connected to $w^{*'}$ check nodes via e' edges. The set $\mathcal{V} = \{x \in \mathcal{V}' : x \text{ is a variable node of type } i \neq (M+1)\}$ is a stopping set in the ensemble generated from protograph P. The stopping-set characteristics of $\mathcal{V}'$ and $\mathcal{V}$ are related by the following expression where primed and unprimed variables correspond to characteristics of $\mathcal{V}'$ and $\mathcal{V}$, respectively:

$$(v' - e' + w^{*'}) = (v - e + w^{*}) - (n_{punc} - n_{M+1}/2) \tag{3.70}$$

where n_{punc} is the number of punctured, type-i variable nodes. By upper bounding $\sum_{v \in V_U} P_{e,P'}^{ub'}(v)$ and observing that $n_{M+1} \leq 2n_{punc}$ in order for a stopping set to be formed, the result in (3.69) can be obtained. □

The requirement that the dominating stopping sets must follow category-P behavior may not be restrictive. This is still under investigation, but the example in Section 3.6 showed that at least for the regular (3,6) LDPC ensembles analyzed, this requirement is usually satisfied.

Theorem 3.7 also applies to expurgated ensembles. Specifically, if the protograph P ensemble is expurgated to remove small stopping sets, then the precoded protograph P' ensemble can be similarly expurgated on the portion of its graph that coincides with the original, protograph P ensemble. Then, Theorem 3.7 can be applied to compare the two expurgated ensembles.

Since Theorem 3.7 only provides a worst-case bound, the precoded ensemble may, in fact, have improved error-floor performance over the original ensemble.

Section 3.3 showed that ensembles with all variable-node degrees greater than or equal to three have good error-floor performance since $(v - e + w^*) < 0$ for all stopping sets. However, current literature, e.g., [11, 18], indicates that variable nodes of degree strictly less than three are useful for achieving good threshold performance. The technique of precoding allows one to include degree-1 variable nodes to improve threshold performance while either improving or, in the worst case, limiting the degradation of the error-floor performance, in terms of sublinear stopping-set behavior.

CHAPTER 4

Conclusion

In this chapter, our main research results are summarized, and possible areas of future work are discussed.

4.1 Research Summary

In this dissertation, we investigated two main topics: time-selective complex-fading channels and finite-length analysis for the BEC. Using LDPC codes along with pilot symbols and iterative joint estimation and detection, we were able to provide practical solutions for channel coding for the time-selective complex-fading channel. Further, we were able to unify the analysis and code design for certain iterative decoding algorithms. Next, finite-length analysis of LDPC codes for the BEC was completed through investigation of sublinearly-sized stopping sets. This analysis provided insights into factors influencing error-floor performance including sublinear stopping-set behavior, advantages of protograph structure, and effects of precoding. Results for both of these main topics will be summarized in the following subsections.

4.1.1 Time-Selective Complex-Fading Channels

In Chapter 2, we analyzed and designed pilot-symbol-assisted (PSA) LDPC codes for time-selective, frequency-non-selective complex-fading channels where the fading affects both the amplitude and phase of the transmitted symbols, and the channel dynamics are taken explicitly into account. Several PSA receivers—the perfect channel-state information (CSI) receiver, the pilot-only (PO) receiver, the pilot and data correct decision feedback (PDCDF) receiver, and the quantized decision feedback (QDF) receiver—were investigated and analyzed using density evolution. The PO and QDF receivers are practical, implementable receivers while the perfect-CSI and PDCDF receivers provide bounds on the performance of practical receivers.

The performance of the perfect-CSI, PO, and PDCDF receivers were shown to be dependent only on a single parameter, b. This result led to several important consequences simplifying and unifying the analysis and code design for these receivers and for all system and channel parameters. First, only one set of analyses based on b is necessary to characterize the performance of all three receiver types for any channel coherence time and for any energy allocation. Second, optimizing the energy distribution between pilot and code symbols only requires a simple closed-form calculation from b. Third, an optimal code for one of these receivers will also be optimal for the other receivers and for any system parameters.

Finally, density evolution and simulation results were presented for regular and irregular LDPC codes in a PSA scheme. The allocation of power to the pilot and the code symbols was optimized, resulting in 1 to 2 dB improvement in performance. By utilizing the iterative joint decoding/estimation provided by the QDF algorithm, the performance improves over the best PO receiver by 0.9 dB with BPSK modulation and 0.2 dB with QPSK modulation. Thus, optimized PO schemes with QPSK modulation present an excellent performance/complexity tradeoff. We note however,

that a 2.4 dB gap still exists between the performance of the best proposed scheme (the optimized QDF receiver) and capacity.

4.1.2 Finite-Length Analysis for the BEC

In Chapter 3, we analyzed finite-length LDPC codes over the BEC to better understand the factors affecting the error-floor performance, which limits the error rates achievable by practical, finite-length codes. For both standard and protograph-based LDPC code ensembles, we analyzed the error-floor performance by studying stopping-set enumerators for stopping sets that grow at most sublinearly with code-length. Approximations to the enumerators were developed, yielding several interesting results.

First, for stopping sets that grow at most logarithmically with codelength n, the stopping-set enumerator follows a polynomial relationship with n. This category-P behavior differs from the category-E behavior, exhibited by stopping sets growing linearly with codelength, where the enumerator follows an exponential relationship with n. Thus, category-P is able to capture finite-length behavior not observed in category-E analysis. By developing a bound on the region where our enumerator approximations are valid, we began to address to question, "Given a finite stopping-set size v and a finite codelength n, which category of behavior does the stopping set follow?" For practical, finite-length codes, this question is important to determine what type of behavior the dominating stopping sets follow and hence, to determine how to design codes with lower error floors.

Using linear and integer programming, we developed a simple method for closely approximating category-P enumerator exponents. Since the exponent can be approximated by αv for some constant α, which can be found through the linear program, α provides a single metric for comparing code performance in terms of sublinear

stopping-set behavior.

Further, we have shown that protograph-based LDPC codes always perform at least as well as standard ensembles, in terms of category-P behavior, and can in fact perform strictly better. Thus, the protograph structure provides benefits in terms of category-P error-floor performance.

An illustrative example with regular (3,6) LDPC ensembles demonstrated the concepts introduced. The polynomial behavior with codelength for category-P stopping sets was verified by exact enumerator calculations, at least for small stopping-set sizes. Further, the ability of the protograph structure to improve enumerator exponents over the standard-ensemble exponents was demonstrated.

In addition, the technique of precoding was analyzed in terms of category-P behavior. The technique of precoding can improve threshold performance by introducing low-degree variable nodes but can either improve or degrade error-floor performance. By bounding the worst-case degradation of error-floor performance, we showed that the worst-case exponent for precoded ensembles is no worse than one plus the exponent for the original ensemble (before precoding).

4.2 Future Work

Several interesting problems still remain for the time-selective complex-fading channel. Although the PSA LDPC codes and iterative joint estimation and decoding algorithms developed in Chapter 2 performed well on the complex-fading channel, there is still a 2.4 dB gap to capacity. Thus, there is still room for improvement. Closing the gap to capacity may require better receiver algorithms, other methods for channel estimation besides the pilot-symbol-assisted scheme used in Chapter 2, other modulation schemes, and/or other strategies.

Further, all analyses completed in Chapter 2 were based on infinite-length analy-

sis. The performance of practical, finite-length codes over the complex-fading channel needs to be studied to gain insight into finite-length effects and how these effects impact real-world performance. Different code design criteria may arise in this case.

In the area of finite-length analysis, there are still many remaining questions of interest, and we present some of these issues below.

First, the question "Given a finite stopping-set size v and a finite codelength n, do the stopping sets follow category-P or category-E behavior?" is still not fully answered. Theorem 3.5 is only able to provide a partial answer by providing a bound on when the approximations in Theorem 3.1 are valid. If this bound can be further tightened, then one can prove that category-P analysis extends to more stopping sets. Additional future research work to help address the question of interest includes (1) determining when category-E analysis is valid, i.e., finding an N_0 such that category-E analysis applies for all $n > N_0$, (2) determining when category-E analysis is *not* valid, i.e., finding an N_1 such that category-E analysis does *not* apply for all $n < N_1$, and (3) determining when category-P analysis is *not* valid. If tight bounds can be found, then we can much more accurately determine under which category of behavior the dominating stopping sets fall and hence, better design codes for improved error-floor performance.

The work presented in Chapter 3 provides some insight into how protograph structure influences error-floor performance. However, there are still many questions that could be answered by further research. For example, another aspect to be examined is the different behavior of even and odd stopping sets in some protograph-based codes and how this influences the overall error-floor performance. Other protograph-related issues are presented below.

Precoding is a useful technique for improving threshold performance, but its effect on error-floor performance is still uncertain. Divsalar [2] showed a partial

proof of the advantage of precoding for category-E stopping sets. However, for the examples in Section 3.8, precoding degraded category-P stopping-set performance. Theorem 3.7 in Section 3.8 showed that the worst-case scenario only increments the category-P enumerator exponent by one, but the result is only a bound and cannot show whether precoding helps or hurts the error-floor performance. Future research of interest would include determining if and when precoding degrades error-floor performance. Is it possible that precoding can simultaneously improve threshold and error-floor performance?

Another ad-hoc method for designing codes is a double-expansion method for creating an LDPC code from a protograph structure. In the first expansion, the protograph is copied only a few times (e.g., four) and then the edges are permuted to form a larger, intermediate protograph. In the second expansion, this intermediate protograph is then replicated multiple times (as many times as necessary to obtain the desired codelength) and the edges are permuted to create the final LDPC code. Typically, the first expansion is a small expansion while the second one is much larger. This method has empirically produced better codes. The intuition behind this method is that the two expansions help to spread the edge connections throughout the graph, i.e., the nodes are more interconnected and thus stopping sets will hopefully be larger. It would be interesting to see if this double-expansion method can be proven to be superior to a single expansion. Further, is there an optimal expansion factor for the first expansion? Are there certain conditions necessary for this method to result in better codes? Is further expurgation necessary, i.e., are more restrictions needed to ensure the generation of a good code?

Another area of future research is extending the work here to more realistic channels, e.g., the AWGN channel. Analysis into trapping sets, a generalization of stopping sets, has been completed in other works, e.g., [35, 57]. However, the

analysis is cumbersome. Is it possible to extend the enumerator approximations here to trapping sets and to gain insight into error-floor performance for AWGN channels?

Although LDPC codes along with iterative-decoding algorithms have assisted communication systems in lowering error rates and approaching capacity on a variety of channels, there is still much interesting research to be done.

APPENDICES

APPENDIX A

Proof of Monotonicity for Fading Channels

The proof of monotonicity of receiver performance for the perfect-CSI, PO, and PDCDF receivers with the unifying parameter b, discussed in Section 2.3.1, is given in this appendix. We will show that if $b_1 < b_2$, then the performance of the receiver characterized by $b = b_2$, through the initial density given in (2.15), is degraded from the performance of the receiver characterized by $b = b_1$.

Proof. For each value of b (regardless of receiver type), there exists an equivalent fading channel with a perfect-CSI receiver. This equivalent channel is described by the following expression:

$$z = c(-1)^a + n \tag{A.1}$$

where z is the channel output, $(-1)^a$ is the channel input where $a \in \{0, 1\}$, c is a zero-mean, unit-energy complex normal fading coefficient (known at the receiver), and n is a zero-mean complex normal additive noise with energy N_0 where N_0 satisfies the equation $b = (\sqrt{1 + N_0} - 1)/2$. It can be shown that the log-likelihood ratio for a given by

$$x = \log \frac{f(z|a = 0, c)}{f(z|a = 1, c)} = \frac{4}{N_0} \Re\{zc^*\} \tag{A.2}$$

is a sufficient statistic for detection in this channel, and that the conditional distri-

bution of x is given by

$$f(x|a=0) = f(-x|a=1) = h(x) \tag{A.3}$$

where $h(x)$ has the form of (2.15) with $b = (\sqrt{1+N_0} - 1)/2$.

For $b = b_1$, choose N_0 such that $b_1 = (\sqrt{1+N_0} - 1)/2$. Since $b_2 > b_1$, we can find $N_0' = N_0 + Q > N_0$ such that $b_2 = (\sqrt{1+Q+N_0} - 1)/2$. Based on the above equivalence, b_2 corresponds to a fading channel with a perfect-CSI receiver where now more additive noise is injected, i.e.,

$$z' = c(-1)^a + n' = c(-1)^a + n + q = z + q \tag{A.4}$$

where the noise component n' has energy $N_0' = Q + N_0 > N_0$ and the log-likelihood sufficient statistic is given by $y = 4/N_0' \Re\{z'c^*\}$. The above expression (A.4) shows that this latter channel is a degraded version of the former channel, i.e., the output z is further degraded by an additional noise q with variance Q. Thus, according to [10, Theorem 1], the performance of any receiver characterized by b_2 is worse than the performance of any receiver characterized by b_1. □

APPENDIX B

Derivation of the Stopping-Set Enumerator for Protograph-Based Ensembles

This appendix provides a proof of Theorem 3.1, which is restated here:

Theorem 3.1. *For a protograph-based LDPC ensemble generated from Z copies of a protograph with M variable-node types and J check-node types connected via E edge types, the expected number of stopping sets of size v is*

$$s(n,v) = \sum_{\mathbf{n}\in\mathcal{S}_p} \left[\prod_{i=1}^{M} \binom{Z}{n_i}^{1-d_{v,i}} \times \prod_{j=1}^{J} \operatorname{coef}\left\{ \left[\prod_{k=1}^{d_{c,j}} (1+x_k) - \sum_{k=1}^{d_{c,j}} x_k \right]^{Z}, \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\} \right] \tag{B.1}$$

where the codelength $n = ZM$ and

$$\mathcal{S}_p = \left\{ \mathbf{n} \in \mathbb{Z}^M : \, 0 \le n_i \le Z, \, \forall i \in \{1, \dots M\}; \, \sum_{i=1}^{M} n_i = v \right\}. \tag{B.2}$$

Proof. The expression for $s(n,v)$ will first be broken down based on $\mathbf{n}$, the distribution of variable-node types in the stopping set. Let $\mathcal{V}$ be the set of all n variable

nodes in the graph and fix a set $\mathcal{V}_v \subseteq \mathcal{V}$ such that $\mathcal{V}_v$ has exactly v variable nodes. For more compact notation, s.s. will denote stopping set(s). Applying the definition of $s(n, v)$,

$$\begin{aligned} s(n,v) &:= E\left[\#\text{ of s.s. of size } v \text{ in } \mathcal{G}\right] \\ &= E\left[\sum_{\mathbf{n}\in\mathcal{S}_p} (\#\text{ of s.s. with distribution } \mathbf{n} \text{ in } \mathcal{G})\right] \\ &= E\left[\sum_{\mathbf{n}\in\mathcal{S}_p} \sum_{\substack{\mathcal{V}_v\subseteq\mathcal{V}: \\ \mathcal{V}_v \text{ has distribution } \mathbf{n}}} I(\mathcal{V}_v \text{ is a s.s. in } \mathcal{G})\right] \end{aligned} \tag{B.3}$$

where the expectation is taken over all graphs $\mathcal{G}$ in the protograph-based LDPC ensemble and $I(\cdot)$ is the indicator function. Taking the expectation inside the summations results in

$$\begin{aligned} s(n,v) &= \sum_{\mathbf{n}\in\mathcal{S}_p} \sum_{\substack{\mathcal{V}_v\subseteq\mathcal{V}: \\ \mathcal{V}_v \text{ has distribution } \mathbf{n}}} E\left[I(\mathcal{V}_v \text{ is a s.s. in } \mathcal{G})\right] \\ &= \sum_{\mathbf{n}\in\mathcal{S}_p} \sum_{\substack{\mathcal{V}_v\subseteq\mathcal{V}: \\ \mathcal{V}_v \text{ has distribution } \mathbf{n}}} P\left(\mathcal{V}_v \text{ is a s.s.}\right). \end{aligned} \tag{B.4}$$

Since the probability in this expression is only dependent on the distribution $\mathbf{n}$ of variable-node types in $\mathcal{V}_v$ and not on the specific $\mathcal{V}_v$, the expression simplifies to

$$s(n,v) = \sum_{\mathbf{n}\in\mathcal{S}_p} N_{\mathbf{n}} P_{\mathbf{n}} \tag{B.5}$$

where $N_{\mathbf{n}}$ denotes the number of sets $\mathcal{V}_v$ that are distributed according to $\mathbf{n}$ and $P_{\mathbf{n}}$ denotes the probability that a set $\mathcal{V}_v$ following the distribution in $\mathbf{n}$ forms a stopping set.

To compute $N_{\mathbf{n}}$, recall that each variable-node type in the protograph is replicated

to create Z copies. Thus, the total number of ways to choose v variable nodes out of the code of length n such that there are n_i nodes of type i for $i = 1, \ldots, M$ is

$$N_{\mathbf{n}} = \prod_{i=1}^{M} \binom{Z}{n_i} \qquad \forall \mathbf{n} \in \mathcal{S}_p. \tag{B.6}$$

To compute $P_{\mathbf{n}}$, we calculate the fraction of subgraphs induced by a fixed subset $\mathcal{V}_v$ distributed according to $\mathbf{n}$ such that $\mathcal{V}_v$ forms a stopping set. Specifically,

$$P_{\mathbf{n}} = \frac{C_{ss}}{C_{tot}} \tag{B.7}$$

where C_{tot} is the total number of subgraphs that can be induced by $\mathcal{V}_v$ and C_{ss} is the number of subgraphs induced by $\mathcal{V}_v$ such that the following property holds: each of the ZJ check nodes is connected to $\mathcal{V}_v$ at least twice or not at all.

First, the total number of possible subgraphs C_{tot} is calculated. For each variable-node type i, consider each of the $d_{v,i}$ edge types emanating from it and the corresponding check-node type j to which the edge type is connected. For each of these edge types, the n_i type-i variable nodes in $\mathcal{V}_v$ will be connected to n_i check nodes out of the Z possible choices of type-j check nodes, resulting in a combinatorial problem without replacement with solution $\binom{Z}{n_i}$. Thus, the total number of possible subgraphs is

$$C_{tot} = \prod_{i=1}^{M} \binom{Z}{n_i}^{d_{v,i}}. \tag{B.8}$$

Next, we calculate C_{ss}, the number of subgraphs induced by $\mathcal{V}_v$ which form a stopping set. Observe that the structure of the protograph keeps edge permutations on one edge type independent of permutations on all other edge types. Thus, the event that check nodes of type j satisfy the condition for forming stopping sets is independent of the same event for all other check-node types. This observation leads

to the following relation:

$$C_{ss} = \prod_{j=1}^{J} C_j \tag{B.9}$$

where C_j is the number of ways to connect the Z type-j check nodes to $\mathcal{V}_v$ such that each check node is connected to $\mathcal{V}_v$ at least twice or not at all.

Consider a check-node type j. Let the exponent of x_k denote the number of connections from check nodes of type j to variable nodes of type $\nu_{j,k}$ on the kth edge type emanating from check-node type j in the protograph. For each check node c_j of type j, the coefficient of the $\prod_{k=1}^{d_{c,j}} x_k^{b_k}$ term in the polynomial

$$p(\mathbf{x}) = \prod_{k=1}^{d_{c,j}} (1 + x_k) - \sum_{k=1}^{d_{c,j}} x_k \tag{B.10}$$

represents the number of possibilities for connecting c_j to b_k variable nodes of type $\nu_{j,k}$ such that c_j has a total of zero or at least two connections. There are Z check nodes of type j, so the polynomial is raised to the Zth power. Also, to form a complete subgraph induced by V_v, the set of Z type-j check nodes must be connected to $n_{\nu_{j,k}}$ variable nodes via type-k edges for every k from 1 to $d_{c,j}$. Thus,

$$C_j = \operatorname{coef}\left\{ \left[\prod_{k=1}^{d_{c,j}} (1 + x_k) - \sum_{k=1}^{d_{c,j}} x_k \right]^Z , \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\} \tag{B.11}$$

and combining this expression with (B.7)-(B.9) results in the following expression for $P_{\mathbf{n}}$:

$$P_{\mathbf{n}} = \frac{\prod_{j=1}^{J} \operatorname{coef}\left\{ \left[\prod_{k=1}^{d_{c,j}} (1 + x_k) - \sum_{k=1}^{d_{c,j}} x_k \right]^Z , \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\}}{\prod_{i=1}^{M} \binom{Z}{n_i}^{d_{v,i}}}. \tag{B.12}$$

Combining (B.5), (B.6), and (B.12) yields the result presented in the theorem. □

APPENDIX C

Derivation of the Approximation to the Stopping-Set Enumerator for Standard Ensembles

This appendix provides a proof of Theorem 3.2, which is restated here:

Theorem 3.2. *For a standard LDPC ensemble with $d_c > 2$, codelength n, and stopping-set size v such that*

$$v \leq xn \min_{\substack{1 \leq i \leq d_v : l_i \neq 0 \\ 1 \leq j \leq d_c : r_j \neq 0}} \left\{ l_i, \frac{2}{d_v}(1-R)r_j \right\}, \tag{C.1}$$

for any constant $x \in [0, 1/d_v)$, the expected number of stopping sets of size v is approximated by

$$s(n, v) = \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v - \lceil e/2 \rceil \pm \frac{O(v \log v)}{\log n}}. \tag{C.2}$$

Before proving Theorem 3.2, we will first need several lemmas.

Lemma C.1. *Let $0 \leq m \leq xn$ for any constant $x \in [0, 1]$. Then,*

$$(1-x)^m n^m \leq \frac{n!}{(n-m)!} \leq n^m \tag{C.3}$$

and

$$\frac{(1-x)^m n^m}{m!} \leq \binom{n}{m} \leq \frac{n^m}{m!} \tag{C.4}$$

Proof. First, consider

$$\frac{n!}{(n-m)!} = n(n-1)(n-2)\ldots(n-m+1).$$

Each of the m terms is upper bounded by n, resulting in

$$\frac{n!}{(n-m)!} \leq n^m \tag{C.5}$$

Also, each of the m terms is lower bounded by $n-m$, resulting in

$$\frac{n!}{(n-m)!} \geq (n-m)^m \geq (n-xn)^m = (1-x)^m n^m \tag{C.6}$$

where the second inequality follows from the condition $m \leq xn$. Combining (C.5) and (C.6) proves the result in (C.3).

To prove the second part of the lemma, observe that the binomial coefficient can be evaluated by the following expression:

$$\binom{n}{m} = \frac{n!}{(n-m)!m!} \tag{C.7}$$

Applying (C.3) directly to this expression proves the result in (C.4). □

Note that when $x = 1$, the lemma yields a lower bound of zero which does not provide a useful lower bound since it is already known that $\binom{n}{m}$ must be at least one. However, when $x < 1$, the lemma provides a more useful lower bound which grows polynomially with n.

Lemma C.2. *Let $k_i \geq 0$ for $i = 1, \ldots, M$ be integers such that $\sum_{i=1}^{M} k_i = v$ for some positive integer v. Then,*

$$\prod_{i=1}^{M} k_i! \leq v!. \tag{C.8}$$

Proof. Let $j = \arg\max_{1 \leq i \leq M} k_i$. Then, for all $i = 1, \ldots, M$, k_i satisfies $k_i \leq k_j \leq v$. Thus,

$$\begin{aligned} v! &= v(v-1)\cdots(k_j+1)k_j(k_j-1)\cdots 1 \\ &\geq (k_j+1)^{v-k_j} k_j! \\ &= k_j!(k_j+1)^{\sum_{i=1, i\neq j}^{M} k_i} = k_j! \prod_{i=1, i\neq j}^{M} (k_j+1)^{k_i} \\ &\geq k_j! \prod_{i=1, i\neq j}^{M} k_i^{k_i} \geq k_j! \prod_{i=1, i\neq j}^{M} k_i! = \prod_{i=1}^{M} k_i! \end{aligned} \tag{C.9}$$

where the second inequality follows from $k_i \leq k_j$. □

Lemma C.3. *Let $k_i \geq 0$ be integers such that $k_i \leq x_v n l_i$ for $1 \leq i \leq d_v$ for any constant $x_v \in [0, 1]$. For $|\mathcal{S}_v| \geq 1$, the quantity A_v given in (3.5) is upper and lower bounded as follows:*

$$n^v [(1 - x_v) \min_{1 \leq i \leq d_v : l_i \neq 0} l_i]^v v^{-v} \leq A_v \leq n^v (v+1)^{d_v}. \tag{C.10}$$

Proof. Recall from (3.5) that the quantity A_v, the first term in the numerator of the standard-ensemble stopping-set enumerator, is given by

$$A_v = \text{coef}\left\{ \prod_{i=1}^{d_v} (1 + yx^i)^{L_i}, y^v x^e \right\} = \sum_{\mathbf{k} \in \mathcal{S}_v} \prod_{i=1}^{d_v} \binom{n l_i}{k_i}. \tag{C.11}$$

Applying Lemma C.1 provides an upper bound on A_v:

$$\begin{aligned} A_v &\leq \sum_{\mathbf{k}\in\mathcal{S}_v} \prod_{i=1}^{d_v} \frac{(nl_i)^{k_i}}{k_i!} = n^v \left\{ \sum_{\mathbf{k}\in\mathcal{S}_v} \prod_{i=1}^{d_v} \frac{l_i^{k_i}}{k_i!} \right\} \\ &\leq n^v \left\{ \sum_{\mathbf{k}\in\mathcal{S}_v} \prod_{i=1}^{d_v} l_i^{k_i} \right\} \leq n^v |\mathcal{S}_v| \leq n^v (v+1)^{d_v}. \end{aligned} \tag{C.12}$$

The second inequality follows from $k_i! \geq 1$ and the third inequality follows from $l_i \leq 1$. The last inequality follows since the number of possible $\mathbf{k}$ vectors in $\mathcal{S}_v$ is limited by the number of possible values for each k_i, and for each $i \in \{1, \ldots, d_v\}$, k_i can take on at most $v+1$ values from 0 to v.

Next, A_v is lower bounded using Lemma C.1 as follows:

$$\begin{aligned} A_v &\geq \sum_{\mathbf{k}\in\mathcal{S}_v} \prod_{i=1}^{d_v} \frac{[(1-x_v)nl_i]^{k_i}}{k_i!} = n^v (1-x_v)^v \left\{ \sum_{\mathbf{k}\in\mathcal{S}_v} \frac{\prod_{i=1}^{d_v} l_i^{k_i}}{\prod_{i=1}^{d_v} k_i!} \right\} \\ &\geq n^v (1-x_v)^v \left\{ \sum_{\mathbf{k}\in\mathcal{S}_v} \frac{\prod_{i=1, l_i\neq 0}^{d_v} [\min_{1\leq i\leq d_v : l_i\neq 0} l_i]^{k_i}}{\prod_{i=1}^{d_v} k_i!} \right\} \\ &= n^v [(1-x_v) \min_{1\leq i\leq d_v : l_i\neq 0} l_i]^v \left\{ \sum_{\mathbf{k}\in\mathcal{S}_v} \frac{1}{\prod_{i=1}^{d_v} k_i!} \right\}. \end{aligned} \tag{C.13}$$

The denominator can be upper bounded with Lemma C.2:

$$\prod_{i=1}^{d_v} k_i! \leq v! \leq v^v. \tag{C.14}$$

Thus, we can further lower bound A_v.

$$\begin{aligned} A_v &\geq n^v [(1-x_v) \min_{1\leq i\leq d_v : l_i\neq 0} l_i]^v \left\{ \sum_{\mathbf{k}\in\mathcal{S}_v} \frac{1}{v^v} \right\} = n^v [(1-x_v) \min_{1\leq i\leq d_v : l_i\neq 0} l_i]^v v^{-v} |\mathcal{S}_v| \\ &\geq n^v [(1-x_v) \min_{1\leq i\leq d_v : l_i\neq 0} l_i]^v v^{-v} \end{aligned} \tag{C.15}$$

where the last inequality makes use of the assumption that $|\mathcal{S}_v| \geq 1$. □

Lemma C.4. *Let $n_{i,j}$ be the number of degree-i check nodes which are connected to the stopping set via j edges, and let $w_i = \sum_{j=2}^{i} n_{i,j}$ be the number of check nodes of degree i connected to the stopping set. Let $w_i \leq x_c n(1-R) r_i$ for $2 \leq i \leq d_c$ and any constant $x_c \in [0,1]$; $0 \leq v \leq 2n(1-R)(1-r_1)/d_v$; and $d_c > 2$. For $|\mathcal{S}_c| \geq 1$, the quantity A_c given in (3.7) is upper and lower bounded as follows:*

$$\frac{[n(1-R)(1-x_c)\min_{2\leq i\leq d_c: r_i \neq 0} r_i]^{\lfloor e/2 \rfloor}}{(e/2)!} \leq A_c \leq \frac{[n(1-R)]^{\lfloor e/2 \rfloor} 2^{d_c^3 e}(e+1)^{d_c^2}}{(e/d_c^3)!}. \quad \text{(C.16)}$$

Proof. First, the total number of check nodes connected to the stopping set is

$$w = \sum_{i=2}^{d_c} w_i = \sum_{i=2}^{d_c} \sum_{j=2}^{i} n_{i,j}. \quad \text{(C.17)}$$

These equalities hold since both i, the check-node degree, and j, the number of connections to the stopping set, must be greater than or equal to two in order for the connected check nodes to have at least two connections to the stopping set. We can bound w as follows:

$$w = \sum_{i=2}^{d_c} \sum_{j=2}^{i} n_{i,j} = \sum_{i=2}^{d_c} \sum_{j=2}^{i} \frac{j n_{i,j}}{j} \leq \frac{1}{2} \sum_{i=2}^{d_c} \sum_{j=2}^{i} j n_{i,j} = \frac{e}{2}. \quad \text{(C.18)}$$

Since w is an integer,

$$w \leq \left\lfloor \frac{e}{2} \right\rfloor. \quad \text{(C.19)}$$

Recall from (3.7) that the quantity A_c, the second term in the numerator of the standard-ensemble stopping-set enumerator, is given by

$$A_c = \text{coef}\left\{ \prod_{i=1}^{d_c} [(1+x)^i - ix]^{R_i}, x^e \right\} = \sum_{\mathbf{n} \in \mathcal{S}_c} \prod_{i=1}^{d_c} \left\{ \frac{(n(1-R)r_i)!}{\prod_{j=0, j\neq 1}^{i} n_{i,j}!} \prod_{\substack{j=0 \\ j\neq 1}}^{i} \binom{i}{j}^{n_{i,j}} \right\}. \quad \text{(C.20)}$$

A_c can be rewritten as

$$\begin{aligned} A_c &= \sum_{\mathbf{n}\in\mathcal{S}_c} \prod_{i=1}^{d_c} \left\{ \frac{(n(1-R)r_i)!}{n_{i,0}! \prod_{j=1}^{i} n_{i,j}!} \prod_{j=2}^{i} \binom{i}{j}^{n_{i,j}} \right\} \\ &= \sum_{\mathbf{n}\in\mathcal{S}_c} \prod_{i=2}^{d_c} \left\{ \frac{(n(1-R)r_i)!}{n_{i,0}! \prod_{j=2}^{i} n_{i,j}!} \prod_{j=2}^{i} \binom{i}{j}^{n_{i,j}} \right\} \end{aligned} \tag{C.21}$$

where the second equality holds since $n_{i,1} = 0$ for all i and the term in brackets is 1 when $i = 1$. An upper bound on A_c can be found by applying Lemma C.1 (with $m = w_i$) to (C.21) and noting that $n_{i,0} = n(1-R)r_i - w_i$:

$$\begin{aligned} A_c &\leq \sum_{\mathbf{n}\in\mathcal{S}_c} \prod_{i=2}^{d_c} \left\{ \frac{(n(1-R)r_i)^{w_i}}{\prod_{j=2}^{i} n_{i,j}!} \prod_{j=2}^{i} \binom{i}{j}^{n_{i,j}} \right\} \\ &= \sum_{\mathbf{n}\in\mathcal{S}_c} \left\{ (n(1-R))^w \frac{\prod_{i=2}^{d_c} r_i^{w_i}}{\prod_{i=2}^{d_c} \prod_{j=2}^{i} n_{i,j}!} \prod_{i=2}^{d_c} \prod_{j=2}^{i} \binom{i}{j}^{n_{i,j}} \right\} \\ &\leq \sum_{\mathbf{n}\in\mathcal{S}_c} \left\{ (n(1-R))^w \frac{1}{\prod_{i=2}^{d_c} \prod_{j=2}^{i} n_{i,j}!} \prod_{i=2}^{d_c} \prod_{j=2}^{i} \binom{i}{j}^{n_{i,j}} \right\} \end{aligned} \tag{C.22}$$

where the last inequality follows from $r_i \leq 1$. The denominator can be lower bounded by

$$\prod_{i=2}^{d_c} \prod_{j=2}^{i} n_{i,j}! \geq \max_{2\leq j\leq i\leq d_c} n_{i,j}! \geq \left(\frac{e}{d_c^3}\right)! \tag{C.23}$$

where the last inequality follows from the following relation:

$$e = \sum_{i=2}^{d_c} \sum_{j=2}^{i} j n_{i,j} \leq \sum_{i=2}^{d_c} \sum_{j=2}^{i} \left\{ d_c \max_{2\leq j\leq i\leq d_c} n_{i,j} \right\} \leq d_c^3 \max_{2\leq j\leq i\leq d_c} n_{i,j}. \tag{C.24}$$

Thus,

$$A_c \leq \sum_{\mathbf{n}\in\mathcal{S}_c} \left\{ \frac{(n(1-R))^w}{(e/d_c^3)!} \prod_{i=2}^{d_c} \prod_{j=2}^{i} \binom{i}{j}^{n_{i,j}} \right\}. \tag{C.25}$$

Observing that $\binom{n}{m} \leq \sum_{m=0}^{n} \binom{n}{m} = 2^n$ and $n_{i,j} \leq e$ for $i \geq j \geq 2$, A_c can be further upper bounded by

$$\begin{aligned} A_c &\leq \sum_{\mathbf{n} \in \mathcal{S}_c} \left\{ \frac{(n(1-R))^w}{(e/d_c^3)!} \prod_{i=2}^{d_c} \prod_{j=2}^{i} 2^{d_c e} \right\} \\ &\leq \sum_{\mathbf{n} \in \mathcal{S}_c} \left\{ \frac{(n(1-R))^w}{(e/d_c^3)!} 2^{d_c^3 e} \right\} = \frac{2^{d_c^3 e}}{(e/d_c^3)!} \sum_{\mathbf{n} \in \mathcal{S}_c} \{[n(1-R)]^w\} \\ &\leq \frac{2^{d_c^3 e}}{(e/d_c^3)!} (n(1-R))^{\lfloor e/2 \rfloor} |\mathcal{S}_c| \end{aligned} \tag{C.26}$$

where the last inequality follows since $w \leq \lfloor e/2 \rfloor$.

To upper bound $|\mathcal{S}_c|$, we only need to consider how many possible values $n_{i,j}$ takes for $i \geq j \geq 2$. Once these values are determined, then all other values of $n_{i,j}$ can be determined uniquely. Specifically, $n_{i,1} = 0$ for all i and thus, $n_{i,0} = n(1-R)r_i - \sum_{j=2}^{i} n_{i,j}$ for all i. Furthermore, for $i \geq j \geq 2$, we have that $0 \leq n_{i,j} \leq w \leq e/2 < e$, so $n_{i,j}$ can take at most e values. The number of terms with $i \geq j \geq 2$ is less than d_c^2 since $i \leq d_c$. Combining these last two results yields $|\mathcal{S}_c| \leq e^{d_c^2}$ which further upper bounds A_c as follows:

$$A_c \leq \frac{2^{d_c^3 e} e^{d_c^2}}{(e/d_c^3)!} (n(1-R))^{\lfloor e/2 \rfloor} \tag{C.27}$$

A lower bound on A_c can be found by applying Lemma C.1 (with $m = w_i$) to (C.21):

$$\begin{aligned} A_c &\geq \sum_{\mathbf{n} \in \mathcal{S}_c} \prod_{i=2}^{d_c} \left\{ \frac{[(1-x_c)n(1-R)r_i]^{w_i}}{\prod_{j=2}^{i} n_{i,j}!} \prod_{j=2}^{i} \binom{i}{j}^{n_{i,j}} \right\} \\ &\geq \sum_{\mathbf{n} \in \mathcal{S}_c} \left\{ [(1-x_c)n(1-R)]^w \frac{\prod_{i=2}^{d_c} r_i^{w_i}}{\prod_{i=2}^{d_c} \prod_{j=2}^{i} n_{i,j}!} \right\} \end{aligned} \tag{C.28}$$

where the last inequality follows from $\binom{i}{j} \geq 1$. By further manipulating the right-

hand side of the inequality, we obtain

$$A_c \geq \sum_{\mathbf{n}\in\mathcal{S}_c} \left\{ [(1-x_c)n(1-R)]^w \frac{\prod_{i=2,r_i\neq 0}^{d_c} \min_{2\leq i\leq d_c: r_i\neq 0} {r_i}^{w_i}}{\prod_{i=2}^{d_c}\prod_{j=2}^{i} n_{i,j}!} \right\}$$
$$= \sum_{\mathbf{n}\in\mathcal{S}_c} \left\{ [n(1-R)(1-x_c) \min_{2\leq i\leq d_c: r_i\neq 0} r_i]^w \frac{1}{\prod_{i=2}^{d_c}\prod_{j=2}^{i} n_{i,j}!} \right\}. \tag{C.29}$$

Observing that $\sum_{i=2}^{d_c}\sum_{j=2}^{i} n_{i,j} = w$ and applying Lemma C.2, we upper bound the denominator by

$$\prod_{i=2}^{d_c}\prod_{j=2}^{i} n_{i,j}! \leq w! \leq \left(\frac{e}{2}\right)! \tag{C.30}$$

where the last inequality follows from (C.18). Thus, combining this result with (C.29) results in

$$A_c \geq \sum_{\mathbf{n}\in\mathcal{S}_c} \left\{ [n(1-R)(1-x_c) \min_{2\leq i\leq d_c: r_i\neq 0} r_i]^w \frac{1}{(e/2)!} \right\}$$
$$= \frac{1}{(e/2)!} \sum_{\mathbf{n}\in\mathcal{S}_c} \left[n(1-R)(1-x_c) \min_{2\leq i\leq d_c: r_i\neq 0} r_i \right]^w. \tag{C.31}$$

The summation in (C.31) is a polynomial in n with degree $\max\{w : \mathbf{n} \in \mathcal{S}_c\}$ and with non-negative coefficients. From (C.19), $\max\{w : \mathbf{n} \in \mathcal{S}_c\} \leq \lfloor e/2 \rfloor$. In fact, we will show that $\max\{w : \mathbf{n} \in \mathcal{S}_c\} = \lfloor e/2 \rfloor$, i.e., there exists $\lfloor e/2 \rfloor$ check nodes which can be connected to the stopping set via e edges with each check node connected at least twice and thus, forming a valid stopping set with e edges.

First, we will show that $\lfloor e/2 \rfloor \leq n(1-R)(1-r_1)$, i.e., the number of check nodes in the stopping set does not exceed the total number of check nodes with degree at least two. Check nodes of degree one are not considered since they cannot be part

of a stopping set. The quantity $\lfloor e/2 \rfloor$ is upper bounded by

$$\begin{aligned} \left\lfloor \frac{e}{2} \right\rfloor &\leq \frac{e}{2} \leq \frac{d_v v}{2} \leq \frac{d_v}{2} \frac{2n(1-R)(1-r_1)}{d_v} \\ &= n(1-R)(1-r_1) \end{aligned} \tag{C.32}$$

where the second inequality follows since the number of edges in the stopping set is at most the number of variable nodes in the stopping set times the maximum variable-node degree. The third inequality follows from the assumption that $v \leq 2n(1-R)(1-r_1)/d_v$. Thus, we can indeed find $\lfloor e/2 \rfloor$ check nodes in the code with degree at least two.

Next, we will show that these $\lfloor e/2 \rfloor$ check nodes can be connected to the stopping set such that each check node is connected at least twice. First, suppose e is even. Then, we can connect each of the $\lfloor e/2 \rfloor$ check nodes to the stopping set via exactly two edges. Thus, there exists a valid stopping set which is connected to $w = \lfloor e/2 \rfloor$ check nodes via e edges.

Now, suppose e is odd. Note that $e \neq 1$ since a valid stopping set cannot be formed with only one edge. From the previous result, there exists a stopping set which is connected to $w = (e-1)/2$ check nodes, and each of these w check nodes is connected to the stopping set via exactly two edges. There remains only one more edge to connect to the check nodes. Out of the w check nodes, choose a check node c_1 with degree greater than two. If no such check node can be found, then choose a check node c_1 with degree greater than two from the other $n(1-R)-w$ check nodes. This can be done since $d_c > 2$. Now, choose any check node c_2 from the w check nodes. Disconnect the two edges connected to c_2, and connect these two edges to c_1. Finally, connect the last eth edge to c_1 so that c_1 is now connected to the stopping set via three edges.

We have demonstrated the existence of a valid stopping set, with variable-node distribution $\mathbf{n} \in \mathcal{S}_c$, connected to $w = \lfloor e/2 \rfloor$ check nodes via e edges. Thus, $\max_{\mathbf{n} \in \mathcal{S}_c} w = \lfloor e/2 \rfloor$.

Using this result with (C.31), we obtain

$$A_c \geq \frac{[n(1-R)(1-x_c)\min_{2 \leq i \leq d_c : r_i \neq 0} r_i]^{\lfloor e/2 \rfloor}}{(e/2)!} \tag{C.33}$$

since all other terms in the summation are non-negative. □

Note that Lemmas C.3 and C.4 only apply when $|\mathcal{S}_v| \geq 1$ and $|\mathcal{S}_c| \geq 1$, respectively. If $|\mathcal{S}_v| = 0$ or $|\mathcal{S}_c| = 0$, then the corresponding term in the summation in (3.4) is zero, so these cases are not of interest.

Next, recall from (3.9) that the quantity A_d, the denominator of the standard-ensemble stopping-set enumerator, is given by

$$A_d = \binom{L'(1)}{e} = \binom{n \sum_{i=1}^{d_v} i l_i}{e}. \tag{C.34}$$

To bound A_d, Lemma C.1 can be directly applied for $v \leq x_d n / d_v$ for any constant $x_d \in [0, 1)$ since $e \leq d_v v \leq x_d n \leq x_d n \sum_{i=1}^{d_v} i l_i = x_d L'(1)$. Thus,

$$\frac{[n(1-x_d)\sum_{i=1}^{d_v} i l_i]^e}{e!} \leq \binom{L'(1)}{e} \leq \frac{[n \sum_{i=1}^{d_v} i l_i]^e}{e!}. \tag{C.35}$$

We can now prove Theorem 3.2.

Proof of Theorem 3.2. We only consider terms in the summation of (3.4) with $|\mathcal{S}_v| \geq 1$ and $|\mathcal{S}_c| \geq 1$ since all other terms will be zero. Also, since valid stopping sets must have $v \leq e \leq d_v v$, we only need to sum terms from $e = v$ to $e = \min\{d_v v, L'(1)\}$.

Let $x = x_v = x_c = x_d / d_v$ be any constant $x \in [0, 1/d_v)$. Then, x_v, x_c, and x_d are all in $[0, 1)$. Using the assumption in (C.1), Lemma C.3 applies since for all

$i \in \{1, \ldots, d_v\}$,

$$k_i \leq v \leq xn \left\{ \min_{1 \leq j \leq d_v : l_j \neq 0} l_j \right\} \leq x(nl_i).$$

Also, Lemma C.4 applies since for all $i \in \{2, \ldots, d_c\}$,

$$w_i \leq w \leq \frac{e}{2} \leq \frac{d_v v}{2} \leq \frac{d_v}{2} xn \left\{ \frac{2(1-R)}{d_v} \min_{2 \leq j \leq d_c : r_j \neq 0} r_j \right\} \leq xn(1-R) r_i$$

and

$$v \leq xn \left\{ \frac{2(1-R)}{d_v} \min_{2 \leq j \leq d_c : r_j \neq 0} r_j \right\} \leq \frac{2n(1-R)}{d_v} \sum_{j=2}^{d_c} r_j = \frac{2n(1-R)(1-r_1)}{d_v}.$$

First, Lemma C.3, Lemma C.4, and (C.35) are used to find an upper bound on the standard-ensemble stopping-set enumerator $s(n, v)$.

$$\begin{aligned} s(n,v) &= \sum_{e=v}^{\min\{d_v v, L'(1)\}} \frac{A_v A_c}{A_d} \leq \sum_{e=v}^{\min\{d_v v, L'(1)\}} \frac{n^v (v+1)^{d_v} \frac{[n(1-R)]^{\lfloor e/2 \rfloor} 2^{d_c^3 e} e^{d_c^2}}{(e/d_c^3)!}}{\frac{[n(1-d_v x) \sum_{i=1}^{d_v} i l_i]^e}{e!}} \\ &= \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v - \lceil e/2 \rceil} \frac{(v+1)^{d_v} [(1-R)]^{\lfloor e/2 \rfloor} 2^{d_c^3 e} e^{d_c^2}}{\left[(1 - d_v x) \sum_{i=1}^{d_v} i l_i \right]^e} \frac{e!}{(e/d_c^3)!} \end{aligned} \tag{C.36}$$

The last term can be upper bounded with Lemma C.1:

$$\frac{e!}{(e/d_c^3)!} \leq e^{e - e/d_c^3} = e^{e(1 - 1/d_c^3)} \leq (d_v v)^{d_v v (1 - 1/d_c^3)} \tag{C.37}$$

where the last inequality follows from $e \leq d_v v$ (the number of edges is at most the number of variable nodes times the maximum variable-node degree) and $(1 - 1/d_c^3) \geq 0$. Thus,

$$s(n,v) \leq \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v - \lceil e/2 \rceil} \frac{(v+1)^{d_v} [(1-R)]^{\lfloor e/2 \rfloor} 2^{d_c^3 e} e^{d_c^2}}{\left[(1 - d_v x) \sum_{i=1}^{d_v} i l_i \right]^e} (d_v v)^{d_v v (1 - 1/d_c^3)}. \tag{C.38}$$

By further upper bounding the right-hand side of the inequality, we obtain

$$\begin{aligned} s(n,v) &\leq \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} \frac{(v+1)^{d_v} 2^{d_c^3 d_v v} [d_v(v+1)]^{d_c^2}}{\left[(1-d_v x)\sum_{i=1}^{d_v} i l_i\right]^v} (d_v v)^{d_v v(1-1/d_c^3)} \\ &= \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} \alpha_1 \alpha_2^v (v+1)^{\alpha_3} v^{d_v(1-1/d_c^3)v} \end{aligned} \tag{C.39}$$

where α_1, α_2, and α_3 are positive constants. The second inequality follows from $v \leq e \leq d_v v \leq d_v(v+1)$ and $R \geq 0$. Comparing the growth of the terms with n,

$$\begin{aligned} s(n,v) &\leq \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} n^{[\log \alpha_1 + v \log \alpha_2 + \alpha_3 \log(v+1) + d_v(1-1/d_c^3) v \log v]/\log n} \\ &= \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} n^{O(v \log v)/\log n} = \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\left\lceil \frac{e}{2} \right\rceil + \frac{O(v \log v)}{\log n}} \end{aligned} \tag{C.40}$$

Next, a lower bound on $s(n,v)$ is found using Lemma C.3, Lemma C.4, and (C.35).

$$\begin{aligned} s(n,v) &= \sum_{e=v}^{\min\{d_v v, L'(1)\}} \frac{A_v A_c}{A_d} \\ &\geq \sum_{e=v}^{\min\{d_v v, L'(1)\}} \frac{n^v \left[(1-x) \min\limits_{\substack{1 \leq i \leq d_v: \\ l_i \neq 0}} l_i\right]^v v^{-v} \left[n(1-R)(1-x) \min\limits_{\substack{2 \leq i \leq d_c: \\ r_i \neq 0}} r_i\right]^{\lfloor e/2 \rfloor} 2e!}{(e/2)! \left[n \sum_{i=1}^{d_v} i l_i\right]^e} \\ &= \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\left\lceil \frac{e}{2} \right\rceil} \frac{(1-x)^{v+\lfloor e/2 \rfloor} \left[\min\limits_{\substack{1 \leq i \leq d_v: \\ l_i \neq 0}} l_i\right]^v \left[(1-R) \min\limits_{\substack{2 \leq i \leq d_c: \\ r_i \neq 0}} r_i\right]^{\lfloor e/2 \rfloor} e!}{v^v \left[\sum_{i=1}^{d_v} i l_i\right]^e (e/2)!}. \end{aligned} \tag{C.41}$$

Since $e \leq d_v v$, we can further lower bound $s(n,v)$ by

$$s(n,v) \geq \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil \frac{e}{2} \rceil} \frac{(1-x)^{v+d_v v/2} \left[\min_{\substack{1 \leq i \leq d_v: \\ l_i \neq 0}} l_i \right]^v \left[(1-R) \min_{\substack{2 \leq i \leq d_c: \\ r_i \neq 0}} r_i \right]^{d_v v/2} e!}{v^v \left[\sum_{i=1}^{d_v} i l_i \right]^{d_v v} (e/2)!}$$

$$\geq \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} \alpha_1' \alpha_2'^v v^{-v} \frac{e!}{(e/2)!} \tag{C.42}$$

where α_1' and α_2' are positive constants. The last term can be bounded with Lemma C.1:

$$\frac{e!}{(e/2)!} \geq \left[\frac{1}{2} e \right]^{e/2} \tag{C.43}$$

Thus, we can further lower bound $s(n,v)$ as follows:

$$s(n,v) \geq \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} \alpha_1' \alpha_2'^v v^{-v} \left[\frac{1}{2} e \right]^{e/2}$$

$$\geq \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} \alpha_1' \alpha_2'^v v^{-v} \left[\frac{1}{2} v \right]^{v/2} \tag{C.44}$$

where the second inequality follows from $e \geq v$. Comparing the growth of the terms with n,

$$s(n,v) \geq \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} n^{[\log \alpha_1' + v \log \alpha_2' - v \log(v) + (v/2) \log(1/2) + (v/2) \log v]/\log n}$$

$$\geq \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil} n^{[-O(v \log(v))]/\log n}$$

$$= \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2 \rceil - \frac{O(v \log(v))}{\log n}} \tag{C.45}$$

Combining (C.40) and (C.45),

$$\sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2\rceil - \frac{O(v\log(v))}{\log n}} \leq s(n,v) \leq \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2\rceil + \frac{O(v\log(v))}{\log n}} \tag{C.46}$$

and thus,

$$s(n,v) = \sum_{e=v}^{\min\{d_v v, L'(1)\}} n^{v-\lceil e/2\rceil \pm \frac{O(v\log(v))}{\log n}} \tag{C.47}$$

□

APPENDIX D

Derivation of the Approximation to the Stopping-Set Enumerator for Protographs

This appendix provides a proof of Theorem 3.3, which is restated here:

Theorem 3.3. *For an LDPC ensemble based on Z copies of a protograph with M variable-node types, J check-node types, E edge types, codelength $n = ZM$, and stopping-set size $v < n$, the expected number of stopping sets of size v is approximated by*

$$s(n, v) = \sum_{\mathbf{n} \in \mathcal{S}_p} Z^{v-e+w^* \pm \frac{O(v \log v)}{\log n}} \tag{D.1}$$

where e is the number of edges emanating from the set of variable nodes, $\mathcal{V}_v$, which is distributed according to $\mathbf{n}$, and $\mathcal{S}_p$ is defined in (3.11)*. The quantity $w^* = \sum_{j=1}^{J} w_j^*$ where $w_j^* = \max_{\mathbf{m} \in \mathcal{S}_m} \{w_j\}$ and*

$$\mathcal{S}_m = \left\{ \mathbf{m} \in \mathbb{Z}^{|\mathcal{B}|} : \; 0 \le m_h \le Z, \; \forall h = 0, \ldots, |\mathcal{B}| - 1; \; \sum_{h=0}^{|\mathcal{B}|-1} m_h = Z; \right. \\ \left. \sum_{h=0}^{|\mathcal{B}|-1} \beta_{h,k} m_h = n_{\nu_{j,k}}, \; \forall k = 1, \ldots, d_{c,j} \right\} \tag{D.2}$$

for each $j \in \{1, \ldots, J\}$.

Proof. The equation for the protograph-based ensemble stopping-set enumerator $s(n, v)$, given in Theorem 3.1, can be expressed as follows:

$$s(n,v) = \sum_{\mathbf{n}\in\mathcal{S}_p} \frac{1}{A_1} \prod_{j=1}^{J} A_2 \tag{D.3}$$

where

$$A_1 = \prod_{i=1}^{M} \binom{Z}{n_i}^{d_{v,i}-1} \tag{D.4}$$

$$A_2 = \text{coef}\left\{ \left[\prod_{k=1}^{d_{c,j}} (1+x_k) - \sum_{k=1}^{d_{c,j}} x_k \right]^{Z}, \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\}. \tag{D.5}$$

First, A_1 will be upper bounded by applying Lemma C.1.

$$\begin{aligned} A_1 &\le \prod_{i=1}^{M} \left[\frac{Z^{n_i}}{n_i!} \right]^{d_{v,i}-1} = \frac{Z^{\sum_{i=1}^{M}(n_i d_{v,i} - n_i)}}{\prod_{i=1}^{M} (n_i!)^{d_{v,i}-1}} \\ &= \frac{Z^{e-v}}{\prod_{i=1}^{M} (n_i!)^{d_{v,i}-1}} \le Z^{e-v}. \end{aligned} \tag{D.6}$$

A_1 is also lower bounded using Lemma C.1:

$$\begin{aligned} A_1 &\ge \prod_{i=1}^{M} \left[\frac{[(1-x)Z]^{n_i}}{n_i!} \right]^{d_{v,i}-1} = \frac{[(1-x)Z]^{\sum_{i=1}^{M}(n_i d_{v,i} - n_i)}}{\prod_{i=1}^{M} (n_i!)^{d_{v,i}-1}} \\ &= \frac{[(1-x)Z]^{e-v}}{\prod_{i=1}^{M} (n_i!)^{d_{v,i}-1}} \ge \frac{[(1-x)Z]^{e-v}}{\prod_{i=1}^{M} (n_i!)^{d_v-1}} \ge \frac{[(1-x)Z]^{e-v}}{(v!)^{d_v-1}} \end{aligned} \tag{D.7}$$

where the second inequality follows since $d_{v,i} \le d_v$ for all i and the last inequality follows from Lemma (C.2).

Next, A_2 will be evaluated and bounded for each check-node type j. Let the exponent of x_k denote the number of connections from check nodes of type j to variable nodes of type $\nu_{j,k}$ on the kth edge type emanating from check node type j in

the protograph. Then, for each of the check nodes of type j, the following polynomial represents connections from it to the stopping set $\mathcal{V}_v$:

$$p(\mathbf{x}) = \prod_{k=1}^{d_{c,j}} (1 + x_k) - \sum_{k=1}^{d_{c,j}} x_k. \tag{D.8}$$

Let $b_k^{(z)} \in \{0, 1\}$ denote the exponent of the x_k term for the zth check node of type j and let $\mathbf{b}^{(z)} = (b_1^{(z)}, b_2^{(z)}, \ldots, b_{d_{c,j}}^{(z)})$ be the vector of all such coefficients. Then,

$$p(\mathbf{x}) = \sum_{\mathbf{b} \in \mathcal{B}} \left[\prod_{k=1}^{d_{c,j}} x_k^{b_k^{(z)}} \right] \tag{D.9}$$

where $\mathcal{B}$ is defined in (3.29). The set $\mathcal{B}$ excludes all vectors containing exactly one 1 since these unit vectors correspond to the case when a check node is only connected once to $\mathcal{V}_v$ and hence a stopping set cannot be formed. Using these notations, A_2 can be expressed as

$$\begin{aligned} A_2 &= \text{coef} \left\{ p^Z(\mathbf{x}), \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\} \\ &= \text{coef} \left\{ \sum_{\mathbf{b}^{(1)} \in \mathcal{B}} \cdots \sum_{\mathbf{b}^{(Z)} \in \mathcal{B}} \left[\prod_{k=1}^{d_{c,j}} x_k^{b_k^{(1)}} \cdots \prod_{k=1}^{d_{c,j}} x_k^{b_k^{(Z)}} \right], \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\} \\ &= \text{coef} \left\{ \sum_{\mathbf{b}^{(1)} \in \mathcal{B}} \cdots \sum_{\mathbf{b}^{(Z)} \in \mathcal{B}} \left[\prod_{k=1}^{d_{c,j}} x_k^{\sum_{z=1}^{Z} b_k^{(z)}} \right], \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\}. \end{aligned} \tag{D.10}$$

The exponent of x_k in $p^Z(\mathbf{x})$ is only dependent on the distribution of the values that the vectors $\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(Z)}$ take in $\mathcal{B}$ and is not dependent on z. Thus, utilizing the enumeration of elements in $\mathcal{B} = \{\beta_0, \beta_1, \ldots, \beta_{|\mathcal{B}|-1}\}$ and the distribution of check

node connections given by $\mathbf{m}$, A_2 can be expressed as follows:

$$A_2 = \text{coef}\left\{ \sum_{\substack{\mathbf{m}\in\mathbb{Z}^{|\mathcal{B}|}: \\ 0\le m_h\le Z \\ \sum_{h=0}^{|\mathcal{B}|-1} m_h = Z}} \binom{Z}{m_0,\cdots,m_{|\mathcal{B}|-1}} \prod_{k=1}^{d_{c,j}} x_k^{\sum_{h=0}^{|\mathcal{B}|-1}\beta_{h,k}m_h}, \prod_{k=1}^{d_{c,j}} x_k^{n_{\nu_{j,k}}} \right\}$$
$$= \sum_{\mathbf{m}\in\mathcal{S}_m} \binom{Z}{m_0,\cdots,m_{|\mathcal{B}|-1}} = \sum_{\mathbf{m}\in\mathcal{S}_m} \frac{Z!}{m_0!\prod_{h=1}^{|\mathcal{B}|-1} m_h!}. \tag{D.11}$$

With this expression for A_2, we can now find bounds on A_2. First, note that the sum of elements in $\mathbf{m}$ must be Z since there are Z check nodes of type j. Also, m_0 denotes the number of check nodes not connected to $\mathcal{V}_v$. Since w_j is the number of check nodes of type j connected to $\mathcal{V}_v$, we have that $w_j = Z - m_0$. Applying Lemma C.1, we obtain the following upper bound on A_2:

$$A_2 \le \sum_{\mathbf{m}\in\mathcal{S}_m} \frac{Z^{w_j}}{\prod_{h=1}^{|\mathcal{B}|-1} m_h!} \le \sum_{\mathbf{m}\in\mathcal{S}_m} \frac{Z^{w_j}}{\max_{1\le h\le|\mathcal{B}|-1} m_h!}. \tag{D.12}$$

To bound the denominator, observe that the number of edges from $\mathcal{V}_v$ that connect to check nodes of type j is

$$e_j = \sum_{h=1}^{|\mathcal{B}|-1}\sum_{k=1}^{d_{c,j}} \beta_{h,k}m_h \ge \sum_{h=1}^{|\mathcal{B}|-1} 2m_h \ge 2\max_{1\le h\le|\mathcal{B}|-1} m_h \ge 2m_h \tag{D.13}$$

for all $h = 1,\ldots,|\mathcal{B}|-1$. The second inequality follows from the fact that for each $\beta \in \mathcal{B}\backslash\beta_0$, the vector β must contain at least two 1's. With this result, A_2 can be further upper bounded by

$$A_2 \le \sum_{\mathbf{m}\in\mathcal{S}_m} \frac{Z^{w_j}}{(e_j/2)!} \le |\mathcal{S}_m|\frac{Z^{w_j^*}}{(e_j/2)!} \tag{D.14}$$

where $w_j^* = \max_{\mathbf{m}\in\mathcal{S}_m}\{w_j\}$. By (D.13), each m_h can take at most $(e_j/2+1)$ values

and thus, $|\mathcal{S}_m|$ can be bounded as follows:

$$|\mathcal{S}_m| \leq \left(\frac{e_j}{2}+1\right)^{|\mathcal{B}|} = \left(\frac{e_j}{2}+1\right)^{2^{(d_{c,j})}-d_{c,j}} \leq (e+1)^{2^{d_{c,j}}}. \tag{D.15}$$

Combining these results, yields the upper bound

$$A_2 \leq Z^{w_j^*}\frac{(e+1)^{2^{d_{c,j}}}}{(e_j/2)!}. \tag{D.16}$$

To obtain a lower bound on A_2, Lemma C.1 is applied to (D.11).

$$A_2 \geq \sum_{\mathbf{m}\in\mathcal{S}_m}\frac{[(1-x)Z]^{w_j}}{\prod_{h=1}^{|\mathcal{B}|-1} m_h!} \geq \sum_{\mathbf{m}\in\mathcal{S}_m}\frac{[(1-x)Z]^{w_j}}{w_j!} \tag{D.17}$$

where the second inequality follows from Lemma C.2 since $\sum_{h=1}^{|\mathcal{B}|-1} m_h = w_j$. Finally, the summation can be lower bounded by a single term:

$$A_2 \geq \frac{[(1-x)Z]^{w_j^*}}{w_j^*!}. \tag{D.18}$$

Now that upper and lower bounds have been obtained for A_1 and A_2, we can now bound $s(n,v)$. First, let $w^* = \sum_{j=1}^{J} w_j^*$ be the largest possible number of check nodes connected to stopping set $\mathcal{V}_v$. Applying (D.7) and (D.16) to (D.3) gives an upper bound on $s(n,v)$:

$$\begin{aligned} s(n,v) &\leq \sum_{\mathbf{n}\in\mathcal{S}_p}\frac{\prod_{j=1}^{J}\left[Z^{w_j^*}(e+1)^{2^{d_{c,j}}}/(e_j/2)!\right]}{[(1-x)Z]^{e-v}(v!)^{1-d_v}} \\ &= \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+\sum_{j=1}^{J} w_j^*}\frac{(1-x)^{v-e}(e+1)^{\sum_{j=1}^{J}2^{d_{c,j}}}(v!)^{d_v-1}}{\prod_{j=1}^{J}(e_j/2)!} \\ &\leq \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*}(1-x)^{v-d_v v}(d_v v+1)^{\sum_{j=1}^{J}2^{d_{c,j}}} v^{v(d_v-1)}. \end{aligned} \tag{D.19}$$

Thus,

$$s(n,v) \leq \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} \alpha_1^v (d_v v+1)^{\alpha_2 v^{\alpha_3 v}} \tag{D.20}$$

where α_1, α_2, and α_3 are positive constants and the second inequality follows from $e \leq d_v v$. Comparing the growth of the terms with Z,

$$\begin{aligned} s(n,v) &\leq \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} Z^{[v\log\alpha_1+\alpha_2\log(d_v v+1)+\alpha_3 v\log v]/\log Z} \\ &= \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} Z^{O(v\log v)/\log Z} \\ &= \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*+\frac{O(v\log v)}{\log Z}} = \sum_{\mathbf{n}\in\mathcal{S}_p} n^{v-e+w^*+\frac{O(v\log v)}{\log n}} \end{aligned} \tag{D.21}$$

Next, applying (D.6) and (D.18) to (D.3) gives a lower bound on $s(n,v)$:

$$\begin{aligned} s(n,v) &\geq \sum_{\mathbf{n}\in\mathcal{S}_p} \frac{1}{Z^{e-v}} \prod_{j=1}^{J} \frac{[(1-x)Z]^{w_j^*}}{w_j^*!} \\ &= \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e} \frac{[(1-x)Z]^{\sum_{j=1}^J w_j^*}}{\prod_{j=1}^J (w_j^*!)} \\ &\geq \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} \frac{(1-x)^{w^*}}{w^*!} \geq \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} \frac{(1-x)^e}{e!} \\ &\geq \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} (1-x)^e e^{-e} \\ &\geq \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} (1-x)^{d_v v} (d_v v)^{-d_v v} \\ &= \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} \alpha_1'^v v^{-d_v v} \end{aligned} \tag{D.22}$$

where α_1' is a positive constant. The second inequality follows from Lemma C.2 and the third inequality follows from the fact that the number of check nodes w^* connected to the stopping set cannot be larger than the number of edges e emanating

from the stopping set. Comparing the growth of the terms with Z,

$$\begin{aligned} s(n,v) &\geq \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} Z^{[v\log\alpha_1' - d_v v\log v]/\log Z} \\ &= \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*} Z^{-O(v\log v)/\log Z} \\ &= \sum_{\mathbf{n}\in\mathcal{S}_p} Z^{v-e+w^*-\frac{O(v\log v)}{\log Z}} = \sum_{\mathbf{n}\in\mathcal{S}_p} n^{v-e+w^*-\frac{O(v\log v)}{\log n}}. \end{aligned} \tag{D.23}$$

Finally, combining (D.21) and (D.23), we obtain

$$\sum_{\mathbf{n}\in\mathcal{S}_p} n^{v-e+w^*-\frac{O(v\log v)}{\log n}} \leq s(n,v) \leq \sum_{\mathbf{n}\in\mathcal{S}_p} n^{v-e+w^*+\frac{O(v\log v)}{\log n}} \tag{D.24}$$

and thus,

$$s(n,v) = \sum_{\mathbf{n}\in\mathcal{S}_p} n^{v-e+w^*\pm\frac{O(v\log v)}{\log n}} \tag{D.25}$$

□

APPENDIX E

Proof of Region of Approximation Validity

This appendix provides a proof of Theorem 3.4, which is restated here:

Theorem 3.4. *Consider a protograph-based ensemble for which at most one edge type connects any variable-node type to any check-node type in the protograph. Given a small fraction $\gamma > 0$ and any constant $a \in (0, 1)$, let N_0 be the solution of n in the following equation*

$$n \ln n = \frac{Mv}{\gamma} \left(1 + \frac{a}{2(1-a)^2}\right) \left(1 + \frac{1}{d_{v,avg} - 2}\right) \tag{E.1}$$

where $d_{v,avg}$ is the smallest average variable-node degree in stopping sets of size v. Then, for all v and n satisfying the conditions $v/n \leq a/M$ and $n > N_0$, the error term is upper bounded by

$$|A| \leq \gamma |(v - e + w^*) \ln n| \tag{E.2}$$

and hence, for small γ, the stopping-set enumerator can be approximated by

$$\ln s_{\mathbf{n},\mathbf{m}}(n, v) \approx \ln c + (v - e + w^*) \ln n. \tag{E.3}$$

Before proving Theorem 3.4, we will first need the following lemmas.

Lemma E.1. *For $y \in [0, a]$ for any constant $a \in (0, 1)$,*

$$-y\left(1 + \frac{a}{2(1-a)^2}\right) \leq \ln(1-y) \leq -y. \tag{E.4}$$

Proof. Consider the Taylor series expansion of $\ln x$ around $x = 1$ with the Lagrange remainder:

$$\ln x = (x-1) - \frac{(x-1)^2}{2x^{*2}} \tag{E.5}$$

for some $x^* \in [1, x]$. Thus, for $y = 1 - x$ around $y = 0$,

$$\ln(1-y) = -y - \frac{y^2}{2(1-y^*)^2} = -y\left(1 + \frac{y}{2(1-y^*)^2}\right) \tag{E.6}$$

for some $y^* \in [0, y]$. The second inequality in the lemma follows directly:

$$\ln(1-y) = -y - \frac{y^2}{2(1-y^*)^2} \leq -y. \tag{E.7}$$

The first inequality of the lemma follows from the bound on $y^* \in [0, y]$:

$$\ln(1-y) \geq -y\left(1 + \frac{y}{2(1-y)^2}\right). \tag{E.8}$$

Now, suppose $y \in [0, a]$ for some constant $a \in (0, 1)$. Then,

$$\ln(1-y) \geq -y\left(1 + \frac{a}{2(1-a)^2}\right). \tag{E.9}$$

□

Lemma E.2. *Let $a \in (0,1)$ be a constant and let A be defined as in* (3.43)*:*

$$A = \sum_{i=1}^{M} \left[(1 - d_{v,i}) \sum_{k=1}^{n_i - 1} \ln\left(1 - \frac{k}{Z}\right) \right] + \sum_{j=1}^{J} \sum_{k=1}^{w_j - 1} \ln\left(1 - \frac{k}{Z}\right). \tag{E.10}$$

For $v/n \leq a/M$ and $d_{v,avg} \geq 4/3$, $|A|$ is upper bounded by

$$|A| \leq \frac{1}{Z} \left(1 + \frac{a}{2(1-a)^2}\right) (d_{v,avg} - 1) \frac{v^2}{2}. \tag{E.11}$$

Proof. First, the restriction on v/n allows us to apply Lemma E.1 to (E.10) since for all $k = 1, \ldots, n_i - 1$,

$$\frac{k}{Z} \leq \frac{n_i - 1}{Z} \leq \frac{v}{Z} = M\frac{v}{n} \leq a \tag{E.12a}$$

and for all $k = 1, \ldots, w_j - 1$,

$$\frac{k}{Z} \leq \frac{w_j - 1}{Z} \leq \frac{e_j}{2Z} \leq \frac{v}{2Z} = \frac{M}{2}\frac{v}{n} \leq \frac{a}{2} \leq a \tag{E.12b}$$

where the third inequality follows from the assumption that at most one edge type connects any variable-node type to any check-node type. Applying Lemma E.1 to (E.10) results in the following upper bound:

$$\begin{aligned} A &\leq \sum_{i=1}^{M} \left[(1 - d_{v,i}) \sum_{k=1}^{n_i - 1} -\frac{k}{Z} \left(1 + \frac{a}{2(1-a)^2}\right) \right] + \sum_{j=1}^{J} \sum_{k=1}^{w_j - 1} -\frac{k}{Z} \\ &= \frac{1}{Z} \sum_{i=1}^{M} \left[(d_{v,i} - 1) \frac{n_i(n_i - 1)}{2} \left(1 + \frac{a}{2(1-a)^2}\right) \right] - \frac{1}{Z} \sum_{j=1}^{J} \frac{w_j(w_j - 1)}{2}. \end{aligned} \tag{E.13}$$

We can further upper bound A as follows:

$$\begin{aligned}
A &\leq \frac{1}{Z}\sum_{i=1}^{M}\left[(d_{v,i}-1)\frac{n_i(n_i-1)}{2}\left(1+\frac{a}{2(1-a)^2}\right)\right] \\
&\leq \frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{1}{2}\sum_{i=1}^{M}(d_{v,i}-1)n_i^2 \\
&\leq \frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{v}{2}\sum_{i=1}^{M}(n_i d_{v,i}-n_i) \\
&= \frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{v}{2}(e-v) \\
&= \frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{d_{v,avg}-1}{2}v^2. \qquad \text{(E.14)}
\end{aligned}$$

A can also be lower bounded using Lemma E.1 as follows:

$$\begin{aligned}
A &\geq \sum_{i=1}^{M}\left[(1-d_{v,i})\sum_{k=1}^{n_i-1}-\frac{k}{Z}\right]+\sum_{j=1}^{J}\sum_{k=1}^{w_j-1}-\frac{k}{Z}\left(1+\frac{a}{2(1-a)^2}\right) \\
&= \frac{1}{Z}\sum_{i=1}^{M}\left[(d_{v,i}-1)\frac{n_i(n_i-1)}{2}\right]-\frac{1}{Z}\sum_{j=1}^{J}\frac{w_j(w_j-1)}{2}\left(1+\frac{a}{2(1-a)^2}\right) \\
&\geq -\frac{1}{Z}\sum_{j=1}^{J}\frac{w_j(w_j-1)}{2}\left(1+\frac{a}{2(1-a)^2}\right). \qquad \text{(E.15)}
\end{aligned}$$

Since $w_j \leq e_j/2$, we can further lower bound A by

$$\begin{aligned}
A &\geq \frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{1}{2}\sum_{j=1}^{J}-\left(\frac{e_j}{2}\right)^2 \\
&= -\frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{1}{8}\sum_{j=1}^{J}\left(\sum_{i=1}^{M}n_i\nu_{i,j}\right)^2 \\
&= -\frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{1}{8}\sum_{j=1}^{J}\sum_{i=1}^{M}n_i\nu_{i,j}\sum_{k=1}^{M}n_k\nu_{k,j} \\
&= -\frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{1}{8}\sum_{i=1}^{M}n_i\sum_{k=1}^{M}n_k\sum_{j=1}^{J}\nu_{i,j}\nu_{k,j} \qquad \text{(E.16)}
\end{aligned}$$

where $\nu_{i,j} \in \{0,1\}$ is the number of connections between variable-node type i and check-node type j in the protograph. Based on the assumption that as most one edge type connects any variable-node type to any check-node type, we obtain

$$\begin{aligned} A &\geq -\frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{1}{8}\sum_{i=1}^{M} n_i \sum_{k=1}^{M} n_k d_{v,k} \\ &= -\frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{1}{8} ve \\ &= -\frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{d_{v,avg}}{8} v^2. \end{aligned} \tag{E.17}$$

This lower bound on A is negative while the upper bound on A in (E.14) is positive. Thus, combining (E.14) and (E.17), the magnitude of A can be bounded by

$$\begin{aligned} |A| &\leq \frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\max\left\{\frac{d_{v,avg}}{4}, d_{v,avg}-1\right\}\frac{v^2}{2} \\ &= \frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{d_{v,avg}-1}{2} v^2 \end{aligned} \tag{E.18}$$

where the equality follow from $d_{v,avg} \geq 4/3$. □

We can now prove Theorem 3.4.

Proof. We wish to find a value of N_0 such that for all $n > N_0$,

$$|A| \leq \gamma |(v - e + w^*) \ln n|. \tag{E.19}$$

This is possible since the bound on $|A|$ in Lemma E.2 decreases monotonically with $n = MZ$. Equating the right-hand side of (E.19) with the upper bound in

Lemma E.2,

$$\gamma|(v-e+w^*)\ln n| = \frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{d_{v,avg}-1}{2}v^2, \tag{E.20}$$

and solving for n provides a value of N_0.

To obtain an expression which is easier to analyze, we assume that $|v-e+w^*| = e-v-w^* \geq 0$. If not, then the stopping-set enumerator grows polynomially with Z, which is an undesirable property for a code. Using this assumption, we obtain the following lower bound:

$$|v-e+w^*| = e-v-w^* \geq e-v-\frac{e}{2} = \frac{v}{2}(d_{v,avg}-2) \tag{E.21}$$

where $d_{v,avg}$ is the average variable-node degree in the stopping set. Using this inequality, another (larger) value for N_0 can be obtained by finding n satisfying the following equality:

$$\gamma|\frac{v}{2}(d_{v,avg}-2)\ln n| = \frac{1}{Z}\left[1+\frac{a}{2(1-a)^2}\right]\frac{d_{v,avg}-1}{2}v^2. \tag{E.22}$$

Rearranging the terms in (E.22), we obtain the desired equality:

$$\begin{aligned} n\ln n &= \frac{Mv}{\gamma}\left[1+\frac{a}{2(1-a)^2}\right]\frac{d_{v,avg}-1}{d_{v,avg}-2} \\ &= \frac{Mv}{\gamma}\left[1+\frac{a}{2(1-a)^2}\right]\left(1+\frac{1}{d_{v,avg}-2}\right). \end{aligned} \tag{E.23}$$

For $n \geq N_0$ where N_0 is the solution of n in (E.23), the inequality in (E.19) holds. Thus, the error term A is γ times smaller than the $(v-e+w^*)\ln n$ term, and hence the approximation in (E.3) holds for small γ. □

APPENDIX F

Proof of w_j^* Evaluation for Protograph-Based Ensemble Enumerator Exponents

This appendix provides a proof of Theorem 3.5, which is restated here:

Theorem 3.5. *Consider a set $\mathcal{V}$ of variable nodes with distribution $\mathbf{n} = (n_1, \ldots, n_M)$ where n_i is the number of type-i variable nodes in the set.*

For each check-node type j, if

$$2 \max_{1 \le i \le M} n_i \nu_{i,j} \le \sum_{i=1}^{M} n_i \nu_{i,j}, \tag{F.1}$$

then we can connect type-j check nodes to $\mathcal{V}$ such that $\mathcal{V}$ is a stopping set. Further,

$$w_j^* = \lfloor e_j/2 \rfloor \tag{F.2}$$

where $e_j = \sum_{i=1}^{M} n_i \nu_{i,j}$ is the total number of edges connected to type-j check nodes.

If (F.1) *is not satisfied for any check-node type j, then $\mathcal{V}$ cannot be a stopping set.*

Proof. Since permutations in the protograph expansion are independent for each edge type in the protograph, the connections from a set of variable nodes to the check

nodes are independent for each check-node type. Thus, we can consider each check-node type j separately for $j = 1, \ldots, J$. The set of variable nodes $\mathcal{V}$ is connected to type-j check nodes through $n_i \nu_{i,j}$ edges from type-i variable nodes for each i.

First, consider the case when (F.1) is not satisfied. Let $i^* = \arg\max_{1 \le i \le M} n_i \nu_{i,j}$. Then, $\nu_{i^*,j} = 1$ and the n_{i^*} type-i^* variable nodes must be connected to n_{i^*} unique type-j check nodes. In order to form a stopping set, the type-j check nodes must be connected at least twice or not at all. Thus, the remaining variable nodes of type $i \neq i^*$ must connect to at least these n_{i^*} type-j check nodes. However, the number of connections to type-j check nodes that we can obtain from the remaining variable-node types is bounded by

$$\sum_{i=1}^{M} n_i \nu_{i,j} - n_{i^*} < n_{i^*} \tag{F.3}$$

since (F.1) is not satisfied. Thus, the remaining variable nodes cannot provide enough edges to cover the n_{i^*} singly-connected type-j check nodes. This result shows that we cannot form a valid stopping set with this variable-node distribution $\mathbf{n}$.

Next, consider the case when (F.1) is satisfied. Since forming a stopping set requires either 0 or at least 2 connections to each check node, the maximum number of type-j check nodes is $\lfloor e_j/2 \rfloor$, where e_j is the total number of edges connecting type-j check nodes to the stopping set. Thus,

$$w_j^* \le \lfloor e_j/2 \rfloor. \tag{F.4}$$

We provide a constructive argument to show that we can indeed form a stopping set with $\mathcal{V}$ connected to exactly $\lfloor e_j/2 \rfloor$ check nodes for each j. Hence, w_j^* achieves its bound, resulting in (F.2).

We connect $\mathcal{V}$ to the type-j check nodes using the following recursive procedure.

Without loss of generality, we assume that the terms $n_i\nu_{i,j}$ are ordered from largest to smallest, i.e., $n_1\nu_{1,j} \geq n_2\nu_{2,j} \geq \cdots \geq n_M\nu_{M,j}$. Define c_k to be the number of type-j check nodes which have exactly one connection at iteration k. Define $s_k = \sum_{i=k}^{M} n_i\nu_{i,j}$ to be the number of edges that still need to be connected to the type-j check nodes at iteration k.

Step 1: Initialize $k = 1$. Connect the $n_1\nu_{1,j}$ type-1 variable nodes to $n_1\nu_{1,j}$ unique type-j check nodes. Then, set $c_2 = n_1\nu_{1,j}$.

Step 2: Increment $k = k + 1$. The next operation depends on the relationship between s_k and c_k.

- If $s_k = c_k$, proceed to **Step 2a**.
- If $s_k = c_k + 1$, proceed to **Step 2b**.
- If $s_k > c_k + 1$, proceed to **Step 2c**.
- The case where $s_k < c_k$ can never occur. For $k = 2$, the condition in (F.1) gives

$$c_2 = n_1\nu_{1,j} = \max_{1\leq i\leq M} n_i\nu_{i,j} \leq s_2. \tag{F.5}$$

  For $k > 2$, the choice of x_{k-1} in Step 2c guarantees that

$$\begin{aligned} c_k &= c_{k-1} - n_{k-1}\nu_{k-1,j} + 2x_{k-1} \\ &\leq c_{k-1} - n_{k-1}\nu_{k-1,j} + 2\left\lfloor \frac{s_{k-1} - c_{k-1}}{2} \right\rfloor \\ &= s_{k-1} - n_{k-1}\nu_{k-1,j} = s_k \end{aligned} \tag{F.6}$$

Step 2a: In this case, $s_k = c_k$. Connect the s_k variable nodes (of type k to M) to the c_k check nodes. Then, all check nodes of type-j will be connected exactly twice or not at all. Thus, type-j check nodes satisfy the condition for $\mathcal{V}$ to be

a stopping set. Further, since the total number of edges connecting $\mathcal{V}$ to the type-j check nodes is $e_j = \sum_{i=1}^{M} n_i \nu_{i,j}$, the total number of type-j check nodes connected through this process is exactly $e_j/2$. The algorithm is now complete.

Step 2b: In this case, $s_k = c_k + 1$. This situation is similar to the $s_k = c_k$ scenario except that in this case, e_j is odd and thus, there is an extra edge that must be connected to a check node which is already connected to two variable nodes, i.e., there is one check node that will need to be connected to $\mathcal{V}$ via exactly three edges. We have two cases: (1) $s_k = n_k \nu_{k,j}$ and (2) $s_k > n_k \nu_{k,j}$.

Case 1: $s_k = n_k \nu_{k,j}$. Note that $k \neq 2$. If $k = 2$, then

$$n_2 \nu_{2,j} = s_2 = c_2 + 1 = n_1 \nu_{1,j} + 1 > n_1 \nu_{1,j} \tag{F.7}$$

which contradicts our assumption that $n_1 \nu_{1,j} \geq n_2 \nu_{2,j}$. Thus, $k > 2$. First, connect c_k of the n_k type-k variable nodes to the c_k singly-connected check nodes. Since $n_k = s_k$ and $s_k = c_k + 1$, this leaves a single type-k variable node left to be connected. There exists an $l < k$ with $l \geq 2$ such that $x_l \neq n_l \nu_{l,j}$ (see Step 2c). If this were not the case, then $x_i = n_i \nu_{i,j}$ and

$$c_{i+1} = c_i - n_i \nu_{i,j} + 2 n_i \nu_{i,j} = c_i + n_i \nu_{i,j} \tag{F.8}$$

for all $2 \leq i < k$. Thus,

$$c_k = c_2 + \sum_{i=2}^{k-1} n_i \nu_{i,j} = \sum_{i=1}^{k-1} n_i \nu_{i,j}. \tag{F.9}$$

Since, $n_k \nu_{k,j} = s_k = \sum_{i=k}^{M} n_i \nu_{i,j}$, then $\sum_{i=k+1}^{M} n_i \nu_{i,j} = 0$. Combining this

result with (F.9), we obtain

$$n_k\nu_{k,j} = s_k = c_k + 1 > c_k = \sum_{i=1}^{k-1} n_i\nu_{i,j} = \sum_{i=1}^{M} n_i\nu_{i,j} - n_k\nu_{k,j}. \qquad \text{(F.10)}$$

Thus, $2n_k\nu_{k,j} > \sum_{i=1}^{M} n_i\nu_{i,j}$ which contradicts (F.1). Thus, we can indeed find an l such that $x_l \neq n_l\nu_{l,j}$.

Since $x_l \neq n_l\nu_{l,j}$, some if not all of the type-l variable nodes were used to connect to singly-connected check nodes. Thus, there exists a check node which is connected to a type-l variable node and to a type-m variable node for some $m < l$. Connect the last type-k variable node to this check node. This results in a single check node with 3 connections: one connection to a type-k variable node, one to a type-l variable node for some $l < k$, and one to a type-m variable node for some $m < l$.

Case 2: $s_k > n_k\nu_{k,j}$. First, connect the $n_k\nu_{k,j}$ type-k variable nodes to $n_k\nu_{k,j}$ of the c_k singly-connected check nodes. Since $s_k > n_k\nu_{k,j}$, there exists an l such that $l > k$ and $n_l\nu_{l,j} > 0$. Connect *one* type-l variable node to a check node that was just connected to a type-k variable node. This results in a single check node with 3 connections: one connection to a type-l variable node, one to a type-k variable node, and one to a type-m variable node for some $m < k$. Now, connect the remaining $s_k - n_k - 1$ variable nodes to the remaining $c_k - n_k$ singly-connected check nodes, noting that $s_k - n_k - 1 = c_k - n_k$.

In both cases, all connected check nodes are now connected exactly twice except for one check node which is connected to three variable nodes. This results in a valid stopping set configuration. Further, since the total number of edges connecting $\mathcal{V}$ to the type-j check nodes is $e_j = \sum_{i=1}^{M} n_i\nu_{i,j}$, the total number

of type-j check nodes connected through this process is exactly $\lfloor e_j/2 \rfloor$. The algorithm is now complete.

Step 2c: In this case, $s_k > c_k + 1$. Let

$$x_k = \min \left\{ \left\lfloor \frac{s_k - c_k}{2} \right\rfloor, n_k \nu_{k,j} \right\}. \tag{F.11}$$

Connect $n_k \nu_{k,j} - x_k$ of the type-k variable nodes to $n_k \nu_{k,j} - x_k$ of the c_k singly-connected check nodes. Connect the remaining x_k type-k variable nodes to previously unconnected type-j check nodes. The resulting number of singly-connected check nodes is

$$c_{k+1} = c_k - (n_k \nu_{k,j} - x_k) + x_k = c_k - n_k \nu_{k,j} + 2x_k. \tag{F.12}$$

Step 3: Go to Step 2.

This algorithm always successfully completes at either Step 2a or Step 2b. We will prove this fact using proof by contradiction. Assume that the algorithm never enters Step 2a or Step 2b, i.e., the algorithm enters Step 2c at all iterations $k = 2, \ldots, M$. Then, there are only three possibilities to consider: (1) $M = 2$, (2) $M > 2$ and for all $k \in \{2, \ldots, M-1\}$, $x_k = n_k \nu_{k,j}$, and (3) $M > 2$ and for some $k \in \{2, \ldots, M-1\}$, $x_k = \lfloor (s_k - c_k)/2 \rfloor$. Note that $M \neq 1$ since in that case, (F.1) cannot be satisfied.

Case 1: $M = 2$. In this case, (F.1) can only be satisfied if $n_1 \nu_{1,j} = n_2 \nu_{2,j}$. Thus, $c_2 = s_2$ and the algorithm would enter Step 2a at iteration $k = 2$. This contradicts the original assumption that the the algorithm never enters Step 2a or Step 2b.

Case 2: $M > 2$ and for all $k \in \{2, \ldots, M-1\}$, $x_k = n_k \nu_{k,j}$. In this case, for each k,

$$\begin{aligned} c_{k+1} &= c_k - n_k \nu_{k,j} + 2 n_k \nu_{k,j} \\ &= c_k + n_k \nu_{k,j}. \end{aligned} \tag{F.13}$$

Thus,

$$c_M = c_2 + \sum_{k=2}^{M-1} n_k \nu_{k,j} = \sum_{k=1}^{M-1} n_k \nu_{k,j} > n_M \nu_{M,j} = s_M \tag{F.14}$$

where the strict inequality follows since $n_k \nu_{k,j} \geq n_M \nu_{M,j}$ for all $k < M$ and $M > 2$. This result contradicts the choice of x_M which guarantees that $c_M \leq s_M$ as shown in (F.6). Thus, this case is not possible.

Case 3: $M > 2$ and for some $k \in \{2, \ldots, M-1\}$, $x_k = \lfloor (s_k - c_k)/2 \rfloor$. For this k,

$$\begin{aligned} c_{k+1} &= c_k - n_k \nu_{k,j} + 2 \left\lfloor \frac{s_k - c_k}{2} \right\rfloor \\ &= \begin{cases} s_{k+1}, & s_k - c_k \text{ even} \\ s_{k+1} - 1, & s_k - c_k \text{ odd} \end{cases}. \end{aligned} \tag{F.15}$$

Thus, the algorithm will stop at either Step 2a or Step 2b at the next $(k+1)$th iteration. This contradicts the original assumption that the algorithm never enters Step 2a or Step 2b.

The proof by contradiction is now complete. This analysis has shown that when (F.1) is satisfied, the algorithm will always successfully complete at Step 2a or Step 2b. As shown in Step 2a and Step 2b above, the final result of the algorithm is a valid stopping set $\mathcal{V}$ connected to exactly $\lfloor e_j/2 \rfloor$ check nodes. Thus, w_j^* achieves its bound $w_j^* = \lfloor e_j/2 \rfloor$.

□

APPENDIX G

Proof of Precoding Theorem

This appendix provides a proof of Theorem 3.7, which is restated here:

Theorem 3.7. *Consider a protograph P whose type-i variable node is precoded to generate the protograph P'. Let V_P and $V_{P'}$ be the sets of stopping-set sizes which dominate the performance of the ensemble based on protograph P and P', respectively, and let $V_U = \{V_P \cup V_{P'}\}$. Assume that for all $v \in V_U$, stopping sets of size v follow category-P behavior for both protograph-based ensembles. Let Z be large enough such that Theorem 3.4 holds and $Z \geq 1/\epsilon$. Then,*

$$P_{e,P'}^{ub} \approx \sum_{v \in V_U} P_{e,P'}^{ub'}(v) \leq (Z+1) \sum_{v \in V_U} P_{e,P}^{ub'}(v) \approx Z \cdot P_{e,P}^{ub}. \tag{G.1}$$

Thus, the largest exponent of Z in the error-probability approximation for protograph-P' ensembles is at most one greater than the largest exponent for protograph-P ensembles, in the error-floor region.

Proof. Let the protograph P, with M variable-node types and J check-node types, be precoded to generate protograph P', with $M+1$ variable-node types and $J+1$ check-node types, as described in Section 3.8.

Consider a finite stopping set $\mathcal{V}'$, in the ensemble generated from the precoded

protograph P', consisting of v' variable nodes connected to $w^{*'}$ check nodes via e' edges. Let n_{punc} be the number of type-i variable nodes in $\mathcal{V}'$, i.e., the number of punctured nodes in the stopping set, and let n_{pre} be the number of type-$(M+1)$ variable nodes in $\mathcal{V}'$.

Now, consider the set $\mathcal{V}$ obtained from $\mathcal{V}'$ by removing all variables of type $(M+1)$, i.e.,

$$\mathcal{V} = \{x \in \mathcal{V}' : x \text{ is a variable node of type } i \neq (M+1)\}. \tag{G.2}$$

This set $\mathcal{V}$ must be a stopping set in the ensemble generated from protograph P. To obtain this result, observe that all check nodes of type $j = 1, \ldots, J$ must be connected to $\mathcal{V}$ at least twice or not at all in order for $\mathcal{V}'$ to be a stopping set.

The stopping-set characteristics of $\mathcal{V}'$ and $\mathcal{V}$ are related by the following expressions where primed and unprimed variables correspond to characteristics of $\mathcal{V}'$ and $\mathcal{V}$, respectively.

$$v' = v + n_{pre} \tag{G.3a}$$

$$v'_{eff} = v - n_{punc} + n_{pre} \tag{G.3b}$$

$$e' = e + 2n_{punc} + n_{pre} \tag{G.3c}$$

$$w^{*'} = w^* + n_{punc} + n_{pre}/2 \tag{G.3d}$$

To derive (G.3d), note that $w^{*'}$ is equal to $w^* + w_{J+1}$ where w_{J+1} is the maximum number of type-$(J+1)$ check nodes connected to the stopping set. Since there are $2n_{punc} + n_{pre}$ edges in the stopping set connected to type-$(J+1)$ check nodes, $w_{J+1} = (2n_{punc} + n_{pre})/2$. Combining the results in (G.3), we obtain

$$(v' - e' + w^{*'}) = (v - e + w^*) - (n_{punc} - n_{pre}/2). \tag{G.4}$$

Since V_U contains all the dominating stopping-set sizes,

$$P_{e,P}^{ub} \approx \sum_{v \in V_U} P_{e,P}^{ub'}(v) \quad \text{and} \quad P_{e,P'}^{ub} \approx \sum_{v \in V_U} P_{e,P'}^{ub'}(v). \tag{G.5}$$

Let n' be the codelength including the punctured nodes and let $\mathbf{n}' = (n_1, \ldots, n_{M+1})$ be the vector $\mathbf{n}$ concatenated with $n_{M+1} = n_{pre}$. Combining (3.68), (G.3), and (G.5) results in an error-probability upper bound

$$\begin{aligned}\sum_{v' \in V_U} P_{e,P'}^{ub'}(v') &= \sum_{v' \in V_U} \sum_{\mathbf{n}' \in \mathcal{S}_p'} \frac{\epsilon^{v - n_{punc} + n_{pre}} \times Z^{(v-e+w^*)-(n_{punc}-\frac{1}{2}n_{pre})}}{\prod_{i=1}^{M+1} (n_i!)^{1-d_{v,i}} \prod_{j=1}^{J+1} \prod_{h=1}^{|\beta|-1} m_{h,j}!} \\ &= \sum_{v' \in V_U} \sum_{n_{pre}=0}^{\min\{v', Z\}} \sum_{\mathbf{n}: \sum_{i=1}^{M} n_i = v} \left[\frac{(\epsilon Z)^{-(n_{punc}-\frac{1}{2}n_{pre})} \epsilon^{\frac{1}{2}n_{pre}}}{\prod_{h=1}^{|\beta|-1} m_{h,J+1}!} \frac{\epsilon^v Z^{v-e+w^*}}{\prod_{i=1}^{M} (n_i!)^{1-d_{v,i}} \prod_{j=1}^{J} \prod_{h=1}^{|\beta|-1} m_{h,j}!} \right]\end{aligned} \tag{G.6}$$

where $m_{h,j}$ is the value of m_h corresponding to the jth check-node type and the last equality follows since type-$(M+1)$ variable nodes have degree 1. In order for type-$(J+1)$ check nodes to be connected to $\mathcal{V}'$ at least twice or not at all,

$$(n_{punc} - \frac{1}{2} n_{pre}) \geq 0. \tag{G.7}$$

Thus, assuming that $Z \geq 1/\epsilon$, then $(\epsilon Z)^{-(n_{punc}-\frac{1}{2}n_{pre})} \leq 1$ and hence, the weighting term in (G.6) is bounded by

$$\frac{(\epsilon Z)^{-(n_{punc}-\frac{1}{2}n_{pre})} \epsilon^{\frac{1}{2}n_{pre}}}{\prod_{h=1}^{|\beta|-1} m_{h,(J+1)}!} \leq 1. \tag{G.8}$$

Thus,

$$\sum_{v' \in V_U} P_{e,P'}^{ub'}(v') \leq \sum_{v' \in V_U} \sum_{n_{pre}=0}^{\min(v',Z)} P_{e,P}^{ub'}(v' - n_{pre})$$

$$= \sum_{v' \in V_U} \sum_{v=v'-\min(v',Z)}^{v'} P_{e,P}^{ub'}(v). \tag{G.9}$$

Assume that $V_U = \{1, \ldots, v_{max}\}$ where $v_{max} = \max_{v \in V_U} v$. If this is not the case, then one can simply expand V_U to include all values between 1 and v_{max}. The resulting expanded V_U still contains all dominating stopping-set sizes, so the approximations in (G.5) are still valid. Applying this assumption,

$$\sum_{v' \in V_U} P_{e,P'}^{ub'}(v') \leq \sum_{v'=1}^{v_{max}} \sum_{v=\max(0,v'-Z)}^{v'} P_{e,P}^{ub'}(v) \leq \sum_{v=1}^{v_{max}} \sum_{v'=v}^{v+Z} P_{e,P}^{ub'}(v)$$

$$= (Z+1) \sum_{v \in V_U} P_{e,P}^{ub'}(v). \tag{G.10}$$

□

BIBLIOGRAPHY

BIBLIOGRAPHY

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

[2] D. Divsalar, "Ensemble weight enumerators for protograph LDPC codes," in *Proc. International Symposium on Information Theory*, Seattle, WA, July 2006.

[3] D. Divsalar, S. Dolinar, and C. Jones, "Low-rate LDPC codes with simple protograph structure," in *Proc. International Symposium on Information Theory*, Adelaide, Australia, 2005.

[4] J. Hou, P. H. Siegel, and L. B. Milstein, "Performance analysis and code optimization of low density parity-check codes on Rayleigh fading channels," *IEEE J. Select. Areas Commun.*, vol. 19, pp. 924–934, May 2001.

[5] C. E. Shannon, "A mathematical theory of communication," *Bell System Tech. J.*, vol. 27, pp. 379–423, July 1948.

[6] ——, "A mathematical theory of communication," *Bell System Tech. J.*, vol. 27, pp. 623–656, Oct. 1948.

[7] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. International Conf. Communications*, Geneva, Switzerland, May 1993, pp. 1064–1070.

[8] P. Oswald and A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Information Theory*, vol. 48, no. 12, pp. 3017–3028, Dec. 2002.

[9] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. L. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

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

[11] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.

[12] A. P. Worthen and W. E. Stark, "Low-density parity check codes for fading channels with memory," in *Proc. Allerton Conf. Commun., Control, Comp.*, Allerton House, IL, Sept. 1998, pp. 117–125.

[13] A. P. Worthen, "Codes and iterative receivers for wireless communication systems," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 2001.

[14] A. Worthen and W. Stark, "Interference mitigation in frequency-hopped spread-spectrum systems," in *Proc. IEEE Intn'l. Conf. Spread Spectrum Tech. and Applications (ISSSTA)*, Parsippany, NJ, Sept. 2000, pp. 58–62.

[15] W. Vijacksungsithi and K. A. Winick, "Joint channel-state estimation and decoding of low-density parity-check codes on the two-state noiseless/useless BSC block interference channel," vol. 53, no. 4, pp. 612–622, Apr. 2005.

[16] R. G. Gallager, "Low-density parity-check codes," *IEEE Trans. Information Theory*, vol. 8, pp. 21–28, Jan. 1962.

[17] R. J. McEliece, "The effectiveness of turbolike codes on nonstandard channel models," in *Proc. International Symposium on Information Theory*, Washington, D.C., June 2001, plenary talk.

[18] D. Divsalar, S. Dolinar, and C. Jones, "Construction of protograph LDPC codes with linear minimum distance," in *Proc. International Symposium on Information Theory*, Seattle, WA, July 2006.

[19] S. L. Fogal, R. McEliece, and J. Thorpe, "Enumerators for protograph ensembles of LDPC codes," in *Proc. International Symposium on Information Theory*, Adelaide, Australia, Sept. 2005.

[20] A. Orlitsky, K. Viswanathan, and J. Zhang, "Stopping set distribution of LDPC code ensembles," *IEEE Trans. Information Theory*, vol. 51, no. 3, pp. 929–953, Mar. 2005.

[21] B. J. Frey, *Graphical Models for Machine Learning and Digital Communications*. Cambridge, MA: MIT Press, 1998.

[22] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University, Linköping, Sweden, 1996.

[23] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[24] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[25] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "Comparison of constructions of irregular Gallager codes," *IEEE Trans. Communications*, vol. 47, no. 10, pp. 1449–1454, Oct. 1999.

[26] T. Tian, C. Jones, J. Villasenor, and R. D. Wesel, "Construction of irregular LDPC codes with low error floors," in *Proc. International Conf. Communications*, Anchorage, Alaska, 2003, pp. 3125–3129.

[27] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *European Trans. Telecommun.*, vol. 9, no. 2, pp. 155–172, March/April 1998.

[28] C. Di, "Asymptotic and finite-length analysis of low-density parity-check codes," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2004.

[29] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2007, to be published.

[30] T. Richardson, A. Shokrollahi, and R. Urbanke, "Finite-length analysis of various low-density parity-check ensembles for the binary erasure channel," in *Proc. International Symposium on Information Theory*, Lausanne, Switzerland, June 2002.

[31] J. Thorpe, "Low-density parity-check (LDPC) codes constructed from protographs," Jet Propulsion Laboratory, Pasadena, CA, Interplanetary Network Progress Report 42-154, Aug. 2003.

[32] T. Richardson and V. Novichkov, "Methods and apparatus for decoding LDPC codes," U.S. Patent 6,633,856 B2, Oct. 14, 2003.

[33] K. Fu and A. Anastasopoulos, "Performance analysis of LDPC codes for time-selective complex fading channels," in *Proc. Globecom Conf.*, Taipei, Taiwan, Nov. 2002, pp. 1279–1283.

[34] ——, "Analysis and design of LDPC codes for time-selective complex-fading channels," *IEEE Trans. Wireless Communications*, vol. 4, pp. 1175–1185, May 2005.

[35] T. Richardson, "Error floors of ldpc codes," in *Proc. Allerton Conf. Commun., Control, Comp.*, Allerton House, Illinois, 2003, pp. 1426–1435.

[36] K. Fu and A. Anastasopoulos, "Stopping-set enumerator approximations for finite-length protograph LDPC codes," in *Proc. International Symposium on Information Theory*, Nice, France, June 2007.

[37] H. Jin and T. J. Richardson, "Design of low-density parity-check codes for noncoherent MPSK communication," in *Proc. International Symposium on Information Theory*, Lausanne, Switzerland, June 2002, p. 169.

[38] R. Nuriyev and A. Anastasopoulos, "Pilot-symbol-assisted coded transmission over the block-noncoherent AWGN channel," *IEEE Trans. Communications*, vol. 51, no. 6, pp. 953–963, June 2003.

[39] T. L. Marzetta and B. Hochwald, "Capacity of a mobile multiple-antenna communication link in Rayleigh flat fading," *IEEE Trans. Information Theory*, vol. 45, pp. 139–157, Jan. 1999.

[40] B. Hassibi and B. Hochwald, "How much training is needed in multiple-antenna wireless links?" *IEEE Trans. Information Theory*, vol. 49, no. 4, pp. 951–963, Apr. 2003.

[41] L. Zheng and D. N. C. Tse, "Communication on the Grassmann manifold: a geometric approach to the noncoherent multiple-antenna channel," *IEEE Trans. Information Theory*, vol. 48, no. 2, pp. 359–383, Feb. 2002.

[42] A. Anastasopoulos and K. M. Chugg, "Adaptive soft-input soft-output algorithms for iterative detection with parametric uncertainty," *IEEE Trans. Communications*, vol. 48, no. 10, pp. 1638–1649, Oct. 2000.

[43] M. C. Valenti and B. D. Woerner, "Iterative channel estimation and decoding of pilot symbol assisted turbo codes over flat-fading channels," *IEEE J. Select. Areas Commun.*, vol. 19, no. 9, pp. 1697–1705, Sept. 2001.

[44] R. Storn and K. Price, "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–3–59, 1997.

[45] A. P. Worthen and W. E. Stark, "Unified design of iterative receivers using factor graphs," *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 843–849, Feb. 2001.

[46] J. G. Proakis, *Digital Communications*, 4th ed. New York: McGraw-Hill, 2001.

[47] I. C. A. Faycal, M. D. Trott, and S. Shamai (Shitz), "The capacity of discrete-time Rayleigh fading channels," in *Proc. International Symposium on Information Theory*, Ulm, Germany, June 1997, p. 473.

[48] R.-R. Chen, R. Koetter, U. Madhow, and D. Agrawal, "Joint noncoherent demodulation and decoding for the block fading channel: A practical framework for approaching Shannon capacity," *IEEE Trans. Communications*, vol. 51, no. 10, pp. 1676–1689, Oct. 2003.

[49] D. Burshtein and G. Miller, "Asymptotic enumeration methods for analyzing LDPC codes," *IEEE Trans. Information Theory*, vol. 50, no. 6, pp. 1115–1131, June 2004.

[50] D. Divsalar, S. Dolinar, and C. Jones, "Protograph LDPC codes over burst erasure channels," in *Proc. IEEE Military Comm. Conf.*, Washington, D.C., Oct. 2006.

[51] C. Di, R. Urbanke, and T. Richardson, "Weight distributions: How deviant can you be?" in *Proc. International Symposium on Information Theory*, Washington, D.C., June 2001.

[52] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 569–584, Feb. 2001.

[53] A. Shokrollahi and R. Storn, "Design of efficient erasure codes with differential evolution," in *Proc. International Symposium on Information Theory*, Sorrento, Italy, June 2000, p. 5.

[54] A. Shokrollahi, "Capacity-achieving sequences," in *Codes, Systems, and Graphical Models*, ser. IMA Volumes in Mathematics and its Applications, B. Marcus and J. Rosenthal, Eds. Springer-Verlag, 2000, vol. 123, pp. 153–166.

[55] D. Divsalar and C. Jones, "Protograph LDPC codes with node degrees at least 3," in *Proc. Globecom Conf.*, San Francisco, CA, Nov. 2006.

[56] C. H. Hsu, "Design and analysis of capacity-achieving codes and optimal receivers with low complexity," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 2006.

[57] O. Milenkovic, E. Soljanin, and P. Whiting, "Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles," *IEEE Trans. Information Theory*, vol. 53, no. 1, pp. 39–55, Jan. 2007.